An Unexpected Development: There and Back Again with Habit Tracker

Realization of the Project

In alignment with our project objectives, we initially outlined key goals, which encompassed the development of a habit class, local file storage, Command Line Interface (CLI) integration, and datetime functionality.

Focus: on translating conceptual elements into tangible features and functionalities.
Approach: iterative, allowing for flexibility in adapting to emerging insights and challenges.
Github repository: https://github.com/AngelikaPolshikova/HabitTracker/tree/main

Deviation from Initial Goals:

However, during the course of development, some adjustments were made to better align with project requirements.

1. Notably, there was a shift from using JSON to SQLite.

This decision was influenced by two main factors: firstly, a desire to apply practical knowledge gained from working with SQLite, and secondly, the realization that debugging JSON posed unexpected challenges.

2. Additionally, the project streamlined from two classes (Habit and Habit_Counter) to a singular class, "Habit," a change made feasible and effective in implementation.

Identifying Challenges and Utilizing Resources:

Acknowledging the time constraints and limited background knowledge, we proactively identified potential challenges, risks, and opportunities for improvement. Leveraging available resources, such as:

- StackOverflow,
- the "Python Crashcourse" by Eric Mattes,
- Visual Studio Code,
- Python libraries: datetime, database, sqlite3, and time_machine,
- lectures recorded for this course.

Let's delve into what sets our app apart and why it stands out in the market.

App Overview

Our app offers a comprehensive suite of features to elevate your habit-tracking experience. With our intuitive interface, users can effortlessly **create, analayze** and **remove habits**, tailoring their routines to personal preferences.
The functionality to check off completed habits provides a satisfying sense of progress, while dynamic streak analysis allows users to delve into their long-term trends and achievements. Whether you're cultivating new habits or breaking free from old ones, our app provides a versatile platform to enhance your journey, fostering a proactive and successful approach to personal development.

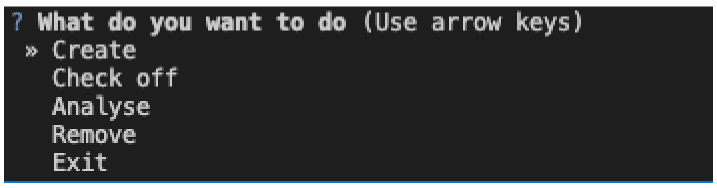
User path

Onboarding:

Users begin their journey by downloading Habit Tracker from their preferred app store (github repository, but girl can dream). The onboarding process is designed to be quick and user-friendly-ish as our app using command-line interface (CLI) accessibility, allowing new users to create an account or log in seamlessly.

Welcome Menu:

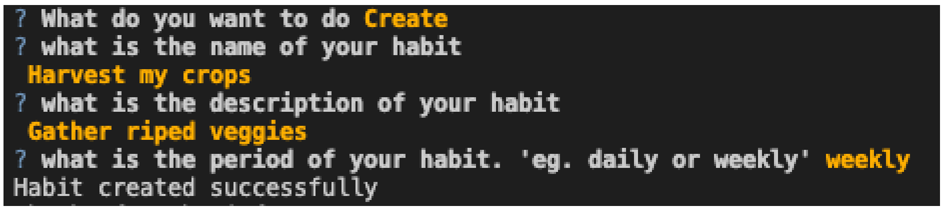
Once installed, users are guided through the process of creating their first habit.



```
? What do you want to do (Use arrow keys)
 » Create
   Check off
   Analyse
   Remove
   Exit
```

Creating Habits:

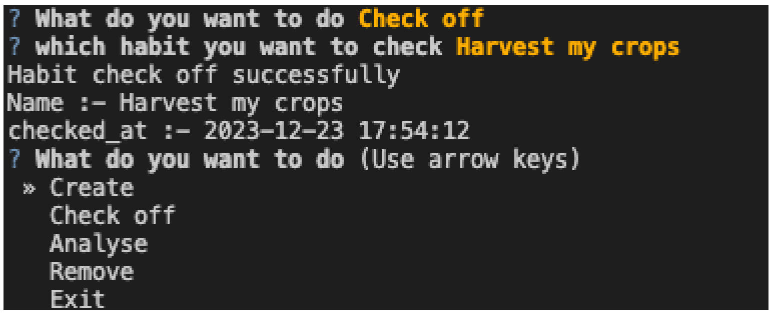
The welcome screen will promt them to input details such as habit name, frequency, and description of their habit.



```
? What do you want to do Create
? what is the name of your habit
 Harvest my crops
? what is the description of your habit
 Gather riped veggies
? what is the period of your habit. 'eg. daily or weekly' weekly
Habit created successfully
```

Performing habits:

Once created, users will be able to perform or check-off the habit by clicking “check-off” from the menu list and then inputting correct name. If the name is invalid, error message will be displayed and the user will be forwarded to the initial menu screen.



```
? What do you want to do Check off
? which habit you want to check Harvest my crops
Habit check off successfully
Name :- Harvest my crops
checked_at :- 2023-12-23 17:54:12
? What do you want to do (Use arrow keys)
 » Create
   Check off
   Analyse
   Remove
   Exit
```

Analyzing habits:

As users continue to interact with the app, they enter the reflection phase, where they analyze their habits' overall performance. The app's streak analysis and dynamic insights aid users in refining and optimizing their habits for continued success.
User is able to analyze either all of their habits, or choose between analyzing habits with the same periodicity (e.g. weekly), their strongest habit or check in with their habit logs.

Removing habits:

If the habit was created by mistake or is no longer needed - user is able to remove it entirely. [enter photo but it stopped working]

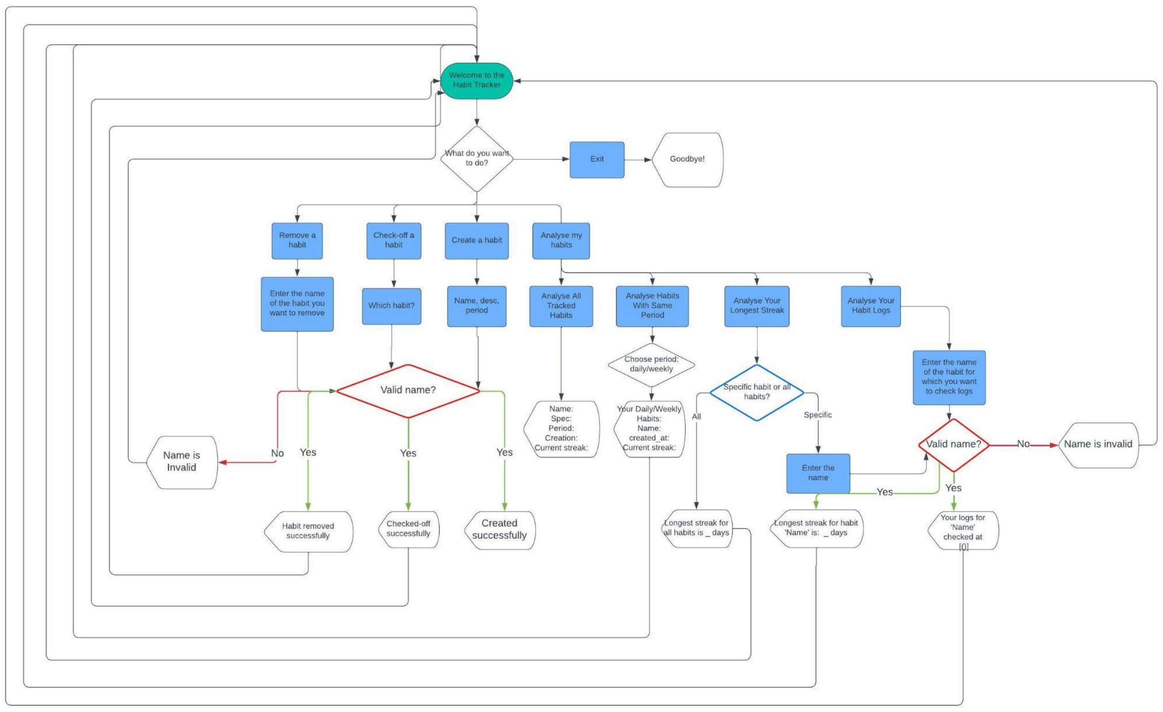
Exploring Features:

Users are encouraged to explore the app's features, including the ability to add, remove, and check off completed habits. The dynamic streak analysis provides insights into their progress, fostering motivation.The app's command-line interface (CLI) accessibility allows for efficient habit management. [enter photo but it stopped working]

Data Security and Accessibility:

SQLite database, responsible for habit storage and retrieval, ensures user’s habit data is securely stored while remaining easily accessible for a seamless experience. [enter photo but it stopped working]

The diagram below will walk you through the user path with displayed messages and decisions made along the way. Please zoom in.

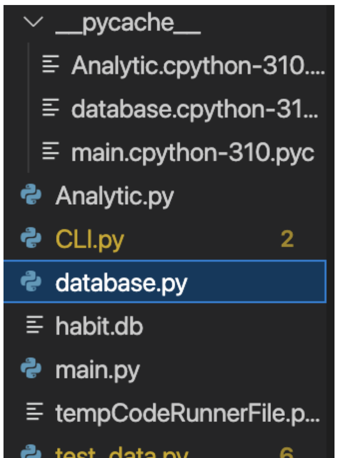


Welcome to the Habit Tracker
What do you want to do?
Exit
Goodbye!
Remove a habit
Check-off a habit
Create a habit
Analyse my habits
Enter the name of the habit you want to remove
Which habit?
Name, desc, period
Analyse All Tracked Habits
Analyse Habits With Same Period
Analyse Your Longest Streak
Analyse Your Habit Logs
Valid name?
No
Yes
Yes
Yes
Name is Invalid
Habit removed successfully
Checked-off successfully
Created successfully
Name: Spec: Period: Creation: Current streak:
Choose period: daily/weekly
Your Daily/Weekly Habits: Name: created_at: Current streak:
Specific habit or all habits?
All
Specific
Enter the name
Longest streak for all habits is _ days
Longest streak for habit 'Name' is: _ days
Enter the name of the habit for which you want to check logs
Valid name?
No
Yes
Yes
Name is invalid
Your logs for 'Name' checked at [()]

In the upcoming segment of this presentation, we will explore the backend and delve into the fundamental components that drive the functionality of this application. This will include an examination of the relationships among modules and their interconnected dynamics.

Exploring the Backend: Unraveling the Inner Workings of the App.

Development of a habit tracking system using Python is multimodular, includes files: Main.py, Database.py, CLI.py, Analytic.py, test_data.py, test.db, and habit.db. These files are interrelated and work together to create a habit tracking and management system.



```
__pycache__
  Analytic.cpython-310....
  database.cpython-31...
  main.cpython-310.pyc
Analytic.py
CLI.py                 2
database.py
habit.db
main.py
tempCodeRunnerFile.p...
test_data.py           6
```

The **Main.py file** contains the Habit class, which represents individual habits. It includes methods for creating, saving, deleting, and updating habits, as well as tracking streaks, adding events, and checking off habits on a daily or weekly basis. It interacts with the Database.py file to store and retrieve habit information.

The **Database.py file** contains functions for interacting with the SQLite database, such as creating tables for habits and habit counts, adding habits, checking off habits, updating streaks, and retrieving habit information. It also includes functions for displaying habit information and analyzing habit data. The Main.py file interacts with the Database.py file to perform database operations.

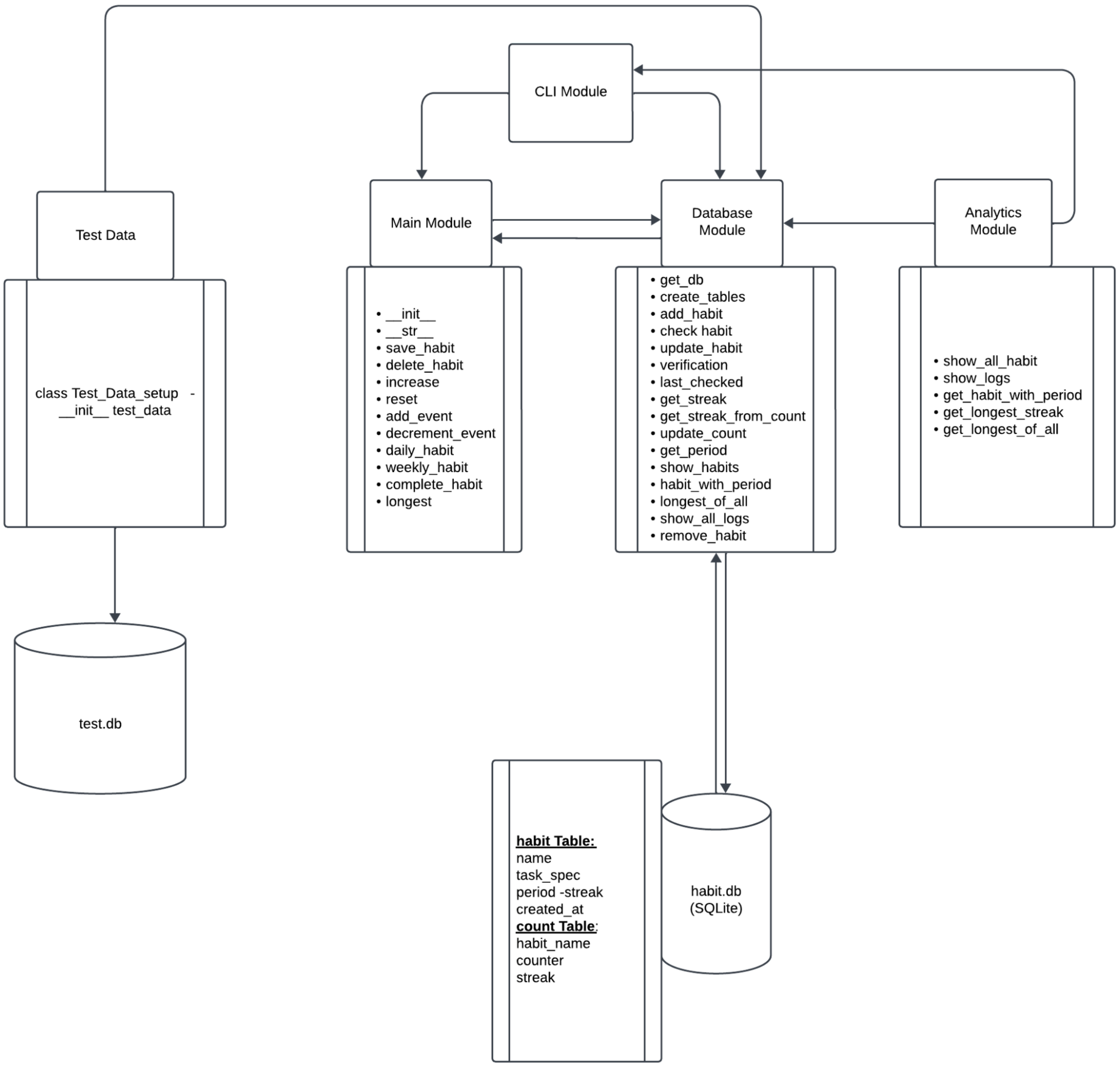
The **CLI.py file** implements a command-line interface using the questionary library. It allows users to create habits, check them off, analyze habits, and remove habits. The CLI.py file interacts with the Main.py file to perform habit-related operations and with the Analytic.py file to perform habit analysis.

The **Analytic.py** file contains functions for analyzing habit data, such as retrieving all habits, analyzing habits with the same period, finding the longest streak, and displaying habit logs. It interacts with the Database.py file to retrieve habit information for analysis and with the CLI.py file to display analysis results to the user.

The **test_data.py** file contains test data for different habits, which is inserted into the test.db database to simulate habit tracking and analysis.

In summary, the modules are interrelated in the following way: Main.py handles habit operations and interacts with Database.py for database operations, CLI.py provides a user interface for interacting with the system and interacts with Main.py for habit-related operations, Analytic.py performs habit analysis and interacts with Database.py for retrieving habit information, and test_data.py provides test data for simulating habit tracking and analysis in the test.db database.

If you have any specific questions or need further details on any aspect of the project, feel free to ask.
Below is the diagram of relationships among modules.



Reflection and Continuous Enhancement

As we conclude this phase, we enter the reflection stage, aiming to analyze the project's overall performance.
The most important **achievement**, in my opinion is that before I refreshed my terminal, the app worked. The first time CLI asked me what I wanted to do, I felt victorious. Habits were stored to the database and retrieved for analysis.
Challenges included not having experience with Python or coding overall.
Especially **grateful** for the lectures professor recorded and the book he recommended.
In the finalization phase, I aim to:

1) debug it [fingers crossed];

2) import screens to make it look nice;
3) connect test db and test_data to the rest of the app.
4) perhaps renaime to “Hobbit Tracker”, providing how many times I wrote “habit” with two “b”.

Thank you for your time, attention, and feedback.