Hormonalab Symptom Logging Prototype - Technical Documentation

# 1. Introduction

This document provides the technical details of the Hormonalab Symptom Logging Prototype. The solution is designed as a web-based React application with mocked APIs to simulate data flow. It demonstrates a Symptom Logging Form, Predictive Insights, Wearable Data integration, and Recent Entries management. The goal is to deliver a functional prototype suitable for evaluation and potential hiring demonstrations.

# 2. Technology Stack

The prototype is built using the following technologies:

**- React (with Vite)**  
- React Query (TanStack Query) for API calls and caching  
- MSW (Mock Service Worker) for mocking backend APIs  
- Tailwind CSS for responsive UI design  
- TypeScript for type safety  
- Deployed on Vercel/Azure/Netlify (compatible with free tiers)

# 3. Project Structure

The application is structured as a standard Vite + React project. Key directories include:

├── src/  
 │ ├── pages/ # Individual UI pages (LogSymptoms, RecentEntries, PredictiveInsights, WearableData)  
 │ ├── components/ # Shared UI components (navigation, loaders, etc.)  
 │ ├── hooks/ # React Query hooks for data operations  
 │ ├── mocks/ # MSW handlers for mocked APIs  
 │ ├── App.tsx # App routing and layout  
 │ └── main.tsx # Application entry point

# 4. Features Implemented

The following core features have been implemented:

- Symptom Logging Form: Input form for logging symptoms (date, type, severity, duration, notes). Includes Reset, Cancel, and Submit actions with validation and feedback.  
- Recent Entries Page: Displays saved symptoms from the session.  
- Predictive Insights Card (Mock): Simulates calling `/api/predict` with 1–2s delay and shows prediction.  
- Wearable Data (Mock): Simulates fetching wearable health metrics (heart rate, sleep). Includes a Sync option.  
- Responsive UI: Fully mobile-friendly, using Tailwind CSS with a yellow/orange theme inspired by Hormonalab branding.

# 5. Mock API Endpoints

MSW (Mock Service Worker) is used to simulate backend APIs. This ensures the app works locally and when deployed without requiring a real backend.  
- POST /api/symptoms → Saves a symptom entry (in memory)  
- GET /api/symptoms → Fetches session symptom entries  
- GET /api/predict → Simulates AI prediction with delay  
- GET /api/wearables → Returns mocked wearable metrics

# 6. Deployment

The prototype can be deployed to any static hosting service (Vercel, Netlify, Azure Static Web Apps). Since MSW is client-side, the prototype is platform-independent and works identically across environments.

# 7. Extension Considerations

- Authentication: Sessions and health data security can be handled using JWT tokens, HTTPS, and secure cookies.  
- Scaling to Database: Replace MSW handlers with real API endpoints (Node.js/Express + DB) while keeping React Query hooks unchanged.  
- Long API Latency: Show skeleton loaders/spinners with optimistic UI updates to keep the app responsive.  
- Staging & Handoff: Use GitHub repo with README, environment configs, and CI/CD workflows for smooth developer handoff.

# 8. Conclusion

This prototype demonstrates a production-like workflow with mocked APIs, responsive UI, and realistic navigation flow. It can serve as a demo or a base to extend into a full-scale medical/healthcare application.