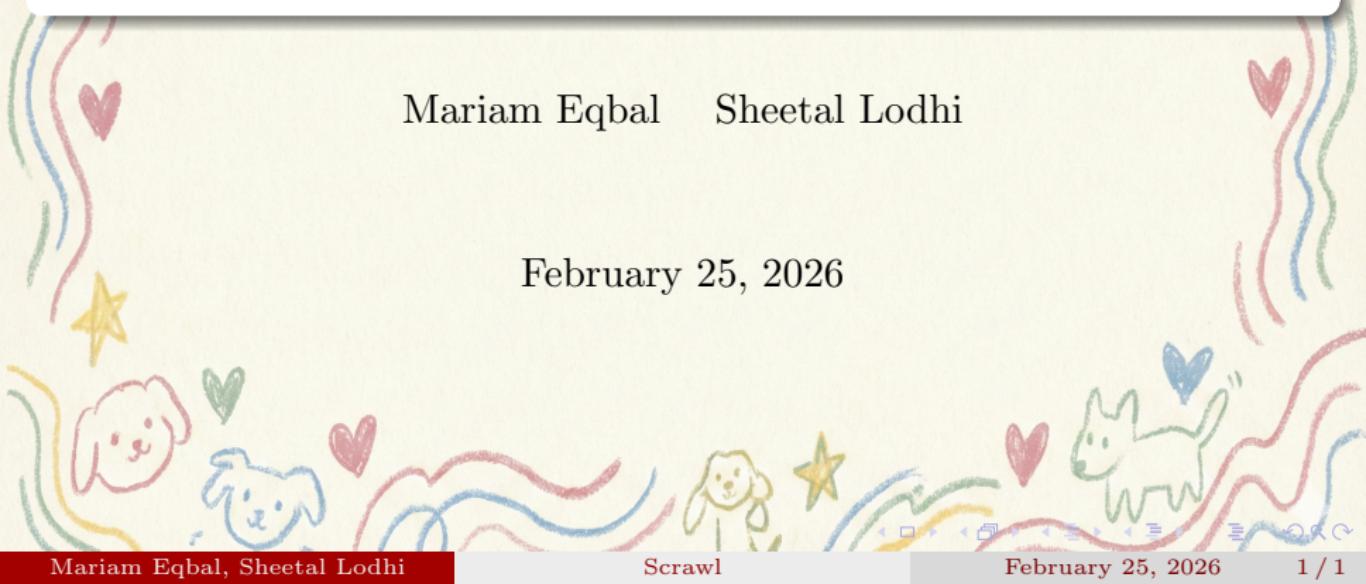




# Scrawl

## A Freeform Shared Journal Application



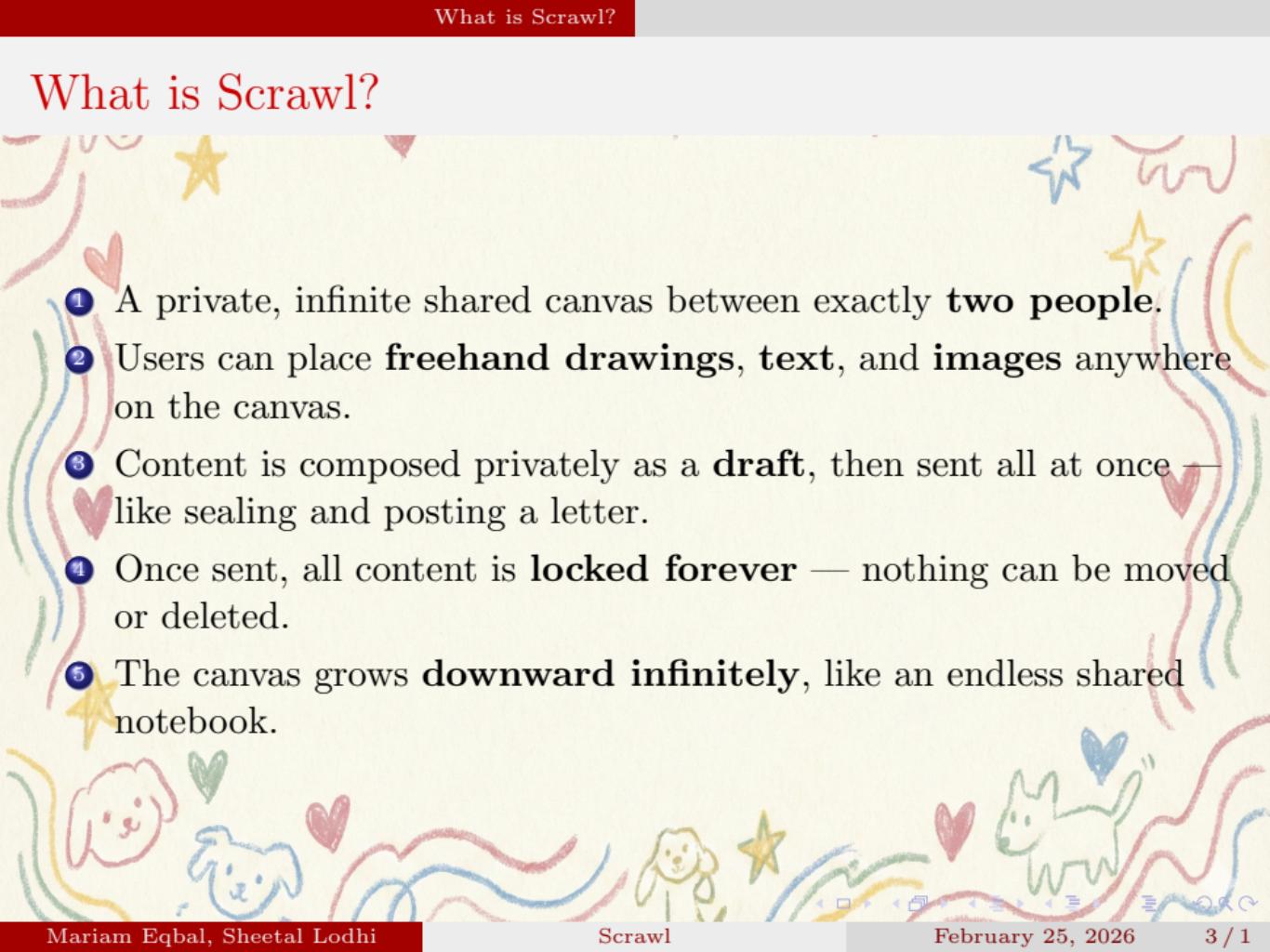
Mariam Eqbal    Sheetal Lodhi

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.

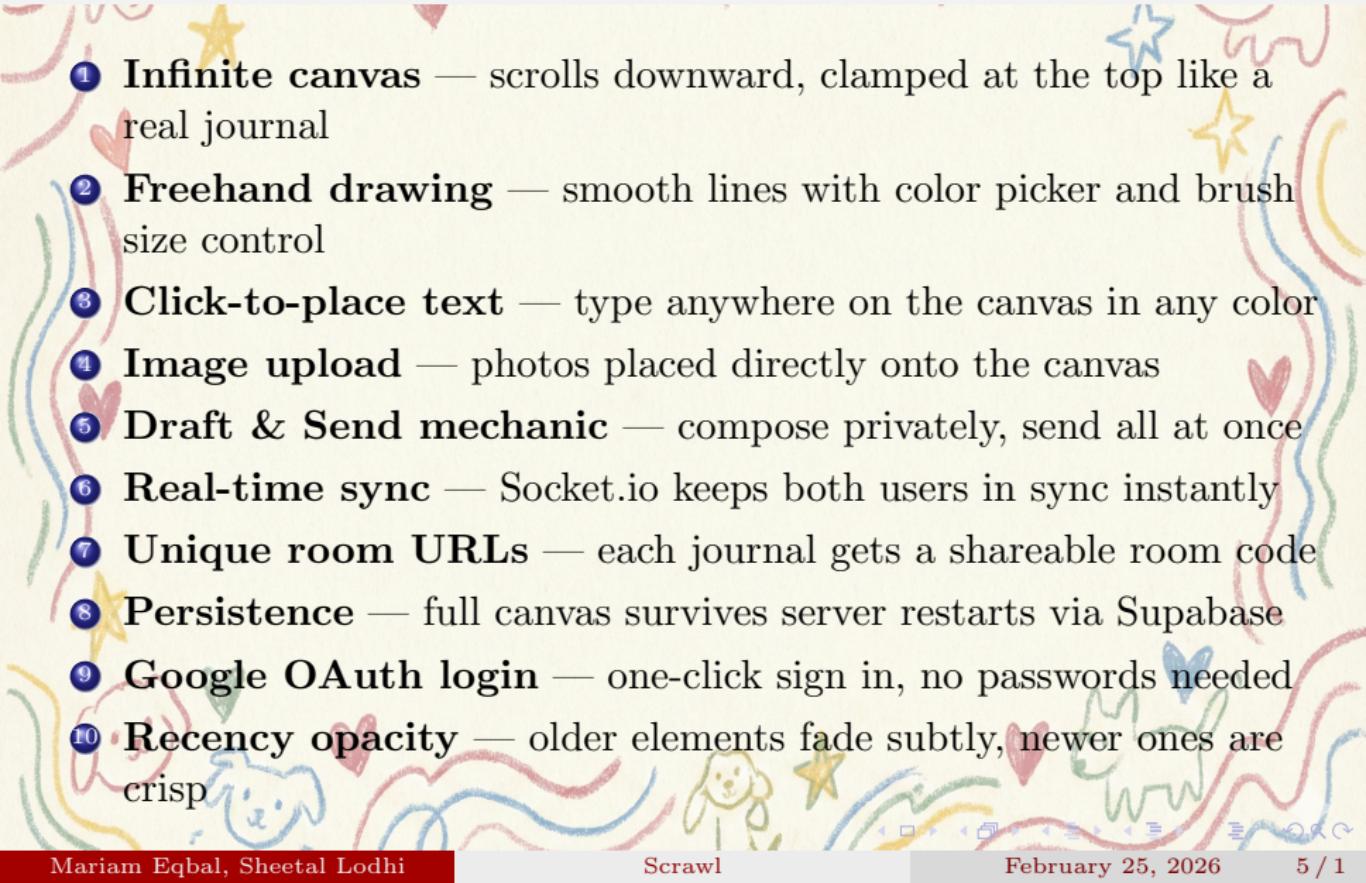
# What is Scrawl?

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

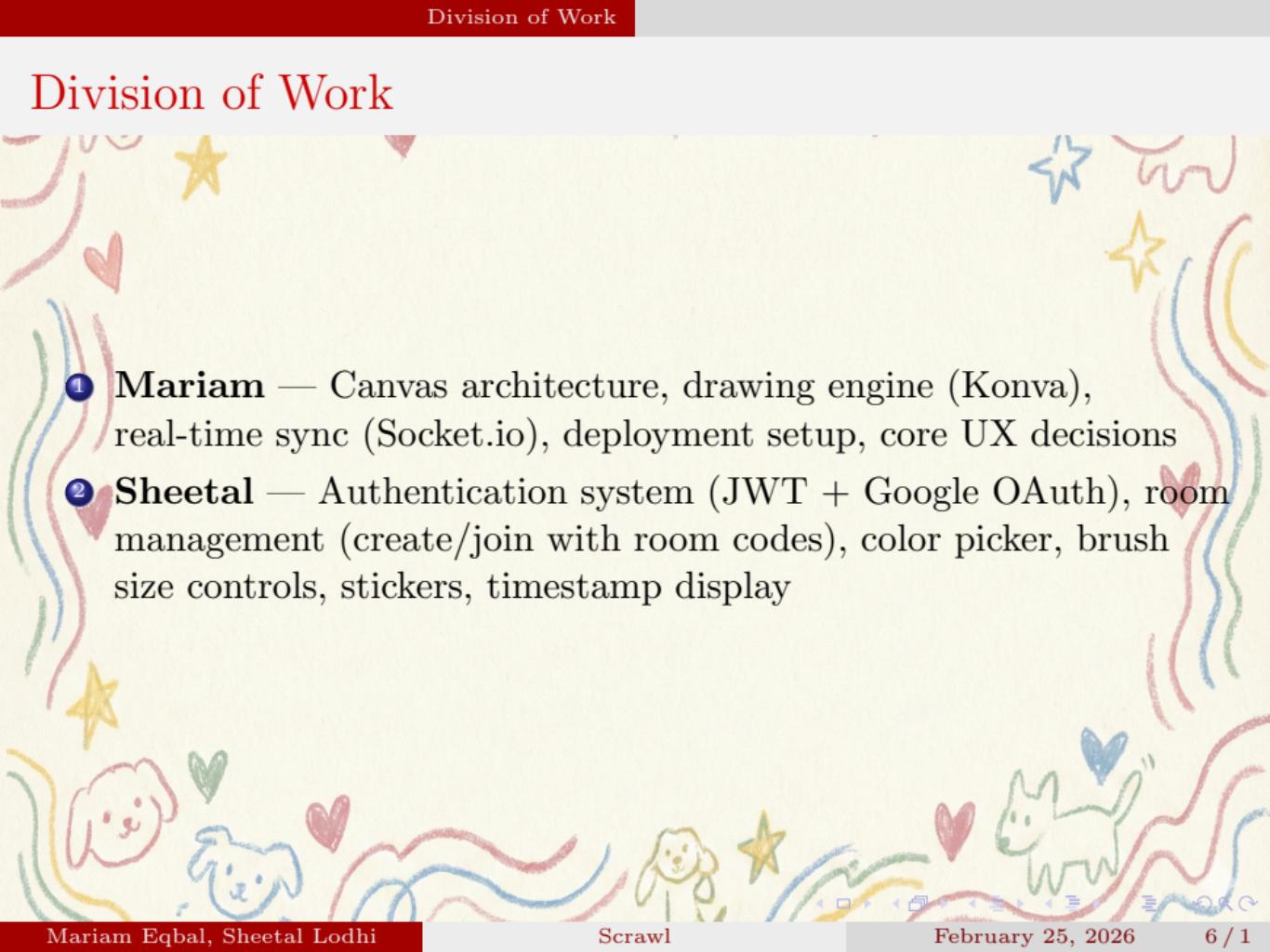
# Tech Stack

- 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)

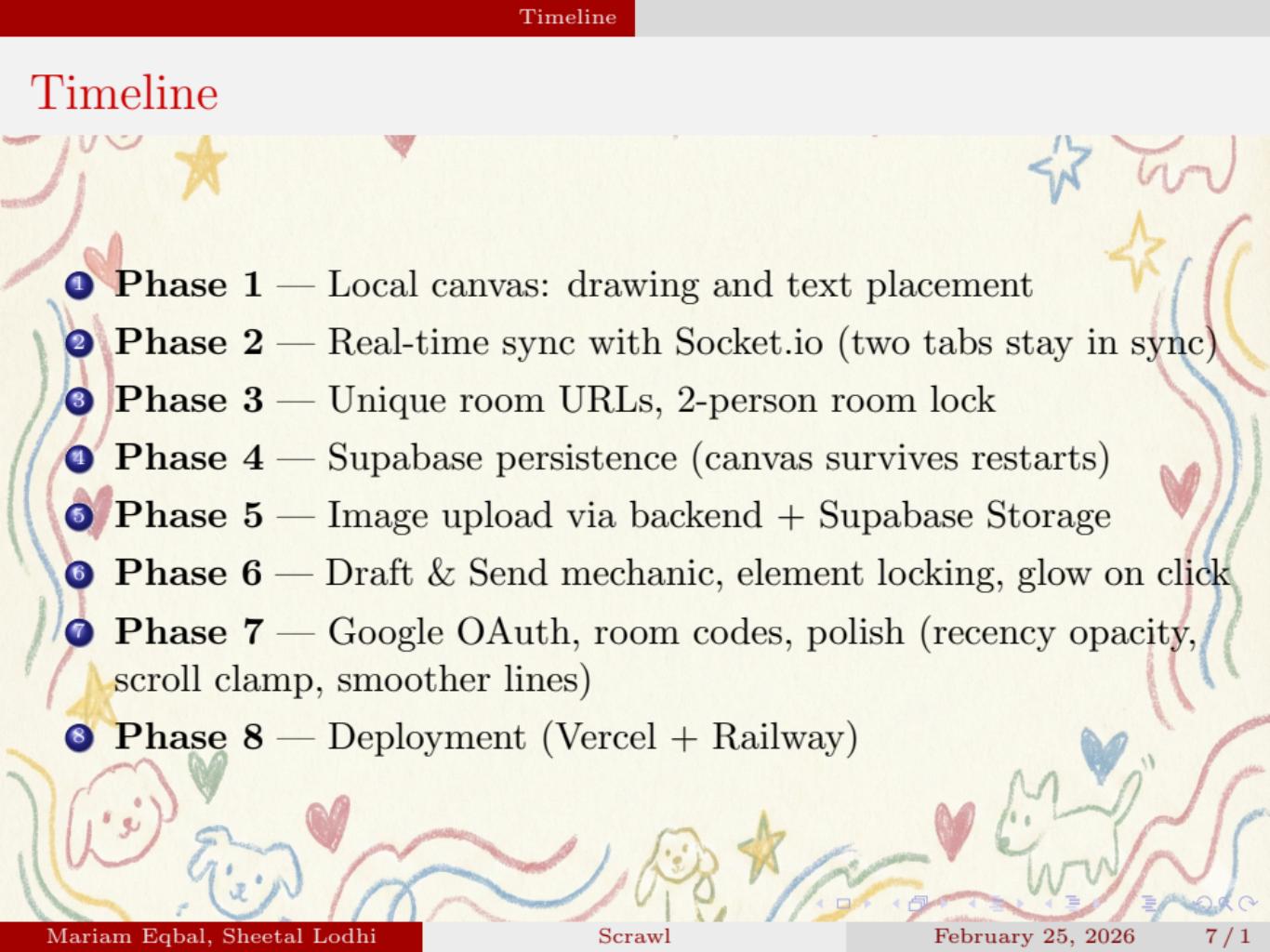
# Features Implemented

- 
- 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)

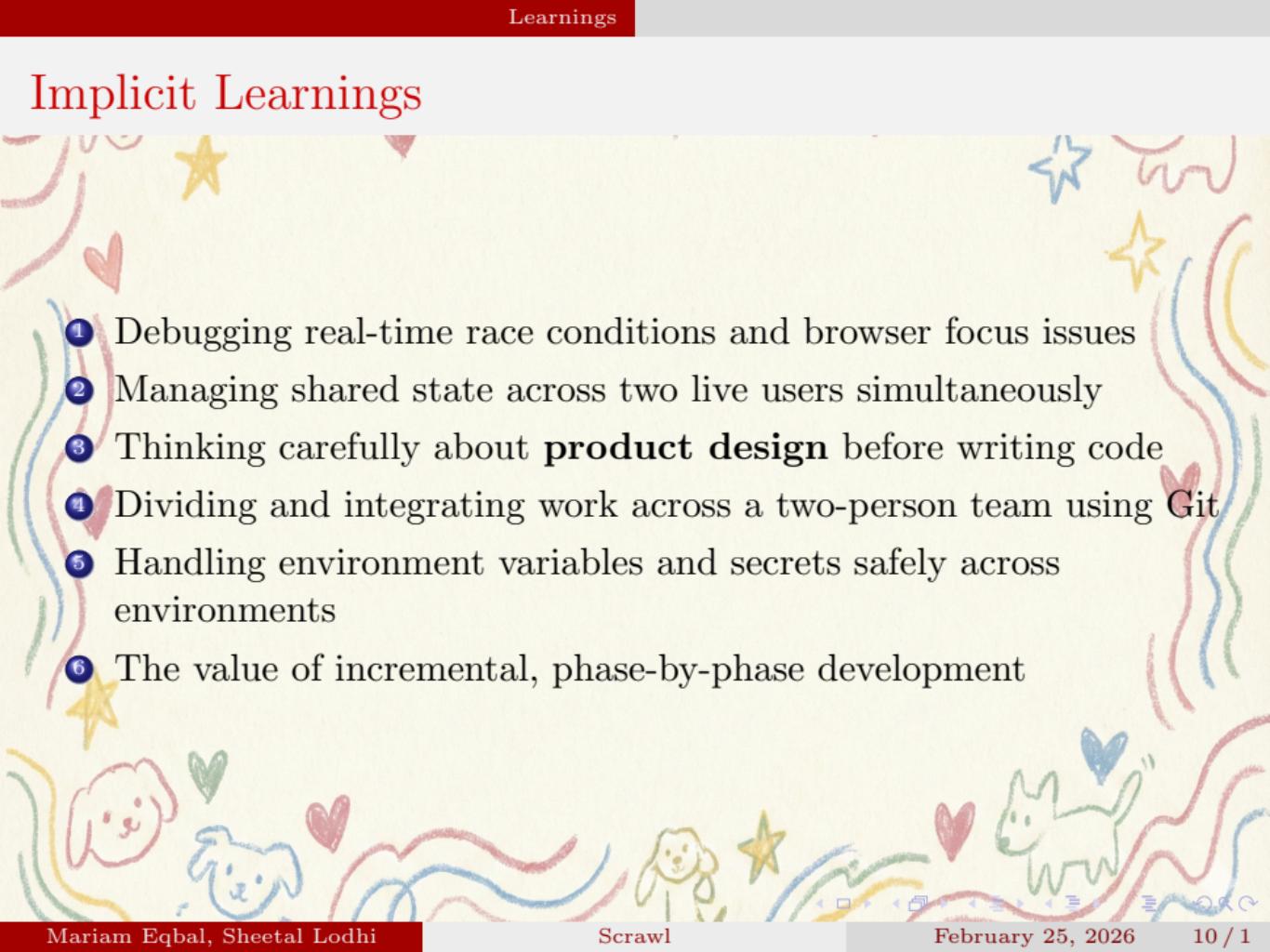
# Repository

► GitHub Repository

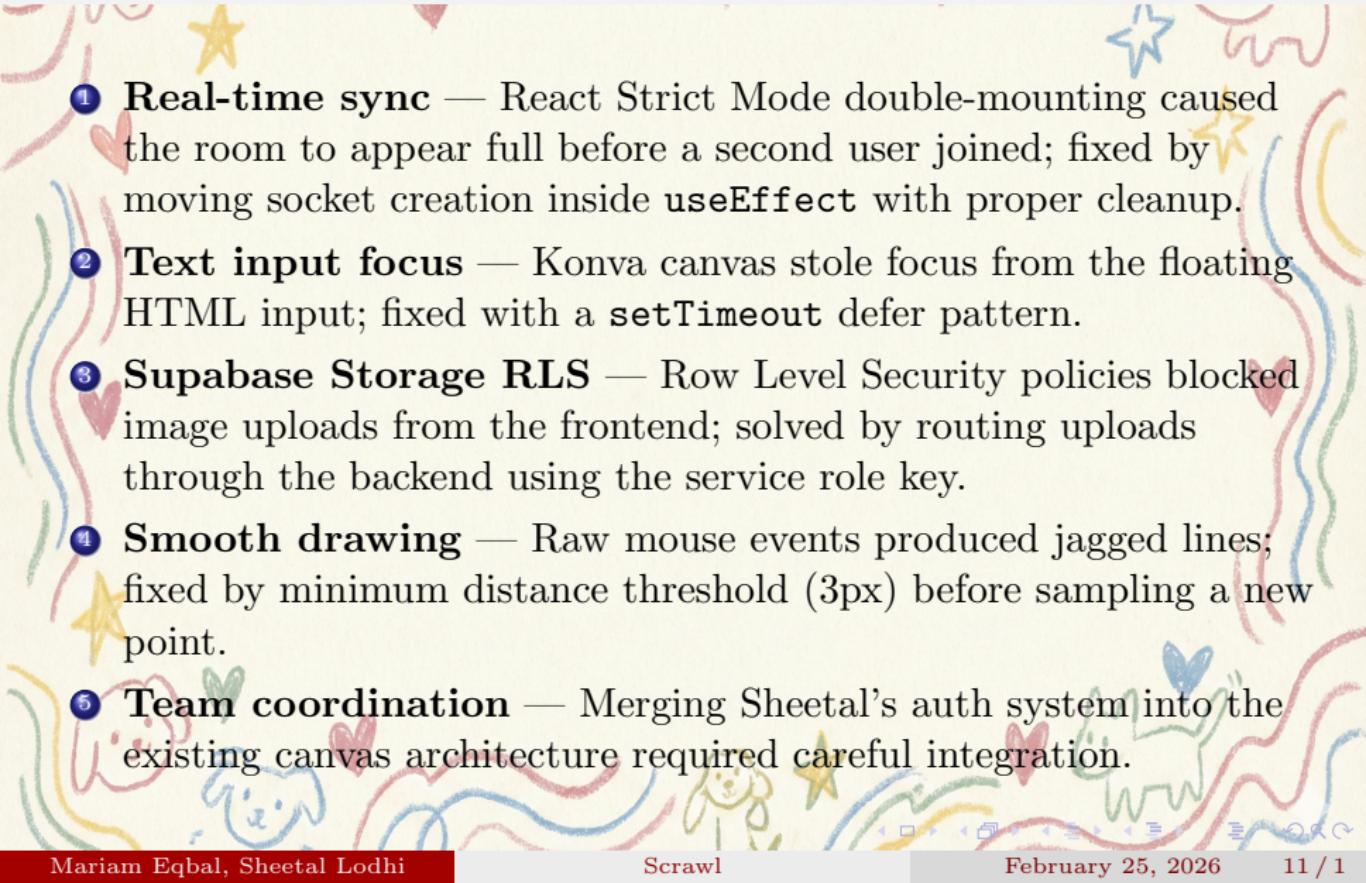
# Explicit Learnings

- ➊ Building real-time applications with **WebSockets** (Socket.io)
- ➋ Canvas programming with **Konva.js** — coordinates, layers, hit detection
- ➌ Full-stack architecture — React frontend talking to a Node/Express backend
- ➍ Database design and persistence with **Supabase** (PostgreSQL)
- ➎ Authentication flows — JWT tokens, Google OAuth, protected routes
- ➏ 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

- 
- ➊ **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.
  - ➋ **Text input focus** — Konva canvas stole focus from the floating HTML input; fixed with a `setTimeout defer` pattern.
  - ➌ **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.
  - ➍ **Smooth drawing** — Raw mouse events produced jagged lines; fixed by minimum distance threshold (3px) before sampling a new point.
  - ➎ **Team coordination** — Merging Sheetal's auth system into the existing canvas architecture required careful integration.

# 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