Author

Isha Sachin Shukla
22f1001359
22f1001359@student.onlinedegree.iitm.ac.in
I'm a third year student pursuing a degree in Computer Science at Cummins College of Engineering alongside this degree. I'm an enthusiastic learner, and a hard-working person. I have a creative mind and I'm always up for new challenges.

Description

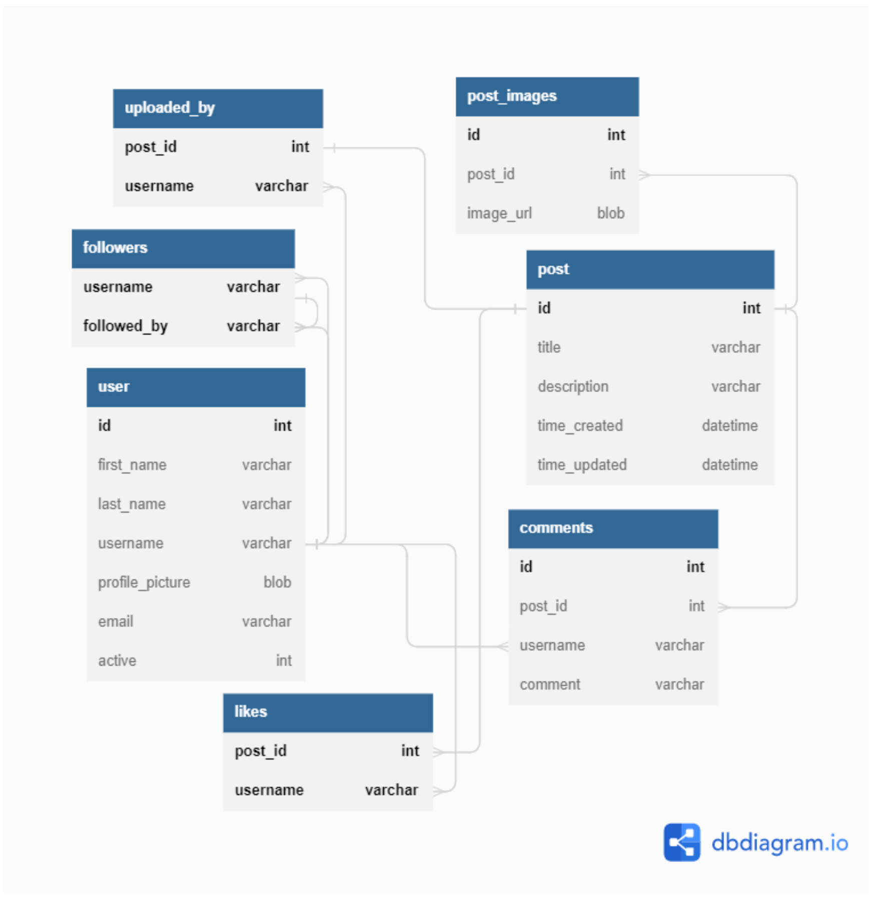
A multi-user application that allows users to upload blogs and share them with other users. Users can like and comment on blogs of others and follow other users to see their blogs on their feed. Users also have a profile page which shows basic statistics about the user such as number of followers, following, posts uploaded etc.

Technologies used

Flask_security for user login, user registration and session based authentication.
Flask_login for user session management
Flask_sqlalchemy for Sqlalchemy support
Flask_excel to export blogs
Sqlalchemy to work with sqlite database
Flask_restful to build REST Api
Wtforms for form validation
Base64 to encode and decode BLOB image file
Bootstrap for frontend styling

DB Schema Design

One post can have multiple images, so images is a multivalued attribute. It has been given its own table 'post_images'. One user can like multiple posts and one post can have many likes. When a user likes a post an entry is made in the 'likes' table. If the user unlikes, the corresponding entry is removed. Same logic is applied for creating the 'comments' table. One user can comment multiple times on the same post and one post can have multiple comments by different users. So each comment is distinguished by id. 'Followers' table has two fields, username and followed_by. The user with username entered in followed_by field follows user with username in username field. Since a user can follow another user only once, each entry is unique and (username, followed_by) together is a composite primary key. Uploaded_by table identifies which user has uploaded the post with corresponding post_id. The user attributes id, email and active field are added to make the user model compatible with the flask_security user model.



API Design

For User

GET (Read) **'/api/user/<username>'** Get details of user with corresponding username along with basic statistics about the user eg. name, username, number of posts, number of followers, number of following
PUT (Update) **'/api/user/<username>'** Update first_name, last_name or username of user.
DELETE (delete) **'/api/user/<username>'** Delete user with corresponding username.

For Post

POST (Create) **'/api/user/<username>/post'** Create a post with title and description entered in the response body for the user with corresponding username.
GET (Read) **'/api/user/<username>/post'** Get array containing details about posts uploaded by the user with the corresponding username. Each post entry has title, description, post_id, author, number of likes, number of comments and number of images attached.
PUT (Update) **'/api/user/<username>/put/<post_id>'** Update title and description of post with corresponding post_id uploaded by user with corresponding username.
DELETE (Delete) **'/api/user/<username>/put/<post_id>'** Delete post with post_id.

Architecture and Features

The **templates** folder contains all html files. The **security** folder inside templates contains html files to override flask_security login and registration page. The **static** folder holds images used in application and contains a **css** folder which has the css file used to style the application.
The **instance** folder contains the database file. The **controllers.py** file has controllers, **app.py** file has configurations as well as the commands to create the application. The **models.py** file has required models, **util.py** file has some functions used in the application. The **api.py** file has the Restful API for application.
User login and registration is done using flask_security by extending flask_security forms and adding custom fields. Validation for custom fields is done at backend as well as in the frontend. For other fields validation is done by flask_security. After login, the user is directed to the home page, where he can see posts by users he follows sorted in descending order by time_created. Users can click on the post to view the full post. Users can like, unlike (if liked), comment and save the post as a csv file. There is a search bar to search for other users implemented by using a 'like' query using sqlalchemy. Users can navigate to other user's profiles by clicking on username. The user profile shows user statistics like number of followers, number of following, number of posts and can view posts posted by the user. Users can view a list of followers, following and who liked a post. Users can edit the title and description of their own posts, as well as edit their own profile. Home page and profile page have a sidebar that contains options to create a new post, and logout. The app also has an api that can be used to perform Read, Update, Delete on Users and Create, Read, Update, Delete on Posts.

Video

Link to video