Habit Tracking Application

Table of contents

Introduction

Everybody wants to stop unhealthy habits and create good habits in its place. They are turning to technology for assistance to achieving this. We want to create a habit tracking application to assist them to achieve their goals.

Problem statement

We need to create a backend for our habit tacking application.

Requirements

Use cases

1. As a user I want to create a habit with a description / specification and frequency[1]
2. As a user I want to flag a task as completed at a specific date and time
3. As a user I want to be able to create a habit for at least two tracking periods e.g., daily, weekly, or monthly
4. As a user I want to analyse the data as follows
    a. List of currently tracked habits
    b. List of habits with the same periodicity e.g., daily, weekly, or monthly habits
    c. What is the longest run streak for a habit?
    d. Which habits do I struggle with?
5. As a user I want to store the data between sessions

[1] Frequency refers to how often a habit must be repeated

Business Rules

1. The user must complete a task once during the period otherwise he breaks the habit
2. The user must complete a task for x consecutive periods to establishes a streak

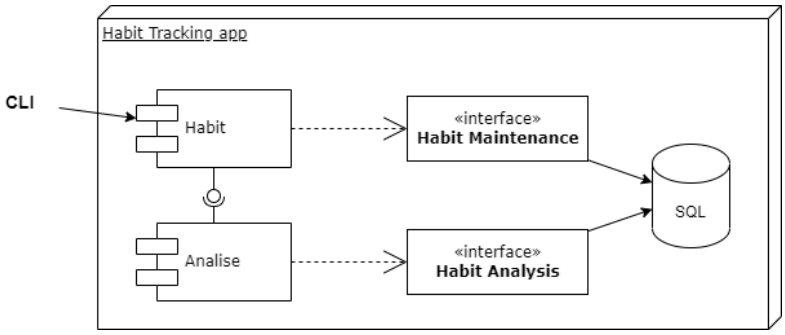
Methodology

I will follow a Test-Driven Development (hereinafter referred to as “TDD”) approach. This is a software development practice that repeat the following steps

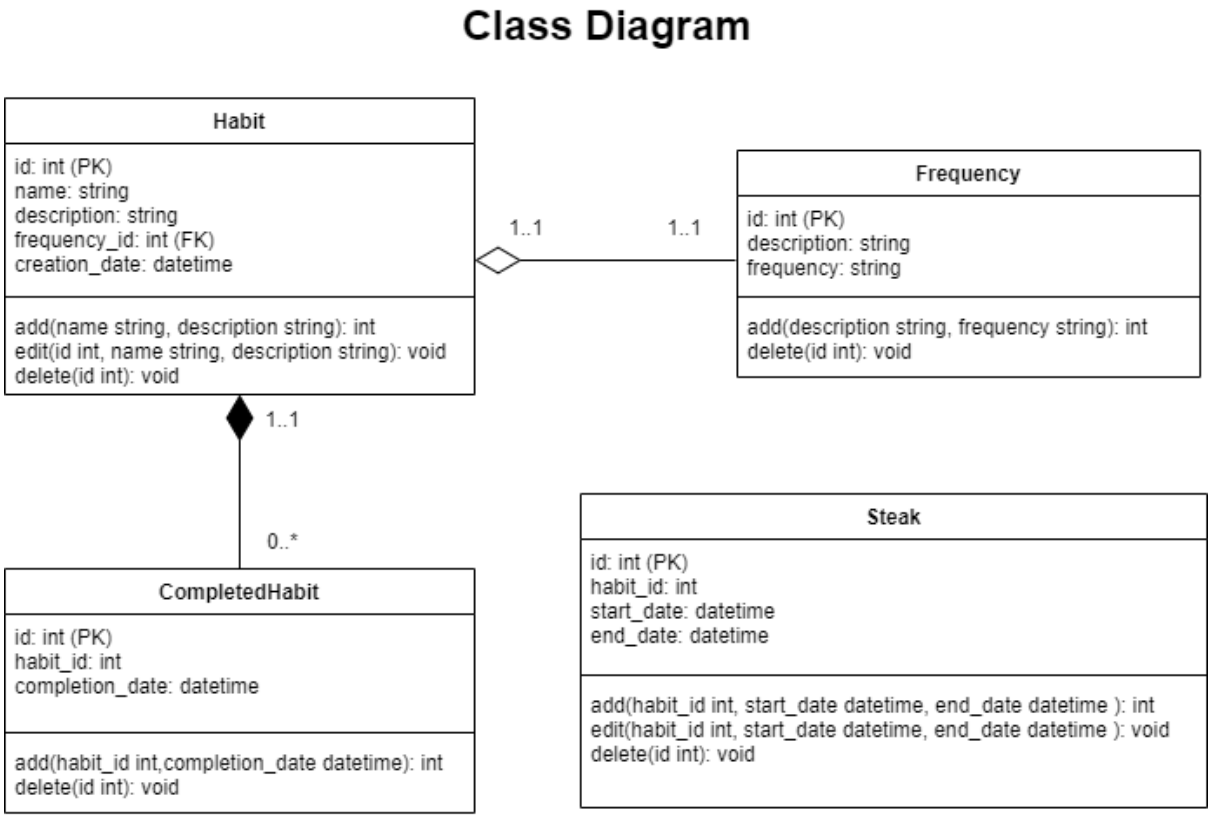
1. Write a test for a feature that fails
2. Write code to make the test pass
3. Refactor the code as needed

Design Overview

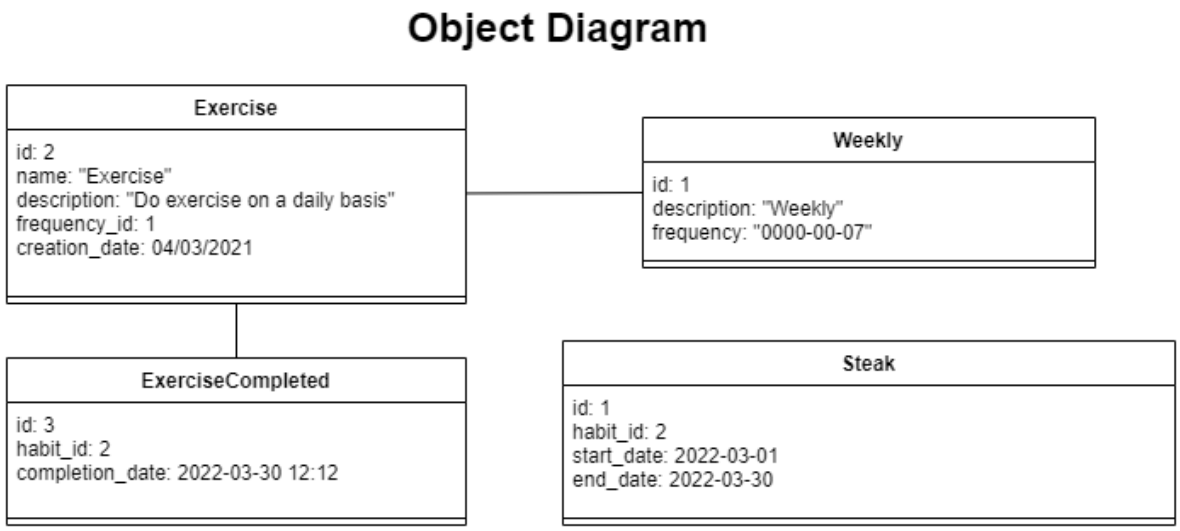
Component Diagram



Class Diagram



Object Diagram



Technology choices

- ***Python version 3.10.3*** – Project requirement to use 3.7 or later
- ***Visual Studio Code*** - Popular IDE / source-code editor that runs on Windows, Linux and macOS.
- ***sqlite3*** – It is a library that provides lightweight disk-based database to persist the data
- ***pytest*** – Framework for writing tests.
- ***FastAPI*** – Framework for building APIs with python. This will provide an alternative for the CLI
- ***click*** – Python library for creating command line interfaces
- ***Docker*** – Container technology for running our application
- ***Pylint*** – Linting tool that checks for coding errors and enforce coding standards
- ***Swagger UI*** – interactive exploration to call and test your API from the browser

Assumptions

1. Preload / seed data on project start-up for testing
    a. Load daily, weekly, and monthly data in the frequency table
    b. Load five predefined habits (at least one weekly and monthly habit)
    c. For each preloaded habit provide four weeks of tracking data
2. The Streak table will only be updated when the analysis module is executed
3. The Habit module will be developed using Object Orientated Programming and Functional programming for the Analysis module.
4. We do not require a frontend, but we will provide a CLI and Swagger documentation (OpenAPI specification - OAS) for the user interaction.
5. Provide detailed instructions in a markup document (Readme.md) on how to start and use the system

Bibliography

1. n.a. (n.d). Google Python Style Guide – Naming https://google.github.io/styleguide/pyguide.html#316-naming
2. Lutz, M (2013). OOP: The Big Picture, Learning Python 5th Edition. O'Reilly

3. Kymberly, F (2018, March 1). UML class diagrams in draw.io https://drawio-app.com/uml-class-diagrams-in-draw-io/
4. Marcus, S (2021, July 16). Test Driven Development with pytest https://stackabuse.com/test-driven-development-with-pytest/