

Author

Vaidehi Agarwal

21F1003880

21f1003880@student.onlinedegree.iitm.ac.in

I am currently pursuing IITM B.S. course and on campus B. tech. Degree from state government college. This is my second Full Stack Web Development project and first project where I used celery, webhook, cache, redis.

Description

The goal of this project is to create mobile and desktop responsive user-tracking application for various self-care activities that uses cache, API, and perform asynchronous tasks concurrently. App should automatically send a monthly report in preferable format 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

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_security: Token Authentication • 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: <input type="text" value="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 Auto increment
- 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
	id
	tid
	choice

Table tracker_user:

- Column 'ut_id': Integer datatype, Primary key and Autoincrement
- Column 'tid': Integer datatype, Not Null
- Column 'uid': Integer datatype, Not Null

Table:	user_tracker
	uid
	tid
	ut_id

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 duplicity 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 CRUD operations on the logging database. It has get, post, 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 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 files are in the templates folder, CSS files 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' file. All APIs are connected through 'api.py' file. Validation of API is done in 'Apivalidation.py' file. All the asynchronous and scheduled tasks handled in celery are in 'tasks.py' file. Caching of functions is done in 'data_acess.py' file and 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 token authentication. Login username, password, email address, webhook ID, report format are collected during signup. All data are user specific.

Dashboard contains cards of trackers, create, edit and delete tracker options, view and add log options, export tracker details option, 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. There are 4 tracker types, i.e., Numerical, Multiple Choice, Boolean, Time Duration. 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 delete-able log-wise, add log, trend line and stats options, options for import and export log data from/to csv file. Line chart and min, max, mean values for numerical and time duration types of tracker, bar chart and count statistics for boolean tracker type, bar chart and least used and most used options count for multi choice tracker type are used for trend line and statistical options. Graphs are plotted using matplotlib.

Exporting is done as an asynchronous task through celery. Message displays to inform user that he can continue further without waiting for completing download.

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 are done using celery, weasyprint, smtplib and requests.

Function caching is done on retrieving trackers and API caching is done on retrieving tracker's log details, which is implemented though Flask-Caching module to improve performance. Cache vanishes after a time interval. API is implemented on CRUD operations of 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.

Video

<https://drive.google.com/file/d/1T3ZN-zsEa5gvdLdnqioNePRGxcfMItr/view?usp=sharing>