Author

P V Sri Harsha
21f20000990
21f2000990@ds.study.iitm.ac.in
I am an undergraduate student in Civil Engineering at BITS Hyderabad. Through this BSc program, I hope to improve my Data Science and Programming skills and build a career as a Data Scientist.

Description

The application is blog manager where people can post and read blogs. We can also follow/unfollow our favourite blog writers. Also implemented backend jobs for monthly engagement report and daily reminders.

Technologies used

Celery[redis]: A distributed task queue used for asynchronous processing.
Redis: An in-memory data structure store used as a database, cache, and message broker.
Flask-security-too: An extension for Flask that provides authentication, authorization, and password hashing.
Flask_restful: A Flask extension for building REST APIs.
Flask_cors: A Flask extension for handling Cross-Origin Resource Sharing (CORS).
Flask_caching: A Flask extension for caching responses.
Bleach: An HTML sanitizing library used for preventing cross-site scripting attacks.
Bcrypt: A password-hashing function used for password security.
Pandas: A data analysis library used for manipulating and analyzing tabular data.
A few other minor libraries were also used

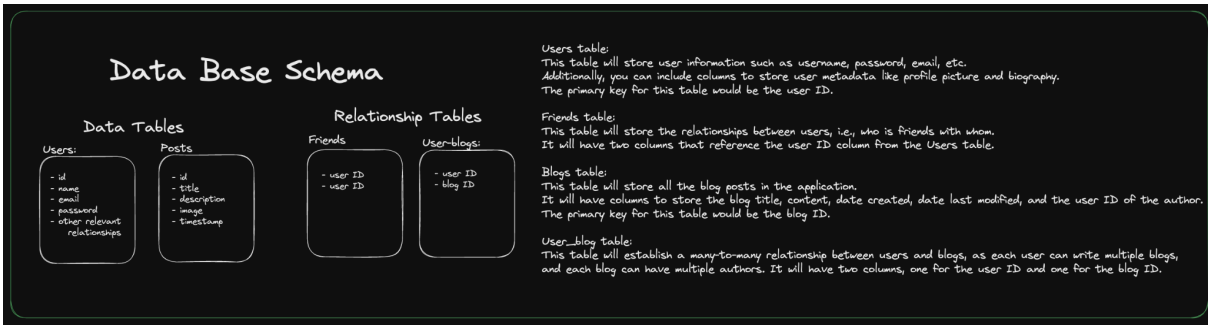
DB Schema Design

The models.py file defines the database schema for the Flask web application. It uses the SQLAlchemy library to define the tables, columns, and relationships between them.

- The User model represents users of the application and is configured to work with Flask-Security, which provides authentication and authorization features. It has columns for id, email, username, password, active, and confirmed_at. It also has a many-to-many relationship with itself through the Friend model, which allows users to follow each other.
- The Blog model represents blog posts made by users. It has columns for id, title, caption, image_url, created_at, updated_at, and user_id. It also has a relationship with the User model, which allows the user who created the post to be easily accessed.

- The UserBlog model represents the relationship between users and the blog posts they have written. It has columns for id, user_id, and blog_id. It is used to enforce the constraint that a user can only write one blog per post.
- The Role model represents roles that users can have in the application. It has columns for id, name, and description.
- The UserRoles model represents the relationship between users and roles. It has columns for id, user_id, and role_id.
- The Token model represents JWT tokens that are used for authentication. It has columns for id, jti, token_type, user_id, and created_at.

models.py defines the database schema for the application and provides a way to easily interact with the database using SQLAlchemy.



API Design

<<Briefly describe what elements you have created an API for and how it was implemented. The YAML file for API to be submitted separately>>

Below are the API's used for the application:

- CurrentUserAPI resource returns the details of the currently authenticated user in the form of a JSON object.
- SearchAPI resource allows users to search for other users based on their username. It returns a list of matching user objects in the form of a JSON array.
- FollowAPI and UnfollowAPI resources allow users to follow and unfollow other users respectively. They check for authentication, user existence, and whether the user is already being followed. If everything is okay, the resource updates the database to reflect the new follow status.
- BlogsAPI resource returns a list of blogs authored by the user and their followers in descending order of update time. It also allows users to update and delete their own blogs.
- ExportCSVAPI resource takes a list of blog objects and exports them to a CSV file. It first creates a directory named after the user ID in the Downloads folder and then creates a CSV file containing the blog data in the directory. The task of exporting to CSV is executed asynchronously using Celery.

Architecture and Features

The project is organised using a typical Flask application structure. The controllers, also known as views, are located in the application/controllers.py file and are responsible for handling the incoming HTTP requests from clients. The models are defined in application/models.py and represent the database tables and their relationships. The app/templates directory contains the Jinja2 templates that are used to render HTML pages. The app/static directory contains static files such as CSS, JavaScript, and images. The config.py file contains configuration variables for the application, such as the database URI and secret key. Overall, the project follows a modular and organized structure that separates different concerns and makes the application easier to maintain and scale.

Features Implemented: Implemented all the core requirements:

- User signup and login
- User profile view with basic stats
- Blog Post Management
- Search and Follow / Unfollow Others
- User's Feed
- Backend Jobs
- Backend Performance

Video

https://drive.google.com/file/d/10ztfRWKf2PC0sCm3n1XObdz3QgofSj7m/view?usp=sharing