Chat Application - Architecture Documentation (MVP Version 1)

1. Project Overview

A real-time one-to-one chat application built using the MERN stack: MongoDB, Express.js, React, Node.js, and Socket.IO with JWT Authentication.

- Full-stack system architecture understanding
- REST API design
- WebSocket-based real-time communication
- JWT authentication
- Database schema design
- Deployment and scalability planning

2. MVP Scope

- User registration & login
- JWT authentication
- One-to-one text messaging
- Real-time message delivery
- Message persistence in database
- Pagination (last 20 messages)
- Online/offline presence tracking
- lastSeen tracking
- Logout functionality

3. High-Level Architecture

- React Client
- Node.js + Express Server
- Socket.IO (same server)
- MongoDB Database

4. Database Design

User Collection Fields:

- _id: ObjectId
- name: String
- email: String (unique)
- password: String (hashed)
- lastSeen: Date
- createdAt: Date

- updatedAt: Date

Message Collection Fields:

- _id: ObjectId
- senderId: ObjectId (ref: User)
- receiverId: ObjectId (ref: User)
- text: String
- isRead: Boolean (default: false)
- createdAt: Date
- updatedAt: Date

Compound Index: (senderId, receiverId, createdAt)

5. Authentication Strategy

REST Authentication:

- JWT generated on login/register
- Stored in HTTP-only cookie
- Verified using middleware

Socket Authentication:

- Client sends JWT in auth field
- Server verifies using jwt.verify()
- userId attached to socket object

6. Real-Time Messaging Flow

- Client emits send_message event
- Server validates and authenticates
- Message saved to DB first
- If successful, emit receive_message to receiver
- If receiver offline, message delivered later via history

7. Presence Handling

- Maintain activeUsers map in memory
- On connect: add userId -> socketId
- On disconnect: remove from map
- Update lastSeen in database

8. Pagination Strategy

- Fetch last 20 messages initially

- Load older messages on scroll
- Avoid loading entire history

9. Security Decisions

- Passwords hashed using bcrypt
- JWT verification on protected routes
- Socket authentication mandatory
- Save message before emitting
- HTTP-only cookies for REST auth

10. Future Scalability

- Separate Socket service
- Redis for presence tracking
- Conversation collection
- Message queue integration
- Horizontal scaling
- Cloud storage for attachments