TimeCraft

Master your fate one day at a time.

Team

Just me 🙂 Benoni Esckinder

Technologies

- Languages: HTML, CSS, JavaScript, Python, MySQL, SQLalchemy, React
- Frameworks: Flask, REST,
- Options and Tradeoffs:
    - Django and flask are both Python web frameworks used to make web apps. While both are good options, I will be using Flask because that is the framework/library I know how to use best. Especially in regards to making API calls and rendering templates.
    - There are many front-end frameworks like React. One of them is Angular. I chose React because I am more familiar with it.

Challenge Statement

- Has it ever felt like 24 hours just isn't enough time in the day to do the things you have to do? You seem to be losing hours and you don't know how. I want to create this app so that I(and hopefully you) can keep better track of your time. Now you can see where all those hours are going. You can also get reports of your time usage across weeks and months.

- This is not intended to be a task management app. It is made for people who want to track how much time they're allocating daily to the things they care about. It is made with people in mind who have goals like "Read a book for 30 minutes everyday" or "Exercise for 2 hours, 3x a week" and

"Work every waking hour of the day".(That last one might be a bit too excessive, but one can only dream).
- This project is direct to consumer and it could help boost productivity. It could also be used as a tool to track the productivity of employees in the workplace.
- This project is currently independent of locale

Risks

- **Technical Risks:** Mapping reports to the calendar might cause issues. Duplicate entries for the same day, accidentally removing the stored report for the day by restarting the program(the second one most likely being an early development issue.
    - **Potential Impact:** Since the main goal is to track and report data across time, loss or damage of the real data would invalidate the results and the very purpose of the app.
    - **Safeguards:** Careful modeling and storage of data, rock-solid persistence mechanisms(Making sure the data isn't changed every time the program is opened or closed). Creating safeguards against unwanted changing and deletion of data.
- **Non-Technical risks:**
    - User Errors: If the initial setup phase isn't made clear and simple for the user, the app may be used incorrectly. (For example: Using it as a daily task manager or assuming by running the program, it automatically tracks time on task etc.)
        - **Potential Impact:** Data collected won't be as useful, users might label the app "too complicated" or "useless" because they have incorrect assumptions of the app's purpose and utility.
        - **Safeguards:** Making clear, simple man pages on each command and in general app itself. I also plan to have a brief explanation at first contact with the app and possibly a walk-through.

Infrastructure

- Since I'm the only member, The application will be divided into development steps(features) and the repository will be branched according to those steps(features). Github is the version control system we will be using.
- A small scale MVP with basic features will be deployed initially.
    - That means the initial service will handle querying the user for their specific goals and division of time
    - CI/CD cycle will be implemented in order to add more features. (More storage options, more complex data)
- I plan to use Python's unittest module since that is the infrastructure I am most familiar with.
    - This might change in the future depending on the requirements of the project

Existing Solutions

- Hive:
    - Similarities: Task management
    - Differences: The approach seems a bit different. TimeCrunch is more focused on the daily allocation of time while Hive is more task focused and concentrates on wider time ranges

- Todoist:
    - Similarities: Task management
    - Differences: More of a todo list and less time on task tracking focused.