SERVER DESIGN

BLOGSFORBLOGS

A Simple BlogPost for users.

How the website looks to user?
The blogsforblogs is a simple website which can be used for writing and posting blogs. For writing and reading blogs the user is required to be logged in.
As soon as the user enters the url - https://blogsforblogs.localtunnel.me/blogsforblogs, he is presented with a registration form which asks him to register if he hasn't. After which he has to login with his registered username and password. With the press of the button SignIn he is directed to the "Home" page which lists the recent posts of blogs. There is an option for him to compose a new one. There is a search bar which searches for the title supplied(This has not been accomplished). After posting or reading a blog, he can logout by simply clicking on the link given on the top right corner which will land him to the login page.

*Here localtunnel opensource software has been used to expose the local host. So to get this above stated url one has to open a another terminal window and type the command:

```
lt --port 5000 -s blogsforblogs
```

*For starting the server the we need to activate the virtual environment as the project has been made in this environment - blogs_env. The command for activation is :

```
. blogs_env/bin/activate
```

From starting the server to logging out ...

*For starting the server the command to be inserted is:

```
python run.py
```

*Enter the url - http://localhost:5000/blogsforblogs or https://blogsforblogs.localtunnel.me/blogsforblogs(if localtunnel is used).

*Clicking on Sign In link - bottom of the page - will result in redirection to login page. Same is the case if one clicks on login link located on the top right corner of the page.

*There are various validators used for valid and user-friendly registration and login process.

*Registration form stores the data supplied by the user in the database.

*Logging in happens if the username exists in the database and the password associated

with the username in the database matches with that of the supplied. The password stored in the database is hashed.

*Clicking on Sign In will result in redirection to Home page.

*Home page consists of "compose" option in the right side of the page.

*Composing is nothing but filling the post form with a suitable title and the matter. Clicking on the Post button will take the user to the Home page. At the bottom of the home page will be the latest post of the user with his name as author and posted date.

*Clicking on the link of the title will open the blog in detail. The link is unique w.r.t its id stored in the database and hence its path.

*There is option for logout in the top right corner if the user is logged in. Similarly Home page link will also be available beside it if user is logged in.

Packages used

*Flask

*Flask-WTF

*WTForms

*Flask-SQLAlchemy

*Flask-Bcrypt

*Flask-Login

The working tree:

*blogsforblogs is the working project dir

*blogsforblogs contains:

*blogs_env/ : a virtual environment

*main/ : the package dir

*run.py which on execution starts the server

*main dir contains:

*__init__.py

*routes.py

*forms.py

- *models.py
- *database.db
- *static/
- * templates/

*templates/ contains:

- *layout.html
- *home.html
- *register.html
- *login.html
- *compose.html
- *posts.html

Our project has been structured into a package. The __init__.py file in the main/ states this.

Need: This was necessary as to avoid the issue of circular import. And for making this project look systematic.

OUR APPROACH:

The approach we chose was objective first - we set an objective at every stage and tried to accomplish that thing first and then moved forward.

Our first objective was to design a registration form and this was done by writing register.html which at the begining stage was just an ordinary html code file and by creating RegisterForm class which included the required fields. The connection that is routes was written to link these. And the extension we used here was Flask-WTF and WTForms.

Our next goal was to create a simple database to store the user's details as supplied by him in the register form. This was achieved by making db - an instance of SQLALchemy class, imported from flask_sqlalchemy module. A class called User derived from Models was first created and this marked our achievement for creation of database. A unique thing of this database is to store hashed passwords of the users i.e the passwords were first hashed and then stored. And this was done using a package called Flask-Bcrypt.

The next task for us was to create a login page. Once again this was done using a basic html coding and Flask-WTF and WTforms. This was similar to register form but with some modifications and additions. The login page had to check whether the supplied username already existed or not (in the database). This was done by using check_password_hash method of Bcrypt class.

The login page had to redirect to Home page once authentication was completed. The team worked parallely over designing of Home page and the login page. So this was done simultaneously. At this juncture we also worked at making our project look much more simpler and structured and much attractive. The team structured the project into a package format. Also it downloaded some bootstrap stuff and CSS and JS code snippets from an authorized website to make the Home page look much attractive. And it did so. Using the feature of JINJA template engine that is to layouting of the template we applied the same attractive look to the login and register pages.
The next objective was to make a compose form which enabled the user to write blogs. This was done with the help of Flask-WTF and WTForms. We achieved the process of storing the posts of the user in the database by creating another class called Post and relating it with User class.
We also succeeded in making the title a link to a detailed version of the post. For this reason we designed the posts.html page.
The next major objective was to design the website in such a way that the Home page is viewed only if the user is logged in otherwise redirect to login page. This became easy with the help of another extension called Flask-Login which uses LoginManager class and userloader function to load the user from the database and we have also implemented UserMixin which is a mixin that extends the methods such as user-authentication and so on. The login_user turned the user into a logged-in status and using logout_user the process of logging out became simple. This package has been utilized to a various uses such as current_user authentication so this ensures user to be logged in to view the Home page. The current_user made possible automatic assignment of the author to a post when posted. So we were able to make changes to home.html and posts.html to show the author of the post and the date when the blog was posted.
*We didn't succeed in making the search bar work properly.

references:

* bootstrap for CSS and JS script for our website.
* localtunnel software for exposing localhost.

bibliography:

* flask.pocoo.org
* flask-sqlalchemy.pocoo.org
* flask-bcrypt.readthedocs.io
* flask-login.readthedocs.io
* Flask tutorial tutorialspoint.com

CONCLUSION:

The objective of this blogging website was to give the sense of joy, comfort and security for the user while writing and reading blogs. Coming back to the server design which was the main topic of this project the server design, it taught us the main theme and basic idea behind the servers that companies and digital industries use. The idea of database tells us the essence of knowing the stats and storing them. Our team is proud today to have designed such a beautiful server and client relationship.

*** Thank You ***