How Does a LIMS Work, and What Makes an Implementation Fail?

A LIMS is not hard to buy and it is hard to implement. The difference is a short list of decisions made badly, early and quietly, long before anyone signs a contract.

Brandon Holland, Marketing & Operations Digital Solutions Architect

July 2, 2026 · 12 min read

IN BRIEF

How does a LIMS work? It makes the sample the unit of record. Every aliquot, transfer, instrument reading, method version and approval attaches to that sample as a time-stamped entry, so any result can be reconstructed years later. How well it works is decided by four choices made before purchase, not by the software.

Key takeaways

- A LIMS makes the sample, not the report, the unit of record. Every other capability follows from that one choice.
- A sample passes through six stages, and the LIMS records who moved it, when, and against which method version.
- Integration effort scales with instruments that write proprietary files, not with how many instruments you own.
- FDA expects audit trails reviewed as often as the data itself, which is a staffing decision rather than a setting.
- Four decisions set the outcome: data model owner, instrument scope, what the lab stops doing, and validation sign-off.

You have been asked to look at a LIMS. Maybe an auditor wrote something up, maybe a technician spent a Friday reconciling two spreadsheets that disagreed, maybe the lab simply grew past what the current system holds. Either way the question in front of you is not what the software does in a demo. It is what it will take

to run your lab on it.

Most LIMS projects that go badly do not go badly on software. They go badly on four decisions made before anyone saw a demo: who owns the data model, which instruments are genuinely in scope, what the lab agrees to stop doing, and who signs off validation. Each is a laboratory decision, cheap to change in month one and expensive in month seven.

This covers how a LIMS works, how a sample moves through one, what the features do mechanically, why integration is where schedules slip, and then those four decisions in the order you have to make them.

What does a LIMS actually do?

A LIMS makes the sample the unit of record. That single sentence is the whole design. In a spreadsheet or a paper logbook, the unit of record is the report, and the sample is a row inside it. In a LIMS it is the other way around: the sample exists first, and every aliquot, transfer, method, instrument reading, calculation and approval attaches to it as a time-stamped entry.

If you are earlier than this and still comparing categories of system, the LIMS resource hub is the better starting point.

The practical consequence is reconstruction. Two years after a result leaves your lab, someone can ask who received the sample, what condition it arrived in, which instrument ran it, which method version applied, who reviewed the number and what it was before it was corrected. A system built around samples answers that in one place. A system built around reports answers it by asking three people to remember.

- **Chain of custody.** Every handoff recorded with who, when and in what condition, rather than reconstructed from initials on a form.
- **Method and version binding.** The result carries the method revision that produced it, which matters the first time a method changes and an old result is questioned.

- **Instrument capture.** Readings arrive from the instrument rather than being retyped, which removes the transcription error nobody can find later.
- **Review as a recorded act.** Approval is an event with a person and a timestamp attached, not a signature at the bottom of a printout.

How does a sample move through a LIMS?

The same six stages, in every lab, whatever the discipline. What differs between labs is the vocabulary, the number of tests hanging off each sample, and how many of the stages are still done on paper. A LIMS does not invent this sequence. It records it, and refuses to let a sample skip a step.

STAGE	WHAT THE LIMS RECORDS	WHAT GOES WRONG WITHOUT IT
Login	A unique identifier, the source, the condition on arrival, the holding-time clock	Two samples share a number, or one arrives and nobody knows when
Test assignment	Which methods apply, at which revision, and who is qualified to run them	A test is run to last year's method and nobody can tell
Preparation and aliquoting	Every child sample linked back to its parent, with the person and the time	A result exists and its parent sample cannot be found
Analysis	The instrument reading, captured directly, against the batch it ran in	A number is typed from a printout and transposed
Review	Who approved, when, and what the value was before any correction	A correction replaces the original and the history is gone
Reporting and release	Which report went to whom, containing which approved results	A client holds a version the lab cannot reproduce

Two things in that table are worth reading twice. The holding-time clock starts at login, not at analysis, so a LIMS that only records analysis has missed the point at which most environmental and food results are

actually invalidated. And the review stage keeps the prior value rather than overwriting it. That is not a design preference; the regulation quoted below requires it in those words.

Which LIMS features actually matter?

Feature lists are long because they are written to be long. The honest version is that six capabilities do the work, and the rest are conveniences built on top of them. Judge a system by how well it does these, mechanically, and not by how many boxes its comparison grid ticks.

- **Sample tracking** is the identifier that survives aliquoting, splitting, relabelling and disposal. If a child sample can lose its parent, nothing downstream can be trusted.
- **Workflow enforcement** is the system refusing to let a sample reach review without the tests assigned to it having results. It is a gate, not a to-do list.
- **Instrument capture** is the reading arriving without a person in the loop. Manual transcription is where the errors that cannot later be found are introduced, and it is the only capability that removes them rather than checking for them.
- **The audit trail** is a change history that shows the previous value. A system that logs that a change happened without showing what it was from does not satisfy the regulation it is sold against.
- **Reporting** is the certificate of analysis assembled from approved results only, with the approved version of each. Dashboards are useful; the report that leaves the building is the thing that carries liability.
- **Inventory and reagent tracking** is lot numbers and expiry dates tied to the results that used them, so a bad lot can be traced forward to every affected result rather than guessed at.

Everything else, from client portals to billing integration to stability studies, is worth having and is not what makes the system a LIMS. A lab that gets the six right on a modest system is in better shape than a lab that bought the long list and configured half of it.

What has to be decided before you shortlist?

Four things, and the order matters because each one constrains the next. If the lab is coming off spreadsheets rather than replacing an older system, the migration path has a different shape and is worth reading alongside this.

Who owns the data model. Someone has to decide what a sample is in your lab, how it relates to a batch, a project and a client, and which of those relationships are allowed to be many-to-many. This is not an IT decision and it is not the vendor's decision. It needs someone in the lab with the authority to settle a disagreement between two sections that each have a reasonable definition.

Which instruments are in scope. Not which instruments exist. Which ones will send data to the LIMS in phase one. This list is almost always too long on the first pass, and the next section explains why that matters more than it appears to.

What the lab stops doing. A LIMS that runs alongside the old spreadsheet has not replaced it, and now there are two systems of record that disagree. Naming the things that stop, and who confirms they have stopped, is the difference between a migration and an addition.

Who signs off validation. Decided at the start, because it determines how much documentation the project produces, and documentation written retrospectively costs several times what it costs written as you go.

> NOTE
>
> The requirements document is an output of these four decisions, not a substitute for them. A lab that cannot describe its current sample flow on a single page is not ready to write requirements, and will end up describing the vendor's workflow back to itself. Write the one page first; the functional, technical and compliance requirements fall out of it in an afternoon.

Why does LIMS integration take longer than expected?

Because the effort is not proportional to the number of instruments. It is proportional to the number of instruments producing a format nobody else reads. An instrument with a documented export, a stable file layout and a vendor who answers email is a few days of work. An older instrument writing a proprietary binary that changes between firmware versions can absorb weeks, and the work is not reusable for the next instrument.

Counting instruments gives you a schedule that is wrong in a specific direction: too optimistic, and wrong by more the older your lab is. Before committing to a date, sort every in-scope instrument into three buckets.

1. **Documented export.** A stable layout and a vendor who answers email. Days of work.
2. **Undocumented but text.** Parseable, fragile, and it breaks on firmware updates. A week or two each, then again after the next update.
3. **Proprietary binary.** Weeks, and the work is not reusable for the next instrument.

Then plan on the third bucket consuming most of the integration time regardless of how few instruments are in it. Instrument integration covers the interface patterns in more depth.

Instruments are half of the integration question. The other half is systems: the ERP that raises the invoice, the ELN where the method was developed, the client portal that shows a result before the PDF is sent. These connect through APIs rather than file parsers, and the constraint is different. The instrument problem is format; the systems problem is ownership. Every integration needs a named person on each side who agrees what a sample identifier means when it crosses the boundary, because a LIMS and an ERP will both have a field called *client* and mean different things by it. Application integration covers connecting a LIMS to systems that are not instruments.

WHERE THE MONTHS ACTUALLY GO ON A MID-SIZE IMPLEMENTATION

Weeks 1 to 4

Data model and requirements. Cheap to change here and nowhere else.

Weeks 4 to 10

Configuration. Usually the part that runs to schedule.

Weeks 8 to 20

Instrument and system interfaces. Overlaps configuration and is the usual source of slip.

Weeks 16 to 28

Validation and user acceptance. Compresses only by deferring work, not removing it.

Weeks 24 to 36

Parallel running, then retirement of what the LIMS replaced.

What does validation actually require?

If your lab holds electronic records under a regulation, the obligation is written down and it is short. 21 CFR 11.10(a) requires “validation of systems to ensure accuracy, reliability, consistent intended performance, and the ability to discern invalid or altered records.” That last clause is the one labs underestimate: the system has to be able to show that a record was altered, not merely prevent it.

The companion requirement is the audit trail, and 11.10(e) is specific in a way worth reading closely. It requires secure, computer-generated, time-stamped audit trails recording operator entries and actions, and then adds a sentence that decides a great deal: “Record changes shall not obscure previously recorded information.” A system where correcting a value replaces it does not satisfy that, however careful the correction.

The part that becomes a staffing decision rather than a software one is review frequency. FDA’s Data Integrity and Compliance With Drug CGMP guidance, issued December 2018, answers it directly: where a regulation specifies a review frequency for the data, the same frequency applies to the audit trail; where it does not, the lab determines the frequency using its own risk assessment. Either way someone is reading

audit trails on a schedule, and that person has to exist in the plan.

Labs outside FDA's reach are not outside this. An ISO/IEC 17025 accreditation assessor asks the same question in different words: can you demonstrate that the system does what you say it does, for the scope you are accredited for? The evidence is the same evidence. Our validation services page covers how the work is structured, and compliance services covers the documentation set it produces.

Where do implementations actually go wrong?

Not in the places the project plan worries about.

The most common failure is scope that grows quietly. Phase one starts as four instruments and eight report templates, and by month three it is eleven instruments and thirty templates, because each addition was individually reasonable. Nobody decided to double the project. It happened by accumulation, and the schedule was never revised to match.

The second is parallel running with no end date. Parallel operation is sensible for a defined period and corrosive once it becomes the normal state, because staff will use whichever system is easier and the LIMS becomes a place where data is copied to rather than created. Set the retirement date at the start and treat it as a deliverable.

The third is training treated as an event rather than a role. A two-day session before go-live teaches the screens, not what to do when a sample arrives in a condition the workflow did not anticipate, which is the moment a technician reaches for the old spreadsheet. Somebody in the lab owns the system after the vendor leaves, and should be named before configuration starts.

The fourth is honest and worth saying plainly: some labs should not implement a LIMS yet. A lab with three people, one instrument and stable methods may be better served by fixing its sample numbering and its backup discipline first. The signal that a LIMS is genuinely warranted is usually not volume. It is that more than one person needs to answer questions about the same sample, and they currently answer them differently.

- The current sample flow fits on one page, and two people agree it is accurate
- A named person in the lab owns the data model and can settle a dispute
- In-scope instruments are sorted by export format, not counted
- Every system the LIMS will exchange data with has a named owner on the other side
- The list of things the lab stops doing exists, with a date and an owner
- Audit trail review frequency is decided, with a person named against it
- Validation sign-off is agreed before configuration starts, not after
- Someone in the lab owns the system after go-live, by name

What a LIMS carries in this work

The mechanism, rather than the benefit. A LIMS enforces the sample as the unit of record, which is what makes reconstruction possible. It records custody transfers as events rather than as signatures. It binds a result to the method version that produced it. It captures instrument output without a person retyping it. And it keeps a change history that shows the prior value rather than replacing it, which is the specific thing 11.10(e) asks for.

None of that removes the four decisions. The software cannot decide what a sample is in your lab, and a system configured against a data model nobody owns will reproduce the confusion it was bought to fix, faster and with an audit trail.

> **How LabLynx handles this**
>
> The LabLynx LIMS is configured against your data model rather than shipping one, which is why the first phase of our implementations is a data model workshop and not a software install. Instrument and system interfaces are scoped by export format and by named owner, for the reasons above, before a date is committed to. That suits labs willing to make the four decisions. It suits a lab looking for something to switch on next week considerably less.

Frequently asked questions

How long does a LIMS implementation take?

For a mid-size lab with a handful of instrument interfaces, four to nine months from kickoff to routine use is typical. Configuration is rarely the long part. Instrument and system interfaces, validation, and the period of parallel running before the old system is retired are where the months go, and each is set by decisions made in the first month.

What does LIMS integration actually involve?

Two different problems. Instrument integration is about file formats: a documented export takes days, a proprietary binary takes weeks. System integration, with an ERP, an ELN or a client portal, is about ownership: both sides have to agree what a sample identifier means when it crosses the boundary. Schedules slip when the second is planned like the first.

Do we need to validate a LIMS if we are not FDA regulated?

Formal computer system validation is required where a regulation demands it, and 21 CFR 11.10(a) is explicit for closed systems holding electronic records. Outside that, an ISO/IEC 17025 assessor asks the equivalent question: can you demonstrate the system does what you say, for your accredited scope? The evidence you keep is the same either way.

What is the difference between LIMS configuration and customisation?

Configuration uses settings the vendor supports and tests: fields, workflows, roles, report layouts. Customisation writes code the vendor does not maintain. Configuration survives upgrades; customisation has to be re-validated after each one. A lab that needs a great deal of customisation has usually chosen a system whose data model does not match how it works.

Can a LIMS replace our instrument software?

No, and treating it as a replacement is a common scoping error. Instrument control software runs the instrument. A LIMS receives what the instrument produces, attaches it to the right sample and batch, and carries it through review and reporting. The two connect; neither does the other's job, and an integration plan that assumes otherwise fails at the first firmware update.

What does the acronym LIMS actually stand for?

Laboratory information management system. The middle word is the useful one: the system manages information about samples, tests and results, not the samples themselves and not the instruments. That is why the sample is its unit of record, and why a lab's data model, rather than its instrument list, is the thing that decides whether an implementation works.

Sources and references

1. 21 CFR 11.10 Controls for closed systems

   Electronic Code of Federal Regulations

2. Data Integrity and Compliance With Drug CGMP: Questions and Answers, Guidance for Industry

   U.S. Food and Drug Administration

3. ISO/IEC 17025:2017 General requirements for the competence of testing and calibration laboratories

   International Organization for Standardization

Brandon Holland

Marketing & Operations Digital Solutions Architect

Brandon Holland is LabLynx's Marketing & Operations Digital Solutions Architect. He builds and runs the digital systems behind the company's public surface, from the website and content architecture to the tooling that keeps product information accurate everywhere it appears. He writes about laboratory informatics from the systems side: how lab records get structured, searched, and kept defensible as a lab scales.

Read the current version online

https://www.lablynx.com/resources/articles/how-does-a-lims-work/

Downloaded September 23, 2026. LabLynx builds laboratory informatics software: LIMS, ELN, LIS and lab automation.