System Design Documentaion

Project Overview

- **Project Name**: Real-Time Chat Application
- **Objective**: Enable real-time text, group chat, and optionally integrate AI-based chatbot functionality.
- **Key Features**:
    - Real-time private and group messaging
    - User authentication and session management using JWT
    - MongoDB for database management
    - Socket.IO for real-time communication
    - Optional AI-powered chatbot

Project Structure

```
IBY_Chat
├─ client
│  ├─ .env
│  ├─ .eslintrc.json
│  ├─ Components
│  │  ├─ button.jsx
│  │  ├─ Chat
│  │  │  ├─ ChatPage.jsx
│  │  │  ├─ ChatWindow
│  │  │  │  ├─ AIChatWindow.jsx
│  │  │  │  ├─ ChatHeader.jsx
│  │  │  │  ├─ ChatInput.jsx
│  │  │  │  ├─ ChatWindow.jsx
│  │  │  │  ├─ Icon.jsx
│  │  │  │  ├─ InputBox.jsx
│  │  │  │  ├─ Message.jsx
│  │  │  │  ├─ SendButton.jsx
│  │  │  │  └─ TextArea.jsx
│  │  │  └─ UserList
│  │  │     ├─ ChatAiBox.jsx
│  │  │     ├─ ChatBar.jsx
│  │  │     ├─ ChatBarMember.jsx
│  │  │     ├─ ChatList.jsx
│  │  │     ├─ ChatListCard.jsx
│  │  │     ├─ CreateChatBox.jsx
│  │  │     ├─ CreateGroup.jsx
│  │  │     ├─ GroupListBox.jsx
│  │  │     ├─ UserChatListBox.jsx
│  │  │     ├─ UserListBox.jsx
│  │  │     └─ UserListCard.jsx
```

```
│   │   ├─ Input.jsx
│   │   ├─ Layout.js
│   │   ├─ LoginBox.jsx
│   │   ├─ SignupBox.jsx
│   │   └─ UserBox.jsx
│   ├─ jsconfig.json
│   ├─ next.config.mjs
│   ├─ package-lock.json
│   ├─ package.json
│   ├─ pages
│   │   ├─ api
│   │   │   └─ hello.js
│   │   ├─ chat.js
│   │   ├─ fonts
│   │   │   ├─ GeistMonoVF.woff
│   │   │   └─ GeistVF.woff
│   │   ├─ index.js
│   │   ├─ login.js
│   │   ├─ register.js
│   │   ├─ _app.js
│   │   └─ _document.js
│   ├─ postcss.config.mjs
│   ├─ public
│   │   ├─ ai_image.png
│   │   └─ favicon.ico
│   ├─ README.md
│   ├─ styles
│   │   └─ globals.css
│   └─ tailwind.config.js
└─ server
    ├─ .env
    ├─ connection
    │   └─ mongoose.js
    ├─ Model
    │   ├─ Group.js
    │   ├─ Message.js
    │   └─ User.js
    ├─ package-lock.json
    ├─ package.json
    ├─ routes
    │   ├─ chatRoutes.js
    │   └─ userRoutes.js
    ├─ server.js
```

Architecture Overview

- **Frontend**:

    - Built with Next.js for server-rendered React applications.
    - Role: Manages user interface, handles requests, and displays real-time chat using web sockets.
- **Backend**:
    - Built with Express.js and Socket.IO for real-time communication.
    - Manages API endpoints, authentication, and database connections.
- **Database**:
    - MongoDB with Mongoose ODM to store users, groups, and messages.
    - Data Schema includes collections for Users, Groups, and Messages.
- **Data Flow**:
    - Users interact with the Next.js frontend, which sends requests to the Express server.
    - The server performs authentication checks, processes data, and interacts with MongoDB to fetch or store data.
    - Socket.IO facilitates real-time communication between clients for messaging and presence updates.
- **API Endpoints**:
    - /api/user: Managing user Registration, Login and Authentication.
    - /api/chat: Coordinate the users for chatting, creating and managing groups for group chat, sending and receiving messages.
- **Libraries and Dependencies**:
    - **Frontend**: next, react, socket.io-client, axio, Cookie.
    - **Backend**: express, mongoose, socket.io, jsonwebtoken, bcrypt, nodemon, openai
    - **Other**: dotenv for environment variables, cors for cross-origin support.

Frontend

Technologies: Next.js, React, Socket.io-client, Axios, Cookie

Components:

- **ChatPage.jsx:** The main page for chat functionalities.
- **ChatWindow:** Contains subcomponents like AIChatWindow.jsx, ChatHeader.jsx, ChatInput.jsx , and others for handling chat interactions.
- **UserList:** Manages the display of users and groups with subcomponents like ChatListCard.jsx, CreateGroup.jsx, etc.
- **Other Components:** LoginBox.jsx, SignupBox.jsx, Layout.js, Input.jsx

Libraries:

- **Next.js** for server-side rendering and routing.
- **React** for building user interfaces.
- **Socket.io-client** for real-time communication.
- **Axios** for making HTTP requests.
- **Cookie** for handling cookies.

Backend

Technologies: Express.js, Mongoose, Socket.io, Jsonwebtoken, Bcrypt, Nodemon, OpenAI

Components:

- **Server Setup:**
    - server.js: Initializes the server and sets up middleware.
    - connection/mongoose.js: Handles MongoDB connection.
- **Models:**
    - User.js: Defines the schema and model for user data.
    - Group.js: Defines the schema and model for group data.
    - Message.js: Defines the schema and model for message data.
- **Routes:**
    - userRoutes.js: Handles user-related API endpoints.
    - chatRoutes.js: Handles chat-related API endpoints.

Libraries:

- **Express.js** for building the backend API.
- **Mongoose** for MongoDB object modeling.
- **Socket.io** for real-time communication.
- **Jsonwebtoken** for authentication.
- **Bcrypt** for hashing passwords.
- **Nodemon** for development.
- **OpenAI** for AI functionalities

Styling:

- **Tailwind CSS** is used for styling the application.

Security

Security measures include using **Jsonwebtoken** for authentication and **Bcrypt** for password hashing. Environment variables are managed using **dotenv**.

Real-Time Communication

Socket.io is used for real-time bidirectional communication between the client and server.