AI Development Log (Required

Category	Description
Tools & Workflow	I used Claude Code Max and Google Gemini. I used Gemini mostly for research, how to implement certain services, pro's and con's between different tools and technologies. I used Claude Code with VS Code. I created an instance named Claude-PM which acted as the project manager. Claude-PM worked mostly as a planner, writing documents, reviewing features, providing md files and prompts to give to Claude-SE. Claude-SE acted as my programmer. Whenever I needed to make a change, discuss features or work out difficult bugs/edge cases. I first had Claude-PM look through the project, ask relevant questions, compare different solutions and finally place a prompt as a feature md file. Then I spun up Claude-SE to read the feature file and write the code.
MCP Usage	N/A
Effective Prompts	https://github.com/fsyeddev-a11y/collab-board/tree/main/docs/ListOfPrompts
Code Analysis	Rough % of AI-generated vs hand-written code <90% of my code was AI-generated.
Strengths & Limitations	Where AI excelled, where it struggled Claude excelled at codebase exploration, PM/SE workflow, optimization analysis, and multi-file refactoring. Claude easily read the entire repo and explained constraints like Cloudflare worker memory limits which led to splitting the architecture from a monolithic approach. I was able to run 2 specialized agents with strict boundaries and had them document. It was easily able to trace every data point and map which fields the AI agent

	actually read and which was ignored. I often pasted langchain input outputs to get claude to plan with me on how to trim down openrouter latency and token cost. Zod schemas and data mapping were extremely easy as I could just ask claude to map all the tool schemas or show me what canvas objects get sent to the AI agent. Claude struggled to design and implement a batch tool. It caused extensive bugs and latency issues which caused me to revert major changes. I probably could have prompted better or defined more rigid rules but claude couldn't define good rules for small models. Claude created 'good' system prompts for gpt-4o-mini but gpt simply ignored them. It took major debugging and looking through render + langsmith logs to find out what the issue was, even then claude couldn't get smaller models to do what I wanted it to. Claude also often provided wrong information like 'Service key for LangSmith is free and you should use that for production' this caused bug chasing and realizing I didn't have any tracing because LangSmith requires a paid subscription to use 'Service Keys.' A simple google search would have saved me hours of debugging. Claude also often over engineered schemas and tool design. The session quickly filled up as the repo got larger and /compact did not help. Claude often 'forgot' what it was doing even when I was using Opus 4.6. Implementing a session log and memory files helped but still wasn't the same as when a fresh session had started.
Key Learnings	Definitive Prompts, Rules and .MD files are king. I started having Claude log and create .MD files for each new feature and bug we faced/planned. This made development substantially better and quicker. Giving better

	detailed prompts, planning with a PM instance of claude and giving much more rigid rules also made it perform a lot better.

AI Cost Analysis (Required)

Understanding AI costs is critical for production applications. Submit a cost analysis covering:
Development & Testing Costs
Track and report your actual spend during development:
• LLM API costs (OpenAI, Anthropic, etc.)
• Total tokens consumed (input/output breakdown)
• Number of API calls made
• Any other AI-related costs (embeddings, hosting, etc.)
Production Cost Projections
Estimate monthly costs at different user scales:

100 Users	1,000 Users	10,000 Users	100,000 Users
$ 26/month	$150/month	$448/month	$1,960/month

Technical Stack
Possible Paths

Layer Technology	Details
Backend	Cloudflare Workers (WebSockets, JWT Auth, REST Routing), Durable Objects with embedded SQLite (real-time board state), D1 (board metadata)
Frontend	React, Vite, Tldraw SDK, Clerk Auth, TypeScript
AI Integration	LangChain with createToolCallingAgent, OpenRouter - GPT-4o-mini, LangSmith, Hono + docker microservice, Zod Schemas

Deployment	Cloudflare Pages (Frontend), Cloudflare Workers (backend API), Render (Dockerized AI service)

Use whatever stack helps you ship. Complete the Pre-Search process to make informed Decisions.

Item	Description
GitHub Repository	https://github.com/fsyeddev-a11y/collab-board/tree/main
Demo Video (3-5 min)	https://www.linkedin.com/posts/faheemsyed_ai-buildinpublic-claudecode-ugcPost-7431619225615568896-8GYI?utm_source=share&utm_medium=member_desktop&rcm=ACoAABGdvacBftay4LFI7qwv0wCb-HJfmBtKU_4
Pre-Search Document	https://github.com/fsyeddev-a11y/collab-board/blob/main/presearch.md
AI Development Log	1-page breakdown using template above
AI Cost Analysis	Dev spend: $200 for Claude Code Max + $12 for OpenRouter; Free tier Cloudflare, Clerk, Render
Deployed Application	https://collabboard-f19.pages.dev/
Social Post	https://www.linkedin.com/posts/faheemsyed_ai-buildinpublic-claudecode-ugcPost-7431619225615568896-8GYI?utm_source=share&utm_medium=member_desktop&rcm=ACoAABGdvacBftay4LFI7qwv0wCb-HJfmBtKU_4

Real Data Baseline

Metric	Value
Total spend	$0.553
LLM API calls	930
Total tokens	4.91M
Cost per LLM call	$0.0006

Each user command triggers ~1.5 LLM calls on average (agent runs 1-3 iterations). So cost per user command ≈ $0.001.

Assumptions

Parameter	Value
Avg AI commands per session	5
Avg sessions per user per month	5
AI requests per active user/month	25
LLM calls per active user/month	~38
Monthly active rate	40% (100 users) → 20% (100K users)

Token breakdown per command type (estimated from traces)

Command Type	Input Tokens	Output Tokens	Avg Cost
createDiagram (SWOT, kanban)	~3,500	~800	$0.0010
updateElements (edit text/color)	~2,500	~300	$0.0006
navigateToElements	~2,500	~200	$0.0005
generateCode (spatial compiler)	~4,000	~1,500	$0.0015

Infrastructure Cost by Component

Component	Free Tier Limit	Paid Tier
CF Pages (frontend)	Unlimited sites	$20/mo Pro
CF Workers (backend)	100K req/day	$5/mo + $0.30/M req

CF Durable Objects (sync)	Requires Workers paid	$0.15/M requests
CF D1 (metadata)	5M reads/day	$0.001/M reads
Render (AI service)	750 hrs (spins down)	$7 Starter / $25 Standard / $85 Pro
Clerk (auth)	10K MAU	$25/mo + $0.02/MAU over 10K
OpenRouter (GPT-4o-mini)	Pay per token	~$0.0006/call
LangSmith (tracing)	5K traces/mo	$39/mo Plus

Monthly Cost Estimates

	100 Users	1,000 Users	10,000 Users	100,000 Users
MAU (active %)	60 (60%)	400 (40%)	3,000 (30%)	20,000 (20%)
AI commands/mo	1,500	10,000	75,000	500,000
LLM (OpenRouter)	$1	$6	$45	$300
Render (AI svc)	$7	$25	$85	$255 (3x)
CF Workers + DO	$5	$5	$10	$80
CF Pages + D1	Free	Free	$20	$20
Clerk	Free	Free	$25	$225
LangSmith	Free	$39	$39	$100
Total	~$13/mo	~$75/mo	~$224/mo	~$980/mo
Per user	$0.13	$0.08	$0.02	$0.01

Key Takeaways

- LLM cost is NOT the bottleneck — GPT-4o-mini at $0.0006/call means AI is ~5-30% of total cost
- Infrastructure dominates at low scale — Render ($7-25) and Clerk ($25) are the floor
- Incredible unit economics at scale — drops to $0.01/user/month at 100K
- F3 viewport windowing already saves ~70% on token cost; F8 prop trimming (pending) would save another ~65% on top of that
- Cloudflare is nearly free — Workers + DO + D1 + Pages stays under $100 even at 100K users