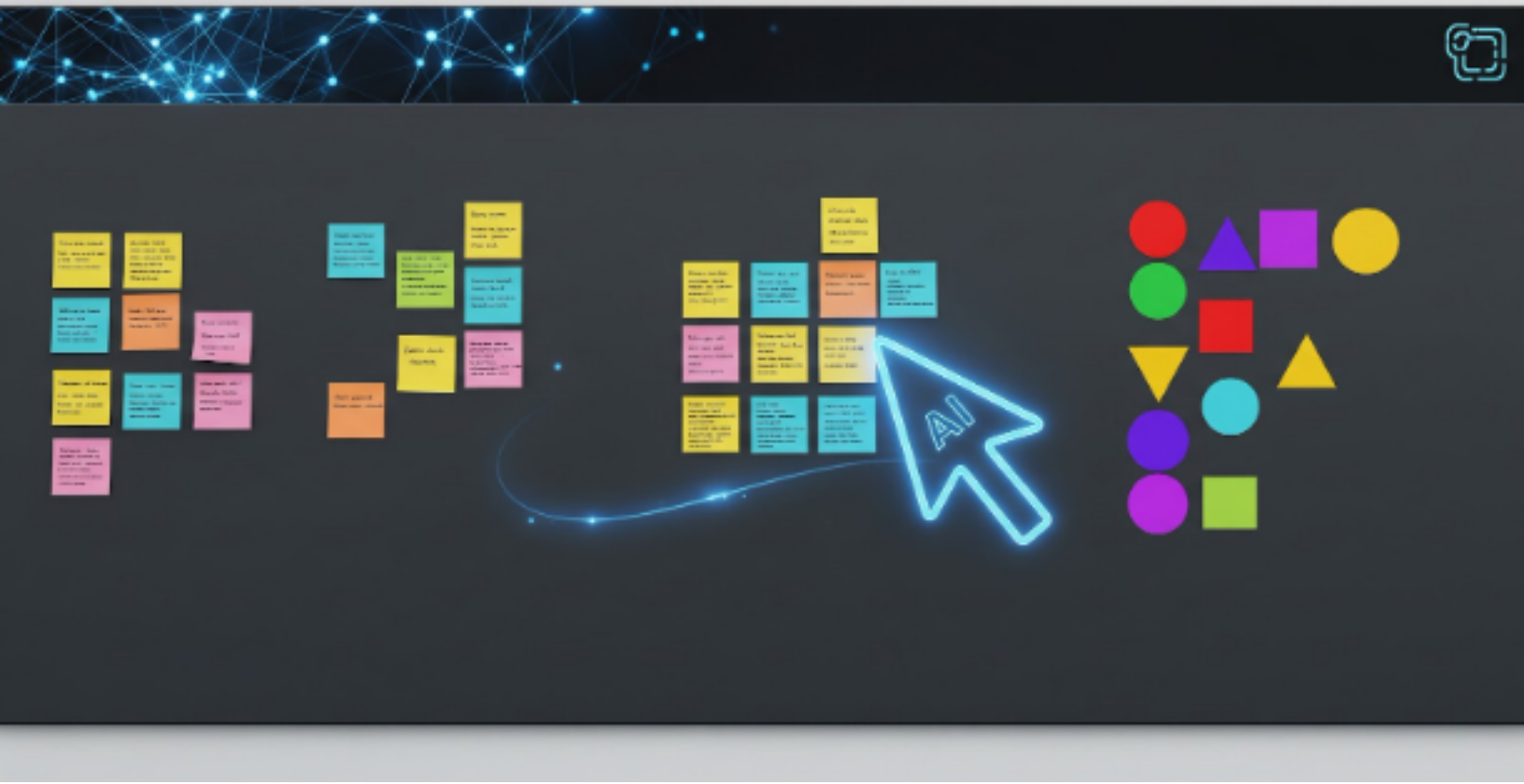

CollabBoard

Building Real-Time Collaborative Whiteboard Tools with AI-First Development

---

Before You Start: Pre-Search (30 Minutes)

Before writing any code, complete the Pre-Search methodology at the end of this document. This structured process uses AI to explore stack options, surface tradeoffs, and document architecture decisions. Your Pre-Search output becomes part of your final submission.

This week emphasizes AI-first development workflows. Pre-Search is the first step in that methodology.

Background

Miro solved hard problems: real-time synchronization, conflict resolution, and smooth performance while streaming data across networks. Multiple users brainstorm, map ideas, and run workshops simultaneously without merge conflicts.

This project requires you to build production-scale collaborative whiteboard infrastructure, then extend it with an AI agent that manipulates the board through natural language. The focus is on AI-first development methodology—using coding agents, MCPs, and structured AI workflows throughout the build process.

Gate: Project completion is required for Austin admission.

Project Overview

One-week sprint with three deadlines:

Checkpoint	Deadline	Focus
MVP	Tuesday (24 hours)	Collaborative infrastructure
Early Submission	Friday (4 days)	Full feature set
Final	Sunday (7 days)	Polish, documentation, deployment

MVP Requirements (24 Hours)

Hard gate. All items required to pass:

- ☐ Infinite board with pan/zoom
- ☐ Sticky notes with editable text
- ☐ At least one shape type (rectangle, circle, or line)
- ☐ Create, move, and edit objects
- ☐ Real-time sync between 2+ users
- ☐ Multiplayer cursors with name labels
- ☐ Presence awareness (who's online)
- ☐ User authentication
- ☐ Deployed and publicly accessible

A simple whiteboard with bulletproof multiplayer beats a feature-rich board with broken sync.

Core Collaborative Whiteboard

Board Features

Feature	Requirements
Workspace	Infinite board with smooth pan/zoom
Sticky Notes	Create, edit text, change colors
Shapes	Rectangles, circles, lines with solid colors
Connectors	Lines/arrows connecting objects
Text	Standalone text elements
Frames	Group and organize content areas
Transforms	Move, resize, rotate objects

Selection	Single and multi-select (shift-click, drag-to-select)
Operations	Delete, duplicate, copy/paste

Real-Time Collaboration

Feature	Requirements
Cursors	Multiplayer cursors with names, real-time movement
Sync	Object creation/modification appears instantly for all users
Presence	Clear indication of who's currently on the board
Conflicts	Handle simultaneous edits (last-write-wins acceptable, document your approach)
Resilience	Graceful disconnect/reconnect handling
Persistence	Board state survives all users leaving and returning

Testing Scenarios

We will test:

1. 2 users editing simultaneously in different browsers
2. One user refreshing mid-edit (state persistence check)
3. Rapid creation and movement of sticky notes and shapes (sync performance)
4. Network throttling and disconnection recovery
5. 5+ concurrent users without degradation

Performance Targets

Metric	Target
Frame rate	60 FPS during pan, zoom, object manipulation
Object sync latency	<100ms
Cursor sync latency	<50ms
Object capacity	500+ objects without performance drops
Concurrent users	5+ without degradation

AI Board Agent

Required Capabilities

Your AI agent must support at least 6 distinct commands across these categories:

Creation Commands

- "Add a yellow sticky note that says 'User Research'"
- "Create a blue rectangle at position 100, 200"
- "Add a frame called 'Sprint Planning'"

Manipulation Commands

- "Move all the pink sticky notes to the right side"
- "Resize the frame to fit its contents"
- "Change the sticky note color to green"

Layout Commands

- "Arrange these sticky notes in a grid"
- "Create a 2x3 grid of sticky notes for pros and cons"
- "Space these elements evenly"

Complex Commands

- "Create a SWOT analysis template with four quadrants"
- "Build a user journey map with 5 stages"
- "Set up a retrospective board with What Went Well, What Didn't, and Action Items columns"

Tool Schema (Minimum)

```
createStickyNote(text, x, y, color)
createShape(type, x, y, width, height, color)
createFrame(title, x, y, width, height)
createConnector(fromId, toId, style)
moveObject(objectId, x, y)
resizeObject(objectId, width, height)
updateText(objectId, newText)
changeColor(objectId, color)
getBoardState() // returns current board objects for context
```

Evaluation Criteria

Command	Expected Result
"Create a SWOT analysis"	4 labeled quadrants (Strengths, Weaknesses, Opportunities, Threats)

"Arrange in a grid"	Elements aligned with consistent spacing
Multi-step commands	AI plans steps and executes sequentially

Shared AI State

- All users see AI-generated results in real-time
- Multiple users can issue AI commands simultaneously without conflict

AI Agent Performance

Metric	Target
Response latency	<2 seconds for single-step commands
Command breadth	6+ command types
Complexity	Multi-step operation execution
Reliability	Consistent, accurate execution

AI-First Development Requirements

This week emphasizes learning AI-first development workflows. You must document your process.

Required Tools

Use at least two of:

- Claude Code
- Cursor
- Codex
- MCP integrations

AI Development Log (Required)

Submit a 1-page document covering:

Section	Content
Tools & Workflow	Which AI coding tools you used, how you integrated them
MCP Usage	Which MCPs you used (if any), what they enabled
Effective Prompts	3-5 prompts that worked well (include the actual prompts)
Code Analysis	Rough % of AI-generated vs hand-written code
Strengths & Limitations	Where AI excelled, where it struggled
Key Learnings	Insights about working with coding agents

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
$___/month	$___/month	$___/month	$___/month

Include assumptions: average AI commands per user per session, average sessions per user per month, token counts per command type.

Technical Stack

Recommended Path

Layer	Technology
Backend	Firebase (Firestore, Realtime DB, Auth) or Supabase
Frontend	React/Vue/Svelte with Konva.js, Fabric.js, PixiJS, or HTML5 Canvas
AI Integration	OpenAI GPT-4 or Anthropic Claude with function calling
Deployment	Vercel, Firebase Hosting, or Render

Alternative Options

Layer	Alternatives
Backend	AWS (DynamoDB, Lambda, WebSockets), custom WebSocket server
Frontend	Vanilla JS, any framework with canvas support

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

Build Strategy

Priority Order

6. Cursor sync — Get two cursors moving across browsers
7. Object sync — Create sticky notes that appear for all users
8. Conflict handling — Handle simultaneous edits
9. State persistence — Survive refreshes and reconnects
10. Board features — Shapes, frames, connectors, transforms
11. AI commands (basic) — Single-step creation/manipulation
12. AI commands (complex) — Multi-step template generation

Critical Guidance

- Multiplayer sync is the hardest part. Start here.
- Build vertically: finish one layer before starting the next
- Test with multiple browser windows continuously
- Throttle network speed during testing
- Test simultaneous AI commands from multiple users

Submission Requirements

Deadline: Sunday 10:59 PM CT

Deliverable	Requirements
GitHub Repository	Setup guide, architecture overview, deployed link
Demo Video (3-5 min)	Real-time collaboration, AI commands, architecture explanation
Pre-Search Document	Completed checklist from Phase 1-3
AI Development Log	1-page breakdown using template above
AI Cost Analysis	Dev spend + projections for 100/1K/10K/100K users
Deployed Application	Publicly accessible, supports 5+ users with auth
Social Post	Share on X or LinkedIn: description, features, demo/screenshots, tag @GauntletAI

Final Note

A simple, solid, multiplayer whiteboard with a working AI agent beats any feature-rich board with broken collaboration.

Project completion is required for Austin admission.

Appendix: Pre-Search Checklist

Complete this before writing code. Save your AI conversation as a reference document.

Phase 1: Define Your Constraints

1. Scale & Load Profile

- Users at launch? In 6 months?
- Traffic pattern: steady, spiky, or unpredictable?
- Real-time requirements (websockets, live updates)?
- Cold start tolerance?

2. Budget & Cost Ceiling

- Monthly spend limit?
- Pay-per-use acceptable or need fixed costs?
- Where will you trade money for time?

3. Time to Ship

- MVP timeline?
- Speed-to-market vs. long-term maintainability priority?
- Iteration cadence after launch?

4. Compliance & Regulatory Needs

- Health data (HIPAA)?
- EU users (GDPR)?
- Enterprise clients (SOC 2)?
- Data residency requirements?

5. Team & Skill Constraints

- Solo or team?
- Languages/frameworks you know well?
- Learning appetite vs. shipping speed preference?

Phase 2: Architecture Discovery

6. Hosting & Deployment

- Serverless vs. containers vs. edge vs. VPS?
- CI/CD requirements?
- Scaling characteristics?

7. Authentication & Authorization

- Auth approach: social login, magic links, email/password, SSO?

- RBAC needed?
- Multi-tenancy considerations?

8. Database & Data Layer

- Database type: relational, document, key-value, graph?
- Real-time sync, full-text search, vector storage, caching needs?
- Read/write ratio?

9. Backend/API Architecture

- Monolith or microservices?
- REST vs. GraphQL vs. tRPC vs. gRPC?
- Background job and queue requirements?

10. Frontend Framework & Rendering

- SEO requirements (SSR/static)?
- Offline support/PWA?
- SPA vs. SSR vs. static vs. hybrid?

11. Third-Party Integrations

- External services needed (payments, email, analytics, AI APIs)?
- Pricing cliffs and rate limits?
- Vendor lock-in risk?

Phase 3: Post-Stack Refinement

12. Security Vulnerabilities

- Known pitfalls for your stack?
- Common misconfigurations?
- Dependency risks?

13. File Structure & Project Organization

- Standard folder structure for your framework?
- Monorepo vs. polyrepo?
- Feature/module organization?

14. Naming Conventions & Code Style

- Naming patterns for your language/framework?
- Linter and formatter configs?

15. Testing Strategy

- Unit, integration, e2e tools?
- Coverage target for MVP?

- Mocking patterns?

16. Recommended Tooling & DX

- VS Code extensions?
- CLI tools?
- Debugging setup?