Author:

Name: Hanani Bathina,
Roll No: 21f1006169,
Email: 21f1006169@student.onlinedegree.iitm.ac.in

Description:

1. Build a Tracker application where users can create trackers, and log values.
2. Users should be able to CRUD on both trackers, and logs.
3. Users should be able to visualize the data they've entered.

Technologies Used:

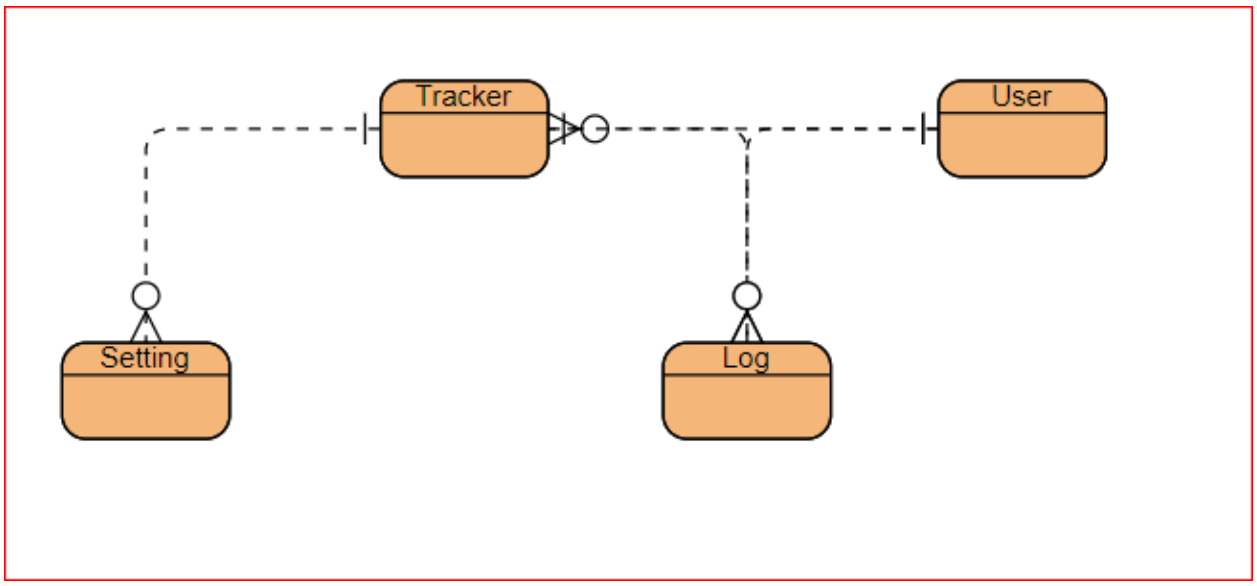
1. Flask: Web Framework
2. Jinja 2: Template Engine
3. Bootstrap: Layout/Styling
4. SQLite: Database
5. SQLAlchemy: ORM
6. Flask-Restful: REST APIs
7. Flask-Security: Authentication
8. HTML/CSS: Obvious
9. YAML: API Spec
10. Python: Language
11. Matplotlib: Data Viz.

DB Design:

1. One User Many Trackers
2. One Tracker Many Logs
3. One User Many Logs
4. One Tracker Many Settings

Note: Settings table is for Multiple Choice Trackers. This is the table for options in a multi-choice tracker.

ER Diagram:



API Design:

I implemented:

1. 6 GET requests endpoints
2. 3 PUT requests endpoints
3. 3 DELETE requests endpoints
4. 4 POST requests endpoints

I can:

1. Create & Read a User
2. Read All The Trackers of a User
3. Create, Read, Update & Delete a Tracker
4. Read All The Logs of A Tracker of A User
5. Create, Read, Update & Delete a User
6. Create, Read, Update & Delete a Log, including timestamp
7. Create Dashboard Visualization
8. Easily retrieve the Last Review of a Tracker

Architectures & Features:

1. User Registers his account with his email_id, and a password.
2. The password is stored after being encrypted using bcrypt.
3. All the pages are authenticated.
4. User can register 5 kind of trackers:
    a. Numerical ⇒ Values are stored using db.Integer
    b. Decimal ⇒ Values are multiplied by 100 and stored using db.Integer
    c. Boolean ⇒ “0” or “1” is stored using db.Integer, where “0” is False, and “1” is True
    d. Time Duration ⇒ No. of minutes are stored using db.Integer
    e. Multiple Choice ⇒ Values are stored using db.Integer. The integer mapping of a choice is maintained in the ***Setting Table***

5. Validation in HTML pages is done using JavaScript, and the Validation is also done in the Controller.
6. Project Structure:

```
├── api.py ⇒                                  All the APIs are stored here.
├── app.py ⇒                                  The Index Page of This Application
├── config.py ⇒                               All the Configurations are stored here
├── controllers.py ⇒                          All the Controllers are stored here
├── database.py ⇒                             The Database Initialization is done here
├── database.sqlite3 ⇒                        The Database
├── models.py ⇒                               All the Schemas are stored here
├── templates ⇒                               The Templates Directory
│   ├── 404.html ⇒                            404 Page
│   ├── edit_log.html ⇒                       Log Edit Page
│   ├── edit_tracker.html ⇒                   Tracker Edit Page
│   ├── index.html ⇒                          Home Page
│   ├── logs.html ⇒                           Logs Page
│   └── trackers.html ⇒                       Trackers Page
├── visualizations ⇒                          Dashboard Visualization Directory
│   ├── 1 ⇒                                   Visualizations of User "1"
│   │   ├── 1_10_1.png ⇒                      Visualization Type "1" of id "1" and tracker_id "10"
│   │   ├── 1_10_2.png
│   │   ├── 1_1_1.png
│   │   ├── 1_1_2.png
│   │   ├── 1_2_2.png
│   ├── 2
│   │   ├── 2_11_1.png
│   │   └── 2_11_2.png
```

README.md, API.yaml, Project_Report.PDF are also included.

7. Each tracker has a Dashboard which has "Scatter Plots", "Trend lines", "Horizontal Bars", "Pie Charts", "Sum", "Max", "Min", "Avg"
8. Two Database Triggers to update the last_review and last_value variables in the tracker.

Video:

https://drive.google.com/file/d/12LXmJhHzUYS7V5vHVB4L-EWa8kCi5Fna/view?usp=sharing