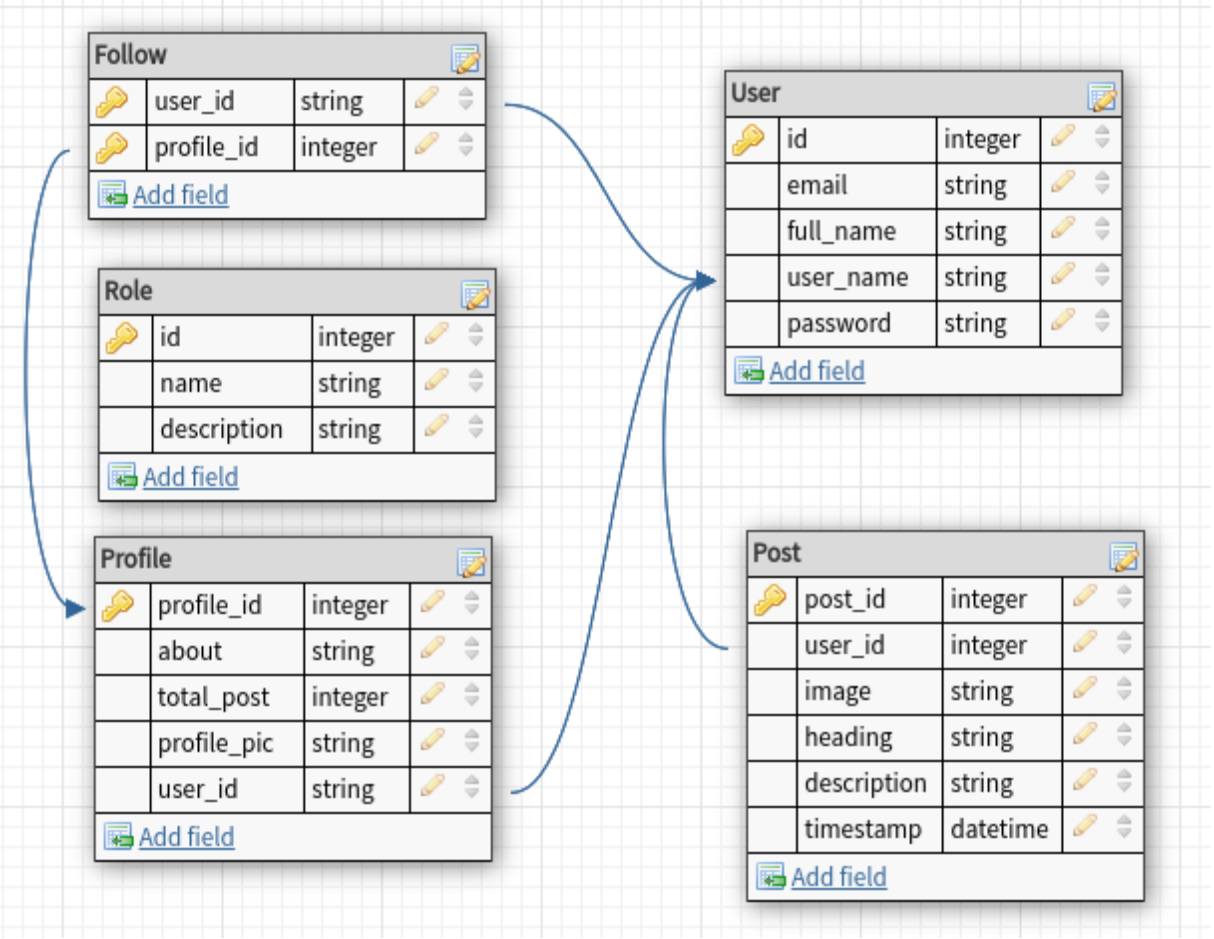
Project Report: BlogLook

Name: Aditi Krishana
Roll: 21f1004270
Email: 21f1004270@ds.study.iitm.ac.in
I am Aditi, a Postgraduate student of Chemistry, trying to gain the basics of data science.

Description

I have used the Flask framework to create the BlogLook app, with authentication fulfilled using flask-security and different requirements of the app accomplished using API built using flask-restful. It presents the engagement graph of the user in posting the blogs, generated using the matplotlib library of python., generated using the matplotlib library of python. The front end is based on VueJS CLI with CSS and Bootstrap, based on the user's click for aesthetics. The reminders and exports are aimed at using smtplib.

DB Schema Design



It is a web-app-based project for public use in which multiple users can log in to create their own profiles and add their own posts, follow, and unfollow other users, with token-based authentication. The project requirement requires secure login for the registered users which is aimed using flask-security. The sign-up page takes the user_name and email to be unique, the id is the primary key and the full name and password are taken in the string format from the User table. After the user enters their account, the sidebar gives the option to go to the profile page and edit their profile, user_id is the foreign key and profile_id is the primary key, total_post is an integer, about and profile_pic is taken as the string in the Profile table. The third option in the sidebar, add_blogs, allows the user to post the blog, in which post_id is taken as the primary key and user_id as the foreign key, image, heading, and description taken in string format in the Post table. Further, the graph is plotted using the total no.of posts done by the particular user which can be seen using the

engagement button made available on the profile page. The search option enables the user to find other usernames and follow or unfollow them.

Technologies used

- **os**: For giving the path of the database and folders containing images
- **Flask**: For creating the application
- **Request:** To get data and methods from forms
- **Flask_sqlalchemy**- SQLAlchemy: For creating models in the database for the app
- **Flask_security:** for ensuring authentication of the application
- **Flask_restful:** for creating APIs for different tasks
- **Flask_caching:** for caching purposes to ensure performance
- **VueJS CLI:** for frontend and UI
- **CSS:** To add styling effects
- **Bootstrap:** To align web pages as required and create selections
- **matplotlib:** For creating graphs
- **DateTime:** To use dates as data
- **celery:** for task and jobs management
- **redis:** for caching
- **smtplib:** for sending emails of reminders and exports

Architecture and Features

The code for the BlogLook app is organized based on the usage and update required. The code for the app can be viewed inside the project folder with the name app.py python file. The app named BlogLook, on starting gives a home page where we have a link for the signup and login, after signing up we can go to the login page which requires the correct combination of user_name and password to log in, which further takes us to the feed where the latest blogs of other users are been shown who is been followed. Further, then in the sidebar of the web page, the profile button redirects to the user's profile, where the stats of the user i.e. total no. of posts, followers, and following is shown as well as the edit profile button are made available for updating the profile image, about and the full name of the user, post engagement button is made available for showing the graph of the post by the user, below the posts by the user is also shown which can be edited, deleted and export button is made available for exporting the posts. Every post's elements can also be exported using the button provided for the purpose. Also, the user gets daily and monthly reminders via mail and exported files in the mail The UI for the app is based on VueJS CLI, the code for which is inside the frontend folder. The different views are based on different frontend routes, the components for which are inside the components folder of the src folder and routes are mentioned in the routes folder of the same as components. The backend is organized such that we have app.py for running the app in which we have defined different URL's for different purposes and the purpose is defined inside the api.py of the application folder. The exports and reminders are defined inside tasks.py which are taken where required. Reminders and Exporting jobs are ensured via celery code which is mentioned in clry.py and used as required in the app.py(for reminders) and tasks.py(for exporting). We also use flask-caching in api.py for caching. The Add_Blog option in the sidebar gives the user the ability to post the blog. The Search button enables the user to search other users with their user_name, in which the option of follow and unfollow is provided. In the sidebar, the following and followers have a link that redirects to show the no. of followers and following of the particular user. Deleting the user's account option, deleting the user's account and redirecting it to the home page, and the Logout option just redirects the user to the login page. Sending reminders and exports over mail is aimed using the code in emailgen.py in which we use smtplib of python to send emails and required files are attached using MIMEMultipart.The app is served at `http://127.0.0.1:5000` and the front UI of the app can be viewed at `http://localhost:8080/`.

Video

Project Video Link