Documentation Kit

Project Charter

Project Name: MeetMe

Goal: To create a real-time class management and appointment booking platform for professors and students.

~~Stakeholders: Professors, Students, Administrators, Developers~~

Deliverables: Flutter mobile app, FastAPI backend, MongoDB database

Constraints: Mobile-first, real-time features, secure login

~~Timeline:~~

System Architecture Overview

Frontend: Flutter (Student & Professor UIs)

Backend: FastAPI with REST and WebSocket support

Database: MongoDB

Authentication: Bcrypt password hashing with user role validation

Hosting: Docker-based deployment with Uvicorn and environment configuration

Use Case Summary

1. Student logs in, views classes, books appointments

2. Professor creates classes, defines slots, manages appointments

3. Real-time chat occurs via WebSocket

4. Notifications trigger on booking/rescheduling/cancellation

Deployment & Environment Setup

- Dockerfile defines FastAPI setup with Uvicorn
- docker-compose.yml maps port 8000 -> 5000
- .env file should contain:

MONGO_URI=mongodb+srv://<user>:<pass>@cluster.mongodb.net

- To start server:

docker-compose up --build

~~Optional: Add NGINX or cloud deployment for production use.~~

Security Considerations

- Passwords are hashed using bcrypt
- Role-based routing restricts access to professor/student-only features
- CORS is enabled for development; should be restricted in production
- Appointments and slots are validated to prevent duplicates or overlaps

Testing Strategy

Frontend:

- Flutter test cases for appointment flow, slot display, login

Backend:

- Manual API testing via Postman
- Unit test suggested tools: `pytest`, `httpx`
- Test areas:
- Authentication (valid/invalid)
- Slot booking (double booking, overlap)
- Appointment cancellation and slot restoration
- Real-time chat (mock WebSocket)

Add automated CI tests with GitHub Actions or similar.

Changelog Template

v1.0.0 - Initial backend API completed

v1.1.0 - Added student/professor routing

v1.2.0 - Slot booking conflict resolution

v1.3.0 - WebSocket integration for chat

v1.4.0 - Notifications added

User Guide (For Students & Professors)

Students:

I. Login with credentials
II. Join class using a class code
III. View classes, tap to book slots
IV. Confirm via modal, view appointments
V. Chat with professors if needed

Professors:

I. Create classes
II. Add available time slots per class
III. View appointments (reschedule/cancel)
IV. Monitor notifications and chat

README Sample

Features

- Login/signup for students and professors

- Class creation and enrollment

- Time slot booking and appointment management

- Real-time chat using WebSocket

- Notifications and calendar sync

Tech Stack

- Flutter, Dart

- FastAPI, Python, MongoDB

- Docker, WebSocket, REST APIs

Running the Project

1. Create .env file with MONGO_URI

2. Run `docker-compose up --build`

3. Connect Flutter app to http://localhost:8000

License