SYSTEM DESIGN DOCUMENT

FastChat Messenger

Table of Contents

1) OBJECTIVE:

FastChat Messenger is a web application built using the FastAPI framework to showcase its capabilities and offer a practical example of a chat platform. The application facilitates user registration and authentication, supports one-on-one messaging, and provides real-time updates. The user interface is designed to be simple and intuitive, ensuring ease of use across various devices. FastAPI's efficiency, combined with SQLAlchemy for database management, bcrypt for password hashing, and JWT for authentication, ensures a secure and high-performance backend.

The application also includes an administrative interface created with **Flask-Admin** for managing user accounts and application data. This modular and scalable design allows for future enhancements and easy maintenance. Overall, **FastChat Messenger** not only demonstrates the strengths of FastAPI but also provides a functional and user-friendly chat solution, showcasing modern web development practices.

2) PROJECT OVERVIEW:

The **FastChat Messenger** focuses on essential chat functionalities such as user registration and authentication, enabling users to send and receive text messages. Real-time message updates are integral to the system, ensuring that conversations are current and interactive. By incorporating these features, the project also explores the integration of FastAPI with technologies like **SQLAlchemy for relational databases** and passlib for secure password hashing. The end goal is to deliver a well-documented, scalable, and secure messaging application that highlights FastAPI's strengths and offers a practical example of modern web development using Python.

3) KEY FEATURES:

User Registration & Authentication:

- Users can register and log into the system securely.
- Passwords are hashed using bcrypt to ensure security.
- JSON Web Tokens (JWT) are used for secure user authentication.
- Logout functionality allows users to delete sessions and invalidate tokens.

Messaging:

- One-on-one messaging between users.
- Real-time message updates to ensure instant feedback during communication.

Session Management:

- Secure token-based sessions for logged-in users.
- Users can log out, where sessions are securely invalidated to prevent unauthorised access.

User Interface:

- A clean and intuitive interface to ensure ease of navigation and use.
- The design follows Atomic Design principles for scalable UI development.

4) ARCHITECTURE DESIGN :

a) High-Level Architecture

Frontend (Next.js): Handles user interaction, rendering components based on **Atomic Design** principles, and making API requests.

Backend (FastAPI): Manages user authentication, messaging, and database interactions. Supports real-time messaging via WebSockets.

Database (SQLAlchemy): Manages user, message, and group data in a relational database.

a) Component Breakdown

Frontend:

- Registration and login forms.
- Chat interface (for private and group messaging).
- Group management (creating/joining groups).

Backend:

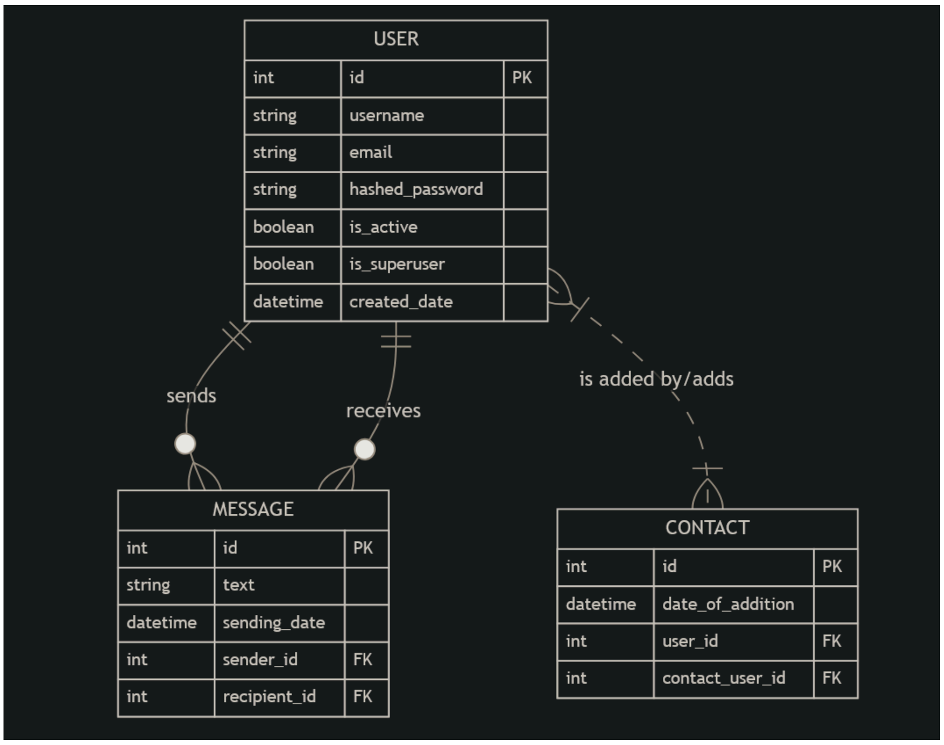
- User authentication (JWT).
- Message and group management (REST APIs).

Database:

- Users: Stores user details (e.g., email, hashed password).
- Messages: Stores messages with sender, recipient, timestamp.

5) DATABASE DESIGN

a) ER diagram



b) Tables

Users Table: Stores user data such as usernames, email addresses, and hashed passwords.

Messages Table: Stores all messages with associated sender, recipient, timestamp, and group references (if applicable).

Contact Table: Stores information of Contacts

6) API DESIGN

a) Endpoints Overview

1) **User Authentication**:
- `POST /register/`: Registers a new user.
- `POST /login/`: Authenticates a user using JWT.

2) **Messaging**:
- `POST /message/`: Sends a message to a user or group.
- `GET /messages/{user_id}`: Retrieves messages for a specific user.

b) API Details

1) **Registration**:
- Input: username, email, password.
- Output: Success message with JWT token.

2) **Messaging**:
- Input: message, sender_id, recipient_id
- Output: Success message or error.

7) DESIGN CHOICES

1. **FastAPI** was selected for its speed and ease of use, especially for developing APIs. It also offers interactive documentation through Swagger and ReDoc, which are automatically generated.
2. **SQLAlchemy** was chosen as the ORM for its capability to work well with relational databases, and it integrates seamlessly with FastAPI.
3. **Next.js** is used for the frontend due to its ability to handle server-side rendering, which improves performance and SEO. Atomic Design principles were used to maintain a structured and scalable component system.
4. **WebSockets** ensure real-time communication, which is crucial for live chat applications. FastAPI's asynchronous capabilities make it well-suited for handling WebSocket connections.
5. **PyJWT** is employed for session management, offering a lightweight and secure method to handle user authentication without needing server-side session storage.

8) SETUP AND RUNNING THE PROTOTYPE

a) Prerequisites

- **Python 3.9+** installed on your system.
- **Node.js** and **npm** for managing frontend dependencies.
- **MySQL** (or any SQL-based DB) for database management.
- **Git** for version control.

b) Prerequisites

1) Clone the repository and create a python virtual environment

```
git clone https://github.com/DeekshithaDommati/FastChat.git
```

2) Navigate to messenger and create-activate the virtual environment :

```
cd messenger
python -m venv venv
venv\Scripts\activate
```

3) Install all the required packages and paste them in requirements.txt :

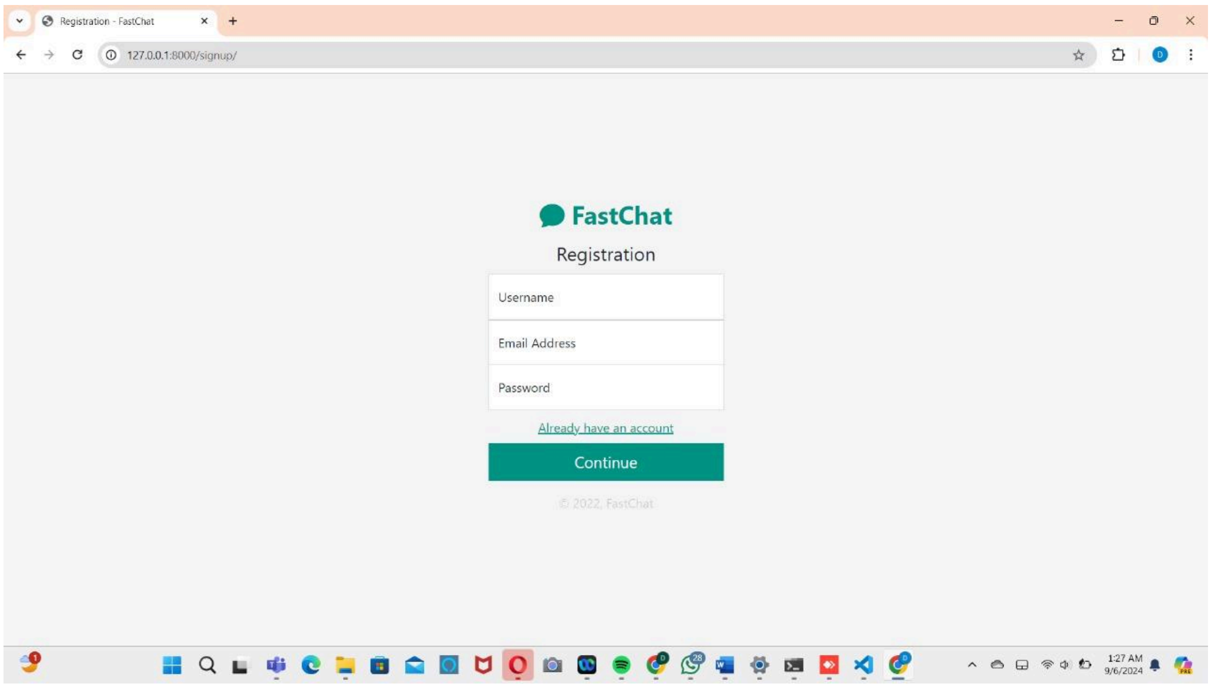
```
pip install -r requirements.txt
pip freeze requirements.txt
```

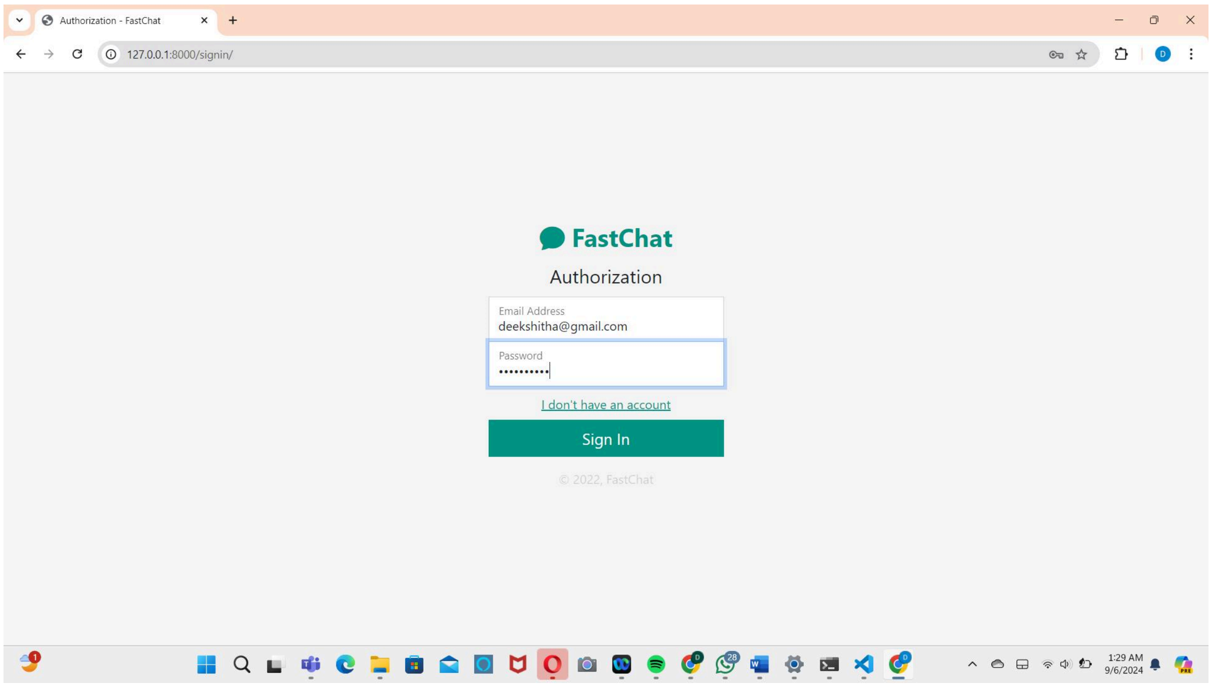
4) Navigate to backend and run the application

```
cd backend
uvicorn main:app --reload
```

5) Running the application :

- Open your browser and navigate to `http://localhost:3000` for the frontend.
- The backend API documentation can be accessed at `http://localhost:8000/docs`.




Authorization - FastChat

127.0.0.1:8000/signin/

FastChat

Authorization

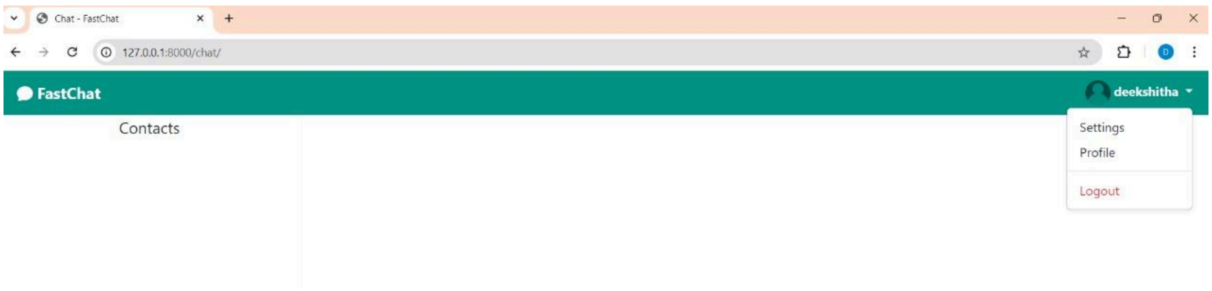
Email Address
deekshitha@gmail.com

Password
••••••••••

I don't have an account

Sign In

© 2022, FastChat


Chat - FastChat

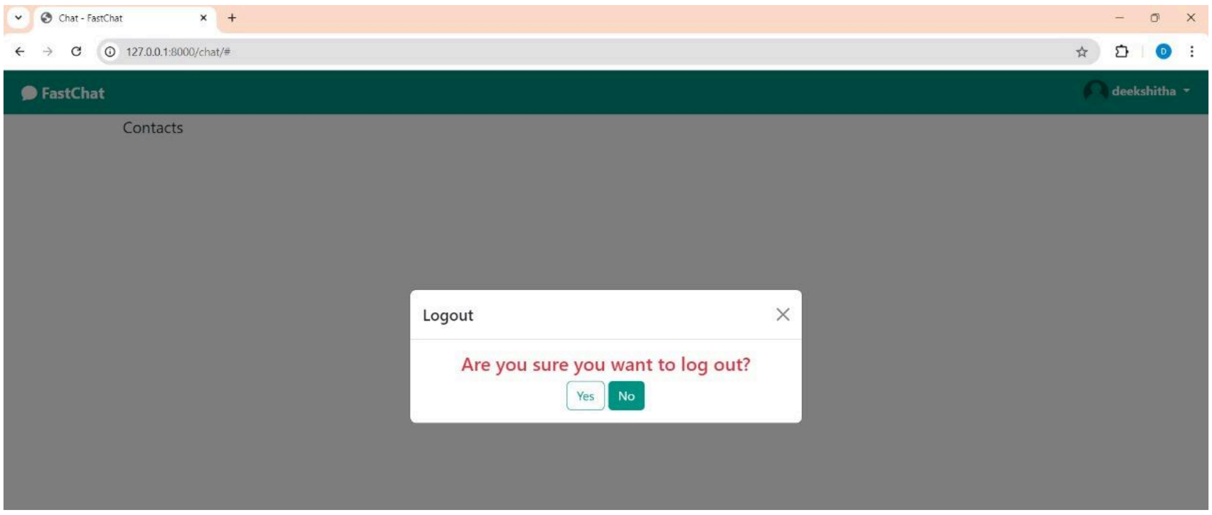
127.0.0.1:8000/chat/

FastChat

deekshitha

Contacts

Settings
Profile
Logout


Chat - FastChat

127.0.0.1:8000/chat/#

FastChat

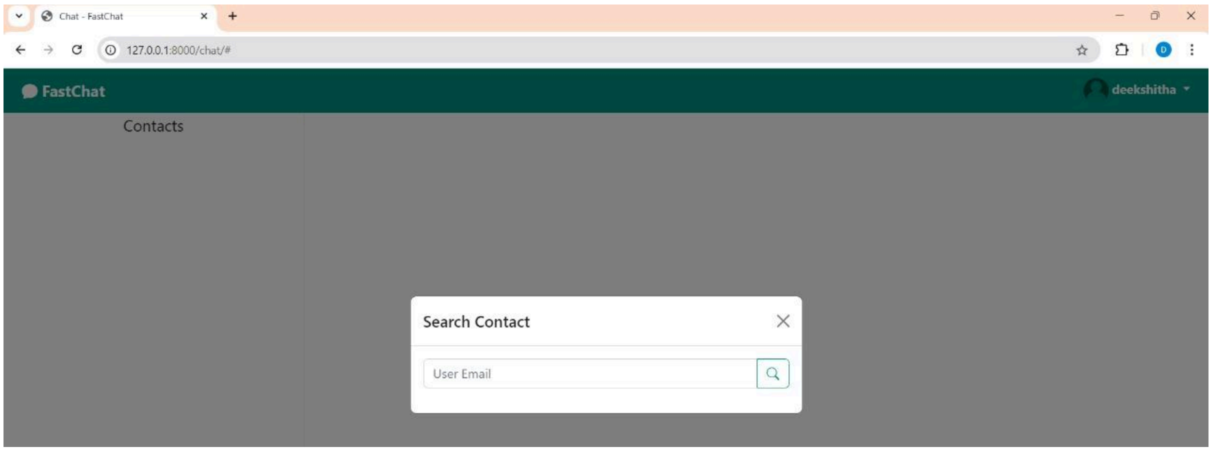
deekshitha

Contacts

Logout

Are you sure you want to log out?

Yes No



9) FUTURE ENHANCEMENTS

- **Media Sharing**: Enable users to share images, videos, and files.
- **Push Notifications**: Implement browser or mobile push notifications for incoming messages.
- **Video and Audio Calling**: Add support for real-time voice and video calls.