Phase 3: Abstract

Simeon Akpanudo

Matriculation: 32105494

https://github.com/Cmion/oofpp-habit-project.git

13, January 2023.

Synopsis

The Habit Tracker project's objective is to create a non-GUI-based application that lets users track their habits weekly or daily. The project relies on the Python programming language and the functional and object-oriented programming paradigm. The most notable pitfalls of a non-GUI application are lack of a good user experience which can be attributed to it being terminal based. The design of the application aims to reduce these limitations by using an interactive approach. For example, users can select habits using the down and up arrow keys on their keyboard, and the process for completing a task such as adding a habit is designed as steps. The application prompts the user to choose a task using an interactive question such as “What would you like to do?” and presents them with a menu of tasks they can complete. The application also uses colorization and icons (ASCII Characters) to add context to the texts displayed on the terminal.

Execution

The app utilizes the vast open-source library of Python to implement some of its features. The app’s entry point is the conventional “main.py” file, here the application is initialized. The main modules of the application are the Habit, Tracker, Database, Analytics, and Command line Manager classes. Each module handles specific tasks such as database connection, and simple CRUD (Create, Read, Update, and Deletion) operations. The Analytics module handles simple analytics, such as getting the same habits with the same periodicity, getting the habit with the current longest streak, or the habit with the longest streak. The Habit module is used as a physical model of habits, the Habit class creates a habit and provides methods to modify them. The command line module is used to interact as a buffer between the app and the terminal, it accepts users' inputs, validates, refines, and calls the appropriate class methods to implement a feature. The command line module also controls the lifecycle of the application using a loop to keep the application running, and only stops the application if the user exits by pressing the key “e” on the keyboard or a forced exit.

The Eureka Effect

One of the features of the application is the ability to define a custom database file during initialization. When adding this feature there was no intent of it being used within the application, it was added as a convenient way to change database (SQLite) files during development.

In phase 2 of the application, I noted the difficulty in testing due to the use of a database and how it will require a mock of the database to test. The feedback on the submission drew my attention to the possibility of using the custom database feature to insert dummy data into a “test.db.” file for testing. This method enabled me to test modules that I could not test initially.

Pitfalls

Some of the tests in the application lacked enough coverage (at least 50% coverage). It was also difficult to ascertain a stable percentage, as the tool used for testing provided varying values.

Conclusion

Based on the unit testing and manual testing I have conducted on the app, the application is stable for use, it meets the criteria and expectations of the project. For real-world use, I will recommend we build a GUI version, since most ordinary non-technical people would find that easier and intuitive to use.