Social Media Backend Requirements Document

1. Introduction

The purpose of this document is to define the functional requirements for building a social media backend using Python, FastAPI, SQLAlchemy, and MySQL. This backend will handle user registration, authentication, profile management, posting, viewing posts, and chat functionalities, providing a seamless experience for users.

2. Modules and Functional Requirements

2.1. User Authentication Module

2.1.1. Registration

- Users can register using their email or phone number.

- An OTP (One-Time Password) will be sent to the provided email for verification.

- OTP will be valid for 2 minutes.

- If OTP is not entered within 2 minutes, the user can request a new OTP.

- If the user closes the window before entering the OTP, they must complete the OTP process before logging in.

2.1.2. Login

- Users can log in using their email, phone number, or username.

- Credentials (email/phone number/username and password) must be validated before allowing access.

- Passwords should be securely stored using hashing algorithms (e.g., bcrypt).

2.1.3. Forgot Password

- Users who forget their password can reset it using their email or username.

- An OTP will be sent to the user's registered email for verification.

- OTP must be validated before allowing the user to reset their password.

- OTP will be valid for 2 minutes.

2.1.4. Reset Password

- Users can reset their password by providing the existing password.

- Password reset is only allowed when logged in.

- New password must be validated (e.g., meet complexity requirements).

2.2. User Profile Module

.2.1. Profile Management

Users can manage the following:

-Profile Picture: Users can upload and change their profile picture.

- Username: Users can change their username, but it must be unique.

- About: Users can display and edit the "About" section of their profile.

- Post Count: The total number of posts made by the user should be displayed.

- Post List: A list of all posts uploaded by the user should be visible.

2.3. Visiting Other Profiles

2.3.1. Profile Visibility

Before displaying details, check if the profile is private or public:

- Private Account: Only the username should be visible to others.

- Public Account: Username, about section, posts, likes, and comments should be visible.

2.4. Posting Module

2.4.1. Create Post

Users can create posts:

- Image with an optional caption.

- Text-only content.

2.5. Home Page Module

2.5.1. Display Posts

- Display posts from other users if their account is public.

- Private account posts should not be displayed.

- Allow users to like and comment on posts.
- Show like and comment counts.

2.6. View Post Module

2.6.1. Post Details

- Display users who liked the post.
- Display users who commented on the post.
- Show the caption of the post.

2.7. Deleting Posts

2.7.1. Remove Post

When a user deletes a post, it should be removed from:

- The main post table.
- Any related "liked by" and "commented" tables.

2.8. Chat Module

2.8.1. Chat Functionality

- Store chat messages in plain text.
- Allow sharing of images in chat.
- When a message is deleted by one user, it should also be deleted for the other user.

3. Solutions to Common Social Media Problems

This project provides solutions to common issues users face on social media platforms:

3.1. Privacy Concerns

Users have control over their profile visibility, allowing them to decide whether their account is public or private. Private accounts ensure only the username is visible to others, helping users manage their privacy.

3.2. Security and Authentication

Secure user authentication is ensured through OTP-based verification during registration and password resets. Additionally, passwords are securely hashed, preventing unauthorized access even in case of data breaches.

3.3. Content Moderation

Users can delete their posts, comments, and chats, ensuring that they maintain control over their content. This addresses concerns of unwanted or outdated information staying online.

3.4. Simplified User Interaction

The platform provides easy access to important actions such as posting content, commenting, liking, and messaging. These features are available within a user-friendly interface, making social interactions efficient.

3.5. Account Recovery

The forgot password and OTP-based recovery system ensures that users can regain access to their accounts quickly if they lose their password, minimizing downtime and improving user satisfaction.

4. Database Design

4.1. Tables

Users Table

- ID (Primary Key)
- Full Name
- Email (Unique)
- Phone Number (Unique)
- Username (Unique)
- Password Hash
- Is Verified (Boolean)
- Is Active (Boolean)
- Created At (Datetime)

- Secret Key (String)

About Table

- ID (Primary Key)
- Account Status (Public/Private)
- Bio
- Profile Image (Image Path)
- Created At
- Updated At

OTP Table

- ID (Primary Key)
- User ID (Foreign Key)
- OTP
- OTP Expiration Time

Posts Table

- ID (Primary Key)
- User ID (Foreign Key)
- Caption
- Created At
- Updated At
- Is Active (Boolean)

ImagePostItem Table

- ID (Primary Key)
- Post ID (Foreign Key)
- File Path

- Image ID
- Is Active (Boolean)

Likes Table
- ID (Primary Key)
- Post ID (Foreign Key)
- User ID (Foreign Key)
- Liked Time (Datetime)

Comments Table
- ID (Primary Key)
- Post ID (Foreign Key)
- User ID (Foreign Key)
- Comment Text
- Created At

Chat Table
- ID (Primary Key)
- Sender User ID (Foreign Key)
- Receiver User ID (Foreign Key)
- Message Text
- Created At
- Deleted Status (Boolean)

5. API Endpoints

5.1. Authentication Endpoints

- POST /register – Register a new user.
- POST /verify-otp – Verify OTP during registration.

- POST /resend-otp – Resend the OTP.

- POST /login – Authenticate a user.

- POST /forgot-password – Initiate forgot password process.

- POST /reset-password – Reset the password.

5.2. Profile Endpoints

- GET /profile – Get user profile details.

- PUT /update-name – Update user username and full name.

- PUT /update-contact – Update user contact details.

- PUT /update-account – Update user profile details.

- PUT /profile-image – Update user profile image.

- PUT /update-password – Update user password.

5.3. Post Endpoints

- POST /post – Create a new post.

- POST /add-image/{post_id} – Add an image to a specific post.

- POST /update-image/{post_id} – Update an image in a specific post.

- GET /show-post/{id} – Get details of a post.

- GET /list-post – Get a list of posts from other users.

- DELETE /post/{id} – Delete a post.

5.4. Like and Comment Endpoints

- POST /post/{id}/like – Like a post.

- POST /post/{id}/unlike – Unlike a post.

- GET /recent-like/{post_id} – Get the most recent like by username.

- GET /like-count/{post_id} – Get the like count of a particular post.

- GET /post/{id}/likes – Get a list of users who liked a post.

- POST /post/{id}/comment – Comment on a post.

- POST /post-uncomment/{post_id} – Delete a comment.

- POST /post-editcomment/{post_id} – Update a comment.

- GET /post/{id}/comments – Get a list of comments on a post.

- GET /post/comment-count – Get the total comment count.

5.5. Chat Endpoints

- POST /chat/send – Send a message.

- PATCH /update-message/{chat_id} – Update a message.

- POST /delete-message/{chat_id} – Delete a message.

- GET /chat/{user_id} – Get chat messages with a specific user.

- DELETE /chat/{id} – Delete a message.

6.Project Code Link

Github repo: https://github.com/Surya0067/social_media_task.git

7. Conclusion

This document provides a detailed overview of the functional requirements, database design, API endpoints, and solutions offered by the social media backend system. The next steps include designing the database schema, setting up the development environment, and starting API development.