Author

Rakesh Kumar Mandal
21F1007049
21f1007049@student.onlinedegree.iitm.ac.in
Hi, I'm Rakesh. I live in a small village of West-Bengal. I did my graduation in Physics. While doing science fair activities, I got interested in coding and joined this program and really enjoying it.

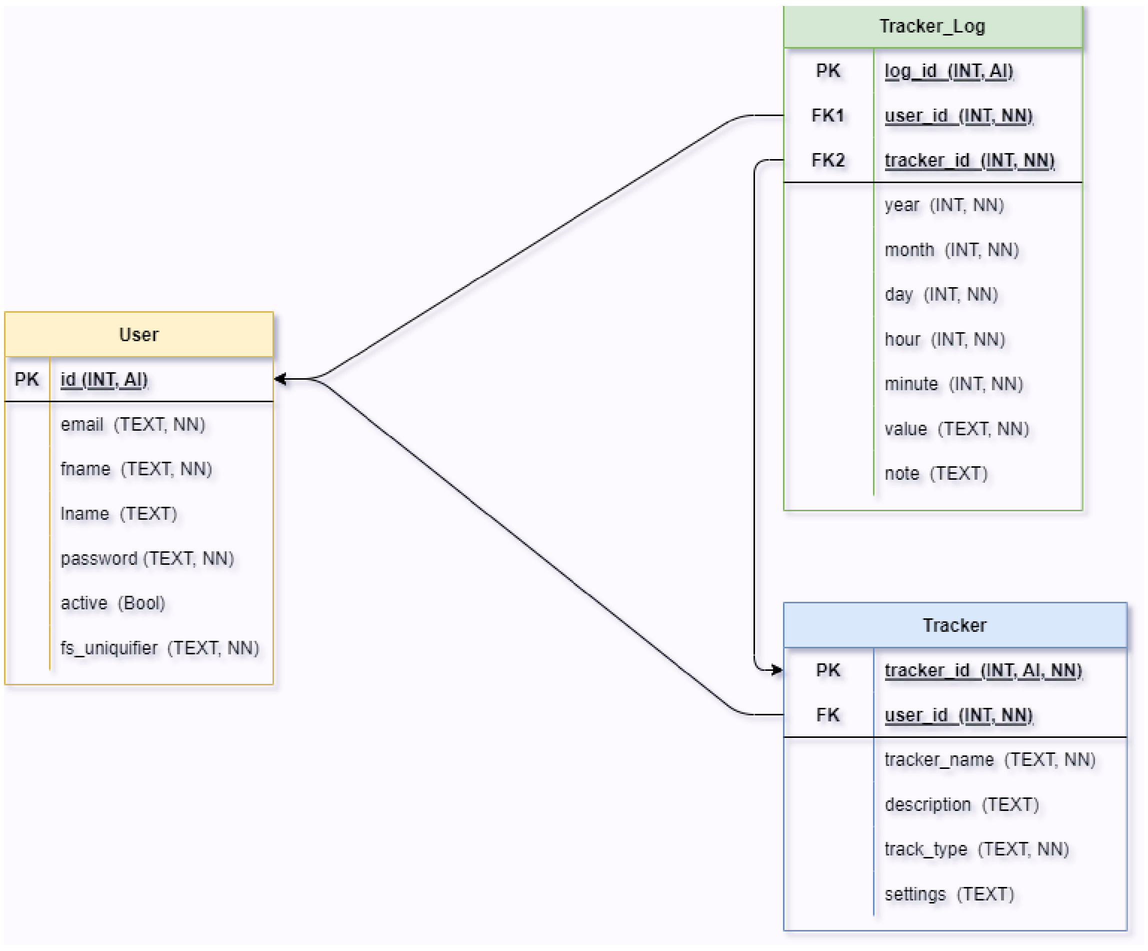
Description

This is Quantified Self Application - V2. A database has to be created for User info, Trackers & Logs. The frontEnd will communicate to the server through API. With a database server, there will be servers for doing asynchronous backend jobs.

Technologies used

- HTML, CSS, Bootstrap, Javascript (Vue JS, Vuex) for Frontend, Chart JS for plotting graphs.
- Python Flask backend, Flask-RESTful API, SQLite3 database with FlaskSQLAlchemy ORM
- Redis for caching, Redis and Celery for batch jobs, MailHog as fake SMTP server
- WeasyPrint for creating PDF reports,

DB Schema Design



API Design

- **"/api/user"** endpoint with POST, GET methods to Create and Read UserAPI
- **“/api/alltrackers”** endpoint with GET method Read Tracker data of all the trackers
- **“/api/tracker”** endpoint with POST method Create new Tracker with request body
- **“/api/tracker/<int:tracker_id>”** endpoint with GET, PUT, DELETE methods Read, Update and Delete a particular Tracker respectively with data as request body (when required)
- **“/api/trackerlog”** endpoint with POST method Create new Log record with request body
- **“/api/alllogs/<int:tracker_id>”** with GET method Reads all Log records of tracker_id
- **“/api/trackerlog/<int:log_id>”** with GET, PUT, DELETE methods Reads, Updates, Deletes particular Log record of log_id respectively with data as request body (when required)
- **“/api/exportascsv/<int:tracker_id>”** with GET method generates CSV file of all the Log records of Tracker and pushes a task to Celery system to send the user’s mail

Architecture and Features

The **Root** directory of the **Project** folder contains two folders: **BackEnd**(entire backend system), **FrontEnd**(entire frontend system) and a **readme.md** file. **BackEnd** has four folders (**applications, csvdatafiles, db_directory, pdfreports**) and five files (**app.py, app_run.sh, app_setup.sh, record.log, requirements.txt**). Main app is **BackEnd/app.py**. The application should be started by running **app_run.sh** file. The SQLite3 database is inside **db_directory**, database ORM models are inside **BackEnd/application/models** folder, API resources are inside **BackEnd/application/resources** folder, Celery system, Mail services, and related templates are inside **BackEnd/application/celery, BackEnd/application/mail_service, BackEnd/application/templates** directory. In the application folder there are **utils** folder and three other files also (**__init__.py** and **config.py, cache.py**).
FrontEnd has five folders (**components, images, router, view, vuex**) and five files (**app.js, config.js, FetchFunction.js, index.html, style.css**). All the components not directly rendered by the router are inside the components folder, and the components to be directly rendered by routers are in the **view** folder.
Features Implemented:-

- First user can **login/signup**, implemented **token-based-authentication** (flask-security)
- From Dashboard users can Create new tracker, Go to any Tracker page, create new log, Edit/Delete existing Trackers. **CRUD** on Trackers and TrackerLogs.
- In the Tracker page statistics from existing log records will be shown as trendlines/bar charts, and all log records will be listed there, users can Create/Edit/Delete new(any) log.
- Users can export logs data related to any tracker as CSV, implemented using the **Celery** system in the backend. Implemented **Caching** in the backend.
- Backend Celery(beat) system and **Redis** db with crontab will ensure that users will get **daily reminders**, **monthly reports** related to their trackers they’ve created.
- **Single UI** for both Mobile and Desktop/Laptop, **NavBar** for navigating around.

Video

Video Link is: Please Click Here