Author

Vaidehi Agarwal
21F1003880
21f1003880@student.onlinedegree.iitm.ac.in
I am currently pursuing my B. tech. Degree, 7th semester in Computer Engineering from state government college along with IITM B.S. course. This is my second Full Stack Web Development project using the Flask module of python.

Description

The goal of this project is to create a user-tracking application for various self-care activities that uses cache, API, and perform asynchronous tasks concurrently. App should automatically send a monthly report regularly, and a reminder, if the user neglects to perform any of the self-care activities. Additionally, users should have the ability to import and export their data in file format and view activities' graphs and statistics.

Technologies used

Frontend Technologies: ● HTML: Creating web pages ● Bootstrap and CSS: Web pages aesthetics and responsive UI ●VueJS: Smooth user interaction and data retrieved vua life cycle hook
Backend Technologies: ● Flask : Web framework with allows to build a web application ● Flask_sqlalchemy : connecting with database ● Flask_restful : support for building REST APIs ● Flask-login : user session management ● Flask_Caching : Caching ● Flask-Mail: send mails ● Matplotlib: plotting graphs ●weasyprint: converting HTML to PDF ● Redis: storing cached values ● Celery: performing asynchronous tasks ● email-validator: Validate Emails ● smtplib: defining SMTP client session object ● Requests: to post or get from an url

DB Schema Design

Data is stored in 5 tables named: Username, Tracker, logging, tracker_type, user_tracker. Table 'Username' stores login data against user_id as primary key. Table 'Tracker' stores details of Tracker against id as primary key. Table 'logging' stores details of log against log_id as primary key. Table 'tracker_type', 'user_tracker' are links between tracker and tracker types, user and tracker respectively. Links are generated while adding data by user.

Table 'Username':
● Column 'uid': Integer datatype, Primary key and Autoincrement ● Column 'username': String datatype, Not Null ● Column 'password': String datatype, Not Null ● Column 'fs_uniquifier': String datatype ● Column 'email': String datatype, Not Null ● Column 'webhook': String datatype, Not Null ● Column 'report_option': String datatype, Not Null

Table: Username

uid	username	password	fs_uniquifier	email	webhook	report_option

Table Tracker:
● Column 'id': Integer datatype, Primary key and Autoincrement ● Column 'tracker_name': String datatype, Not Null ● Column 'description': String datatype, Not Null ● Column 'tracker_type': String datatype, Not Null ● Column 'time': String datatype, Not Null ● Column 'value': String datatype, Not Null

Table: Tracker

id	tracker_name	description	tracker_type	time	value

Table logging:
● Column 'log_id': Integer datatype, Primary key and Autoincrement
● Column 'timestamp': String datatype, Not Null
● Column 'Value': String datatype, Not Null
● Column 'Note': String datatype, Not Null

Table: logging

log_id	timestamp	Value	Note	tid

● Column ‘tid’: Integer datatype, Not Null

Table tracker_type:
● Column ‘lid’: Integer datatype, Primary key and Autoincrement
● Column ‘tid’: Integer datatype, Not Null
● Column ’choice’: String datatype, Not Null

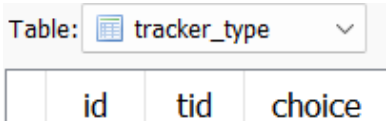
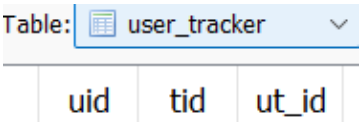


Table tracker_type:
● Column ‘ut_id’: Integer datatype, Primary key and Autoincrement
● Column ‘tid’: Integer datatype, Not Null
● Column ’uid’: Integer datatype, Not Null



This database is designed in such a way to reduce redundancy, time of working with the database. The link table is separated to access the information together without duplicacy when needed. These tables are in BCNF, hence normalised. Also, these modules, like the user module, can be imported to other projects.

API Design

In API design, the project contains signup, login, logout, CRUD operations on the trackers and logging database. It has get, post, put and delete methods, with success responses, error messages and not found messages. The code is stored in an api.py file. The documentation is stored in the api.yaml file. The APIs are also cached with cache expiry to increase its performance.

Architecture and Features

The project is divided into different modules like app module, database module, controller module, celery module, cache module, api module. All the HTML are in the templates folder, CSS in style folder and VueJS scripts are in the static folder. All the controllers are in the validation.py file. All the Database tables are in model.py. All APIs are connected through api.py. Validation of api is done in Apivalidation.py file. All the asynchronous and scheduled tasks handled in celery are in tasks.py file. Api caching is done in api.py file. Everything is part of the app.py module. ‘Cache_initialization.py’, ‘config.py’, ’database.py’ and ‘workers.py’ are initialization files for cache, app, database and celery respectively.
In this project, dashboard and other functionalities cannot be accessed if not logged in. Login is through Flask-Security cookie authentication. Data are user specific. Dashboard contains cards of trackers, create, edit and delete tracker options, view and add log buttons, export tracker details, option for choosing monthly report format (html/pdf) and logout. Each tracker card displays tracker id, tracker name, description, tracker type and time and value of last review. On clicking the view log button, the log data screen will open. Log Data screen contains a list of tracker logs, which is editable and deletable, add log, trendline and stats buttons, import and export log data from/to csv file. Line chart and min, max, mean values for numerical and time duration types of trackers, bar chart and count statistics for boolean tracker type, bar chart and least used and most used options count for multichoice tracker type are used. Graphs are plotted using matplotlib. Exporting is done as an asynchronous task through celery. Auto triggered scheduled tasks i.e. daily reminder message in evening and emailing monthly progress report in user specified format html/pdf on first day of every month using celery, weasyprint, smtplib and requests. Api caching is done on retrieving trackers, tracker’s log details, which is implemented though Flask-Caching module to improve performance. API is implemented on CRUD operations of trackers, tracker’s log data. HTML pages are responsive for both mobile and desktop. Different tracker cards on the dashboard have different colours based on tracker type, i.e., Numerical, Multiple Choice, Boolean, Time Duration.

Video

https://drive.google.com/file/d/11G6s8LP99yDGqvouGnDTbz4Z4Jw5oZ2M/view?usp=sharing