Case Study

AI-Assisted Testing for a Rapidly Evolving K-12 Enrollment Platform

Magic EdTech

The Client

The client is a leading K-12 education technology provider building enrollment and administrative software for school districts in the U.S.

The Challenge

The client's new K-12 enrollment platform was still evolving, with workflows and requirements changing frequently in response to stakeholder feedback. This made traditional test automation impractical, while manual exploratory testing took 8-12 hours per cycle and relied heavily on individual tester judgment and documentation. The team needed a lightweight AI-assisted testing model to improve consistency, traceability, and reporting without limiting exploratory flexibility.

Critical Success Parameters

- ✓ Reduce repetitive manual effort across exploratory testing cycles.
- ✓ Support testing on an unstable application where requirements and workflows changed frequently.
- ✓ Improve consistency across exploratory testing sessions.
- ✓ Capture test actions, screenshots, UI structure, observations, and findings.
- ✓ Reduce post-session documentation effort.
- ✓ Separate confirmed defects from product observations.
- ✓ Maintain human review before accepting AI-generated findings.
- ✓ Keep the framework lightweight and reusable across similar projects.
- ✓ Improve AI output quality while managing hallucinations and context-window limits.

Our Approach

- ✓ Built a lightweight AI-assisted exploratory testing framework for the enrollment platform.
- ✓ Connected an AI coding assistant to a live browser through a browser automation MCP.
- ✓ Created structured context files to guide AI behavior, application understanding, and session strategy.
- ✓ Used a file-based setup with instruction files, project context, session prompts, test cases, session logs, screenshots, and UI snapshots.
- ✓ Enabled the AI assistant to execute real browser actions such as navigation, form entry, clicks, validation checks, screenshots, and UI structure capture.
- ✓ Ran exploratory sessions where the AI read the context, tested defined flows, observed outcomes, and documented findings in real time.
- ✓ Added human checkpoints to review new features, observations, and potential defects before updating the framework context.
- ✓ Separated confirmed bugs from standards-based observations that required product owner review.
- ✓ Refined prompts and context files when the AI misunderstood flows or produced incorrect results.
- ✓ Shared validated observations with product owners, who converted relevant items into improvement user stories.

Key Result Highlights

Reduced exploratory testing effort from 8-12 hours to approximately **4-5** hours per cycle.

Saved **2-3** hours of manual documentation effort by generating session reports during test execution.

Improved testing traceability through session logs, screenshots, UI snapshots, observations, and findings.

Identified hidden consistency issues missed during manual testing, including an incorrect date-format pattern.

Enabled exploratory testing for an application that was changing too frequently for traditional automation.

Created a reusable AI-assisted testing model that other internal and client-side teams showed interest in adopting.