Author

SATYAM AWASTHI
22dp1000096
22dp1000096@student.onlinedegree.iitm.ac.in
I'm a DAD student and currently pursuing my diploma in programming from IIT Madras.
This is the Project Documentation for MAD-2 project.

Description

We have to create a simple BlogLite App depicting the core functionality of a Social Platform.
It is a multi-user app and User have to follow other users in order to get their created blogs in the feed. User can create any number of blogs and choose whether to archive it or not. Archive blogs will not be shown to other users. The Blogs could be edited and deleted. User can view his stats on MyProfile page. There is one user triggered backend job which exports the zip file and two scheduled backend jobs which send alerts on daily and monthly basis.

Technologies used

Vue.js: for building a declarative and component-based model at front-end for user interface.

Vue Router: for making my Web app like a Single Page Application at front-end.

Flask: for creating web Framework at backend for BlogLite Application

Flask-SQLAlchemy: for defining models and interaction with database.

flask-security-too: for protecting API with token authentication and current user tracking.

flask-restful: for building REST API for our BlogLite application.

flask_bcrypt: for hashing of passwords

Flask-Caching: to add caching support at various endpoints of REST API to boost performance.

matplotlib & pyplot: to generate the pie & bar graph for engagement report.

flask_cors: for handling Cross Origin Resource Sharing (CORS)

redis: to build an in memory data structure for caching and Message broker for celery.

celery[redis]: to build a Celery system and distribute tasks across workers and schedule task

email_validator: to validate the syntax of email provided by user during sign up.

Flask-Mail: to set up SMTP and send mails to the users.

weasyprint: to convert the engagement report to pdf from html.

DB Schema Design

Assosciation Table: roles_users

This table is used as a helper table for defining many to many relationship between user and role.

Column	Type	Constraints
user_id	INTEGER	FOREIGN KEY references user.id
role_id	INTEGER	FOREIGN KEY references role.id

Table Name: user

This table is used to store the details of users

Column	Type	Constraints
id	INTEGER	PRIMARY KEY
username	VARCHAR(20)	NOT NULL, UNIQUE
password	VARCHAR	NOT NULL
full_name	VARCHAR(30)	NOT NULL
email	VARCHAR	NOT NULL, UNIQUE
created_at	DATETIME	NOT NULL
dob	DATETIME	NOT NULL
profile_photo	VARCHAR	---
notifications	VARCHAR	---
last_login_at	DATETIME	---
report_format	VARCHAR	NOT NULL
active	BOOLEAN	---
fs_uniquifier	VARCHAR	NOT NULL, UNIQUE
roles	RELATIONSHIP (MANY TO MANY)	MAPPING CONSTRAINT with role table
posts	RELATIONSHIP (ONE TO MANY)	MAPPING CONSTRAINT with blog table
following	RELATIONSHIP (ONE TO MANY)	MAPPING CONSTRAINT with blog table

Table Name: role

This table is used to store the details related to roles of users

Column	Type	Constraints
id	INTEGER	PRIMARY KEY
name	VARCHAR	UNIQUE
description	VARCHAR	---

Table Name: blog

This table is used to store the details of blogs created by user.

Column	Type	Constraints
post_id	INTEGER	PRIMARY KEY
title	VARCHAR	NOT NULL
caption	VARCHAR	NOT NULL
image_url	VARCHAR	NOT NULL
time_stamp	DATETIME	NOT NULL
user_id	INTEGER	FOREIGN KEY references user.id, NOT NULL
username	VARCHAR	FOREIGN KEY references user.username, NOT NULL
archive_switch	BOOLEAN	NOT NULL
updated	BOOLEAN	---
report_format	VARCHAR	NOT NULL
likes	RELATIONSHIP (ONE YO MANY)	MAPPING CONSTRAINT with like table
comments	RELATIONSHIP (ONE YO MANY)	MAPPING CONSTRAINT with comment table

Table Name: followmap

This table is used to store the connection between users.

Column	Type	Constraints
map_id	INTEGER	PRIMARY KEY
follower_id	INTEGER	FOREIGN KEY references user.id, NOT NULL
following_id	INTEGER	FOREIGN KEY references user.id, NOT NULL

Table Name: like

This table is used to store the details of likes received on a blog

Column	Type	Constraints
like_id	INTEGER	PRIMARY KEY
like_username	VARCHAR	FOREIGN KEY references user.username, NOT NULL
post_id	INTEGER	FOREIGN KEY references blog.post_id, NOT NULL

Table Name: comment

This table is used to store the details of comments received on a blog

Column	Type	Constraints
comment_id	INTEGER	PRIMARY KEY
comment_username	VARCHAR	FOREIGN KEY references user.username, NOT NULL
post_id	INTEGER	FOREIGN KEY references blog.post_id, NOT NULL
content	VARCHAR	NOT NULL

Table Name: inbox

This table is used for generating the chatbox and providing inbox feature. It is used as a container for messages between a pair of users.

Column	Type	Constraints
inbox_id	INTEGER	PRIMARY KEY
sender_id	INTEGER	FOREIGN KEY references user.username, NOT NULL
sender_username	VARCHAR	FOREIGN KEY references user.username, NOT NULL
reciever_id	VARCHAR	FOREIGN KEY references user.username, NOT NULL
reciever_username	DATETIME	FOREIGN KEY references user.username, NOT NULL
blocked	INTEGER	NOT NULL
messages	RELATIONSHIP (ONE YO MANY)	MAPPING CONSTRAINT with message table

Table Name: message

This table is used to store the messages sent and received between a pair of users.

Column	Type	Constraints
message_id	INTEGER	PRIMARY KEY
message	VARCHAR	NOT NULL
sent	BOOLEAN	NOT NULL
time_stamp	DATETIME	NOT NULL
inbox_id	INTEGER	FOREIGN KEY references inbox.inbox_id, NOT NULL

API Design

- to perform CRUD operations on users.
- To perform CRUD operation blogs created by users
- To establish the connections between users like the relation of follow and following
- To trigger a backend job which exports the csv file
- To generate inbox chat feature.

Architecture and Features

Architecture:

/22dp1000096_MAD2 is the root folder which will consist of several folders, readme.md, requirements.txt, project_doc_link.txt files and main.py file for running the app.

- /22dp1000096_MAD2/application folder will consist all the business logic files and config.py file.
    - /22dp1000096_MAD2/application/controller will contain api.py and controllers.py file.
    - /22dp1000096_MAD2/application/data will contain models.py file.
    - /22dp1000096_MAD2/application/jobs will contain tasks.py and workers.py file.
    - /22dp1000096_MAD2/application/utils will contain security.py file.
- /22dp1000096_MAD2/db_directory folder will consist of test.db sqlite file
- /22dp1000096_MAD2/docs folder will consist of 22dp1000096_report.pdf and RestApi.yaml file.
- /22dp1000096_MAD2/static folder will consist of javascript files and images.
- /22dp1000096_MAD2/templates folder will consist of all the HTML templates

Features:

Default Features:

- User can sign up by providing credentials. Email should be deliverable and valid.
- User can login and run the application Token Authentication is used for security.
- User can post multiple times and also can make the post archive if they want to restrict its view. A post will consist of title, caption, archive switch and the image file uploaded by user.
- A user can follow other users by going to the Search tab and typing their initials. The posts of all the users which current user follows will be shown in his feed.
- Order of posts shown in the user feed is determined by their timestamp.
- User can exports his posts in the form of zip file. The zip file will contain a csv file and all the images of posts created by user. This is handled by a user trigerred backend job.
- Daily reminder is sent to the users as an email. All the user which have not logged in or posted anything will get a reminder.
- Monthly engagement report is sent to all the users as an attachment in the email body.
- Caching is provided in the application in order to boost the performance.

Additional Features:

- The web application is made as a Single Page Application using Vue-Router.
- User can choose the format of monthly engagement report between HTML and PDF
- Like and Comment feature is added in the post. User can also view the total likes and comments and the user which have liked/commented on the post.
- Notifications feature is added to view the notifications for the user.
- Inbox feature is added to provide the Chat feature to the app. User can search the users which he want to message and add them in the inbox.
- Validation is provided at frontend and backend

Video

https://drive.google.com/file/d/1FcvibuceMBURUh91-m_nuJmQlLcn_SQL/view?usp=share_link