ABSTRACT

TRACK A HABIT

## 1. Introduction

This document describes the software development of a back-end habit tracking application, called Track A Habit. The purpose of the project is to create a Python-based backend application; therefore, a graphical user interface is not within the scope of this version. The objective of the application is to provide users the possibility to create habits, mark them as completed, and analyze them. Additionally, the system provides habit's statistics, such as habits longest streak, and habits which the user struggled the most during the habit's periodicity.

## 2. Development Description

The system was implemented using Python programming language, with emphasis on modular design, using small and independent components to enhance reusability, maintainability, and scalability.

Regarding the technologies and design paradigms applied, two main approaches were used. The first and most important paradigm used was object-oriented design, which was used to build Habits.py module. On this module a habit is treated as an object, on which attributes and functions were created. The Habit class encapsulates the logic of creating, managing, and interacting with habit records in the database. Its constructor has three attributes: the habit's name, habit's description, and the habit's periodicity (daily, weekly or monthly). The Habit class has a few methods to interact with the database:

- *Add_habit()*: inserts the habit into the database.
- *Checkoff_habit()*: marks the habit as checked off in the database, the method also lets the user know that the habit was marked as completed.
- *List_all_habits()*: retrieves all habits from the database.
- *__str__*: string representation of the habit.

There are a few considerations about the Habit class:

- It is assumed that a database module already exists.
- All database operations are delegated to the database module.
- Error handling for database operations is handled on the database module.

Functional programming paradigm was applied to build the database.py module, by organizing logic into pure set of functions in order to manage the SQLite database. There are no classes or global variables used in the module.

A database was chosen to persistently store data regarding a habit, the name, periodicity, description, and checked off date and time is stored on a database. There are four tables created for the database manipulation, the usage of primary key, and thus, foreign key was chosen to :

- Periodicity table: currently hardcoded values are inserted: daily, weekly, and monthly.
- Habit_Plan table: stores the habit data.
- Habit_Tracker: stores all instances of checked off habit by using the primary key from the Habit_Plan table. By using a foreign key, it avoids storing redundant data.
- Habit_Default: a table to store predefined habits which could be used by the end user.

The application intentionally minimizes the user input on the command line interface. For instance, when a user wants to check off a habit, the application itself will provide a list of the already created habits in order to avoid typos. Similarly, when the user selects the habit's period the system provides the list of periods from the database (Periodicity table).

## 3. Python Libraries

Track A Habit system makes use of the following libraries to support reusability and functionality:

- Datatime: used for date and time manipulation.
- Sqlite3: used to provide database operations.
- Calendar: used to calculate month based on the number of the days of the month.
- Questionary: used to enhance user experience while interacting with the terminal.
- Pytest: for automated testing of the code.

## 4. Evaluation of Results

Track A Habit currently fulfils the objectives of the assignment, it meets the criteria of building a backend application using Python 3.9, itself contains a README.md document, the code is commented, object-oriented and functional programming were applied accordingly, the solution comes with five predefined habits. It also includes an analytics module, while a database is used to store habits information, and finally it comes with a test suite included.

The most difficult part of development was the way consecutive streaks are calculated. Given the fact that the Habit_Tracker table stores the date and time timestamp for the entry, when a streak is calculated, the system retrieves all entries for a particular habit and it also needs to know its periodicity. Based on the timestamps, the system calculates how many consecutive streaks were

checked by the user. A daily habit was easy to calculate, knowing that today's entry was checked off, the systems checks if today minus one day was also checked. If so, the streak counter is incremented. In case it was not checked off, the counter is set back to zero. However, for the weekly habits the system needs to identify which particular week of the year it was checked off, and by retrieving the next checked off habit compare it to the previous one. Furthermore, the system also checks if the last week of the year is the previous week of the first week of the year.

## 5. Future Improvements and Conclusion

The system currently does not allow the user to create customizable periodicities; in case the user wants to create a habit to be checked off every two weeks the system will not allow it. The analytics module should incorporate a logic to handle customizable periodicities. Following this improvement, the system should be able to analyze the customizable periodicity.

In conclusion, Track A Habit represented a challenge to apply theoretical knowledge into a practical and tangible application. It encouraged me to plan in advanced and to adopt a problem-solving approach whenever the initial planning did not evolve as expected. This project not only strengthened my technical knowledge but also enhanced my capacity to reflect and develop a real-world scenario.

GitHub repository: https://github.com/cris2carrere/TrackAHabit.git