Author

Akhil P L
21F1006584
21f1006584@student.onlinedegree.iitm.ac.in
I am a working professional, currently doing third term of Diploma in this program.

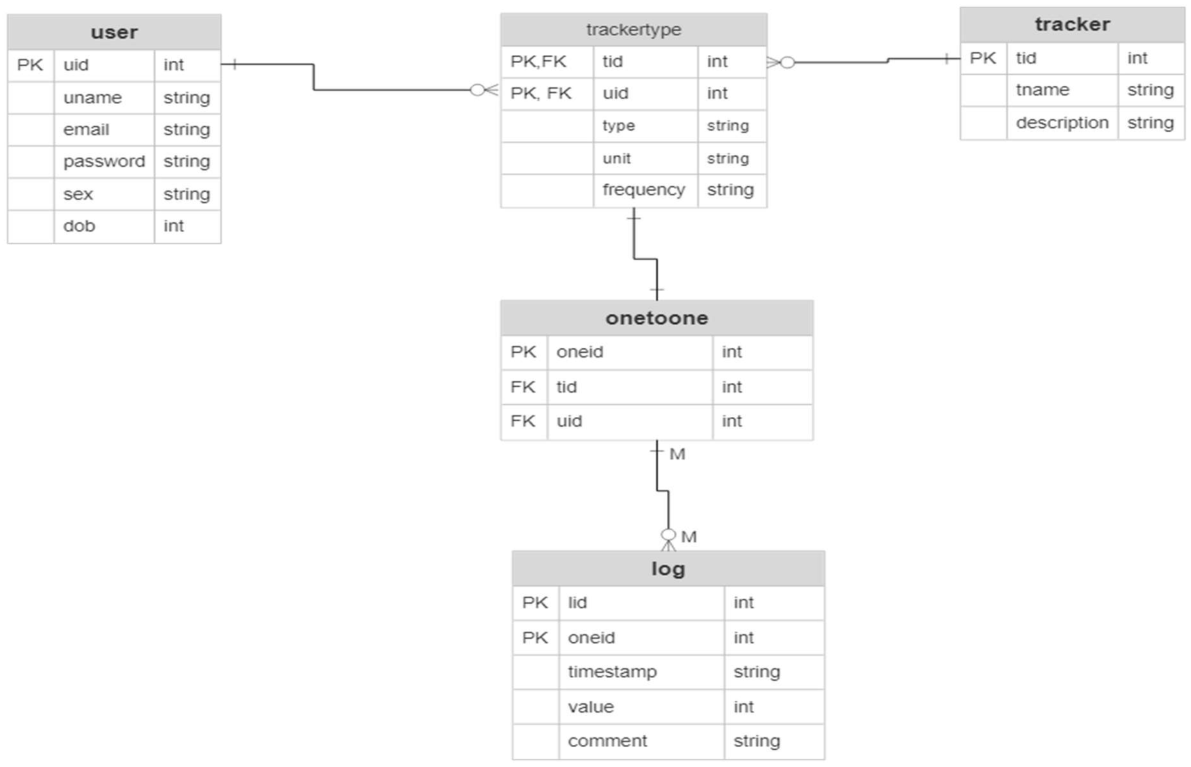
Description

The aim of the project is to build a self-tracking app, in which user is able to track any type of parameters (Numeric, Text, Multiple choice, Rating etc). The goal is to reduce the load on server by doing rendering of UI & UX on the client frontend. Meanwhile server should be able to do scheduled jobs or other jobs like sending user triggered emails, caching etc.

Technologies used

- Jinja 2 – For html template rendering for sending emails and partially as frontend.
- Flask – Flask framework is used for the app creation as it is simple and as it has most of the essential extensions inbuilt.
- Flask-SQLAlchemy – Is used for modelling and querying of database.
- Flask-Security – Is used for authentication of user and to provide a basic security of data.
- VueJS – Is used as a JavaScript framework for building UI & UX.
- Redis – A in-memory data structure store used as message broker for managing jobs queue.
- Celery – An asynchronous job queue for scheduling and executing jobs.
- MailHog – A fake SMTP server for email-testing

DB Schema Design



This is the ER diagram of the database schema. Apart from all primary keys, email in **user** table and tname in **tracker** table is having a unique constrain. **Trackertype** table is a one-to-many relation table of user table and tracker table. This allows the user to customise any tracker based on their preferred type (Numeric, Boolean, Rating, Multiple-choice) and frequency (Hourly, Daily, Weekly, Monthly)

An additional one-to-one relation table named **onetoone** was included as it was difficult to have two foreign keys from trackertype table as primary key along with lid in **log** table. Value column in log table is an integer attribute. Boolean and Multiple-choice values need to be converted to corresponding numeric values for storing in the database. This is helpful not only to have a single log table, but also will be easy to change type of an existing tracker without losing data till date (e.g., A Boolean value Yes/No stored as 0/1, can be changed to Multiple choice value Good/Bad/Neutral stored as 0/1/2).

API Design

API elements are created for CRUD operations of individual User, Tracker, Trackertype and Log. Apart from that API elements to get all Trackers, all Logs of a trackertype and to send email of all Logs of a trackertype to the specific user is also created. The yamil file is present in along with the project code.

Architecture and Features

The entire project is organised as a proper full stack structure. All the HTML files are stored inside 'templates' folder. The Python codes are separated according to their purposes and are stored inside corresponding folders inside 'application' folder. The SQlite database file is stored inside 'db_directory' folder. The external packages need to be installed are listed in the 'requirements.txt' file. The project can be executed buy running "local_setup.sh" file followed by "local_run.sh" file in a Linux based command prompt. Then for the scheduled jobs to execute:

1. Start a Redis server (redis-server)
2. Run "local_beat.sh"
3. Run "local_workers.sh"
4. Also, for viewing the email install and run a MailHog server

On running the app will be accessible only by registering and logging in. Once a user is logged in user will be taken to their home page. Where they can view complete list of trackers they are currently using. There are options for viewing details of each tracker or to add latest log to a tracker or to even add a new tracker all together. Following are the features available

- ✓ UI with Vue & Vue Components
- ✓ Tracker management using fetch API
    - Create or add new tracker specifying type and frequency of tracker
    - View logs of each tracker
    - Add, edit or delete logs
- ✓ Token based authentication for fetch API's
- ✓ Reminders to add logs via email based on the frequency selected
- ✓ Able to send HTTP logs report as email on user request
- ✓ Caching of data that don't change frequently

Video

https://drive.google.com/file/d/1GP_AlSaX0gHvy54P-pFydr_Raleqgy7q/view?usp=sharing