Creating a Habit Tracking App

Conception phase

DLBDSOOFPP01 - Object Oriented and Functional Programming with Python

B.Sc. Data science

26/02/2023

Author: Mohammadsadegh Solouki

Matriculation no.: **32007275**

Tutor: Prof. Max Pumperla

1. Project goal

We must create a habit tracker app with the goal of assisting users in forming constructive habits and monitoring their development. The program should provide an intuitive command-line interface that enables users to interact with it by choosing commands. Anybody wishing to monitor their habits and better their lives should find our habit tracking program to be a wonderful resource. The program should be simple to use and provide a multitude of options that make it simple to stay motivated and on track. Future plans for the program should include the development of new features and interfaces that will boost its functionality and efficiency.

2. Requirements

The habit.py file, which is essential to the app's operation, must specify the Habit class. A habit object is defined by the Habit class and consists of the habit's name, periodicity (daily, weekly, or monthly), and ID. Every habit has a creation date, which must be indicated as the day the habit was first started, and a completion date, which is updated when the habit is completed. Also, there are methods in the Habit class for adding, removing, and marking completed habits. We should also determine the longest and current habitual streaks that may be updated as we designate a habit as complete.

The db.py file manages the database class. The program makes use of the lightweight and quick SQLite3 database, which enables the storage and retrieval of habit data. Because to its simplicity of usage, minimal overhead, and compact size, SQLite3 is a popular option for small to medium-sized applications. A table for habits, a table for completions, and a table for streaks are all included in the database. The habit table includes the ID, name, periodicity, date of creation, latest completion date, and total number of completions for the habit. The habit ID and the completion

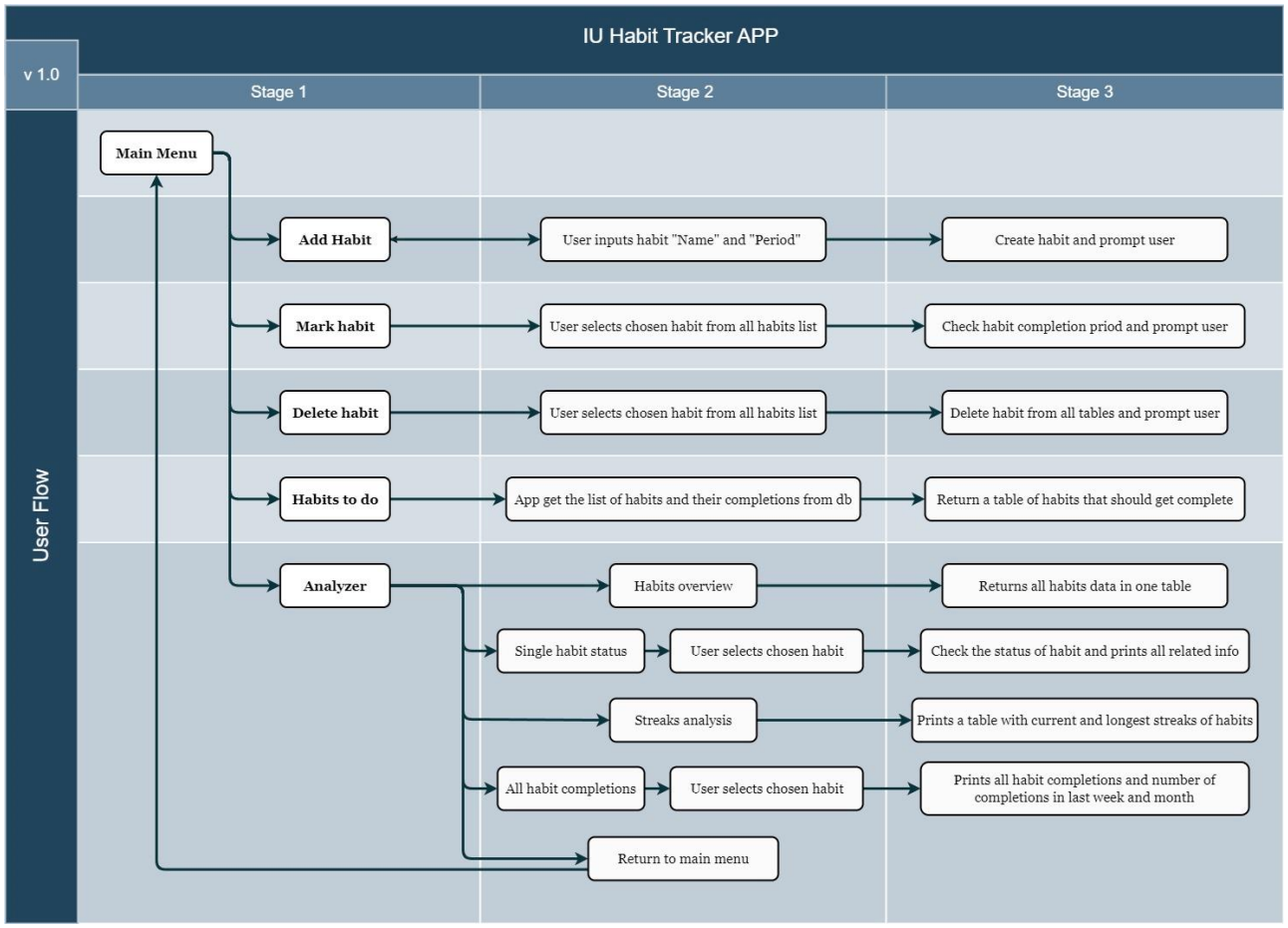
date are both included in the completions table. The longest streak, current streak, and habit ID are all kept in the streaks table.

The command-line interface defines how users communicate with the software. To make it more interactive, we can just utilize the Python input syntax or even the Questionary module. Users just need to choose "Add habit" and then enter the habit's name and frequency to create a new habit. Adding a "study" habit that happens "daily," for instance. Users must choose the "Mark habit as complete" option, then choose a habit name based on their ID to mark a habit as completed. Users may choose any habit from the list of all habits and then click the "Delete habit" button to get rid of it. A "Habits to do" option will allow users to view a list of the habits that have not been finished within their period in order to be reminded of which ones they should do today. They will be guided by the "Analyzer" option through various features that our analyze module would provide consumers to extract all the necessary data about their routines.

We must add an analyzer module to the app programmed in the analyze.py file to provide them a better knowledge of their progress. Users may get an "overview" of their habits using the analyze module, which includes the creation date, latest completion date, and total number of completions. By choosing "Single habit status," users may examine more information about each habit, such as streaks and a prompt informing them of whether they performed the habit within its given timeframe or not. Based on the database's completion dates, the app will determine the habit's current and longest streaks. By entering "streaks analysis" users may display the current and longest streaks for all habits. Users may also see all habit completion dates as well as the total number of completions in the most recent week and month to determine which behavior they have lately struggled with the most.

Using the Pytest library, we must create test cases for the Habit class, database class, analyzer module, and other capabilities that our app would provide in order to guarantee the quality of the code. The popular Python testing framework Pytest offers concise and straightforward feedback on test outcomes. We should make use of this framework to test app functionalities to make sure they are stable and reliable. With the use of these tests, we can make sure that the code performs as intended and that any changes do not impair already-existing functionality.

3. User flow diagram



4. Database schema diagram