Author

Name : Kaustav Goswami
Roll number : 21F1001588
Email id : 21f1001588@student.onlinedegree.iitm.ac.in
I love to do maths as well as programming, which data science provides me in the perfect proportion. I'm currently pursuing B.Sc Computer Science (Hons). I love to participate and create new projects.

Description

The main aim of this project is to help a user track different types of measurements in his/her daily life. So the user should be able to login to their account and should be able to do CRUD operations on each tracker and also on each of the logs in a tracker. However, a user will not be able to edit the tracker type after the tracker has been created.

Technologies used

Here's are the technologies I used in this project.

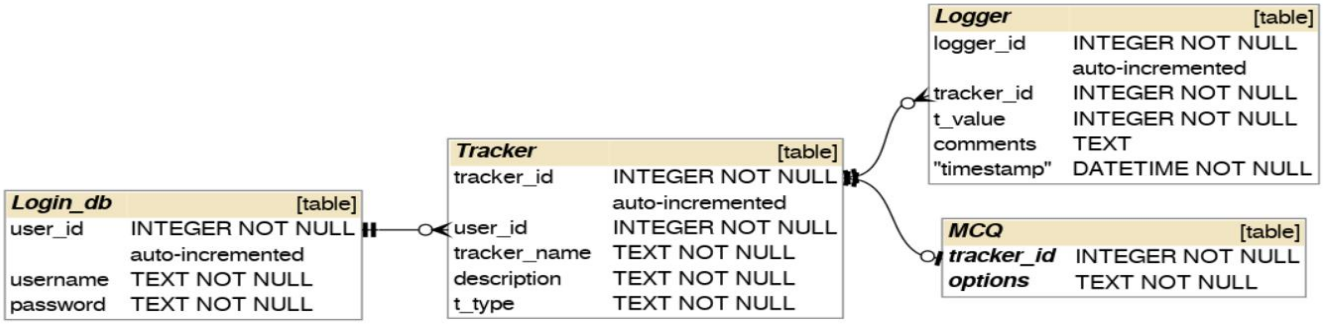
Python, HTML/CSS/JS, Jinja, bootstrap, Flask, Flask-Login , Flask-SQLAlchemy, Matplotlib's pyplot.

- Python is the core programming language used.
- Flask is the main framework used for the Web-app.
- Flask-Login is used for managing multiple user login and keeping a session alive.
- Flask-SQLAlchemy is the SQL toolkit used to connect with the database file.

DB Schema Design

Table Name	Columns	Description	Constraints
Login_db	user_id	User's unique id	Integer, Primary key, Auto increment
	username	Username of the user	Text, Unique , Not Null
	password	Password of the user's account	Text, Not Null
Tracker	tracker_id	Tracker's unique ID number	Integer, Primary key, Auto increment
	user_id	User's id who created the tracker	Foreign key to Login_db.user_id
	tracker_name	Name of the tracker created	Text, Not Null
	description	Description of the tracker created	Text, Not Null
	t_type	Type of the tracker created (Numerical or Multi-Choice)	Text, Not Null
Logger	logger_id	Unique ID number for each log	Integer, Primary key, Auto increment
	tracker_id	Tracker's ID for which the log is	Foreign key to Tracker.tracker_id
	t_value	Value of the log recorded	Integer, Not Null
	comments	User's comments for this log	Text
	timestamp	Time at which the log was recorded	Datetime
MCQ	tracker_id	Tracker's ID for which the setting is	Foreign key to Tracker.tracker_id
	options	Store the options given by the user for the tracker	Text, Not Null

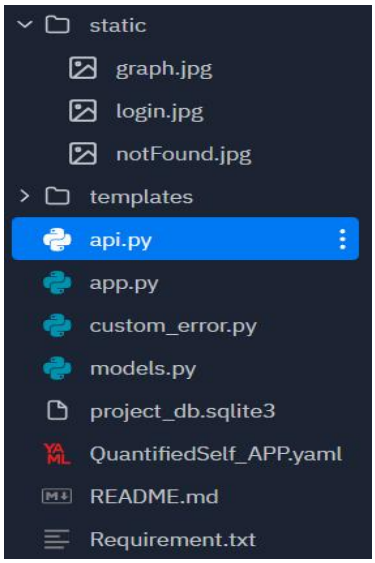
ER Diagram



API Design

I have implemented Get, Put, Delete and Post for Login_db table as User_api class, Tracker table as Tracker_api class, Logger table as Log_api class. For MCQ table I have implemented Get, Delete and Post only as Setting_api class. I did not implement Put operation for Setting table because once a tracker is created, it's type or the user given settings cannot be changed.

Architecture



Here, I have 2 folders:

- **static** - which holds the image files
- **templates** - which holds all the HTML files.

Then I have made 4 python files :

- **api.py** file has all the code related to API's
- **app.py** file has the code for the main code to start the Web App
- **models.py** file has the all the code related to the different models
- **custom_error.py** file has the code related to the custom errors that I have created for my Web App.

QuantifiedSelf_APP.yaml file is the yaml file for the API I have implemented. **Project_db.sqlite3** is the database file. **README.md** has the instructions on how to start the Flask Web App. **Requirement.txt** has the required packages name.

Features

Here's a list of features :-

- Multiple users can use the Web App at the same time.
- Users can access the Web App even after closing and re-opening the browser and it wouldn't take the user to the login page unless the user have logged out or have accessed the login page explicitly.
- Interactive page which shows an alert if incorrect username/password is given or if the password given does not match the re-typed password during new registration.
- CRUD operations on tracker - Create, Read, Update, Delete.
- CRUD operations on log - Create, Read, Update, Delete.
- Graphs for each tracker – summarizes all the logs graphically.

Video

Link to my video