Frontend Documentation with Chat Features

Overview

MeetMe is a role-based scheduling and communication app designed for academic environments. It allows students to join classes, schedule appointments, and chat with professors. Professors can create classes, offer time slots, manage appointments, and respond to messages.

The app is built using Flutter on the frontend, FastAPI on the backend, and MongoDB for persistent storage.

It leverages REST APIs for data flow and WebSockets for real-time chat.

Real-Time Chat System (WebSocket)
The chat system uses a WebSocket-based connection per user. Each user connects to a unique endpoint:
- 'ws://<host>/ws/chat/{user_id}'

The WebSocket channel is managed through:
- 'ApiService.connectToChat(userId)'
- 'ApiService.sendMessage(senderId, receiverId, message)'
- Messages are JSON-encoded and sent to the backend, which routes them to the recipient if online.

Features implemented in Chat:
- Optimistic UI updates (message appears before confirmation)
- Persistent chat history fetched from '/chat/history'
- Auto-scroll to the latest message
- Message styling based on sender (align right if sent, left if received)
- Stateless handling of reconnection upon ChatScreen build

Chat messages are stored in MongoDB with timestamps and sender/receiver IDs. The backend handles user presence via an in-memory dictionary.

Chat UI Logic (prof_page4.dart & prof_page5.dart)

- 'prof_page4.dart': Lists all enrolled students across classes. Uses Set<String> to avoid duplicate entries.

Tapping a student opens a 'ChatScreen'.
- 'prof_page5.dart' (ChatScreen):
- On 'initState()', establishes WebSocket connection and listens for incoming messages
- Parses and appends messages to 'messages' list
- Scrolls to the bottom when a message is received
- '_sendMessage()' handles optimistic updates and backend dispatch
- On 'dispose()', gracefully closes the WebSocket connection

Chat bubbles are styled with different background and alignment depending on sender. The input bar includes an ‘IconButton’ that calls ‘_sendMessage()’ on tap.

Backend Expectations for Chat

- WebSocket endpoint on ‘/ws/chat/{user_id}’ maintains active user connections
- Chat messages are routed via FastAPI's in-memory connection manager and saved to MongoDB
- A RESTful endpoint ‘/chat/history?user1=A&user2=B’ is available to retrieve past messages
- Messages are stored with:
- sender_id
- receiver_id
- timestamp
- content (text)

The system ensures that chat functionality is available only to enrolled users, and students can only message professors they have classes with.

Benefits of WebSocket for MeetMe

- Real-time experience enhances responsiveness and user satisfaction
- Avoids constant polling, improving battery and data efficiency on mobile devices
- Enables future features like typing indicators, message read receipts, or group chats

The design allows scalability with minimal change, such as supporting multiple conversation threads or chat rooms per class.