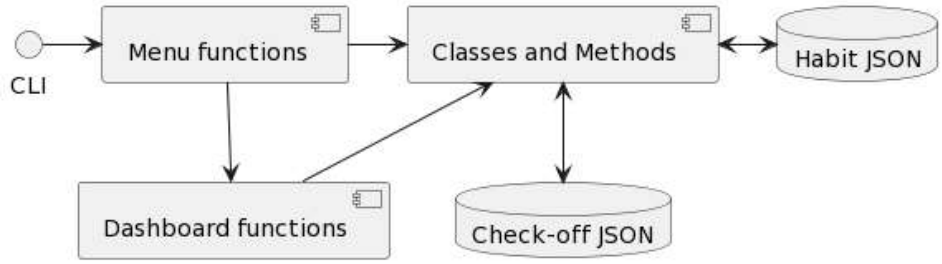
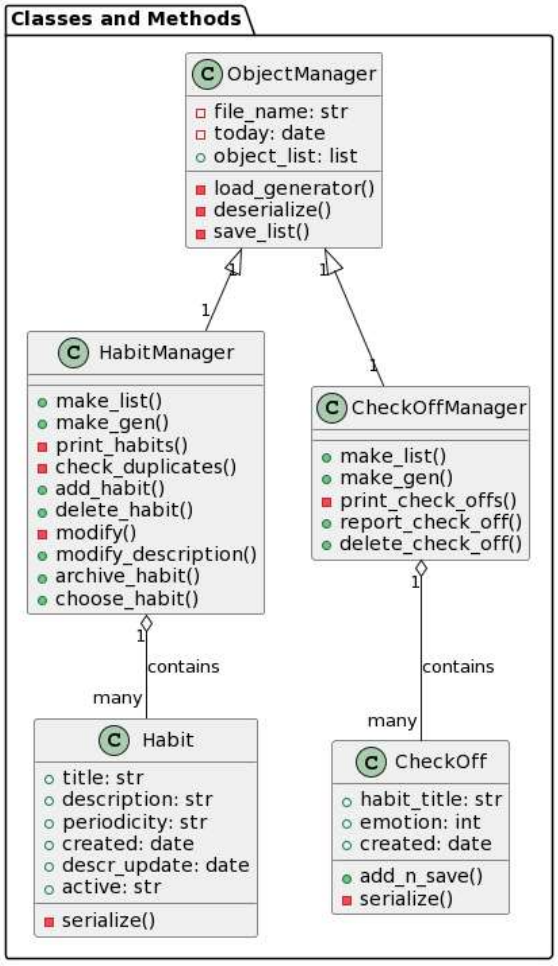
Habit tracker project abstract.

Project goal was to build the backend of a habit tracker app using Python.

Command Line Interface (CLI) with simple menus is used to call classes and their methods for managing habits and check-offs (create, delete, archive, modify, save and load). Data is stored in JSON files. Users also can access dashboard functions generating habits statistics.

Below is a UML diagram of main components calling each other and data files.





There are **five classes**: two frozen data classes (Habit and CheckOff), two management classes (HabitManager and CheckOffManager) and one parent management class (ObjectManager) for common methods.

Menu and dashboard functions are calling methods directly to manage the objects or retrieve data from JSON files.

Data is streamed to a list or generator (to save memory). For some “through operations” like printing or processing the whole history of check-offs (which can be long) generators are preferred.

CheckOff also has a special method for saving a new check-off directly into the end of a JSON file without loading it also due to possibly long check-off history.

Dashboard - is a table of several statistics per each active habit. They show how old is habit, is it in streak or broken and for how long, what is the average emotion level from 0 to 5 for the last 5 check-offs and its trend (slope sign in linear

regression). If the trend is negative, users might decide to change something before they get too bored and abandon the habit.

Habit	Type	Tenure	Status	Streak	Hiatus	Max streak	Aver emo	Emo trend
Evening meditation	Daily	28	Streak	1	0	4	2	Negative
Morning meditation	Daily	35	Streak	3	0	4	3.2	Positive
Morning run	Daily	31	Broken	0	2	4	3	Positive
Evening yoga	Weekly	5	Streak	2	0	3	5	Neutral
Sweaming in pool	Weekly	5	Streak	3	0	3	4.6	Positive

Dashboard is done in a functional approach, meaning processing immutable data streamed by CheckOffManager generator. We tried several different algorithms to find max_streak, “timeit” showed that a simple “for loop” is the fastest.

The classes and methods code was tested by pytest (17 tests) with coverage 97% and functions - 54% (we focused on dashboard functions). Mypy is also complete.