

EN.601.727 Machine Programming — Assignment 3

Building WAA and Three Web Apps with an LLM

Shaobo Liang, sliang24@jh.edu

October 24, 2025

Abstract

This report documents the design, implementation, and results for Assignment 3. I implemented a Web-App Agent (WAA) with tool execution, stateful history, and deterministic tool-call protocols; then used it to build (1) a personal website, (2) a chat room with RESTful APIs and a dynamic UI, and (3) a creative project: a *visual novel reader* that reveals one sentence at a time with persistent progress and a white background using a static SVG.¹

1 Overview

Goal. Build an agentic system that reads instructions from `.waa/instruction.md`, calls tools (`fs.*`, `npm.*`, `playwright.*`, `supertest.*`, `todo.*`), edits files, runs servers/tests, and iterates until success or max turns. Then apply it to three target apps.

Highlights.

- Deterministic tool calling: `<tool_call>{...}</tool_call>` and `<terminate>` protocols.
- Secure FS tools with working-dir confinement & protected-files checks.
- Robust logs: agent queries/responses, tool calls/results, server logs.
- Practical debugging in WSL: port collisions, Playwright deps, protected-file guardrails.

2 Agent Architecture

2.1 Core Loop

1. Initialize tool registry and environment (working dir, protected files).
2. Load system prompt and user instructions into history.
3. Repeat up to `max_turns`: query LLM \rightarrow parse \rightarrow execute tool or terminate.

2.2 Tools Implemented

File system: `fs.write`, `fs.read`, `fs.edit`, `fs.delete`, `fs.mkdir`, `fs.rmdir`, `fs.ls`, `fs.tree`.

Server/tests: `npm.init/start/stop/status/logs`, `supertest.init/run`, `playwright.init/run`.

TODO: `todo.add/list/complete/remove`.

2.3 Safety

- All paths resolved relative to working dir; out-of-tree paths rejected.
- `protected_files` from `.waa/config.json` cannot be modified.

¹Assignment structure and requirements referenced from the course handout. :contentReference[oaicite:0]index=0

3 Targets and Results

3.1 Personal Website

Spec. Static `index.html` with a dark/monospace aesthetic, skills (Rust, TypeScript), and “About Me” content. **Result.** Built by WAA; Express serves `public/`. UI tests pass.

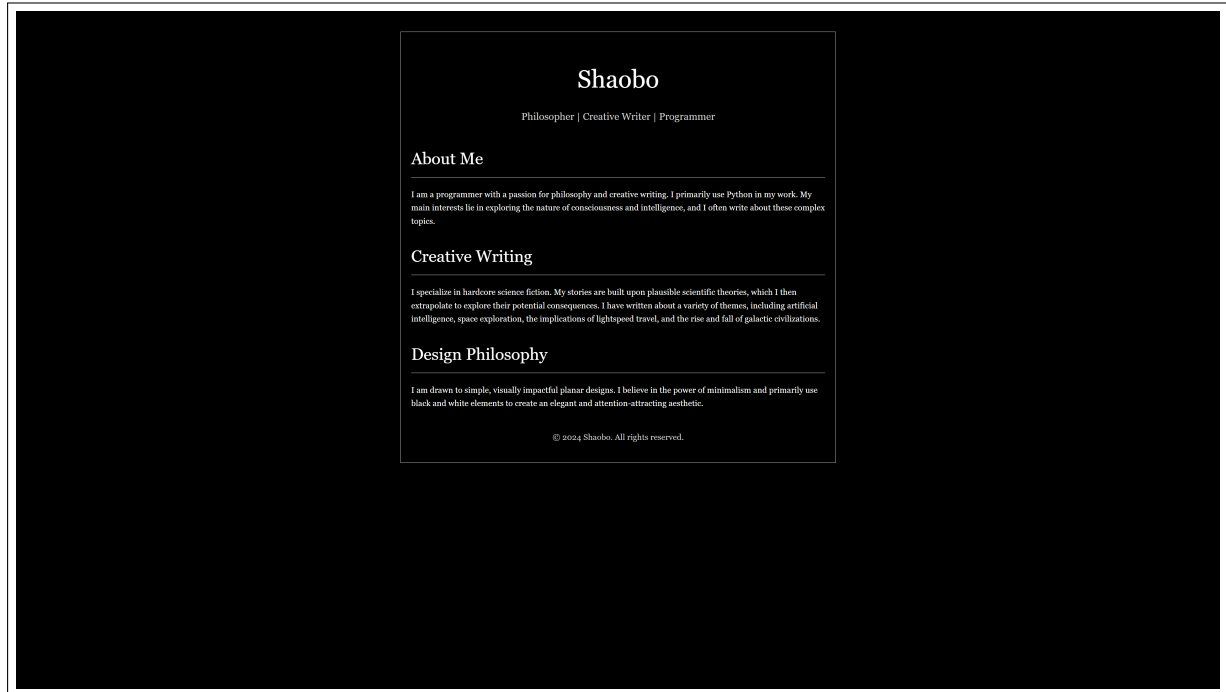


Figure 1: Personal website homepage (screenshot).

3.2 Chat Room

Spec. REST APIs: `GET /api/messages`, `POST /api/message`, `DELETE /api/messages`; handlebars-rendered UI; periodic refresh; clean black/white UI; separate CSS/JS. **Notes.** Ensured `views/layouts/main.h` exists (layout), Express static serving, and Playwright/Jest setup. Managed WSL port conflicts (`EADDRINUSE: 3000`) by terminating stale processes.

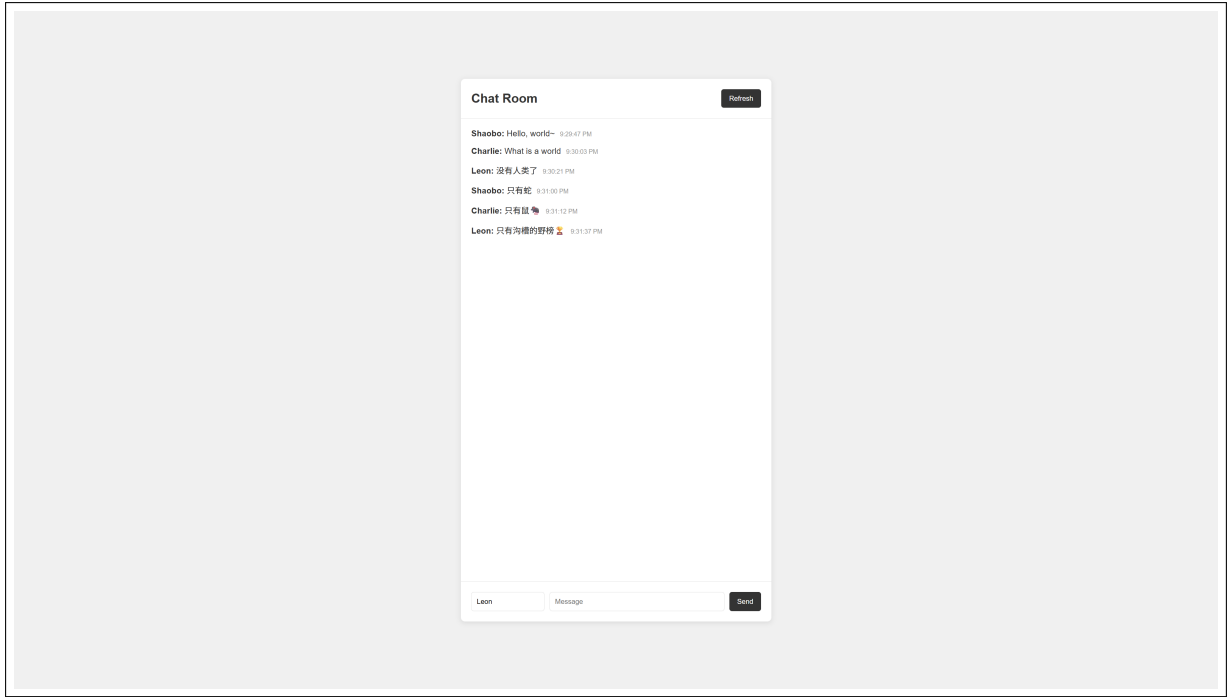


Figure 2: Chat room UI with messages and form (screenshot).

3.3 Creative Project: Visual Novel Reader

Spec. Single-page app: click to advance *one sentence at a time* at the bottom panel; stores progress (localStorage); white page background with a static SVG at `public/assets/background.svg` (CSS: center/cover/fixed). **Story Source.** My short story (provided as input to the agent). **Result.** WAA created `index.html`, `style.css`, `script.js`, SVG asset, and wired persistence.

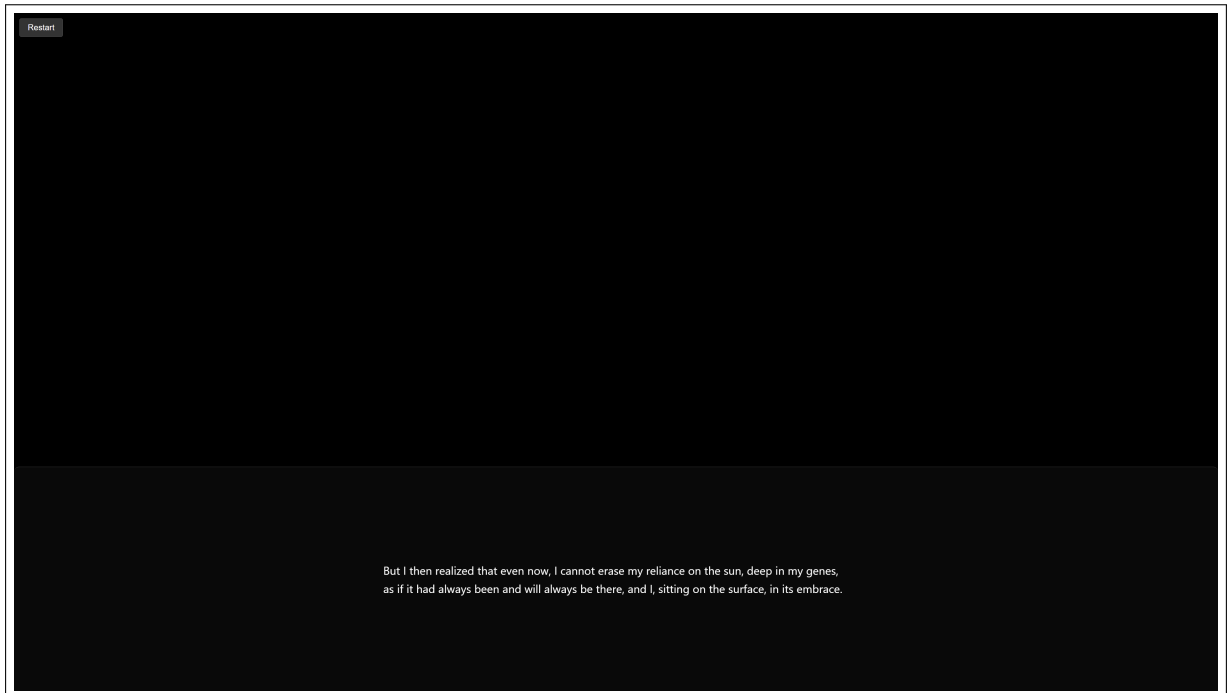


Figure 3: Visual novel reader showing sentence panel (screenshot).

4 Experiments and Turn Counts

Target	Model	Max Turns	Actual Turns	Outcome
Personal Website	Gemini 2.5 Pro	50	~ 10	Pass (UI tests)
Chat Room	Gemini 2.5 Pro	50	~ 20	API/UI tests status: --
Visual Novel	Gemini 2.5 Pro	60–75	~ 10	Functional

Table 1: Agent runs summary.

5 Engineering Notes

5.1 Key Issues and Fixes

- **Protected files.** When FS writes were blocked, it stemmed from `protected_files` and/or missing server stop before edits. Resolution: adjust config and ensure `npm.stop` prior to edits to files in use.
- **WSL port conflicts.** `EADDRINUSE: 3000`. Fix: kill stale node processes or run `npm.stop`.
- **Playwright dependencies.** If host lacks browser deps, initialize with `playwright.init` and (if allowed) install system deps; otherwise focus on API tests or mock where specified by assignment.
- **Layouts for Handlebars.** Missing `views/layouts/main.handlebars` caused startup failure; instruction update asked WAA to create the layout file explicitly.

5.2 Instruction Authoring Tips

- Be explicit about *required files* and *paths* (e.g., “create `views/layouts/main.handlebars`”).
- Spell out server behavior: serve `public/`, respond / with `index.html`.
- Include a small “Build Protocol” checklist: inspect → create missing files → start server → run tests → iterate.

6 What Worked Well vs. Challenges

Worked well.

- Deterministic tool-call protocol simplified parsing and execution.
- Minimal, explicit edits with `fs.read` → `fs.edit` reduced accidental overwrites.

Challenges.

- Environmental constraints (browser deps, permissions) can derail UI tests.
- LLM occasionally attempts to edit protected or non-existent files; logs were crucial to iterate.

7 Acknowledgements

I did not collaborate with humans. I used GPT-5 to help me make plans, program, edit prompts, and write.