Habit Tracker – Conception Phase

The purpose of this written concept is to provide an overview of the habit tracker app. The habit tracker app is a Python application that allows users to create new habits, track habit completion, and get statistics about their habit behavior. Users create habits they would like to track and assign the habit to be either daily or weekly. When they complete a habit, they can mark it as completed for a given day or week. The app includes five predefined habits that users can use to test the app. The app is created using object-oriented programming and consists of two classes: the Habit class and the HabitTracker class. The Habit class collects information such as the habit name, frequency (daily or weekly), date of completion, and streaks. The HabitTracker class connects to a SQLite3 database and sends and retrieves information to and from the database. Information is retrieved and sent to the database using SQL. It also creates the predefined habits and 4 weeks of data for each habit. Additionally, the HabitTracker class contains a method that allows users to see a list of their habits, a method for listing habits by frequency (daily or weekly habits), a method for showing the longest streak out of all the habits, and a method for showing the longest streak for a specified habit. The last three methods use functional programming to get the statistics. The application uses a main loop with input functions as a user interface.

The Habit Tracker app has a simple design regarding how a user can interact with the app. A user must use a CLI to use the app. They must first go to the location of the app on their computer and then execute it from there. When the script of the application is run, users will see a list of 7 actions they can perform: create a new habit, mark a habit as completed, list all habits, list habits by frequency, show the longest streak of all habits, show the longest streak for a specific habit, and exit the application. They type the number corresponding to the action they want to perform on CLI and press enter to perform it. Depending on the action, they might prompted for more information for the action to be performed. Once the user has provided all the information required to perform an action, the CLI will show the result followed by the main menu again. The user can once again perform an action by referring to the main menu. The app is closed by selecting option 7 from the main menu.

The structure and flow of this app have been designed as described for several reasons. Firstly, object-oriented programming to create the habit classes and functional programming for the analysis of habits were used because they were the required approaches for the app. Apart from being required, the object-oriented design allows the code to be more modular, easy to maintain, and reusable. Two classes were created to separate the information collected from the app functionality. Regarding functional programming, functions like map(), filter(), and reduce() are concise and efficient ways to calculate statistics. Additionally, they are easier to test because they always return the same output for a given input. SQLite3 was used to make a user's data persistent, allowing data from previous sessions to not be deleted. Although there are other ways

to make data persistent, SQLite3 was chosen because it is built into Python's standard library, is easy to use, and uses a relational structure, which is an appropriate structure for storing the kind of data that the app produces.

Figure 1. UML Class Diagram for Habit Tracker App