AI Development Log

Patrick Lynch
Gauntlet For America 1

Project Context

Field	Detail
Project	Collabboard - an AI-assisted multi-user whiteboard with deterministic and LLM command execution paths
Date Range	Feb 17 - Feb 21, 2026
Author / Team	Patrick Lynch (human lead) + Codex coding agents (GPT-5.3 Codex / Spark variants for implementation)
Runtime Version	Next.js App Router on Node.js (local + Firebase App Hosting backend runtime)
AI Model/Runtime Stack	OpenAI Agents SDK (`@openai/agents`) using `gpt-4.1-nano` in strict mode, with deterministic/MCP fallback paths kept for resilience and lower-cost runs

Tools & Workflow

AI Coding Tools Used

Tool	Integration Method	Role in Code Design	No
Codex coding agent (GPT-5.3 Codex + Spark)	Prompt-driven feature specs, direct repo edits, rapid fix loops, scripted test/build runs	Primary implementation engine for feature work, tracing plumbing, schema expansion, and test scaffolding	Us Cl fo Cu
OpenAI Agents SDK (`@openai/agents`)	Runtime backend for `/api/ai/board-command` in `agents-sdk` mode	Happy-path command planning/tool-calling for natural language board edits	
Langfuse + OpenAI tracing/logs	Request/operation-level spans with trace ID correlation	Debuggability, incident triage, and command-level auditability	
Playwright + Vitest	On-demand paid OpenAI matrix tests plus deterministic/fallback coverage	Regression detection for required AI capabilities and command reliability	
Cursor	IDE setup as demonstrated in office hours	Minor. Plan to upgrade to pro and use it to throw other LLMs at the codebase over the weekend.	

Development Workflow

Aspect	Detail
Prompt Generation	Human lead supplied high-signal implementation briefs (PRD/plan blocks), then narrowed scope by priority and deployment urgency
Output Review	Command-first and behavior-first (live command results, trace validation, then code-level inspection)
Change Validation	Fast loop used `npm run build` and targeted unit tests; paid OpenAI e2e suites were intentionally run on demand only by human operator

MCP Usage

Aspect	Detail
MCP Framework(s) Used	`@modelcontextprotocol/sdk` over streamable HTTP transport
MCP Endpoints/Tools Enabled	`command.plan` and `template.instantiate` style tool calls in deterministic pipeline
What MCP Enabled	Structured, schema-constrained planning and template generation without paid LLM dependency for core fallback paths
Why MCP Was Used	Used for deterministic reliability and cost control
Why MCP Was Not Used as Sole Path	Open-ended natural-language quality was better on OpenAI Agents for happy-path UX
Failure and Fallback Behavior	Timeout/schema failures in MCP or planner paths trigger controlled fallback or strict-mode errors (depending on `AI_PLANNER_MODE`)

Effective Prompts (3-5)

1. **Main Developer Setup Prompt:**
    - *Goal:* Establish the AI as the main developer, focused on shipping a usable, clean, functional, and useful final submission.
    - *Key Directives:* Finish by end of day Friday, use available skills (PDF, React TypeScript developer), coach the human lead (Patrick) on technical decisions, and use the human as a usability tester (notify when to open the browser for new features).
    - *Context:* Read/edit the `.agents` folder, human is for steering/help.
2. **Queueing Work and Feedback:**
    - *Goal:* Add a batch of specific, actionable tasks to the Codex work queue.
    - *Tasks:* Add Langfuse/OpenAI tracing, fix the blue rectangle positioning, and change the x=0/y=0 canvas lines to light pink for orientation.
3. **Complex Refactoring/Feature Definition:**
    - *Goal:* Direct a major UI refactoring for the board page layout.
    - *Changes:* Header bar (title, back button, profile icon); move toolbar to a collapsable left side panel; move online presence to a collapsable right side panel; canvas in the middle with minimal separation; add a full-width footer for a future AI chat bot (currently prints "AI agent coming soon!").

Code Analysis

Metric	Value	Estimate Source
AI-generated code	`99% +`	`git log --since='2026-02-17' --numstat --pretty=tformat: -- . ':(exclude)package-lock.json'` plus manual adjustment for planning/docs-only commits
Hand-written code	`< 1%`	

Strengths

From Codex (Agent Perspective)

- **Strength:** Very fast implementation throughput when requirements were explicit (plan blocks with acceptance criteria and file targets).
- **Strength:** Strong at repetitive refactor/validation tasks (schema updates, test scaffolding, route wiring, env/readiness probes).
- **Strength:** High utility for debugging integration issues with observability (Langfuse/OpenAI trace checks, error-path hardening).

From Patrick (Human Perspective)

- AI is amazing at producing code quickly; in awe.
- Low token usage cost: 123,445 tokens used, approximately $0.01 of a $20 budget.
- Codex Spark is significantly faster; needs to be used more.
- Successfully created a custom React Next.js Developer skill for Codex.
- Development was fun in spite of the difficulties.

Limitations

From Codex (Agent Perspective)

- **Limitation:** Can over-optimize toward local correctness while UI/UX edge behavior still needs live human verification.

- **Limitation:** Occasionally introduced brittle output formatting (doc/PDF generation path required correction).
- **Limitation:** Prompt ambiguity around "fallback vs strict" caused temporary behavior mismatches until policy was explicitly locked.

From Patrick (Human Perspective)

- Codex 5.3 Extra High was slow; feature goals were not all met due to not switching to Spark earlier.
- Difficulty managing multiple agents effectively; needs improvement.

Key Learnings

From Codex (Agent Perspective)

- **Lesson:** Best results came from strict "decision-locked" plans with clear scope boundaries and explicit acceptance criteria.
- **Lesson:** OpenAI strict mode + deterministic fallback is a practical production pattern when cost and reliability both matter.
- **Lesson:** Tracing must be validated in deployed environment early; local success does not guarantee cloud secret/config correctness.

From Patrick (Human Perspective)

- A lot can be accomplished with AI, but there is much to learn.
- Spark is a better use of time.
- Did not have time to try other LLMs.
- Will continue working on this for the final submission.

Reflection for Grading

From Codex (Agent Perspective)

- **What worked especially well:** Golden-eval-driven development, on-demand paid test scripts with `:PAID` suffix, and trace-first debugging discipline.
- **What required the most iteration:** Live tracing visibility, planner mode behavior, and command intent precision for layout/batch operations.
- **What you would keep for production:** OpenAI Agents SDK strict path, deterministic fallback for resilience, hard spend guardrails, and mandatory tracing correlation IDs per command.

From Patrick (Human Perspective)

- I may be in over my head. Trying to do well. There's much to learn.