CollabCanvas Demo Video - Talking Points

Target Audience: Figma Users, UI/UX Professionals, Design Teams
Duration: 10-15 minutes
Version: 1.0
Date: October 2025

🎯 Opening Hook (30 seconds)

Key Message

"Real-time collaborative design canvas that brings the power of Figma's multiplayer experience to a lightweight, AI-powered whiteboard."

Talking Points

- Show multiple cursors moving simultaneously
- Highlight instant shape updates across all users
- Mention: "Built for design teams who need quick visual collaboration without the complexity of full design tools"
- Hook: "What if you could combine the simplicity of a whiteboard with AI assistance and Figma-style real-time collaboration?"

👥 Target Audience & Use Cases (1-2 minutes)

Primary Audience

UI/UX Professionals & Design Teams who:

- Are familiar with Figma's multiplayer features
- Need quick brainstorming and wireframing tools
- Want lightweight alternatives for early-stage design
- Value real-time collaboration without tool complexity
- Are exploring AI-assisted design workflows

Use Cases

1. **Early-Stage Wireframing**: Rapid layout exploration before moving to Figma/Sketch
2. **Team Brainstorming**: Live design discussions with remote teammates
3. **Client Presentations**: Interactive mockup walkthroughs
4. **Design Sprints**: Quick iteration on concepts with AI assistance
5. **Workshop Facilitation**: Collaborative drawing sessions
6. **Documentation**: Visual system architecture diagrams

What Makes It Unique for Designers

- **Familiar Figma-like interface** with multiplayer cursors and selection awareness
- **AI-powered shape creation** - describe layouts instead of drawing them
- **Zero learning curve** - if you know Figma, you know this
- **No installation required** - runs entirely in browser
- **Group-based workspaces** - each team gets isolated canvases

---

🏗️ Architecture Overview (2-3 minutes)

High-Level System Design

```
┌─────────────────────────────────────────────────────────────┐
│              CLIENT (React + TypeScript)                     │
│  • 60 FPS Canvas Rendering (Konva.js)                       │
│  • Optimistic UI Updates                                    │
│  • AI Agent (GPT-4o + LangChain)                            │
└─────────────────────────────────────────────────────────────┘
                            ▼ ▼ ▼
┌─────────────────────────────────────────────────────────────┐
│              FIREBASE BACKEND (Serverless)                   │
│                                                          │
│  ┌──────────────┐  ┌──────────────┐  ┌──────────────┐    │
│  │   Firebase   │  │  Firestore   │  │   Realtime   │    │
│  │     Auth     │  │              │  │   Database   │    │
│  │  • Sessions  │  │  • Shapes    │  │  • Cursors   │    │
│  │  • Users     │  │  • Groups    │  │  • Presence  │    │
│  └──────────────┘  └──────────────┘  └──────────────┘    │
└─────────────────────────────────────────────────────────────┘
```

Talking Points

- **"Dual-database architecture"** - Firestore for persistent data, Realtime Database for ephemeral
- **"Figma's multiplayer DNA"** - Same approach to real-time collaboration
- **"Optimistic updates"** - See changes instantly, sync happens in background
- **"Client-side AI"** - No server required, your OpenAI key stays in your browser

Technical Stack (For Technical Audience)

- **Frontend**: React 18 + TypeScript + Vite
- **Canvas**: Konva.js (same rendering engine family as Figma's canvas)
- **AI**: OpenAI GPT-4o + LangChain (19 specialized tools)
- **Backend**: Firebase (Auth, Firestore, Realtime Database, Hosting)
- **Styling**: Tailwind CSS (utility-first, like Figma's design system approach)

✨ Core Features Walkthrough (5-7 minutes)

1. Real-Time Collaboration (Like Figma)

Demo Steps

1. Show **multiple browser windows** with different user names
2. Move a shape in one window → **instant update** in all others (< 50ms)
3. Show **multiplayer cursors** with name labels moving in real-time
4. Select a shape → **selection glow** appears in other users' views

Talking Points

- "See exactly who's editing what, just like in Figma"
- "Selection awareness with colored glow in each user's color"
- "Pastel color-coded users for clear visual distinction"
- "< 50ms object sync, < 30ms cursor sync - feels instant"

- "Last-Write-Wins conflict resolution with timestamps"

Architecture Note

- **Firestore** handles shape persistence (< 100ms sync)
- **Realtime Database** handles cursors/presence (< 50ms sync)
- **Optimistic UI** updates local state first (0ms perceived latency)
- **Throttled writes** (50ms) prevent Firebase rate limiting

---

2. Advanced Shape Manipulation

Demo Steps: Two-Step Drawing

1. **Rectangle**: Click corner → drag to preview → click to finalize
2. **Line**: Click first anchor → move to preview → click second anchor
3. **Circle**: Click center → move to set radius → click to finalize
4. Show **real-time preview** during drawing

Demo Steps: Intelligent Resize Handles

1. **Line**: Drag either anchor point to change direction/length
2. **Rectangle**: Drag any corner handle → opposite corner stays fixed
3. **Circle**: Drag edge handles (N/E/S/W) to grow/shrink radius
4. **Middle Mouse Pan**: Show panning still works while editing

Talking Points

- "Two-step drawing with live preview - same as Figma's pen tool workflow"
- "Smart resize handles that behave how designers expect"
- "Only selected shapes are draggable - prevents accidental moves"
- "Handles hide during drag for cleaner visual experience"
- "Middle mouse button panning works even when shapes selected"

Architecture Note

- **LineAnchors component**: Calculates opposite anchor constraint
- **ResizeHandles component**: Four corners with inverse resize logic
- **CircleResizeHandles**: Group positioning with distance-based radius calc
- **Shape-specific rendering**: Conditional handle display based on type

---

3. Color & Styling System (Designer-Friendly)

Demo Steps

1. Select shape → show **dual color pickers** (border + fill)
2. Change border color → picker shows as **outline square**
3. Change fill color → picker shows as **solid square**
4. Text shape → show **single color picker** (contextual)

Talking Points

- "Contextual color controls - only see options relevant to selected shape"
- "Dual pickers for shapes with both stroke and fill (like Figma properties)"
- "Visual indicator: outline picker for borders, solid picker for fills"
- "Professional defaults: transparent fills, black borders"
- "Chrome Color Picker integration for familiar UX"

Architecture Note

- **ColorPicker component** with `outline` prop for visual mode
- **Conditional rendering** based on `shape.type`
- **Color parsing** supports hex and rgba formats
- **Smart defaults** in shape creation functions

---

4. 🤖 AI-Powered Shape Creation (Unique Feature)

Demo Steps: Basic Commands

1. "Create a red circle at 300, 200"
2. "Make a 3x3 grid of blue squares"
3. "Move the blue rectangle 200 pixels left"
4. "Find the text that says 'Login' and center it in the rectangle"
5. "Bring the red circle to the front"

Demo Steps: Complex Layouts

1. "Create a login form with fields for email and password"
2. "Arrange these circles horizontally with 50px spacing"
3. "Find a blank space and create a circle there"

Talking Points

- "Natural language interface - no need to learn commands"
- "19 specialized tools: creation, movement, transformation, layout, layers"
- "Understands spatial relationships - 'move left', 'center in', 'find blank space'"
- "Context-aware - 'move them left' references previous selection"
- "GPT-4o model with LangChain orchestration"
- "Your API key never leaves your browser - client-side architecture"

Architecture Deep Dive (Technical)

```
User Message → AIChat Component → CanvasAIAgent
                                        ↓
                               LangChain AgentExecutor
                                        ↓
                            GPT-4o Analyzes Intent
                                        ↓
                         Selects Tool(s) from 19 options
                                        ↓
                    Zod Schema Validation of Parameters
                                        ↓
                    Execute Tool Function(s)
                                        ↓
                Canvas Mutations (addShape, updateShape, etc.)
                                        ↓
                Firebase Real-time Sync to All Users
```

Tool Categories (19 Total):

- **Creation (6)**: create_circle, create_rectangle, create_text, create_line, create_grid, create_multiple_circles
- **Movement (3)**: move_shape (absolute), move_shape_relative (dx/dy), move_multiple_shapes (batch)
- **Transformation (2)**: resize_shape, rotate_shape

- **Layout (2)**: arrange_horizontal, align_text_to_shape
- **Layers (4)**: bring_to_front, bring_forward, send_backward, send_to_back
- **Utility (2)**: get_canvas_info, find_blank_space

System Prompt Engineering

- **Center-based coordinates**: All shapes positioned by center (like Figma frames)
- **Batch operation requirements**: 5+ operations must use batch tools
- **Relative vs absolute movement**: "move 100 left" vs "move to 100, 200"
- **Conversation context**: Last 10 messages for pronoun resolution
- **Safety rules**: Only manipulate recently created shapes (< 30s old)

Shape Matching Algorithm

1. **Text content matching** (highest priority): "text that says 'Login'"
2. **Type + color matching**: "red circle"
3. **Type-only matching**: "rectangle"
4. **Color-only matching**: "blue"
5. **Recency sorting**: Prefer recently created shapes

---

5. Group-Based Workspaces

Demo Steps

1. Show registration with **group name input**
2. Create "Design Team" group
3. Open second browser as different user → enter "Design Team"
4. Show **isolated canvas** - only team members see it
5. Create "Marketing Team" group in third browser → **completely separate canvas**

Talking Points

- "Like Figma teams - each group gets isolated workspace"
- "No accidental interference between different teams"
- "Share group name with teammates to invite them"
- "Group name displayed in toolbar for context"
- "Normalized group IDs (e.g., 'My Team' → 'my-team')"

Architecture Note

- **Firestore structure**: `groups/{groupId}/canvases/main-canvas/shapes/`
- **Security rules**: Users can only read/write shapes in their group
- **User document**: Stores `groupId` and `groupName` fields
- **Realtime Database**: Group-scoped cursors and presence paths

---

6. Viewport Management & Canvas Navigation

Demo Steps

1. Show **pan** by dragging empty space
2. Show **middle mouse button pan** while shape selected
3. **Zoom** with mouse wheel (10% - 500%)
4. Show **boundary clamping** - can't pan beyond grid edges
5. Press **Spacebar** to reset to default centered view
6. Show **minimap** during pan/zoom with cursor coordinates

Talking Points

- "5000x5000 virtual canvas - plenty of space for large projects"

- "Boundary clamping keeps work area accessible"
- "Default centered at 50% zoom for overview"
- "Middle mouse button for dedicated pan control"
- "Minimap shows cursor position in real-time"
- "Smooth 60 FPS rendering during all interactions"

Architecture Note

- **Viewport state**: `{ x, y, scale }` managed in `useCanvasViewport` hook
- **Clamping utility**: `clampViewportPosition()` in `canvasHelpers.ts`
- **Full-height canvas**: `window.innerHeight - 104px` (toolbar + footer)
- **Dynamic resize**: `useEffect` listener updates on window resize

---

7. Selection & Multi-Object Management

Demo Steps

1. **Single select**: Click any shape
2. **Multi-select**: Ctrl/Cmd+Click to add to selection
3. **Select mode**: Press V → drag box selection
4. **Duplicate**: Select shape → press Ctrl/Cmd+D
5. **Delete**: Press Delete or Backspace
6. **Precise positioning**: Show X/Y input fields in toolbar

Talking Points

- "Multi-select like Figma - Ctrl/Cmd+Click to add"
- "Select mode (V key) for box selection without dragging shapes"
- "Duplicate and delete with familiar keyboard shortcuts"
- "Precise positioning with X/Y coordinate inputs"
- "Arrow keys for 1px nudging, Shift+Arrow for 10px"

Architecture Note

- **Selection state**: `selectedShapeIds: string[]` array in App.tsx
- **Multi-select logic**: `useCanvasInteraction` hook handles Ctrl/Cmd modifier
- **Box selection**: Implemented in select mode with drag area calculation
- **Keyboard handlers**: `useKeyboardShortcuts` hook for all shortcuts

---

8. Undo/Redo System

Demo Steps

1. Create several shapes
2. Press **Ctrl/Cmd+Z** to undo
3. Press **Ctrl/Cmd+Y** to redo
4. Show undo/redo buttons in toolbar (enabled/disabled states)
5. Open second browser → perform action → show undo syncs

Talking Points

- "Full history tracking - up to 50 states"
- "Undo/redo works across all operations: create, move, delete, resize"
- "Syncs via Firestore - all users see consistent state"
- "Keyboard shortcuts match Figma: Ctrl/Cmd+Z, Ctrl/Cmd+Y"
- "Visual indicators: buttons disable when no history"

Architecture Note

- **useUndo hook**: Maintains `history` array of shape snapshots
- **Firestore integration**: Uses same `useShapes` hook for sync
- **Circular reference prevention**: `isRestoringHistory` flag
- **Memory limit**: 50 states to prevent memory issues

---

9. Performance Optimizations (For Technical Audience)

Demo Steps

1. Create 500+ shapes → show **60 FPS maintained**
2. Drag shape rapidly → show **smooth movement**
3. Open DevTools → show **throttled Firebase writes**
4. Multiple users editing → show **no conflicts**

Talking Points

- "Optimistic UI: See changes instantly, sync happens async"
- "Throttled writes: 50ms batching with 200ms final flush"
- "Component memoization: React.memo on performance-critical components"
- "Cursor throttling: 75ms updates = 33% less network traffic"
- "Last-Write-Wins conflict resolution with timestamps"
- "Smart caching: Color calculations memoized in Map"
- "60 FPS maintained with 500+ shapes and 5+ concurrent users"

Performance Metrics Table

Metric	Target	Achieved	Optimization
Frame Rate	60 FPS	✅ 60 FPS	React.memo + throttling
Object Sync	< 100ms	✅ ~50ms	Optimistic updates
Cursor Sync	< 50ms	✅ ~30ms	75ms throttling
User Capacity	5+	✅ 5+ concurrent	Efficient rendering
Shape Capacity	500+	✅ 500+ shapes	Layer separation
Conflict Resolution	N/A	✅ < 10ms	Timestamp comparison

Architecture Note

- **Throttling layer**: Closures maintain pending updates Map
- **Optimistic state**: `localShapeUpdates` Map merges with Firestore
- **Memoization**: Applied to ShapeRenderer, CursorLayer, CanvasGrid, UsersList, Toolbar, Footer
- **Color cache**: Module-level Map in `canvasHelpers.ts`
- **Layer separation**: Background, shapes, cursors on separate Konva layers

---

🎨 UI/UX Design Philosophy (1 minute)

Design Principles

- **Familiar**: Figma-inspired interface for zero learning curve
- **Minimal**: Only show controls relevant to current selection

- **Consistent**: All buttons standardized at 40px height
- **Professional**: Gradient toolbar, pastel colors, clean typography
- **Accessible**: 40px touch targets meet WCAG guidelines

Visual Design Highlights

1. **Gradient Toolbar**: White to grey gradient with 2px grey bottom border
2. **Symmetrical Footer**: Matching 2px grey top border creates balanced frame
3. **Pastel User Colors**: 12 carefully chosen colors for user indicators
4. **Contextual Color Pickers**: Outline for borders, solid for fills
5. **Icon-First Interface**: Large, clear Lucide icons with hover tooltips
6. **Graph Paper Grid**: Visual coordinate reference like design tools
7. **Comprehensive Help Modal**: 6 organized sections with kbd styling

Talking Points

- "Design language inspired by professional design tools"
- "Contextual UI - only see what's relevant to your selection"
- "Consistent 40px buttons throughout for visual harmony"
- "Pastel colors reduce eye strain during long sessions"
- "Help modal with step-by-step instructions for onboarding"

🔧 Architecture Deep Dive (3-4 minutes)

Frontend Architecture

```
App.tsx (Root Component)
├── Auth.tsx (when not authenticated)
└── Authenticated View
    ├── Toolbar
    │   ├── Select Mode Toggle (V key)
    │   ├── Shape Buttons (Rectangle, Circle, Line, Text)
    │   ├── Undo/Redo Buttons
    │   ├── Color Pickers (contextual: single/dual)
    │   ├── Font Size Controls (text only)
    │   ├── Position Controls (X/Y inputs)
    │   ├── Layer Controls (4 buttons)
    │   ├── Duplicate/Delete Buttons
    │   ├── UsersList (online indicators)
    │   └── Logout Button
    │
    ├── Canvas
    │   ├── CanvasGrid (graph paper background)
    │   ├── Shapes Layer
    │   │   ├── ShapeRenderer (React.memo)
    │   │   ├── LineAnchors (draggable endpoints)
    │   │   ├── ResizeHandles (rectangle corners)
    │   │   ├── CircleResizeHandles (edge handles)
    │   │   └── TextEditor (inline editing)
    │   ├── CursorLayer (remote cursors, React.memo)
    │   └── Minimap (during pan/zoom)
    │
    └── Footer
        ├── "Made with ❤️" message
        ├── AI Assistant Button
        └── Help Button (opens modal)
```

Custom Hooks Architecture

```
// Authentication & Groups
useAuth()               // Firebase auth state, user object
useGroupAuth()          // Group membership, validation

// Canvas State
useShapes()             // Firestore shapes CRUD operations
useOptimisticShapes()   // Local updates + merge logic
useInterpolatedShapes() // Smooth remote updates

// Interaction
useCanvasInteraction() // Drag, select, multi-select
useCanvasViewport()    // Pan, zoom, boundary clamping
useTextEditing()       // Inline text edit mode
useTransformHandlers() // Resize handle logic

// Multiplayer
useCursors()           // Realtime DB cursor positions
usePresence()          // Online/offline status
useSelections()        // Selection awareness

// Operations
useShapeOperations()   // Create, update, delete shapes
useShapePlacement()    // Two-step drawing with preview
useStyleHandlers()     // Color, font size changes

// Navigation & Shortcuts
useKeyboardShortcuts() // All keyboard shortcuts
useEdgePanning()       // Auto-pan at canvas edges

// History
useUndo()              // Undo/redo state management
useShapeHistory()      // Shape change tracking
```

Database Schema

Firestore (Persistent Data)

```
// Group metadata
groups/{groupId}
  name: string              // Display name
  normalizedName: string    // URL-safe identifier
  memberCount: number
  createdAt: timestamp
  // Canvas shapes
  canvases/main-canvas/shapes/{shapeId}
    id: string
    type: 'rectangle' | 'circle' | 'line' | 'text'
    x: number                // Center X
    y: number                // Center Y
    width?: number           // Rectangle only
    height?: number          // Rectangle only
    radius?: number          // Circle only
    points?: number[]        // Line only [x1,y1,x2,y2]
    text?: string            // Text only
    fontSize?: number        // Text only
    fill: string             // Hex or rgba()
    stroke?: string          // Hex or rgba()
    strokeWidth?: number
    rotation?: number
    zIndex?: number          // Layer order
    createdBy: string        // User ID
    createdAt: timestamp
    updatedAt: timestamp     // For conflict resolution

// User profiles
users/{userId}
  id: string
  name: string
  email: string
  color: string              // Pastel color assignment
  groupId: string            // Normalized group identifier
  groupName: string          // Display name
  online: boolean
  lastSeen: timestamp
  createdAt: timestamp
```

Realtime Database (Ephemeral Data)

```
// Group-scoped ephemeral data
groups/{groupId}/

  // Cursor positions (high-frequency updates)
  cursors/{userId}
    userId: string
    userName: string
    x: number                // Canvas coordinates
    y: number
    color: string            // User's pastel color
    timestamp: number

  // Presence tracking (online/offline)
  presence/{userId}
    online: boolean
    lastSeen: timestamp      // Server timestamp

  // Selection awareness (who's editing what)
  selections/{userId}
    shapeIds: string[]       // Currently selected shapes
    timestamp: number
```

Data Flow: Shape Creation

```
1. User clicks "Add Rectangle" button
   ↓
2. Two-step drawing begins (useShapePlacement hook)
   ↓
3. First click stores corner position
   ↓
4. Mouse move updates preview state
   ↓
5. Second click finalizes shape
   ↓
6. Generate UUID for shape ID
   ↓
7. Create shape object with defaults
   ↓
8. Optimistic update: Add to localShapeUpdates Map
   ↓
9. Immediate render: User sees shape instantly (0ms latency)
   ↓
10. Background: Write to Firestore (throttled to 50ms)
    ↓
11. Firestore broadcasts to all connected clients
    ↓
12. Remote clients: onSnapshot receives new shape
    ↓
13. Merge logic: Compare timestamps, render shape
    ↓
14. All users see shape (< 50ms total sync time)
```

Data Flow: AI Command Execution

```
1. User types: "Create a red circle at 300, 200"
   ↓
2. AIChat.handleSendMessage()
   ↓
3. Create CanvasAIAgent(apiKey, context)
   ↓
4. agent.execute(message, conversationHistory)
   ↓
5. LangChain AgentExecutor.invoke()
   ↓
6. GPT-4o analyzes intent → Selects create_circle tool
```

```
   ↓
7. Zod schema validates parameters:
   { x: 300, y: 200, radius: 50, color: "red" }
   ↓
8. Tool execution: context.addShape(newShape)
   ↓
9. Same flow as manual creation (optimistic + Firestore)
   ↓
10. Firebase syncs to all users
   ↓
11. Return natural language response to chat
   ↓
12. Display: "Created a red circle at position (300, 200)"
```

Conflict Resolution: Last-Write-Wins

```
Scenario: Two users edit same shape simultaneously

User A (dragging shape):
  1. Optimistic update: localShapeUpdates.set(shapeId, { ...shape, x: 100, updatedAt: 
1000 })
  2. Throttled write to Firestore (50ms delay)
  3. Eventually writes: { x: 100, updatedAt: 1000 }

User B (changing color):
  1. Optimistic update: localShapeUpdates.set(shapeId, { ...shape, fill: 'red', 
updatedAt: 1005 })
  2. Throttled write to Firestore (50ms delay)
  3. Eventually writes: { fill: 'red', updatedAt: 1005 }

Firestore:
  1. Receives both updates
  2. Last write (1005 > 1000) wins
  3. Broadcasts final state: { x: 100, fill: 'red', updatedAt: 1005 }

All Users:
  1. onSnapshot receives merged state
  2. Compare timestamps: 1005 > local timestamp
  3. Remote version wins → Apply changes
  4. Clear local updates for this shape
  5. Consistent state across all clients
```

Security Rules

Firestore Rules:

```
rules_version = '2';
service cloud.firestore {
  match /databases/{database}/documents {
    // Users can only read/write their own profile
    match /users/{userId} {
      allow read: if request.auth != null;
      allow write: if request.auth.uid == userId;
    }

    // Group members can read/write shapes in their group
    match /groups/{groupId}/canvases/{canvasId}/shapes/{shapeId} {
      allow read: if request.auth != null
        &&
get(/databases/$(database)/documents/users/$(request.auth.uid)).data.groupId ==
groupId;
      allow write: if request.auth != null
        &&
get(/databases/$(database)/documents/users/$(request.auth.uid)).data.groupId ==
groupId;
    }
  }
}
```

Realtime Database Rules:

```
{
  "rules": {
    ".read": "auth != null",
    ".write": false,
    "groups": {
      "$groupId": {
        "cursors": {
          "$userId": {
            ".write": "auth != null && auth.uid == $userId"
          }
        },
        "presence": {
          "$userId": {
            ".write": "auth != null && auth.uid == $userId"
          }
        },
        "selections": {
          "$userId": {
            ".write": "auth != null && auth.uid == $userId"
          }
        }
      }
    }
  }
}
```

💡 Key Differentiators (1 minute)

vs Figma

✅ **Lightweight**: No project files, instant access
✅ **AI-powered**: Natural language shape creation
✅ **Simpler**: Focus on quick collaboration, not full design
❌ **Less features**: No components, auto-layout, or plugins
❌ **Less polish**: Canvas is more whiteboard, less design tool

vs Miro/Mural

✅ **Developer-friendly**: Built for technical teams
✅ **AI integration**: GPT-4o powered commands
✅ **Faster sync**: < 50ms vs 200ms+
✅ **Free & open source**: No per-user licensing
❌ **Fewer templates**: No sticky notes, frameworks
❌ **Less collaboration features**: No voting, timers, etc.

vs Google Jamboard

✅ **Unlimited canvas**: 5000x5000 vs fixed board size
✅ **Figma-style UX**: Familiar for designers
✅ **Real-time**: < 50ms sync vs 1-2 second delays
✅ **AI assistant**: Natural language commands
❌ **No mobile apps**: Browser only (Jamboard has mobile)

Unique Selling Points

1. **AI + Real-time**: First collaborative canvas with integrated GPT-4o
2. **Figma DNA**: Multiplayer experience familiar to designers
3. **Open source**: Self-host, customize, extend
4. **Serverless**: Firebase backend = zero maintenance
5. **Client-side AI**: Your API key never leaves your browser

---

🚀 Demo Flow Suggestions (Complete Script)

Act 1: The Hook (1 minute)

1. **Open app** → Show clean, familiar Figma-like interface
2. **Two browsers side-by-side** → Different user colors
3. **Move shape in browser A** → Instant update in browser B
4. **Show multiplayer cursors** moving in real-time
5. **Select shape in browser B** → Glow appears in browser A
6. **Say**: "This is CollabCanvas - Figma's multiplayer magic meets AI-powered whiteboarding"

Act 2: Manual Drawing (2 minutes)

1. **Two-step line drawing**:
   - Click → Move → Click
   - Show live preview
2. **Two-step rectangle**:
   - Click corner → Drag → Click
   - Resize with corner handles
3. **Circle with edge handles**:
   - Click center → Move → Click

    - Drag edge handle to resize

4. **Text editing**:
    - Click to place → Double-click to edit
    - Show font size controls
    - Change color with dual pickers
5. **Say**: "All the fundamentals you expect, with Figma-quality interactions"

Act 3: AI Assistant (3 minutes)

1. **Click AI button in footer**
2. **Command 1**: "Create a red circle at 300, 200"
    - Show instant creation
    - Point out natural language
3. **Command 2**: "Make a 3x3 grid of blue squares"
    - Show batch operation
    - Highlight speed (19 shapes in 1 second)
4. **Command 3**: "Move the blue circle 200 pixels left"
    - Show relative movement
    - Point out shape matching
5. **Command 4**: "Find the text that says 'Login' and center it in the rectangle"
    - Show spatial awareness
    - Highlight layout capabilities
6. **Command 5**: "Bring the red circle to the front"
    - Show layer control
    - Point out z-index management
7. **Say**: "19 specialized tools, powered by GPT-4o. Describe what you want, not how to do it."

Act 4: Real-Time Collaboration (2 minutes)

1. **Show group creation**: Register with "Design Team"
2. **Second browser**: Join "Design Team"
    - Show same canvas
3. **Third browser**: Join "Marketing Team"
    - Show different canvas (isolation)
4. **Back to Design Team**:
    - User A creates shapes
    - User B moves them
    - Show selection awareness
    - Show cursors moving
5. **Say**: "Group-based workspaces keep teams isolated. No accidental interference."

Act 5: Performance & Polish (1 minute)

1. **Create 100+ shapes rapidly**
    - Show 60 FPS maintained
2. **Drag shapes smoothly**
    - Show optimistic updates
3. **Open DevTools**:
    - Show throttled writes

   - Point out < 50ms sync times

4. **Undo/Redo**:
   - Ctrl+Z → Watch shapes disappear
   - Ctrl+Y → Watch them reappear
5. **Say**: "Built for performance. 60 FPS with 500+ shapes and 5+ concurrent users."

Act 6: The Close (1 minute)

1. **Show help modal**:
   - 6 organized sections
   - Keyboard shortcuts
2. **Show health check page**:
   - Real-time API status
   - Service monitoring
3. **Final screen**: Architecture diagram
4. **Say**:
   - "CollabCanvas: Real-time collaboration meets AI assistance"
   - "Built for designers, loved by teams"
   - "Open source, serverless, and ready to deploy"
   - "Try it at [your-demo-url]"

🎯 Key Metrics to Highlight

Performance

- **60 FPS rendering** with 500+ shapes
- **< 50ms object sync latency**
- **< 30ms cursor sync latency**
- **< 10ms conflict resolution**
- **5+ concurrent users** supported
- **Optimistic updates** = 0ms perceived latency

User Experience

- **Zero learning curve** for Figma users
- **Two-step drawing** with live preview
- **Contextual UI** (only relevant controls shown)
- **40px button heights** (meets WCAG guidelines)
- **Pastel color palette** (reduced eye strain)

AI Capabilities

- **19 specialized tools** for canvas operations
- **GPT-4o model** (latest OpenAI technology)
- **LangChain orchestration** for multi-step operations
- **Natural language** command interface
- **Conversation context** (last 10 messages)
- **Spatial awareness** (blank space detection)

Architecture

- **Dual-database design** (Firestore + Realtime DB)
- **Client-side AI** (API key never leaves browser)

- **Serverless backend** (zero maintenance)
- **Group-based isolation** (team workspaces)
- **Last-Write-Wins** conflict resolution
- **Optimistic updates** with background sync

🔮 Future Roadmap (30 seconds)

Planned Features

1. **Advanced AI Tools**:
    - Vision capabilities ("describe this layout")
    - Fine-tuned model on canvas-specific operations
    - Auto-generate wireframes from descriptions
2. **Enhanced Collaboration**:
    - Voice comments
    - Live video cursors
    - Session replay
3. **Designer Tools**:
    - Components & instances
    - Shared color palettes
    - Typography system
    - SVG import/export
4. **Performance**:
    - Viewport culling (only render visible shapes)
    - WebRTC for peer-to-peer cursor sync
    - Spatial indexing for 10,000+ shapes
5. **Enterprise**:
    - SSO integration
    - Audit logs
    - Version history
    - Canvas templates

Say:

"This is just the beginning. We're building the collaborative canvas that designers deserve - powered by AI, optimized for teams, and built for the future."

📊 Technical Specifications Summary

Frontend Stack

- **Framework**: React 18.2.0
- **Language**: TypeScript 5.x
- **Build Tool**: Vite 5.0.8
- **Canvas**: Konva.js 9.3.0 + React-Konva 18.2.10
- **Styling**: Tailwind CSS 3.3.6

- **Icons**: Lucide React 0.294.0

AI Stack

- **Model**: OpenAI GPT-4o
- **Orchestration**: LangChain 0.3.36
    - @langchain/openai 0.6.16
    - @langchain/core 0.3.78
- **Validation**: Zod 3.25.76
- **Tools**: 19 custom DynamicStructuredTools

Backend Stack

- **Authentication**: Firebase Auth (email/password)
- **Persistent Database**: Firestore (shapes, users, groups)
- **Ephemeral Database**: Firebase Realtime Database (cursors, presence, selections)
- **Hosting**: Firebase Hosting
- **Security**: Firestore Rules (group-based access control)

Performance Characteristics

- **Frame Rate**: 60 FPS (consistent)
- **Shape Sync Latency**: < 50ms (avg)
- **Cursor Sync Latency**: < 30ms (avg)
- **Conflict Resolution**: < 10ms (timestamp comparison)
- **Throttle Intervals**:
    - Cursor updates: 75ms
    - Shape updates: 50ms batch + 200ms final flush
    - Firebase writes: 50ms with auto-flush

Scalability

- **Concurrent Users**: 5+ (tested)
- **Shape Capacity**: 500+ (tested)
- **Canvas Size**: 5000x5000 pixels
- **Zoom Range**: 10% - 500%
- **History Depth**: 50 states
- **Conversation Context**: 10 messages

🎬 Closing Remarks

Elevator Pitch

"CollabCanvas brings Figma's real-time multiplayer experience to a lightweight whiteboard, powered by GPT-4o AI assistance. Perfect for design teams who need quick brainstorming without the complexity of full design tools."

Target Audience Reminder

- **Primary**: UI/UX professionals who use Figma daily
- **Secondary**: Product teams needing visual collaboration
- **Tertiary**: Developers building design-adjacent tools

Why It Matters

1. **Fills a gap**: Between simple whiteboards and complex design tools
2. **AI-native**: Built for the era of AI-assisted design
3. **Open source**: Extensible, customizable, self-hostable
4. **Serverless**: Zero maintenance, infinite scale (via Firebase)
5. **Designer-first**: Built by designers, for designers

Call to Action

- **Try it**: [your-demo-url]
- **Explore the code**: [github-url]
- **Read the docs**: [architecture-docs-url]
- **Join the community**: [discord/slack-url]

📚 Additional Resources

Documentation

- **Architecture Deep Dive**: `/docs/architecture/ARCHITECTURE.md`
- **AI Assistant Guide**: `/AI_CHAT_GUIDE.md`
- **Groups User Guide**: `/GROUPS_USER_GUIDE.md`
- **Deployment Guide**: `/docs/deployment/DEPLOYMENT.md`
- **Health Check Docs**: `/docs/HEALTHCHECK.md`

Live Links

- **Production App**: https://collabcanvas-andy.web.app
- **Health Check**: https://collabcanvas-andy.web.app/healthcheck
- **GitHub Repository**: [your-repo-url]

Contact

- **Developer**: [your-name]
- **Email**: [your-email]
- **Twitter**: [your-twitter]
- **LinkedIn**: [your-linkedin]

End of Document

Last Updated: October 2025
Version: 1.0
Target Audience: Figma Users, UI/UX Professionals, Design Teams