

The company asset list, in one place

A plain-English plan for replacing three (or five) department spreadsheets with one internal record that IT, HR and the vendor can all read, compare and update — with a log of who changed what.

Prepared for review · October 2026 · Internal use only · Discussion document, not a commitment

The short version: the portal is the easy part. Before any software gets built, the company has to agree on *what a piece of equipment is* — one record per device, one serial number as its identity, and a clear owner for each field. Once that exists, every department's spreadsheet becomes an import into the same record instead of a competing version of the truth. Get that wrong and a portal just gives us five places to be wrong.

1 What's actually wrong today

This isn't a guess — it's visible in the two reports that were shared. Both are the same kind of report, exported from the same system at the same time, and they don't agree with each other on structure:

- **Same data, different columns.** One file calls the column `Device Status Name`, the other calls it `Device Status`. The column order is different. Anyone comparing them by eye is doing translation work, not analysis.
- **Columns drift inside a single file.** In the aged report, one row has the operating system text (`Microsoft Windows 11 Pro`) sitting in the *Manufacturer* column. That's a spreadsheet doing spreadsheet things. A database won't allow it.
- **Seven different words for "where is it."** `In Use`, `In Stock - Client`, `In Stock - DeepNet`, `Spare - Client`, `Decommission Prepared`, `Likely need replacement soon`, `already e-wasted`. Two of those are statuses, one is an opinion, one is a sentence. Nobody can total that column.

- **59 devices with no age data.** The aged report has a bucket literally called *Unknown* (9), plus blanks in serial numbers, models and manufacturers. Those are the machines nobody can account for, and they're invisible in every summary.

None of this is anyone's fault. It's what happens when a report is designed to be *printed* instead of *queried*. The fix is structural, not a cleaner spreadsheet.

2 The idea, in plain English

Right now the company keeps several card catalogues for the same library, and each one is organised differently. Ours gets organised like this:

- **One card per device.** A laptop is one record, not one row per department. It is identified by its *serial number* — the only piece of data that doesn't change, can't be mistyped twice the same way, and is physically on the machine.
- **Each department owns its own fields on that card.** IT owns status, warranty, OS and who it's assigned to. HR owns the person, their department, and whether they still work here. The vendor owns warranty and last-seen data. Nobody overwrites anyone else's column, because nobody else's column is on their screen.
- **Every change is written down.** Who, what, before, after, when. Like a sign-out sheet that can never be edited after the fact.
- **Every spreadsheet becomes an import.** A vendor export, an HR roster, a finance list — each one lands in a staging area first, gets matched to the right card by serial number, and a human approves it. Nothing lands directly in the live record.

Who owns which field

This little table is the actual design. Everything else is plumbing.

FIELD	OWNER	WHO CAN SEE IT	WHY IT MATTERS
Serial, model, manufacturer	IT	All	The identity. If this is wrong, nothing else can be trusted.
Device status (in use / stock / retired)	IT	All	The core question: where is it right now.
Warranty dates	IT VENDOR	IT, Finance	Budget and replacement planning.
Assigned person	IT	All	The join between a device and a human.
Employee, department, employment status	HR	IT, HR, managers (own dept)	Detects leavers still holding equipment.
Cost, purchase order, depreciation	FINANCE phase 2	Finance, IT	Optional in phase 1. Usually the next department to ask for access.

THE RULE THAT SAVES THE WHOLE PROJECT

One field, one owner. If two systems both claim to own "assigned person," you don't have one record — you have two records that disagree, forever. Everything that looks like a "sync problem" in these projects is really a "nobody decided" problem.

3 What it would look like

Four screens, and that's it. The mockup that came with this document shows the third one in detail.

1 Assets

— a searchable list of every device. Filter by status, department, warranty expiry. This replaces the "can you send me the latest spreadsheet" email.

2 Imports

— drop in the vendor export or the HR roster, confirm the column mapping once, preview the changes, then approve. Repeat imports remember their mapping, so month two is one click.

3 Reconciliation

— the screen that earns the project. Side-by-side view of what each source says about the same device, with the disagreements pulled to the top by rule: *in stock but HR has*

a user, leaver still assigned, no serial number, in use past warranty. This is the thing that gets used weekly. Not data entry — exception handling.

4 Audit log

— every change, filterable by person, device or date. Read-only, nobody can edit history.

The mockup is HTML, opens in any browser, no install. The HR and vendor columns in it are invented for illustration — the IT values and the device list come from your two files.

4 Buy or build — the honest answer

At ten hours a week, this matters more than the technology choice. Three real options:

OPTION	WHAT YOU GET	EFFORT	RISK
Snipe-IT free, open source, self-hosted	Asset list, check-in/out, serial tracking, CSV import, AD/LDAP login, audit log, API, runs on PHP + MySQL — the same stack you already work in.	1–2 weeks to install, brand and import. Then add the reconciliation view as a custom page on top of its API.	Low. It's mature and widely used. You'd be configuring, not inventing.
GLPI free, open source	The same, plus a helpdesk/ticketing module built in.	Similar, slightly heavier setup.	Low-medium. More features than needed; more to learn and patch.
Custom portal Laravel + MySQL	Exactly the workflow the company actually has, nothing more.	6–10 weeks of part-time work for phases 1–4.	Medium. You own the bugs, the security and the documentation.

RECOMMENDATION

Start with **Snipe-IT** unless there's a workflow it genuinely can't express. It gets 80% of this done in two weeks, which leaves your ten hours for the part that's actually unique to this company — the cross-department reconciliation. If they later want something bespoke, you build that *on top of* a system that already works, instead of starting from an empty folder.

ASK THE VENDOR FIRST

Before choosing anything, ask the MSP what RMM or inventory tool they run — ConnectWise, Datto, Lansweeper and similar all hold an asset inventory already. If they do, an API feed from their system is a far better data source than any Excel export, and it's already being updated daily. Don't rebuild what the vendor already pays for.

5 Security, in plain English

Logins: use the network account, not a new password

The portal can authenticate against Active Directory. In practice: people log in with the username and password they already use for their computer. Nothing new to remember, nothing new to reset, and when someone leaves, disabling their network account locks them out of the portal at the same time — automatically. The application never stores a password; it just asks the domain controller "is this person who they say they are."

Permissions come from AD groups too — put someone in Assets - IT, Assets - HR or Assets - Vendor and they get that role. Removal is one group edit, and it's auditable in AD as well.

Who sees what

ROLE	CAN SEE	CAN CHANGE
IT	Everything	Asset fields, imports, resolutions
HR	Device + assigned person + department. Not costs, not vendor notes.	Employee and department fields only
VENDOR	Only devices they're responsible for; warranty and last-seen data	Warranty / service fields only
Department manager	Their department's devices and people	Nothing — read-only
Auditor / owner	Everything, including the audit log	Nothing — read-only by design

The audit trail is the point

Every change stores who made it, when, from where, what the value was before and what it is now. The table that holds this can only be appended to — no editing, no deleting, not even by an administrator. When someone asks "who changed this serial number in March," the answer takes ten seconds instead of a meeting. Build it in from day one: it's about an hour of work at the start and a miserable retrofit later.

Keeping it internal

- **No internet exposure.** Internal-only address, no port forwarding, no public DNS entry, no vendor cloud dependency.
- **Encrypted in transit even internally.** An internal certificate is enough — the goal is that credentials and asset data aren't crossing the LAN in clear text.
- **Least privilege on the database.** The application's database user gets exactly the rights it needs: read and write rows, no schema changes, and no update or delete on the audit table.
- **Nightly backups to a different machine,** and test a restore once. An untested backup is a hope, not a backup.
- **Time-boxed vendor access.** Standing MSP accounts are the most common finding in an audit. Enable the account when they need it, disable it when they're done.

6 The rollout, in phases

Each phase should stand on its own, so if the project loses steam the company still has something useful.

PHASE	WHAT GETS DONE	ROUGH EFFORT	DONE WHEN...
0 · Agree	The field list, the serial number as the key, and one owner per field. Written down, one page, signed off by IT and HR.	2–4 hours + one meeting	IT and HR both nod at the same page
1 · Get it standing	Pick the tool, stand up a VM, install, back it up, get it answering on the internal network.	1 week	You can log in and see an empty asset list
2 · Load the data	Import the two reports you already have. Clean serials, agree the status list, dedupe. This is the messy phase — budget for it.	1–2 weeks	~59 devices loaded, no blanks in serial
3 · Give people access	AD groups, roles, permissions. IT first, then HR read-only, then the vendor.	2–3 days	An HR user can log in and see only what they should
4 · Reconciliation	The side-by-side view and the four starting rules. This is the deliverable people will actually ask for.	1–2 weeks	A weekly exception list is being used
5 · Harden	Backups tested, audit review, documentation a non-author can follow, handover notes.	3–5 days	Someone else could run it without you

At ten hours a week, phases 0–2 is roughly a month. That's the honest number — most of the calendar time goes into agreeing definitions and cleaning data, not into software.

7 What not to do

- **Don't import straight into the live record.** Always stage, preview, then approve. One bad mapping can overwrite 59 devices in a single click.
- **Don't accept a device without a serial number.** A row without a serial can't be matched to anything, so it can never be reconciled. If it can't have a serial (a VM, a switch), it needs a separate asset ID scheme, not a blank cell.
- **Don't run this on MongoDB.** This data is relationships: device → person → department → warranty. It's genuinely better in a relational database, and reporting is far easier.
- **Don't run production on Laragon.** Laragon is a development environment and it's the right place to prototype. Production wants Apache/Nginx, PHP and MySQL installed as proper Windows services, or better, a small Linux VM with less to patch.

- **Don't let the scope creep into procurement or helpdesk.** Those are real needs and they'll arrive as "while we're in here..." requests. Note them, put them in phase 6, don't do them now.
- **Don't go live without the audit log.** Retrofitting history is impossible — the data you didn't capture at the start is gone.

8 Questions to answer before building

1. How many devices company-wide, and how many departments will want access? (The 59 in these reports look like one client or one site, not the whole company.)
2. Is there on-premises Active Directory, and can IT provide a read-only service account for the login integration?
3. What does the MSP use for inventory — is there an API, or only exports?
4. Does an asset change ever need approval, or can IT just make it?
5. Who owns the definition of "in stock" versus "spare" versus "decommissioned"? Someone has to, or the seven statuses come back.
6. Is there any existing asset system Finance or Purchasing already uses? Merging with it later is much harder than importing from it now.

9 If you have to say it in one minute

"Right now every department keeps its own version of the asset list, in its own layout, and nobody can compare them without manual work. I want to start with something small: agree on one record per device, keyed on serial number, with each department owning its own fields. Then any spreadsheet becomes an import instead of a competing truth. The first deliverable isn't a portal — it's a weekly exception list that tells us which leavers still have laptops and which machines are in use past warranty. That's cheap to build, it's useful on day one, and it earns the right to build the rest."

If a VM gets pushed back: the fallback is a single internal Windows box running the same stack, or Snipe-IT on the existing server if there's headroom. The plan doesn't change — only the host does. What it must not

become is another spreadsheet with a nicer header.

Two source reports examined: All_Computers_with_Expired_Warranty_Client_Portal (18 rows) and All_Computers_Aged (~59 devices, 9 age buckets). Figures quoted come from those files. The companion mockup uses their real device data with invented HR and vendor columns for illustration.