CollabBoard Week 1 Submission Package

Date: 2026-02-16
Project: Gauntlet Cohort G4 - CollabBoard
Repository: https://github.com/appDevelopment-tech/gauntlet-cohort-1

Deliverables (Required)

- Deployed apps
- Demo video
- Pre-Search doc
- AI development log (1 page)
- LinkedIn or X post about what was done in 1 week
- AI cost analysis
- Doc submission in PDF format

1) Deployed Apps

Status: Live

- Production URL: https://mvp-1-collab-board.web.app
- Preview URL: n/a
- Auth mode for MVP: Google OAuth (Firebase Auth)

2) Demo Video

Status: In progress

- Target length: 3-5 minutes
- Must show:
    - real-time collaboration (2+ users)
    - multiplayer cursors + presence
    - conflict behavior under simultaneous edits
    - AI command execution (single-step and multi-step)
    - architecture overview and decisions summary
- Recording link: TBD

2.1 Test Evidence

Status: Complete

- File: TEST_EVIDENCE.md
- PDF: submission/TEST_EVIDENCE.pdf
- Latest artifact bundle:
    - submission/test-artifacts/latest-critical-checks.json
    - submission/test-artifacts/latest-critical-checks.log

- Includes validation for:
    - simultaneous AI commands from multiple authenticated users
    - FIFO queue sequencing
    - idempotency (clientCommandId)
    - throttled/disconnect retry behavior
    - 5-user authenticated command burst

3) Pre-Search Document

Status: Complete and strengthened

- File: PRESEARCH.md
- Source requirements: G4 Week 1 - CollabBoard-requirements.pdf
- Added based on feedback:
    - explicit BoardObject and CursorPresence schemas
    - explicit LWW/version conflict model
    - sync throughput and listener/index strategy
    - AI execution architecture with idempotency and FIFO ordering
    - offline/reconnect strategy details
    - concrete AI cost modeling and projections

4) AI Development Log (1 Page)

Status: Complete

- Standalone source: AI_DEVELOPMENT_LOG.md
- Standalone PDF: submission/AI_DEVELOPMENT_LOG.pdf

Tools and Workflow

- **Primary AI tools**: Claude (Anthropic), Cursor, Codex
- **Workflow**:
    1. Extract official rubric and hard-gate requirements from provided PDF
    2. Create structured planning docs (PRESEARCH, PRD, MVP, DECISIONS, TASKS)
    3. Create Linear issues mapped to execution timeline (MAX-19 through MAX-25)
    4. Implement React + Konva frontend with Firebase backend
    5. Deploy Firebase Functions for AI command dispatcher
    6. Write e2e tests with Playwright

MCP Usage

- **Linear MCP**: Created 7 implementation tickets (MAX-19 through MAX-25)
- **Chrome MCP**: Used for browser automation testing
- **Playwright MCP**: E2E test execution

Effective Prompts (examples)

1. "Review this requirements PDF and confirm whether it is the project requirements doc."
2. "Generate PRD, MVP, decisions log, and tasks from this rubric with hard deadlines."
3. "Identify missing requirements coverage and patch docs to close gaps."

4. "Fix drag offset bug where objects don't stay under cursor during drag."
5. "Implement AI board agent with 6+ command types for CollabBoard whiteboard."

Code/Docs Analysis

- **AI-generated**: ~85% (planning docs, boilerplate, Firebase Functions)
- **Hand-written**: ~15% (drag fix logic, custom Konva integration, test scenarios)
- **Planning vs code**: ~60% planning/docs, ~40% implementation

Strengths and Limitations

Strengths:

- Fast structure creation with clear rubric traceability
- Explicit architecture defense points in decisions log
- AI excelled at Firebase Functions pattern matching logic

Limitations:

- Drag bug required manual debugging (AI couldn't see runtime behavior)
- OAuth flow testing required manual browser interaction
- E2E tests need authenticated sessions (can't automate fully)

Key Learnings

1. **Explicit technical contracts reduce mid-build ambiguity** — BoardObject and CursorPresence schemas prevented confusion
2. **AI concurrency/idempotency must be designed early** — FIFO queue and locking added before first sync test
3. **Local drag state is critical** — Objects jumping during drag taught us to isolate local state from Firestore sync
4. **Testing strategy matters** — E2E tests caught routing issues that unit tests wouldn't

Session Statistics

- **Hours**: ~8 hours across 1 day
- **Claude Opus calls**: ~50
- **Files created**: 20+
- **Lines of code**: ~2000
- **Deploy services**: Firebase Hosting, Firestore, RTDB, Cloud Functions

5) LinkedIn/X Post Draft (1 Week Summary)

Status: Draft

Draft text:
"Week 1 at Gauntlet Cohort G4: we built a real-time collaborative whiteboard + AI board agent foundation with rubric-driven Pre-Search, PRD/MVP specs, decision logging, and Linear execution mapping. We upgraded architecture docs with explicit conflict strategy, sync performance model, AI command concurrency handling, and cost projections. Next: finalize MVP hard gate and demo. #GauntletAI #BuildInPublic"

6) AI Cost Analysis

Status: Complete with actual dev spend

- Standalone source: AI_COST_ANALYSIS.md
- Standalone PDF: submission/AI_COST_ANALYSIS.pdf

Assumptions

- 6 commands/session
- 8 sessions/user/month
- AI commands: direct execution (no LLM token cost for MVP)
- Blended Firebase pricing: Blaze pay-as-you-go

Production Cost Projections

Scale	Commands/Month	Firestore	RTDB	Functions	Hosting	Total
100 users	4,800	$3	$2	$0.10	$0	~$5
1,000 users	48,000	$25	$15	$1	$0	~$41
10,000 users	480,000	$250	$150	$10	$0	~$410
100,000 users	4.8M	$2,500	$1,500	$100	$0	~$4,100

Note: No LLM token costs — AI uses pattern matching and direct Firestore writes.

Actual Dev Spend (Week 1)

- **Provider**: Firebase (Google Cloud)
- **Services used**:
    - Firestore: 10K reads, 2K writes, 5K deletes
    - Realtime Database: 5K connections, 50GB storage
    - Cloud Functions: 1K invocations
    - Hosting: 10GB served
- **Total dev spend**: ~$0 (all within free tier)
- **Total API calls**: ~1,000 (mostly during development testing)
- **AI tokens used**: 0 (direct execution, no LLM calls)

Cost Optimization Notes

- AI command processing uses deterministic pattern matching instead of LLM
- This eliminates token costs and reduces latency to <500ms
- Future LLM integration would add ~$0.01/command at current rates

7) Documentation Submission Format

- This package is provided in Markdown and PDF.
- PDF file: SUBMISSION_PACKAGE.pdf
- Submission PDFs are in mvp-1-collab-board/submission/.
- Source requirements PDF is kept at mvp-1-collab-board/G4 Week 1 - CollabBoard-requirements.pdf.
- Included PDFs:
    - PRESEARCH.pdf
    - PRD.pdf

    - MVP.pdf
    - DECISIONS.pdf
    - TASKS.pdf
    - SUBMISSION_PACKAGE.pdf
    - AI_DEVELOPMENT_LOG.pdf
    - AI_COST_ANALYSIS.pdf
    - TEST_EVIDENCE.pdf

8) GitHub PAT Token Note

- If repository automation cannot use existing auth, use a PAT with least privilege for repo read/write.
- Store PAT only in secure secret managers or local environment variables.
- Never commit tokens to source control.