# Collaborative Whiteboard Guide

## 1. Setup Instructions

1. Clone the repository:  
 git clone https://github.com/Jsid21/Collaborative-Whiteboard-Development-.git  
 cd Collaborative-Whiteboard-Development-  
  
2. Install dependencies:  
 - Server:  
 cd server  
 npm install  
  
 - Client:  
 cd ../client  
 npm install  
  
3. Configure environment:  
 - Copy sample files:  
 cp server/.env.example server/.env  
 cp client/.env.example client/.env  
 - Edit each .env to set MONGODB\_URI, JWT\_SECRET, CLIENT\_URL.  
  
4. Run in development:  
 - Server (port 5000):  
 cd server  
 npm run dev  
  
 - Client (port 3000):  
 cd ../client  
 npm start  
  
Open browser at http://localhost:3000

## 2. API Documentation

### Socket.IO Events

|  |  |  |  |
| --- | --- | --- | --- |
| Event | Direction | Payload | Description |
| join-room | client → server | { roomId: string, userName: string } | Join an existing board (creates socket room) |
| init-canvas | server → client | { strokes: Stroke[] } | Send the current drawing history on join |
| draw | client → server | { roomId, stroke: Stroke } | Broadcast a new stroke to other clients |
| clear-board | client → server | { roomId } | Clear all strokes on the board |
| cursor-move | client ↔ server | { roomId, x: number, y: number } | Track pointer position (optional UX) |
| leave-room | client → server | { roomId } | Leave the board (cleanup) |

### REST Endpoints

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Method | Endpoint | Request Body | Response | Description |
| GET | /api/boards | — | Board[] | List all saved boards |
| POST | /api/boards | { name: string } | Board | Create a new board (returns its ID) |
| GET | /api/boards/:id | — | Board & { strokes: Stroke[] } | Fetch board metadata + drawing history |
| PUT | /api/boards/:id | { strokes: Stroke[] } | Board | Overwrite saved strokes (auto-save) |
| DELETE | /api/boards/:id | — | { success: boolean } | Delete a saved board |

## 3. Architecture Overview

1. Client (React + Socket.IO-client):  
- Canvas component captures events and emits draw events.  
- Listens for peer events to update the canvas in real time.  
  
2. Server (Node.js + Express + Socket.IO):  
- HTTP layer (/api/\*) handles CRUD for boards, persists to MongoDB.  
- WebSocket layer manages rooms, broadcasts drawing events between clients.  
- On join-room, loads saved strokes and emits init-canvas.  
  
3. Database (MongoDB):  
- Boards collection: {\_id, name, createdAt, updatedAt}  
- Strokes embedded per board.  
  
4. Flow:  
 1. Client lists or creates a board via REST.  
 2. Joins room via Socket.IO → receives init-canvas.  
 3. User draws → draw event → server broadcasts and persists.

## 4. Deployment Guide

1. Prepare production build:  
- Client: npm run build (outputs to client/build)  
- Server: npm run build (if using TypeScript)  
  
2. Configure environment on the server:  
- NODE\_ENV=production, PORT, MONGODB\_URI, JWT\_SECRET, CORS\_ORIGIN.  
  
3. Serve static client:  
- Option A: Via Express static middleware.  
- Option B: Nginx: serve client/build, proxy /api and /socket.io to Node.  
  
4. Process management:  
- Use PM2 or Docker to run and auto-restart the server.  
  
5. SSL & Domain:  
- Use Let’s Encrypt for HTTPS or cloud load balancer SSL.  
- Point DNS to your server's IP.  
  
Once complete, access your whiteboard over HTTPS at your domain.