DURANGO

Tasks implemented

SURYANSH KUMAR
18CS30043

I have used the Flask framework of Python along with Sqlite 3 as a database , Bootstrap and CSS for the frontend interface and Javascript for certain dynamic GUIs. Following functionalities have been implemented:

- Interactive Home page consisting of links to **Login**, **Register**, **Contact us** and many other quick links to navigate across the application.
- During registration, the ORM **SQLALCHEMY** is used to check the uniqueness of entered credentials. This is followed by verifying the user's mobile number through an OTP system. Users can select either **text** or **on-call** OTP. All credentials are added to the database on successful verification and the user is redirected to 'login' page.
- Features to **add/update/delete any activity** and **view** all activities with their corresponding status (To-do, running, completed or failed) have been added. User can also **update/delete account** and **reset password** via mail
- A **search** bar which uses the **KMP string matching algorithm** is implemented to facilitate search.
- I 've used **celery** (a task queue) to handle the asynchronous task of **sending sms reminders**. **AWS SQS** is used as the broker. It works as following:
    - The AWS server receives the task(send_sms) as soon as the user sets a reminder. Celery sets the 'remind' time (entered by the user) as the execution time for the task.
    - As the time is reached, a celery worker executes the sms API.
- **Pie charts** are developed using javascript. The data is fed to the chart using **JSON dumping**. User can select the date and the chart displays the percentage of all types of activities performed on that date. Similar procedure is followed for other graph.
- The **mail extracting feature** is implemented using **IMAPLIB**. First, the user needs to provide an app password for his gmail account which grants our app secure access to the same. This is followed by parsing the user's mails and searching for particular words (from a predefined list of keywords). If any of those keywords are found either in the mail's subject or in its body, that mail is extracted and rendered to the webpage. User can **view the mail and add it directly to tasks**.
- A **tutorial** link on How to set and manage app passwords has been provided on the homepage to facilitate the use of the above feature.
- In the 'Contact us' page, the user can **send message** to the admin or report a bug. The message is sent in the form of a mail using **SMTP** and the **Flask-mail** module.
- Users can **connect** with one another. A basic **chat** functionality has also been added.