Author

Name: Nikita Sharma

Roll number: 21f1000637

Email: 21f1000637@ds.study.iitm.ac.in

Graduated from Delhi University last year as Software Developer and currently interning as a Data Scientist at Syngenta, Pune.

Description

The main aim of this project is to create a blog application using Flask, HTML, SQLite, etc. CRUD on blogs and user profiles along with alert messages and the ability to follow/unfollow other users, proper login system for improved security has been implemented in the project. The user will get daily reminder in the email provided and a detailed monthly report of his/her status of blogs. A CSV file will be sent to the email, if the user wants to export the data on the application.

Technologies used

Here are the technologies used in this project: Python, HTML/CSS/JS, Bootstrap, Flask, Flask-Login, SQLite, Flask-SQLAlchemy, Flask-caching, Flask-restful, weasyprint, Redis, celery, Extensions (like: request, os).

- Python is the core programming language used.
- Flask is the main framework used for the Web-app.
- Flask-Login is used for managing multiple user login and keeping a session alive.
- Flask-SQLAlchemy is the SQL toolkit used to connect with the database file.
- Flask-caching for making cache and improving performance
- Flask-restful for creating REST architecture based API
- Weasyprint for creating PDF's from HTML
- Celery for back-end asynchronous jobs
- Redis for caching

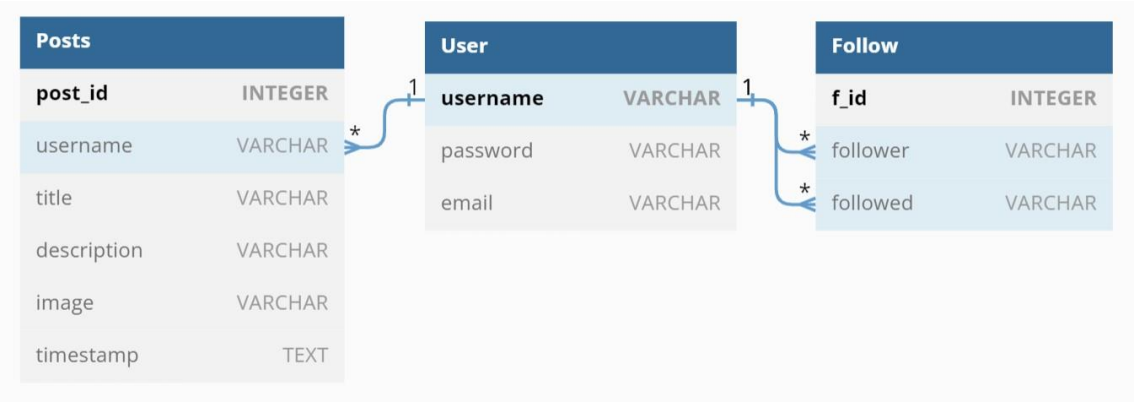
API Design

I have implemented Get, Post, Put and Delete for Users Table as api_user class and for Posts table as api_post class for performing CRUD operations. Get and Post for Follow table as api_follow because I haven't included the feature to remove your followers and the delete operation is taking place within post only.

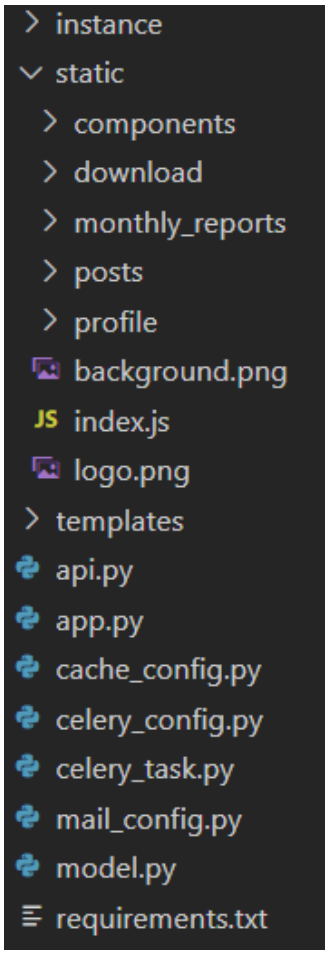
DB Schema Design

Table Name	Columns	Description	Constraints
Users	username	Username of the user	String, Unique, Primary_key
	email	Email ID of the user	String, Unique
	password	Password of the user	String
Posts	id	Unique id for each blog	Integer, Primary_key, Autocrement
	image	Name by which each blog image will be stored	String, Unique
	username	Username of the user who created the post	String, Foreign_key
	title	Title of the post	String
	description	Description of the post	String
	timestamp	Date & time when the post was created	String
Followers	f_id	Unique id for each follow/following	Integer, Primary_key, Autocrement
	follower	Users who are following current user	String, Foreign_key
	followed	Users who are being followed by current user.	String, Foreign_key

ER Diagram



Architecture and Features



ARCHITECTURE

The layout of the project is shown on the right and here's what they have:

- static - which holds the JS, CSS and image files.
- templates - which holds all the HTML files.

Then, I have made 7 python files:

- api.py file has all the code related to APIs
- app.py file has the code for the main code to start the Web App
- models.py file has the all the code related to the different models.
- cache_config.py, celery_config.py, mail_config.py files have the configurations for the respective parts
- celery_task.py file has all the asynchronous tasks to be done by celery
- README.md has the instructions on how to start the Flask Web App.
- project_db.sqlite3 is the database file.
- Requirement.txt has the required packages name

FEATURES

- Multi-user app where user can post blogs with image, title and description
- Users can access the Web App even after closing and re-opening the browser and it would not take the user to the login page unless the user have logged out or have accessed the login page explicitly.
- Interactive page which shows an alert when we delete a blog or user or if the password given does not match the re-typed password during new registration.
- CRUD operations on tracker - Create, Read, Update, Delete.
- CRUD operations on log - Create, Read, Update, Delete.
- Daily reminder and monthly report will be sent to the user's email-id.
- Sending an email to the user's email-id containing a csv file that consists of the user's posts.

Video Link

Link to video