HABIT TRACKING BACKEND APPLICATION

Finalization Phase

Submitted by:

Atheek Mohamed Rafi

Matriculation: 9212873

Enrolled in:

B.Sc. Applied Artificial Intelligence

Object Oriented and Functional Programming with Python

Finalization Phase submitted to the

IU International University

In fulfilment of the partial requirements for the course of

DLBDSOOFPP01

Instructor Name: Prof. Max Pumperla
Date: 17/02/2024

GitHub Link: https://github.com/mhdatheek136/Habit_Tracker.git

Finalization Phase - Habit Tracking Application Backend

1. Introduction.

The Habit Tracker application is crafted to assist individuals in efficiently managing and tracking their habits, be it daily, weekly, monthly, or even yearly. I will delve into the technical approach employed in developing this application, outline the challenges encountered during its creation along with the strategies to overcome them, highlight the achievements, and finally, discuss potential areas for future improvements.

2. Technical Approach

- **Programming Language:** Python's readability and efficiency played a vital role in the project's success, providing a clear and maintainable codebase.
- **Database Management:** SQLite3 was chosen for its simplicity and seamless integration with Python, facilitating efficient data persistence.
- **Object-Oriented Programming (OOP):** The core structure revolves around the Habit class, applying OOP principles for a modular and organized codebase, encapsulating essential attributes.
- **Functional Programming:** The implementation incorporates various functions, allowing users to perform key actions, ensuring code modularity, and enhancing user experience.

3. Classes and Modules

My solution comprises several classes, and modules, each serving a specific purpose.

1. db.py (Database module):
   - Responsible for managing the database.
   - Handles storage and retrieval of habit and event data.

2. habit.py (Habit Class):
   - Represents the core entity, a habit.
   - It includes attributes such, as name, description, periodicity, current streak, and creation time.
   - Utilizes the Database module to access and manipulate habit related data.

3. analytics.py (Analytics Module):
   - This module analyses habit data from the database.
   - It provides functionality to retrieve insights such as the longest streaks.

4. main.py (Entry Point):
   - The main.py file initializes the application.
   - It manages user interactions.

5. helper.py (Helper Module):
   - It acts as mediator between the main.py and other modules.
   - It is basically an extension of main.py helping to get user inputs.

4. Challenges and Pitfalls

- Data Presentation in CLI: Ensuring a clear and non-redundant display of data in the Command Line Interface (CLI) was challenging. Leveraged the prettytable library to enhance data presentation.
- Time Manipulation for Testing: Conducting intensive tests for streak calculation required time manipulation. Employed the freezegun library to freeze time during testing.
- Testing Strategies: Determining comprehensive test cases, particularly for complex methods, posed a challenge but it was essential for ensuring application reliability.
- Handling User Input Edge Cases: Managing edge cases in user input, such as misspellings, was addressed by providing choices using the questionary library instead of allowing free-form input.

5. Achievements

- Flexible Habit Scheduling: Users now have the flexibility to schedule habits daily, weekly, monthly, or yearly, ensuring a personalized experience.
- Highly Customizable Options: Users can now customize existing habits, including name, description, and periodicity, providing a high degree of personalization.
- Interactive CLI: The CLI is designed to be simple, clean, and interactive, minimizing user input. Options are presented for convenient selection.
- High Test Coverage: Critical code segments, such as habit.py and analytics.py, are thoroughly tested to achieve high code coverage.

6. Future Improvements

- Graphical User Interface (GUI): While the CLI is user-friendly, developing an appealing graphical user interface for a broader audience would be a valuable future enhancement.
- Enhanced Analytics with Visual Elements: Introducing more analytics options with graphs and visual elements to depict progress would enhance the user experience.
- Customizable Time Range: Instead of fixed options like daily, weekly, monthly, yearly, providing users with the ability to set a customizable range, such as every 10 days, would offer more flexibility.

7. Conclusion

The development journey enhanced understanding of Python, SQLite, and application development processes. The application successfully addresses habit tracking needs, promoting productivity and personal development. Valuable lessons learned in careful planning, continuous learning, and efficient time management.

Github Link: https://github.com/mhdatheek136/Habit_Tracker.git