Author

Mayank Walia
21f1004343
21f1004343@student.onlinedegree.iitm.ac.in
I am a third year BSc Mathematics student, an enthusiastic learner and want to make my career into Data Science.

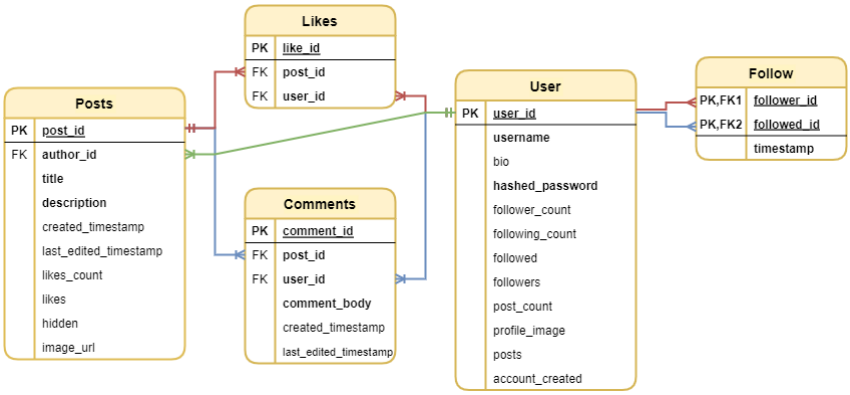
Description

A web application and an API is created for the multiuser social app which allows the user to create and edit their blogs by attaching the images. Users can follow and unfollow others, like and comment on different blogs. Each user has its own profile page and a personalized feed based on his following. Users can find new connections by searching using their usernames.

Technologies used

- **Flask framework** was used for developing the application.
- **Jinja2** was used for templating and HTML generation.
- **Bootstrap 5** was used for styling and designing purposes. It also provide support for icons and in achieving the responsiveness.
- **Flask-SQLAlchemy** and **sqlite3** were used for the database operations.
- **Flask-RESTful** was used for implementing the RESTful APIs.
- **Flask-wtf** and **wtforms** were used for rendering web forms and for validation purposes.
- **Matplotlib** is used for generating the insights on blog posts with the help of bar plots.
- **Flask-Login** was used for providing proper login functionality and session management.
- **csv** is used for exporting the engagement data like count of likes and comments on blogs.
- Various other modules like **os, random, uuid, datetime** and **sys** were also used.

DB Schema Design



- **User table** : **user_id** is *PRIMARY-KEY and AUTO-INCREMENTED*. **username** and **hashed_password** are *NOT NULL*. **follower_count, following_count** and **post_count** are assigned and updated automatically. **followed, followers, posts, likes** and **comments** are obtained using *backref* from the existing relationships and helps in directly obtaining the data.
- **Posts table** : **post_id** is *PRIMARY-KEY and AUTO-INCREMENTED*, **title** and **description** are *NOT NULL*. **hidden** is by default *FALSE.*
- **Comments table : comment_id** is *AUTO-INCREMENTED*. **comment_body** is *NOT NULL*. **post_id** and **user_id** are the *FOREIGN KEYS* from the Posts and Users table respectively.
- **Follow table** : **follower_id** and **followed_id** are *FOREIGN KEY* from Users Model.

- **Likes table** : **like_id** (*AUTOINCREMENT*), **post_id** (*FOREIGN KEY*) and **user_id** (*FOREIGN KEY*)
- All **timestamps** in the models are *NOT NULL* and are updated automatically

There are multiple users and each user can create many blogs. There is **One-to-Many Relationship** between **User** and **Posts**. Each user can have many followers and can follow many users, therefore a **separate association table** called **Follow** is created to manage **self-referential Many-to-Many Relationship** existing on the User table. A user can give likes and write comments on various posts. A post can have several likes and comments from different users. Therefore, **One-to-Many Relationships** exists between *User – Likes, User – Comments, Posts – Likes* and *Posts – Comments.*

API Design

There are 3 resources created for the API using the **Resource** class from **Flask-RESTful**. They can handle *HTTP GET/PUT/DELETE/POST* or *CRUD* requests and their responses are in *JSON* format.

- **UserApi** - Resource for CRUD on User model.
- **BlogApi** - Resource for CRUD on Blog model.
- **FeedApi** - Resource for getting all the blog posts that will be shown a User feed.

Endpoints are:

- **/api/user/{user_id}** : For reading, updating and deleting the user.
- **/api/user** : For creating a new user.
- **/api/blog/{blog_id}** : For reading, updating and deleting the blog.
- **/api/user/{user_id}/blog** : For creating a new blog for a particular user.
- **/api/feed/{user_id}** : For getting the collection of all posts present on a user feed.

Architecture and Features

The project makes use of blueprints so that it can be structured properly into different folders and files. It improves readability and maintainability of the code. Separate blueprints for posts and users are registered in the main app. The root folder contains **app.py, api.py** (contains code for API)**, configuration.py** (contains settings and configuration values) and **a folder** named **application** (contains all the application code). The application folder contains 4 subfolders: **posts, users, static** and **templates** and files **__init__.py** (for creating flask app, initializing and creating database), **models.py** (contains definitions of models) and **utility.py** (contains helper functions). All the controllers related to the *users* are contained in *views.py* inside *users subfolder* and controllers related to *posts* are present in *post_views.py* inside *post*s subfolder. The static folder contains all the *images, CSS and CSV* files. The templates folders contains *HTML files and macros*.

Some additional features are:

- User can archive or unarchive their posts.
- Flash messages indicating success/failure are displayed on performing some task.
- Users can export the engagement data on their blog posts as CSV file.
- Users can view the insights on blogs using different charts and infographics.
- Passwords are stored as hashes and users can also update their password.
- Users can update or remove profile picture and image uploaded on blog afterwards also.

Video

https://drive.google.com/file/d/1lbaT7bqmBP84ZWX_cp1HP-SGOERkFr1r/view?usp=share_link