BUSINESS OVERVIEW · NON-TECHNICAL SUMMARY

AI-Assisted Software Upgrade — How It Works

Our web applications are built on a technology platform that, like all software, needs to be periodically updated. This document explains how we use AI to handle those upgrades in a structured, low-risk way — and where you, as a stakeholder, are involved.

February 2026

Process: Step 1: AI Plans → You Approve → Step 2: Safety Check → Step 3: AI Upgrades → Step 4: AI Tests → Step 5: Optional Next Steps

---

STEP 1 — AI REVIEWS THE SYSTEM AND BUILDS A PLAN

Before anything changes, the AI reads through the current state of the software — what version it is running, what other tools it depends on, and how the code is currently structured. It then reads the official documentation for the new version to understand what is different and what might break.

What the AI looks for:

- Which version of the software is currently installed, and which version we are upgrading to
- Any tools or libraries that depend on the old version and may need updating too
- Features or functions that have been removed or changed in the new version
- Whether the server environment meets the minimum requirements for the new version

What you receive:

A written **Migration Plan** in plain language, covering: what will change, what the risks are, an estimate of how long each step should take, and how to undo the upgrade if something goes seriously wrong.

> **Human Approval Required — Gate 1.**
> The AI stops completely at this point. Nothing is changed until you review the plan and give the go-ahead. You can ask questions, request that steps be changed, or decide to pause the upgrade entirely. You are in full control.

STEP 2 — SAFETY CHECKS BEFORE ANYTHING CHANGES

Once you have approved the plan, the AI performs two safety checks before touching any file:

1. **Creates a backup snapshot.** The current state of all code is saved so that, if anything goes wrong, everything can be restored to exactly how it was before. This is similar to creating a “restore point” on a computer before installing updates.
2. **Checks the server environment.** The AI confirms that the server or computer running the application meets the minimum requirements for the new version. If it does not, the upgrade is paused immediately and you are notified — no changes are made.

STEP 3 — AI CARRIES OUT THE UPGRADE

The AI works through the upgrade step by step, in the exact order set out in the approved plan. It keeps a live checklist so you can see which steps are complete and which are in progress.

What happens if something goes wrong?

The AI does not simply stop at the first sign of a problem. It follows a structured process before deciding whether to fix the issue itself or ask for your help:

Level	What the AI does
First attempt	The AI tries the most common automatic fixes: adjusting settings, updating references to renamed features, or retrying the step with alternative options.
Second attempt	If the first attempt does not work, the AI searches official documentation and known issue records for the specific problem, then applies the documented solution.
Escalate to you	If neither attempt resolves the problem, the AI stops at that step. It writes a clear summary document listing: what was completed successfully, what the problem is, what still needs to be done, and suggested next steps for a developer. The AI then notifies you and waits for guidance.

If escalated, no further changes are made. The completed work is preserved and the remaining steps are clearly documented so a developer can continue from exactly where the AI stopped.

STEP 4 — AI TESTS EVERYTHING AUTOMATICALLY

After the upgrade (whether complete or partial), the AI runs a standard set of quality checks without any manual effort required. Think of this as the AI doing a structured inspection of the application before handing it back.

How the AI decides what to fix vs. what to escalate:

If a check finds problems, the AI decides how to respond based on how many issues were found:

Number of Issues Found	What the AI Does
None	Check passed. Move on.
A few (fewer than 10)	AI fixes each one individually, re-runs the check to confirm, then continues.
A moderate number (10 to 50)	AI fixes the recognisable, common issues, re-runs the check once, then reports what remains and continues to the next check.
A large number (more than 50)	**AI stops attempting fixes.** A list of all issues is saved to a handover document. The AI finishes running the remaining checks (to get a full picture) but makes no further changes. You are then notified and the decision is yours on how to proceed.

Why stop at 50 issues? At that scale, the problems are almost always caused by something fundamental — a third-party tool that is not yet compatible with the new version, or a decision that requires human judgement. Trying to automatically fix hundreds of issues would risk making things worse and could take an unreasonably long time.

What gets checked:

Check	What it verifies	If it fails
Code integrity	No coding errors that would prevent the application from being built	Apply the issue-count policy above
Test version build	The application compiles and packages correctly in a development environment	Apply the issue-count policy above
Live version build	The application is ready to be deployed to production	Skipped automatically if the test build failed with too many issues
Code quality	The code meets defined quality standards; common issues are auto-corrected	Apply the issue-count policy above to any remaining issues
Application startup	The application actually starts and runs without crashing	Skipped if the build failed; otherwise the error is recorded
File size check	The application has not become significantly larger (which would slow it down for users)	Flagged for review if the size increased by more than 20%

A summary report is produced showing the result of each check, how many issues were found, and whether they were fixed, flagged, or handed over.

STEP 5 — OPTIONAL RECOMMENDATIONS

After completing the upgrade and testing, the AI produces a list of optional improvements that were outside the scope of the upgrade itself but are worth considering. These are prioritised into three groups:

Priority	Examples
Should do soon	Modernise parts of the code that still use older patterns; enable stricter quality checks; update outdated functions to current equivalents
Do when possible	Clean up unnecessary code; adopt newer, faster ways of writing logic; switch to a faster build tool
Nice to have	Install developer productivity tools; update the automated testing setup to a more modern framework

> **Your decision — Gate 2.**
> This list is informational only. You decide which items (if any) to act on. If you choose to proceed with any of them, a separate, focused process is started for those specific improvements.

DOCUMENTS PRODUCED DURING THE PROCESS

Document	When you receive it	Purpose
Migration Plan	End of Step 1	Full description of the upgrade: what changes, what the risks are, how long it takes, and how to roll back. Requires your approval before anything proceeds.
Progress Checklist	During Step 3	A live record of every task in the upgrade, updated as each step is completed.
Handover Document	Only if the AI gets stuck	A clear summary of what was done, what the problem is, what is left to do, and suggested next steps for a developer to continue.
Recommendations	End of Step 5	Prioritised list of optional improvements to consider after the upgrade.
Completion Summary	End of process	Final record of everything that was upgraded, the test results, and any outstanding items.