Portfolio

Conception phase – Habit tracker

DLBDSOOFPP01 – Object oriented and functional programming with Python

Rafael Mutis-Maschmann

rafael.mutis-maschmann@iu-study.org

Table of content

List of figures

1. Introduction

Who was not dreaming about realizing a new, desirable habit but failing miserably when it comes to successful realization?

First we have a look at the definition and characteristics of a habit: A habit is a response pattern that is performed automatically by a person in a specific situation (Lally and Gardner, 2013). It is an internalized reaction whose execution is no longer linked to any specific goal but triggered by a specific situation. This automatism is highly efficient as it allows us to pay attention on other topics while performing the habit. One of the success factors to implement such an automatic routine is prolonged practise (Boakes, 1993).

Such a prolonged practise can be achieved by using a habit tracker app on a mobile phone. As the mobile phone is used frequently it is the perfect platform for a tracker app. This app can support the user to stay on track by showing and reminding of pending actions. Moreover the possibility of an analysis shall motivate the user. This concept paper contains the general functions as well as requirements and structure of such a habbit tracker app: HabiTrack

2. Functional overview and requirements of the HabiTrack

The figure below gives an overview of the different use cases the HabiTrack application provides to its users:

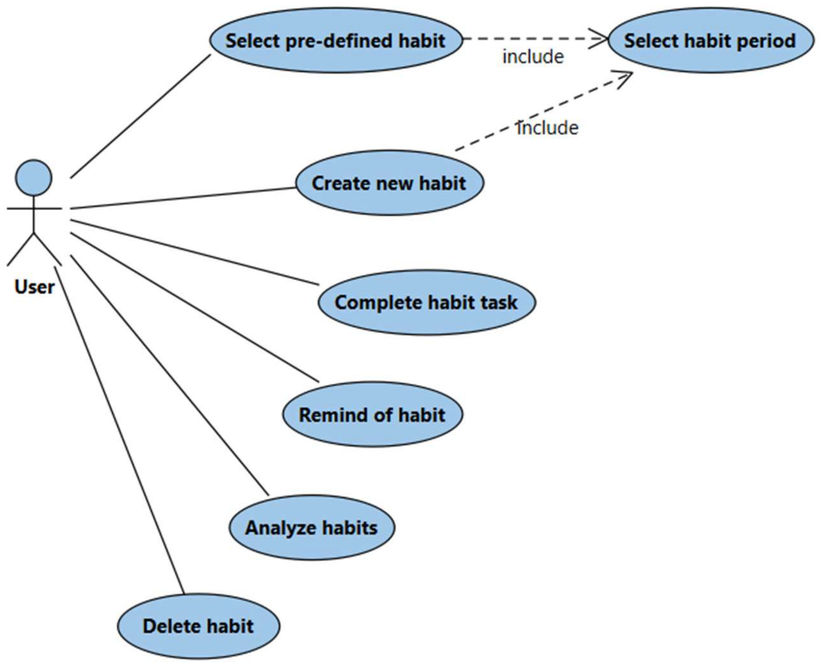

Figure 1: UML Use Case of habit tracker

The programming will be realized in PyCharm using Python 3.12 and the cli-package for interaction with the user. When starting the application these use cases form part of the main menu for the user.

As a prerequisite for habit tracking the user needs first to define the relevant habits whether by creating or selecting a suggested habit.

- Select pre-defined habits:
  - Req-01: The user shall be able to select one or more of 5 suggested habits.
- Create new habit:
  - Req-02: The user shall be able to create a new habit by entering its name.
- Select habit period: This step is mandatory after creating or selecting a suggested habit. With the habit and period defined it is possible to start the habit tracking.
  - Req-03: The solution contains pre-defined habit periods (at least weekly and daily). The user cannot create new habit periods.
  - Req-04: For each habit selected / created the user must select one of the pre-defined habit periods.
- Complete habit task:
  - Req-05: For each habit selected / created the user shall be able to mark it as completed if due date, based on the start date and the habit period, is not exceeded.
  - Req-06: If the due date is exceeded the system shall mark the habit as missed. Note: To prevent cheating the user shall not be able to mark a habit task as completed after expiration of the due date.
  - Req-07: Once a habit is completed or missed the system shall calculate the successor habit task.
- Remind of habit:
  - Req-08: The system shall provide to the user an overview of all habits selected / created and the next due date.
- Analyse habit:
  - Req-09: The user shall be able to analyze his/her habits:
    - return a list of all currently tracked habits
    - return a list of all habits with the same periodicity
    - return the longest run streak of all defined habits
    - return the longest run streak for a given habit
  - Req-10: The user shall be able to analyze his/her habits:
    - return the strongest habit from all defined habits (strongest = habit with the highest ratio of habit task completed to the sum of habit tasks completed + missed)
    - return the weakest habit from all defined habits (weakest = habit the lowest ratio of habit task completed to the sum of habit tasks completed + missed)
- Delete habit:
  - Req-11: The user shall be able to delete any selected / created habit.

3. Structural class diagram

The following diagram illustrates the structure of the HabiTracker:

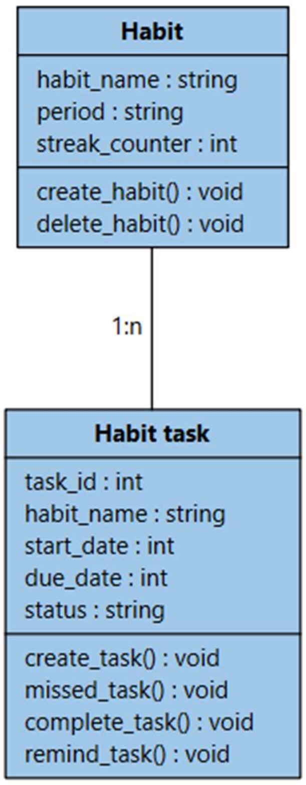

Figure 2: UML class diagram of HabiTracker

The class habit is used to have an overview of all defined habits including the selected period. The storage will be done using sqlite3-DB with the habit name as primary key. The function create_habit() will create a new instance of this class while the function delecte_habit() will not only delete the selected habit object but also all linked habit tasks.

The class habit task is used for detailed tracking of the habit task. The attribute habit_name is used here as link to the previously defined habit object. The attribute status is defined as type string to distinguish the different status of a habit task: open, completed or missed. The function create_task() will create a new habit task based on the selected period of the habit object and calculates the respective due date. The function missed_task() will be used as an update-function in order to check for any entries with an exceeded due date and change its status to “missed”. If done so the function shall also call create_task() to create the successor task. The function complete_task() is designated for the user to check off a habit task and change the status to “completed”. The function remind_task() is used for the use case “remind of habit”.

4. Bibliography

Lally P., Gardner B. (2013) Promoting habit formation. Health Psychology Review, 7 (2013), pp. S137-S158. https://doi.org/10.1080/17437199.2011.603640

Boakes R.A. (1993) The role of repetition in transforming actions into habits: The contribution of John Watson and contemporary research into a persistent theme. Mexican Journal of Behavior Analysis, 19 (1993), pp. 67-90. ISSN 0185-4534