TM - Time - Tracker HOSTED ON HEROKU

A normal CRUD app without the "U" + a chart feature

Under the hood...

Language Python | Framework Flask | DB postgresql | Template Jinja

Python dependencies

- *flask-sqlalchemy* - A library for easy ORM
- *python-dotenv* - For easily tracking environment variables in `.env` files
- *gunicorn* - Server for heroku
- *psycopg2-binary* - PostgreSQL driver

Front-end depedecies

- FontAwesome
- jQuery
- Bootstrap
  - Popper
  - Bootstrap Table
  - DateRange Picker
    - Moment
- D3.js

Installation

Run the python virtual machine to isolate dependency installations. This will also automatically detect and install dependencies in Pipfile

```
$ pipenv shell
```

Run flask to start a development server. By default, the site would be served in http://localhost:5000

```
$ flask run
```

To test *"production"* server used in Heroku, start `gunicorn` instead

```
$ gunicorn app:app
```

Credentials and Local Access

Default login credentials:

- **Username:** `admin`
- **Password:** `admin`

> *Note:* I didn't have enough time to create UI for user signup, but new users may be added through the API (See section on **API endpoints** and **Postman**), or directly in the databse

Local Access

1. *Database:* Heroku PostgreSQL

   > Credentials are available in the `.env` file I uploaded in the Google Drive shared folder

2. *Postman* This is helpful for easier access to the API endpoints. The following files are available in `\postman`:
   - *API Collection Import:*
     - `api.postman_collection.json`
   - *Environment Imports:*
     - `local.postman_environment.json`
     - `heroku.postman_environment.json`

The code in a nutshell...

There are two parts to this *(...and really most)* implementation *(s)*: front-end (`Jinja`, `HTML`, `JS`, `CSS`) and back-end (`Python`). The front-end is what you see from the browser; and for the back-end, I've made several API Endpoints available.

API Endpoints

- *User:*
  - `/api/login`
  - `/api/logout`
  - `/api/user/add`
- *Checkins:*
  - `/api/checkin/add`
  - `/api/checkin/delete`
  - `/api/delete/table`
- *Tags:*
  - `/api/tags/search`

> These may be accessible through Postman for easier testing/viewing. Available parameters are also included in the import/export file. Refer to **Local Access** section

> **Implementation Notes:** `tags` are a separate entity/table for scalability.

> **Implementation Notes:** I also contemplated of whether to design this with API's or to do backend processes directly since they're both on a single server anyway. I went with API's because it was easier to test. It took a little more time setting up, but I think it's worth it

Front-end / Client-side

These are available in `\templates`. Everything else is pretty straightforward

Pages:

1. Login Page
2. Loading Page
3. Table/Chart Page
4. *(Extra)* Hello World Page

Features:

1. Login/Logout

   Uses Flask Session to store logged in user data

   > **Implementation Note:** I'm known to always go the extra step but for this module, I'm so sorry to say that passwords aren't encrypted. I WAS going to add this later on, but given that I'm already late in submitting this, I no longer had enough time. If I **were** to add password encryption, I'd just use a basic one way encryption algorithm such as MD5 or SHA in the python code before processing to database.

2. Filter by Date Range

   This is my solution to this requirement:

   > 2. Show all check ins, grouped by day

   > **Implementation Note:** This is unfortunately all I managed to do, but the filter function is implemented dynamically so it would be simpler to add other filter functions

   > **Implementation Note:** Both the table and chart gets updated when filtered. Any operation on the table data will be reflected on the chart/s as well

3. Add/Delete Checkin

   Each checkin has a `status` attribute with values `active` or `deleted`. The `Delete` operation does not actually delete the data, but marks it as `deleted`

4. Pie Chart

   This is a D3 Pie chart. The data available (filtered/unfiltered) is grouped by tags. It's interactable (SVG) and has some basic transitions when filtering the data.

   > **Implementation Note:** There's a bug in the transition when only one tag remains. This is fixable but I didn't have enough time

   > **Implementation Note:** `SVG` is easiest to implement when it comes to interactable charts but when dealing with bigger datasets, a `canvas` implementation would be better suited.

General Implementation Notes

1. For the front-end, I didn't spend too much time on error and asynchronous handling, but at the bare minimum, they do exist for both add/delete operations
2. On local, I sometimes get a database connection error. I put out an `alert` to refresh page when this happens. On the flipside, you get to really appreciate the loading screen when this error occurs since the page doesn't go further anymore.
3. I'll disclose this just in case my python code isnt optimal, this is the first time I'm using Python that isn't a personal project. Took a risk here, but I hope it's not too shoddy

File Index

```
├── postman # Contains postman import files
├── src
|   ├── api
|   |    ├── checkin.py # Contains available /api/checkin routes
|   |    ├── user.py # Contains available /api/checkin routes
|   |    └── checkin.py # Contains available /api/checkin routes
|   |
|   └── models # ORM files (flask-sqlalchemy)
|        ├── checkin.py
|        ├── user.py
|        └── checkin.py
|
|
├── static # Directory for assets such as scripts and images
|   ├── css # Contains stylesheets per pages
|   └── js # Contains scripts per module
|
├── templates # Jinja templates
|   ├── components # Reusable component templates
|   └── pages # Page templates
|
├── .env
├── .env-sample
|
├── app.py # Main/start python code
├── routes.py # Contains available browser routes.
|
└── README.md
```

Miscellaneous Links:

- **Google Drive Shared Folder** - https://drive.google.com/drive/u/1/folders/1Ye28ZJQKigaFUhn0G5QcBu2LyAr7M65I
- **Github Repository** - https://github.com/kayecandy/tm-time-tracker
- **Live Demo (Heroku)** - https://tm-time-tracker.herokuapp.com/