Investigation Report: Plain English Overview

Date: 2025-06-08

Executive Summary:

Between June 5 and June 7, our application experienced unexpected downtime of certain features. This happened because a small but important piece of code was accidentally removed, and our automated checks did not catch the problem. This report explains:

- What happened and when
- What we thought we had updated but had not
- Why the problem occurred in the first place
- How we fixed it
- How we will prevent similar issues in the future

1. What Happened

In our efforts to streamline and improve our application, we made several coordinated changes to our code and deployment process. Unfortunately, a fallback mechanism that keeps our health-monitoring feature running failed to make it into the final version. Without that mechanism, the health endpoint began returning errors in production.

2. Timeline of Key Events

- **June 5, 2025 (5:49 PM PDT):** We performed a significant cleanup of the code base, which inadvertently removed the backup code that helps our system gather performance information.
- **June 6, 2025:** We simplified how our application starts up, accidentally limiting it to only a subset of its usual functionality. This change removed many features from the active application.
- **June 7, 2025 (8:45 AM UTC):** Users began seeing errors when checking the system health status. This was the first sign that the missing code path caused the application to fail in production.

3. What We Thought We Had Updated But Did Not

- **Health Monitoring Backup Code:** We believed we had included a fallback for health checks but never committed that code.
- **Automated Test Coverage:** We intended for our continuous integration (CI) system to run all tests, but one critical test step was missing.
- **Full Application Startup:** We assumed all features would start normally, but our streamlined startup script only loaded a minimal set of functions.
- **Working Directory Cleanup:** We expected old environment files and scripts to be ignored, but leftover files cluttered our workspace and distracted our review.

4. Root Causes

The problem can be attributed to three main factors:

1. **Accidental Removal:** In a broader code cleanup, an important fallback for health monitoring was removed, and this slipped through undetected.
2. **Incomplete Testing:** Our automated CI process did not include a full test run, so no tests failed when the code was missing.
3. **Miscommunication:** Multiple simultaneous changes—cleanup, refactoring, and CI updates—made it hard to track exactly what was live in production.

5. How We Fixed It

As soon as the issue was identified, we took the following steps:

- Restored the missing health-monitoring fallback in the code.
- Updated our CI pipeline to include a complete test suite on every code change.
- Reconfigured the application startup so it loads all its features by default.
- Cleaned up our repository to ignore unnecessary files and avoid distractions.

6. Preventing Recurrence

To ensure this never happens again, we will:

- **Enforce Full Test Runs:** Every code change must pass the entire test suite before it can be merged.
- **Automated Checks for Uncommitted Work:** Our CI will now fail if there are any uncommitted or untracked files in the repository.
- **Review Checklist for Releases:** We have introduced a simple checklist that requires sign-off on critical areas (fallback mechanisms, feature availability, test coverage) before deployment.
- **Team Training and Documentation:** We will hold a brief training session to walk everyone through these new steps and update our internal documentation accordingly.

By following these measures, we are confident that similar issues will be caught early in the future, keeping our application reliable for all users.