HabitTracker - The Making Of

Tom Schmaeling

Introduction

This document serves as a companion and describes the creation of the project. The project's structure and functionality are not the focus of this document. They will thus only be mentioned when necessary.

In this document I would to explain the development process, but also mention parts of the application I am proud of, what I have learned and how I would further develop this prototype into a functional application.

Link to the GitHub repository: https://github.com/LeftEmpty/habit_tracker/

Development Process

The project took around 2 months of on-and-off work. I initially ran into a couple issues because one, I am new to python, this being my first full-fledged project using it. And two, I underestimated the importance of proper planning for UI/UX when designing an application with a GUI.

This in combination with a continuously improving understanding of python resulted in many parts of the projects being refactored constantly or even rewritten from scratch.

While this may sound like pure chaos and poor planning (and it probably is), this was also intended when I set out to create this application. The idea was to use a 'big-bang' approach, because I knew from the get-go that I would hate about 90% of the code I wrote about a week later as I improved.

Thus, not planning rigorously also meant that I lost less time in a sense when the inevitable refactor of the project came along about two to three weeks into the project.

With the major re-factorization also came more diligent project planning and management. This included simple UI designs, set timeframes for features, and improvement of the UML ER- and Class-diagrams.

The rest was just execution which went mostly smooth with the occasional bug here and there.

Interesting Features

All things considered this project can be improved a lot, but I'll talk more about this in the Future Outlook section. That being said, there are mainly two features in this project I am quite happy with.

The main feature, that might make this project somewhat stand out compared to other submissions is the decoupling of a habit's "content/identifiers" and it's "streak/completion data". What I mean with this will become clear very fast when inspecting the project, but to quickly summarize.

A "Habit" as intended by a task may have a name as well as data that determines the progress/state of it. In this project, this construct is divided into `HabitData` that describes the habit and `HabitSubscription` which includes data about completions, streaks, the owning user, etc.

This solution enables a lot more freedom when creating new features and when handling data. It should also serve to lower required storage as habit's can be partly reused by multiple users.

Although not at all the focus of this project, I like the GUI's 'screen system' used for to display different `Screen` classes (widgets) inside the main GUI window.

This in combination of the sidebar, which populates depending on the initialized screens makes for a very clean solution in my opinion.

What I Learned

Other than the previously mentioned mistake of underestimating UI/UX design. I made a couple of major and minor mistakes during this project.

The most major mistake, which is quite obvious when reading the task and then looking at the project is undeniably my lack of scope-control. The scope-creep in this project is quite bad. This is likely due to the chosen development approach and my urge to immediately try out 'new cool python stuff' as I continued to learn.

In future projects, I would like to be less loose with the planned result and instead firmly decide a precise outcome for the project. When learning a new language and/or framework, as in this case, a test project might be a good idea.

Future Outlook

As of now I don't plan on further developing this project too much. But there are a couple of things I would like to improve when I find the time.

For starters, to make this project actually feel professional, the UI/UX needs to be improved drastically to meet modern standards. The GUI doesn't look very good/modern and there are a couple of issues with the UX that could be tweaked for improved QoL (Quality of Life).

Some sections of code in the project could also be more generalized, this could reduce bloat of this project.

Although I may sound quite negative about this project, I am all in all very happy with the result.