Messaging App Documentation

Table of Contents

---

1. Introduction

This documentation provides a detailed explanation of the messaging app, including its features, architecture, and technical implementation. The app supports one-to-one chat, group chat, video calls, and secure user authentication, providing a seamless communication platform.

2. Features Overview

- **User Authentication**: Secure sign-up and login functionality using email/password or OAuth.
- **One-to-One Chat**: Private chat between two users with text, emojis, and file-sharing capabilities.
- **Group Chat**: Allows users to create groups and participate in group conversations.
- **Video Call**: Supports high-quality video calling between users or within groups.

3. Architecture Overview

The app follows a client-server architecture with a backend API server handling business logic and a frontend client for user interaction. Real-time updates are managed using WebSockets.

Components:

- **Frontend**: Built with React/Ionic for a responsive and intuitive UI.

- **Backend**: REST APIs built with Node.js/Express or Python/FastAPI for handling requests.
- **Database**: SQL/NoSQL database (e.g., MongoDB, PostgreSQL) to store user data, messages, and group information.
- **WebSocket Server**: For real-time message delivery and updates.
- **Media Server**: To handle video calls using WebRTC.

4. Technical Stack

- **Frontend**: React.js / Ionic React
- **Backend**: Node.js (Express) / Python (FastAPI)
- **Database**: MongoDB / PostgreSQL
- **WebSocket Server**: Socket.IO
- **Media Server**: WebRTC / Twilio
- **Authentication**: JWT (JSON Web Tokens), OAuth 2.0

5. User Authentication

5.1. Overview

User authentication is implemented using JWT for secure sessions. Users can register, log in, and reset passwords securely.

5.2. Features

- **Sign Up**: New users can register using email and password.
- **Login**: Users authenticate with their credentials.
- **Password Reset**: Password reset through email.
- **OAuth**: Optional login through third-party providers (Google, Facebook).

5.3. Flow

1. **Sign Up**: User submits credentials → Credentials validated → User created in the database → JWT issued.
2. **Login**: User submits credentials → Credentials validated → JWT issued for session management.
3. **Token Validation**: Each API call includes a JWT for authorization.

6. One-to-One Chat

6.1. Overview

Provides private messaging between two authenticated users with support for text, emojis, and file sharing.

6.2. Key Features

- **Real-time messaging**: Messages are sent and received in real-time using WebSockets.

- **Message Status**: Seen/unseen indicators.
- **File Sharing**: Images, documents, and multimedia can be shared between users.

6.3. Flow

1. **Initiate Chat**: User selects a contact to start a conversation.
2. **Send Message**: Message sent through WebSocket → Backend processes and stores message → Receiver notified in real-time.
3. **Message Storage**: Messages stored in the database with timestamps and status updates.

7. Group Chat

7.1. Overview

Allows multiple users to communicate in a shared group chat environment.

7.2. Key Features

- **Create Groups**: Users can create groups and invite others.
- **Admin Control**: Group admins can manage members.
- **Real-time Updates**: Messages are broadcasted to all group members in real-time.

7.3. Flow

1. **Create Group**: User creates a group → Group data saved in the database.
2. **Send Group Message**: Message sent to WebSocket server → Distributed to all group members.
3. **Admin Actions**: Add/remove members, manage group settings.

8. Video Call

8.1. Overview

Provides video calling capability between users or within groups using WebRTC.

8.2. Key Features

- **One-to-One Calls**: Private video calls between two users.
- **Group Calls**: Group video calls with multiple participants.
- **Call Notifications**: Incoming call notifications for active users.

8.3. Flow

1. **Initiate Call**: User initiates a video call → Call request sent to the recipient.
2. **Connection**: WebRTC establishes peer-to-peer connection for video/audio streaming.
3. **Call Controls**: Mute, video on/off, and call end options are available.

9. Database Design

9.1. Schema Overview

- **Users**: Stores user information (ID, name, email, password hash).
- **Messages**: Stores chat messages (sender, receiver, message content, timestamp).
- **Groups**: Stores group details (group name, members, created by).
- **Calls**: Stores call logs (caller, receiver, call duration, type).

10. API Documentation

10.1. Authentication API

- **POST /auth/signup**: Registers a new user.
- **POST /auth/login**: Authenticates an existing user.
- **POST /auth/reset-password**: Sends password reset link.

10.2. Chat API

- **GET /chat/{userId}**: Fetches chat history with a specific user.
- **POST /chat/send**: Sends a message to a user.

10.3. Group API

- **POST /group/create**: Creates a new group.
- **POST /group/send**: Sends a message to a group.

10.4. Call API

- **POST /call/start**: Initiates a video call.
- **POST /call/end**: Ends the video call.

11. Frontend Overview

The frontend is built using React/Ionic React with a focus on a clean, intuitive UI that allows seamless navigation between chat, group, and call interfaces. WebSocket integration ensures real-time updates.

12. Deployment

The application can be deployed on a cloud platform like AWS, Google Cloud, or Azure. Docker can be used for containerization, and Kubernetes for orchestration if scaling is required.

13. Security Considerations

- **Data Encryption**: All communication is encrypted using SSL/TLS.
- **Authentication**: JWT tokens are securely signed and validated.
- **Input Validation**: Protects against SQL injection, XSS, and CSRF attacks.

14. Future Enhancements

- **AI Chatbot**: Integrate an AI-powered chatbot for user assistance.
- **End-to-End Encryption**: Implement end-to-end encryption for messages and calls.

- **Push Notifications**: Add push notifications for real-time updates.