# Scrawl

## A Freeform Shared Journal Application

Mariam Eqbal  Sheetal Lodhi

February 25, 2026

# Objective

1. To build a real-time, freeform shared canvas for two people to communicate expressively.
2. To learn and establish proficiency in a modern full-stack web development stack (React, Node.js, Socket.io, Supabase).
3. To design an intimate, asynchronous digital experience that feels like passing a physical journal.

# What is Scrawl?

1. A private, infinite shared canvas between exactly **two people**.
2. Users can place **freehand drawings**, **text**, and **images** anywhere on the canvas.
3. Content is composed privately as a **draft**, then sent all at once - like sealing and posting a letter.
4. Once sent, all content is **locked forever** - nothing can be moved or deleted.
5. The canvas grows **downward infinitely**, like an endless shared notebook.

# Tech Stack

1. **Frontend:** React (Vite), react-konva, React Router
2. **Backend:** Node.js, Express, Socket.io
3. **Database:** Supabase (PostgreSQL) for element persistence and room management
4. **Storage:** Supabase Storage for uploaded images
5. **Authentication:** Custom JWT + Google OAuth via `@react-oauth/google`
6. **Deployment:** Vercel (frontend), Railway (backend)

# Features Implemented

1. **Infinite canvas** - scrolls downward, clamped at the top like a real journal
2. **Freehand drawing** - smooth lines with color picker and brush size control
3. **Click-to-place text** - type anywhere on the canvas in any color
4. **Image upload** - photos placed directly onto the canvas
5. **Draft & Send mechanic** - compose privately, send all at once
6. **Real-time sync** - Socket.io keeps both users in sync instantly
7. **Unique room URLs** - each journal gets a shareable room code
8. **Persistence** - full canvas survives server restarts via Supabase
9. **Google OAuth login** - one-click sign in, no passwords needed
10. **Recency opacity** - older elements fade subtly, newer ones are crisp

# Division of Work

1. **Mariam** - Canvas architecture, drawing engine (Konva), real-time sync (Socket.io), deployment setup, core UX decisions
2. **Sheetal** - Authentication system (JWT + Google OAuth), room management (create/join with room codes), color picker, brush size controls, stickers, timestamp display

# Timeline

1. **Phase 1** - Local canvas: drawing and text placement
2. **Phase 2** - Real-time sync with Socket.io (two tabs stay in sync)
3. **Phase 3** - Unique room URLs, 2-person room lock
4. **Phase 4** - Supabase persistence (canvas survives restarts)
5. **Phase 5** - Image upload via backend + Supabase Storage
6. **Phase 6** - Draft & Send mechanic, element locking, glow on click
7. **Phase 7** - Google OAuth, room codes, polish (recency opacity, scroll clamp, smoother lines)
8. **Phase 8** - Deployment (Vercel + Railway)

# Repository

- GitHub Repository

# Explicit Learnings

1. Building real-time applications with **WebSockets** (Socket.io)
2. Canvas programming with **Konva.js** - coordinates, layers, hit detection
3. Full-stack architecture - React frontend talking to a Node/Express backend
4. Database design and persistence with **Supabase** (PostgreSQL)
5. Authentication flows - JWT tokens, Google OAuth, protected routes
6. File upload and cloud storage (Supabase Storage)

# Implicit Learnings

1. Debugging real-time race conditions and browser focus issues
2. Managing shared state across two live users simultaneously
3. Thinking carefully about **product design** before writing code
4. Dividing and integrating work across a two-person team using Git
5. Handling environment variables and secrets safely across environments
6. The value of incremental, phase-by-phase development

# Challenges

1. **Real-time sync** - React Strict Mode double-mounting caused the room to appear full before a second user joined; fixed by moving socket creation inside `useEffect` with proper cleanup.
2. **Text input focus** - Konva canvas stole focus from the floating HTML input; fixed with a `setTimeout` defer pattern.
3. **Supabase Storage RLS** - Row Level Security policies blocked image uploads from the frontend; solved by routing uploads through the backend using the service role key.
4. **Smooth drawing** - Raw mouse events produced jagged lines; fixed by minimum distance threshold (3px) before sampling a new point.
5. **Team coordination** - Merging Sheetal's auth system into the existing canvas architecture required careful integration.

# What next

1. **UI fixes** - Make the UI prettier
2. **Canvas limitations** - Add limitations to the canvas for better UX
3. **Drawing options** - Increase styling options
4. **Home page** - Create ahome page where the user can access all their journals in one place

# Future Scope

1. **Group journals** - lift the 2-person cap; the userId-based architecture already supports this
2. **Audio notes** - drop a short voice recording onto the canvas as a playable waveform
3. **Time-lapse replay** - replay the journal from the beginning, stroke by stroke
4. **Reactions** - hover over an element and leave a small floating reaction
5. **Mobile app** - React Native port for a natural touch and stylus experience
6. **Export** - download the full journal as a PDF keepsake
7. **End-to-end encryption** - encrypt canvas content so even the server cannot read it

# Thank You