INTERNAL PRE-MEETING DEEP DIVE

Ring RS3000 — Every New Feature, From First Principles

What we built, whether it actually works, and exactly what to say when you are grilled on it. 87 features, independently verified against the source code.

Ringwood Starch-Glue Mixing Plant · Ring RS3000 vs legacy RS-360 · for internal preparation

Contents

Chapter 2 — Running a Batch Without the Guesswork

- Keep your plant's memory — and a roadmap of upgrades built on top of it
- Confirm exactly what gets sent before the plant arms
- A live, honest checklist of where the batch stands
- A searchable, permanent record of every batch
- Edit a recipe once — screen, controller, and audit trail stay in sync
- Spot a drifting batch against your own good runs
- Pin operator notes to the exact batch
- Re-run the usual recipe by the name operators use
- Split a batch to the right second tank, safely

Chapter 3 — Turn Plant Memory Into Money

- See your spend and stop the leaks before they cost you
- Make faster calls with decision screens the legacy never had
- Watch trends live and review any tank's last 30 days
- Catch a failing pump before it scraps a batch
- Get every report you rely on, from one database, with PDF
- Answer show me our data in one click — you own it
- Never trust a frozen number: live, stale, or off-target at a glance
- Keep a true history — automatically, with zero extra load on the controller

Chapter 4 — Smarter Alarms, Faster Fixes

- Every alarm reads exactly like the controller says it
- One tap from the alarm to the screen that fixes it
- A built-in first thing to check card for priority alarms
- Tell a live critical from a handled warning at a glance
- An ON HOLD flag on the dashboard the moment a batch stalls
- See how long a fault has been sitting unhandled
- A full-screen alarm pop-up that won't let a live fault be ignored
- An are you sure? guard before Hold, Resume, or Reset
- Tie downtime to the exact batch that was running
- Honest coverage status: 76 alarms correct today, 94 legend rows tracked
- On a screen left running for days, a live fault never quietly vanishes

Chapter 5 — On the Floor Standing Up a New Site

- Stand up a whole new station in a few guided minutes
- See your own equipment names on day one — imported from RS-360
- Never miss a comms loss or a live alarm — safety is visible on every screen
- Cut missed-context errors between crews with durable shift handoff
- See every tank group live, by the plant's own names
- Rename, reconfigure, and back up yourself — no developer, no callout fee
- Fix faster and stay informed off-site — alarm emails with fix steps, scheduled reports
- Turn the HMI into an owner's management tool — cost, alarm guidance, maintenance
- Operators read the HMI in their own language and script
- End the where am I / how do I get back confusion
- Root-cause faster and protect your data
- Get hand-logged readings off paper and into reports
- Set agitators to cycle themselves instead of managing paddles by hand
- Keep operators productive through the switch — manual a click away, F1 cheat-sheet
- A home screen shaped by what operators actually glance at

Chapter 6 — Own Your Data, on a Platform Built to Last

- Own all your plant data in one open file, with no vendor in the loop
- Self-service backups, a checked restore, and one-click Excel exports
- Keep the batch you just finished through a power blip — and never freeze under load

Part 0 — The Whole Thing From Zero (read this first)

This part assumes you know **nothing** about the plant, the software, or the technology. By the end of it you'll understand what the plant makes, how it makes it, what the old system was, what Ring is, and — most importantly — the handful of sentences you must phrase carefully so a knowledgeable person in the room can't poke a hole in them. Everything after Part 0 explains the individual features; this part is the ground you stand on while doing it.

0.1 The whole story in one breath

The plant mixes industrial **glue**. It has run for ~20 years on a 2002-era operator screen called **RS-360** that does the job but is blind and forgetful — it keeps no usable history, knows nothing about cost, shows cryptic alarm codes, and is spread across ~97 screens. We built **Ring**, a modern operator screen that runs on the **same controller** (we did not touch the machine that physically runs the plant) and closes those gaps: one command-center screen, a permanent searchable history, an owner cost view, plain-language alarms, and 13 operator languages. It already ran **live on the real plant controller** in a read-only mode and read every value correctly. That's the whole pitch; the rest is detail.

0.2 What the plant actually makes

The plant is a **corrugating-adhesive kitchen** — it makes the **starch-based glue** that holds corrugated cardboard together. (Corrugated board is the wavy-middle, flat-outside material that boxes are made of; the wavy layer is glued to the flat layers.) The glue is mixed from a few raw ingredients — **starch, water, caustic (sodium hydroxide), and borax** — in heated, stirred tanks, following a **recipe**. It is *not* food; nobody eats it. That matters because it changes which concerns are real: there is no food-safety or recall regime to satisfy, but the **consistency of the glue (its thickness, or "viscosity") is the critical quality spec** — too thin or too thick and the cardboard plant downstream has problems.

One honest technical note worth knowing before someone asks: **the controller does not have a viscosity sensor wired in.** So Ring cannot show a live viscosity number — it shows temperature, tank levels, and ingredient amounts, which are the things the controller actually measures. If anyone asks "does it read viscosity?", the answer is "the controller doesn't expose a viscometer signal, so neither system can; that was scoped out deliberately."

0.3 The physical process, from zero

A few plain-language terms you'll use all meeting:

- **Tank** — a vessel that holds liquid. This plant has **13 in service**: one **mix tank** (where glue is made), six **storage tanks** (where finished glue waits to be used), and six **doser/feed tanks** (that meter ingredients in). The old system and an early version of Ring only showed a subset; the real plant runs all 13.
- **Ingredient** — a raw material added to the mix (starch, water, caustic, borax, steam for heat, etc.). Each ingredient addition is one **step** in the recipe.
- **Recipe / formula** — the ordered list of steps that makes one product: "add this much water, heat to this temperature, add this much starch, stir this long," and so on. The plant mostly runs default-named formulas;

only one ("SF 1") is custom-named.

- **Batch** — **one production run of one recipe.** It is the plant's core unit of work. A batch has a start, a sequence of steps, and an end, and it produces a tank of glue. The plant has made roughly **1,404 batches** of recorded history.
- **Step / operation** — one instruction inside a recipe, identified inside the controller by a numeric **op-code** (e.g. a code that means "add starch" or "turn the agitator on"). The old system showed operators these raw numbers; Ring translates them into words.
- **Agitator** — the motorized paddle that stirs a tank. Turning it on/off is itself a recipe step the controller sequences.

So a normal shift is: pick a recipe, start a batch, the controller runs the steps (open valves, dose ingredients, heat, stir) while the operator watches, and at the end you have a tank of glue and a record of what happened. Ring is the **window** the operator watches and controls that through — it is **not** the thing opening the valves.

0.4 The single most important distinction: the controller (PLC) vs. the operator screen (HMI)

If you remember one thing, remember this, because half the tough questions hinge on it:

- The **PLC (Programmable Logic Controller)** is the rugged industrial computer bolted in the plant that **physically runs everything** — it opens valves, starts pumps, drives the mixer, and enforces safety interlocks. Here it's an **Allen-Bradley CompactLogix (model 1769-L36ERM)**. It runs its own control program. **Ring does not replace it, change it, or reprogram it.**
- The **HMI (Human-Machine Interface)** is the **screen software** an operator uses to *see* what the PLC is doing and *send it requests* ("start this batch," "put this on hold"). **RS-360 is the old HMI. Ring is the new HMI.** Replacing the HMI is like replacing a car's dashboard, gauges, and logbook while keeping the engine, transmission, and brakes exactly as they are.

The analogy to use out loud: **"We kept the engine and replaced the dashboard."** The engine (PLC) is proven and untouched; the dashboard (HMI) is what we modernized.

Why this matters for the meeting: every time someone worries "is this risky to the running plant?", the honest answer traces back to here — Ring is a screen on top of an unchanged controller, and during the live trial it was running in a **read-only mode that suppresses every command to the controller** (more on that in 0.7).

0.5 The old system (RS-360) and its four real gaps

RS-360 is a **DOS-era application written in 2002 in Borland C++** (an obsolete toolset). It is not broken — it still runs the plant today — but it has four gaps that quietly cost money and time every shift. Ring exists to close all four:

1. **It forgets.** It has no usable, surviving on-screen memory — no continuous trend charts, no record of how long each step took, no shift summaries, no equipment-runtime history, no "best batch to compare against." Whatever scrolled off the screen was gone. *(Careful phrasing — see 0.8: the legacy did keep some history in database files; what it lacked was a* usable, *living historian and trends. Don't say "it records nothing.")*

2. **It can't see cost.** It never computed or kept a single number about money — not what a batch cost, not where ingredients were over-dosed (paid-for product given away), not what a scrapped batch or an hour of hold time was worth.
3. **Its alarms are cryptic.** When something went wrong, operators often saw a bare numeric code (in some cases a placeholder with no description at all) and had to decode it from a printed legend, mid-fault, under pressure.
4. **It's hard to run.** The plant is spread across ~97 separate screens, and the operator has to already know the map — which screen fixes which problem and how to get there. A new hire could take months to get fluent, so the plant leaned on a few veterans.

Underneath all four sits a fifth, structural problem: RS-360 runs on **dead 2002 tools no one can support or hire for**, and its data lives in an **abandoned database format (Paradox/BDE)** with no maintained driver. That's the "why now" — the risk grows every year the platform ages.

0.6 What Ring is (and what it deliberately doesn't change)

Ring is a modern Windows control-room application (built on current Microsoft .NET/WPF technology) that gives operators **one screen to run the whole plant**, keeps a **permanent, searchable, exportable record** of everything it makes, and turns plant data into **decisions and dollars** — all on the **same untouched Allen-Bradley controller.**

What Ring **does not** change, on purpose:

- It does not replace or reprogram the PLC.
- It does not change the physics, the recipes the controller runs, or the safety interlocks.
- During the live trial it did not send the controller a single command — it only read.

That "we changed the screen, not the controls" boundary is not a limitation to apologize for; it's the **reason the upgrade is low-risk.** You modernize the part operators touch, while the proven control logic keeps running.

0.7 How Ring talks to the controller — and the "read-only safety gate"

Two facts here defuse most safety questions:

- **Modern, open connection.** Ring talks to the controller over **EtherNet/IP** — a standard, open industrial protocol — using an open library, asking for values by name. The old system reached the controller through bespoke Borland helper components that have to be kept alive. Ring's path is the supportable, modern one, and it ships with on-screen diagnostics so the floor can see the connection's health without specialist tools.
- **Read-only cutover gate.** Ring ships with a setting (`ReadOnlyMode = true`) that **suppresses every write to the controller.** In that mode Ring can *watch* the live plant for as long as you like — days, weeks — and prove itself next to the old system **without ever being able to change the process.** Writes are turned on later, deliberately, only when you choose to go live. This is the single most reassuring thing you can say about risk.

When we say it "ran live on the real plant," we mean: plugged into the actual production controller, in read-only mode, and it **read all 133 of the controller's live values correctly** — proof it works on your real hardware, not just a demo. (Phrase this as "without issuing a single command that could change the process," not "without sending a single byte" — see 0.8.)

0.8 The five claims you must phrase carefully (this is your armor)

These are places where the short marketing wording is technically vulnerable. A controls engineer who knows the legacy could challenge them. Use the **right-hand "say this" version** and you're bulletproof; these are corrected against the plant's own data and code.

#	The risky wording	Why it's vulnerable	Say this instead
1	"The old system **records nothing** durable."	The legacy *did* keep history in database files — batch headers (~~1,404), step rows~~ (16,776), shift and alarm records. A buyer can open those files.	"The legacy kept records **locked in a dead, corruption-prone format** with **no living historian, no trends, no cost, no best-batch comparison, and no step-timing timeline.** Ring keeps the same history **usable, searchable, and exportable.**"
2	"Read the plant **without sending a single byte.**"	Reading values over the network *does* send request packets; an engineer could see them on a network trace.	"Read all 133 values live **without issuing a single command that could change the process** — every write is suppressed by the read-only gate."
3	"**0 bytes** to the controller."	Same as #2 — it's about commands, not bytes.	"**Zero write commands** sent under the read-only safety gate."
4	Legacy "**split batching wrote ambiguous indices.**"	That ambiguous-index issue was a **Ring** bug we found and fixed — not legacy behavior. The legacy actually supported splitting (the plant just never used it).	Don't attribute it to legacy. Say: "We found and fixed an internal split-batch addressing bug during our own review; it's pending a controls-engineer sign-off before live split batching."
5	"**~70–80%** parity" / "verified **on-site**."	The parity number is **~80%** from a **source-code audit (June 18)**; the *on-site* event was the separate live-PLC read. Mixing them invites "which is it?"	"**~80%** feature parity-or-better, per an engineering **code audit**; separately, we verified **133/133 live reads on-site** on the real controller."

Two more numbers to keep honest:

- **Tests:** say **"2,400+ automated tests, zero failures, on a zero-warning build"** rather than quoting an exact count like 2,416 — the number moves with every build, and a stale exact figure looks wrong.
- **"87 features":** that's the count of distinct capabilities in our inventory across five themes. It's a fair headline; if pressed, it breaks down into the chapters that follow, and a portion are honestly labeled *in-progress* or *roadmap* (we never hide that).

0.9 How the rest of this document works

The remaining chapters walk through **every one of the 87 capabilities**, grouped into the themes below. For **each feature** you get the same five-part treatment, always in plain language:

1. **What it is** — explained as if you've never seen the screen.
2. **Why it matters** — the real problem on the floor it removes.
3. **How it works** — the mechanism, in plain terms (no jargon left undefined).
4. **Versus the old RS-360 system** — the honest before/after.
5. **Status & honest caveats** — is it live, in-progress, or roadmap; is it on a real screen or built-but-not-yet-wired; and anything you must be ready to defend.

…and then the part you specifically asked for:

6. **Likely questions & how to answer them** — the pointed questions a skeptical owner, controls engineer, or operator could ask, each with a crisp answer you can say out loud.

The five themes (and the chapters that carry them):

- **Make or save money** — the owner cost view, waste/giveaway, best-batch comparison, predictive drift. *(Chapter: Turn Plant Memory Into Money.)*
- **Stay ahead of problems** — live trends, tank time-to-empty, run-dry and equipment-drift warnings. *(Chapters: Insight + Command Center.)*
- **Easier for operators** — one command-center screen, guided batch start, the floor & onboarding tools, languages. *(Chapters: Command Center, Run a Batch, On the Floor.)*
- **Smarter alarms** — plain-language alarms, severity at a glance, first-response playbook, alarm-to-batch links. *(Chapter: Faster, Safer Fault Triage.)*
- **Own your data** — a modern open database, one-click export, the automatic historian, reliable connection, and the honest "why now." *(Chapters: Own Your Data, Reliability, Why Now, Where the Work Stands.)*

A glossary and a master "hardest questions" appendix sit at the very end. You can read cover to cover, or jump to a chapter from the contents.

> Bottom line for Part 0: Ring rebuilds a blind, forgetful 2002 operator screen into a modern command center that sees the whole plant, remembers everything it makes, and turns that into dollars — **on the same trusted controller, proven by a live read-only trial.** Keep the engine/dashboard framing, use the "say this instead" wordings in 0.8, and the rest of the meeting is just walking through the features.

Feature Status Scorecard — All 87 at a Glance

14 of 87 carry a "phrase carefully" flag; the other 73 are solid to state plainly.

Every feature, with the status confirmed by reading the source. ⚠ = **phrase carefully** (there is a per-feature "Watch-out" in its chapter). Everything else is solid to state plainly.

Chapter 1 — Run a Whole Shift From One Screen

Feature	Verified status	
Walk up and know the whole plant in one glance	Live	✓
Spot trouble from across the control room in one glance	Live	✓
Catch a stalled step before the controller alarm does	Live	✓
Never act on a frozen reading that looks live	Live	✓
One tap from the number to the screen that fixes it	Live	✓
No fake '100% done' for the last 20 minutes of a batch	Live	✓
Step away and still see what just finished	Live	✓
Catch a developing tank problem the PLC can't even alarm on	Live	✓
Make production decisions on real history, not invented numbers	Live	✓
Let the floor self-troubleshoot without opening Studio 5000	Live	✓
End false 'PLC down' calls — know an outage from an idle controller	Live	✓

Chapter 2 — Running a Batch Without the Guesswork

Feature	Verified status	
Stop losing 20 years of plant knowledge to a system that forgets on restart	Roadmap	⚠
Prevent a wrong batch before it arms the plant — confirm in plain language	Live	✓
See exactly where the batch is — progress that never lies through a network hiccup	Live	✓
Keep a searchable record of every batch — even one interrupted by a reboot	Live	✓
Edit a recipe once — the database, the controller, and the audit trail stay in sync	Live	✓
Catch a drifting batch early — overlaid against your own proven good runs	Live	✓
Tie tribal knowledge to the exact batch — for the next quality investigation	Live	✓
Re-run the usual recipe in one tap — by the name operators actually call it	In progress	✓
Split a batch to the right second tank — with overfill protection that follows it	In progress	✓

Chapter 3 — Turn Plant Memory Into Money

Feature	Verified status	
See your spend and stop the leaks before they cost you	Live	✓
Make faster calls with decision screens the legacy never had	Live	✓
Watch trends live and review any tank's last 30 days	Live	✓
Catch a failing pump before it scraps a batch	Live	⚠
Get every report you rely on, from one database, with PDF	Live	✓
Answer 'show me our data' in one click — you own it	Live	⚠
Never trust a frozen number: live, stale, or off-target at a glance	Live	✓
Keep a true history — automatically, with zero extra load on the controller	Live	✓

Chapter 4 — Smarter Alarms, Faster Fixes

Feature	Verified status	
No translation guesswork mid-fault - alarms read exactly like the controller	Live	✓
From alarm fires to fixing it in one tap - even for a new operator	Live	✓
A green operator gets expert first-response guidance the instant a covered alarm fires	Live	✓
Triage a live critical from a resolved warning across the room - color-blind safe	Live	✓
Catch a stalled batch before it costs you - BATCH ON HOLD shown right on the hero	Live	✓
Spot the fault that's been sitting too long - live age plus an oldest active callout	Live	✓
Reserve operator attention for real criticals - E-STOP front and center, calm for warnings	Live (partial)	⚠
An accidental tap cannot disrupt a live batch - confirm before every PLC write	Live	✓
Pin downtime to the exact batch and recipe that was running	Live	✓
Honest status: today's alarms are correct; 94 legacy legend rows are a tracked seed task	In progress	✓
On a 24/7 plant box, a live fault never silently disappears	Live	✓

Chapter 5 — On the Floor & Standing Up a New Site

Feature	Verified status	
Stand up a new station in minutes - no developer, no INI archaeology	Live	✓
See your own equipment names on day one - imported from RS-360 in any language	Live	✓
Never miss a comms loss or a live alarm - safety is visible on every screen	Live	✓
Cut missed-context errors between crews with durable shift handoff	Live	✓
See every tank group live, by the plant's own names	Live (partial)	⚠
Rename, reconfigure, and back up yourself - no developer, no callout fee	Live	✓
Fix faster and stay informed off-site - alarm emails with the fix steps, scheduled reports	Live	✓
Turn the HMI into an owner's management tool - cost, alarm guidance, maintenance	Live	✓
Operators read the HMI in their own language and script from day one	In progress	✓
End the 'where am I / how do I get back' confusion of a deep menu tree	Live	✓
Root-cause faster and protect your data - tag inspector, snapshot, one-button backup	Live	✓
Get hand-logged readings off paper and into reports - with your own field labels	Live (partial)	⚠
Set agitators to cycle themselves instead of managing paddles by hand	In progress	⚠
Keep operators productive through the switch - manual a click away, F1 cheat-sheet	Live	✓
A home screen shaped by what operators actually glance at every hour	In progress	⚠

Chapter 6 — Own Your Data, Platform Built to Last

Feature	Verified status	
Hand an auditor every record on day one, no vendor required	Live	✓
Self-service backups, a verified restore, and Excel exports on demand	Live	✓
Keep the batch you just finished through a power blip; never freeze under load	Live (under the hood)	✓
Upgrade without a DBA, and an interrupted upgrade never corrupts your history	Live (under the hood)	✓
Inventory totals you can trust, even after a controller reset	Live (under the hood)	✓
Your tank, ingredient, and recipe names (and language) stay put forever	Live	✓
A glance-from-15-feet look that re-skins as a config swap, not a rewrite	Live	✓
Your fixes and features arrive sooner, with a contained blast radius	Live (under the hood)	✓
A screen that never freezes, and data that holds up to a security audit	Live (under the hood)	✓
Faster, cheaper support and months of unattended 24/7 uptime	Live (under the hood)	✓
No second copy fighting for the PLC, and clear guidance when one is frozen	Live (under the hood)	✓
Any .NET contractor can build it, and every release is checksum-verifiable	In progress	✓

Chapter 7 — A Reliable, Honest Connection

Feature	Verified status	
Stop chasing false 'PLC is down' calls - Ring knows unplugged vs frozen vs running	Live	⚠
A command never lands in a frozen controller - writes fire only into a provably-running PLC	Live (under the hood)	✓
Operators never decide on a frozen number that looks live	Live	✓
No more morning HMI restarts - it reconnects on its own after a power blip	Live	⚠
A frozen screen can't lie to your operators - it's caught and recorded	In progress	✓
See exactly what flows to the PLC - and never depend on a 2002 Borland driver again	Live	✓
No error storms, no network overload - it rides out an outage and snaps back	Live (under the hood)	✓

Chapter 8 — Why Replace It Now

Feature	Verified status	
Buy out your single point of failure before it forces your hand	Live (under the hood)	✓
Own your recipes, batch history, and alarms — instead of holding them hostage in a dead engine	Live	✓
Make changes any .NET contractor can do — not a 2002 build wired to one retiring PC	Live (under the hood)	✓
Buy bounded, shrinking risk — and stop paying for unbounded, growing risk	Live (under the hood)	✓
Verify every claim against the code before you sign	Live (under the hood)	✓

Chapter 9 — Where the Work Honestly Stands

Feature	Verified status	
You go live on evidence, not optimism: an audited GO-WITH-CONDITIONS with every condition named	In progress	✓
Your engineer can audit the whole decision: a dated NO-GO to GO-WITH-CONDITIONS to live read-only GO trail	Live (under the hood)	✓
Your operators keep what they know and gain what the old system never had: ~80% audited parity, gaps named	In progress	⚠
You hear it from us first: no per-operator named-user audit trail yet, scoped to your regulatory class	In progress	⚠
You troubleshoot on the floor without Studio 5000: self-service diagnostics and one-button DB backup	Live	✓
Your operators read the screen in their language today, switchable live: 13-language engine shipped	In progress	⚠
Nothing here blocks daily work: the gated-but-built follow-ups, listed plainly	In progress	⚠
You get a precise scoping conversation up front: the questions only your team can close	Roadmap	✓
Your floor adopts the new interface now while the control-logic port is scheduled separately	Roadmap	✓

Chapter 1 — Run a Whole Shift From One Screen

Before any of the features make sense, here is the plant in one breath. A **PLC** (Programmable Logic Controller) is the rugged industrial computer that actually runs the glue kitchen — it opens valves, starts pumps, and drives the mixer. The PLC does the work; nobody watches a PLC directly. To see what it is doing, an operator needs a screen — an **HMI** (Human-Machine Interface). The old screen was **RS-360**, a 2002 DOS-era program written in Borland C++. **Ring is the new HMI.** It is a modern Windows application that talks to the *exact same* Allen-Bradley CompactLogix PLC — Ring replaces the screen, not the controller. The recipes, valves, and safety logic in the PLC are untouched.

One more word you'll hear: a **batch** is a single production run of one glue recipe — fill a tank, dose in starch and the other ingredients step by step, mix, and finish. Each batch is made of numbered **steps**. A **tag** is just a named value living inside the PLC (for example, the mix-tank temperature). A **KPI** is a headline number worth watching ("Key Performance Indicator").

The floor problem this chapter solves: on the old system, an operator who walked up to the screen learned almost nothing at a glance. RS-360 opened to a splash screen and was organized one screen per area of the plant — one form for this storage tank, another for that mixer, another for alarms. To answer the simple question "is the plant healthy, and what's running?" the operator had to click through several screens and assemble the picture in their head. Ring opens to **one dashboard** that answers that question before anyone touches anything.

Two honesty points to carry into the meeting. First, on safety: Ring shipped to the live plant with a **read-only gate** turned on, which suppresses every command that could change the process. On-site it read all 133 live tags it needed and never issued a single command that could move a valve. Reading does put request packets on the network (it has to ask the PLC for each value), so the precise, defensible phrase is **"zero write commands,"** not "zero bytes." Second, on the old system: RS-360 was *not* a system that recorded nothing — it did keep batch, step, shift, and alarm history. The catch is that it stored them in a dead Paradox/BDE database format with no living, searchable historian, no trends, no step-durations, and no cost. So the honest contrast is "the old data was locked away and unusable," never "the old system stored nothing."

Walk up and know the whole plant in one glance

WHAT IT IS

When the operator opens Ring, the entire shift picture is already on the screen: the batch that's running, an overall plant-health verdict, the four numbers that drive the next decision, any active alarms, and the tanks. Nothing has to be clicked together — it's all on one scrolling page often called a "single pane of glass."

WHY IT MATTERS

A shift starts with *situational awareness* instead of detective work. A new hire becomes useful faster because they don't have to memorize which menu holds which fact, and a veteran runs the whole shift from one surface. Less clicking means fewer missed conditions.

HOW IT WORKS

The dashboard is one tall page built from stacked rows — 15 in total: a small "names not imported yet" nudge for a fresh install, plus 14 content rows (the batch hero, the four-number KPI strip, the live recipe tracker, the active-

alarms panel, the tank cards, recent completed batches, and several newer value cards). It does **not** re-poll the PLC on its own; a coordinator simply reuses the same live data the tank and alarm screens already read, so the home view and the detail screens always agree. The page is rebuilt fresh each time you navigate to it and cleans itself up when you leave, so you never see a stale leftover.

VERSUS THE OLD RS-360 SYSTEM

On RS-360, building this same picture cost several screen changes — it opened to a splash and was laid out one screen per plant area, so the crew navigated form by form to piece the state together. There was no consolidated command center at all. Ring opens straight to one always-live home view.

STATUS & HONEST CAVEATS

Status: Live and on a real screen. The command-center spine — batch hero, KPIs, recipe tracker, alarms, and tank cards — is fully built and running. Watch-out: Be precise that "15 rows" means *a fresh-install nudge plus 14 content rows* — row 0 is the nudge, not the hero. The first eight rows (the command-center spine) are all live; some of the *lower* rows (spend, equipment health, email health, stock forecast, batch ETA, run-dry, yield) are newer value cards, and a few of those are still "built but not yet shown everywhere" features covered elsewhere. If asked, say the core shift screen is live and the extra value cards are the growth area.

LIKELY QUESTIONS & HOW TO ANSWER THEM

Q Is this actually a working screen, or a mock-up?

A It's the real, running home screen — it's set as the startup view, and the same data feeds it and the detail screens, so they can't disagree. You can open it on our hardware today.

Q Does opening this dashboard put extra load on the controller?

A No. The dashboard reuses the readings the tank and alarm screens already pull; it doesn't open its own separate conversation with the PLC, so there's no extra polling just to draw the home page.

Q What happens to the old screen-per-area views — did you lose detail?

A No detail is lost. The per-tank and per-mixer screens still exist for deep work; the dashboard is a summary layer on top that saves the operator from assembling that summary by hand.

Q Some of those lower cards — are they all finished?

A The core shift spine is finished and live. A handful of the extra value cards lower on the page are still being wired onto every screen; I'd rather tell you that than pretend everything is 100% done. The shift-running part is complete.

Spot trouble from across the control room in one glance

WHAT IT IS

A large plant-health verdict sits at the top of the alarms area. It reads "All systems normal" in a calm green when nothing is wrong, and flips to a red "Attention required" the instant a real alarm goes active — readable from across the control room without walking up to the screen.

WHY IT MATTERS

Plant health becomes a glance, not a chore. Just as important, the red signal stays meaningful: a list full of *already-resolved* historical alarms will not falsely turn the panel red, so when it does go red, it means something is wrong right now.

HOW IT WORKS

The header color, the verdict text, and the count badge all key off one number — the count of *active* alarms. Zero active means green; one or more turns it red. Crucially it watches the active count, not the length of the list, so old resolved rows sitting in the history don't tint it. When there are genuinely no active alarms, a positive overlay shows a green check and "No active alarms."

VERSUS THE OLD RS-360 SYSTEM

On RS-360 the operator had to open a dedicated alarm form just to learn whether anything was wrong — there was no health verdict on a home screen. Ring puts an honest, distance-readable verdict on the home screen.

STATUS & HONEST CAVEATS

> Status: Live and on a real screen. The verdict is driven by live active-alarm count. Watch-out: If a skeptic claims "it'll cry wolf every time an old alarm is in the list," you can answer firmly that it's genuinely wired to the *active* count, not the list length — this is implemented in code, not a marketing aspiration.

LIKELY QUESTIONS & HOW TO ANSWER THEM

Q If there are twenty resolved alarms in the history, does the panel scream red?

A No. The header only reacts to *active* alarms. Resolved rows can sit in the list all day and the verdict stays green, which is exactly what keeps the red signal trustworthy.

Q Can an operator read it from across the room, honestly?

A Yes — that's the design point. It's a full-width colored header and a large word, not a tiny icon, so the green-versus-red reads at a distance.

Q Does this replace looking at the actual alarm list?

A No — it's the "do I need to walk over?" signal. The detailed alarm list is right below it for when the answer is yes.

Catch a stalled step before the controller alarm does

WHAT IT IS

A live recipe tracker shows exactly which step of the batch is running and how long it has taken. Finished steps get a green check with their recorded time, the current step glows gold with a live ticking timer, and upcoming steps are dimmed. If a step runs unusually long, that row turns amber as a stall warning.

WHY IT MATTERS

A hung step used to be invisible until something downstream went wrong. With a live per-step timer and an amber warning, the crew can see a stall developing and step in earlier, instead of discovering a silent stall after it has already cost the batch.

HOW IT WORKS

The tracker is a table tied to the running batch's step list, with columns for the step glyph, number, operation, preset, actual, mix time, and duration. Coloring is automatic: complete steps muted with a check, the in-progress step gold with a live minutes:seconds timer, pending steps faded. A step turns amber when its elapsed time passes a threshold of **max(2 minutes, 3× the step's expected mix time)**. The whole card hides when the plant is idle and auto-scrolls so the active step is always in view.

VERSUS THE OLD RS-360 SYSTEM

RS-360 showed the *current* step but gave it no elapsed timer, no recorded durations on completed steps, and no stall indicator — so a step could hang with no visual feedback at all. Ring puts a live timer and a stall threshold on every step, so a hung step shows on the dashboard.

STATUS & HONEST CAVEATS

Status: Live and on a real screen. The per-step timers and the amber stall threshold are fully wired to the running batch. Watch-out: Two phrasing traps. (1) Don't cite "alarm #26" as *the* controller stall alarm — that specific alarm number isn't confirmed in the alarm legend; instead lean on the provable point: the old HMI showed the current step but had zero per-step timing or stall indication. (2) Don't promise Ring's amber *always* beats the controller's own alarm — Ring's threshold is tunable and on-glass, but the PLC's internal stall timeout isn't in Ring's code, so frame it as "Ring surfaces the stall on-screen at a threshold the old HMI never had," not a guaranteed race.

LIKELY QUESTIONS & HOW TO ANSWER THEM

Q **Is the stall warning real, or just a colored row that never triggers?**

A It's real and live. The current step carries a ticking timer, and when elapsed time crosses the max(2-minute, 3×-mix) threshold the row goes amber. It's tied to the actual running step, not a static decoration.

Q Does Ring always catch the stall before the PLC's own alarm?

A I won't oversell that. Ring shows the stall on-screen at a threshold we can tune, which the old HMI never offered at all. Whether it beats the controller's internal timeout depends on how that timeout is set — the honest claim is "visible earlier on the operator's screen," not "guaranteed first."

Q What about steps that finished before Ring was even watching?

A Those show an em-dash for duration rather than a fabricated time. Ring only displays a recorded duration for steps it actually timed.

Q Can the threshold be adjusted for a slow recipe?

A Yes — the threshold scales to 3× the step's own expected mix time, with a 2-minute floor, so a naturally long step won't false-trip.

Never act on a frozen reading that looks live

WHAT IT IS

If a hiccup in the link to the PLC stops a value from updating, Ring doesn't keep showing it as if it were current. It dims the tile and stamps when the value was last fresh ("STALE · as of 14:07:32"), so a frozen number is never mistaken for a live one.

WHY IT MATTERS

The single most dangerous thing on an operator screen is a stale number that still *looks* live — the operator makes a decision on data that stopped minutes ago. Communication drops are a real, observed failure mode at this plant, so this guard protects against the worst-case mistake directly.

HOW IT WORKS

Each guarded tile knows the timestamp of the last successful read from the PLC. A converter compares that to "now"; once the reading is more than 10 seconds old, the tile fades and its tooltip flips to a "STALE · as of …" stamp. The age is computed in a clock-immune way (in UTC), so a daylight-saving change or someone setting the clock backward can't trick a frozen value into looking fresh. The big batch hero and the recipe tracker dim the same way when their data goes stale.

VERSUS THE OLD RS-360 SYSTEM

On RS-360, fill graphics and readouts simply froze looking live during a comms drop — no dimming, no last-fresh stamp, no way for the operator to tell. Ring stamps every guarded live tile with its last-fresh time and dims it within about 10 seconds of a drop.

STATUS & HONEST CAVEATS

Status: Live and on a real screen. The freshness dimming and "as of" stamp are wired onto all four KPI tiles and the hero strip. Watch-out: An *older* internal audit doc once listed these freshness converters as "dormant — wire later." That doc is stale; the current code has them wired onto all four KPI tiles. If challenged, point to the live code, not that old note. Also be precise in scope: the guard covers the *live PLC readings* — the KPI tiles, the hero batch strip, and the step tracker — not literally every card on the page. Say "every live PLC reading carries provenance," not "every reading."

LIKELY QUESTIONS & HOW TO ANSWER THEM

Q Is this actually turned on, or is it code that isn't hooked up?

A It's hooked up and live on all four KPI tiles and the hero strip. There was an old planning note calling it "dormant" — that note is out of date; the current screen dims and stamps stale tiles today.

Q How fast does it catch a comms drop?

A Within about 10 seconds — that's the live threshold. After that the tile fades and the tooltip shows the last-fresh time.

Q Could a clock change make a stale value look fresh again?

A No — that's specifically defended. The age is measured in UTC, and a negative age (which a backward clock-set would produce) is forced to "stale" rather than "fresh."

Q Does this guard every single card on the dashboard?

A It guards the live PLC readings — the KPI tiles, the batch hero, and the step tracker. I'd describe it as "every live reading is provenance-stamped," not literally every card, because some of the lower value cards aren't live PLC tiles.

One tap from the number to the screen that fixes it

WHAT IT IS

Four color-coded numbers sit across the top of the dashboard — open alarms, mix-tank temperature, batch percent done, and which tank needs attention. Each is a button: tapping it jumps straight to the screen that handles that issue, so there's no hunting through menus to act on what you just saw.

WHY IT MATTERS

Decisions land faster. The four numbers that actually drive an operator's next move are always on glass and color-coded (red when bad, green when fine), and each is one tap from the screen that resolves it — no navigating to find where to act.

HOW IT WORKS

The KPI strip is a four-column row. Tile 1 (open alarms) goes red above zero, green at zero. Tile 2 shows the mix-tank temperature. Tile 3 shows batch percent done, weighted by how much material has actually gone in. Tile 4 points at the tank needing attention, green only when nothing is flagged. Each tile is styled as a clickable shortcut with a little chevron, and tapping it navigates to the matching detail screen.

VERSUS THE OLD RS-360 SYSTEM

On RS-360 these numbers, where they existed at all, were raw readouts buried on per-area forms — no consolidated strip and no way to click a number to go act on it. Ring makes them a single color-coded strip where each tile is a shortcut.

STATUS & HONEST CAVEATS

Status: Live and on a real screen. All four tiles are wired and navigate on tap. Watch-out: If asked about the tank-attention tile's routing order, be accurate: the temperature-control (TVC) condition always wins first; after that it routes to the highest-priority flagged tank's *own* screen, with storage as the default fallback. It's not a rigid "Use-tanks-always-beat-Storage" rule — describe it as "TVC first, then the top-flagged tank's screen."

LIKELY QUESTIONS & HOW TO ANSWER THEM

Q Do the tiles really navigate, or are they just colored boxes?

A They really navigate — each tile is a button with a chevron, and tapping it opens the relevant detail screen. The alarms tile opens alarms, the temperature tile opens the mix tank, and so on.

Q What decides the color on the alarm tile?

A The live count of open alarms: red above zero, green at zero. Same honest signal as the big health header, just in number form.

Q Where does the "tank needs attention" tile send me?

A To the screen for whatever's flagged. A temperature-control issue takes priority and routes there first; otherwise it opens the most urgent flagged tank's screen, defaulting to storage if nothing else is ranked higher.

Q Is the percent-done number trustworthy?

A Yes — it's weighted by how much material has actually been dosed, not just step count, and it's shown to one decimal. There's also an honesty guard on the final phase, covered in the next feature.

No fake "100% done" for the last 20 minutes of a batch

WHAT IT IS

At the very end of a batch there are usually final steps that aren't metered by quantity (mixing, holding, finishing). On a naïve display those make the math read 100% while the batch is still running. Ring instead shows "99%" with a "Finishing — Step N of M" label, so the headline doesn't claim the batch is done when it isn't.

WHY IT MATTERS

On a real ("golden") batch we captured at the plant, the un-guarded display read "100.0% / est. < 1 min" for about 21 minutes while the operator was still working the batch (15:08 to 15:30). An operator trusting that headline would have walked away too early. Ring keeps the headline honest right where the old math was most wrong.

HOW IT WORKS

When the batch enters its final un-metered steps, the percent math would naturally round to 100 while steps remain. Ring detects this "finishing phase" — defined honestly as *the fraction has effectively hit full but we're still sitting on a recipe step* — and caps the headline at 99%, swaps in a "Finishing — Step N of M" label, and changes "Time Left" to show elapsed-in-step instead of a fake countdown. Once the batch passes its truly last step, it shows an honest 100%.

VERSUS THE OLD RS-360 SYSTEM

This is a fix to an early Ring display flaw as much as a legacy contrast — the un-guarded hero read 100% for roughly 21 minutes on the documented golden batch. The guard caps the headline at 99% and labels the finishing phase explicitly, so the display stays honest exactly where it was previously wrong.

STATUS & HONEST CAVEATS

Status: Live and on a real screen. The 99% cap and finishing label are wired into the hero, the percent KPI, and the step caption. Watch-out: Don't oversell this as a universal "it can never show a wrong 100%." It specifically guards the trailing un-metered finishing steps. A batch that runs long on a *metered* step is not "finishing" and won't be capped — which is correct behavior, but be ready to say so rather than claim a blanket anti-100% guarantee. The exact golden-batch window (15:08:42 to 15:30:16, ~21 minutes) is documented in our code, so those timestamps are defensible.

LIKELY QUESTIONS & HOW TO ANSWER THEM

Q Where does this even matter — isn't 100% just 100%?

A It matters at the end. The final mix-and-hold steps aren't measured by quantity, so the simple math hits 100% with real minutes of work left. We caught it reading 100% for about 21 minutes on a live batch. Ring caps it at 99% and labels "Finishing" until the batch genuinely ends.

Q How do I know it won't just lie the other way and stall at 99 forever?

A The cap only applies during the finishing steps. Once the batch passes its actual last step, it shows a true 100%. The 99% is deliberately temporary and labeled "Finishing — Step N of M."

Q Does this fix every possible wrong percentage?

A No, and I won't claim that. It fixes the end-of-batch finishing-phase case, which is the one that bit us. A batch running long on a measured step isn't "finishing," so it shows the real progress rather than a cap — that's intentional.

Step away and still see what just finished

WHAT IT IS

When a batch completes, the result doesn't vanish. The hero holds a one-line summary — "Batch complete — Formula 1 → Storage Tank 2, 12 steps, 2h 04m, completed 15:30" — until the next batch starts or the operator dismisses it.

WHY IT MATTERS

A two-hour batch used to disappear in a three-second snap the instant it ended. An operator who stepped away for a moment came back to a blank "Idle" screen with no trace of what had just run. Holding the summary means they can still see exactly what finished.

HOW IT WORKS

When the running step count drops to zero (the batch ended), the code captures the recipe, target tank, step count, duration, and finish time *that it recorded while the batch was running*, and shows them in a hold panel instead of instantly flipping to "Idle." A "Dismiss" pill clears it. The refresh is driven by the completion event itself, not a polling timer that would erase it.

VERSUS THE OLD RS-360 SYSTEM

RS-360 had no held completion summary on a home screen — once a batch ended there was nothing waiting on the home view. Ring holds the completed-batch summary until the next batch or an explicit dismiss.

STATUS & HONEST CAVEATS

Status: Live and on a real screen. The hold panel and dismiss action are wired and event-driven.
Watch-out: The summary only fills in duration and finish-time if the batch recorder logged that batch within a roughly 10-minute window. If the recorder missed it, the summary *omits* duration/time rather than guessing — so be ready to say "it leaves out what it can't verify, it never pins a wrong number." That's a deliberate honesty choice, not a bug.

LIKELY QUESTIONS & HOW TO ANSWER THEM

Q **Is this just the next batch's data showing early?**

A No — it captures the finishing batch's own details while that batch is still running, then holds them. It's the batch that just ended, not the next one.

Q **What if the recorder didn't catch the batch — does it make up a duration?**

A No. If it can't confidently attribute the completed-batch record, it shows the parts it knows and leaves the duration or finish time blank rather than inventing one.

Q **How long does it stay up?**

A Until the next batch begins or the operator taps Dismiss. It won't auto-erase on a timer.

Catch a developing tank problem before it stops the line

WHAT IT IS

The fourth KPI tile points the operator straight at the next tank needing attention — including a developing temperature-control problem. The plant's temperature-control units (the jacket-water/heater units that hold a tank at temperature, called "TVC") can have a condition like low jacket water that, in a specific situation, the controller's own alarm won't raise.

WHY IT MATTERS

At the live plant we found a real gap: a TVC unit was running low while the dashboard read "all systems normal," because that condition wasn't surfaced anywhere the operator was looking. This tile routes the operator to the developing problem before it grows into a line stoppage.

HOW IT WORKS

The attention tile reads the *same* TVC input bits the TVC detail screen shows and raises a plain-language flag like "TVC 1 — jacket water LOW." The temperature-control case is ranked first; otherwise the tile points at the most urgent flagged tank's screen. Importantly, it reads the live input directly rather than waiting on the controller's alarm logic — so it can flag the condition in the specific case where the controller's own alarm can't.

VERSUS THE OLD RS-360 SYSTEM

Legacy operators relied on a red TVC indicator on the main navigation. An early Ring build actually hid TVC behind a screen nobody opened, so the dashboard read "normal" while TVC 1 was low at the live plant. Ring now synthesizes that low-water condition into the attention tile and routes there on tap, recovering the legacy indicator's coverage and adding one-tap navigation.

STATUS & HONEST CAVEATS

Status: Live and on a real screen. The attention tile is wired and the TVC low-water flag is fed in.
Watch-out: Be careful with the absolute. The controller *does* have a TVC water alarm (#841) and it has been observed firing on the live PLC — so don't say "the controller can't raise it" in general. The accurate statement is: "the #841 alarm can't raise it *when that TVC's control sequencer is disabled*," which is exactly the situation we hit. Also clarify it's a *synthesized convenience flag* reading the same input bits the TVC screen shows — not a new PLC alarm.

LIKELY QUESTIONS & HOW TO ANSWER THEM

Q You're saying the PLC can't alarm on low TVC water — that's not true, is it?

A You're right to push there, and I'll be precise: the controller does have alarm #841 for it, and we've seen it fire. The case Ring covers is when that specific TVC's control sequencer is switched off — then #841 is gated out and can't fire, yet the water can still be low. Ring flags it from the raw input in that gap.

Q Is Ring creating a new alarm the controller doesn't have?

A No — and that's an important distinction. Ring isn't inventing a PLC alarm; it reads the same TVC input bits the TVC screen already shows and surfaces them as a convenience flag on the dashboard. The controller logic is untouched.

Q How do I know this is a real gap and not a hypothetical?

A Because we hit it live. At the plant, TVC 1 was low while the dashboard read "all normal" in an early build. This feature is the direct fix for that observed gap.

Q Where does tapping the tile take me?

A To the TVC screen for the flagged unit, since temperature-control conditions are ranked first. If the flag were a different tank, it would route to that tank's screen instead.

Make production decisions on real history, not invented numbers

WHAT IT IS

The trends card plots genuine completed-batch data — real dates, with an "As of HH:mm" timestamp — so supervisors read actual production numbers. If there's no history yet, it shows an honest empty state instead of fake figures.

WHY IT MATTERS

A trends chart is only useful if you can trust it. Supervisors and anyone evaluating the system should be basing decisions on real production numbers, not demo placeholders dressed up to look authoritative.

HOW IT WORKS

The chart reads completed batches from Ring's batch database for a date range and groups them by shift or by day. Labels are real calendar dates (like "Wed 4"), never hardcoded weekday names, and an "As of HH:mm" caption stamps every refresh. If the query fails or there's no data, it degrades to an honest "no history" message rather than throwing an error or showing nothing meaningful. An earlier Ring build had filled this card with random demo numbers — those were removed entirely.

VERSUS THE OLD RS-360 SYSTEM

RS-360 had limited-to-no on-glass trend visualization of this kind. Its "Graph" screen was a live-value *mimic* — a background diagram with up-to-eight live numeric readouts — not a chart plotting values over time. Ring sources the trend from the real batch repository with a visible "As of" caption.

STATUS & HONEST CAVEATS

Status: Live and on a real screen. The chart reads the real batch repository, stamps "As of," and shows an honest empty state. Watch-out: Two things. (1) A buyer who searches the old code *will* find files named "ScreenGraph" and the word "Graph" — be ready to explain the difference: the legacy screen is a live-value mimic with instantaneous gauges, it plots nothing over time. Say "limited-to-no on-glass *trend* visualization," which is the fair, hedged phrasing. (2) On Ring's side, confirm before any live demo that the volume chart's vertical-axis label reads gallons, not "lbs" — there was a note that the axis title might still say pounds while the data is summed gallons. Verify that label is fixed if you show it.

LIKELY QUESTIONS & HOW TO ANSWER THEM

Q Is this real data or a demo chart?

A Real. It reads completed batches from Ring's own database and stamps each refresh with "As of" a real time. An earlier build did have random demo numbers here — we removed them precisely because fake data on a trends card is disqualifying.

Q The old system had a "Graph" screen — so this isn't new, right?

A The old "Graph" is a live mimic — a diagram with live numeric readouts and gauges. It has no time axis and stores no history, so it doesn't plot anything over time. Ring's trend actually charts completed batches across days and shifts. They're different tools.

Q What does it show before there's any history?

A An honest empty state — a clear "no history yet" message — not a flat line or invented numbers. And if the database query ever fails, it degrades to that same honest state instead of crashing.

Q Are the units labeled correctly?

A They should read gallons on the volume view. I'll confirm that axis label live before showing it — there was a note to double-check it wasn't still labeled pounds.

Let a tech self-troubleshoot on the floor without opening Studio 5000

WHAT IT IS

A set of maintenance tools built into Ring so a password-holding technician can diagnose comms problems right on the floor — without a protocol analyzer or Allen-Bradley's engineering software, **Studio 5000**. It includes a Tag Inspector (read any PLC value), a read-only PLC Health drawer, a Diagnostic Console showing the raw network frames, a crash-dump pipeline, and a one-tap diagnostic bundle. The operator-facing touches — a numeric keypad for gloved hands, an F1 help overlay, and a Ctrl+B "go to batch" shortcut — round it out.

WHY IT MATTERS

When comms act up, the usual answer is "call someone with Studio 5000 and a laptop." These built-in tools let a qualified tech answer "is the PLC link healthy, and what is this tag reading?" in place, on the same screen the crew already uses.

HOW IT WORKS

Ring talks to the PLC over **EtherNet/IP** (an open industrial-network standard) using an open library called **libplctag**, addressing tags by name. Because that's a direct, standards-based link, there are no separate bridge programs to keep running. The Tag Inspector and Diagnostic Console are password-gated dialogs; the PLC Health drawer shows the last-success time and recent link-state transitions; the Diagnostic Console shows the raw transmit/receive (TX/RX) frame tail. Poller startup is staggered so the controller's limited number of simultaneous network sessions doesn't get swamped at launch.

VERSUS THE OLD RS-360 SYSTEM

RS-360 logged comms to disk but offered no modern instrumentation — no tag inspector, no connection-state view, no one-tap bundle — and it reached the PLC through separate bespoke Borland "bridge" programs (an ASAB/TCP exe, DH+/TCP modules) that had to be kept alive alongside it. Ring talks EtherNet/IP directly with named tags and ships first-class diagnostics, so there are no sidecar bridge processes to babysit.

STATUS & HONEST CAVEATS

> Status: Live and on a real screen. Every named tool exists on disk and is reachable; the direct EtherNet/IP link with no bridge exes is real. Watch-out: Two phrases need softening. (1) "Let the *floor* self-troubleshoot / read *any* tag in place" is true for a **supervisor/admin password holder**, not any operator — the Tag Inspector and Diagnostic Console are password-gated dialogs, hidden until you authenticate. Frame the gating as a deliberate safety feature: "a password-holding tech self-troubleshoots in place without Studio 5000." (2) "Non-blocking confirmations" is uneven — Ring does have non-blocking toast pop-ups, but the most prominent action, Acknowledge-All-alarms, uses a blocking Yes/No dialog. Don't claim *all* confirmations are non-blocking. Also: the PLC Health drawer shows last-success time and link-state transitions; the raw TX/RX frames live in the *Diagnostic Console*, not the drawer — keep those straight. And drop the "drove startup errors to zero" phrasing; the accurate claim is "staggered startup smooths the session ramp and avoids the startup error burst" (the "zero" was never measured).

LIKELY QUESTIONS & HOW TO ANSWER THEM

Q Can any operator read any tag, or is it locked down?

A It's deliberately locked down. The Tag Inspector and Diagnostic Console require a supervisor/admin password and the buttons are hidden until you authenticate. That's a safety feature — a password-holding tech can self-troubleshoot in place, but a random operator can't go poking at raw tags.

Q You say no bridge software — prove it.

A The old system reached the PLC through separate Borland programs — an ASAB/TCP bridge and DH+/TCP modules that had to run alongside it; those files are right there in the legacy source. Ring talks EtherNet/IP directly through the open libplctag library with named tags, so there's no separate bridge process to keep alive.

Q Are all the confirmations really non-blocking pop-ups?

A Not all of them, and I won't claim otherwise. Ring does have non-blocking toast notifications, but the Acknowledge-All-alarms confirmation is a blocking Yes/No dialog. The "non-blocking" point is true for the toasts, not universally.

Q Does the health drawer show the raw network packets?

A The drawer shows the last-success time and recent link-state changes. The raw transmit/receive frames are in the separate Diagnostic Console. Two surfaces, on purpose.

Q Does staggering the pollers actually fix the startup errors?

A The controller only allows a handful of simultaneous network sessions, so launching all pollers at once caused a burst of session errors that self-recovered. Staggering their startup smooths that ramp and avoids the burst. I'd phrase it that way rather than claiming a measured "zero," which we didn't formally quantify.

End false "PLC down" calls — know an outage from an idle controller

WHAT IT IS

Ring classifies the state of its connection to the PLC into clear categories, so the team can tell the difference between a controller that is genuinely down and one that is reachable but simply not running its program (idle). That stops the crew from chasing "the PLC is down!" calls when it isn't.

WHY IT MATTERS

A reachable-but-idle controller and a true outage used to look identical — data just stopped. That ambiguity drives false escalations. Naming the difference means the team responds to real failures and ignores false alarms, and the very same signal drives the on-screen dimming the operator already sees.

HOW IT WORKS

A heartbeat tracker watches a value the PLC updates and resolves the link into states: **Connecting, Connected, Stale, Starved, and Disconnected**. "Starved" is the key one — it means the PLC is reachable but its heartbeat isn't advancing (the controller's main task is idle), i.e. *reachable-but-idle*, distinct from a true "Disconnected" outage. The same last-success timestamp that sets this state also feeds the dashboard's freshness dimming on a one-second tick, so connection state and on-glass staleness are one consistent signal.

VERSUS THE OLD RS-360 SYSTEM

RS-360 logged comms but had no connection-state classifier — it couldn't tell *why* data stopped, so a reachable-but-idle controller and a real outage looked the same to the operator. Ring's heartbeat classifier is the single source feeding both alerting and the dashboard's staleness dimming.

STATUS & HONEST CAVEATS

Status: Live and on a real screen. The classifier runs and feeds both the connection display and the KPI dimming. Watch-out: Describe it as a **five-state** classifier (Connecting / Connected / Stale / Starved / Disconnected), not "four-state" — earlier copy undercounted it. Technically there's also an initial "Unknown" state before the first read, so if someone counts six in the code, that's the startup placeholder; "five operational states" is the fair description. Quote the heartbeat tracker itself as the source of truth, not the sales doc.

LIKELY QUESTIONS & HOW TO ANSWER THEM

Q How is this different from just "connected or not connected"?

A Because "not updating" has more than one cause. Ring separates a true outage ("Disconnected") from a controller that's reachable but idle ("Starved"), plus the in-between "Stale" and the "Connecting" startup state. That's what stops false "PLC down" calls.

Q Is it four states or five?

A Five operational states: Connecting, Connected, Stale, Starved, Disconnected. There's also an "Unknown" placeholder before the first read, so the code technically lists six, but the five above are the ones that mean something to a user.

Q What exactly is "Starved"?

A The PLC is reachable on the network, but the heartbeat value isn't advancing — its main program task isn't ticking. That's a reachable-but-idle controller, which is a very different situation from a cable being unplugged, and Ring names it as such.

Q Does this connect to what the operator actually sees?

A Yes — directly. The same last-success timestamp that drives this state classification also drives the freshness dimming on the KPI tiles, on a one-second tick. The diagnostic state and the operator's dimming are the same underlying signal, so they never disagree.

Chapter 2 — Running a Batch Without the Guesswork

A "batch" is one production run of one glue recipe: the operator tells the plant which recipe to make and how much, and the plant doses each ingredient into a mixing tank in the right order and amount. This plant makes corrugating adhesive — the industrial starch glue that holds cardboard layers together — so a "recipe" here is a list of ingredients with target weights and mix times, not a food formula.

Two pieces of vocabulary you must be able to define on demand. The **PLC** (Programmable Logic Controller) is the rugged industrial computer that physically runs the valves, pumps and mixer. It is the thing that actually *makes the glue*. The **HMI** (Human-Machine Interface) is the operator's screen — the window a person uses to tell the PLC what to do and to watch what it is doing. **Ring is the new HMI. It does not change the PLC.** Same controller, same wiring, same physics — only the screen in front of the operator is new. The old screen was "RS-360", a 2002 DOS-era program written in Borland C++; Ring replaces that screen on the exact same untouched controller.

The old way of running a batch was guesswork dressed up as procedure. On RS-360 the operator typed numeric "op-codes" — short number codes that each stand for an ingredient or an action — against PLC "data-file addresses" (numbered memory slots). There was no plain-language "here is exactly what I'm about to send" check, the live picture of a running batch vanished on a restart, and the historical record was trapped in a dead 1990s database format nobody could search. This chapter is about removing that guesswork: a guided start with a plain-sentence safety check, a live step tracker that stays honest when the network hiccups, a searchable permanent history, and analytics that compare today's batch against your own proven good ones.

One frame to set before the feature list, because a skeptical buyer will test it: **Ring currently ships with a "read-only gate" turned on** (a setting called `ReadOnlyMode=true`). With that gate on, every command that could change the plant is suppressed — Ring reads the controller but issues no write commands. On-site it read all 133 live data points without sending a single command that could change the process. (Reading does put request packets on the network, so the honest line is "zero write commands," never "zero bytes.") Several features below are therefore *built and validated but deliberately gated off* until the owner flips that switch — and saying so plainly is the point.

Keep your plant's memory — and a roadmap of upgrades built on top of it

WHAT IT IS

This is the "why switch at all" feature, and it has two halves. The first half is real and shipped: Ring keeps a durable record of what the plant makes — a record that survives a reboot — and on top of that record it already shows a small "when will this batch finish" estimate on the main dashboard. The second half is a *plan*: a vetted wish-list of bigger upgrades (the headliner is a "Batch Dagplanner" — Dutch for "day planner" — that would tell the operator *when the batch will next need them*, e.g. "hold around 07:12, ready around 08:05," so they can do other rounds instead of standing at the panel). The wish-list is documented and agreed, but the big items are not built yet.

WHY IT MATTERS

Today the live, in-the-moment picture scrolls away. If the HMI restarts mid-batch, the running picture is gone, and there is no step-by-step timing kept that a supervisor could mine later. Keeping that record is what unlocks every downstream improvement — predictive timing, "is this batch running long," shift summaries. An owner should

care because this is the difference between a screen that forgets and a system that accumulates 20 years of operating knowledge you can actually use.

HOW IT WORKS

The shipped foundation is a step-timing table (internally `BatchStepEvent`) that is written as each batch step transitions, plus a "Process Historian" that quietly samples tank temperature and level once a minute *from data Ring already reads* — so it adds no new load on the controller. A completion-estimate card on the dashboard reads the *medians* of that step-timing history (no controller reads or writes at all) and shows "Batch done in ~22 min," with honest "no active batch" and "collecting data" states. The larger Dagplanner timeline, a practice/replay mode, a legacy-look skin, and a few other items are written up as a scored design plan — proposed, not coded.

VERSUS THE OLD RS-360 SYSTEM

Be precise here, because this is the single most fact-checked claim in the deck. RS-360 did **not** record nothing — it kept roughly 1,404 batch headers, 16,776 ingredient step rows, and shift and alarm records too. The honest gap is not "no history"; it is that RS-360 kept all of it **locked in a dead Paradox/BDE database format** (a corruption-prone 1990s file format that needs a custom reader to open), with **no step *durations* or timing, no continuous time-series historian or surviving trends, no equipment-runtime stats, no golden-batch profiles, no operator notes, and no way for an operator to search any of it.** Ring keeps step timing, a persistent historian, and a searchable record — and surfaces the history the legacy wrote for two decades but never let anyone read.

STATUS & HONEST CAVEATS

Status: Partly live, partly roadmap. The durable record, the Process Historian, the step-timing table, and the completion-ETA card are shipped and wired onto the dashboard; the headline upgrade slate (full Dagplanner timeline, practice mode, legacy skin, shadow-write ledger, parallel-run scoreboard, legacy-history import) is roadmap — a committed *plan*, not yet built.

Watch-out: The marketing copy slips in two directions, so steer it back yourself. (1) It *undersells* Ring by implying the predictive-timing idea is wholly unbuilt — a completion-ETA card backed by a real step-timing table is already live on the dashboard; if asked "is there any predictive batch timing today?" the honest answer is **yes, a completion estimate**, only the richer per-step appointment timeline is roadmap. Do not call the ETA card "not built." (2) It *oversells* the legacy gap — never say RS-360 "records nothing that survives." Say it kept history but in a dead, unsearchable format. Also: don't read out concrete class names for the unbuilt items as if they exist, and note that a couple of evidence paths in the copy are stale (the legacy-import script lives under `merged_code/scripts/`, and a cited commit can't be verified from this folder).

LIKELY QUESTIONS & HOW TO ANSWER THEM

Q So is the headline "Batch Dagplanner" actually in the product, or is it a slide?

A The full per-step "next time it needs you" timeline is a design on the roadmap, not in the product yet — I won't pretend otherwise. What *is* in the product today is a working completion-time estimate on the dashboard, built on a real step-timing record, which is the foundation the Dagplanner sits on.

Q You said the old system "keeps nothing." My engineer says we have years of batch files. Which is it?

A You're right and the slide overstates it — the old system did keep batch, step, shift and alarm records. The real problem is they're trapped in a corruption-prone 1990s Paradox format with no step durations, no trends, and no way to search them. Ring keeps the same kinds of records in an open, searchable form and adds the timing the old format never had.

Q Does any of this new historian put extra load on our controller?

A No. The historian samples temperature and level from data Ring is already reading once a minute, and the completion estimate is computed purely from Ring's own database — zero additional reads and zero writes to the controller.

Q When will the rest of the roadmap actually be built?

A It's scoped and prioritized but not scheduled into the product yet, so I'd treat it as "planned, not promised." The value case stands on what's already shipped — the durable record, the historian, and the ETA card — with the rest as upside.

Confirm exactly what gets sent before the plant arms

WHAT IT IS

A guided way to start a batch that ends in a plain-English safety check. The operator picks a recipe from a list, types the target fill level, chooses the batch mode, and presses Start — and before anything is sent to the controller, Ring shows **one plain sentence of exactly what will be sent** ("Formula 1, Single Batch, No splitting, 625 gallons") and asks them to confirm it.

WHY IT MATTERS

On the floor, the danger is a fat-fingered code arming the plant without anyone seeing what they actually requested. This feature puts a human-readable checkpoint between the operator's keystrokes and the controller, so a wrong recipe or a wrong volume is caught *before* it reaches the equipment rather than discovered in a ruined batch.

HOW IT WORKS

Configuration is written first; arming happens last. When the operator confirms, Ring writes the recipe number, target level, and split settings to the controller in order, and only *then* writes the single "go" signal that arms the batch. If that final arm step fails, Ring makes a best-effort attempt to disarm so a half-configured batch can't sit there armed. A "heartbeat" check (a continuous are-you-alive ping to the controller) blocks the whole thing if the PLC link is down, and the read-only gate, when on, short-circuits to an honest "validated but NOT sent" message instead of touching the plant.

VERSUS THE OLD RS-360 SYSTEM

On RS-360 the operator keyed recipes and levels against numeric op-codes — the ingredient list itself doubled as the code legend — with only a plain Yes/No pop-up and no read-only safety gate, so a mistyped code could arm

the plant unseen. Ring shows the request in plain language, arms last, disarms on failure, and can run the entire flow with writes suppressed for safe rehearsal.

STATUS & HONEST CAVEATS

Status: Live and on a real screen. The guided flow, the plain-language confirmation, and the arm-last/disarm-on-failure ordering are all built, wired into the navigation, and tested.

Watch-out: Two things to keep straight. First, the original slide title called this a "typed confirmation gate" — that's misleading, because you don't type a confirmation word; you read a plain sentence and click confirm (the only typing is the fill level earlier). Call it a **plain-language confirmation gate**. Second, with the read-only cutover gate on (as it ships today), the final arm command is suppressed by design — the flow validates and shows "NOT sent." If asked "does it actually start a batch on the plant today?", the honest answer is: the screen and safety flow are complete and tested, but writes are intentionally gated off during cutover until the owner sets `ReadOnlyMode=false`.

LIKELY QUESTIONS & HOW TO ANSWER THEM

Q Do I have to type a secret word to confirm, like the slide implies?

A No, and I'd correct that wording — you read one plain sentence of exactly what will be sent and click to confirm. It's a plain-language confirmation, not a type-to-confirm box.

Q Can a half-configured batch accidentally arm the plant?

A No. Ring writes all the settings first and the single "go" signal last; if that last step fails it tries to disarm, and if the controller link is down the write is blocked entirely. The arm-last design is specifically there so a partial configuration can't go live.

Q Right now, does pressing Start actually run a batch on our line?

A Not while the read-only safety gate is on, which is how it ships — it validates everything and tells you plainly "validated but NOT sent." It becomes a live start the moment you turn that gate off, which is a deliberate, single-switch cutover step.

Q How is this safer than our old Yes/No prompt?

A The old prompt just asked Yes or No to numeric codes you'd already keyed in — it never showed you in words what those codes meant. Ring spells out the recipe, mode and volume in plain English before anything is sent, and adds the arm-last and read-only protections the old screen never had.

A live, honest checklist of where the batch stands

WHAT IT IS

While a batch runs, Ring shows the recipe as a live checklist: every ingredient with its target ("preset") amount next to its actual amount, a Complete / In Progress / Pending status per step, an overall percent-done, and a current-step caption that highlights and scrolls itself into view.

WHY IT MATTERS

Operators used to infer where a batch was by squinting at gauges. This shows it directly — including whether each ingredient landed on target — so the operator knows at a glance what's done, what's running, and what's left, without guessing.

HOW IT WORKS

A background reader checks the controller's "current step" every two seconds (off the screen's drawing thread, so the display never freezes), reads the whole array of step targets and actuals, and publishes a snapshot the screen redraws on a timer. It's deliberately hardened against network hiccups: a failed read *keeps the last good picture* instead of blanking, and ending a batch requires three confirmed "idle" reads in a row (about six seconds) so one momentary blip can't wrongly declare the batch finished. The percent-done is weighted by ingredient amount and explicitly flags the "finishing" phase, so the trailing zero-amount mix/transfer steps don't show a fake "100% / under a minute."

VERSUS THE OLD RS-360 SYSTEM

RS-360 did show a per-step target/actual table on the mixer screen — but with no amount-weighted overall percent and no "step N of M" or ETA strip. And on your own 2026-06-10 golden batch it read "100% / under 1 minute" for the *entire 21-minute* finishing phase. Ring gives an amount-weighted percent with an explicit "finishing" label, plus the keep-last-snapshot and three-confirmed-idle logic so the display stays truthful through a network flap.

STATUS & HONEST CAVEATS

> Status: Live and on a real screen. The tracker, the anti-flap logic, and the finishing-phase fix are all built and wired onto the dashboard tracker grid.

> Watch-out: The "100% for 21 minutes" story is from your *own* golden batch and is written into the code's comments, so it's solid to cite. The one honest limitation: the tracker reads live controller data, so it only updates while the heartbeat is connected — if the link drops, it holds the last good picture rather than going blank (by design), which is honest behavior, not a stall.

LIKELY QUESTIONS & HOW TO ANSWER THEM

Q What happens to the display if the network stutters mid-batch?

A It holds the last good snapshot instead of flashing to blank or to a fake "idle," and it won't declare the batch finished until it sees three confirmed idle reads in a row. The design goal is that the screen never lies to the operator during a hiccup.

Q Where does the "100% for 21 minutes" claim come from — is that a competitor jab?

A No, it's from our own golden batch on 2026-06-10, and the fix is documented right in the code comments. The old display hit 100% the moment the dosing was done and sat there through the 21-minute mix/transfer phase; Ring labels that as a finishing phase instead.

Q Does polling every two seconds slow the controller or the screen down?

A No — the reads run on a background thread, never the drawing thread, so the screen stays responsive, and a two-second cadence is light traffic for this controller.

Q Can the operator see if an ingredient came up short?

A Yes — every step shows target amount next to actual amount, side by side, so an under- or over-dose is visible immediately rather than inferred from a tank gauge.

A searchable, permanent record of every batch

WHAT IT IS

Every batch is recorded automatically — formula, tank, start and end time, target versus actual volume, and a status (Running, Complete, Cancelled, or Interrupted). Operators can browse, filter by date or tank, search by batch number or formula, and print or export a Batch History report as a PDF.

WHY IT MATTERS

Quality questions and management questions both come down to "what exactly did we make, and when?" This gives a complete, queryable answer instead of reconstructing it from memory or paper. The dedicated "Interrupted" status is the honest touch: if the HMI rebooted mid-batch, the record says so rather than leaving a batch that looks like it ran forever.

HOW IT WORKS

Ring records a batch as "Running" the moment it arms, into an open **SQLite** database (a standard, single-file, industry-standard database — the same engine inside most phones and browsers). When the batch ends, Ring finalizes it together with its ingredient-usage rows and actual volume in one all-or-nothing transaction, guarded so two events can't race and corrupt it. A batch left "Running" after a crash or reboot is reconciled to "Interrupted" on the next startup. A global "Go to Batch" shortcut (Ctrl+B) searches every batch by number or formula.

VERSUS THE OLD RS-360 SYSTEM

RS-360 *did* keep batch history (roughly 1,404 batch headers and 16,776 step rows) and could print a report — that part is parity, and you should grant it. What it lacked: operator notes, any crash/reboot distinction, and full-text search; and the data was locked in Paradox `.DB` files that need a custom reader to extract. Ring adds full-text "Go to Batch" search, an honest "Interrupted" status, and one-transaction finalization on an open database anyone can read.

STATUS & HONEST CAVEATS

Status: Live and on a real screen. Recording, the Interrupted reconciliation, Ctrl+B search, and the printable/PDF report are all built and wired.

Watch-out: One honest point a buyer should hear up front — **Ring's history starts empty on a fresh install.** Your 20 years of legacy batches only appear after the legacy-history import is run, and that importer stage is on the roadmap, not shipped today. So "searchable record of every batch" is true going forward from install; the historical backfill is a separate, planned step.

LIKELY QUESTIONS & HOW TO ANSWER THEM

Q If the HMI reboots during a batch, do I get a batch stuck "running" forever?

A No — that's exactly the case the "Interrupted" status handles. On the next startup Ring reconciles any batch still marked Running to Interrupted, so the record is honest about what happened instead of looking like it never ended.

Q Will all our old batches show up on day one?

A No, and I want to be straight about that — Ring's database starts empty and records everything from install forward. Importing the 20-year legacy archive is a planned step that has to be run separately; it's on the roadmap, not in the box today.

Q Is the data locked into a proprietary format like the old one?

A The opposite. It's SQLite — a single open database file that standard tools can read — so you're never stuck needing a custom reader the way you are with the old Paradox files.

Q Can I get a clean report for a quality auditor or a customer complaint?

A Yes — you can filter by date or tank, search to the exact batch, and print or export it as a branded PDF report.

Edit a recipe once — screen, controller, and audit trail stay in sync

WHAT IT IS

Supervisors can edit any of the six recipes step by step — operation, amount, mix-time — with add, insert, delete and reorder, plus a live calculated batch weight and volume. One "Apply" saves the change to Ring's database *and* pushes it down to the matching recipe slot in the controller. Every save is snapshotted, so you can see what a recipe looked like at any past point.

WHY IT MATTERS

The classic failure is drift: the recipe on the screen, the recipe in the controller, and the written record slowly disagree. A single Apply that writes all three at once keeps them in lockstep, and the version snapshots give you an audit trail — who changed what recipe, and when.

HOW IT WORKS

The editor is a spreadsheet-like grid of steps. When you open it, Ring reads the live recipe bank from the controller; when you Apply, it writes the steps to Ring's database *and* to the controller's recipe slots (recipes 1 through 6, up to 30 steps each), maps each operation name to its numeric op-code, and shows a live weight/volume estimate using the same calculation the legacy used. On a successful Apply it also serializes the step list to a dated version snapshot — done "best-effort" so that even if the snapshot fails, your save still succeeds.

VERSUS THE OLD RS-360 SYSTEM

On RS-360, recipe edits went against a Paradox `Recipe.DB`, then pushed to the controller through a scratch-tag plus a multi-step handshake with a five-PC "dirty-flag" sync. Crucially, `Recipe.DB` held only the *current* values — no versioning — so the only way to tell that operators had edited presets was to mine 16,776 historical step rows after the fact. Ring's one Apply does the database write, the explicit controller write, and an automatic dated version snapshot, with a clear per-result message.

STATUS & HONEST CAVEATS

> Status: Live and on a real screen. The editor, the dual DB-plus-controller write, and the version snapshots are all built and wired into the setup navigation.

> Watch-out: One nuance not to oversell. The per-ingredient "amount limits" that clamp obviously-wrong entries are documented as *unverified guesses* against the real plant's settings — they are a safety backstop, not plant-calibrated values. So pitch the live weight/volume readout as the mistake-catcher, and don't claim the limit-clamping is tuned to your actual recipes. Also: only recipes 1 through 6 push to the controller.

LIKELY QUESTIONS & HOW TO ANSWER THEM

Q If I edit a recipe, do I have to separately update the controller?

A No — that's the whole point. One Apply writes the database, pushes the recipe to the controller's slot, and snapshots the version, so the screen, the plant, and the record can't drift apart.

Q Are those ingredient limits tuned to our real recipes?

A Not yet — I'll be honest, they're a conservative safety backstop, documented as unverified against the real plant settings. The mistake-catcher you should trust today is the live weight and volume readout, which updates as you edit.

Q Can I see who changed a recipe and what it used to be?

A Yes — every successful Apply takes a dated snapshot of the full step list, so you can view any past version. The old system kept only the current values, with no version history at all.

Q What if the version-snapshot step fails — do I lose my edit?

A No. The snapshot is best-effort and runs after the save, so a snapshot failure never blocks the actual recipe change; you'd still get your edit applied with a clear result message.

Spot a drifting batch against your own good runs

WHAT IT IS

Ring builds a "golden" reference curve from past good batches of a recipe and overlays the current live batch on top of it, flagging when the temperature is wandering away from that proven profile — with honest "still collecting data" states until enough good batches exist to build the reference.

WHY IT MATTERS

Today you find out a batch went bad *after* it's finished. This gives an early, visual "this run is drifting off the recipe" signal grounded in the plant's own history of good batches — and it does it without adding any load to the controller, because it reuses data Ring already reads.

HOW IT WORKS

The Process Historian samples the tank readings Ring already has in memory once a minute (no new controller reads) and keeps 30 days of it, self-pruning. The golden-batch screen slices that history into per-batch temperature traces, builds a median "golden" reference once it has at least three completed batches of that recipe, and draws golden-versus-live as two lines with current and maximum deviation and a drift flag when temperature strays past a five-degree band. **The scope is temperature only** — there is no viscosity sensor on this controller to read.

VERSUS THE OLD RS-360 SYSTEM

RS-360 had no process historian and no golden-batch comparison at all — only per-shift production totals and batch headers, never a time-series temperature trace per batch. So there was simply no way to overlay a live

batch against historical good ones. Ring adds a self-pruning 30-day per-tank time-series, a median golden reference, and a live deviation flag — net-new analytics with no added controller traffic.

STATUS & HONEST CAVEATS

Status: Live and on a real screen. The historian, the golden-reference math, the overlay screen, and the honest empty states are all built and wired into the Reports navigation.

Watch-out: Two honest caveats to volunteer. (1) Scope is **temperature drift only** — this plant's controller exposes no viscometer (viscosity sensor) tag, so don't let anyone infer viscosity statistical process control from this. (2) The historian only accumulates *forward* and needs at least **three completed batches** of a recipe before any golden curve appears — on a fresh install or a live demo it will correctly show "collecting data — needs N batches," so seed or pre-run before demoing.

LIKELY QUESTIONS & HOW TO ANSWER THEM

Q Does this measure glue viscosity? That's our real spec.

A No, and I won't imply it does — this controller has no viscosity sensor, so the overlay tracks temperature drift only. Viscosity SPC would require a sensor the plant doesn't currently have on the line.

Q How does it know what "good" looks like?

A It builds the reference from your own past good batches of that recipe — a median curve across at least three completed runs — so the standard is the plant's own proven history, not a number we made up.

Q If I demo this tomorrow on a fresh system, will it show a nice curve?

A Only if there are at least three completed batches of that recipe in the history — otherwise it honestly shows "collecting data." For a demo you'd want to seed or pre-run a few batches first; that "collecting data" state is the system being truthful, not broken.

Q Does building all this history hammer the controller?

A No — it samples readings Ring already has in memory once a minute, so it adds analytics with zero extra reads or writes to the controller.

Pin operator notes to the exact batch

WHAT IT IS

An operator can attach a free-text note to any batch — "added extra borax, starch was clumping," or "stopped early, corrugator down" — edited from the batch report and saved with that batch forever.

WHY IT MATTERS

The "why" behind a batch used to live on a paper log or in someone's head and was effectively lost. Tying that context to the exact batch it describes — searchable later, printed on the report — is invaluable when you're chasing a quality complaint or explaining a bad batch weeks later.

HOW IT WORKS

The batch record gained a notes field, added to the database safely so the schema can't drift, and shown on every screen where a batch appears. The operator types or clears the note on the Batch Report window and it saves straight to that batch; clearing it empties the field.

VERSUS THE OLD RS-360 SYSTEM

On RS-360 operator context simply wasn't captured — the batch table was fixed columns (PC ID, formula, tank, times, weights, status) with no free-text field, so the reasoning behind a batch was gone. Ring lets you attach a free-text note to every batch, searchable and printed on the batch report.

STATUS & HONEST CAVEATS

Status: Live and on a real screen. Fully built end to end — the data field, every read path, and the edit box on the Batch Report.

Watch-out: Nothing material to defend here. The only honest note: the field is free-text and not validated or mandatory — it's a place to capture context, not an enforced data-entry field.

LIKELY QUESTIONS & HOW TO ANSWER THEM

Q Where does the operator actually type the note?

A On the Batch Report window — there's a note box and a save button right there, and the note then shows everywhere that batch appears.

Q Is the note required, or validated?

A No — it's deliberately free-text and optional, meant to capture the operator's "why" in their own words. If you wanted it mandatory or structured, that would be a separate change.

Q Did the old system have anything like this?

A No — the old batch records were fixed columns with no free-text field at all, so the reasoning behind a batch was simply never captured.

Re-run the usual recipe by the name operators use

WHAT IT IS

Two related conveniences. Ring is moving to show recipes and tanks by their *real* names — like the plant's own Dutch recipe name "SF 1" — instead of generic "Formula 1." And each tank's Batch Start screen offers one-tap

buttons for that tank's most-used recipes, so re-running the usual batch is a single tap.

WHY IT MATTERS

Operators think in the names they actually use, not in "Formula 3." Matching the screen to their language cuts mistakes and training time, and the recent-recipe buttons make the most common action — the same recipe on the same tank — a single tap instead of a fresh search every time.

HOW IT WORKS

A name resolver relabels the recipe picker in place using custom names stored in Ring's database (while keeping the underlying recipe number intact), and saved names hot-reload across the app. The Batch Start screen queries each tank's top three recipes from the last 30 days of history and shows them as quick-select buttons.

VERSUS THE OLD RS-360 SYSTEM

On RS-360, the recipe showed as an operator-entered name on the tank card and mixer panel ("Formula : "). So name display itself isn't brand new — this is a parity-plus-convenience improvement. Ring wires that resolver into the pickers and persists custom names, but the 2026-06-10 on-site audit still found literal "Formula N" on a couple of surfaces (**two P1 mismatches — the storage tank card and the mixer panel**). Later commits narrowed the gap, so present this as in-progress parity, not done.

STATUS & HONEST CAVEATS

Status: Live but partly limited. The picker relabeling, the name persistence, and the recent-recipe buttons are built and wired — but full name-parity across every surface is still in progress.

Watch-out: Be ready to name exactly which surfaces still show "Formula N": the **storage tank card and the mixer panel** (two P1 mismatches from the parity audit — not three). The pickers and recent-recipe buttons already resolve names. Also note the recent-recipe buttons only appear once a tank has batch history in the last 30 days; they're hidden when there's none.

LIKELY QUESTIONS & HOW TO ANSWER THEM

Q So does it show our real recipe names everywhere?

A Not on every single screen yet — I'll be specific: the picker and the recent-recipe buttons resolve real names today, but the parity audit found two spots, the storage tank card and the mixer panel, still showing "Formula N." Those are the in-progress items.

Q Why don't I see the quick-recipe buttons on a tank?

A They only show once that tank has batch history in the last 30 days, and they surface its top three recent recipes. On a tank that hasn't run recently they're simply hidden, which is intended.

Q Did the old system not show recipe names at all?

A It did, actually — it showed an operator-entered name on the tank and mixer screens. So this is parity plus the one-tap recent-recipe convenience, not a brand-new capability, and I'd rather state that honestly.

Q **Will renaming a recipe confuse the controller about which one ran?**

A No — the display name is just a label; the underlying recipe number is preserved, so the right recipe still runs and records regardless of what you call it on screen.

Split a batch to the right second tank, safely

WHAT IT IS

"Splitting" means sending part of a batch into a second storage tank instead of all into one. Ring lets the operator pick that second tank by its real name ("Split with Storage Tank 2/3/4"), and the confirmation dialog names the actual target tank — not a position in a list.

WHY IT MATTERS

If the operator picks a destination and the plant's overfill protection ends up guarding a *different* tank, you can overfill a tank with no high-level alarm watching it. Naming the real tank — and making the overfill protection follow that same tank — is a genuine safety fix, not a cosmetic one.

HOW IT WORKS

Each Batch Start screen has split radio buttons whose *label* (e.g. "Split with Storage Tank 3") is mapped to the **absolute tank number** (3), not the button's position in the list. That matters because the live controller program reads the written value as an absolute tank number, then drives that tank's fill valve and watches that tank's high-level alarm at the same absolute index. The dialog shows the absolute tank that will fill.

VERSUS THE OLD RS-360 SYSTEM

Important correction to make in your own words: do **not** say the legacy "wrote ambiguous indices" — that was a *Ring* pre-fix bug, not legacy behavior. RS-360 actually supported splitting cleanly (its report even printed "Batch gedeeld met Tank"), and in six months and 1,404 batches the plant **never used it** — every record shows splitting off. So this is parity-plus-safety, not a legacy regression. Ring now maps to the absolute tank number with overfill protection that follows the chosen tank — but, honestly, that fix still needs controls-engineer confirmation and a bench write-and-observe sign-off before split batching runs live.

STATUS & HONEST CAVEATS

Status: Built and wired, but gated — not live-proven. The absolute-tank mapping is in code, wired into the Batch Start screens, and confirmed correct against the live controller program; it is deliberately not cleared for live splitting yet.

Watch-out: This is **safety-critical and not field-proven**, so handle it carefully. The mapping is now confirmed correct against the live ladder logic — the earlier version, which wrote the radio button's *position* instead of the tank number, really would have guarded the wrong tank (that was the real P0 defect, and it's fixed). But the code itself still demands controls-engineer confirmation and a bench write-and-observe before split batching runs live. If a buyer asks "has split batching run live?" the answer is no — and the plant never used splitting in 1,404 historical batches anyway — so this is parity-plus-safety, gated, not proven on the line.

LIKELY QUESTIONS & HOW TO ANSWER THEM

Q Has split batching actually run on our plant?

A No — I won't claim it has. The fix is in code and confirmed correct against your live controller program, but it's deliberately gated until a controls engineer confirms it and we do a bench write-and-observe. And for context, the plant never used splitting in 1,404 historical batches, so nothing depends on it day one.

Q Your own notes mention a P0 bug here — what was it?

A The earlier version wrote the radio button's list position instead of the actual tank number, which meant the overfill protection could have watched the wrong tank. That's fixed now — Ring writes the absolute tank number, verified against the live ladder logic — but because it's safety-critical we're keeping it gated until it's signed off.

Q Wasn't the old system the one with the ambiguous tank problem?

A No, and I'd correct that if it came up — the ambiguous-index issue was a Ring bug we caught and fixed, not a legacy flaw. The old system supported splitting cleanly; it just was never used here.

Q How do I know the overfill alarm will watch the tank I picked?

A Because the controller drives the fill valve and reads the high-level alarm at the same absolute tank number Ring writes — that's confirmed against the live program. The remaining step is a bench sign-off to prove it end-to-end before it's enabled live, which is exactly why it's still gated.

Chapter 3 — Turn Plant Memory Into Money

Every shift your plant produces two things: glue, and a record of how it made that glue — which recipe ran, how much of each ingredient went in, what alarms fired, who was on shift. Call that second thing your *plant memory*. The problem with the old system was never that it failed to write any of it down. The problem is that it wrote it into a filing cabinet nobody could open. This chapter is about taking that memory out of the dead filing cabinet and turning it into two things an owner actually cares about: faster decisions, and visible money.

A few plain-language terms first, because the whole chapter leans on them. The **PLC** (Programmable Logic Controller) is the rugged industrial computer that physically runs the valves, pumps, and mixer — Ring does not change it and, on this site, is not even allowed to send it a command. The **HMI** (Human-Machine Interface) is the operator screen; Ring *is* the new HMI, replacing the 2002 DOS-era screen called RS-360. A **tag** is one named value living inside the PLC, like "Tank 2 temperature." A **batch** is one production run of one glue recipe. And **SQLite** is simply a single, open-standard database file that lives on the Ring PC — think of it as one tidy spreadsheet-style store you own outright, not a sealed proprietary vault.

Here is the honest version of the old situation, and you should lead with this honest version because a buyer who knows RS-360 can check it. RS-360 *did* store batch, step, shift, and alarm records — but in a Paradox/BDE format (an early-1990s database engine that has since gone dead) that needs a custom-built reader just to extract anything. There was no living historian recording values over time, no on-screen trends, no step-duration timing operators could use, no cost numbers of any kind, and no way for an operator to search it. The records existed; they were just stranded and silent. Ring's pitch is not "they recorded nothing" — it is "they recorded it into a dead format and could never use it; we keep that same memory alive and turn it into decisions and dollars."

One safety fact underpins everything below: all of these features read and store data on the Ring PC. None of them sends a command to the PLC. On top of that, Ring currently ships with a read-only safety gate switched on, which suppresses every PLC write. On-site Ring read all 133 live tags without issuing a single command that could change the process. (Reads do put request packets on the network — that is how any HMI asks the controller for a value — but not one of them was a write that could move a valve or change the process.) So whenever someone asks "could this analytics stuff touch my running plant?", the answer is a flat no.

See your spend and stop the leaks before they cost you

WHAT IT IS

A spend-and-leak dashboard for the owner. After a one-time setup, Ring shows you what each batch and each week of glue actually cost in dollars, and it highlights three specific ways money leaks: over-dosing (putting in more expensive ingredient than the recipe called for, which is pure giveaway), scrapped batches, and hold time. You open it by clicking the "Spend (this week)" card on Ring's main dashboard.

WHY IT MATTERS

Right now nobody at the plant can answer "what did this week of production cost us?" without a manual exercise. Over-dosing in particular is invisible money walking out the door — an operator who consistently runs an ingredient 5% rich is gifting a customer free product every batch, and no one sees it. Putting a dollar figure next to each batch turns a vague feeling that "we waste material" into a specific number you can chase down.

HOW IT WORKS

Ring's cost engine is a piece of deterministic math (it always gives the same answer for the same inputs — no estimating). For each batch it multiplies the *actual* amount of each ingredient used by that ingredient's *unit price as of the batch's date*, so a price change last spring doesn't rewrite last spring's costs. Crucially, it is built to never *under*-state spend: if an ingredient was used but you haven't entered a price for it, that batch is flagged as "not fully priced" and the missing ingredient is named, rather than silently treated as free. The figures come entirely from Ring's own local database — no PLC reads, no writes.

VERSUS THE OLD RS-360 SYSTEM

On cost, the contrast is total and fair: RS-360 had no cost engine of any kind. It never computed or retained cost, giveaway, or leak data — those questions simply went unanswered. This is one place where "the legacy did nothing" is literally true, because cost was never a feature of the old screen.

STATUS & HONEST CAVEATS

Status: Live and on a real screen — but the numbers stay blank until two one-time setup steps are done. The cost engine, the report, and the screen are all built, shipped, and reachable today. Watch-out: Two things to be ready for. First, **how you open it**: the real entry point is the "Spend (this week)" card on the dashboard, not a button in the Setup menu (that menu button is currently hidden). Demo it by clicking the dashboard card. Second, **it needs feeding**: the screen shows dashes and zeros until you (a) import the roughly 1,404-batch legacy history and (b) enter your current ingredient prices. Do not say the six months of spend analytics are there "on day one" or "out of the box" — say "after a one-time guided import of our ~6-month legacy history and a one-time price entry, the owner sees per-batch and per-week spend with price history preserved." Seed that data before you demo.

LIKELY QUESTIONS & HOW TO ANSWER THEM

Q Are these real numbers or estimates?

A They are real arithmetic — actual measured amounts times the price in effect on that batch's date — not a model or a guess. And if a price is missing, Ring refuses to fake it: it flags the batch as not fully priced and names the missing ingredient, so the total can never quietly understate what you spent.

Q Can I see this without entering all my prices first?

A You'll see the structure and any batches you have priced, but the headline spend numbers fill in only once current ingredient prices are entered. That's a deliberate honesty choice — it would rather show you a gap than invent a cost.

Q Does old history carry its old prices, or get repriced at today's rates?

A Each batch is costed at the price that was effective on its own date, because Ring keeps a dated price history. So last quarter's batches reflect last quarter's prices, which is what makes the trend meaningful.

Q **Could running this cost report disturb the plant?**

A No. It reads only Ring's local database and never contacts the PLC, and the read-only safety gate would block a write even if one were attempted.

Make faster calls with decision screens the legacy never had

WHAT IT IS

A set of seven brand-new "decision" screens that turn raw events into answers: an alarm-analytics screen (which alarms fire most and how long they take to handle), a tank CIP board (which tanks are clean, dirty, or in use — "CIP" means Clean-In-Place, the automated wash cycle), an alarm-to-batch correlation screen (which batch was running when a given alarm fired), a batch-queue planning board, a downtime/OEE screen (OEE = Overall Equipment Effectiveness, the standard score for how productively a line runs), a golden-batch overlay (compare a run against your best-ever "golden" batch), and a prioritized alarm summary.

WHY IT MATTERS

On the old screen, every one of these questions was answered by memory, paper, or guesswork. "Which alarm is killing us?" "Is Tank 4 actually washed and ready for a changeover?" "What was running when that fault hit?" These are the daily questions that decide whether a shift runs smoothly or chases its tail. Putting each on a dedicated screen turns tribal knowledge into something anyone can read.

HOW IT WORKS

All seven are read-only screens that draw from Ring's local SQLite database and never write the PLC. They compute familiar metrics — for alarms, things like average time-to-acknowledge and time-to-resolve, and a Pareto chart (the "80/20" bar chart that ranks the worst offenders first); for alarm handling, the ISA-101 and ISA-18.2 industry-standard layouts that let an operator "shelve" a known nuisance alarm. Each opens through the Reports area of the navigation bar.

VERSUS THE OLD RS-360 SYSTEM

RS-360 had a single alarm-history query form and basic printouts — and nothing else in this category. No analytics, no OEE, no CIP board, no batch-queue planner, no golden-batch overlay, no alarm shelving. So these seven are genuinely net-new decision tools, not reworked old ones.

STATUS & HONEST CAVEATS

Status: Live and on a real screen. All seven exist, are reachable from the Reports menu, and confirmed never to write the PLC. Watch-out: Two honest points. First, the marketing copy says "eight screens" — that count double-counts the Tank History viewer, which is the same screen credited to the live-trending feature below. The honest number here is **seven distinct screens**; say seven. Second, two of the seven are empty until data accrues: the Downtime/OEE screen needs operators to start logging downtime events, and the Golden-Batch Overlay needs the historian to accumulate some temperature traces first. Don't demo those two cold on a fresh install — demo the alarm analytics, CIP board, and correlation screens, which work as soon as there's event history.

LIKELY QUESTIONS & HOW TO ANSWER THEM

Q Do any of these screens send anything to the controller?

A No. They are all read-only and pull from Ring's local database. One of them is a batch-queue *planner*, and even that was specifically checked to confirm it writes nothing to the PLC.

Q You said eight screens, I count seven — what gives?

A Fair catch. There are seven distinct new analytics screens here; the eighth in the brochure is the Tank History viewer, which I also count under live trending — it got listed twice. Seven is the honest number.

Q Why are the OEE and golden-batch screens blank on your demo machine?

A Because they're driven by data that has to accumulate — operators logging downtime, and the historian collecting temperature traces. On a system that's been running, they fill in; on a fresh box they start empty, which is honest rather than showing fake data.

Q Is the alarm shelving going to let someone hide a real safety alarm?

A Shelving follows the ISA-18.2 standard, which is the recognized industrial practice for temporarily quieting a known nuisance alarm with a record of who did it — it's a managed feature, not a way to make safety alarms disappear unaccountably.

Watch trends live and review any tank's last 30 days

WHAT IT IS

Two related tools. The first is a live trend chart: you pick PLC values you care about and watch them plot in real time on a rolling graph, complete with markers showing when batches started and alarms fired, and a one-click export to CSV (a plain comma-separated text file any spreadsheet opens). The second is a Tank History viewer that lets you look back over each tank's temperature and level for the past 30 days.

WHY IT MATTERS

On the old screen, a number appeared, updated, and was gone — if you wanted to know whether a temperature was climbing or a level was drifting, you had to be staring at it at the right moment. Engineers needed the separate Studio 5000 programming software just to watch a trend, which most operators can't and shouldn't open. Letting anyone watch a live trend or scroll back 30 days means a question gets answered in seconds instead of after the evidence has already scrolled away.

HOW IT WORKS

The live trend screen opens as its own window and quietly polls the chosen live values every two seconds, drawing them on a custom chart. It keeps a rolling eight-hour buffer, lets you zoom from five minutes to eight hours, hover for exact values, and export to CSV — with no charting add-on and no PLC writes. The Tank History viewer renders the per-minute temperature and level series that Ring's historian (described later in this chapter) has been quietly recording, across a 30-day window.

VERSUS THE OLD RS-360 SYSTEM

RS-360 had no live time-series trend plot at all. The thing that looked like a graph in the old system — its ScreenGraphForm — is a static, configuration-driven mimic/graph builder, not a moving chart of values over time, and there was no per-tank history viewer either. The legacy parity audit simply lists trending as Missing. So this is genuinely new ground, not a tidied-up old graph.

STATUS & HONEST CAVEATS

Status: Live and on a real screen. Both the live trend window and the Tank History viewer are shipped and reachable from the navigation bar. Watch-out: Live trending works immediately whenever Ring is connected to the PLC. The Tank History viewer, though, only shows data the historian has actually recorded since deployment — there is no backfill, so on a freshly installed system the last-30-days view starts empty and fills in as the plant runs. Frame Tank History as "starts recording the day we go live," not "instantly shows the last month." (Also, if a sharp reviewer checks the cited line counts, note the "939 lines" refers to the code-behind file, not the screen layout file — a citation nit, not a capability claim.)

LIKELY QUESTIONS & HOW TO ANSWER THEM

Q Don't I need Studio 5000 or an engineer to see a trend?

A No — that's the whole point. Any authorized operator picks the values and watches them plot, right inside Ring. The engineering software stays in the engineer's hands; the trend doesn't require it.

Q Can I get the trend data out to analyze it myself?

A Yes, one click exports to CSV, which opens directly in Excel or any analysis tool. Your trend data isn't trapped in the screen.

Q Why is the 30-day tank history empty right now?

A Because the historian only records going forward from install — there's no retroactive backfill of values that were never captured. It starts filling the moment we deploy, so within a month you have the full rolling window.

Q Is watching live values adding load to my controller?

A It's a light read every couple of seconds while the window is open, and the loop shuts itself off when the last trend window closes. It reads only; it never writes the PLC.

Catch a failing pump before it scraps a batch

WHAT IT IS

A quiet watchdog on your feed equipment. Ring measures how fast each ingredient is actually being fed and compares it to that ingredient's normal pace. When a pump or line drifts noticeably off its baseline, Ring raises an

early warning — in plain words like "starch feed is running 18% slower than usual — check the pump/line" — so maintenance can look before that drift turns into an alarm, a scrapped batch, or a line stoppage.

WHY IT MATTERS

A pump rarely fails all at once; it slows down first. On the old system that early slowdown was invisible — you only found out when a batch went out of spec or the line tripped. Surfacing the drift while it's still a 15% slowdown turns a future emergency into a scheduled five-minute check, which is the difference between planned maintenance and scrapped product.

HOW IT WORKS

For each recipe-and-ingredient pair, Ring keeps a rolling baseline (the median feed rate of the last ten batches) and flags when a batch runs more than 15% off that baseline — but only when the drift holds across two batches in a row, so a single fluke doesn't cry wolf. The warning appears on the dashboard's Equipment Health card. It is also routed into Ring's notification rules so it *can* be included in a daily email digest.

VERSUS THE OLD RS-360 SYSTEM

RS-360 never computed feed-rate drift or surfaced any predictive warning — it offered no early signal of equipment trouble at all. (To be precise and fair, the legacy did store step records in its dead Paradox files, but it never turned them into a feed-rate baseline or a warning anyone could act on.)

STATUS & HONEST CAVEATS

Status: Live and on a real screen — the monitor and its dashboard card are built and wired — but it is deliberately silent until it has enough history to be trustworthy. Watch-out: This is the most "needs context" feature in the chapter, so be ready. First, it is **baseline-honest**: it flags nothing until roughly ten prior *completed, Ring-recorded* batches exist for a given recipe/ingredient, with step timing captured. The legacy import does not backfill that step timing, so on a fresh install the Equipment Health card correctly reads "all monitored equipment nominal" — that's the feature working honestly, not broken. Lead the demo with the dashboard card and explain it earns its baseline over the first weeks. Second, **email is plumbing-only**: the routing exists in code as a daily digest, but actual delivery needs an email server configured, so do not promise alerts will land in an inbox today. Say "it shows on the dashboard, and email delivery is wired and ready to switch on once we configure the mail server."

LIKELY QUESTIONS & HOW TO ANSWER THEM

Q Will this warn me on day one?

A No, and that's by design. It needs about ten completed batches per recipe/ingredient to learn what "normal" looks like before it dares to flag drift. On a fresh install it stays quiet — it would rather say nothing than fire false alarms off a baseline it hasn't earned.

Q **Is this going to flood operators with false alarms?**

A It's tuned against that: it only flags drift past 15% and only when that drift persists across two batches in a row, so a single odd reading is ignored.

Q **Do I get an email when a pump drifts?**

A The dashboard card is the reliable surface today. The email path is built — it's set up as a daily digest, not an instant alert — but it needs our mail server configured before anything actually sends, so I wouldn't lean on email until we've switched that on.

Q **Is this reading viscosity or some new sensor I'd have to install?**

A No new hardware. It's derived purely from how fast ingredients are already being fed during normal batching, using data Ring already records.

Get every report you rely on, from one database, with PDF

WHAT IT IS

The full set of printable reports operators and supervisors depend on — what was batched, how much was used, batch and alarm history, recipes, inventory, and per-shift consumption — now with modern print preview and real PDF output, all pulled from one local database.

WHY IT MATTERS

Quality audits, supervisor handoffs, and management reviews all run on these reports. The value here isn't a flashy new capability; it's that the reports you already rely on survive the move off the dead engine, gain PDF and date/tank filtering, and come from a single queryable store instead of a brittle 1990s file format.

HOW IT WORKS

Every report is backed by Ring's SQLite database and hosted inside the Reports area. Reports render as an on-screen preview you can print, save to PDF (via a real, properly-licensed PDF exporter — not a stub), or export to CSV, and several support date and tank filters. The per-shift consumption report even correctly credits an overnight shift's usage back to the prior day's shift, matching how the floor actually counts it.

VERSUS THE OLD RS-360 SYSTEM

RS-360's reports were trapped in printouts over the dead Paradox/BDE engine, and several backing tables — inventory and shift data — were effectively abandoned or readable only through a custom extraction tool. There was no PDF and no single queryable store. Ring gives you the same report set from one open database, with print preview, PDF, CSV, and filtering, plus revived shift-consumption and inventory history the old system left stranded.

STATUS & HONEST CAVEATS

Status: Live and on a real screen. The reports exist, are reachable from the Reports menu, and nine of them wire the real PDF exporter. Watch-out: The honest dependency is that a batch-history report is only as complete as the data behind it. To show "six months" of batch history, the roughly 1,404-batch legacy set has to be imported first; out of the box the batch table starts empty. So phrase it as "every report you rely on, against whatever history is loaded — including the full legacy set once it's imported," not "six months of history built in." (The detailed legacy claims about Inventory.DB and Shift.DB lean on the parity audit document, which is a fair source but wasn't re-opened in this pass.)

LIKELY QUESTIONS & HOW TO ANSWER THEM

Q Do I keep all the reports my operators use today?

A Yes — the dependable set carries over: batched amounts, usage, batch and alarm history, recipes, inventory, and per-shift consumption. The goal was no report left behind, plus PDF and filtering on top.

Q Is the PDF real, or does it just print to a printer?

A It's a real PDF export through a proper, openly-licensed PDF library, used by nine of the report screens — you get an actual file you can email or archive, not just a print job.

Q Will the batch report show my last six months automatically?

A It will show whatever batch history is loaded. To get the full six-month legacy picture, we run a one-time import of the ~1,404 legacy batches; after that, the report spans the whole set and is schedulable.

Q Where does the report data come from — is it touching the PLC?

A It comes from Ring's single local SQLite database. Reports never read or write the PLC.

Answer "show me our data" in one click — you own it

WHAT IT IS

A self-serve data exporter. When an auditor or manager says "show me our data," you pick a table, choose which columns and an optional date range you want, and save it to a CSV file — a standards-compliant plain-text file that opens directly in Excel. Your operational data is portable and yours.

WHY IT MATTERS

The deepest fear with any plant software is that your own records get held hostage by the vendor's format. This feature is the antidote: anyone authorized can pull any table out to an open file in seconds, with no special tools and no dependence on a fragile engine. It's the practical proof of "you own your data."

HOW IT WORKS

A simple three-step dialog — pick table, pick columns, set an optional date range — writes out an RFC-4180 CSV (RFC-4180 is just the published standard for how a proper comma-separated file is formatted, so it imports cleanly everywhere). It's reachable from the Reports menu and reads only from Ring's database.

VERSUS THE OLD RS-360 SYSTEM

RS-360 had an export equivalent, but the underlying data was locked inside the dead Paradox/BDE engine and hard to extract once that engine aged out. Ring exports any table from an open SQLite store, so the data stays portable and stays yours rather than aging into inaccessibility.

STATUS & HONEST CAVEATS

Status: Live and on a real screen. The export dialog is shipped and reachable from the Reports menu. Watch-out: This is a real grilling risk in the brochure. The marketing line says "CSV or Excel file" — but the code exports **CSV only**. There is no native Excel/.xlsx export path (the Excel library was deliberately removed). The corrected, true claim is: "pick a table, choose columns and an optional date range, and save your data to a standards-compliant CSV file." If a buyer asks for Excel, do not claim it — say "we export CSV, which opens directly in Excel as a spreadsheet." Get that one word fixed in the deck before the meeting.

LIKELY QUESTIONS & HOW TO ANSWER THEM

Q Can I export straight to an Excel .xlsx file?

A We export to CSV, which Excel opens directly as a spreadsheet — double-click and it's there. There isn't a native .xlsx writer; we standardized on CSV deliberately because it's a universal, dependency-free format.

Q How do I know I'm not locked into your software?

A Because the data lives in an open SQLite database and this one-click export pulls any table out to a standard CSV file. Nothing about your records depends on Ring continuing to exist — that's the design intent.

Q Can I limit an export to a date range or just certain columns?

A Yes — you choose the table, the specific columns, and an optional date range before exporting, so an auditor gets exactly the slice they asked for and nothing more.

Q Does exporting touch the live process?

A No. It reads only the local database; it never contacts the PLC.

Never trust a frozen number: live, stale, or off-target at a glance

WHAT IT IS

A set of plain-language trust indicators on your readings. Ring shows whether a value is genuinely live or has gone stale, flags temperatures and levels that are off target, and shows how far a batch over- or under-delivered against its recipe. On top of that, any tile or pill whose data source goes quiet automatically fades — dimming toward 45% opacity — and shows an "as-of" timestamp tooltip, so a number that's secretly frozen can't keep looking confidently live.

WHY IT MATTERS

The single most dangerous thing an HMI can do is keep displaying a confident-looking number after the connection has quietly dropped — the operator acts on a temperature that's actually ten minutes old. These indicators make staleness *visible*, so during a comms blip nobody makes a decision on a dead reading. That's a safety feature dressed as a UI detail.

HOW IT WORKS

A small library of badges and chips — link state, tank attention, time-to-empty, health, batch variance, alarm triage — is wired into the live screens. A "freshness" converter set is registered globally and bound to the opacity and tooltip of the dashboard's key tiles and the status strip, using each value's last-sample timestamp. The clever part: a once-per-second internal tick keeps re-checking the timestamp *even when the PLC link has gone silent*, so a frozen tile genuinely keeps dimming instead of freezing bright. The staleness threshold is ten seconds.

VERSUS THE OLD RS-360 SYSTEM

RS-360 showed raw numbers with no sense of whether they were live, healthy, or on-target. A comms blip left a confident-looking but dead reading on screen with zero indication anything was wrong. Ring attaches link state, health, and batch variance to readings and fades any stale tile or pill toward 45% opacity with an as-of timestamp — so a frozen number announces itself instead of fooling you.

STATUS & HONEST CAVEATS

> Status: Live and on a real screen. The badges, chips, and the freshness auto-dim are all wired and working today — live on the dashboard KPI tiles and the status strip. (An earlier internal note called the auto-dim "dormant/v1.1 polish"; that note is out of date — the wiring has since been done and verified, so don't repeat the "dormant" framing.) Watch-out: Two precision points so you don't over-claim. The auto-dim is live on the dashboard KPI tiles and the status strip; the window-chrome surface is only declared, not actually consuming the dimming, so name two surfaces, not three. And one small badge, the "Tank Attention" strip, is a duplicate of an existing dashboard tile and is slated to be pulled — the deck already flags that.

LIKELY QUESTIONS & HOW TO ANSWER THEM

Q If my network blips, will a stale number just sit there looking live?

A No — that exact failure is what this guards against. A once-a-second internal tick keeps re-evaluating the reading's age even when the PLC has gone silent, so the tile visibly fades toward 45% and shows an "as-of" timestamp. It can't keep looking fresh while it's frozen.

Q How stale is "stale"?

A Ten seconds. Past that without a fresh sample, the affected tiles dim and surface their last-good timestamp.

Q Which screens actually do this, versus just claiming to?

A The freshness auto-dim is live and consumed on the dashboard KPI tiles and the status strip. I won't claim the window frame does it — that surface declares the capability but doesn't currently use it, so I list two surfaces, not three.

Q Is "batch variance" telling me something the operator can act on?

A Yes — it shows how far a batch over- or under-delivered against its recipe target, so an operator immediately sees a run drifting rich or lean rather than discovering it after the fact.

Keep a true history — automatically, with zero extra load on the controller

WHAT IT IS

The quiet recorder underneath everything else in this chapter. About once a minute, Ring writes down each tank's temperature, level, and current recipe, building a real, continuous history you can chart, compare, and cost later. It is the memory that powers live trending, the golden-batch comparison, and the cost engine — and it does it without adding any new traffic to the PLC.

WHY IT MATTERS

You can't trend, compare, or cost what you never recorded. The old system had no living historian, so the moment a value left the screen it was gone. This service is what makes "plant memory" real: it captures the raw material that every analytics feature in this chapter feeds on, automatically and honestly.

HOW IT WORKS

A background timer fires about every 60 seconds and reads the snapshot of tank values that Ring *already holds in memory* for the operator screen — so it adds zero new reads to the PLC — then writes one row per installed tank into the SQLite database in a single batched transaction. It keeps a 30-day rolling window, pruning older data hourly. It is built to be honest about gaps: it won't record while the link is classified as dead (so a frozen connection never writes a row of fake flat data), and it skips empty tank slots rather than littering the history with blank ghost rows. The history table is indexed so charts pull quickly.

VERSUS THE OLD RS-360 SYSTEM

RS-360 kept some records in its dead Paradox files but had no continuous per-tank historian and no way to trend or do statistical process control (SPC — charting a value over time to spot drift) over historical process values; its text logs were purged unread. Ring captures a per-tank temperature/level/recipe row roughly once a minute into an indexed, open database with 30-day retention and honest gap handling, and it does so with no extra PLC traffic.

STATUS & HONEST CAVEATS

Status: Live and on a real screen — actually running. The historian is started when Ring launches and is the verified backbone behind trending, golden-batch overlay, and cost trends. Watch-out: Two honest framings. It's a **30-day rolling** window, not a permanent archive — older data is pruned, by design. And there is **no historical backfill** from this service: it records going forward from the day you deploy, so the downstream views (golden-batch overlay, cost trends) fill in over the first weeks rather than appearing complete on install. Your legacy batches come in through the separate one-time import, not from this recorder.

LIKELY QUESTIONS & HOW TO ANSWER THEM

Q You're reading the tanks every minute — doesn't that load my controller?

A No. It reuses the snapshot Ring already has in memory for the operator screen, so it performs zero new PLC reads. The recording is essentially free from the controller's point of view.

Q What stops it from recording garbage when the connection drops?

A It refuses to write while the link is classified as dead, so a frozen connection never lays down a row of fake flat values. A gap in the history is shown as a gap, not faked.

Q How far back does it keep data?

A Thirty days, on a rolling basis — older rows are pruned hourly. It's a working window for trending and comparison, not a permanent archive; for long-term retention we'd export or extend retention deliberately.

Q Does it record the recipe step too?

A It records temperature, level, and which recipe is running. It does *not* record the individual step, because the tank snapshot it reads doesn't surface step per tank — and rather than invent one, it leaves that field empty. Honest blank over a fabricated value.

Q If it's recording all the time, can it ever interfere with a batch?

A No. It only ever writes to Ring's local database; it never sends anything to the PLC, and the read-only safety gate would block a write regardless.

Chapter 4 — Smarter Alarms, Faster Fixes

A few terms first, because the whole chapter rests on them. The **PLC** (Programmable Logic Controller) is the rugged industrial computer that physically runs the plant — it opens valves, starts pumps, and drives the mixer. The **HMI** (Human-Machine Interface) is the operator's screen; it shows what the PLC is doing and lets the operator press buttons. **Ring is the new HMI.** It is a modern Windows program that replaces the plant's 24-year-old DOS-era screen ("RS-360"), but it talks to the *exact same untouched PLC*. Ring does not change how the plant runs; it changes what the operator sees and how fast they can react.

An **alarm** is the PLC's way of shouting "something is wrong" — a tank too hot, an emergency-stop pressed, a batch stuck on hold. Inside the PLC each alarm is just a number (an "Alarm Number," for example #4 or #831). On the floor, the real cost of a fault is not the fault itself — it is the minutes lost while an operator figures out *what the number means, which screen fixes it, and what to do first*. On a corrugating-adhesive (industrial glue) starch line that runs around the clock, those lost minutes are wasted batches and idle production.

This chapter covers eleven things Ring does to shrink that gap: it shows every alarm in the controller's own words, links each one to the screen that fixes it, offers a "first thing to check" card, sorts the urgent from the handled at a glance, flags a stalled batch right on the dashboard, shows how long a fault has gone unanswered, raises a hard-to-ignore pop-up, guards the three process buttons (Hold/Resume/Reset) with a confirmation, ties downtime back to the exact batch that was running, and stays correct on a screen left on for days. One of the eleven is a deliberately honest "status" item — a tracked alarm-coverage gap — rather than a finished feature, and one widely-pitched pop-up behavior is built but not yet switched on. All of that is spelled out below so nothing surprises you in the room.

One framing point an outsider should grasp up front: Ring currently ships with a **read-only safety gate** turned on (a setting called `ReadOnlyMode`). While that gate is on, Ring *suppresses every command that could change the process* — it can read the controller but cannot write to it. (Reading still sends small network request packets over the wire, so the honest phrasing is "zero write commands," never "zero bytes.") That gate is the backdrop for several features below.

Every alarm reads exactly like the controller says it

WHAT IT IS

When a fault fires, Ring shows the operator the same number and the same plain description the PLC itself uses — for example "E-STOP has Been Pressed" or "Make Ready Tank Temperature Too High." There is no homemade label sitting between the controller and the operator that could say something different from what the machine actually means.

WHY IT MATTERS

In the middle of a fault, an operator's worst enemy is doubt about what the alarm even means. If the screen's wording drifts from the controller's wording, people hesitate, second-guess, or act on the wrong thing. Matching the controller bit-for-bit removes that doubt so the operator trusts the alarm and acts.

HOW IT WORKS

Ring keeps a small built-in dictionary (the "alarm catalog," seeded in code) that maps each Alarm Number to its real text and a severity letter — **A** for Alarm/critical, **W** for Warning. The PLC also encodes an alarm's *state* by adding an offset to the number: the base number means "Active," base **+1000** means "Silenced," and **+2000** means "Resolved." Ring strips that offset back off to recover the true alarm and its state. Those offset rules were checked against the controller's own logic and confirmed against live plant data read on-site, so what Ring shows lines up with both the controller and the old HMI.

VERSUS THE OLD RS-360 SYSTEM

RS-360 stored alarm wording as positional Dutch text in flat files, riddled with gaps and blanks — entries 30/31 literally read "Thermische veiligheid ??", 97/98 were blank, and alarm #4 had no entry at all, so operators just saw the bare code "A0004." Ring seeds the controller's active alarm catalog (76 numbers today) with real descriptions and decodes the same +1000/+2000 state bands the controller uses, even down to preserving a legacy quirk for exact parity.

STATUS & HONEST CAVEATS

Status: Live and on a real screen. Every covered alarm renders with the controller's own number, text, and severity on the Process > Alarms screen, with a preview on the dashboard. Watch-out: Do not call this the "full" catalog. It covers the controller's active catalog — exactly 76 numbers today. The plant's full Dutch legend has roughly 94 more numbers Ring has not seeded yet (that is the tracked "coverage gap" item later in this chapter). Say "76 numbers today," not "all alarms."

LIKELY QUESTIONS & HOW TO ANSWER THEM

Q How do I know the wording isn't just made up to look good?

A It is not made up — it is seeded from the controller's catalog and the decoding of the +1000/+2000 state offsets was independently confirmed against live tag data we read on the plant's own PLC. The text matches both the controller and the old HMI.

Q So does it show *every* alarm the plant can throw?

A It correctly shows the 76 numbers in the controller's active catalog today. The plant's older Dutch legend has about 94 additional numbers we have not seeded yet; that is a known, tracked task, and I cover it openly later in this chapter.

Q What was actually wrong with the old labels?

A The legacy text had real gaps — two entries read "Thermische veiligheid ??", two were blank, and alarm #4 showed only the raw code "A0004" because it had no description. Ring shows a real description for that one.

Q Could reading the catalog ever change something in the PLC?

A No. The catalog is a fixed list inside Ring; it is read-only reference data and touches no PLC tag at all.

One tap from the alarm to the screen that fixes it

WHAT IT IS

Each alarm row offers a single button that jumps straight to the screen where you actually deal with that fault — for example "Go to Storage Tank 2," "Go to Make Ready Tank," or "Go to TVC" (the temperature-control area). The operator does not need to memorize the plant's screen map.

WHY IT MATTERS

On the old system, knowing *which* screen fixed a given fault was tribal knowledge — fine for a veteran, slow for a new hire at 2 a.m. A one-tap link turns "where do I even go for this?" into a reflex, dropping the time from fault to fix to seconds.

HOW IT WORKS

A small lookup (a "link resolver") maps each decoded alarm number to a target screen and a label. Storage-tank alarms resolve to the right tank by its number; Make Ready Tank faults go to the MRT screen; certain temperature-control alarms go to TVC. The link is pure read-only information inside Ring — it sends nothing to the PLC — and the button is coded so that tapping it doesn't accidentally trigger the row's other behavior.

VERSUS THE OLD RS-360 SYSTEM

RS-360's alarms were inert text on the alarm form. To fix something you had to already know which area screen to open and navigate there by hand. Ring puts the right destination one tap away.

STATUS & HONEST CAVEATS

> Status: Live and on a real screen. The "Go to ..." links appear on the dashboard alarm-preview rows and navigate correctly. Watch-out: Plant-wide alarms — E-STOP, PLC battery, the silo, borax/caustic, corrugator pump 811 — deliberately have no link. If someone asks "why doesn't E-STOP have a Go-to button?", the honest answer is that no single screen fixes a plant-wide fault, so sending the operator to one screen would be misleading.

LIKELY QUESTIONS & HOW TO ANSWER THEM

Q Why doesn't E-STOP get a "Go to" button?

A On purpose. An emergency stop or a PLC-battery fault isn't fixed on one area screen, so we intentionally leave those with no link rather than send the operator somewhere unhelpful. Tank and Make-Ready faults — which *do* map to one screen — get the link.

Q Does tapping the link send any command to the controller?

A No. It's pure on-screen navigation using read-only reference data; no PLC tag is written.

Q Does the link know *which* tank, or just the area?

A It resolves to the specific tank — the storage-tank alarm numbers carry the tank position, so "Storage Tank 2" lands on Tank 2, not just the tank group.

Q **What if a new operator taps it and lands on the wrong screen?**

A For the covered alarms it lands on the correct screen by design; the only alarms without a link are the plant-wide ones, where we chose to show no button rather than a wrong one.

A built-in "first thing to check" card for priority alarms

WHAT IT IS

For a set of priority alarms, Ring offers plain-language first-response guidance — what the alarm means, the first thing to check, and who to call — without the operator hunting through a paper manual. Alarms that don't have a card yet point the operator to the manual, and a supervisor can add or edit cards over time.

WHY IT MATTERS

The slowest moment in a fault is a green operator paging through a binder. For the highest-priority alarms, having "first check: ..." right on the screen lets a brand-new operator respond like a veteran, and lets supervisors keep raising the team's floor by writing better cards.

HOW IT WORKS

Ring ships a small built-in file of playbook entries keyed by alarm number, and a database override so a supervisor's edited card wins over the shipped default. Two surfaces use it: hovering a covered alarm on the dashboard shows the *first check* line in the tooltip, and **selecting** that alarm row on the Process > Alarms screen opens the full card — meaning, first check, and who to call — in a dedicated panel. A supervisor editor (Setup > Alarm Playbook) lets you add or override entries for any alarm number. The whole feature is observational: it never touches the PLC, the alarm queue, or the silence/acknowledge path.

VERSUS THE OLD RS-360 SYSTEM

RS-360 left operators on their own for most faults, with at most static "possible causes" text on a couple of hardcoded pop-ups. Ring ships a curated first-response playbook (meaning / first check / who to call) for about 10 priority alarms today, with a supervisor editor to add or override entries for any alarm number.

STATUS & HONEST CAVEATS

Status: Live and on a real screen, scoped to roughly 10 priority alarms. It is wired on two surfaces (dashboard hover and the Alarm screen's playbook panel). Watch-out: Do not say "every alarm has a card" or "catalog-wide." Exactly 10 alarms have a card today; for the rest, the screen shows "No playbook entry; check the manual." Also be precise about the two surfaces: the *hover tooltip shows the first-check line only*, while the *full card (meaning / first check / who to call) appears when you SELECT the alarm row* on the Process > Alarms screen — don't promise the whole card on hover. Finally, the card wording is operator-facing text that an engineer should still sign off; it has not yet been plant-reviewed.

LIKELY QUESTIONS & HOW TO ANSWER THEM

Q Does every alarm have one of these cards?

A No — about 10 priority alarms have a card today. For any other alarm, the screen honestly says "check the manual." A supervisor can author cards for the rest over time, and that's the intended path to growing coverage.

Q You said "hover and it tells you everything" — does it?

A Slight simplification on my part. Hovering shows the *first check* line. The full card — meaning, first check, and who to call — appears when you select the alarm row on the Alarm screen. Both are real; they just live on two surfaces.

Q Who wrote the guidance, and is it trustworthy?

A We shipped sensible starter content, but it has not yet been reviewed by your process engineers. The right move before go-live is to have your team review and edit the cards — which the supervisor editor is built for.

Q Can an operator break something by reading a card?

A No. The playbook is read-only guidance; it never touches the PLC, the alarm queue, or the silence/acknowledge path.

Q What happens for an alarm with no card?

A It shows "No playbook entry; check the manual." — an honest fallback, not a blank or a guess.

Tell a live critical from a handled warning at a glance

WHAT IT IS

Every alarm row shows two things instantly: how serious it is (Critical / Warning / Info) and where it is in its life (Active / Silenced / Resolved). Crucially, the state is shown by a distinct *shape* as well as a color, so a color-blind operator reads the same answer just as fast.

WHY IT MATTERS

On a monochrome list, every line looks equally urgent, so operators waste attention re-reading handled faults or — worse — mistake a long-resolved warning for a live emergency. Encoding urgency and state visually lets an operator triage from across the room.

HOW IT WORKS

Ring maps the catalog severity letter to a tier (A = Critical/red, W = Warning/amber) and downgrades already-acknowledged or silenced rows to Info. State is drawn as a shape, never color alone: Active is a filled triangle, Silenced a half-circle, Resolved a check mark. Resolved and silenced rows are dimmed, and the list sorts Active

alarms to the top (newest first), then Silenced, then Resolved. A small cache avoids re-reading the database on every 2-second refresh.

VERSUS THE OLD RS-360 SYSTEM

RS-360 forced operators to read every line equally — a monochrome DOS list with no severity tiers, no lifecycle states, and no color-blind affordance. (An earlier Ring build briefly painted every row an identical red "Alarm"; that's already fixed.) You now see severity and state at a glance as color plus shape, with resolved rows dimmed to muted info chips instead of staying red.

STATUS & HONEST CAVEATS

Status: Live and on a real screen, and color-blind-safe (shape and color both encode state). Watch-out: The honest before/after openly mentions that an *earlier* Ring build wrongly colored every row red. If that comes up, confirm it is already fixed — the current build dims resolved/silenced rows and uses three distinct glyphs.

LIKELY QUESTIONS & HOW TO ANSWER THEM

Q Is this just color? My lead operator is color-blind.

A It is color *and* shape. Active is a filled triangle, Silenced a half-circle, Resolved a check mark — so the state is readable without relying on color at all.

Q I heard an earlier version made everything red — is that still the case?

A That was an earlier build and it's fixed. The current version dims handled rows and uses the three distinct glyphs, so a live critical and a resolved warning never look the same.

Q Where does "Critical" vs "Warning" come from — is it your opinion?

A It comes straight from the controller catalog's severity letter (A = Critical, W = Warning), not from us. Acknowledged or silenced rows drop to Info so handled faults don't keep screaming.

Q Does recomputing this every couple of seconds slow the screen down?

A No — there's a small cache so it doesn't re-open the database on every refresh tick.

An "ON HOLD" flag on the dashboard the moment a batch stalls

WHAT IT IS

If the running batch goes on hold — held for inspection, held too long, or running too slowly — an amber "ON HOLD" chip with the reason appears right on the dashboard's active-batch banner, and the time-remaining reads "(paused)" instead of pretending the batch is still moving forward.

WHY IT MATTERS

The expensive failure mode is a batch that's quietly stalled while the screen still shows a progress bar creeping along — nobody notices for minutes. Surfacing the hold and its reason on the main screen catches a stall early, before it costs a batch.

HOW IT WORKS

Ring watches the active alarms for the "hold" family (held for inspection, held too long, progress too slow), picks the most-escalated one, and shows its description in the amber chip on the hero banner. The estimated-time-remaining gains a "(paused)" tag while the hold is active. It is purely a re-presentation of the alarm data Ring already reads every 2 seconds — no new PLC reads.

VERSUS THE OLD RS-360 SYSTEM

On the live batch we observed, the legacy/early-Ring hero kept showing normal progress while the batch actually sat on hold for about 12 minutes — RS-360 had no hero hold chip tying the hold alarm to the running batch. Ring now flags the hold and the reason on the dashboard and stops the progress bar from faking forward motion.

STATUS & HONEST CAVEATS

> Status: Live and on a real screen. The amber ON HOLD chip and the "(paused)" suffix appear on the dashboard hero banner. Watch-out: The "~12 minutes on hold while the hero showed normal progress" figure is real captured telemetry (289 samples), not a guess — cite the dashboard UX roadmap if challenged.

LIKELY QUESTIONS & HOW TO ANSWER THEM

Q Is the "12 minutes lost" a marketing number or did you actually measure it?

A Measured. It comes from 289 real telemetry samples we captured from the live batch, where the batch sat on hold while the old hero still showed normal progress. It's documented, not estimated.

Q Does this add load to the PLC by polling more often?

A No. It reuses the alarm snapshot Ring already reads every 2 seconds and just re-presents it on the dashboard — no extra PLC reads.

Q Which holds does it catch?

A The hold family: held for batch inspection, held too long, and progress too slow. If more than one is active it shows the most serious one.

Q Why does the time-left say "(paused)"?

A Because during a hold the batch isn't progressing, so a ticking estimate would be a lie. The "(paused)" tag keeps the screen honest about the stall.

See how long a fault has been sitting unhandled

WHAT IT IS

Each alarm shows how long it has been going — "4m", "1h 12m" — and the panel header calls out the single oldest unhandled (still-active) alarm. That "oldest active" line is the key "is anyone actually dealing with this?" signal for a supervisor.

WHY IT MATTERS

A fault that's been ignored for an hour is a very different problem from one that fired ten seconds ago, but the old screen made them look identical. Showing age — and especially the oldest unhandled alarm — lets a supervisor catch the one that fell through the cracks.

HOW IT WORKS

Ring records when each alarm was raised and renders a compact age ("<1m", "Nm", "Nh Nm"), recomputed on the existing 2-second refresh with no extra timer. The "oldest active" header scans the full set of active alarms against the live clock and shows the single oldest one's age; it stays blank when the raise time is unknown, so an age is never invented.

VERSUS THE OLD RS-360 SYSTEM

RS-360 stamped an alarm's time but had no live, counting-up relative age and no "oldest active" callout — supervisors were left eyeballing timestamps to guess how stale a fault was.

STATUS & HONEST CAVEATS

Status: Live and on a real screen. Both the per-row age and the "oldest active" header are wired on the dashboard and the Alarm screen. Watch-out: Be precise about "live." The **oldest-active caption recomputes from the live clock every refresh, so it's always fresh.** An **individual row's age caption can briefly lag by one refresh period** (on both the dashboard preview and the Alarm grid) when the alarm set is otherwise unchanged — a deliberate, documented cosmetic tradeoff that keeps the operator's selection and scroll position from jumping during triage, and it self-heals on the next real alarm change. So "live age" on a single row is slightly optimistic wording; the headline oldest-active number is the always-fresh one.

LIKELY QUESTIONS & HOW TO ANSWER THEM

Q **Is the age truly live, or does it freeze?**

A The headline "oldest active" age is recomputed against the live clock every refresh — always fresh. A single row's age caption can lag by one refresh cycle when nothing else about the alarm set changed, because we deliberately avoid rebuilding the list (which would throw away the operator's selection and scroll). It catches up on the next real change.

Q **Why did you trade away perfectly live per-row ages?**

A Because rebuilding the whole list every two seconds just to tick one number would yank the operator out of whatever row they had open mid-triage. We chose to protect the operator's place; the small age lag self-heals. The "oldest active" alarm — the one that matters most — is always current.

Q **Could it ever show a made-up age?**

A No. If a raise time is unknown, the age stays blank rather than guessing.

Q **Does the age counter add a timer that could bog down a screen left on for days?**

A No new timer — it piggybacks on the 2-second refresh that's already running.

A full-screen alarm pop-up that won't let a live fault be ignored

WHAT IT IS

When a genuinely new fault fires from the controller, Ring raises a full-window alarm overlay that takes over the screen, flashes a red beacon, shows the live alarm, and locks navigation until the operator either acknowledges it or explicitly chooses to keep monitoring. It fires only on a *new* fault — not every time the same alarm is acknowledged or silenced — so the screen doesn't nag.

WHY IT MATTERS

On the old system, operators learned to tune alarms out. A modal overlay that blocks the screen on a genuinely new fault makes sure the important one actually lands, and the "new fault only" trigger keeps it from becoming background noise.

HOW IT WORKS

Ring polls the controller's alarm tags, detects when a *new* base alarm appears (not a state change of one already showing), and raises the full-window overlay, which hosts the live alarm grid and flashes a red beacon. The operator dismisses it by acknowledging or by tapping "Continue monitoring." Severity inside that overlay is conveyed by the same shape-plus-color glyphs as the alarm list.

VERSUS THE OLD RS-360 SYSTEM

RS-360 trained operators to ignore alarms — pop-ups were hardcoded per alarm with a fixed red gradient and a forever-blink regardless of severity, and there was no navigation-locking modal overlay at all. Ring adds a modal overlay that locks the screen behind a genuinely new live PLC alarm until it's acknowledged or the operator explicitly chooses to continue monitoring, and it suppresses re-pop-ups on ack/silence of the same alarm.

STATUS & HONEST CAVEATS

Status: Live but partly hidden/limited. The full-window navigation-locking overlay is live and real; the richer *severity-aware* pop-up behaviors that were pitched are built but not yet wired in (dormant). Watch-out: This is the one feature in the chapter to be careful with. The pitch describes a pop-up that **blinks only for criticals, shows an EMERGENCY STOP button, and stays calm for warnings.** That behavior lives in a separate pop-up class that **is never actually invoked in the running app — it's dormant code.** The live pop-up is the **uniform red overlay**: it has **no EMERGENCY STOP button** (only "Continue monitoring"), and a Warning fires the **same loud red modal as a Critical.** Do NOT claim the live pop-up is severity-themed, that warnings stay calm, or that E-STOP is one tap away on the pop-up, and do not try to demo "calm warning vs blinking critical." What IS real and defensible: the nav-locking overlay, the new-fault-only trigger that stops re-pop-up nagging, the severity glyphs inside the embedded grid, and the read-only posture (under the read-only gate, dismissal is via "Continue monitoring" rather than a silence command).

LIKELY QUESTIONS & HOW TO ANSWER THEM

Q Show me the calm warning pop-up versus the blinking critical with the E-STOP button.

A I can't demo that today, and I want to be straight with you: that severity-themed pop-up is built but not yet switched on in the running app. What's live is a single full-window red overlay that takes over the screen on any genuinely new in-range alarm and locks navigation until it's dealt with. Differentiating warning-vs-critical *theming in the pop-up* is on the list, not done.

Q So does a minor warning throw the same scary full-screen red box as an emergency?

A Today, for in-range alarms, yes — the live overlay is uniform. Severity is still visible via the glyphs in the embedded alarm grid, but the *overlay styling* doesn't yet calm down for warnings. That's the honest current state.

Q Is there an EMERGENCY STOP button on the pop-up?

A Not on the live overlay — it offers "Continue monitoring" and acknowledgement. The E-STOP-button behavior is in the dormant pop-up class. E-STOP itself is a physical/PLC function; I won't claim a screen button that isn't wired.

Q Does it nag every time I touch the same alarm?

A No — that part is real and working. It only raises on a *genuinely new* fault, not when you acknowledge or silence one that's already showing.

Q Under the read-only gate, can I even dismiss it?

A Yes. With the read-only gate on, the silence command is suppressed, so you dismiss the overlay with "Continue monitoring." The overlay also won't instantly re-summon the same alarm.

An "are you sure?" guard before Hold, Resume, or Reset

WHAT IT IS

The familiar Process buttons that pause (Hold), restart (Resume), or reset a batch now each pop a Yes/No confirmation before they send the command to the PLC, so an accidental tap can't disrupt a running batch. Resume and Reset stay hidden until Hold has been pressed.

WHY IT MATTERS

These three buttons drive the live process. On the old system a single mis-tap went straight through. A confirm step keeps the operator's existing muscle memory but puts a deliberate "are you sure?" in front of an action that can interrupt production.

HOW IT WORKS

The Process submenu wires Hold/Resume/Reset to handlers that each call a Yes/No confirmation prompt — which defaults to "No" — before the actual write to the PLC runs. Resume and Reset only appear after Hold is pressed. The bits Ring pulses are the same control bits the legacy HMI used.

VERSUS THE OLD RS-360 SYSTEM

RS-360 sent the command on the first click — Hold/Resume/Reset pulsed the same control bits, but with no confirm-before-write guard. Ring keeps the identical control behavior and adds an explicit confirmation step in front of each live PLC command.

STATUS & HONEST CAVEATS

Status: Live and on a real screen. All three Process buttons are gated by the confirmation prompt before any write. Watch-out: Two honesty points. First, the headline phrase "every PLC write" is too broad — this specific feature covers the three Process buttons (Hold/Resume/Reset); other write paths have their own guards, so scope it to these three if pressed. Second, while the read-only safety gate is on, the write is additionally suppressed app-wide, so today the prompt is real but no command leaves the PC until the gate is turned off. Be ready to explain that if a buyer taps Hold in a demo and nothing happens to the plant.

LIKELY QUESTIONS & HOW TO ANSWER THEM

Q If I tap Hold in a demo, will it actually hold my live batch?

A Not while the read-only safety gate is on — the prompt appears, but the command is suppressed before it reaches the controller. That's the cutover safety posture: zero write commands until you deliberately turn the gate off. When enabled, the confirmed command pulses the very same control bit the old system used.

Q "Confirm before every PLC write" — is that literally every write in the app?

A This feature specifically covers the three Process buttons: Hold, Resume, and Reset. Other write paths have their own safeguards. I'd scope the claim to those three rather than overpromise.

Q **Does the confirmation default to "Yes" so people just bang through it?**

A No — the prompt defaults to "No," so a stray Enter or reflexive tap doesn't push the command through.

Q **Is this a different command than my operators know today?**

A No. It pulses the identical control bits the legacy HMI used, so the behavior is the same; we only added the confirmation in front.

Q **Why are Resume and Reset hidden at first?**

A They only make sense after a Hold, so they stay hidden until Hold is pressed — which also prevents resetting or resuming a batch that was never held.

Tie downtime to the exact batch that was running

WHAT IT IS

A report that, for any alarm, shows which batch (and which recipe) was running at the moment the alarm fired — so maintenance and supervisors can attribute downtime or a quality issue to a specific batch and formula instead of leaving it a mystery.

WHY IT MATTERS

"What was the plant making when this went wrong?" is the first question in any root-cause or accountability conversation, and on the old system it was unanswerable without manually cross-reading two separate logs. Linking the two turns guesswork into facts.

HOW IT WORKS

The correlation is a straightforward time-window join: for each alarm, Ring finds every batch whose start-to-end window contains the moment the alarm fired. Because alarms don't carry a tank identity, it matches all batches running at that instant (usually just one), and it widens the lookback so a long batch that started before the window still counts. The result is shown on a dedicated Alarm ↔ Batch report screen (Reports > Alarm ↔ Batch).

VERSUS THE OLD RS-360 SYSTEM

RS-360 kept alarm history and batch history as separate, unlinked logs — both existed, but there was no way to join them and ask "what was running when this alarm fired?" Ring answers that question on one screen.

STATUS & HONEST CAVEATS

Status: Live and on a real screen, reachable at Reports > Alarm ↔ Batch. Watch-out: Two honest points. Because alarms carry no tank tag, an alarm is matched to *every* batch running at that instant — usually one, but if two tanks were running at once it shows the concurrent batches, so "the exact batch" can really be "the concurrent batches." And the report reads Ring's own batch and alarm history, so on a fresh cutover it will be thin until that history accumulates.

LIKELY QUESTIONS & HOW TO ANSWER THEM

Q How can it know the "exact" batch if alarms don't carry a tank?

A It matches by time: the batch (or batches) running at the instant the alarm fired. Usually that's one batch. If two tanks were running concurrently it honestly shows both — it can't pick one out, because the alarm itself doesn't say which tank.

Q Will this work on day one of cutover?

A The feature works on day one, but it reads Ring's own recorded history, so the report is thin until Ring has been logging for a while. It fills in as batches and alarms accrue.

Q Didn't the old system have alarm and batch logs too?

A It did — but as two separate, unlinked logs. There was no way to join them. Ring's contribution is the join on one screen, not the existence of the logs.

Q Could this report change anything in the plant?

A No. It only reads stored history; it issues nothing to the PLC.

Honest coverage status: 76 alarms correct today, 94 legend rows tracked

WHAT IT IS

This is a transparency item, not a screen feature. Ring's alarm catalog faithfully reproduces 76 controller alarm numbers correctly today. The plant's full Dutch legend, though, contains about 94 additional alarm numbers Ring hasn't seeded yet — and until the catalog is expanded, importing the legacy legend silently skips them.

WHY IT MATTERS

A skeptical buyer or controls engineer will eventually count the catalog. Disclosing the gap up front — exactly what's covered, exactly what isn't, and that the fix is straightforward and tracked — builds more trust than a glossy "all alarms covered" claim that falls apart under inspection.

HOW IT WORKS

An audit of the legacy Dutch legend against Ring's seed found 94 numbers that need a new catalog row (plus 46 that need updated text and 30 already fine). The legacy-legend importer currently does a "blind update" — it only refreshes numbers that already exist in the seed, so any legend row with no matching seed entry is silently dropped rather than added. The fix is purely additive: expand the seed with the missing numbers and switch the importer to an "upsert" (update-or-insert).

VERSUS THE OLD RS-360 SYSTEM

The legacy legend was itself fragile — alarms 30/31 read "Thermische veiligheid ??", 97/98 were blank, and alarm #4 had no entry, so it showed "A0004." Ring already shows the full meaning for alarms the legacy left as placeholders (including the real text for #4); the remaining 94 numbers are a known, additive seed expansion.

STATUS & HONEST CAVEATS

Status: In progress (tracked, not done). The 76 seeded alarms render correctly today; expanding to the remaining legend rows is an open, additive task. Watch-out: The honest selling point here *is* the disclosure. Be ready to confirm plainly: the 76 seeded alarms are correct now; the 94 un-seeded legend numbers (including the whole 9xx block) are skipped on a legacy-legend import because the importer only updates and never inserts; and the fix (seed + upsert) is additive and tracked, not yet shipped.

LIKELY QUESTIONS & HOW TO ANSWER THEM

Q So your alarm coverage isn't complete — why should I trust the rest of the pitch?

A Because I'm telling you exactly where it stands rather than hiding it. The 76 alarms in the controller's active catalog are correct today. The legacy legend has about 94 more numbers we haven't seeded yet — a known, tracked, additive task. That transparency is the point of this item.

Q What actually happens if I import the old legend right now?

A The importer only updates numbers that already exist, so the 94 numbers with no seed row are silently skipped — they don't error, they just don't appear. The fix is to add those rows and change the importer to insert-or-update.

Q Is fixing this risky — will it touch the controller?

A No. It's a pure data expansion inside Ring — add catalog rows and change one import behavior. It changes nothing in the PLC.

Q When will the 94 be done?

A It's a tracked task with a clear, additive plan. I'd rather commit to it as scheduled work than claim it's already finished — the 76 we have are correct now.

On a screen left running for days, a live fault never quietly vanishes

WHAT IT IS

A set of robustness behaviors for an HMI that stays on for days at a time: a live (still-active) alarm never silently disappears because of a stale date filter, and reading a playbook for a live alarm isn't interrupted by the 2-second refresh throwing away your place.

WHY IT MATTERS

A plant HMI runs 24/7. A naive screen can quietly hide an old-but-still-active fault behind a date window that froze at boot day, or it can rip the operator out of the alarm they're working every two seconds. Both are exactly the failures that erode trust in an alarm screen. These behaviors prevent them.

HOW IT WORKS

The default date window rolls forward on every refresh, so a screen left on for days never freezes at the day it booted. Active alarms are always shown regardless of the date window, so a weekend-old still-active fault never vanishes. And when the displayed set of alarms hasn't actually changed, Ring skips rebuilding the list — which preserves the operator's selected row (and any open playbook), sort order, and scroll position across refreshes. The only thing deliberately excluded from that "did anything change?" check is the drifting age text, so it can't defeat the guard.

VERSUS THE OLD RS-360 SYSTEM

These are modern robustness behaviors for a long-running Windows HMI; the single-shot DOS-era RS-360 display simply didn't face the same date-window and refresh pitfalls, so this is a genuinely new concern rather than a like-for-like comparison.

STATUS & HONEST CAVEATS

Status: Live and on a real screen. The rolling window, always-show-active rule, and selection-preserving refresh are all in the running build. Watch-out: This is honestly framed as a modern-HMI concern with no legacy equivalent. The one tradeoff to own — and it ties directly to the alarm-age feature — is that this very "don't rebuild the list" guard is *why* a single row's age caption can briefly lag. It's a deliberate choice to keep the operator's open playbook and selection alive during triage.

LIKELY QUESTIONS & HOW TO ANSWER THEM

Q If I leave this screen up all weekend, will an old active alarm scroll out of view and get lost?

A No. Active alarms are always shown regardless of the date filter, and the default date window rolls forward each refresh — so a still-active fault from days ago stays visible rather than being hidden behind a stale window.

Q Does the 2-second refresh kick me out of the alarm I'm reading?

A No — that's the point of this feature. When nothing about the alarm set actually changed, Ring skips rebuilding the list and keeps your selected row, open playbook, sort, and scroll exactly where they were.

Q Earlier you admitted the per-row age can lag — is that a bug?

A It's the deliberate flip side of this guard. To keep your place during triage we don't rebuild the list for a cosmetic age tick, so one row's age can lag a cycle and then self-heals. We chose protecting the operator's selection over a perfectly live per-row age; the "oldest active" headline is always current.

Q You're comparing against a DOS system that ran differently — is this comparison fair?

A We say plainly it's not a like-for-like: the old single-shot DOS display didn't face multi-day refresh and date-window problems. This is new engineering for a 24/7 Windows HMI, not a feature the legacy had and lost.

Chapter 5 — On the Floor & Standing Up a New Site

This chapter is about the un-glamorous half of the job: the day a crew actually uses the new screen on a live shift, and the day someone has to set up a second screen on a fresh machine. None of it changes the glue. Before the feature list, fix three words in your head. **PLC** is the controller — the Allen-Bradley box that physically opens valves, runs pumps, and times the mix. Ring never changes it. **HMI** is the operator screen — the window a person looks at and clicks. Ring *is* the HMI; it replaces the 2002 DOS-era screen called RS-360 while leaving the same PLC untouched underneath. A **batch** is one production run of one glue recipe.

The real floor problem this chapter addresses is "everything that isn't the recipe itself." Shifts hand off by memory and scribbled paper. A communications drop between the screen and the PLC can pass unnoticed until something goes wrong. Standing up a new station on the old system meant rebuilding a 2002 developer's machine by hand, with settings buried in **INI files** (plain-text config files) and the plant's real Dutch equipment names trapped in a dead database format. And the screen itself, fresh out of the box, felt like foreign software because it showed generic names nobody on the floor uses.

One honest framing you should carry into the meeting: the old RS-360 was not a black hole. It *did* record batch, step, shift, and alarm history — but in a corruption-prone, long-dead **Paradox/BDE** database format with no living "historian" (no continuously-running data recorder), no on-screen trends, no cost numbers, no step-duration timing, and no way for an operator to search it. So the honest pitch is never "the old system stored nothing." It is "the old system locked its data in a format only a custom extraction tool can open, and gave operators no living, searchable, on-glass access to it." Ring is the living, searchable, on-glass layer on top of the same plant.

One more phrase to get exactly right, because a controls engineer will pounce on it. Ring ships with a **read-only safety gate** turned on (a setting called `ReadOnlyMode=true`) that blocks every command that could change the process. On-site it read 133 of 133 live values from the PLC without issuing a single command that could move anything. Reading still puts request packets on the network — so never say "zero bytes." Say **"zero write commands."** Reads ask; they never tell.

A **tag** (used throughout below) is just a named value inside the PLC — like `Tank_IO[1].O_Agitat` — that the screen can read or, when allowed, write.

Stand up a whole new station in a few guided minutes

WHAT IT IS

The first time Ring boots on a new machine, it does not dump the technician into a blank settings file. It runs a setup wizard — the same friendly "next, next, next" experience as setting up a new phone. It walks the person through, one step at a time: pick a language, connect and test the PLC, confirm the safe read-only mode, set up email and who gets alerts, seed some default data, add the plant's logo and branding, set the shift pattern, and import the plant's real equipment names. At the end the station is live.

WHY IT MATTERS

This is the difference between "a developer flies out for a day" and "a commissioning technician does it before lunch." It also means a supervisor can re-run the same wizard later from the Setup menu to fix a setting, without

ever opening a config file. The read-only safety step is deliberately a human decision: writes to the plant are never switched on by accident — someone has to choose it on that screen.

HOW IT WORKS

The wizard is an ordered list of steps the program runs in sequence. Each step is self-contained and **fail-safe**: if a step hits a bad input internally, it lets you finish anyway rather than trapping you. A small flag is saved once the wizard completes, so it does not nag you on every launch — but it survives a machine rebuild, and it re-prompts if a newer version of Ring adds steps. A splash screen shows while it loads.

VERSUS THE OLD RS-360 SYSTEM

On RS-360, standing up a station meant reconstructing a 2002-era developer machine by hand: there was a splash screen but no guided setup, with configuration buried in INI files (one config file alone is roughly 7,500 lines) plus a separate helper program. Ring replaces all of that with one guided, fail-safe wizard that is re-runnable any time settings need a fix.

STATUS & HONEST CAVEATS

Status: Live and on a real screen. Verified end to end: first-boot wizard runs, and it re-opens from Setup → Setup Wizard.

Watch-out: Say **"eleven steps," not "ten."** A Shift step was added after the marketing copy was written, so the real sequence is Welcome, Language, PLC, Read-Only Safety, Email (SMTP), Recipients, Seed Defaults, Branding, Plant Names, Shift, Finish. If anyone challenges the splash-screen line count (~208 lines), it is correct on disk. The exact code line numbers in the sales doc are a few dozen lines stale, but every step exists and is wired.

LIKELY QUESTIONS & HOW TO ANSWER THEM

Q Can a non-developer really do this, or is that marketing?

A Yes — it is a guided wizard a commissioning technician runs, the same shape as phone setup. The only judgement calls are picking the language, pointing it at the PLC's address, and deciding whether to leave read-only safety on (we recommend yes for cutover).

Q What if a step fails or someone types something wrong?

A Each step is built to let you finish rather than trap you, so a bad entry never bricks the setup. You fix it afterward by re-running the wizard from Setup, or in the specific Setup screen for that item.

Q If I rebuild the machine, do I redo everything from scratch?

A The "wizard completed" marker and your settings persist, and the wizard is re-runnable. You re-run it only if you want to change something or if a Ring update added a new step.

Q How many steps is it exactly?

A Eleven: Welcome, Language, PLC connection, Read-Only Safety, Email, Recipients, Seed Defaults, Branding, Plant Names, Shift, and Finish.

See your own equipment names on day one — imported from RS-360

WHAT IT IS

On day one, the new screen can show the plant's own familiar names — the Dutch tank names, ingredient names, custom recipe names, even alarm text — instead of generic "Storage Tank 3." It does this by reading those names straight out of the old RS-360's files and loading them in. A preview screen shows a supervisor every name it is about to set, side by side with what is there now, so a human signs off before anything is written.

WHY IT MATTERS

Two payoffs. First, nobody re-types dozens of fields by hand. Second, the screen stops feeling like imported foreign software the moment it boots, because it speaks the plant's own vocabulary — the exact tank and ingredient words the operators already use. That is a real trust-builder in front of a skeptical senior operator.

HOW IT WORKS

Two design rules make this work in any language. (1) It reads each old file using the correct character encoding (the default fits Western-European text; several others, including Cyrillic, Turkish, Japanese, and Chinese, are supported) and stores everything in modern universal text. (2) It matches things by **slot number, not by label** — the old system always put the mix tank in slot 1, storage in slots 2–7, dosers in 8–13 — so it maps correctly whether a field reads "Natief Zetmeel" or "Domestic Starch." Every proposed name is tagged New / Changed / Unchanged with a confidence flag, factory-default "Formula N" names are skipped, and only after the supervisor confirms does it write — through Ring's own internal save paths, so the screens refresh live.

VERSUS THE OLD RS-360 SYSTEM

Migrating off RS-360 normally means re-keying everything by hand: its data sits inside Paradox/BDE tables and INI files with no documented export path, and is genuinely awkward to extract. Ring is a purpose-built, preview-then-commit, slot-keyed importer that liberates exactly that data and writes it into Ring so the screens update immediately.

STATUS & HONEST CAVEATS

Status: Live and on a real screen. Reachable both inside the wizard (Plant Names step) and from Setup → Legacy Import.

Watch-out: To demonstrate it with real names, you must first point it at a **copied RS-360 `Ini\MultiPanel` folder** (for example the on-site copy at `C:\PlantDrop\Ringwood\RS-360`). Ring's own name tables ship **empty**, so on a brand-new database the preview is blank until you supply that source folder. Have the folder staged before any live demo.

LIKELY QUESTIONS & HOW TO ANSWER THEM

Q Could this importer touch or change the running plant?

A No. It reads old files on disk and writes names into Ring's own database. It never speaks to the PLC, so it cannot move a valve or change a recipe in the controller.

Q How do I know it mapped the right name to the right tank?

A It maps by the old system's fixed slot numbering, not by guessing from text, and it shows you a preview with old-value-versus-new for every row. A supervisor approves before anything is saved.

Q What if our files are in Dutch — or some other language entirely?

A That is the whole point. It decodes each file by character encoding and stores universal text, and it matches by slot rather than by the words, so Dutch, Turkish, Chinese, or anything else imports the same way.

Q Will you demo it live with our data?

A Yes, as long as we have a copy of your RS-360 `Ini\MultiPanel` folder staged first. On a fresh Ring database with no source folder, the preview is correctly blank — that is not a bug, it just has nothing to read yet.

Never miss a comms loss or a live alarm — safety is visible on every screen

WHAT IT IS

Every Ring screen is wrapped in a fixed frame of status indicators that you cannot navigate away from. A banner across the top shows whether the PLC is connected. A loud banner shows when the system is in safe read-only mode. If an alarm fires, a full-screen overlay drops in and locks the menu until it is dealt with. Small pop-up "toasts" confirm that an action went through without blocking your work. A strip along the bottom always shows the clock and the connection state. And a small badge shows not just "up/down" but five distinct connection states, so "truly offline" reads differently from "reachable but idle."

WHY IT MATTERS

A missed communications drop is the nightmare: the screen looks normal but is showing frozen numbers. Ring makes that structurally hard to miss — the link state is always on screen. The read-only banner means nobody is ever confused about whether the system can write to the plant. And the five-state badge cuts the "the PLC is down!" false alarms, because operators can see the difference between a genuinely dead link and a link that is fine but momentarily idle.

HOW IT WORKS

The main window is laid out as fixed rows plus stacked overlays. One row is the PLC banner (it disappears when connected and shows a "Retry now" button when disconnected); one is the read-only banner, deliberately styled loud; the bottom row is the clock-plus-connection-plus-data-freshness strip. The alarm overlay sits on the very top layer and locks navigation; toasts and the help overlay sit just below; the active-alarm banner is placed so it

never covers the menu's click area. The connection state is computed from the PLC's heartbeat and surfaced as named states.

VERSUS THE OLD RS-360 SYSTEM

RS-360 had an alarm form and a comms log, but no persistent link-state banner, no read-only safety banner, no navigation-locking alarm pop-up, no non-blocking toasts, and no five-state heartbeat — so a comms drop could pass unnoticed until something went wrong. Ring keeps link state, safety mode, and live alarms visible on every screen at all times, and tells offline apart from idle so operators trust the badge.

STATUS & HONEST CAVEATS

Status: Live and on a real screen. The banners, overlays, bottom strip, and link badge are all wired; the layering line references are exact.

Watch-out: If you are grilled on the **"five-state badge," name all five: Connected, Stale (lagging), Starved (reachable but the control logic is idle), Connecting, and Disconnected.** The marketing text lists only four — the one it omits is "Connecting." Say "Starved" as "reachable but control logic idle" in plain English.

LIKELY QUESTIONS & HOW TO ANSWER THEM

Q What are the five connection states, exactly?

A Connected, Stale, Starved, Connecting, and Disconnected. "Stale" means data is lagging; "Starved" means the link is reachable but the control logic is sitting idle; the others are self-explanatory.

Q Can an operator click past a live alarm and ignore it?

A No. A live alarm drops a full-screen overlay on the top layer that locks navigation until it is addressed. That is by design.

Q How does this stop a frozen-screen disaster where the numbers look live but aren't?

A The bottom strip carries data-freshness, and the link badge shows the real state of the connection on every screen. A stalled link no longer hides behind a normal-looking page.

Q Why do the toasts not get in the way?

A They are deliberately non-blocking confirmations — they appear, confirm the action, and fade, on a layer that does not eat your clicks, so they never interrupt the task.

Cut missed-context errors between crews with durable shift handoff

WHAT IT IS

A set of screens that let crews hand off in writing instead of by memory. You can start and end a shift, see shift status, leave a written end-of-shift note for the next crew, and read a one-page summary of what actually happened during a shift window — the batches run, the alarms, and what is still running.

WHY IT MATTERS

Most shift mistakes are missed context: the incoming crew doesn't know what the outgoing crew knew. A durable, timestamped note trail plus a "what happened this shift" summary means the next crew inherits real information, not a scribble on a clipboard that gets lost.

HOW IT WORKS

The Shifts menu has three pieces. Shift Control backs real Start/End actions with a saved event record and a status banner, stamping who did it from the logged-in session. Handoff Notes is an append-only list — notes are added, never overwritten, and the last ten are shown. Shift Handover is a read-only summary that queries the local database for the chosen shift window and lays out batches, alarms, and still-running items.

VERSUS THE OLD RS-360 SYSTEM

RS-360 crews handed off verbally or on paper. It had a shift control tab and shift user forms, but no handoff-note trail and no shift summary at all. In fact Ring's own earlier audit had once flagged its own shift screen as a "visual mock only" — that has since been replaced with a real, repository-backed screen, and the Handoff Notes trail and Handover summary are genuinely net-new with no legacy equivalent.

STATUS & HONEST CAVEATS

Status: Live and on a real screen. Start/End, the notes trail, and the handover summary are all backed by the local database.

Watch-out: The **Shift 1 / Shift 2 / Shift 3 buttons all open one shared Shift Control screen**, not three separate per-shift screens — if asked, say so plainly; it is one control surface that knows which shift is active. (The earlier "visual mock" criticism was Ring's own honest self-audit of a prior state; it is fixed now, and that is a strong story to tell, not hide.)

LIKELY QUESTIONS & HOW TO ANSWER THEM

Q Is the handoff note actually saved, or does it vanish on restart?

A It is saved to the local database as append-only rows — notes are added, never erased — so they survive restarts and you keep a real trail.

Q Did the old system already do this?

A It had shift start/end and shift forms, but no written handoff-note trail and no shift summary. Those two are net-new in Ring with no legacy counterpart.

Q **Why do all three shift buttons open the same screen?**

A Because there is one shift-control surface that tracks which shift is active; the three buttons are just entry points to it, not three different screens to maintain.

Q **What's in the handover summary?**

A A read-only, one-page view of the selected shift window: which batches ran, which alarms fired, and what is still running — the context the next crew needs.

See every tank group live, by the plant's own names

WHAT IT IS

Dedicated live screens for each part of the plant, each named the way the plant names it: the mixer (Make Ready Tank), the storage tanks as a group, the use/doser tanks as a group, an agitator overview, the viscometer readings, and a detailed view of any single tank. So the bond-quality-critical reading and any one feed tank are a click away, not buried inside a group view.

WHY IT MATTERS

Operators need both the wide view (all storage tanks at a glance) and the deep view (this one doser tank, in detail). On a glue kitchen, viscosity is the spec that makes or breaks the bond, so having a place to log and trend that reading matters. Naming the screens the plant's way removes the "which tank is this again?" friction.

HOW IT WORKS

The Main Screen menu opens live, polling overview screens — they re-read the PLC about once a second and update. The Use Tanks menu drills into a single feed tank, with the card headers resolved from the plant's own name tables. All the single-tank views share the same underlying data-fetch machinery.

VERSUS THE OLD RS-360 SYSTEM

RS-360 scattered this across a mixer form, several storage forms, a use-group screen, the TVC overview, a viscometer form, and use-tank forms — and Ring's earlier 2026-05 audit had flagged the Use Tank Group, the viscometer main screen, and the MF1 drill-down as missing or scaffold-only. Those are now built out and named the plant's way.

STATUS & HONEST CAVEATS

Status: Live but partly hidden/limited. The group screens, the viscometer screen, and the MF1 single-tank drill-down are live and wired.

Watch-out: Two things to disclose proactively. First, the **MF2 and Double Backer single-tank drill-downs are built but hidden** (set to collapsed) — only MF1 is reachable today — pending the plant team's answer on what to name those tanks. Second, the **viscometer screen is manual entry**, because the PLC exposes no viscometer tag; it logs, calibrates against an alarm band, and trends an operator-typed reading, not a live meter feed. Both are fine to say out loud; both are honest limitations, not defects.

LIKELY QUESTIONS & HOW TO ANSWER THEM

Q Is the viscometer reading live off a meter?

A No, and we should be clear about that: the controller does not expose a viscometer value, so this screen is where an operator logs the manual reading, checks it against the alarm band, and trends it over time. It turns a hand-written number into a tracked, reportable one.

Q You mention MF2 and Double Backer — are those on a screen?

A The screens are built but currently hidden, because we are waiting on the plant team to confirm the correct names for those tanks. Only MF1 is exposed today; un-hiding the other two is a naming decision, not new development.

Q How fresh is the data on these overviews?

A They poll the PLC roughly once a second, so the values track the plant in near-real-time, and the freshness indicator on the bottom strip tells you if a feed goes stale.

Q Do the screens show our tank names or generic ones?

A Your own names — the card headers are resolved from the plant's name tables, which the legacy importer can populate from your RS-360 data.

Rename, reconfigure, and back up yourself — no developer, no callout fee

WHAT IT IS

A password-gated Setup area where a supervisor or admin runs their own configuration instead of phoning a developer. Behind the password sit roughly two dozen-plus screens: rename tanks and groups, edit recipes, set shift names and times, manage inventory, set the PLC clock, back up the database, and more. Changes are saved and survive a restart.

WHY IT MATTERS

On the old system, a tank rename or a shift-time change was a developer callout. Here it is a supervisor with a password. That removes both the cost and the wait of getting a vendor on the phone for routine config changes.

HOW IT WORKS

The Setup menu opens behind a password dialog that returns "Supervisor" or "Admin." Supervisors see about two dozen configuration screens; admins see those plus a handful of higher-privilege tools. Renames and edits are written to the local database (a lightweight, single-file database called **SQLite**), so they stick across restarts.

VERSUS THE OLD RS-360 SYSTEM

RS-360 had a Setup menu family (tank, group, formula, shift, viscometer, time/date, import/export setup), but several of Ring's own equivalents had once been session-only — for example a tank rename used to be lost on restart. That is now fixed: changes persist, and the gate cleanly separates supervisor from admin actions.

STATUS & HONEST CAVEATS

Status: Live and on a real screen. About 23 supervisor screens plus 8 admin screens behind the password gate.

Watch-out: Be ready to **name the intentional "coming soon" placeholder buttons** — Alarms-admin, Communication, Import/Export Setup DB, Mixers, Distribution, and an edit-passwords stub — so roughly 26 of the ~31 buttons are genuine working screens and a few are deliberate placeholders for re-commissioning power-user tools. Also: "edits survive restart" was spot-verified for the Costs screen and the importer specifically, not separately confirmed on every single screen — phrase it as "changes persist to the database" and lean on the verified examples.

LIKELY QUESTIONS & HOW TO ANSWER THEM

Q How many Setup screens are real versus coming-soon?

A About 26 genuine working screens behind roughly 31 buttons; the handful that aren't are clearly-labeled placeholders for power-user re-commissioning tools. I can name them: Alarms-admin, Communication, Import/Export Setup DB, Mixers, Distribution.

Q If I rename a tank, does it stick after a reboot?

A Yes — renames and edits write to the local database, so they survive restart. That was specifically a weakness in an earlier Ring build that has been fixed.

Q Who can get into Setup?

A It is gated by a password that distinguishes Supervisor from Admin. Supervisors get the configuration screens; a few higher-privilege tools (like factory reset) are admin-only.

Q Do I need a developer for any routine change?

A No. Renaming tanks, editing recipes, setting shift times and the clock, and backing up data are all supervisor-level tasks behind the password — no callout.

Fix faster and stay informed off-site — alarm emails with fix steps, scheduled reports

WHAT IT IS

A notification platform built into Ring. It can email an alarm — and the steps to fix it — to whoever you choose; mail any report on a daily/weekly/monthly schedule; email a completion notice when a batch finishes (today via a config flag; an in-app on/off toggle is a follow-up); and ping Microsoft Teams or Slack. It is set up in the wizard and in a Setup screen, with a "test send" button to prove it works.

WHY IT MATTERS

A maintenance tech can get the alarm and the fix on their phone before driving in. The owner gets the Monday-morning report without coming to the plant. And operators don't get buried in alarm spam, because the system batches repeated alarms into a digest and enforces a cooldown.

HOW IT WORKS

Email goes out through a shared sender that queues messages and sends them on a background worker, so the screen never freezes waiting on email. It retries up to three times with increasing delays and logs every message (with the full error text if one fails). The alarm path adds storm control (a cooldown per alarm, and digest mode when too many fire in a window), stamps each alarm with the recipe and step at the moment it fired, includes the first-response checklist, and can send a follow-up when the alarm clears. The report scheduler works out which subscriptions are due and attaches a PDF.

VERSUS THE OLD RS-360 SYSTEM

RS-360 had no email or notification capability at all — alarms and reports were screen- and print-only, so you were blind off-site. Ring's full notification platform (queued, retried email; scheduled report attachments; completion notices; Teams/Slack) is net-new value with no legacy peer.

STATUS & HONEST CAVEATS

> Status: Live and on a real screen. Alarm emails, scheduled reports, and the Teams/Slack webhooks are real and wired; there is a working test-send.

> Watch-out: Two honest caveats. First, **nothing sends until you configure the SMTP host and "from" address** — out of the box it is dormant by design. Second, the **batch certificate** (a completion email with an optional yield/quality PDF) is **enabled by a config flag only and is off by default; there is no in-app on/off toggle yet — that toggle is a stated follow-up.** Phrase the certificate as "available today via a config flag, with an in-app switch coming," not as a one-click in-the-box feature.

LIKELY QUESTIONS & HOW TO ANSWER THEM

Q Does this work the moment I install it?

A The platform is built and wired, but it correctly sends nothing until you enter your email server details — host and from-address — which you do in the wizard or the Setup screen, and there's a test-send button to confirm.

Q The batch certificate — is that a checkbox an operator flips on?

A Not yet. Today it is enabled by a configuration flag and is off by default; an in-app toggle is a planned follow-up. The honest line is "it's available now via config, the one-click switch is coming."

Q Won't a noisy plant flood inboxes with alarm mail?

A No — there is a per-alarm cooldown, and if too many fire in a window it switches to a single digest. That storm control is built specifically to prevent spam.

Q Does sending email slow down the screen?

A No. Email is queued and sent on a background worker, fire-and-forget, with retries — the operator UI never blocks waiting on it.

Q Are Teams and Slack actually working?

A The webhook senders are built and validated (HTTPS-only). I'd describe them as wired but, in candor, I have not personally traced an end-to-end fire in this review, so I'd want to test-send them on your environment during commissioning.

Turn the HMI into an owner's management tool — cost, alarm guidance, maintenance

WHAT IT IS

A cluster of Setup screens that give the owner management tooling the old system never had: ingredient costs and currency (so you can see what a batch costs), a playbook of what to do when each alarm fires, calibration and maintenance reminders, and a way to export or import the whole plant configuration as a portable profile to carry to a second station.

WHY IT MATTERS

This is the line between "operate the plant" and "manage the plant." Cost visibility turns the HMI into something an owner reads. The alarm playbook captures the senior tech's knowledge so it doesn't walk out the door. The maintenance board stops calibrations slipping. And the portable profile means standing up a second station inherits the first one's configuration.

HOW IT WORKS

Each is a built, persisted Setup screen: the Costs screen writes currency and a downtime rate into the config plus per-ingredient costs into the database; the Alarm Playbook screen stores writable first-response notes per alarm; the Maintenance screen is a colour-coded due/overdue board; the Plant Profile screen exports and imports a commissioning subset. They live in the same Setup family as the email, language, and import screens.

VERSUS THE OLD RS-360 SYSTEM

RS-360 gave owners no management layer at all: INI files, no cost module, no alarm playbook, no maintenance board, no config export. Ring turns the HMI into an owner-facing management tool rather than just a control panel.

STATUS & HONEST CAVEATS

Status: Live and on a real screen. All four screens are built, wired, and persist.

Watch-out: **Currency and Plant-Profile changes are restart-required** (the per-ingredient unit costs apply immediately, but the currency/downtime-rate config and the profile settings take effect on next launch). And the **alarm-playbook content** — the actual "who to call / what to check" wording — needs a plant-knowledgeable person to sign off; the tool is built, but it ships without your specific procedures filled in.

LIKELY QUESTIONS & HOW TO ANSWER THEM

Q Where do the cost numbers come from?

A You enter ingredient unit costs and a currency in the Costs setup screen. Ring then shows what a batch costs. The unit costs apply immediately; the currency and downtime-rate settings take effect after a restart.

Q Does the alarm playbook come pre-filled with the right steps for our plant?

A No — the editor is built, but the actual first-response wording is yours to enter, because only someone who knows your plant can write "who to call, what to check." That's a sign-off item for commissioning, not a code gap.

Q What does the portable plant profile actually carry?

A A commissioning subset of your configuration that you can export from one station and import into a second one, so you don't reconfigure from scratch. Note the imported profile settings take effect on restart.

Q Did the old system have any of this?

A None of it. No cost module, no playbook, no maintenance board, no config export. This is the management layer that didn't exist before.

Operators read the HMI in their own language and script

WHAT IT IS

Ring runs in 13 languages — English, Dutch, German, French, Spanish, Italian, Portuguese, Turkish, Polish, Korean, Japanese, and both Chinese scripts — and you can switch between them on the fly, without restarting. English is the default. The Dutch words come from the live plant's own old dictionary, so Dutch operators see the exact terms they already know.

WHY IT MATTERS

The screen stops feeling like imported foreign software when an operator reads it in their own language and script. And being able to flip the language live, in front of a senior operator, is a genuine trust-builder during a sale or a cutover.

HOW IT WORKS

Ring keeps an English string set as a permanent base and layers the chosen language on top, "last wins." Any word that hasn't been translated yet simply falls through to English — never a blank. Switching language swaps the active dictionary live, so the screen repaints instantly, and the choice is remembered for next launch. Importantly, the language switch deliberately does **not** touch the system's number formatting, so the batch math and PLC number parsing are never affected by which language is displayed.

VERSUS THE OLD RS-360 SYSTEM

Be careful and precise here, because a controls engineer who knew RS-360 will catch an overstatement. The honest, QA-corrected line: **RS-360 shipped many language dictionaries and had an in-app language menu, but switching required a reload/reconfigure to take effect and it had no English fallback for missing strings.** Ring switches the live UI instantly with no restart and always falls through to English. Do **not** say RS-360 "couldn't switch without reconfiguring" as an absolute — it had a menu; the real differentiators are *live* switching and the safe English fallback.

STATUS & HONEST CAVEATS

> Status: In progress (honestly labelled "progress," and that is correct). The translation engine and the core operator spine — wizard, dashboard, navigation bar, common buttons, core equipment nouns — are translated and switch live.

> Watch-out: **Full coverage of every screen body is not done.** An internal audit flags roughly 80–91 submenu items and screen bodies still hard-coded in English. The safe phrasing: "the engine and the operator spine are shipped; full screen-by-screen coverage is still in progress, with English as the never-blank fallback." Also clarify the two mechanisms: the plant's exact Dutch **equipment names** (Voorraadtank, Doseertank, etc.) come from the **RS-360 importer**, while the **interface words** come from the language **dictionary** — two different systems, don't conflate them.

LIKELY QUESTIONS & HOW TO ANSWER THEM

Q Is the whole screen translated, or just parts?

A The engine plus the core operator spine — wizard, dashboard, nav bar, common buttons, core nouns — are translated and switch live. Full coverage of every screen body is still in progress; an audit lists roughly 80–91 items still in English. Anything untranslated falls back to English automatically, so you never see a blank label.

Q Does switching language change the math or the numbers sent to the PLC?

A No, and that was a deliberate design choice. The language switch only changes displayed text; it never touches number formatting, so batch math and PLC number parsing are unaffected.

Q Where do our Dutch tank names come from — the language setting?

A No — those are two different things. The interface words come from the language dictionary; your specific equipment names (Voorraadtank, Doseertank, and so on) come from the RS-360 importer pulling them out of your old system.

Q Did the old RS-360 already do languages?

A It had language dictionaries and an in-app menu, but a switch required a reload to take effect and there was no fallback for missing words. Ring's advance is instant live switching with a safe English fallback.

End the "where am I / how do I get back" confusion

WHAT IT IS

A single menu bar that runs down the left edge of every screen and is always there. Click a section like Reports or Setup and a small menu flies out next to it. The button for the screen you are currently on stays highlighted, and a Back button retraces your steps.

WHY IT MATTERS

The old system was a sprawl of separate windows and per-tank menu items — roughly 97 forms — with no "you are here." Operators got lost and couldn't find their way back. One consistent rail with an active-screen highlight and a Back button removes that friction entirely.

HOW IT WORKS

The rail is a fixed strip with eleven top-level buttons; nine of them open a small fly-out menu that closes when you click away or press Escape. Clicking a menu item swaps the main content area to that screen (re-using a cached copy where it makes sense) and marks the rail button as active with a coloured accent. Back is a separate history stack.

VERSUS THE OLD RS-360 SYSTEM

Getting around RS-360 meant menu-diving through roughly 97 separate forms in a 2002-era windowing interface with hand-built per-tank menu entries and stacked child windows, and no persistent "you are here." Ring replaces

all of that with a single persistent rail, active-screen highlighting, and a Back stack.

STATUS & HONEST CAVEATS

Status: Live and on a real screen. The rail, fly-outs, active highlight, and Back stack are all wired.

Watch-out: A minor evidence nit only — the sales doc cites the active-highlight code at one line number; the file has grown and it now sits about 50 lines lower. The method exists and works; this is bookkeeping, not a functional gap.

LIKELY QUESTIONS & HOW TO ANSWER THEM

Q How does an operator know which screen they're on?

A The rail button for the current screen is highlighted with a coloured accent bar and filled background — a clear "you are here" — and Back retraces your steps if you wander off.

Q Doesn't a left rail with fly-outs get cluttered?

A No — only eleven top-level buttons are always visible; the detail lives in fly-out menus that open on click and close when you click away or hit Escape, so the rail itself stays short.

Q Is this faster than the old window-juggling?

A Yes. Instead of menu-diving through dozens of separate windows with no sense of place, every screen is one rail away, with a consistent Back button.

Root-cause faster and protect your data

WHAT IT IS

A set of engineer and maintenance tools so the floor can self-serve troubleshooting and keep data safe: read any live PLC value without opening the programming software, capture a one-shot system snapshot to send to support, watch a live feed of PLC messages to see why comms hiccuped, set the PLC clock, run one-button database backup and restore, and — admin only — factory reset.

WHY IT MATTERS

When something is off, the question is "is it the network, the PLC, or the screen?" These tools answer that on the floor in minutes, without flying in a specialist or opening Studio 5000 (Rockwell's PLC programming software). And the one-button backup protects the plant's accumulated history.

HOW IT WORKS

These are pop-up dialogs. The Tag Inspector lets you paste a tag name (like `Tank_IO[1].O_Agitat`) and reads its current live value. The Diagnostic Console takes a one-shot snapshot of host, PLC, and traffic, behind a supervisor password. The Communication Monitor tails the live send/receive messages. The Time/Date tool sets the

controller clock. Backup/restore uses the database's own safe online-backup, re-prompting for the password. Factory reset is admin-only and re-prompts before it runs.

VERSUS THE OLD RS-360 SYSTEM

RS-360 had a comms log with disk logging and a time/date form, but no live tag inspector, no one-shot diagnostic snapshot, no built-in database backup or restore, and no factory reset. The Tag Inspector and the Diagnostic Console are the strongest all-new technical tools, sitting alongside safe backup/restore and a guarded factory reset.

STATUS & HONEST CAVEATS

Status: Live and on a real screen. All the dialogs exist and are wired behind the appropriate password tiers.

Watch-out: The **Tag Inspector is read-only by design** (it reads values; it does not write) — say that confidently, it is the safe answer. And the **PLC clock-set tool is suppressed while the read-only cutover gate is on**, because setting the clock is a write to the controller; that is correct behaviour, not a defect. If line counts are challenged (Diagnostic Console ~1,149 lines, Tag Inspector ~260), they are accurate on disk.

LIKELY QUESTIONS & HOW TO ANSWER THEM

Q Can the Tag Inspector accidentally change a value in the running PLC?

A No. It is read-only by design — you paste a tag and it shows the live value. It has no path to write. That is exactly what you want in a diagnostic tool.

Q Does an engineer need Studio 5000 to read a tag now?

A No — that's the point. The Tag Inspector reads any live PLC value from inside Ring, so floor troubleshooting doesn't require opening the programming software.

Q Why doesn't the clock-set work right now?

A Because setting the controller clock is a write to the PLC, and the read-only cutover gate is intentionally suppressing all writes. Turn off read-only mode after sign-off and the clock-set is available.

Q How is the plant's data protected?

A A one-button backup uses the database's safe online-backup, and there's a guarded restore. Factory reset exists too, but it's admin-only and double-prompts before it runs.

Get hand-logged readings off paper and into reports

WHAT IT IS

The things operators write down by hand — shift readings, manual measurements — become digital and reportable. There are five fill-in forms, and a supervisor can rename the field labels to match the plant's own terms, so the data lines up with how the plant already talks.

WHY IT MATTERS

Hand-logged readings on paper are invisible to everyone who isn't holding the clipboard. Capturing them in Ring makes them searchable, trendable, and reportable — and letting the supervisor name the fields means the form matches the plant's existing workflow instead of forcing a new vocabulary.

HOW IT WORKS

A hub lists the five forms as buttons, each showing the supervisor-customized first-field label. Each form renders up to 11 fields parameterized by which screen it is. A separate setup hub lets a supervisor rename those 11 labels per screen, saved to the database. The results feed a Data Entry report.

VERSUS THE OLD RS-360 SYSTEM

RS-360 kept this on screen and on paper with an equivalent set of forms, but Ring's earlier audit headline was literally "Zero WPF mirror exists" for this whole family — it hadn't been built yet. It has since been built in full (the five-screen family plus setup and report), with supervisor-controlled labels.

STATUS & HONEST CAVEATS

Status: Live but partly hidden/limited. The five forms, the label-rename setup, and the report are all built and wired.

Watch-out: There is a real **access-tier awkwardness to disclose: the data-entry forms themselves open only behind the ADMIN password**, while the supervisor-tier button only renames the labels. That is clumsy for an operator's daily hand-logging. Expect the question "how does a line operator log a reading without admin rights?" — the honest answer is that today it requires admin access and we'd move it to a lower tier. (A stale code comment also still calls Data Entry "coming soon," but it *is* wired to the real hub — don't be thrown by that comment.)

LIKELY QUESTIONS & HOW TO ANSWER THEM

Q How does a line operator log a reading without admin rights?

A Honestly, today the entry forms sit behind the admin password, which is more restrictive than ideal for daily logging — the supervisor button only renames the labels. We'd lower the entry forms to operator or supervisor tier; the forms themselves are fully built, this is a permissions setting, not new development.

Q **Can we label the fields with our own terms?**

A Yes — a supervisor renames all 11 fields per screen, saved to the database, so the form reads in the plant's own vocabulary.

Q **Did the old system have this?**

A It had paper-and-screen equivalents, but Ring hadn't built its version until recently — that's now done: five forms, setup, and a report. Captured readings become reportable instead of trapped on paper.

Q **I saw "coming soon" in the code — is it actually finished?**

A That's a stale comment. The feature is built and wired to the real hub; the comment just didn't get cleaned up.

Set agitators to cycle themselves instead of managing paddles by hand

WHAT IT IS

A per-tank agitator screen for each storage tank. You can turn the agitator on or off, or set it to automatic mode with on/off timer presets so the paddle cycles by itself instead of an operator babysitting it.

WHY IT MATTERS

Manual paddle management is exactly the kind of repetitive attention that eats an operator's shift. Automatic timed cycling, set per tank, removes that chore — in principle.

HOW IT WORKS

Each tank's screen runs a small timer-driven state machine: it counts down an "on" interval and an "off" interval and flips the agitator state, with the timer rows enabled only when the tank is in Automatic mode. The screens are carefully written to rebuild their timer when revisited, to avoid a crash on re-open.

VERSUS THE OLD RS-360 SYSTEM

RS-360 exposed agitator control through a per-tank form, and Ring's earlier 2026-05 audit had listed all four of Ring's agitator handlers as "TODO: send command to PLC" stubs — i.e., not finished. They are now full per-tank state-machine screens (roughly 320–390 lines each; tank 1 is the largest at 386).

STATUS & HONEST CAVEATS

Status: In progress (honestly labelled "progress," and that is the right label). The four per-tank screens exist and are wired into the menu.

Watch-out: This is the most important honesty point in the chapter. **Do NOT claim these cycle the real plant paddles today.** They are local on-screen state machines. On-site evidence suggests **the PLC itself owns the agitator as a sequenced step inside the batch**, and the read-only safety gate is blocking all writes anyway, so command authority is **deliberately deferred to live observation before any write is enabled.** Also correct the line-count claim: it is **not** "four 386-line screens" — only tank 1 is 386; the others are about 322–380. Lean on the "progress / deferred authority" framing — it is accurate and defensible.

LIKELY QUESTIONS & HOW TO ANSWER THEM

Q Does this actually cycle the paddles in our plant right now?

A No, and I won't overstate it. These are the operator screens and the cycling logic, but on-site evidence suggests the PLC controls the agitator as part of the batch sequence, and our read-only safety gate is blocking all writes. So command authority is deliberately deferred until we observe the live plant and confirm who owns the agitator.

Q So what works today?

A The four per-tank screens, the on/off/automatic modes, and the timer state machine are all built and wired into the menu. What's pending is whether Ring should send those commands or leave them to the PLC's batch sequence — a live-observation decision, not a coding gap.

Q Are all four screens the same big 386-line build?

A Close but not identical — tank 1 is 386 lines; the others are roughly 322 to 380. They're all full state-machine screens; I just don't want to quote one number for all four.

Q Why defer the write instead of just enabling it?

A Because enabling a write to a function the PLC may already be sequencing could fight the controller. The safe, correct order is: observe the live plant, confirm ownership, then enable. That's the whole philosophy of the read-only cutover gate.

Keep operators productive through the switch — manual a click away, F1 cheat-sheet

WHAT IT IS

Help that keeps operators moving during the changeover: the original RS-3000 PDF manual stays one click away, a Contact Us screen gives a clear support path, and pressing F1 brings up a cheat-sheet of the app's keyboard shortcuts.

WHY IT MATTERS

The fear with any new system is that operators slow down while they learn it. Keeping the familiar manual within reach, giving a clear "who do I call," and surfacing the shortcuts means people pick up the new screen without losing speed.

HOW IT WORKS

The Help menu has the manual, Contact Us, and (hidden for now) an About entry. "Contents" opens the legacy manual PDF by searching a few likely locations on disk and launching it. Contact Us is a simple content screen. App-wide, pressing F1 opens an in-window overlay listing the shortcuts (Ctrl+B, F1, Esc, Enter, Alt+F4).

VERSUS THE OLD RS-360 SYSTEM

RS-360 had a Help Contents entry for the manual, but no in-app keyboard-shortcut help and no Contact Us screen. Ring keeps the manual one click away and adds the F1 shortcuts overlay and the Contact Us path.

STATUS & HONEST CAVEATS

Status: Live and on a real screen. The manual launcher, Contact Us, and the F1 overlay are all wired.

Watch-out: Two small honesty points. The **manual is the legacy RS-3000 PDF, not a rewritten Ring manual**, and it must physically be present on disk — the launcher searches a few locations and, if the file isn't there, nothing opens. And the **About entry is hidden ("coming soon")**. Both are easy to state plainly.

LIKELY QUESTIONS & HOW TO ANSWER THEM

Q Is the manual updated for Ring, or the old one?

A It's the legacy RS-3000 PDF — the manual operators already know — kept one click away for continuity. A Ring-specific manual would be a separate deliverable.

Q What if the manual PDF isn't on the machine?

A The launcher searches a few standard locations; if the file isn't present, nothing opens. So part of commissioning is making sure that PDF is staged on the machine.

Q What does F1 show?

A An in-window cheat-sheet of the keyboard shortcuts — Ctrl+B, F1, Esc, Enter, Alt+F4 — so operators pick up the new shortcuts fast without leaving the screen.

A home screen shaped by what operators actually glance at

WHAT IT IS

The Ring home screen is built around what operators really look at, not a designer's guess. A real operator stood in front of both the old and new screens side by side and pointed out the six things the old main screen shows at

a glance. All six are now on the Ring home screen — and the design keeps evolving from that kind of floor observation.

WHY IT MATTERS

A home screen that matches what operators already scan every hour means zero relearning of the most-used view. It is also a strong sale story: the layout came from an actual operator at the plant, not a slide deck.

HOW IT WORKS

The six field-observed items are: the current formula, the make-ready tank weight, the borax/caustic weight, the ingredient-addition progress, the mix-time progress, and the mix-time-remaining countdown. All six are now bound on the dashboard to data Ring was *already* polling — so adding them was a layout change with **zero new PLC reads**. The home screen went from showing two of the six to all six.

VERSUS THE OLD RS-360 SYSTEM

Give the old system its due here — this is a balanced contrast, which makes it credible. RS-360 docks a comms / run-hold / alarm status bar on every screen and keeps a live 17-row formula step table always visible. But that table forgets every step's duration the instant it scrolls off. Ring gives at-a-glance batch and alarm awareness shaped by the floor, and Ring can persist the step durations the legacy throws away.

STATUS & HONEST CAVEATS

Status: In progress (partly shipped). The six at-a-glance items are genuinely live on the home screen today (6/6), added with no new PLC reads.

Watch-out: Separate the shipped fact from the roadmap proposals. **Shipped and verifiable: all six items on the dashboard, 6/6.** Also now wired: the bottom-strip pills dim and show an "as of HH:mm:ss" caption when data goes stale (an earlier "dormant features" doc that says this is unused is itself out of date). **Still roadmap proposals, NOT shipped: the persona "10/10" scores, the "big numbers readable from 15 feet," the "needs-you-next" batch timeline, an off-dashboard run/hold/alarm status strip, and stale-signaling on the dashboard's KPI tiles.** Do not present those as done. Phrase the persona scores as "an operator scored these proposals a 10 in design review," not "the feature scores 10."

LIKELY QUESTIONS & HOW TO ANSWER THEM

Q How many of the six glance-items are actually on the home screen?

A All six — current formula, make-ready weight, borax/caustic weight, ingredient-addition progress, mix-time progress, and mix-time-remaining. The home screen went from two to all six, and we added them using data already being polled, so zero new reads of the controller.

Q Where did this layout come from?

A A real operator stood in front of both HMIs side by side and pointed out exactly what the old main screen shows at a glance. We built the home screen to match that, rather than guessing.

Q **The "big numbers from 15 feet" and the needs-you-next timeline — are those live?**

A Those are roadmap proposals, not shipped yet. They scored highly in design review with an operator persona, but I won't claim they're on the screen today. What's shipped today is the 6/6 glance parity plus stale-signaling on the bottom-strip pills.

Q **Does the old system show this stuff too?**

A Fairly, yes — RS-360 has an always-on status bar and a live 17-row step table. The difference is the legacy table forgets every step's duration the moment it scrolls, and Ring can keep those durations. We're not claiming the old screen showed nothing.

Q **Did adding all six items add load on the controller?**

A No — they bind to values Ring was already reading, so it was a pure layout change with zero new PLC reads.

Chapter 6 — Own Your Data, on a Platform Built to Last

Every production run your plant makes — every batch of glue, every alarm, every drum of starch consumed — leaves a record. The question this chapter answers is a simple one a buyer will press hard on: **who owns those records, can you actually get at them, and will the software that holds them survive the next ten years of power blips, upgrades, and staff changes?** Two terms before we start. The **PLC** (Programmable Logic Controller) is the Allen-Bradley industrial computer that physically runs the valves, pumps, and mixer — Ring does not change it. The **HMI** (Human-Machine Interface) is the operator's screen — that is what Ring is, the replacement for the old 2002 DOS-era "RS-360" screen. A **batch** is one production run of one glue recipe.

The floor problem is this. The old RS-360 system *did* keep history — batch headers, step records, shift summaries, alarm logs — but it stored them in a long-dead database format called **Paradox**, reachable only through an abandoned 32-bit Borland Database Engine ("BDE") that has no maintained driver. In practice that means the data sits locked in a 2002 vault: no living "historian" (a system that continuously records readings over time), no trends, no cost figures, no step-by-step timing, no equipment-runtime stats, and no way for a supervisor to search it. Getting the live plant's own data out required writing a brand-new reader from scratch. So the honest distinction is not "the old system recorded nothing" — it is "the old system recorded plenty, then sealed it where nobody can use it."

Ring's answer is to put all of that in **SQLite** — a single, open, standard database file (think of one self-contained file, like a spreadsheet, that any free tool can open) — and then wrap it in the kind of engineering that keeps an industrial PC running unattended for months. This chapter covers the data store itself and the under-the-hood engineering that makes it trustworthy.

One honesty rule that runs through this whole chapter, because a controls engineer will test it: on-site, Ring read **133 of 133 live PLC values without issuing a single command that could change the process**. Reading does send small network request packets — so never claim "zero bytes." The correct, defensible claim is **zero write commands**: every instruction that could move a valve or change the running plant is suppressed by a read-only safety gate.

A few more terms you will meet below, each defined where it first appears: a **tag** is a single named value inside the PLC (like "Tank 2 level"); a **table** is one named list of records inside the database (like the "Batch" table); a **migration** is an automatic, ordered upgrade to the database's structure; and **WAL** is a journaling mode that protects the database through a power cut.

Own all your plant data in one open file, with no vendor in the loop

WHAT IT IS

All of Ring's data — batches, alarms, inventory, recipes, shifts, costs, maintenance, email logs — lives in **one SQLite database file** on the plant PC. SQLite is a free, open, widely used database format; the whole thing is a single file you can copy to a USB stick, back up, email, or open with any free SQLite viewer. There is no proprietary engine and no vendor you must call to read your own records.

WHY IT MATTERS

When an auditor, a plant manager, or a prospective buyer of *your* business says "show me your data," you answer on the spot by handing over one file. You are not dependent on a software vendor staying in business, returning your call, or selling you a license to see records you generated. You own them outright.

HOW IT WORKS

On first run, Ring builds roughly **40 well-structured tables** (Batch, AlarmDefinitions, AlarmLifecycle, IngredientUsage, InventorySnapshots, FormulaSteps, ShiftDefinition, ProcessHistory, EquipmentRuntime, IngredientCost, and more). Each table is guarded by rules that keep bad data out — for example, an alarm's severity can only be "A" (alarm) or "W" (warning), nothing else. All of it runs on the standard `System.Data.SQLite` library, an open format any free SQLite tool can read.

VERSUS THE OLD RS-360 SYSTEM

RS-360 traps the same kinds of data across many **Paradox** tables, each one split into a quartet of files, reachable only through the deprecated 32-bit Borland Database Engine, **with no maintained driver**. Because no maintained, supportable BDE/ODBC path remains, even pulling the live plant's own data out required writing a from-scratch pure-.NET Paradox reader. Ring keeps everything in one inspectable file readable by any free SQLite tool.

STATUS & HONEST CAVEATS

Status: Live and on a real screen. The data store is in daily use, and the "hand over one file" benefit is delivered through Backup and Export screens that have been verified reachable.

Watch-out: Do not say "no BDE/ODBC driver exists anywhere" — BDE is abandonware but old copies still circulate, and a buyer could refute the absolute. Say "**no maintained driver**," which is the defensible truth. Also note: Ring's name tables (tank/ingredient names) ship **empty** — your plant's actual Dutch names appear only after a one-time legacy import is run (covered later in this chapter).

LIKELY QUESTIONS & HOW TO ANSWER THEM

Q Can I really open this file myself, without you?

A Yes. It is a standard SQLite file — download any free SQLite browser and open it; there is no license, password vault, or proprietary engine between you and your records.

Q What stops the data getting corrupted when the app or the network crashes mid-write?

A SQLite is a transactional database, so a crash leaves the last committed records whole rather than shredding an index the way the old Paradox tables could — and the durability settings that back this up are covered two features down.

Q You say "40 tables" — are those real, or just empty placeholders?

A They are real, typed tables with integrity rules built and populated in production; the count and the rules were verified directly in the code that creates them.

Q If I sell the plant, does the buyer get the data cleanly?

A Yes — that is the whole point. One file, open format, no vendor handoff required.

Self-service backups, a checked restore, and one-click Excel exports

WHAT IT IS

Three things you can do yourself, with no vendor and no special tooling: **back up** the whole database while the app is still running, **restore** from a backup that Ring checks before trusting, and **export** any table to a spreadsheet (CSV — the standard format Excel opens) in one click.

WHY IT MATTERS

Backups are your insurance for cutover day and for any future hotfix. The checked restore means you are not trusting a backup blindly — Ring checks it is sound *before* you rely on it. And the export means "send me this in Excel" is answered in seconds, not by a callout.

HOW IT WORKS

Backups use SQLite's **online backup** feature, so they capture a consistent snapshot *while the app runs* — no shutdown required. Before a restore, Ring runs an integrity check plus a **presence check of core startup tables**, takes an automatic safety copy of the current database first, then swaps files and cleans up leftover temporary files. Exports use a defined per-table column catalog with an optional date-range filter, covering Batch, Ingredient Usage, Alarm History, Shift, Inventory and more, with a table-and-column picker. PDF reports are built from the same database.

VERSUS THE OLD RS-360 SYSTEM

RS-360 gives you **no built-in backup or restore at all**; its reporting is print-outs from an abandoned report component, over data locked in the dead engine and awkward to extract. Ring gives you the familiar report-export workflow but over an open, portable store, backed by a checked restore the old system never had.

STATUS & HONEST CAVEATS

Status: Live and on a real screen. Backup/restore and export dialogs are reachable from the Setup and Reports areas and verified working.

Watch-out: An earlier draft claimed the restore validates "**all 22** startup tables, closing the silent-data-loss gap." That is overstated — the app actually ensures ~40 tables at startup, while the restore's checklist currently names a smaller core subset. Say the honest version: "the restore runs an integrity check plus a **presence check of core startup tables**, takes an automatic safety copy, then swaps files." Do not claim it verifies every table or that it fully closes the gap; the code-level fix to re-sync that checklist to the full table set is a known next step.

LIKELY QUESTIONS & HOW TO ANSWER THEM

Q Can I back up without stopping production?

A Yes — it uses SQLite's online backup, so you take a consistent snapshot while the screen keeps running.

Q How do I know a backup is actually good before I rely on it?

A The restore checks the source's integrity and confirms the core tables are present before doing anything, and it makes an automatic safety copy of your current database first, so a bad restore can be undone.

Q Does the restore guarantee every single table is intact?

A Honestly, not yet — it checks integrity plus the **core** startup tables, and re-syncing that checklist to the full ~40-table set is a known code fix. The automatic pre-restore safety copy is your backstop in the meantime.

Q Can my plant manager get an Excel file without calling anyone?

A Yes — pick a table, optionally a date range, and export to CSV that Excel opens directly.

Keep the batch you just finished through a power blip — and never freeze under load

WHAT IT IS

Two durability guarantees that run automatically on every database connection. First, a sudden power loss will **not cost you the batch you just completed**. Second, the plant PC keeps responding even when several screens are reading and writing the database at the same time — nobody waits on a frozen screen mid-shift.

WHY IT MATTERS

A night-shift power blip or a busy moment where the dashboard, batch screen, trending, and alarms all hit the database at once are exactly when an industrial system must not lose data or lock up. This removes both fears.

HOW IT WORKS

Every time Ring opens the database it applies three settings: a **15-second busy-timeout** (so concurrent access waits its turn instead of throwing a "database is locked" error), **WAL journaling mode** — Write-Ahead Logging,

which lets many readers and one writer work at once — and a deliberate **"FULL" durability override** that forces each save to be physically flushed to disk. That last setting trades a little write speed for the guarantee that committed records survive a power loss, which is the right trade for a plant PC. These settings apply uniformly across every part of the app.

VERSUS THE OLD RS-360 SYSTEM

Paradox/BDE gives you no WAL, no documented concurrency guarantees, and a corruption-prone store — recovering a corrupted table during a night-shift line stop is a genuine hazard. Ring runs WAL concurrency with a busy timeout and full flush-on-commit durability, so commits survive power loss.

STATUS & HONEST CAVEATS

Status: Live — an always-on engine guarantee. This is not a screen you click; it runs on every database connection automatically.

Watch-out: This is an engineering property, not a visible feature, so if asked "show me," the honest answer is "it's in the code that opens every connection, not a button." The "FULL" durability setting is slightly slower on writes than the default — that is the intended, documented trade for power-loss safety.

LIKELY QUESTIONS & HOW TO ANSWER THEM

Q If the power drops the instant a batch finishes, do I lose it?

A No — the database forces each committed save to flush physically to disk, so the last completed transaction survives a power loss.

Q What happens when five screens hit the database at once?

A WAL mode allows concurrent readers with a writer, and a 15-second busy-timeout makes any contention wait briefly rather than error out, so the screen stays responsive.

Q Isn't forcing every write to disk slow?

A It is marginally slower than the default, and that is a deliberate choice — on a plant PC, surviving a power loss is worth more than shaving milliseconds off a write.

Q Can you prove this is on?

A Yes — it is applied in the single shared routine that opens every database connection; we can show you that code line.

Upgrade without a database expert — and an interrupted upgrade never corrupts your history

WHAT IT IS

When you install a new version of Ring, the database **upgrades its own structure automatically and safely** — no database administrator, no manual scripts. It tracks a version number, applies only the new changes in order, and is built so that a power cut mid-upgrade **rolls back or safely re-runs** rather than leaving a half-broken database.

WHY IT MATTERS

Software will get updated over the plant's life. Without this, every update is a risk: a botched or interrupted upgrade could corrupt years of history, and you would need a specialist on hand. This makes upgrades a drop-in you can trust.

HOW IT WORKS

The database stores a version number. On startup, Ring reads it and applies each pending change (called a **migration**) in ascending order, then stamps the new version. The chain currently runs from version 1 through version 18 — adding operator identity, the cost module, alarm-email rules, process history, equipment runtime, maintenance, batch notes, and more. Each step wraps its structural change *and* its version stamp together in a single **savepoint** (a database checkpoint it can roll back to), so a crash mid-step undoes both together rather than leaving the version number and the structure out of sync.

VERSUS THE OLD RS-360 SYSTEM

RS-360 has **no version concept and no migration mechanism**. Its table structure is frozen in the 2002 build; any change means hand-editing tables in a dead toolchain with no rollback safety. Ring's schema versions itself, applies only the new deltas, and wraps each step so it rolls back or safely re-runs.

STATUS & HONEST CAVEATS

Status: Live — an always-on engine guarantee, runs on every startup.

Watch-out: Do not say "every step is strictly all-or-nothing." One specific step (version 7, the inventory rebuild) commits its own internal transaction, so its structural change becomes durable *before* the version stamp — which is exactly why that kind of step is written to be **safe to re-run** if interrupted. The honest framing: ordinary add/change steps get the full atomic rollback; the one self-committing step is deliberately built to safely re-run. Say "rolls back **or safely re-runs**," never "rolls back cleanly, always."

LIKELY QUESTIONS & HOW TO ANSWER THEM

Q Do I need a database expert to apply an update?

A No — the database upgrades itself on startup, applying only the changes newer than its current version, in order.

Q What if the power dies halfway through an upgrade?

A Ordinary steps roll back to a savepoint so the upgrade is undone cleanly; the one step that commits its own work is written to be safely re-runnable, so on the next start it either completes or harmlessly repeats. Either way you don't get a half-broken database.

Q So is it truly "all-or-nothing" for every step?

A Honestly, not literally for every step — one inventory-rebuild step commits internally, so it's idempotent (safe to re-run) by design rather than rollback-protected. That's a documented, deliberate choice, not a gap.

Q How do I know which version my database is on?

A It's stamped as a number inside the database; the upgrade chain runs 1 through 18 today and we can read the current value directly.

Inventory totals you can trust, even after a controller reset

WHAT IT IS

Inventory numbers that stay accurate. Ring never double-counts a repeated signal from the PLC, and — crucially — it never silently drops a genuinely new record after the controller is power-cycled. Your starch, caustic, and borax usage matches what the plant actually consumed.

WHY IT MATTERS

Inventory drives reordering and costing. A silent under-count is the worst kind of error because nobody notices until stock runs short or the numbers don't reconcile. This closes a specific trap where a controller reset could have quietly dropped records forever.

HOW IT WORKS

Each inventory snapshot the PLC reports carries a counter value. The PLC's counter wraps back to a low number at 32,767, and a controller power-cycle restarts it too — so without care, a brand-new snapshot could look identical to an old one and get thrown away while the system still told the PLC "got it," losing the record. Ring's fix (migration version 7) makes each record unique on the combination of its counter **plus the PLC's own event timestamp**. A true repeat has the same timestamp and is correctly ignored; a post-reset record has a different timestamp and is correctly kept. The rebuild only runs if the old design is detected, preserves all existing rows, and runs inside a savepoint so an interrupted rebuild can't leave the table missing.

VERSUS THE OLD RS-360 SYSTEM

RS-360's Paradox inventory tables have no equivalent notify/acknowledge protocol or wrap-aware deduplication in the extracted code; correctness leaned on the 2002 engine and logic with no recovery tooling. Ring uses a timestamp-aware uniqueness rule that tells a real new record from a reset collision, so totals stay correct across power cycles.

STATUS & HONEST CAVEATS

Status: Live — a background data-capture guarantee. Not a screen you click, but the resulting records are exportable through the wired Inventory Snapshots export.

Watch-out: The fix relies on the PLC supplying a **distinct event timestamp**. If the controller ever stamped two genuinely-different post-reset events with the identical time, those two could still collapse into one — an acceptable edge case, but be ready to acknowledge it honestly rather than claim perfection.

LIKELY QUESTIONS & HOW TO ANSWER THEM

Q Why would a controller reset ever lose inventory data?

A Because the PLC's counter restarts at a low number after a power-cycle, a new snapshot could look like an old one and get discarded — Ring distinguishes them by also matching on the PLC's event timestamp.

Q Is this watertight?

A It is robust in normal operation; the one honest edge is if the PLC ever emitted identical timestamps for two genuinely different post-reset events, which could merge them. That's an accepted, narrow edge, not a routine failure.

Q Did changing this risk the existing inventory rows?

A No — the rebuild preserves all rows, only runs when the old design is present, and is wrapped in a savepoint so an interrupted rebuild can't leave the table missing.

Q Can I see and export these numbers?

A Yes — the snapshots are exportable through the Inventory Snapshots export, so you can reconcile them in a spreadsheet.

Your tank, ingredient, and recipe names — and language — stay put forever

WHAT IT IS

When you rename a tank, an ingredient, or a recipe — and when you choose the operating language — those names and that choice are remembered **in the database**. They survive reinstalls and rebuilds, and they update live across every open screen instead of getting lost.

WHY IT MATTERS

On the floor, the plant's own names ("Voorraadtank," "Doseertank") are how operators navigate. If a reinstall or a build wiped them, you'd be re-typing them and risking mistakes. Storing them in the database means they stick, and a software rebuild that wipes local config files can't reset your language or names.

HOW IT WORKS

Tank, group, and recipe names live in dedicated database tables that update-in-place and then **notify open screens to refresh live**. The chosen language and first-run-wizard status live in a generic settings table inside the database — placed there precisely because a rebuild wipes the local config file but never the database. All these reads are failure-safe: if anything is missing they return a sensible fallback rather than crashing the startup.

VERSUS THE OLD RS-360 SYSTEM

RS-360 scatters these names across flat INI/text files and Paradox tables, fixes language per-install via a code-page file with no live refresh, and spends thousands of lines on config-file marshalling alone. Ring keeps the names and language in the database with live cross-screen refresh and failure-safe reads.

STATUS & HONEST CAVEATS

Status: Live and on a real screen. The edit screens are wired, seven views refresh live on a name change, and the persistence is verified.

Watch-out: The name tables ship **empty**, so your plant's actual Dutch names appear only after the one-time legacy import is run — until then you'll see defaults. Also, an earlier draft named specific legacy files (`ingredients.txt` , `formulas.txt`); only the tank names file and the INI directory are directly evidenced, so cite "INI/text files" generally rather than naming files you can't show.

LIKELY QUESTIONS & HOW TO ANSWER THEM

Q If I reinstall Ring, do I lose all my renamed tanks?

A No — names live in the database file, not in config files, so they survive reinstalls and rebuilds.

Q Why do I see generic names instead of our real tank names on a fresh install?

A The name tables ship empty by design; running the one-time legacy import loads your plant's actual Dutch names, after which they persist.

Q If I rename a tank on one screen, do other screens update?

A Yes — a rename notifies open screens to refresh live; seven views subscribe to that, so the change appears across the HMI without a restart.

Q Could a missing name crash the app on startup?

A No — every name and setting read is failure-safe and returns a fallback rather than throwing, specifically to protect the startup path.

A glance-from-15-feet look, designed to re-skin by swapping one file — for screens already migrated

WHAT IT IS

One consistent visual language across the HMI — colors, spacing, buttons, status indicators — defined once with meaningful names like "danger" and "surface" rather than scattered hard-coded color codes. Because the look is centralized, screens built on it could be re-skinned (for example into a dark control-room theme) by swapping one definition file rather than rewriting every screen.

WHY IT MATTERS

An operator reading a screen from 15 feet away needs calm, consistent numerals and state colors, with saturated red reserved for genuine live danger. Defining the look in one place means a future restyle is a config change for those screens, not a line-by-line rewrite, and the look stays consistent as new screens are added.

HOW IT WORKS

Ring loads its visual styles in a strict, documented order, ending with a self-contained **semantic token layer** — a single file that defines the named colors and styles (Surface, Card, Accent, Success, Warn, Danger) and the type, spacing, and component styles every migrated screen draws from. It's built on the MahApps.Metro UI toolkit, with an app-wide fix for a known button-text clipping issue.

VERSUS THE OLD RS-360 SYSTEM

RS-360 is a 2002 fixed-pixel design with hand-built per-tank menus and no design system — each screen styled on its own. Ring centralizes the visual language into a single named token source that the migrated screens share.

STATUS & HONEST CAVEATS

> Status: Live and on a real screen — the visual language and the token layer are shipped and visible on every screen today. But this is the one feature in the chapter to phrase carefully.

> Watch-out: There is **no dark theme to turn on today** — only the architecture that would make one a config swap. And token migration is **incremental**: the app's startup file and some unmigrated screens still carry hard-coded color codes, so a full re-skin today would still need those screens migrated first. So do **not** say "a dark theme you can turn on" or "the whole HMI re-skins as a config swap today." The accurate framing: "re-skinnable by swapping one file **for screens already on the Ring tokens**; a full dark theme is a future config swap, not a toggle that exists now." What *is* real and visible everywhere today: the 15-feet visual language, the button-clip fix, the standard progress styling, and token-driven navigation.

LIKELY QUESTIONS & HOW TO ANSWER THEM

Q Show me the dark theme.

A There isn't one to toggle today — what's shipped is the token architecture that makes a dark theme a future swap of one definition file. I'd rather be straight about that than demo something that doesn't exist.

Q So could you re-skin the whole HMI by changing one file?

A For the screens already migrated to the named tokens, yes — that's the design. Some screens still carry hard-coded colors, so a *full* re-skin would need those migrated first; it's not a single-file swap across the entire HMI yet.

Q What do I actually get from this today, then?

A A consistent, readable-from-15-feet look on every screen, the uppercase button-clipping fix applied app-wide, and a single source of truth for the migrated screens — plus the option of a future theme delivered as a config change rather than a rewrite.

Q Why does it matter that colors have names instead of codes?

A Because "danger" means one thing everywhere, so a change to the danger color updates every migrated screen at once, and operators see consistent meaning rather than ten slightly different reds.

Fixes and features arrive sooner, with a contained blast radius

WHAT IT IS

The app is assembled from many small, swappable building blocks rather than one giant tangled program. An engineer can replace or fix one block — say an email rule or an alarm channel — without disturbing the rest, so changes ship faster and with less risk of breaking something unrelated.

WHY IT MATTERS

Over the plant's life you'll want fixes and new features. In a tangled program, every change risks rippling somewhere unexpected. This architecture keeps the "blast radius" of any change contained, which means lower-risk, faster delivery — and it doesn't lock you to one developer.

HOW IT WORKS

Ring uses a standard **dependency-injection container** (a Microsoft library that wires the building blocks together). At startup it registers the whole graph — **39 data-access blocks** (one per table area), plus logging, configuration, alarm escalation, email delivery, the historian, and more — behind named interfaces, so any one can be swapped for another without touching the screens. The layering is a clean separation of screens, services, and data access.

VERSUS THE OLD RS-360 SYSTEM

RS-360 wired screens straight to database columns and used shared-memory communication across 21 interdependent sub-programs — a wide blast radius where a change anywhere could ripple anywhere, with no service abstraction. Ring wires interfaces through a real container, so blocks are swappable and the blast radius is contained.

STATUS & HONEST CAVEATS

Status: Live — an architecture guarantee underlying every screen. Not a screen you click.

Watch-out: Two honest nuances to own. It is a real container *plus* a small "service locator" bridge that the code itself labels as a migration-era step toward full constructor-based wiring. And the PLC tag readers/writers are deliberately created directly rather than wired through the container, because they take per-tag arguments. Both are documented, defensible choices — present them as deliberate, not as gaps.

LIKELY QUESTIONS & HOW TO ANSWER THEM

Q Why should an owner care about "architecture"?

A Because it determines how fast and how safely you can get fixes and features — small swappable blocks mean an engineer changes one thing without endangering the rest.

Q Is "39 building blocks" a real number or marketing?

A It's exact and verified — there are 39 data-access blocks, each behind its own interface, registered at startup.

Q Is this a pure modern design or are there compromises?

A There's one honest compromise: alongside the real container there's a small "service locator" bridge the code openly marks as a migration-era step, and PLC readers are created directly because they take per-tag arguments. Both are documented, deliberate choices.

Q Does this lock me to your developers?

A No — it's standard Microsoft dependency injection and a clean layered design, which any competent .NET engineer can read and extend.

A screen that never freezes, and data that holds up to a security audit

WHAT IT IS

Two things. First, the display stays responsive no matter how busy the PLC is, because talking to the controller always happens on **background workers**, never on the screen's own thread. Second, the data stands up to a

security audit because every database query uses safe **parameterized commands** — values are passed separately from the query text, so malformed or malicious input can't corrupt or expose data.

WHY IT MATTERS

A frozen HMI mid-shift is dangerous and erodes trust. And "could a bad value or an attacker break or read our data?" is a question a serious buyer's IT team will ask. This addresses both.

HOW IT WORKS

PLC reading runs on background timers that fill thread-safe **snapshot holders**; the screen reads those snapshots on its own gentle timer (every half-second to two seconds) and never touches the PLC directly. All pollers stop when the window closes, and a background timer self-recovers if the PLC or a network switch was still booting when Ring started — no operator has to click "Retry." On the data side, queries use parameterized commands throughout — a code scan found hundreds of parameterized bindings and only one harmless exception, and the security review rated every repository clean.

VERSUS THE OLD RS-360 SYSTEM

RS-360 wired screens straight to database columns over the old engine with bespoke communication and shared memory, with no documented threading model or query-safety discipline, and a data engine with no concurrency guarantees. Ring uses a documented background-snapshot model and uniformly parameterized queries, audited clean.

STATUS & HONEST CAVEATS

> Status: Live — an engineering guarantee behind every screen.

> Watch-out: An earlier draft said "all **40** repos parameterized," which clashes with the rest of the chapter's count of 39. Say "**every repository** parameterized" (or "all 39") to stay consistent. The "holds up to a security audit" line leans on a testing review document; what's verified directly in code is the parameterization itself, so anchor on that. The exact binding count drifts as new tables are added — quote "hundreds, with only one benign exception," not a frozen number.

LIKELY QUESTIONS & HOW TO ANSWER THEM

Q What stops the screen freezing when the PLC is slow or busy?

A All PLC communication runs on background workers that fill snapshots; the screen only reads those snapshots, so it never blocks on the controller.

Q What if the controller or network switch is still booting when Ring starts?

A A background timer automatically re-probes and recovers a never-started state, so the screen comes alive on its own without anyone clicking Retry.

Q **Is the database safe from injection attacks or bad input?**

A Yes — every query is parameterized (values passed separately from the query text), verified by a code scan that found only one harmless exception, and the security review rated the repositories clean.

Q **Is it "all 40 repositories" — or 39?**

A It's every repository — 39 of them; the "40" was a copy slip from a source doc. The substance, hundreds of parameterized bindings with one benign exception, is verified either way.

Faster, cheaper support — and months of unattended 24/7 uptime

WHAT IT IS

On the rare crash, the app writes a small **forensic report to disk first**, captured from three different kinds of failure, so support can diagnose it fast and cheap. And the operator's screen doesn't slow during a long stretch of PLC outages, because logging is built to never block the screen — and the app even repairs its own log folder if someone deletes it, so it runs for months unattended.

WHY IT MATTERS

An HMI runs around the clock. When something rare goes wrong, the difference between "we have a forensic file" and "we have nothing" is the difference between a quick fix and an expensive guessing game. And a logger that quietly dies or drags down the screen during an outage storm undermines exactly the reliability you bought it for.

HOW IT WORKS

A crash logger installs first thing at startup and hooks the **three** ways a .NET app can fail, writing one small file per crash to a crashes folder — while still letting the operating system's own dialog fire. Logging itself runs on a background writer draining a capped queue, so the screen's hot paths just hand off a line and move on, even during a PLC backoff storm; logs rotate at 10MB with backups, and the logger self-heals a deleted log directory. A UI-hang sentinel watches for the screen thread stalling.

VERSUS THE OLD RS-360 SYSTEM

RS-360 offers a communication log and disk logging but no modern instrumentation and no crash-report pipeline — no documented async self-healing logger and no hang sentinel. Ring adds three-hook crash capture, an async self-healing logger, and a UI-hang sentinel.

STATUS & HONEST CAVEATS

Status: Live — an engineering guarantee running on every session.

Watch-out: One honest detail if pressed on exactly what a *hang* report contains: on this runtime, the full screen-thread stack capture is a placeholder, so a hang report logs the thread's id, name, and state — not a complete call stack. Crash reports are full; only the hang-stack detail is limited. The marketing copy doesn't overclaim this, but be ready to state it plainly.

LIKELY QUESTIONS & HOW TO ANSWER THEM

Q If the app crashes at 3am, what do I have to work with?

A A small forensic file written to a crashes folder the instant it failed, captured from any of the three failure types — that's what makes support fast and cheap.

Q Does heavy logging during a comms outage slow the operator's screen?

A No — logging hands each line to a background writer with a capped queue, so the screen never waits on disk, even during a PLC backoff storm.

Q What if someone deletes the log folder?

A The logger detects that and recreates the folder on the next write, so logging doesn't quietly die — part of why it runs for months unattended.

Q Does a hang report give you a full stack trace?

A Honestly, not on this runtime — a hang report captures the thread's id, name, and state, not a full call stack; that stack capture is a placeholder. Crash reports themselves are complete.

No second copy fighting for the PLC, and clear guidance when one is frozen

WHAT IT IS

You can't accidentally end up with two copies of Ring fighting over the same PLC and database. Double-click the icon when Ring is already running and it brings the running copy to the front instead of opening a second one. If the running copy is **frozen**, it tells you the exact process to end rather than silently doing nothing.

WHY IT MATTERS

Two instances driving the same controller and writing the same database is a "split-brain" hazard that can corrupt data or double-send commands. And the common real-world frustration — "I double-clicked and nothing happened" — is replaced with a clear instruction.

HOW IT WORKS

At startup, before any database or PLC connection, Ring claims a named system-wide lock. A second launch detects the running copy and checks whether its window is responsive: if healthy, it focuses that window and exits quietly; if frozen or windowless, it shows a message naming the exact process number to end, and exits with a

distinct code a watchdog can read. Because this runs before any database or PLC connection, a duplicate can never open a second database handle or a competing poller.

VERSUS THE OLD RS-360 SYSTEM

No equivalent guard is documented in the legacy stack; this behavior — and the distinct exit codes for an automated watchdog — is part of the modern robustness layer RS-360 lacks. Ring adds a hardened guard that distinguishes a healthy from a wedged instance and tells the operator what to do.

STATUS & HONEST CAVEATS

Status: Live — a startup-time guard. It's operator-facing through a pop-up message on a second launch, not a navigation screen.

Watch-out: Be precise that this is a startup-time behavior surfaced via a message box, not a screen you browse to. The legacy contrast is honestly hedged ("no equivalent guard is **documented**"), so keep that hedge rather than asserting the old system definitely had nothing.

LIKELY QUESTIONS & HOW TO ANSWER THEM

Q Can two copies of Ring run at once and corrupt the data?

A No — Ring claims a system-wide lock before touching the database or PLC, so a second launch can't open a competing instance.

Q I double-clicked the icon and nothing seemed to happen — is it broken?

A That's exactly what this fixes — a second launch brings the running copy to the front instead of silently doing nothing.

Q What if the running copy is actually frozen?

A Then Ring tells you the exact process number to end and exits with a distinct code, so you get a clear instruction instead of a silent failure — and an automated watchdog can read that code too.

Q Are you certain the old system had no such guard?

A We've stated it honestly: no equivalent guard is *documented* in the legacy stack. We're not claiming more than the evidence shows.

Any .NET contractor can build it, and every release is checksum-verifiable

WHAT IT IS

You are never locked to one developer's machine. The whole app builds with standard, freely available Microsoft tools; the screen layouts are plain text files an engineer can read, compare, and review like any other code; and a

packaging script produces a release plus a **checksum** (a fingerprint that proves a file wasn't altered) so you know exactly what you're installing.

WHY IT MATTERS

Software you can't rebuild is a hostage situation. If your developer disappears, you need any competent contractor to pick it up, build it, review changes, and ship a verified release. This guarantees that, and the checksum reduces the risk of installing the wrong or tampered build.

HOW IT WORKS

The project uses a standard, classic-style build with **pinned package versions** (every dependency locked to an exact version) and a deterministic compiler setting, built with the standard Microsoft build tools. A release script stages the build, zips it, and emits a per-file **SHA-256** checksum manifest (SHA-256 is a standard fingerprinting algorithm). Helper scripts cover watchdog and crash-dump deployment. The screen layouts are text files you can compare line-by-line.

VERSUS THE OLD RS-360 SYSTEM

RS-360 has no build documentation and no automated build pipeline; its project files hardcode one developer's absolute folder paths, and its forms are dozens of binary files you cannot compare line-by-line — so rebuilding means reconstructing a 2002 machine. Ring uses standard build tools, pinned packages, text layouts you can diff, and a checksummed release.

STATUS & HONEST CAVEATS

Status: In progress — honestly marked "progress," not "shipped." The local build is fully reproducible today; broader automated build-and-test in the cloud (continuous integration, or "CI") is still on the roadmap, currently a single workflow.

Watch-out: Two honest nuances. First, "any .NET contractor" needs the standard Microsoft build tools installed (the same toolchain any .NET shop has, not a bespoke 2002 machine like the legacy required) — so frame it as "any contractor with the standard MS toolchain." Second, the deterministic setting removes compiler randomness, but **don't overclaim bit-identical builds across arbitrary machines** — the thing you can actually verify is the checksummed release package the script produces. Status "progress" is honest precisely because CI is minimal; defend "reproducible **locally** today, broader CI is roadmap."

LIKELY QUESTIONS & HOW TO ANSWER THEM

Q If our developer vanishes, can someone else build this?

A Yes — it builds with the standard Microsoft toolchain any .NET shop has, with all dependencies pinned to exact versions, and the screen layouts are text you can read and review.

Q **Why is this marked "in progress" and not "done"?**

A Because the *local* build is fully reproducible today, but the broader cloud build-and-test pipeline (CI) is still minimal — a single workflow — and expanding it is roadmap. We'd rather mark that honestly.

Q **Can I verify I'm installing exactly the right build?**

A Yes — the release script emits a SHA-256 checksum for every file and the package, so you can confirm nothing was altered before you install.

Q **Are the builds bit-for-bit identical on any machine?**

A We don't claim that — the deterministic setting removes compiler randomness, but the thing you actually verify is the checksummed release package the script produces, which is what proves what you're installing.

Q **Why does "text layouts you can diff" matter to me?**

A Because it means any engineer can review a change line-by-line in normal code-review tools, unlike the old system's binary form files, which you simply couldn't compare.

Chapter 7 — A Reliable, Honest Connection to the Controller

Every feature in this chapter is about one thing: the link between the operator's screen and the controller that actually runs the plant. Two pieces of vocabulary you must be comfortable saying out loud. The **PLC** (Programmable Logic Controller) is the rugged industrial computer that physically opens valves, runs pumps, and drives the mixer — it *is* the plant's brain, and Ring does not change a line of its program. The **HMI** (Human-Machine Interface) is the operator's screen — and that is what Ring is. Ring is a brand-new HMI that talks to the *same untouched* Allen-Bradley CompactLogix PLC the old 2002 screen, "RS-360," used to talk to. We replaced the window, not the brain behind it.

The real floor problem is trust in what that window shows. An HMI is just a viewer; it reads values out of the PLC and paints them. The dangerous failure is not a screen that goes blank — everyone notices that. It is a screen that keeps showing a number that *looks* live but is actually frozen, while the operator makes a real decision on it. A second floor problem is the morning ritual: after a power blip the old screen often had to be restarted by hand before anyone could see the plant. This chapter's features attack both problems — they make the connection honest about its own health, and resilient enough to heal itself.

A few more terms you'll meet below, defined once. A **tag** is a named value living inside the PLC (for example, a tank level). A **heartbeat** is a special counter inside the PLC that ticks upward *only while the control program is actually running* — a frozen program stops ticking. **EtherNet/IP** is the modern, open, industry-standard language Allen-Bradley controllers speak over ordinary Ethernet network cable. **Read-only mode** is a master safety switch in Ring that blocks every command to the controller, so the screen can watch the live plant without being able to change it.

One honesty rule that governs this whole chapter, because a sharp engineer *will* test it: when Ring "reads" the plant it does send small request messages over the network (that is how reading works on any system). What read-only mode blocks is **write commands** — anything that could change the process. So we never say "zero bytes." We say "zero write commands," and that is exactly true: on-site on 2026-06-10, Ring read 133 of 133 live tags and issued nothing that could move a valve.

Knowing the difference between unplugged, frozen, and running

WHAT IT IS

A simple connection light has only two states: green (connected) or red (disconnected). Ring's status is smarter. By watching the PLC's heartbeat counter, Ring can tell apart three genuinely different situations and name each one in plain English on an always-visible status banner: **unplugged** (the controller isn't answering at all), **frozen / wedged** (the controller answers the network but its control program has stopped, so the heartbeat is stuck), and **running** (answering *and* the heartbeat is ticking up). It also has a short "stale" grace state for a momentary hiccup, so a one-second blip doesn't trigger a false alarm.

WHY IT MATTERS

The most dangerous case on any plant floor is the middle one: the network is perfectly healthy, so a naive screen stays green, but the control program has actually halted and every number on the screen is silently frozen. Ring calls that out by name ("PLC reachable but heartbeat stopped — controller logic not running") instead of showing

a comforting green light. The flip side is fewer wasted maintenance calls: when the heartbeat is genuinely ticking, Ring says so, so nobody gets dispatched to chase a "PLC is down" that isn't.

HOW IT WORKS

Inside Ring a small piece of always-on logic (a "state machine" — it's always in exactly one named state and moves between them by fixed rules) checks the heartbeat value about once a second. If the value *changed* since last time, the state is "Connected." If it answered but the value is unchanged for more than 4 seconds it goes "Stale," and past 12 seconds "Starved" (reachable but wedged). If nothing answers at all for 12 seconds, or three reads in a row fail, it goes "Disconnected." All of that timing runs off a forward-only stopwatch, not the wall clock, so someone changing the PLC's date/time mid-batch cannot trick Ring into hiding a real comms loss.

VERSUS THE OLD RS-360 SYSTEM

Be careful and precise here, because this is the one place an original author can push back. The old RS-360 was **not** blind to a frozen heartbeat — its comms module did flip a heartbeat indicator to "not OK" when the counter stuck, and even had a yellow transitional color. So do **not** claim the old screen "had no liveness model" or "let operators trust numbers that were already dead." Ring's honest, defensible advantage is that it separates "unplugged" from "frozen" *by name* where the legacy collapsed both into a single binary OK/not-OK; it adds the 4-second blip tolerance; it runs the timing on a tamper-proof forward-only clock; and — uniquely — it *uses* that state to actively block commands, not just to color a light.

STATUS & HONEST CAVEATS

Status: Live and on a real screen. The four named states each drive their own color and operator text on the always-visible top banner, verified end-to-end in the code and live on-site.

Watch-out: Two things to disclose before someone else finds them. (1) The legacy comparison: the old system *did* have a heartbeat-freeze indicator, so frame Ring's win as "tells unplugged from frozen by name, with a tamper-proof clock and write-gating," not as "the old one had nothing." (2) The honest edge case already baked into our own description: a controller that is reachable but *deliberately paused* (its main task inhibited) also stops ticking the heartbeat, so Ring shows it as Stale/Starved/Disconnected rather than green — correct behavior, but worth saying plainly. The three-way distinction was verified live only since 2026-06-10, when the heartbeat was confirmed advancing.

LIKELY QUESTIONS & HOW TO ANSWER THEM

Q Didn't the old screen already show a heartbeat status — so what's actually new here?

A Yes, RS-360 had a single OK/not-OK heartbeat light, and we don't pretend otherwise. What's new is that Ring splits the bad case into two named states — "Disconnected" (unplugged) versus "Starved" (reachable but the program stopped) — adds a short tolerance window so a one-second blip isn't a false alarm, and feeds that state into a rule that blocks commands. It's a sharper, action-driving version of an idea the old screen had only in rough form.

Q Could someone fool it by changing the PLC clock or by an NTP time jump mid-batch?

A No. Every elapsed-time calculation runs off a forward-only stopwatch baseline, not the wall-clock date/time, so setting the clock backward or a network time correction cannot suppress comms-loss detection. That was a deliberate design choice.

Q If the controller is just sitting idle between batches, will it falsely scream "frozen"?

A Only if the control *program* itself has been paused. The heartbeat ticks whenever the program is scanning, batch or no batch, so a normally-running idle plant reads "Connected." The honest exception is a controller whose main task is deliberately inhibited — that genuinely stops the heartbeat, and Ring correctly flags it rather than showing a false green.

Q How fast does it notice a real loss?

A Roughly: a momentary stall is tolerated up to 4 seconds (shown "Stale"), a wedged-but-reachable controller is flagged by about 12 seconds ("Starved"), and a full silence is declared "Disconnected" after 12 seconds or three consecutive failed reads, whichever comes first.

Commands only reach a controller that's provably alive

WHAT IT IS

This is a background safety rule, not a screen. Even when Ring is allowed to send commands, it will only actually send one if it has *just* confirmed the controller is fully running. If the heartbeat so much as hesitates, the command waits rather than firing into a possibly-frozen controller. Crucially, reading the plant uses a more forgiving rule than writing to it, so the screen keeps updating smoothly through a tiny blip while commands stay strictly protected.

WHY IT MATTERS

A command sent into a wedged or transitioning controller can be lost or misapplied — a batch start, a formula selection, a setpoint, an agitator command, an alarm silence. By refusing to write unless the controller is demonstrably running, Ring guarantees that commands land only where they can actually be acted on. And because reads use the looser rule, the screen doesn't flicker to "disconnected" every time the network twitches.

HOW IT WORKS

There are two deliberately different gates. The **read** gate allows polling in either "Connected" or the brief "Stale" state — a 4-to-12-second blip must not freeze every value on the screen (an earlier version had a bug where one hiccup froze all readings during a live batch). The **write** gate is stricter: it allows a command *only* in the full "Connected" state, because "Stale" is not proof the controller's program is advancing. That one difference — permissive reads, strict writes — is the load-bearing safety rule, and it's enforced at the single point where Ring would actually send a command.

VERSUS THE OLD RS-360 SYSTEM

The legacy screen's heartbeat status was, as far as we can tell, used for *display* — it did not gate writes on a verified-advancing scan, so in principle a command could be issued toward a controller that wasn't actually

running its program. Ring turns liveness from a light into an enforced precondition: it writes only when the controller is provably alive, and that guarantee is kept separate from the more relaxed read rule.

STATUS & HONEST CAVEATS

Status: Live and enforced in the background — a safety rule in Ring's controller-communication layer, not a screen of its own. It is active and checked at the single write point, not a dormant idea.

Watch-out: Today, on top of this gate, Ring also ships with the master read-only switch ON, which suppresses *all* commands globally regardless of heartbeat. So at this moment no command fires under any condition — the heartbeat write-gate is the safety layer that takes over the day you enable writes. Also: we've confirmed Ring's behavior in code; we did not crack open the legacy write path line-by-line, so phrase the legacy comparison as "the old heartbeat status was display-only" rather than asserting exactly how its writes behaved.

LIKELY QUESTIONS & HOW TO ANSWER THEM

Q If this is so important, why can't I see it on a screen?

A Because it's a guardrail, not a gauge — it lives in the communication layer and runs on every command automatically. You can *witness* it indirectly: the communication monitor (covered later in this chapter) logs a blocked command with the reason "PLC link unavailable (heartbeat)" if one is ever held back.

Q With read-only mode on, isn't this gate doing nothing right now?

A Correct, and that's by design. Read-only mode blocks every command first, before this gate is even consulted, as the cutover safety posture. This heartbeat gate is the next layer down — the protection that remains in force the day you flip writes on, so a command still can't land in a frozen controller.

Q Won't waiting on the heartbeat make commands feel sluggish?

A No. In normal running the controller is in the "Connected" state continuously, so commands go straight through. The gate only ever delays a command during the exact window when the controller's liveness is in genuine doubt — which is precisely when you'd want it to wait.

Q Why are reads allowed during "Stale" but writes aren't — isn't that inconsistent?

A It's intentional, and it's the whole point. Showing a value that's a few seconds old during a brief blip is harmless and keeps the screen stable; *sending a command* into that same uncertainty is not harmless. So we tolerate stale reads and forbid stale writes.

A stale number never poses as a live one

WHAT IT IS

Ring refuses to display an old, frozen reading as if it were current. If the controller stops answering, or only partly answers, the affected numbers are flagged as out-of-date — on the live tank cards a value that's gone stale visibly blanks out to a dash ("—") instead of sitting there looking fresh and green. The screen would rather show you "I don't currently know" than a confident lie.

WHY IT MATTERS

The worst operator decision is one made on data that looks current but is minutes old. Tank levels, weights, temperatures, step state — if any of those silently freeze during a comms outage and keep displaying, an operator can act on a dead number. This feature removes that trap: the numbers are either genuinely current or visibly marked stale.

HOW IT WORKS

Each batch of readings Ring takes is stamped with the exact time it was captured and is never edited afterward. When Ring asks the consumer screens "is this fresh enough?", it answers using that capture time against a freshness window. The important fix is in the failure path: if the big bulk read of the controller's data array fails, Ring falls back to reading just the low-numbered values individually — and if *every* one of those also fails, it reports the read as failed and does **not** re-stamp the old cached values with a fresh timestamp. It also tracks the case where only the low values refreshed, so any screen relying on a higher-numbered value (such as the inventory-update word) correctly still sees it as stale.

VERSUS THE OLD RS-360 SYSTEM

A legacy HMI that simply caches the last reading and keeps painting it has no concept of *per-field* freshness. It's fair to say the old system had a system-wide comm OK/not-OK status — so phrase the contrast as "no per-tag freshness," not "no notion of staleness at all." Ring adds field-by-field freshness *and* a poller that flatly refuses to re-stamp old data as new — an explicit defense against exactly the frozen-but-live-looking trap.

STATUS & HONEST CAVEATS

> Status: Live and on a real screen. On the live tank ("Make Ready") cards, a weight that goes stale or non-physical de-renders to a dash with a stale badge — verified wired in the code.

> Watch-out: The proven, claimed consumer here is the live tank cards. A *separate* set of freshness-dimming converters for some dashboard summary tiles is not the same thing and is not fully wired everywhere — so defend "the live tank readings blank out when stale," and don't over-promise that every dashboard tile dims. And remember the legacy framing: say "no per-tag freshness," not "the old screen had no idea about comms health."

LIKELY QUESTIONS & HOW TO ANSWER THEM

Q Show me — what does an operator actually see when a reading goes stale?

A On the live tank card the number is replaced by a dash and marked stale, instead of continuing to show the last value as though it were current. The operator sees, unmistakably, "this reading is not live right now" rather than a frozen-but-green figure.

Q How do you know a number is stale rather than just unchanged because nothing moved?

A We don't infer it from the value staying the same — we track *when each reading was captured*. If a fresh, successful read didn't happen inside the freshness window, the value is stale regardless of what it shows. A genuinely still tank that's being read successfully stays live.

Q What happens if only part of the data comes back?

A Ring handles that explicitly. A partial read refreshes only the values it actually got, and any screen depending on a value that *didn't* refresh correctly continues to see it as stale. If nothing comes back at all, the whole read is marked failed and the old data is not re-stamped as new.

Q Does every screen in Ring do this, or just the tank cards?

A The proven, shipped consumer is the live tank cards. We're honest that per-field freshness isn't uniformly wired into every summary tile yet, so the defensible claim is the live tank readings — which are the ones an operator acts on most directly.

It reconnects itself — no morning restart

WHAT IT IS

If Ring starts before the PLC is ready — common first thing in the morning or after a power blip — or if the network drops, Ring keeps quietly re-checking and reconnects on its own the moment the controller comes back, with no operator clicks. There's also a "Retry now" button on the status banner to force an immediate attempt.

WHY IT MATTERS

It kills the morning ritual and the help-desk call. The old pattern was: plant powers up, the screen came up before the controller was reachable, and someone had to restart the HMI by hand to get the plant back on screen. Ring just comes back by itself when comms return — set-and-forget reliability.

HOW IT WORKS

At startup Ring quietly tests whether it can reach the PLC over the network, and it does this *off* the part of the program that paints the screen, so the window never freezes while it tries. If the controller isn't reachable, Ring schedules a background timer that re-runs the exact same connect-and-start sequence roughly every 30 seconds until it succeeds, then shuts that timer off. Overlapping attempts are prevented, and the "Retry now" banner button runs the identical path on demand. A subtle crash that an abandoned connection attempt used to cause has been closed off.

VERSUS THE OLD RS-360 SYSTEM

Honesty matters here, because the original copy overstated it. Do **not** claim RS-360 did "one startup probe then gave up" — its comms module actually ran a continuous timer that kept re-checking the driver, the link, and the heartbeat every tick. So the legacy was monitoring continuously, not abandoning the connection. The honest, still-valuable framing: Ring fixes the morning-restart ritual with proven, code-verified self-healing logic — present it as Ring's reliability, not as something the old system provably lacked.

STATUS & HONEST CAVEATS

Status: Live and on a real screen. The 30-second auto-reprobe and the "Retry now" banner button are both real, wired, and code-proven.

Watch-out: Two precision points. (1) Don't assert the old system "gave up" after one try — it didn't; keep the comparison to Ring's own proven behavior. (2) The literal "every 30 seconds" describes the case where Ring started *before* comms existed; if comms drop *after* Ring is already running and polling, recovery comes through the pollers' own retry-and-backoff instead — two mechanisms, same end result (it comes back on its own).

LIKELY QUESTIONS & HOW TO ANSWER THEM

Q So it really comes back without anyone touching it?

A Yes. If Ring started before the PLC was reachable, it retries about every 30 seconds and connects itself the instant the controller answers. If the drop happens while Ring is already running, its data pollers retry and snap back automatically. Either way, no operator action is required — and there's a "Retry now" button if someone wants to force it.

Q Are you saying the old system couldn't reconnect?

A No, and we're careful not to. The old comms module did keep monitoring and retrying continuously. What we're claiming is that Ring's self-healing is proven in code and removes the morning-restart pain — we're standing on Ring's own behavior, not on a weakness we can't substantiate.

Q Will all that retrying freeze the screen or hammer the network?

A No. The reachability check runs off the screen-drawing thread, so the window stays responsive, and attempts are spaced about 30 seconds apart and never allowed to overlap. The next feature on graceful backoff covers the broader "don't flood the network" behavior.

Q What's the "Retry now" button for if it's automatic?

A It's a courtesy for an operator who knows comms are back and doesn't want to wait for the next automatic cycle. It runs the exact same connect sequence, just immediately, and disables itself while the attempt is in progress.

Catching a frozen screen before it fools anyone

WHAT IT IS

A background watchdog inside Ring continually checks that the display itself is still responsive. If the screen ever locks up — keeps showing old data but stops actually updating — for a full minute, Ring detects it and writes a forensic record of the freeze. The goal is to make sure a hung screen is *caught and diagnosable* rather than silently misleading an operator.

WHY IT MATTERS

A frozen HMI that still shows plausible-looking numbers is the worst kind of failure, because nobody knows it happened until someone notices nothing is changing. This watchdog turns that silent failure into a detected, recorded event — giving the plant a forensic trail and a defined path toward automatic recovery.

HOW IT WORKS

A background timer posts a tiny "are you alive?" ping to the screen's drawing thread every 5 seconds; that thread records the ping only when it actually runs it. If the gap between recorded pings ever exceeds 60 seconds, the screen thread is wedged — the pings keep being sent but stop getting answered — and the watchdog writes a single, deduplicated hang report capturing how long the freeze lasted and what the screen was doing. The gap math is written to be immune to the timer counter wrapping around. By deliberate choice it is **log-only** today: an auto-restart wrapper script exists, but the automatic kill-and-restart is left switched off until that wrapper is verified on the actual plant computer.

VERSUS THE OLD RS-360 SYSTEM

The old single-threaded 2002 Windows HMI had no way to monitor its own responsiveness, so a hung legacy screen was a manual-discovery event — found only when someone noticed nothing was moving. That contrast is fair. Ring adds an internal sentinel that converts that silent failure into a detected, recorded one, with a staged path to automatic recovery.

STATUS & HONEST CAVEATS

> Status: In progress. Detection and the forensic hang log are shipped and active — the watchdog starts automatically when Ring launches. Automatic recovery (auto-restart) is deliberately not yet enabled.

> Watch-out: Be completely straight on this one. If asked "does it recover the screen by itself today?" the honest answer is **no** — it detects the freeze and writes a hang report, but it does not yet restart the screen, because the auto-restart wrapper hasn't been verified on the target machine. So today a hang yields a *logged but still-frozen* screen, not a self-healing one. That's why the status is "in progress," and that honesty is the point.

LIKELY QUESTIONS & HOW TO ANSWER THEM

Q If it detects the freeze, why doesn't it just restart the screen?

A It can — the restart wrapper is written — but we've intentionally left the automatic restart switched off until that wrapper is proven on your specific plant computer. Restarting an HMI automatically is exactly the kind of thing you verify before you arm it, so today it detects and records; recovery is the next, staged step.

Q So right now this doesn't actually fix anything?

A It does the hard half: it converts an invisible failure into a recorded, diagnosable one with timing and a snapshot of what the screen was doing. That's the difference between "the screen mysteriously froze last Tuesday and we'll never know why" and a forensic record you can act on. Full self-recovery is honestly still pending on-machine verification.

Q Could the watchdog itself slow the screen down?

A No. It's a tiny ping every 5 seconds and a single recorded timestamp; the threshold to even flag a problem is a full 60 seconds of unresponsiveness. It has effectively no cost in normal operation.

Q Won't it spam hang reports if the screen is wedged for a long time?

A No — the report is deduplicated, so one freeze produces one record, and it re-arms only after the screen recovers. You get one clean forensic entry per event.

An open, watchable link to the controller

WHAT IT IS

Ring talks to the PLC using a modern, open, widely-supported software library over standard Ethernet, instead of a closed vendor-specific driver wired into the old screen. And every read and write between Ring and the controller can be watched live in a monitor and saved to a daily log — so an engineer can see exactly what's flowing to and from the PLC.

WHY IT MATTERS

Two wins. First, you shed a proprietary dependency: the old screen's controller link was a closed component you'd have to keep alive forever. Ring uses an open, well-maintained standard you can support with ordinary skills and tooling. Second, total visibility: when something looks wrong, maintenance can open a live tail of every message to the controller and a saved daily record, rather than guessing.

HOW IT WORKS

Ring uses **libplctag** — a popular open-source library — to speak **EtherNet/IP** (the open Allen-Bradley network protocol) directly to the CompactLogix controller, with no proprietary middle-man software. A logging service keeps the most recent 500 messages in memory for the live monitor and can also write a per-day file to disk; those disk writes happen off the controller-communication path so they never slow it down, old logs are pruned automatically, and any logging error is swallowed so it can never disrupt a controller call. Notably, even

suppressed writes (the ones read-only mode blocks) are logged — so the monitor shows you exactly what Ring *would* have sent.

VERSUS THE OLD RS-360 SYSTEM

Use the corrected wording — the original copy was wrong on the specifics and a controls engineer would catch it. The old RS-360 reaches the CompactLogix through its **built-in Borland ActiveX/OCX comms module (called ASABTCP) over Ethernet** — an ActiveX/OCX is a closed, plug-in Windows software component bolted into the program. It carries old SLC-500-era "mirror" tag arrays kept alive purely as compatibility shims from the plant's earlier controller generation. It is **not** a DOS-era driver, **not** separate helper programs, and **not** the old DH+ network for the live link. Ring replaces that closed driver with the open libplctag EtherNet/IP stack and re-implements the comm monitor with batched, retention-managed logging plus visibility into suppressed writes.

STATUS & HONEST CAVEATS

Status: Live but partly hidden/limited. The open EtherNet/IP stack is fully live and is how Ring reads the plant today. The live communication monitor is real and wired — but it lives behind a supervisor-tier menu, not on the main operator screen.

Watch-out: Three precision points. (1) Get the legacy facts right: it's a Borland ActiveX/OCX comms module over **Ethernet** — never say "DOS-era," "sidecar/helper exes," or "DH+" for the live link (DH+ existed only in an inventory-only side module from the plant's older era). (2) The comm monitor is reachable from the setup/supervisor menu, not the operator home screen. (3) Don't confuse the working live *monitor* with the separate "Setup Communication" configuration screen, which is still a coming-soon stub — the monitor and its disk-logging toggle are the wired parts.

LIKELY QUESTIONS & HOW TO ANSWER THEM

Q Is this an open standard or just another proprietary library I'll be locked into?

A It's open. Ring speaks EtherNet/IP — the published, industry-standard Allen-Bradley network protocol — through libplctag, a widely-used open-source library. There's no closed vendor driver in the path, which is precisely the dependency the old screen forced you to keep alive.

Q Can I actually see what Ring sends to my controller?

A Yes. There's a live communication monitor that tails every read and write, and you can switch on a per-day saved log to disk. It even logs commands that read-only mode *suppressed*, so you can see exactly what Ring would have sent before any writes are enabled. The honest caveat: that monitor is behind the supervisor/setup menu, not on the operator home screen.

Q Your copy called the old driver "DOS-era" — that's not right, is it?

A You're correct to push, and we've corrected it. The old HMI is a Windows Borland C++Builder program, and its controller link is a built-in Borland ActiveX/OCX comms component over Ethernet — not DOS, not a separate helper program, not DH+. The real, accurate point stands: it's a closed, vendor-specific driver baked into the screen, and Ring replaces it with an open standard.

Q Does all this logging risk slowing down or destabilizing the controller link?

A No. The live view is just an in-memory buffer of recent messages, disk writes are batched on a separate task off the controller path, old files are pruned automatically, and any logging hiccup is contained so it can never propagate into a controller call.

Q Is the communication *setup* screen finished too?

A Be precise: the live *monitor* and its disk-logging toggle are finished and wired. The separate "Setup Communication" *configuration* screen is still a coming-soon stub. We don't claim that one is done.

Riding out an outage without flooding the network

WHAT IT IS

When the PLC isn't answering, Ring doesn't hammer the network or spam errors — it automatically slows its polling down, then speeds back up the instant the controller responds. At startup it deliberately spaces out its connections so it doesn't overwhelm a small controller, and it runs cleanly with no PLC attached at all, for office demos or operator training.

WHY IT MATTERS

An HMI that retries aggressively during an outage creates an error storm and can overload a modest controller — making a bad situation worse and harder to diagnose. Ring stays quiet and stable through an outage, recovers to full speed automatically the moment comms return, and can be run completely PLC-free for demos and training without errors.

HOW IT WORKS

After a couple of failed read cycles, each poller backs off — slowing the heartbeat check from about 1.5 seconds to 5, the fast snapshot from half a second to 5, and the big data block from 5 seconds to 8 — and it restores fast speed on the very next success. At startup a single coordinator staggers the seven per-tank pollers' first reads about 150 milliseconds apart, so they don't all open network connections at once; that matters because a CompactLogix controller of this class only allows a handful of simultaneous network sessions, and a sudden pile-up can briefly exhaust that limit (a known, self-recovering condition). The coordinator also skips a tick if the previous read is still running, and a shutdown timing bug that could leak errors during reconnects has been closed.

VERSUS THE OLD RS-360 SYSTEM

Phrase this one as a centralization win, not as a flat claim that the old system had none — it's an absence-of-evidence point. We found no documented adaptive backoff, startup staggering, or no-PLC graceful start in the

monolithic legacy HMI. Ring's contribution is concrete and defensible: one coordinator that centralizes start/stop, cadence, and recovery for all the pollers — something the old single-block design didn't offer.

STATUS & HONEST CAVEATS

Status: Live and enforced in the background — controller-communication infrastructure that's active on every run, with no operator screen of its own. All the backoff numbers and the 150-millisecond stagger are verified in code.

Watch-out: This is backend behavior, not a screen, so there's nothing to "show" directly. And keep the legacy comparison honest: say "Ring centralizes adaptive backoff and stagger in one coordinator" rather than asserting the old system definitely had none — we're claiming what Ring does, not proving a negative about RS-360.

LIKELY QUESTIONS & HOW TO ANSWER THEM

Q Where would I see this working?

A It's a background behavior, so you won't see a dedicated screen — but you'd see its *effects*: the screen stays calm during an outage instead of throwing errors, and it snaps back to live updates on its own when comms return. The communication monitor would also show the slower retry cadence during a loss and the return to full speed on recovery.

Q Why does it deliberately stagger its connections at startup — isn't that just slower?

A It's about 150 milliseconds total and it prevents a real problem: this class of controller allows only a handful of simultaneous network sessions, and if all seven pollers connect at the same instant they can briefly exhaust that limit. Spacing them out avoids a self-inflicted connection spike for a fraction of a second's delay.

Q Can I run Ring for a demo or training without a controller attached?

A Yes. With no PLC present Ring starts cleanly and runs without errors — it's designed for office demonstrations and operator training away from the plant, which the old screen wasn't built to do.

Q Are you sure the old system didn't already do this?

A We're honest that it's an absence-of-evidence point: we found no documented adaptive backoff, startup staggering, or no-PLC start in the legacy HMI. What we can prove is what Ring does — one coordinator centralizing backoff, stagger, and recovery for every poller — so we stand on that rather than on a claim about what the old code lacked.

Chapter 8 — Why Replace It Now

Before the feature list, get the one idea that holds this whole chapter together: **the old system is not broken today, and that is exactly the argument.** A quick vocabulary check first, because everything below depends on it. The **PLC** (programmable logic controller) is the rugged industrial computer bolted in the panel that physically opens the valves, runs the pumps, and drives the mixer. The **HMI** (human-machine interface) is the operator screen people actually touch — it shows what the PLC is doing and sends the operator's button-presses to it. **RS-360** is the 2002 HMI you run now. **Ring** (full name Ring RS3000) is the new HMI that replaces RS-360 while leaving the PLC completely untouched. A **batch** is one production run of one glue recipe. Your plant makes corrugating adhesive — industrial starch glue for cardboard — and it has run about 1,404 batches' worth of history.

The real floor problem this chapter addresses is not a bug. It is **a single point of failure that gets quietly riskier every month.** RS-360 is written in a programming tool from 2002 that you can no longer buy, patch, or hire for; its data sits in a database format no current software can open; and the whole thing can only be rebuilt on one specific old developer's PC with instructions that exist nowhere. It keeps running only because nothing has forced it to change. The day something does force a change — a dead hard drive, a Windows update, a corrupted file, or the one developer who knew it retiring — you would be doing emergency surgery on a 24-year-old system with no vendor to call.

"Why now" is the answer to that. Replacing it **while it still works** means you do it calmly, on your schedule, screen by screen, with the old system still installed and parked as a fallback — instead of in a panic after it has already failed. Everything in this chapter is a piece of that argument: get off the dead toolchain, get your data into a format you own, and make the system maintainable by any ordinary contractor.

One honest framing note that runs through the chapter: most of these are **platform and business-case arguments, not individual screens you click.** That is not a weakness — it is the point — but it means the right answer to "show me that on the screen" is sometimes "this one isn't a screen, it's the foundation under all of them," and you should say so plainly.

Buy out your single point of failure before it forces your hand

WHAT IT IS

This is the headline reason to move: you are removing the biggest hidden risk in the plant — a control screen built on tools that no longer have a maker, a buyer, or a support line — and doing it now, on a calm schedule, instead of after a failure forces a rescue. Because the old screen still runs the line today, this is insurance, not a fire drill.

WHY IT MATTERS

A plant can tolerate a slow, clunky system. What it cannot tolerate is a system that one day simply stops and cannot be rebuilt. RS-360 is one retirement, one disk failure, or one breaking Windows update away from that situation, and there is no vendor anywhere to call. Buying the replacement while the line is still running converts an open-ended emergency into a planned project.

HOW IT WORKS

RS-360 is locked to **Borland C++ Builder 6**, a programming tool released in 2002 with no security updates and no successor. Its screens are built with **VCL** (Borland's visual component library) and its reports with **QuickReport**, both long abandoned. Its settings live in **INI files** — plain-text configuration files — that are read and written by sprawling hand-written C++ code; the single module that handles that INI logic, `CommINI.cpp`, is about 7,466 lines on its own. Ring replaces all of that with mainstream, still-supported software: Microsoft **.NET 4.7.2** and **WPF** (the modern Windows screen-building framework), styled with the widely-used **MahApps.Metro** library, organized in a clean, standard structure (the "MVVM + Services + Repository" pattern — just the conventional way modern apps separate the screen, the logic, and the data). The move is done one screen at a time so it stays controlled and reversible.

VERSUS THE OLD RS-360 SYSTEM

Old: an end-of-life toolchain, a corruption-prone deprecated database, a build that cannot be reproduced, and a shrinking pool of people who even know these 2002 tools — it works only because it has not had to change, and changing it gets harder and riskier every month. New: mainstream, widely-supported software and an open database on a build any modern shop can reproduce, cut over deliberately with the legacy app left installed and parked, and a **documented rollback checklist with a ~10-minute target** if switchover day goes sideways.

STATUS & HONEST CAVEATS

Status: Shipped and verified — but understand this is a platform and architecture fact, not a single on-screen feature (UI = not applicable). Every legacy fact in it was checked against the actual source code and confirmed exactly.

Watch-out: Two phrasings to keep honest. First, the 7,466-line figure is the size of the C++ *code module that handles* the INI settings — not the size of a settings file; say "the INI-handling code module alone is about 7,466 lines," not "the config is 7,466 lines." Second, the ~10-minute rollback is a **documented target**, not a stopwatch-measured guarantee — the rollback document itself calls 10 minutes "an aspiration." Say "a documented rollback checklist with a ~10-minute target, and the old app stays installed," never "a guaranteed 10-minute rollback."

LIKELY QUESTIONS & HOW TO ANSWER THEM

Q If it still works, why spend money on it at all?

A Because the cheapest time to replace a single point of failure is before it fails, not after. Right now you can move calmly, screen by screen, with the old system as a fallback; after a failure you would be doing emergency recovery on a 2002 toolchain with no vendor to call, on the plant's timeline instead of yours.

Q Is the 2002 claim real or sales talk?

A It is real and we checked it in the code: all 21 of the legacy project files carry the internal marker "BCB.06.00," which is Borland C++ Builder version 6, released in 2002. That is not an opinion — it is stamped in the files.

Q Could moving the screen disturb the running plant?

A No. The PLC — the controller that actually runs the valves and pumps — is left completely untouched. Ring is only the operator screen that replaces RS-360; it talks to the same controller the old screen did. And it ships with every control command suppressed by a read-only safety gate until you deliberately turn writing on.

Q What part of this isn't finished?

A This particular argument is the platform itself, and the platform is shipped and verified. The unfinished items are disclosed elsewhere and honestly: loading the historical data into Ring, and a separately-quoted bit of controller-side cleanup. None of those change the fact that the new platform is on mainstream, supportable software today.

Own your recipes, batch history, and alarms instead of leaving them trapped in a dead database

WHAT IT IS

Your recipes, your batch history, and your alarm log currently live inside a 2002 database format that no current tool can open and that can corrupt silently with no recovery path. Ring's data instead lives in an open, single-file database you can read, back up, and export yourself — so a night-shift line stop never turns into permanent data loss.

WHY IT MATTERS

Plant data is the plant's memory: what we made, when, how much, and what went wrong. Today that memory is locked in a format that is awkward to read and prone to silent corruption, with no vendor to recover it. Owning your data in a portable, inspectable format means an auditor's "show me your records" is answerable, and a database hiccup is a backup-restore, not a catastrophe.

HOW IT WORKS

The legacy data is stored in **Paradox `.DB` files** — a 1990s database format — that, in normal operation, the old app reaches only through the **Borland Database Engine (BDE)**, deprecated middleware (a go-between driver) that Microsoft's modern Windows no longer ships and nobody maintains. To even get the data out, the team had to write a brand-new reader from scratch in modern code (`PdxReader.cs`) that opens those files directly, byte by byte, **without BDE**, precisely because no maintained driver exists. Ring stores everything in **SQLite** — an open, single-file database used in billions of devices — running in **WAL mode** (write-ahead logging, a crash-safety setting that protects the file if power drops mid-write), with an integrity check on restore, a proper online backup, and a one-click export to **CSV** (the universal spreadsheet format) with a picker that lets you choose exactly which tables and columns to export. The export button is reachable from the navigation bar, so an operator can do it without a developer.

VERSUS THE OLD RS-360 SYSTEM

Here is the honest, corrected before/after — and it matters that you get this right, because a controls engineer who knows the old system will catch an overclaim instantly. The old system **did keep** batch, step, shift, and alarm

history (1,404 batch headers, 16,776 step rows, 2,880 alarm records, 327 shift records were all pulled off-site). What it did **not** have was any safe, modern, openable home for that data: it is locked in a corruption-prone, dead Paradox/BDE format that needed a custom-written reader just to extract, with no maintained driver, no integrity checking, no continuous trend/time-series historian, no step-duration timeline, and no recovery tooling. Do **not** say "the old system recorded nothing" — that is false and easily disproven. The true and still-strong line is: "it kept the records, but trapped them in a dead engine you cannot safely open, back up, or recover, with no living trends, step timing, or operator-searchable access." Ring's side: portable single-file SQLite with WAL crash-safety, integrity check, online backup, and one-click CSV export with a column picker.

STATUS & HONEST CAVEATS

Status: Live and on a real screen — the SQLite engine and the CSV export are shipped and reachable from the nav bar — but your 1,404 historical batches are extracted to files, not yet loaded into Ring. The "own your data" engine is real and demoable today; importing the legacy history is a separate, signed-off cutover step that has not been run yet.

Watch-out: This is the single most likely place to get caught, so be precise. If the buyer says "show me my recipes and my 1,404 old batches inside Ring," the honest answer is that they are extracted to readable CSV/JSON files but **not yet imported** — Ring's recipe-name, batch-history, shift, and alarm-definition tables ship empty, and the import is a one-time, human-signed-off step that hasn't been performed. The one-click CSV export works on Ring's own current data, not yet on the historical batches. Frame it exactly as: "your data is already portable, inspectable, and fully extracted — loading it into Ring is a one-time cutover step we sign off together," never "your history is already live in Ring." Also soften the "can only be opened through Borland middleware" line: in normal operation that is true, but we proved it can be opened with a custom reader — so say "no maintained driver exists," not "can only ever be opened via BDE."

LIKELY QUESTIONS & HOW TO ANSWER THEM

Q Are my 1,404 historical batches in Ring right now?

A Not yet, and I won't pretend otherwise. They are fully extracted into readable, portable files today, and the import into Ring is a one-time step we run and sign off together at cutover. The point is the data is no longer hostage to a dead engine — it is out, readable, and ours.

Q You said the old data can only be opened by abandoned Borland middleware — but then you said you opened it. Which is it?

A Fair catch, and the honest version is the stronger one: in normal day-to-day operation the old app reaches that data only through the abandoned middleware, and there is no maintained driver for it anywhere. Recovering it required writing a custom reader from scratch — which we did — so the data is extractable, but only because we built the tool the market doesn't provide.

Q **Why is SQLite safer than what we have?**

A SQLite is an open, single-file database used in billions of phones, browsers, and devices, so it will be readable and supportable for decades. It runs with crash-safe write logging, checks its own integrity on restore, and supports proper online backups — none of which the old Paradox/BDE setup gives you. And you can export any of it to a plain spreadsheet file with one click.

Q **Did you verify any of this against our actual data?**

A Yes. The on-site drop confirmed the real legacy tables and their exact sizes, and the SQLite settings, backup, integrity check, and CSV export were all verified in Ring's source code, not just claimed. The export button is wired into the navigation bar where an operator can reach it.

Make changes any .NET contractor can do, not a 2002 build wired to one retiring PC

WHAT IT IS

Today, changing RS-360 is wired to one person's 2002-vintage developer PC with no written rebuild instructions, which makes every change slow, expensive, and one resignation away from impossible. Ring is built with standard, current tools any competent .NET contractor can pick up and modify with confidence.

WHY IT MATTERS

"Maintainability" sounds abstract until the day you need a small change — a new report, a tweaked screen — and discover the only person who can do it is gone and the build only exists on a dying machine. That is key-person risk and rebuild risk, and it quietly raises the cost and danger of every future change. Removing it means routine changes stay routine.

HOW IT WORKS

The legacy app is 21 interlocking sub-projects passing data through shared memory, totalling roughly 134 code files, 130 header files, and 97 screen-layout files — with two single files over 5,000 lines (the main form at ~5,647 lines and the INI handler at ~7,466). The project files **hardcode one developer's exact folder paths** (`C:\Projects\CB6\RS360\ ...`, with an even older `C:\Projects\CB3\ ...` reference still hanging around), and there is **no README, no build document, and no automated build setup anywhere** in the tree. On top of that, the **great majority of the screen files — 88 of the 97 — are stored in Borland's binary IDE format**, which can't be meaningfully compared, reviewed, or merged the way normal text can. Ring, by contrast, builds with **Visual Studio 2022's MSBuild** from **XAML** — the text format that describes WPF screens, which any developer can read, diff, and review — and ships with a production-readiness binder so a contractor knows exactly what they are looking at.

VERSUS THE OLD RS-360 SYSTEM

Old: no patch path and no second build environment — rebuilding realistically means reconstructing a specific 2002 IDE, the old database runtime, and third-party add-ons on one machine, and most of the screen files are binary-only so they can't be properly code-reviewed or merged. New: built with current Visual Studio tooling from diffable, reviewable text screen files, backed by a documented binder, so updates ship safely.

STATUS & HONEST CAVEATS

Status: Shipped and verified — like the platform argument, this is an architecture fact rather than an on-screen feature (UI = not applicable). Every counted number was checked against the actual files.

Watch-out: Be careful with the word "binary." It is **88 of the 97** screen files that are binary, not all of them — 9 are in a readable text format. If a buyer's engineer opens one of those 9 and finds it readable, you don't want to have claimed "all the forms are binary." Say "the great majority — 88 of 97 — are binary IDE files that can't be meaningfully reviewed or merged." Also, one minor item: the "release manifest" mentioned in the sales material wasn't independently opened in code — the production-readiness binder is verified, but lean on the binder, not the manifest, since the manifest isn't load-bearing to the point.

LIKELY QUESTIONS & HOW TO ANSWER THEM

Q If our developer retires, what actually happens with each system?

A With RS-360, you are in trouble: the build is tied to his specific PC with no written instructions, so reproducing it means rebuilding a 2002 environment from memory. With Ring, you hire any standard .NET contractor, hand them readable text source and a documented binder, and they make the change — that is the whole point of this item.

Q Are the legacy file counts and the "no build doc" claim real?

A Yes, all checked directly. The exact file counts, the two 5,000-plus-line files, the hardcoded developer paths, and the complete absence of any README, build document, or automated build were all confirmed by scanning the actual legacy tree.

Q Aren't all the old screen files unreadable binary?

A Most are — 88 of the 97 — and those can't be properly reviewed or merged. Nine are in a readable text format, so I'd never say "all of them." Even so, "the great majority are binary IDE-only files" is the accurate and damning version, and Ring's screens are all readable text.

Q Could a contractor break the plant making a change in Ring?

A Ring only drives the operator screen, never the controller logic that runs the valves and pumps — that lives in the untouched PLC. And Ring ships with all control commands suppressed by the read-only gate, so a contractor working on a screen can't issue a command to the process until writing is deliberately enabled.

Buy bounded, shrinking risk and stop paying for unbounded, growing risk

WHAT IT IS

This is the business case stated honestly: you are trading an open-ended, growing liability (an unsupportable 2002 system) for a known, scoped one (a planned cutover with a short tail of remaining work). The pitch is deliberately conservative — it sells risk-avoidance and supportability today, not a productivity miracle — with no invented payback period, uptime figure, or customer count to erode your trust later.

WHY IT MATTERS

Owners get pitched inflated ROI numbers constantly, and a sharp buyer punishes the first one that doesn't hold up. The strength here is the opposite: the case rests on a risk comparison anyone can see is true. Legacy risk is unbounded and growing — a database corruption with no recovery, a Windows update breaking an unpatchable runtime, the one developer retiring, none with a vendor to call. The new system's risk is bounded and shrinking — a known, catalogued cutover tail with remediation plans. That asymmetry is the close.

HOW IT WORKS

The case is laid out in the sales document's ROI/risk section. It deliberately avoids hard productivity claims and instead frames the value as risk avoidance: the dollars are the avoided cost of emergency Borland/database talent, the avoided cost of a Paradox recovery with no tooling, and a lower cost-per-change on a clean modern architecture versus surgery on binary legacy screens. The one external line item is quoted plainly and separately: the controller-side ladder-logic restoration ("ladder logic" is the program inside the PLC), estimated at roughly **$5,000–$15,000 of engineer time, quoted precisely per site and kept separate from the HMI**.

VERSUS THE OLD RS-360 SYSTEM

Old: the legacy platform's risk *is* the entire ROI argument — it cannot even produce a remediation list for itself, and emergency recovery has no vendor to call. New: the remaining work is catalogued with plans, and the one external line item is a precise, scoped quote, so you see a known number rather than a surprise.

STATUS & HONEST CAVEATS

Status: Shipped and verified as a business-case document — not an on-screen feature (UI = not applicable). The risk framing, the scoped $5,000–$15,000 controller line item separate from the HMI, and the "both sides catalogued with remediation plans" point all check out word-for-word against the source.

Watch-out: There is a **misquote in the marketing copy** you must fix before the meeting. The copy puts in quotation marks the phrase "*a sound modernization, not a productivity miracle*" — that exact phrase appears **nowhere** in the actual documents (zero hits when searched). What the document actually says (and you can safely quote) is **"insurance with upside, not a productivity miracle."** A technical buyer who opens the document to verify will not find the first phrase, which would undercut the very "you can check everything we say" trust this whole chapter is built on. Use the real quote.

LIKELY QUESTIONS & HOW TO ANSWER THEM

Q What's my payback period and uptime improvement?

A I'm not going to give you a manufactured number, and that's deliberate — this case is built on risk and supportability, not an invented productivity figure. The honest return is avoiding an unbounded, growing liability and lowering the cost of every future change; the bigger productivity gains are scoped for later versions, not promised today.

Q You quoted a line — let me read it. (Buyer opens the document.)

A Good, please do — and let me give you the accurate quote so you find it: the document frames the value as "insurance with upside, not a productivity miracle." Everything in this section is meant to be verifiable against that source, which is the whole point.

Q What's this extra controller-side cost, and is it hidden?

A It's not hidden — it's quoted separately on purpose. It's roughly $5,000 to $15,000 of engineer time to restore the controller-side process logic, priced precisely per site and kept distinct from the HMI so you see exactly what it is. That is the scoped, known tail of the cutover.

Q How do I know the legacy risk is really "unbounded"?

A Because every legacy failure mode I named has no vendor to call: a Paradox database corruption with no recovery tool, a Windows update breaking an unpatchable 2002 runtime, or the one developer retiring. The new system's risk, by contrast, is a finite list of catalogued items with remediation plans — that contrast is the entire argument.

Verify every claim against the code before you sign

WHAT IT IS

This is the trust mechanism: a complete positioning guide that ties every claim back to the actual code and audits, names the real gaps out loud, and forbids inventing numbers — so a technical buyer can check the pitch instead of taking it on faith.

WHY IT MATTERS

A skeptical buyer or controls engineer assumes vendors overstate. The fastest way to win that person is to hand them the receipts and point out your own gaps before they find them. A pitch that survives verification closes; one that can't be checked invites suspicion of everything.

HOW IT WORKS

The guide is a ten-section playbook (executive summary, the "burning platform" case, a legacy-versus-Ring comparison table, top reasons to buy, value by stakeholder, the actual pitch, objection handling, the ROI/risk narrative, a one-page leave-behind, and honest positioning notes). Its final section explicitly **separates what is shipped from what is roadmap**, and proactively discloses the three real gaps: no per-operator named-user audit trail yet, the modern dark-theme visual pass deferred to the next version, and the controller-side process

logic still needing a scoped cutover window. It also sets hard "do not" rules — no claimed uptime, ROI, payback, or customer count, and explicitly **not** claiming Ring is "already running a plant"; the truthful statement is that it was verified live against a controller at the bench. Every claim is cross-referenced to its grounding file.

VERSUS THE OLD RS-360 SYSTEM

Old: there is nothing to verify on the legacy side — no README, no build document, no collateral; the legacy tree is undocumented and locked inside the IDE and can't even produce a remediation list for itself. New: a grounded, code-cross-referenced positioning guide with an explicit gap list and "do not invent" rules, so a technical buyer can check the pitch against the source.

STATUS & HONEST CAVEATS

Status: Shipped and verified — and it is the strongest-grounded item in this chapter; the ten sections, the shipped-versus-roadmap split, the three disclosed gaps, and the do-not rules all match the source exactly. As a positioning document rather than a screen, UI is not applicable.

Watch-out: One precision point about how the demo stays safe. Say the demo is write-safe **because of the read-only safety gate**, not because the features themselves can't write. Several shipped features *can* write to the controller (operator-confirmed Hold/Resume/Reset, batch and shift close) — they're simply suppressed because Ring ships with all PLC writes turned off by default. The correct phrasing is: "the demo runs on shipped features with all PLC writes suppressed by the read-only safety gate, and the legacy app stays installed with a documented ~10-minute-target rollback." Also, the read-only on-site session **read 133 of 133 live tags** — but never say it sent "zero bytes." Reading tags necessarily sends network request packets. Say "zero write commands" or "not a single byte that could change the controller's state."

LIKELY QUESTIONS & HOW TO ANSWER THEM

Q How do I know you're not just telling me what I want to hear?

A Because the positioning guide cross-references every claim to the actual code or audit file, separates shipped from roadmap, and lists its own three gaps before you ask. You can open the source and check it — and where this very pitch had loose wording, we corrected it rather than defend it.

Q What are the gaps you're admitting to?

A Three, stated plainly: there's no per-operator named-user login audit trail yet, the modern dark-theme visual polish is deferred to the next version, and the controller-side process logic still needs a scoped cutover window. None are hidden — they're written into the guide's own gap list.

Q **Did you connect to a real controller, or is this all simulated?**

A We verified it live against a real CompactLogix controller at the bench, reading 133 of 133 live values. We are careful to say "verified live at the bench," not "already running a plant" — and it read those values without issuing a single command that could change the process, because the read-only gate suppresses every write.

Q **When you ran on the live controller, did you change anything in the process?**

A No — and not "zero bytes," because reading values does send small network request packets. What it did not send was **a single write command**. The read-only safety gate suppresses every command that could move a valve, pump, or mixer, so we read everything and changed nothing.

Q **Was the demo itself write-free because the features can't write?**

A The honest version: several shipped features *can* write to the controller, but the demo is safe because Ring ships with all writes suppressed by the read-only gate, and the legacy app stays installed as a fallback with a documented ~10-minute-target rollback. The safety comes from the gate, not from the features being incapable.

Chapter 9 — Where the Work Honestly Stands

This chapter is the honesty ledger. Everywhere else in this guide we explain what Ring *does*; here we tell you plainly what is finished, what is half-finished, what is built but not yet switched on, and what is still an open question for your own team. The whole point of this section is that you can win a hard meeting by being the most candid person in the room, because every line below has been independently checked and your engineer can re-check it.

A few terms first, since the rest of the chapter leans on them. The **PLC** (Programmable Logic Controller) is the industrial computer that physically runs the valves, pumps, and mixer that make your glue. The **HMI** (Human-Machine Interface) is the operator screen — that screen is what Ring is. Ring replaces the 2002 DOS-era screen ("RS-360") but runs on the *same untouched* Allen-Bradley CompactLogix PLC; **installing Ring does not change how your plant runs**. A **batch** is one production run of one glue recipe. A **tag** is a single named value inside the PLC (a temperature, a valve state). The **read-only gate** is a safety switch (`ReadOnlyMode=true`) that suppresses every command Ring could send to the PLC, so Ring can watch the live plant without ever being able to change it.

One honesty rule that runs through this whole chapter: when Ring read the live plant, it read 133 of 133 values **without issuing a single command that could change the process**. Reading still sends small network request packets — so never claim "zero bytes" — but it sent **zero write commands**. That distinction is the kind of thing a skeptical engineer will test, so we get it exactly right.

The status the auditors gave Ring is **GO-WITH-CONDITIONS**: a strong replacement, cleared to go live behind a short, written list of fix-first items. That is deliberately *not* the same as "everything is perfect." Throughout, **P0** means a critical fix, **P1** an important one, and **P2** a minor one. Each feature below carries a plain **Status** line and, where there is any way to get caught out, a **Watch-out** telling you the exact thing to be ready to defend and the safe way to say it.

An audited go-live decision with every condition named

WHAT IT IS

Instead of a vendor saying "trust us, it's ready," you get a written verdict from an independent review: Ring is a strong replacement for the old system, cleared to go live *once a short, named list of fixes is closed*. Every item on that list is written down with a plan to close it, so nothing is hand-waved.

WHY IT MATTERS

Replacing the screen that runs your glue kitchen is a real risk if you do it on optimism. This turns a leap of faith into a checklist you can hold the vendor to. You can see exactly what stands between today and a clean cutover, and watch that list shrink.

HOW IT WORKS

An independent audit compared Ring screen-by-screen against the old system and found roughly 80% of features at full parity or better. It then produced an enumerated punch list — a short set of critical, important, and minor items — each with a remediation plan. Connecting Ring to a live PLC clears the "can it read and write

at all" questions, but it does *not* clear the verification items still outstanding (a fresh side-by-side diff, deliberate fault-injection on the bench, a multi-day soak test, staged rollback media, and signed operator acceptance).

VERSUS THE OLD RS-360 SYSTEM

With RS-360 you inherit unbounded, quietly growing risk: decades of field history but no reproducible build, no security-patch path, and a corruption-prone Paradox/BDE database (an obsolete 1990s file format) that can fail without warning. With Ring, your only remaining risk is a known, scoped, shrinking cutover tail — with written remediation plans the old app could never even produce.

STATUS & HONEST CAVEATS

Status: In progress — this is an audited *decision*, not a screen. Ring is cleared to go live behind a short written list of conditions, and that list is shrinking but not yet empty.

Watch-out: GO-WITH-CONDITIONS is explicitly **not** "production-ready." The 48-72 hour endurance ("soak") test is still **unrun**, and a fresh-export diff, bench fault-injection, and signed operator sign-off are still outstanding. Be ready on three specifics: (1) the open-blocker count — present it as "roughly a dozen items still open," because the source document gives both a 13-row blocker table and its own higher "15 blockers / 45 gaps" header, so reconcile to "about 11 still open" rather than quoting one number as gospel. (2) The "~10-minute rollback" is *documented* but not yet executable as written — the rollback media still has to be physically staged at the controller cabinet. (3) The fixes closed on 2026-06-22 were all **P1 (important)** items; the one **P0 (critical)** fix — a false "batch started" message under read-only — was closed earlier, on 2026-06-18. Don't bundle them as "P0s closed on the 22nd."

LIKELY QUESTIONS & HOW TO ANSWER THEM

Q Is "GO-WITH-CONDITIONS" just a fancy way of saying "not ready"?

A No — it means an independent review judged Ring a strong replacement and cleared it to go live once a short, named list closes. The honest part is that it is *not yet* production-ready: the endurance soak is unrun and a few verification items remain, and those are all written down so you can hold us to them.

Q How many open items are actually left?

A About a dozen blockers, with several already closed in code. I'll give you the exact list from the gap analysis rather than a marketing round-number, and your engineer can open the document and count them.

Q You said critical bugs were fixed last week — were they?

A Honestly, the items closed on 2026-06-22 were all *important* (P1) fixes, like formula-name saving and friendly error dialogs. The one *critical* (P0) fix — a screen that falsely said a batch had started while in read-only mode — was closed on 2026-06-18. I won't conflate the two.

Q Can you actually roll back to the old system in ten minutes if cutover goes wrong?

A The procedure is documented and short, but it is not executable until the rollback media is staged at the controller cabinet — that's one of the listed conditions. We will dry-run it before go-live; until then I'd describe rollback as "planned and documented, staging pending," not "proven."

Q Who decides when a condition is cleared — you?

A Your engineer can verify each one independently. Every condition is a concrete, checkable item with file-and-line evidence, so the sign-off doesn't rest on our word.

A dated, auditable trail of every go/no-go decision

WHAT IT IS

A written, dated record of every safety decision made about Ring: it started as a deliberate **NO-GO** when a review caught a serious bug, became GO-WITH-CONDITIONS after the fixes, and earned a clean **GO** for a safe read-only session against your real plant. Your own engineer can read the whole trail.

WHY IT MATTERS

Anyone can claim "we tested it." Far fewer can hand you a paper trail that shows they *failed their own first review on purpose*, fixed what they found, and re-checked. That is the difference between a vendor's word and an auditable decision that survives scrutiny.

HOW IT WORKS

The first formal go/no-go review returned NO-GO: it caught one critical issue in the new split-batch feature plus five important ones. All were fixed and re-reviewed; the build came back clean. A later review — an adversarial double-check run by 23 separate review agents — issued a GO for a read-only live session, and confirmed the single most important safety property: that Ring's write-guard fails *safe*, suppressing every command to the PLC, with all 24 places in the code that could write to the controller short-circuiting to a logged, do-nothing "suppressed" path.

VERSUS THE OLD RS-360 SYSTEM

With RS-360 there is no readiness review and no documented verdict process at all; it "works" because it has not had to change, and its correctness becomes unprovable the day you actually need to change it. With Ring, every verdict, date, and fix is written down and reproducible.

STATUS & HONEST CAVEATS

Status: Complete — this is a documented, auditable record (the documents exist and are yours to inspect), not a screen feature.

Watch-out: If you quote test numbers, say "761 tests passed (0 failed, 0 skipped) on a clean build" — *not* "761 tests, zero warnings," because the cited review certifies a clean build, which is not the same as a warning-free build. Also be ready to state that the split-batch fix is in the code but deliberately **held back** from live use until a controls engineer signs off at the plant and a bench "write-and-observe" test confirms it.

LIKELY QUESTIONS & HOW TO ANSWER THEM

Q Why should I trust a review the vendor ran on itself?

A Because every verdict is dated and evidence-backed, and your engineer can re-open the same documents and re-run the same tests. We also failed our own first review on purpose and wrote down why — that's not what an overselling vendor does.

Q You started at NO-GO — what was the bug?

A The new "split a batch across two tanks" feature wrote the wrong tank reference (a button position instead of the actual tank number). It's fixed in code, but we've gated it: it stays disabled on the live plant until a controls engineer signs off and we verify it on the bench.

Q Did the read-only safety actually get verified, or is it just a setting?

A It was verified by an adversarial 23-agent double-check. The write-guard fails safe even with a missing or garbled configuration, and all 24 code paths that could send a command to the PLC short-circuit to a logged "suppressed" no-op.

Q Does "clean build" mean zero warnings?

A No, and I won't claim that. The review certifies a clean build with 761 tests passing — zero failed, zero skipped. "Zero warnings" is a stronger claim the document doesn't make.

About 80% screen-for-screen parity, with the gaps named

WHAT IT IS

A feature-by-feature comparison of every old screen against its Ring equivalent. About 80% of features are at full parity or better, with Ring beating the old system on trending charts, batch time-to-finish estimates, exports, formula history, and safety. The remaining 20% is split into deliberate scope cuts and a short list of genuine, named gaps.

WHY IT MATTERS

Your operators don't want to relearn their job, and you don't want to discover a missing capability after cutover. This map tells you exactly what carries over and exactly what doesn't — so there are no surprises on the floor.

HOW IT WORKS

An audit using 11 parity reviewers produced a comparison map with file-and-line evidence for each claim, not a marketing summary. It confirmed where Ring is a *superset* of the old system (batch ETA with stale-data warnings, full live trending, PDF/CSV export, filter presets, formula-history snapshots, the read-only safety gate, data-freshness indicators) and named the genuine high-severity gaps up front.

VERSUS THE OLD RS-360 SYSTEM

With RS-360 you lose the live picture the moment it scrolls off the screen. The old system *did* record batch, step, shift, and alarm data — but it locked that data inside a dead Paradox/BDE database (an obsolete 1990s file format) that needs a custom reader just to open, with no living historian, no surviving on-glass trends, no operator-usable step-duration timeline, no equipment-runtime stats, and no way for an operator to search it. Its one clear remaining edge today is the live mixer screen that animates roughly 20 valves and pumps. With Ring you gain durable, operator-searchable history and safety the old system kept locked away; the one named tradeoff is that Ring's mixer component popup currently shows equipment state as UNKNOWN until that animation is wired in.

STATUS & HONEST CAVEATS

Status: Live but partly limited — most screens are at full parity and on real screens; a few named areas are incomplete.

Watch-out: Concede the named gaps plainly if pressed, and do not soften them. The mixer component popup currently renders *every* device as a dash/UNKNOWN because the status-chip code behind it is dead. The use-tank screen's probe and additive states are hardcoded placeholder values, not live readings. The alarm Silence/Acknowledge buttons are inert under the read-only gate — that's by design at cutover. And the legacy live mixer animation is genuinely the one thing the old system still does better today. The claim is honest about all of this; owning it is the strongest move.

LIKELY QUESTIONS & HOW TO ANSWER THEM

Q What's in the missing 20%?

A Two kinds of things: deliberate scope cuts (features we chose not to port because they're obsolete on a single-site plant) and a short list of genuine named gaps. I'll show you the list — it's mostly the mixer component animation, hardcoded use-tank states, and the read-only-gated alarm buttons.

Q What does the old system still do better than Ring?

A The live mixer screen that animates about 20 valves and pumps. Ring's equivalent popup currently shows those devices as UNKNOWN until the animation is wired in. Everywhere else with a difference, Ring is ahead.

Q So is the mixer screen broken in Ring?

A The main mixer workflow works; it's the per-component *popup* that shows UNKNOWN because the status logic behind it isn't connected yet. It's a named, scoped gap, not a hidden one.

Q **How do I know "80%" is real and not a sales figure?**

A It's a file-and-line evidence map from an 11-reviewer audit — a source-code read, not a survey or an on-site walkthrough. Your engineer can open it and check any single claim against the actual code.

The audit-trail gap we disclose first: no per-person logins yet

WHAT IT IS

We're telling you this before you find it at an audit: Ring does not yet have real per-person logins. It uses a shared "Supervisor" password and a date-derived "Admin" password, there is no in-app way to create users or change passwords yet, and it records *what* was done but not *which named person* did it.

WHY IT MATTERS

For an industrial-adhesive plant, a shared-role login model is genuinely acceptable. But if you operate under stricter food or pharmaceutical record-keeping rules, "who did this?" accountability is a legal requirement — and you want that scoped *before* you sign, not discovered during an inspection. Naming it first is how you stay in control of that conversation.

HOW IT WORKS

Today Ring has two roles: Supervisor and Admin (the Admin password is partly derived from the date). The password-management screen is a "coming soon" stub. Ring does persist an operator's typed-in name, but on the honor system — there's no login-derived identity tied to each action, and no dedicated security-event log. Adding true named-user logins is a scoped piece of work; some of the underlying data plumbing already exists.

VERSUS THE OLD RS-360 SYSTEM

The old system was no better here — and be careful exactly *how* you say it. RS-360 also had no per-person forensics: it used access levels and **14 shared, unhashed plaintext credential slots** (named super1–super6 and oper1–oper8) found in the plant drop. Those credentials live in the **HMI's configuration files (an INI directory), not inside the PLC** — say "HMI-config-resident," not "PLC-resident," because an engineer who opens the file will see it's a screen-config file. Accountability in the old system was role-level, never person-level.

STATUS & HONEST CAVEATS

Status: In progress — a deliberately disclosed gap. Per-person named-user logins are not built yet; Ring carries forward only the Supervisor/Operator role model (never the old passwords).

Watch-out: Two wording nits a sharp engineer will catch, so fix them yourself first. (1) The legacy passwords are in the HMI's INI configuration files, **not** "PLC-resident" — there's no PLC copy in evidence. (2) Don't say "nine access levels (0-9)," because 0-9 is *ten* values; say "a 9-tier model" or "levels 0-9 (ten)." Also temper "the data plumbing is already built": the shift-start, handoff, and crash-logger records do have a name column, but the formula-history and alarm-lifecycle tables would each need a small database schema change before they could record *who*. The Ring-side substance — shared Supervisor password, date-derived Admin, no security audit log, partial name-capture plumbing — is fully accurate and verified.

LIKELY QUESTIONS & HOW TO ANSWER THEM

Q So anyone who knows the password can act as Supervisor?

A Yes, today — Ring uses shared role passwords, not per-person logins. For a chemical plant that's an accepted model; if you're under food or pharma rules, named-user logins are a scoped add-on we'd confirm with you before go-live. We're flagging this as gap number one on purpose.

Q Does Ring record who did what?

A It records *what* was done and keeps an event record, and it stores an operator's typed name on the honor system — but it does not yet tie each action to a verified named login. That's the gap.

Q Was the old system any better on accountability?

A No. RS-360 also had only role-level access, with 14 shared plaintext passwords sitting in its HMI configuration files. Neither system gives per-person forensics today; the difference is Ring already has part of the data plumbing to add it.

Q How big a job is it to add real named logins?

A Scoped, not huge — some plumbing already exists. A couple of history tables would need a small schema migration to record the user, and we'd build the password-management screen that's currently a stub. We'd size it against your actual compliance class.

Q Is this a dealbreaker?

A Only your regulatory class decides that, which is exactly why we want to settle that question with your team before you sign rather than after.

Floor troubleshooting without Studio 5000, plus one-button backup

WHAT IT IS

A set of built-in tools so your floor can diagnose problems and protect their data without specialist software. There's a one-tap system snapshot, a live reader for any PLC value, a live feed of PLC messages, a tool to set the

controller's clock, one-button database backup and restore, and an admin-only factory reset. ("Studio 5000" is Allen-Bradley's engineering software — normally you'd need an engineer with it to read a raw PLC value.)

WHY IT MATTERS

Today, deep troubleshooting on the old system means pulling in an engineer with Studio 5000 — slow and expensive. With these tools, floor staff can read a tag, see why communications hiccuped, and capture a support snapshot themselves, turning multi-hour callouts into minutes. The backup/restore protects your history.

HOW IT WORKS

Each tool is a built-in dialog. The Tag Inspector lets you paste a tag name (for example a tank's agitator output) and read its live value. The Diagnostic Console captures a one-shot host/PLC/traffic snapshot for support. The Communication Monitor shows a live feed of messages to and from the PLC. The Database Backup tool makes a safe online copy of Ring's history database (stored in **SQLite**, a small self-contained database file) and can restore it. The sensitive tools are password-gated, and the factory reset re-prompts before it runs.

VERSUS THE OLD RS-360 SYSTEM

With RS-360 you get a communications log and a Time/Date form, but no live tag inspector, no one-shot diagnostic snapshot, no built-in backup/restore, and no factory reset — so deep troubleshooting means calling in an engineer with Studio 5000. With Ring, that becomes a built-in, in-app task your own floor can do.

STATUS & HONEST CAVEATS

> Status: Live and on a real screen. All six tools exist, are reachable from buttons in the app, and are verified in code (including the exact dialog sizes and launcher locations).

LIKELY QUESTIONS & HOW TO ANSWER THEM

Q Can an operator read a live PLC value without Studio 5000?

A Yes — the Tag Inspector lets them paste a tag name and read its live value right in Ring. That's a built-in capability the old system never had.

Q Could the Tag Inspector accidentally change something in the running plant?

A No. It reads values; it does not write them, and in any case the read-only gate suppresses every write command. Reading sends a request packet but issues zero commands that could change the process.

Q Is the factory reset a danger sitting on the screen?

A It's admin-tier only and re-prompts for the password before it does anything, so it can't be triggered casually. The same goes for the backup/restore, which asks for a password when it opens.

Q Did the old system really not have backup?

A It had a comms log and a Time/Date form, but no built-in database backup, no live tag inspector, and no diagnostic snapshot. Those are net-new in Ring.

A 13-language screen you can switch live — with honest coverage limits

WHAT IT IS

Ring ships a 13-language interface with English as the default, and you can switch language on the fly — no restart, no reinstall. The translations were seeded from your plant's own Dutch dictionary. **But be precise about coverage**: today the translation reliably covers the first-run setup wizard, the top-level menu buttons, the common buttons, and core domain words — not yet the deep operator screens.

WHY IT MATTERS

Your floor reads the plant in Dutch; the old system runs in Dutch. An English-only screen would be a real adoption problem. The live-switch engine is real and the high-traffic navigation is translated — but a half-translated screen can read *worse* than none, so we're honest about exactly how far it currently reaches.

HOW IT WORKS

A localization service swaps a single per-language set of strings, so every translated label repaints instantly with no restart. Crucially, it deliberately never changes the system's number formatting or culture settings, so batch math and the numbers Ring exchanges with the PLC are completely unaffected by the language choice. Missing translations fall back to English, so the screen never goes blank.

VERSUS THE OLD RS-360 SYSTEM

With RS-360 you pick one language at commissioning and it's frozen there — roughly 7 language dictionaries (13 source files including regional variants), but one language per install, with no runtime switching. With Ring you match that multi-language reach with a modern engine that switches live and extends incrementally, screen by screen.

STATUS & HONEST CAVEATS

> Status: Live but partly limited — the engine and high-traffic navigation are translated; the deep operator screens are still English.

> Watch-out: Do not claim it "covers menus and high-traffic operator screens" — that overstates today's breadth. What's actually translated today is the first-run wizard, the roughly 11-12 top-level menu buttons, the common buttons, core domain nouns, and two dashboard *header* strings (about 162 Dutch keys in total). The submenus (~91 items) and the main operator screens — mixer, storage, use-tank, formula editor — and even the dashboard's KPI cards still render in English. If an engineer switches to Dutch and opens any main screen, they'll see mostly English. Own the audit's own phrase: "half-translated can read worse than none," which is exactly why we disclose the boundary.

LIKELY QUESTIONS & HOW TO ANSWER THEM

Q If I switch to Dutch, is the whole screen in Dutch?

A Honestly, no — not yet. The setup wizard, the top-level menus, and the common buttons switch to Dutch; the deeper operator screens (mixer, storage, use-tank, formula editor) and the dashboard's data cards are still English. We're keying those in screen by screen, and we'd rather tell you that than pretend it's complete.

Q Does changing language affect the batch numbers or what gets sent to the PLC?

A No, by design. The language switch only swaps display text; it deliberately never touches number formatting or the values exchanged with the controller, so batch math is unaffected.

Q Why ship it half-translated at all?

A Because the engine is real and the parts operators touch most — the wizard and the top-level navigation — are done and switch live. The rest is a known, scoped follow-up, not a surprise. We'd rather have the framework live and extend it than wait.

Q The old system was multilingual — what's actually better here?

A The old system locks one language per install at commissioning. Ring switches live at runtime across 13 languages and extends incrementally. The honest tradeoff is that Ring's coverage is still filling in on the deep screens.

The built-but-gated follow-ups that don't block daily work

WHAT IT IS

A short list of items that exist but need a small finishing touch: an automatic batch-completion email that runs once it's enabled in settings, and two Setup areas that still read "coming soon" — the in-app communication-adapter settings and user/password management. None of these blocks day-to-day operation.

WHY IT MATTERS

You should know the exact edges of "done." These three items are the difference between "fully wired" and "wired but gated." The good news is none of them stops anyone from running glue; the honest news is they each need a step before they're fully self-service.

HOW IT WORKS

The batch-completion email is real, working code with a real mail transport built in. It's gated off by default and only attaches when enabled. The communication-adapter settings (timeouts and retry behavior) live in a config file today rather than on a screen, and the password-management screen is a "coming soon" stub.

VERSUS THE OLD RS-360 SYSTEM

With RS-360 there is no batch-completion email at all, while comm and password settings are exposed as editable forms. With Ring, even the config-gated email is a net gain; we deliberately hardcode the single-site tag

layout, trading editable forms for simplicity, with the toggle and Setup screens as honest follow-ups.

STATUS & HONEST CAVEATS

Status: Built but partly gated — the email feature is real code switched on via config; the comm-adapter and password Setup screens are genuine "coming soon" stubs not yet wired.

Watch-out: Don't say the email "just runs once you flip a setting." It needs **two** things at commissioning: the enable flag turned on *and* the mail server (SMTP) host, sender, and recipients configured. Out of the box those mail-server fields are empty, so if you flip *only* the flag, Ring will quietly *log* a "batch complete" entry instead of actually sending mail. Set both, and it sends. The comm-adapter and password Setup screens are correctly disclosed as not-yet-built but not blocking daily work.

LIKELY QUESTIONS & HOW TO ANSWER THEM

Q What exactly is "coming soon"?

A Two Setup screens: the in-app communication-adapter settings and user/password management. The settings behind the comm screen live in a config file today, and the password screen is a stub. Neither blocks running the plant.

Q Does the batch-completion email actually work?

A The code and mail transport are real. To send mail it needs both the enable flag on *and* your mail server details filled in at commissioning. If only the flag is on, it logs the event instead of emailing — so we configure both, or we tell you it'll log rather than send.

Q Do any of these gated items stop my operators from doing their jobs?

A No. The email is a bonus scorecard, and the comm settings are rarely changed and live in config. The missing Setup screens matter mainly at re-commissioning, not day to day.

Q Why isn't there just a checkbox for the email?

A The UI toggle is a minor (P2) follow-up. Today it's a one-time config edit at commissioning rather than an in-app switch. It's a convenience gap, not a capability gap.

The scoping questions only your team can answer

WHAT IT IS

A short list of facts we *can't* settle for you and need to confirm together before cutover: your plant's regulatory class, the full tank scope, and the provenance of a second controller identity we saw in an export. These are flagged as honest questions for you, not claims we invented.

WHY IT MATTERS

Commissioning-day surprises are expensive. Settling these three now means no mismatched tank counts, no wrong-language screens, and no controller confusion when the system goes live. It's cheaper to ask than to discover.

HOW IT WORKS

Three open questions. First, your **regulatory class** (chemical-only vs. food vs. pharmaceutical) decides how much audit-trail work the named-user logins really need. Second, **tank scope**: the live plant runs 13 tanks (1 mixer + 6 storage + 6 doser) with Dutch names, not the 1/2/4 demo subset Ring ships with — so a name-import and a scope confirmation are needed before the floor sees correct names. Third, **controller provenance**: the live box is a CompactLogix 1769-L36ERM on firmware v20.12, but a separate "running export" we were given carries a controller identity that does *not* match it — and the newer Studio software can't even upload from a v20 controller — so whether a second controller exists is genuinely an open question for your point of contact (POC).

VERSUS THE OLD RS-360 SYSTEM

With RS-360 you inherit a system pre-bound to one site's tank set and one Dutch install, with no documented provenance trail for the controller it talks to. With Ring, these are explicit, documented open questions to resolve with your team before cutover — which is itself the safer position.

STATUS & HONEST CAVEATS

Status: Roadmap — these are open scoping questions for your team, not built features.

Watch-out: Present these strictly *as unresolved questions*, and don't imply they're already answered. The controller provenance (one box or two?) is genuinely open and only your POC can settle it. And the 13-tank reality means a name-import plus a scope confirmation is required before operators see the correct tank names — that work is real and named.

LIKELY QUESTIONS & HOW TO ANSWER THEM

Q Is there a second controller in my plant?

A Honestly, that's an open question for your team. The live controller is a CompactLogix 1769-L36ERM on v20.12, but an export we were given carries a different controller identity that doesn't match it. We're flagging it as a question to settle with your POC, not asserting an answer.

Q Why does Ring only show 3 tanks when I have 13?

A Ring ships with a small demo subset (tanks 1/2/4). Your live plant runs 13 — one mixer, six storage, six doser — with Dutch names. Bringing those in is a name-import plus a scope confirmation, which we'd do before the floor uses it.

Q **Why are you asking *me* these — shouldn't you already know?**

A These three depend on facts only you hold: your compliance class, your exact tank list, and the history of your controllers. We could guess, but the honest and safe move is to confirm them with your team rather than discover a mismatch at cutover.

Adopt the new screen now; the control-logic port is scheduled separately

WHAT IT IS

You can put up Ring — the new operator screen — on its own timeline, independent of any control-logic work. The HMI is finished and waiting. The separate question of which control program runs the plant, and any work to bring Ring's analyzed control logic fully live, is a scoped engineering engagement scheduled on purpose, not a surprise on cutover day.

WHY IT MATTERS

This is the single most important framing point in this chapter, and the easiest to get wrong in a hostile room. Ring replaces the *screen*, not the *controller*. Keeping those two cleanly separable means your floor can adopt the modern interface immediately while any controller-logic engagement is scheduled deliberately — and it means installing Ring does not touch the program that actually runs your line.

HOW IT WORKS

Two distinct things must not be confused. (1) Our pre-cutover snapshot of the controller's project file ships *inert* — its main task is inhibited and the batch routines are commented out. (2) On-site reads of the **live** plant PLC show the opposite: the main task is scanning and real batches are running. So that inert state describes *our checked-in snapshot*, **not** the box on your floor. Bringing Ring's analyzed control logic fully live is a scoped engagement using v20-era Studio 5000 (our bench controller is a firmware twin at v20.12), with a couple of routines needing real porting. The HMI is complete and waiting; the plant goes fully live once that engineer engagement completes.

VERSUS THE OLD RS-360 SYSTEM

With RS-360 the control logic is entangled with bespoke Borland bridges and sidecar executables, with no clean separation between the screen and the control logic. With Ring, the HMI is cleanly separated from the controller program, so the interface can go up now and the control-logic work can be restored on a scoped, scheduled basis.

STATUS & HONEST CAVEATS

Status: Roadmap — a scheduled engineering engagement, kept separate from the HMI so neither blocks the other.

Watch-out: This carries the highest framing risk in the section, so say it out loud. The live plant's control logic **already runs** — the main task is scanning and real batches are executing on the same Allen-Bradley **CompactLogix** PLC (call it CompactLogix, not ControlLogix — that's the exact hardware, and an engineer will flag the mislabel). Ring is an HMI replacement on that *same* PLC: **you are not claiming Ring authored or currently controls the running logic.** The "inert snapshot" applies only to our checked-in copy of the project file, not to the running plant. And do not assert "the legacy logic still runs the plant" as settled fact — *which* program is live is part of the open provenance question for your POC.

LIKELY QUESTIONS & HOW TO ANSWER THEM

Q Does installing Ring change how my plant actually runs?

A No. Ring is the operator screen; it talks to the same controller the old screen talked to. Under the read-only gate it only *reads* — zero write commands. The control logic on the PLC is untouched by installing Ring.

Q You said your control logic is "inert" — so is it running or not?

A Two different things. The *live plant's* controller is scanning and running real batches right now. The "inert" state describes only our checked-in *snapshot* of the project file, where the task is inhibited and routines are commented. Which exact program is the authoritative live one is still an open question we want to confirm with your POC.

Q Can I put Ring up without doing any of the control-logic work?

A Yes — that's the whole point of keeping them separable. The HMI is finished and goes live on its own timeline; the control-logic engagement is scheduled independently and doesn't gate the screen swap.

Q What is the control-logic engagement, concretely?

A A scoped piece of work in v20-era Studio 5000 — our bench is a firmware twin of your controller at v20.12 — to bring the analyzed logic fully live, with a couple of routines that need real porting. We schedule it deliberately rather than improvising it on cutover day.

Q Is it CompactLogix or ControlLogix?

A CompactLogix — specifically a 1769-L36ERM. They're the same Allen-Bradley protocol family, but the hardware is CompactLogix, and I'll always name it precisely.

Appendix A — Plain-Language Glossary

Every term in this guide, defined as if for the first time. If a question turns on a word, find it here.

The plant and the process

- **Corrugating adhesive / starch glue** — the industrial glue this plant makes; it bonds the layers of corrugated cardboard. Not food.
- **Viscosity** — how thick or thin the glue is. It's the critical quality spec, but the controller has **no viscosity sensor**, so neither the old system nor Ring can display a live viscosity number.
- **Batch** — one production run of one recipe; the plant's basic unit of work. ~1,404 in recorded history.
- **Recipe / formula** — the ordered steps that make one product (add water, heat, add starch, stir…).
- **Step / operation** — one instruction in a recipe, identified inside the controller by a numeric **op-code**.
- **Op-code** — the controller's number for an operation (e.g., "add starch"). The old system showed operators these raw numbers; Ring turns them into words.
- **Tank** — a vessel holding liquid. 13 in service: 1 mix tank, 6 storage, 6 doser/feed.
- **Mix tank / Make-Ready tank** — where the glue is actually mixed.
- **Doser / feed tank** — meters an ingredient into the mix.
- **Agitator** — the motorized paddle that stirs a tank; turning it on/off is a recipe step the controller sequences.
- **Giveaway / overdose** — product you over-make beyond target: ingredients you paid for and effectively gave away. Ring can price it.
- **CIP (Clean-In-Place)** — cleaning a tank without dismantling it; "CIP state" = whether a tank is clean, dirty, or in use.
- **OEE (Overall Equipment Effectiveness)** — a standard factory score combining availability, performance, and quality; Ring's downtime screen estimates it.

The two computers

- **PLC (Programmable Logic Controller)** — the rugged industrial computer that physically runs the plant (valves, pumps, mixer, safety interlocks). Here: an **Allen-Bradley CompactLogix 1769-L36ERM**. Ring does **not** replace or reprogram it.
- **HMI (Human-Machine Interface)** — the operator's screen software. **RS-360** = the old HMI (2002, Borland C++, DOS-era). **Ring** = the new HMI.
- **Tag** — one named value inside the controller (e.g., a tank's temperature). "133 tags read" = every live value Ring needs was read correctly.
- **Studio 5000** — Allen-Bradley's engineering software for programming the PLC. A selling point is that Ring gives the floor diagnostics **without** needing Studio 5000 open.

How Ring connects

- **EtherNet/IP** — a standard, open industrial network protocol Ring uses to talk to the controller. (Different from a generic office network; it's the Allen-Bradley standard.)
- **CIP frame / request packet** — the small network message sent to *read* a tag. (This is why "we sent zero bytes" is wrong but "we sent zero **commands** that could change the process" is right.)
- **libplctag** — the open-source library Ring uses to speak EtherNet/IP. Replaces the old system's bespoke Borland helper components.
- **Read-only mode / cutover gate (`ReadOnlyMode=true`)** — a Ring setting that **suppresses every write** to the controller, so Ring can watch the live plant safely before go-live.
- **Heartbeat** — a value in the controller that keeps changing while the program is running; Ring watches it to tell "running" from "frozen" from "unplugged."
- **Stale / freshness** — whether an on-screen reading is currently live or has stopped updating. Ring dims stale readings and stamps "as of HH:MM" so a frozen number is never mistaken for a live one.

The software underneath

- **.NET / WPF** — the modern Microsoft technology Ring is built on (current, supportable, hireable). Contrast with the legacy's dead 2002 Borland C++ toolset.
- **SQLite** — the modern, single-file, open database Ring stores its data in. Any standard tool can open it; one-click export to CSV/Excel.
- **Paradox / BDE (Borland Database Engine)** — the **abandoned** 1990s database format the legacy used; no maintained driver, which is why recovering legacy data required writing a custom reader.
- **WAL (Write-Ahead Logging)** — a SQLite durability feature: a batch you just finished survives a power blip, and the screen doesn't freeze under load.
- **Migration** — a controlled, all-or-nothing upgrade of the database structure when Ring is updated; an interrupted upgrade won't corrupt your history.
- **Process Historian** — a background recorder that logs each tank's temperature/level/recipe about once a minute, building the history that powers trends, best-batch comparison, and the cost engine — with no extra load on the controller.
- **Golden batch** — your best, proven good run; Ring can overlay a current batch's temperature trace against it to catch drift early.
- **Dashboard / command center** — Ring's one home screen that shows the whole plant at a glance.
- **KPI tile** — one of the key headline numbers on the dashboard (open alarms, mix-tank temperature, batch % done, which tank needs attention); each is tappable to jump to the screen that fixes it.

Status words used in this guide

- **Live (wired)** — built, finished, and reachable by an operator on a real screen today.
- **Dormant** — the code is built and tested but **not yet placed on a screen** an operator can reach. Honest status; some "shipped" items are here.
- **In progress** — partially built; core works, finishing touches remain.

- **Roadmap** — planned and specified, not yet built.
- **Parity** — Ring matches what the old system did for a given capability. "~80% parity-or-better" = Ring matches or beats the legacy on about 80% of capabilities (per a code audit), with the gaps named.

People who might grill you

- **Controls engineer** — knows the PLC and the legacy intimately; will challenge anything that misstates the controller or the old system. Most of Part 0 §0.8 exists for this person.
- **Operator** — runs the plant daily; cares about "is it easier, and is it honest about what's live?"
- **Owner / buyer** — cares about cost, risk, lock-in, and what isn't finished.

Appendix B — The Hardest Questions (cross-cutting), and How to Answer Them

The per-feature Q&A lives in each chapter. This appendix is for the **big, whole-product questions** that aren't about one feature — the ones most likely to decide the meeting. Each answer is written to be said out loud, and each is honest, because honesty is the thing that survives a grilling.

The five "say it this way" rephrasings (your armor — repeated from Part 0 because it's that important)

If you're tempted to say…	A sharp person can refute it because…	Say this instead
"The old system records nothing."	Its database files hold ~1,404 batches and ~16,776 step rows.	"The old records are **locked in a dead format** with **no living historian, trends, cost, or step-timing** — Ring keeps the same history **usable and exportable.**"
"We read the plant without sending a single byte."	Reading values sends network request packets.	"We read all 133 live values **without issuing a single command that could change the process** — every write was suppressed."
"Zero bytes to the controller."	Same as above.	"**Zero write commands** under the read-only safety gate."
"The old system couldn't tell if it was connected."	It had an OK/Not-OK heartbeat light.	"The old light was one on/off signal; Ring **names unplugged vs. frozen vs. running** and **gates writes** off that state."
"~70–80% parity, verified on-site."	The number and the on-site event are two different things.	"**~80%** parity-or-better per an engineering **code audit**; separately, **133/133 live reads** verified **on-site** on your controller."

The whole-product questions

Q **Why rebuild the whole thing instead of just patching RS-360?**

A Because the four things you most want — cost visibility, a real searchable history, trends, and readable alarms — can't be patched into it; it never had the foundations for them, and it runs on 2002 tools (Borland C++, the Paradox/BDE database) that no one can support or hire for anymore. Patching buys nothing and the platform risk grows every year. Rebuilding the screen is the only way to get those capabilities and get off the dead toolchain — while keeping the proven controller.

Q Does any of this touch the machine that actually runs the plant?

A No. The Allen-Bradley controller, the recipes it runs, the valves, the pumps, and the safety interlocks are all untouched. Ring is the operator's screen on top of it — we kept the engine and replaced the dashboard. And it shipped with a read-only gate that blocks every command, so during the trial it could only watch.

Q How do I know these features actually work, and aren't just slides?

A Two independent proofs. First, the software ran **live on your real controller** and correctly read all 133 of the values it needs — not a simulator, your hardware. Second, every feature claim in our materials was checked **against the source code, line by line, by an independent audit** that re-verified anything questionable; all 87 capabilities exist in the code, and where a claim was too strong we corrected it ourselves before this meeting. That's why I can tell you plainly which parts are fully live and which are still being wired in.

Q Is our data locked into your software the way it's locked into the old one?

A The opposite. Ring stores everything in SQLite — a single open database file any standard tool can read — with one-click export to CSV that opens directly in Excel. You can hand an auditor your whole history on day one, no vendor required. The lock-in problem is the old system's, not Ring's.

Q Will my operators have to relearn everything?

A It's designed to *reduce* training, not add it: one green/red dashboard instead of ~97 screens, alarms in plain words with a "first thing to check" card, a guided setup wizard, built-in help, and their own language. The per-tank detail screens they already know still exist underneath. Most operators are faster on day one, not slower.

Q What isn't finished? Tell me straight.

A A short, named list, none of it blocking daily use: (1) the screen-by-screen **translation** is partial — the wizard, menus, and key terms are translated, but the deep operator screens still render English; (2) **per-operator named logins** aren't in yet — it uses shared role passwords today, and we disclose that as an audit-trail gap; (3) a few **analytics screens and value cards** are built but not yet wired onto every screen; (4) one **severity-themed alarm pop-up** is built but not switched on (the live alarm pop-up is a single full-screen alert); (5) **email** is wired but needs your mail server configured to actually send; (6) the **agitator auto-cycle** screens exist but we won't claim they drive the real paddles until we confirm it live. We label every one of these in the product so you always know what's real today.

Q The "87 features" number — is that inflated?

A It's the count of distinct capabilities across five themes, and every one was verified to exist in the code. It's fair as a headline. If you'd rather be conservative, lead with the handful of flagships — one command-center screen, a permanent searchable batch history, the owner cost view, plain-language alarms, 13 languages — and let the 87 be the depth behind them.

Q Who can maintain this after you? Are we trading one single-point-of-failure for another?

A Ring is built on current Microsoft .NET/WPF and an open database — the most common business-software stack there is. Any .NET contractor can build, change, and support it, and every release is checksum-verifiable. The old system is the single point of failure: one aging toolchain, one obscure language, a dead database engine. Ring moves you off that.

Q Could switching over go wrong on cutover day?

A The old system stays installed and parked, with a documented rollback target of about ten minutes if anything looks wrong. And because Ring runs read-only first, you prove it next to the live plant for as long as you want before you ever switch writes on. The switch is deliberate and reversible, not a leap.

Q What's the one thing you'd challenge me on if you were the buyer — and what's the answer?

A Probably the alarm pop-up: the marketing implies a severity-aware pop-up (calm warnings, blinking criticals, an E-STOP button) and that specific behavior is built but **not yet switched on** — today every in-range alarm raises the same full-screen alert. The honest answer is to say exactly that, then pivot to what *is* live and strong: alarms in the controller's own words, a one-tap jump to the fixing screen, a first-response card, severity shown by shape-and-color in the list, and downtime tied back to the batch it hurt. Conceding the one soft spot makes everything else more credible.

Takeaway: you don't win this meeting by claiming perfection — you win it by being the person in the room who already knows, and will say, exactly what's done, what's partial, and why it's still a generational leap over a blind 2002 screen on a dead toolchain. This document exists so that person is you.