**Product Requirements Document (PRD) Hai**

**Hai**

A real-time, secure, web-based 1-on-1 chat platform with end-to-end encryption.

**Goal**

Build an MVP web messenger app that allows users to:

* Sign up, log in, and chat 1-on-1
* Send and receive encrypted text messages in real-time
* Get notified when a message is delivered or seen
* Receive in-browser notifications for new messages

**Target Users**

* general users who need secure, lightweight chat
* Desktop/laptop web users

**Core Features (MVP Scope)**

1. User Authentication
   1. Sign up / Sign in with email + password
   2. Authenticated sessions with JWT or Firebase
   3. Logout flow
2. 1-on-1 Messaging
   1. Start a conversation
   2. Send/receive text messages
   3. Messages are stored encrypted
   4. Load previous chat history
3. Real-Time Communication
   1. Real-time message delivery via WebSockets
4. Message Seen/Delivered Status
   1. Seen: User opened chat
   2. Delivered: Message sent to recipient
5. End-to-End Encryption (E2EE)
   1. Messages encrypted on the client
   2. Backend stores ciphertext only
   3. Key exchange between users for secure sessions
6. Notifications
   1. Web Push Notifications
   2. Optional sound/vibration

**Out of Scope (For MVP)**

* Voice/video calls
* Group chats
* Media/file sharing
* Mobile app
* Contact syncing/invite friends

**Technical Overview Backend**

Backend:

* Language: Golang
* Framework: Echo
* Real-time: Gorilla/WebSocket
* Database: PostgreSQL
* Auth: JWT or Firebase Auth
* E2EE: libsodium, AES + RSA
* Deployment: Docker, Fly.io or Render

Frontend:

* Framework: React + Vite
* UI: Tailwind CSS or ShadCN
* State Management: Zustand or hooks
* Real-time: WebSocket Client
* Crypto: Web Crypto API or [libsodium.js](http://libsodium.js)

**Data Models (Draft)**

User:

* id: uuid
* email, username, password\_hash
* public\_key: string
* created\_at: timestamp

Message:

* id: uuid
* sender\_id, receiver\_id
* content: encrypted\_text
* status: sent | delivered | seen
* timestamp: datetime

**User Flow**

Auth Flow:

1. Sign up -> store user + public key
2. Log in -> receive JWT
3. Fetch contacts and messages

Chat Flow:

1. Open chat -> fetch messages
2. Send -> encrypt on client
3. Backend saves and pushes
4. Receiver decrypts and displays
5. Seen status updates

**Timeline (Example MVP Plan)**

Week 1: Auth + Database setup

Week 2: Real-time WebSocket chat

Week 3: Seen/Delivered status

Week 4: E2EE encryption

Week 5: Web notifications + UI polish

Week 6: Testing, deployment, docs

**Success Metrics**

* Real-time messaging works
* Encryption/decryption successful
* UI updates message status
* Notifications trigger
* Demo ready in browser