Habit Tracking Application

Table of contents

[Introduction 1](#_Toc104560128)

[Overview 1](#_Toc104560129)

[Definitions 1](#_Toc104560130)

[Project Overview 1](#_Toc104560131)

[User Interaction 1](#_Toc104560132)

[Command line interface (CLI) 1](#_Toc104560133)

[Swagger UI 2](#_Toc104560134)

[Technology 3](#_Toc104560135)

[Project Structure 4](#_Toc104560136)

[Technology choices 4](#_Toc104560137)

[Assumptions 4](#_Toc104560138)

[Bibliography 4](#_Toc104560139)

# 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.

## Overview

We used Python version 3.10.3 with FastAPI to create our backend for our Habit tracking application.

## Definitions

1. ***Habit:*** Regular activity or practice that you want to create or stop e.g. *Go running every day*
2. ***Completed Habit:*** A activity or practice that you completed at a specific time e.g Running at 1pm today.
3. ***Tracked Habit:*** A habit that was completed at least once is considered a tracked habit.

# Project Overview

We created a backend with a command line interface and a Swagger UI to simplify user interaction. We have limited the input of data to the Swagger UI for simplicity. You could also use tools like Postman to interact with the REST API.

## User Interaction

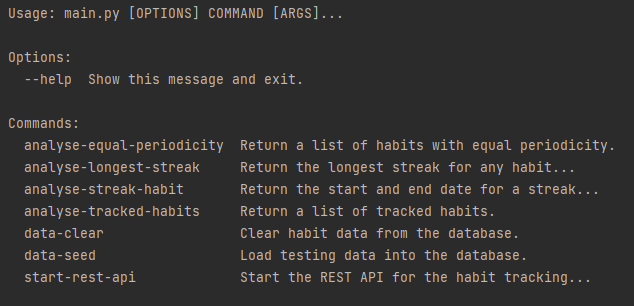
All the commands listed below should be run from the ***habits\_backend*** folder of the cloned project.

### Command line interface (CLI)

#### Help Menu

To load the command line help menu, run the following command

python main.py –help



#### Load Data

To load testing data into the database run the following command

python main.py data-seed

#### Clear Data

To clear the database of all habits and completed habits run the following command

python main.py data-clear

#### Get all tracked habits

This will return a list of all tracked habits.

python main.py analyse-tracked-habits

#### Get habits with the same period (equal periodicity)

python analyse-equal-periodicity --frequency daily

Equal Periodicity:

[{'id': 1, 'name': 'Running', 'repeated': 'Daily', 'count': 15}, {'id': 2, 'name': 'Meditation', 'repeated': 'Daily', 'count': 8}]

#### Habit with the longest run streak

python main.py analyse-longest-streak

Longest Streak any habit:

{'start': datetime.datetime(2022, 4, 27, 14, 17, 45), 'end': datetime.datetime(2022, 5, 3, 15, 57, 21), 'cnt': 8, 'habit\_id': 2}

#### Longest streak for a habit

python main.py analyse-streak-habit --habit\_id 1

Streak Details:

{'start': datetime.datetime(2022, 4, 27, 14, 17, 45), 'end': datetime.datetime(2022, 5, 2, 15, 57, 21), 'cnt': 6}

### Swagger UI

To start the Swagger UI for a visual and interactive interface run the following command.

python main.py start-rest-api

Then if you browse to the <http://localhost:8000/> URL in your browser, you should see the following interface.

Graphical user interface, application

Description automatically generated

#### Frequencies

We have pre-loaded three default frequencies for which habits can be repeated.

Shape

Description automatically generated with low confidenceGraphical user interface, application

Description automatically generated

#### Habits

## Technology

* ***Python version 3.10.3*** – Project requirement to use 3.7 or later.
* ***PyCharm 2022.1.1*** - Popular IDE / source-code editor that runs on Windows, Linux and macOS. We found that it works better than Visual Studio Code for python development.
* ***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
* ***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

## Project Structure

A picture containing text

Description automatically generated

## Technology choices

## Assumptions

1. Preload / seed data on project start-up for testing
   1. Load daily, weekly, and monthly data in the frequency table
   2. Load five predefined habits (at least one weekly and monthly habit)
   3. 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. Marcus, S (2021, July 16). Test Driven Development with pytest <https://stackabuse.com/test-driven-development-with-pytest/>
4. n.a. (n.d). [Zalando RESTful API and Event Guidelines](https://opensource.zalando.com/restful-api-guidelines/#_zalando_restful_api_and_event_guidelines)

<https://opensource.zalando.com/restful-api-guidelines/>

1. n.a. (n.d). [FastAPI](https://opensource.zalando.com/restful-api-guidelines/#_zalando_restful_api_and_event_guidelines) - SQL (Relational) Databases

<https://fastapi.tiangolo.com/tutorial/sql-databases/>