Author

Name: Devansh Singh Parmar

Roll no: 21f1001594

Official IITM student mail id: [**21f1001594@ds.study.iitm.ac.in**](https://app.onlinedegree.iitm.ac.in/student_dashboard/profile)

Description

A blogging type social media website that lets users create a profile and lets users share blogs that includes text and image. The site allows users follow other users and view blogs posted by these oher users

Technologies used

1. Vue.js: Frontend framework to render webpages of my site and to make the site reactive
2. **Flask** (micro web framework) : Used as backend framework
3. Flask extensions like:
   1. **Render\_template, requests**: to load HTML templates on the browser and get requests types
   2. **flask\_restful**: to create and integrate APIs with the web app
   3. **Make response**: to return error responses
   4. **flask-cors:** For cross server requests
   5. **request:**  to make standar api requests
   6. **Flask-JWT-Extended:**  for token based authentication
   7. **flask\_sqlalchemy:** to connect the framework with database
4. **Sqlite** : database used to store user and blogs information
5. **Sqlalchemy** : python toolkit to interact with the sqlite database
6. **Celery:**  used to schedule a task for reminder mail
7. **Reddis:** alternate database for celery
8. **Smtplib:**  to connect and send mails to server from flask
9. **Mailhog:** mail testing tool to send my dummy test mails
10. **Jwt:** for token based authentication
11. **Weasyprint:** to create pdfs from html
12. **Jinja**: used as template to create pdf reports
13. **Requests:** to make get/put/post/delete requests to api
14. **BOOTSTRAP :** for styling purpose

DB Schema Design

* Created a **users** tabe to store every blog’s information. The blogs table include the following collumns:
  + Uid = primary\_key of the table. Each user has a unique Uid associated with it
  + username = text field to store a unique name associated with users
    - this field is used to differentiate between users displayed on webpage if they have the same name
  + name = any name user would like to use. This field does not have to be unique
  + password = text field to store the password of the user profile
  + about = text field to store some bio about the user. Initialised to none but can updated from edit profile page
  + profile\_pic\_url = text field to store url of the image stored on server
    - this will be used to display the profile pic on user profile by retrieving the image using this url
  + posts = integer field that keeps track of number of posts posted by user
    - used to display number of posts on profile page instead of running a query every time
  + n\_following = integer field that keeps track of number of people that user is following
    - used to display number of following on profile page instead of running a query every time
  + n\_followers = integer field that keeps track of number of people that follow user
    - used to display number of followers on profile page instead of running a query every time
  + email: text field that stores email of the user
  + last\_activity: that stores last date-time of user’s login
* Created a **blogs** tabe to store every blogs information. The blogs table include the following collumns:
  + blog\_id = primary\_key of the table.each blog has a unique id.
  + user\_id = foreign key to Uid of user who has posted the blog
    - used to retrieve who posted this particular blog
  + title = text for title of the blog
  + description = text for description of the blog
  + image\_url = text column used to store url of an image associated with the blog
    - whenever a blog has to be displayed its image is retrieved using this url
  + time\_stamp = a date time column that automaticaly adds the time and date of posting of the blog
    - this data is used to order the blogs from latest to oldest in users feed
  + likes
  + n\_comments: number of comments on the post
  + n\_views: everytime a post is fetched to a user’s screen
* Created **follows** table that maintains a relationship between multiple users following each other. This table has the following columns:
  + has follower and following collumn
  + **follower** collumn ids are following (verb) **following** column ids
* Created **comments** table that maintains a relationship between multiple blogs and coments on those blogs. This table has the following columns:
  + Comment\_id
  + Blog\_id
  + Poster\_user\_id
  + comment

API Design

* FOR FEED
  + **FeedBlogsApi:**  to retrieve blogs posted by people followed by the logged in user
    - When this api is called it first retrieves the list of all users the current logged in user is following. It then gets all the blogs posted by these users and sorts them in ascending order of how long it was posted (latest to oldest)
    - It then retruns a list of each blog data along with some user data (each blog’s user)
* For Browse page: **BrowseBlogsAPI**:
  + - Used to retrieve random blogs from database
* ForUser: **ProfileDataAPI**
  + **Create**: On a post request to the above api data is taken from html form on **SIGNUP** page which is then sent to the above api in json format
  + The above api vaildates the information and creates a database entity for the following user and adds it to the database
  + **Read**: on get request (parameter = user id) it retrieves all the data for the given user
    - This api call is used in my profile page
  + **UPDATE:** on a put request(parameter = user id) it updates user data in the database
    - this api call is used in edit profile page
  + **DELETE:** on a delete request to this api (parameter = user id) the profile is deleted from the database
* ForBLOGS: **BlogsApi**
  + **Create**: On a post request to the above api data is taken from html form on **share a post**  page which is then sent to the above api in json format
  + The above api vaildates the information and creates a database entity for the following blog and adds it to the database
  + **Read**: on get request (parameter = blog id) it retrieves all the data for the given blog
  + **UPDATE:** on a put request(parameter = blog id) it updates blog data in the database
    - this api call is used in edit blog page
  + **DELETE:** on a delete request to this api (parameter = blog id) the blog is deleted from the database
* Forfollowing: **FollowStatusApi**
  + This api gets and updates data for when a user clicks on the follow button of another user
* Forsearch: **SearchUsersApi**
  + This api gets usernames similar to what is written on the search box
* **FollowersListApi & FollowingListApi**
  + This api gets list of followers or followings of a user

Architecture and Features

Architecture:

The project is organized into one main directory. Under the main directory, there are 3 main folders: backend, fronent, and docs.

Backend folder includes Python files include an main.py file which is the main file.There is an APIs.py file that includes all the APIs.A model.py file to make a suitable database model for the app which includes several database tables.

The frontend folder includes javascript files for vue.js.  
the frontend folder will include views folder which has templates for all webpages for the websites.

The docs folder includes a a project report, and a checklist

Features:

* SHARE A POST:
  + Users can create and share a blog post by clicking on share a post button on navbar
  + Users will have to provide a title , description and an image to post the blog.
* FEED: webpage that displays blogs poted by people followed by logged in users
  + Blogs are displayed from latest to oldest order , with respect to date-time.
  + users can view each blogpost in a separate webpage for enlarged form of blog
  + users can see who posted a specific blog as username is displayed above each blog.
  + Users can click on this username to go to that particular username’s profile
* User Engagement:
  + Users can like and comment and view the comments as well
* USER PROFILE:
  + Ability to view some other users profile
  + This page displays number of posts of that particular user along with number of followers and number of followings
  + One can also view who the particular user is following or is followed by through clicking on following or followers respectively
  + The user profile page also displays some information about that particular user in an about section
  + The user profile also displays some recent posts posted by that particular user
  + Logged in user can follow or unfollow the particular user by clicking on the follow/unfllow button provided on user profile
* SEARCH
  + Users can search for other users by typing their username in the search bar
  + This also works if some part of username is submitted instead of the full username
* MY PROFILE
  + Webpage that displays logged in user’s profile which has all functionalities of user profile page without the follow/ unfollow button\
  + One can view their own blog posts in my profile page
  + An **edit profile** button that allows users to edit their name, username, password. Or add new profile pic or information for the about section
  + Users can also **delete profile** from edit profile page
* EDIT BLOGS
  + One can **edit blogs** from my profile page by clicking on edit text under one of the blog
  + On edit blog page user can change title , description or an image posted for the blog
  + User can also **delete blog** from edit blog page

Video