Michael Jachim, “Object Oriented and Functional Programming with Python” (Portfolio)

Concept :

Creating a habit tracking application, which allows users to define and track one or multiple habits with specified periodicity. Users can mark tasks as completed for each period (daily or weekly), track their progress, and receive analytics regarding their habits, e.g. streaks or missed habits.

1. Application structure :

1.1 User Interface (UI) :

Create a CLI-class which will allow users to :

- Input Command : Accepts a command from the user and executes it.
- Show Habits : Shows a list of all the habits the user is attempting.
- Add Habit : For users to be able to add a habit, asking for “name” and “frequency”
- Remove Habit : For users to be able to remove a habit
- View Statistics : Displays the statistics, regarding streaks and missed habits.

1.2 Habit Class :

The “Habit”-class will store all relevant information regarding the user’s habit :

- Task Description : The action the user wants to complete (e.g. "Working out", “Studying” etc.).
- Frequency : How often the habit should be completed (daily or weekly).
- Completion Dates : A list of dates or time stamps showing when the user completed the habit.

1.3 User Class :

The “User”-class stores information about individual users, including :

- User Data : Personal information, like username, age etc..
- Add/Remove Habits : The ability to add or remove habits.
- Habits List : A list of habits or “Habit-objects” the user created

This allows to specify the user, and allows each user to track each habit independently.

1.4 Data Storage :

The app needs to store (and retrieve) habit data continuously, since users will want to return to the app and see their past habit data without losing progress.
A solution can be a SQLite database (or JSON) to store habit and user data.
This database will be queried to load user-specific habits, completion statuses and analytics.
The interaction with the database will be done via simple SQL queries (e.g. INSERT, SELECT, UPDATE etc.), or simple JSON queries.

1.5 Analytics :

Analytics module to provide insights like :

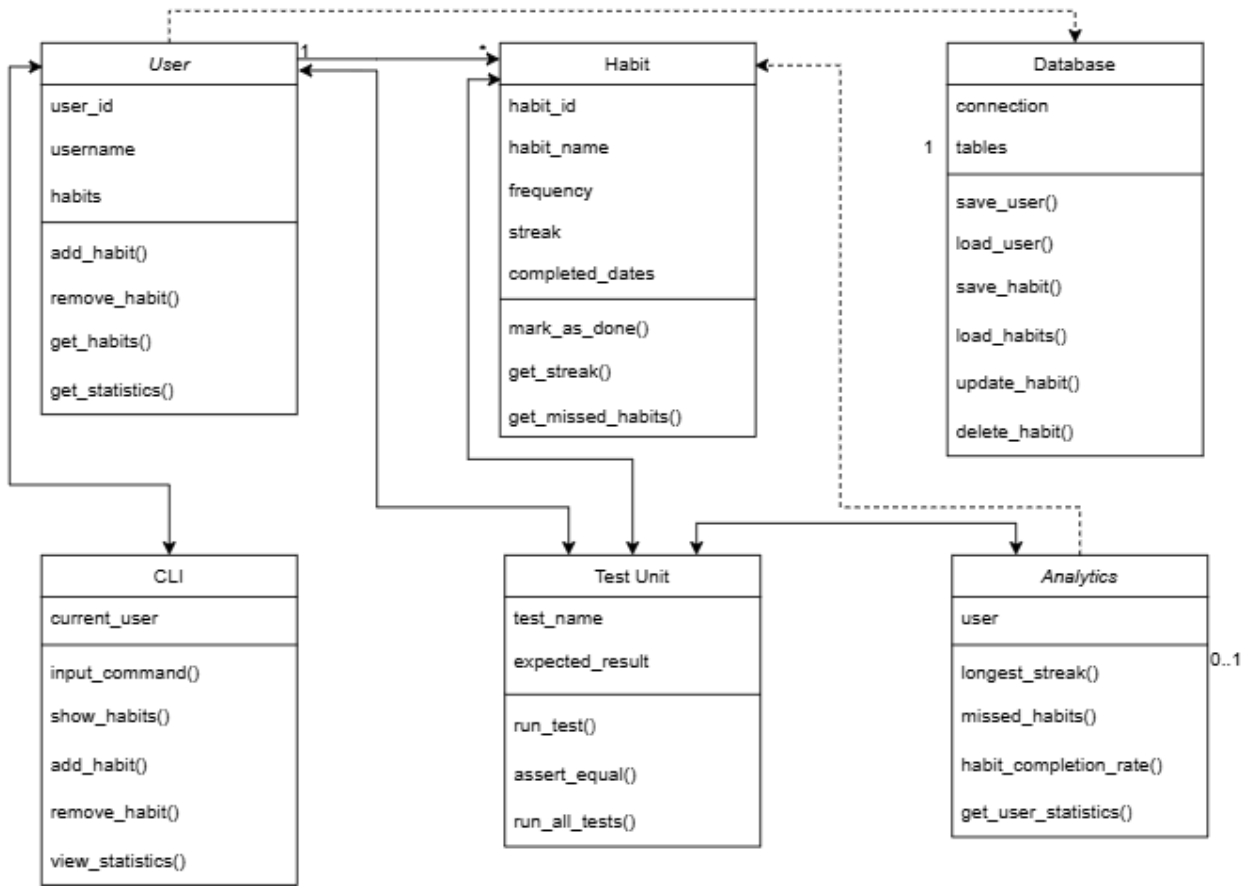
- Longest Streak : Consecutive amount of times the habit was completed.
- Missed Habits : Track which habits have been ignored.
- Get User Statistics : Aggregates and returns (various) statistics,

This will be a module, where functions query the database to generate reports, such as calculating streaks or identifying missed habits.

1.6 Test Environment :

The app needs a basic test unit, I will use “unittest” with example-inputs for the habits.

2. UML-Diagram (basic logic) :



The reason why I decided to go for several classes, is because I think (and hope) it creates an easier understanding and overview regarding the different components, and it's probably easier this way to extend the program in specific ways.

The above mentioned classes will be imported into and initialized by a small separate main-class.

Also using a CLI-Tool is straightforward and makes interactions with the app simple but effective, if needed, a different programming-language or tool can be used to create an application-interface.