CollabCanvas

Building Real-Time Collaborative Design Tools with AI

Background

Figma revolutionized design by making collaboration seamless. Multiple designers could work together in real time, seeing each other's cursors and making edits simultaneously without merge conflicts.

This required solving complex technical challenges: real-time synchronization, conflict resolution, and 60 FPS performance while streaming data across the network.

Now imagine adding AI to this. What if you could tell an AI agent "create a login form" and watch it build the components on your canvas? Or say "arrange these elements in a grid" and see it happen automatically?

This project challenges you to build both the collaborative infrastructure and an AI agent that can manipulate the canvas through natural language.

Why This Matters

The future of design tools isn't just collaborative — it's co-creative. You'll be building the foundation for how humans and AI can design together, in real time.

Project Overview

This is a one-week sprint with three key deadlines:

- MVP: Tuesday (24 hours)
- Early Submission: Friday (4 days)

- Final: Sunday (7 days)

You'll build in two phases: first the core collaborative canvas with real-time sync, then an AI agent that manipulates the canvas through natural language.

MVP Requirements (24 Hours)

This is a hard gate. To pass the MVP checkpoint, you must have:

- ☐ Basic canvas with pan/zoom
- ☐ At least one shape type (rectangle, circle, or text)
- ☐ Ability to create and move objects
- ☐ Real-time sync between 2+ users
- ☐ Multiplayer cursors with name labels
- ☐ Presence awareness (who's online)
- ☐ User authentication (users have accounts/names)
- ☐ Deployed and publicly accessible

The focus is on collaborative infrastructure.

The MVP isn't about features — it's about proving your foundation is solid. A simple canvas with bulletproof multiplayer is worth more than a feature-rich canvas with broken sync.

Example Architecture

At minimum, you should have:

1. A backend (Firestore, Supabase, or custom WebSocket server) that broadcasts updates.
2. A front-end listener that updates local canvas state and rebroadcasts deltas.
3. A persistence layer that saves the current state on disconnects.

Core Collaborative Canvas

Canvas Features

Your canvas needs a large workspace with a smooth pan and zoom. It doesn't need to be truly infinite, but should feel spacious. Support basic shapes — rectangles, circles, and lines with solid colors. Add text layers with basic formatting.

Users should be able to transform objects (move, resize, rotate). Include selection for single and multiple objects (shift-click or drag-to-select). Add layer management and basic operations like delete and duplicate.

Real-Time Collaboration

Every user should see multiplayer cursors with names moving in real time. When someone creates or modifies an object, it appears instantly for everyone. Show clear presence awareness of who's currently editing.

Handle conflict resolution when multiple users edit simultaneously. (A "last write wins" approach is acceptable, but document your choice.)

Manage disconnects and reconnects without breaking the experience. Canvas state must persist — if all users leave and come back, their work should still be there.

Testing Scenario

We'll test with:

1. 2 users editing simultaneously in different browsers.
2. One user refreshing mid-edit to confirm state persistence.
3. Multiple shapes being created and moved rapidly to test sync performance.

Performance Targets

- Maintain 60 FPS during all interactions (pan, zoom, object manipulation).
- Sync object changes across users in <100ms and cursor positions in <50ms.
- Support 500+ simple objects without FPS drops and 5+ concurrent users without degradation.

We'll test performance on your deployed app, so make sure it works under load.

AI Canvas Agent

The AI Feature

Build an AI agent that manipulates your canvas through natural language using function calling.

When a user types "Create a blue rectangle in the center," the AI agent calls your canvas API functions, and the rectangle appears on everyone's canvas via real-time sync.

Required Capabilities

Your AI agent must support at least 6 distinct commands showing a range of creation, manipulation, and layout actions.

Creation Commands:

- “Create a red circle at position 100, 200”
- “Add a text layer that says ‘Hello World’”
- “Make a 200x300 rectangle”

Manipulation Commands:

- “Move the blue rectangle to the center”
- “Resize the circle to be twice as big”
- “Rotate the text 45 degrees”

Layout Commands:

- “Arrange these shapes in a horizontal row”
- “Create a grid of 3x3 squares”
- “Space these elements evenly”

Complex Commands:

- “Create a login form with username and password fields”
- “Build a navigation bar with 4 menu items”
- “Make a card layout with title, image, and description”

Example Evaluation Criteria

When you say:

“Create a login form,” …We expect the AI to create at least three inputs (username, password, submit), arranged neatly, not just a text box.

Technical Implementation

Define a tool schema that your AI can call, such as:

```typescript
createShape(type, x, y, width, height, color)
moveShape(shapeId, x, y)
resizeShape(shapeId, width, height)
rotateShape(shapeId, degrees)
createText(text, x, y, fontSize, color)
```

```
getCanvasState() // returns current canvas objects for context
```

We recommend OpenAI's function calling or LangChain tools for interpretation.
For complex operations (e.g. "create a login form"), your AI should plan steps upfront (create fields, align, group) and execute sequentially.

Shared AI State

All users must see the same AI-generated results. If one user asks the AI to create something, everyone should see it. Multiple users should be able to use the AI simultaneously without conflict.

AI Agent Performance Targets

- Latency: Responses under 2 seconds for single-step commands.
- Breadth: Handles 6+ command types.
- Complexity: Executes multi-step operations.
- Reliability: Consistent and accurate execution.
- UX: Natural interaction with visible, immediate feedback.

AI Development Log

You must submit a 1-page breakdown documenting your AI-first development process.

Include:

1. Tools & Workflow: What AI coding tools you used and how you integrated them.
2. Prompting Strategies: 3–5 effective prompts that worked well.
3. Code Analysis: Rough percentage of AI-generated vs hand-written code.
4. Strengths & Limitations: Where AI excelled and where it struggled.
5. Key Learnings: Insights about working with AI coding agents.

Technical Stack

Recommended (not required):

- Backend: Firebase (Firestore, Realtime DB, Auth), or AWS (DynamoDB, Lambda, WebSockets)

- Frontend: React, Vue, Svelte, or Vanilla JS with Konva.js, Fabric.js, PixiJS, or HTML5 Canvas
- AI Integration: OpenAI GPT-4 or Anthropic Claude (function calling support)

Use whatever stack helps you ship the best product.

Build Strategy

Start with the Hard Part

Multiplayer sync is the hardest and most important part.
Get two cursors syncing → objects syncing → handle conflicts → persist state.
Only after this is solid should you add shapes, transformations, and AI.

Build Vertically

Finish one layer at a time:

1. Cursor sync
2. Object sync
3. Transformations
4. Basic AI commands
5. Complex AI commands

Avoid half-finished features. Multiplayer last = failure.

Test Continuously

Use multiple browser windows, throttle network speed, test with 3–4 users, and run simultaneous AI commands. We'll test under real conditions — test like we will.

Submission Requirements

Submit the following by **Sunday 10:59 PM CT**:

1. **GitHub Repository** – with setup guide, architecture overview, and deployed link.
2. **Demo Video (3–5 minutes)** – show real-time collaboration, AI commands, and explain your architecture.
3. **AI Development Log (1 page)** – using the provided template.
4. **Deployed Application** – publicly accessible, supporting 5+ users with authentication.

We recommend deploying on **Vercel**, **Firebase Hosting**, or **Render** for simplicity.

Final Note

This project mirrors what the best AI startups are doing right now — combining real-time collaboration with intelligent AI creation.

The closer you get to that experience, the more you'll understand what it takes to build the next Figma.

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

Let's build something real.