Blog-Lite v2 Report

Author:

- **Name**: Sherry Shajimon P T
- **Roll No**: 21f3001449
- **Email**: 21f3001449@student.onlinedegree.iitm.ac.in
- **About Me**: I graduated with a Bachelor of Arts degree in Philosophy from Christ University in 2021 and was also the top of my class. Currently, I am pursuing a Bachelor of Science in Data Science from IIT-Madras and am at the diploma level.

Description:

Blog-Lite is a web-based application made with Vue3 and Python-Flask. It enables multiple users to share their ideas in the form of text-based blog entries accompanied by a suitable image. It facilitates users to follow and search other bloggers, and express their views by commenting on their posts. Additionally, a Rest API has been developed adhering to the OpenAPI Specifications, which permits access to the app's database for limited data modification purposes.

Technologies Used:

- **Python:** responsible for developing the controllers and serving as the host programming language for the app
- **Vue.js:** used to develop the front-end of the app
- **HTML:** responsible for developing the required Vue components and templates
- **CSS:** responsible for styling the web-pages
- **Bootstrap:** used to make the front-end appealing and easy to navigate
- **SQLite:** serves as the database for the app
- **Flask:** serves as the web-framework for the app
- **Flask-Restful:** used to develop the RESTful API for the app
- **Flask-SQLAlchemy:** used to access and modify the app's SQLite database
- **Flask-Celery**: used for asynchronous background jobs at the backend
- **Flask-Caching**: used for caching API outputs and increasing performance
- **Redis**: used as an in-memory database for the API cache and as a message broker for celery
- **Swagger OpenAPI:** used to create the documentation for the API developed for the app
- **Seaborn:** used to create the various required charts
- **Git:** responsible for version control

Database Schema:

The database has four tables and the schema is as follows:

User Table	Post Table
● **Roll** (Integer): Primary Key, Auto Increment ● **Username** (String): Unique, Not Null ● **Email** (String): Unique, Not Null ● **Password** (String): Unique, Not Null ● **Img** (String): Not Null, Default 0 ● **PDF** (Integer): Not Null, Default 0	● **Roll** (Integer): Primary Key, Auto Increment ● **Author** (String): Foreign Key (User.roll), Not Null ● **Img** (String): Not Null ● **Text** (String): Not Null ● **Date** (String): Not Null ● **Title** (String): Not Null ● **Views** (Integer): Not Null, Default 0 ● **Likes** (Integer): Not Null, Default 0

Comment Table	Follow Table
● **Roll** (Integer): Primary Key, Auto Increment ● **Post** (Integer): Foreign Key (Post.roll), Not Null ● **Author** (String): Foreign Key (User.username), Not Null ● **Comment** (String): Not Null	● **Roll** (Integer): Primary Key, Auto Increment ● **Following** (Integer): Foreign Key (User.username), Not Null ● **Follower** (String): Foreign Key (User.username), Not Null

API Design:

The Flask-Restful library for Python was used to create a RESTful API adhering to the OpenAPI Specifications. All database tables have CRUD operations available through the API. Authentication tokens are used for specific requests that require them. These tokens can only be obtained from the user's account page when signed in. For further details, please refer to the openapi.yaml file.

Architecture and Features:

The architecture of Blog-lite follows a client-server model, where Vue.js serves as the front-end framework and Python-Flask as the back-end framework. Vue.js handles the presentation layer and manages user interactions through its MVVM architecture, while Python-Flask handles the server-side logic, such as HTTP requests and responses, asynchronous tasks, and database interactions.

The features of the application are as follows:

- **User authentication**: Signup and Login
- **User profile**: View own posts, followers, and follows
- **Explore other users**: View their posts, followers, and follows
- **User-specific API tokens**: Generate tokens to use user-specific requests
- **Post analytics**: View clicks, likes, and comments on posts
- **Data export**: Download user's posts and analytics as a CSV file
- **Social features**: Search, follow, and unfollow other users
- **Personalized feed**: View posts from followed users
- **Content management**: Create, view, edit, and delete posts
- **Account management**: Create, view, edit, and delete user accounts
- **User feedback**: Comment and like posts to express opinions
- **RESTful API**: API available for posts, users, comments, and follows
- **Import blogs:** Ability to import blogs in bulk
- **Export blogs**: Export blog content as PDFs
- **Reminders**: Receive daily reminders to post
- **Monthly engagement report**: Receive a report as an email or PDF summarizing engagement for the month
- **Mobile Interface**: Adaptive Interface for devices of various sizes and shapes
- **PWA Installation**: The app can be installed on various devices as an app for offline browsing

Video:

For the video, click here!