ABSRACT

**Habit Tracker Application**

# Introduction

The Habit Tracker application is a user-friendly tool designed to help individuals manage and track their daily, weekly, and monthly habits. In today's fast-paced world, maintaining consistent habits and routines can be challenging. That's why our application offers a simple and intuitive platform for habit management. Python serves as the backbone of our application, providing flexibility and ease of development. Tkinter, a Python GUI toolkit, is utilized to create an attractive and user-friendly interface, ensuring a seamless experience for users. For database management, we employ JSON (JavaScript Object Notation), a lightweight and easy-to-use data interchange format. JSON allows for efficient storage and retrieval of habit-related data, providing users with a reliable way to track their progress over time. With the Habit Tracker application, users can take control of their habits and routines, making it easier to stay consistent and achieve their goals in the midst of modern living.

# Technical Approach

Python, a versatile and high-level programming language celebrated for its readability and efficiency, served as the foundation for developing the Habit Tracker application. The wealth of frameworks and modules within the Python ecosystem expedited the construction of the application, enabling rapid development and deployment.

We opted for JSON, a lightweight disk-based data storage format that operates without the need for a separate server process, for data persistence. Habit-related information, including their names, frequencies, and completion statuses, was stored using JSON. The main.py file serves as the entry point of the application,

The application’s database is locally managed by two simple JSON files contained in the “data” folder so there is no need for establishing connections with online servers. It’s important to mention that all credentials have no encryption or follow any security protocols whatsoever as this application was not built for commercial distribution but rather for academic purposes. However, having said this, it would be easy to modified the database module to implement a more robust database and security protocols.

The GUI is initiated and the event loop starts where the graphical user interface (GUI) is constructed and the interaction with the user begins. Tkinter, a Python binding to the Tk GUI toolkit, was selected for its simplicity and usability in crafting the GUI. Tkinter facilitated the creation of an interface that is user-friendly and enhances the overall user experience.

In terms of functionality, the application offers features to create new habits, modify existing ones, remove habits, display all habits, filter habits by frequency, and access habit analytics for deeper insights. These analytics include identifying the longest streak among all habits and determining the highest completion rate among all habits.

# Component Interaction and Communication

The interaction between the different components of the Habit Tracker application is illustrated using a component diagram. Figure 1 showcases the high-level components and their relationships, depicting how they communicate and exchange data. The communication and data flow of habit tracker is listed below:

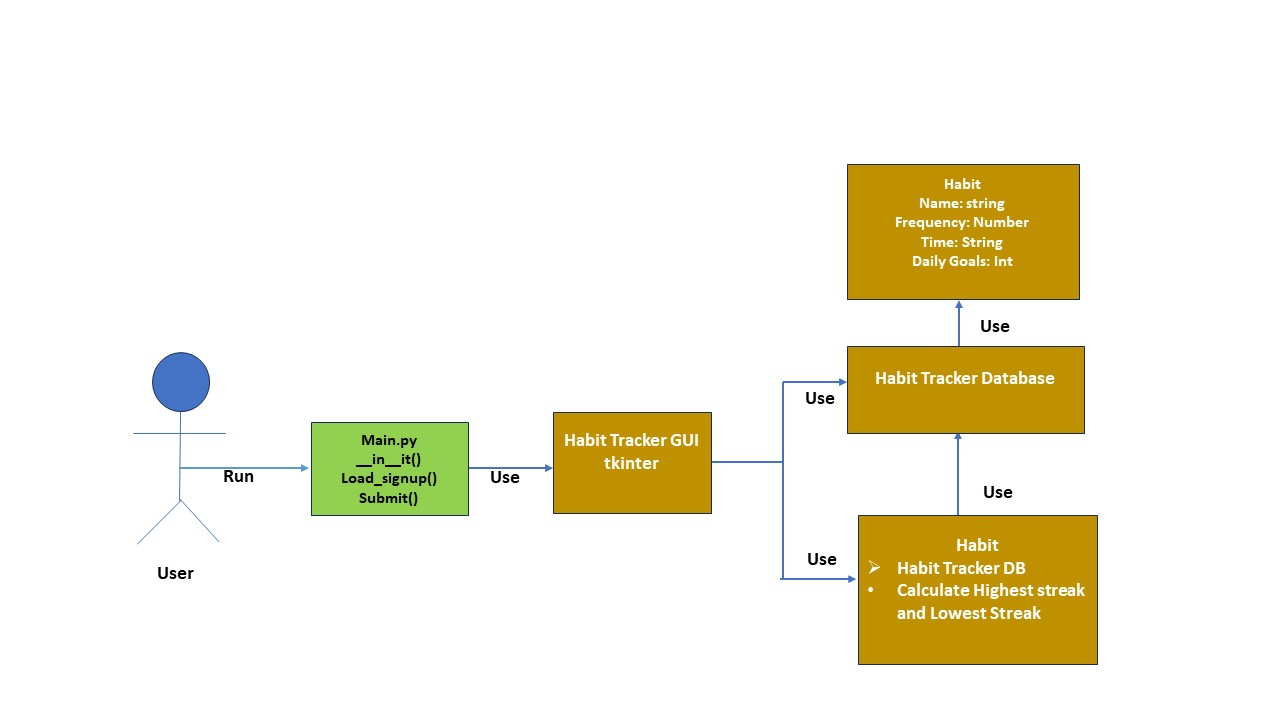


Figure 1.1: Component Diagram of Habit Tracker

1. The user opens the application by running the main.py file.
2. Main.py having functions like \_\_init\_\_(), Load\_signup(), submit() and also importing the packages here.
3. Here we use python library called tkinter used for creating graphical user interfaces (GUIs). It provides a set of tools and widgets for building desktop applications with graphical elements such as windows, buttons, text boxes, labels, and more.
4. Habit tracker database storing name of the habit and frequency, time etc….

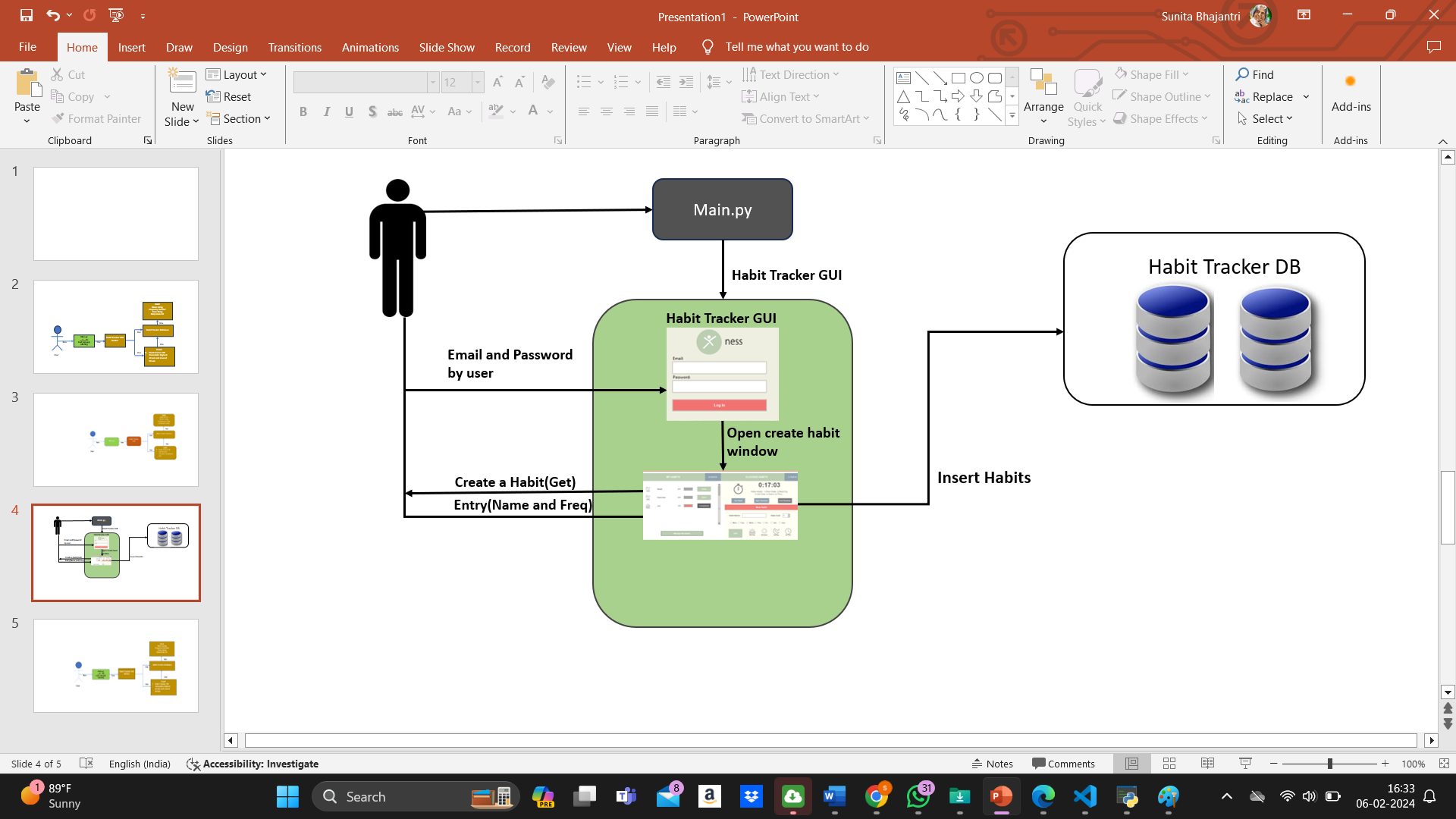


Figure 1.2: Component interaction for creating a new habit

1. After running the main.py file the application open with login page. Here we providing user credential like email and password and enters into the main page.
2. The user can add the new habit by providing habit name, frequency, time and days and click on the add button. The newly created habit is goes and stored inside the database
3. The user can select the "Manage my Habits" button to view all created habits. “Habit” class communicates with “Database” to retrieve all habit data.
4. “Habit class” utilizes “Database” to fetch habit records, calculate completion rate and compare completion rates with other habits.
5. In this project we use clock why because to get the alert. Example when we set frequency of Drinking water as 1min. After completion of 1 min it will give the pop like you should Drink Water now.
6. And also we use two buttons like Habit status for checking status of the habit and clock status for checking status of the clock.

**Usage**

**Login Page/Signup Page:**

* This window allows the user log in into his or her account. New users can create an account by clicking on the “New? Sign Up!” button. Follow instructions displayed to properly create a new user.

**My Habits Panel:**

* This panel displays the habits pending for every specific day of the week the user is using the app. Clicking on the “Done” button will register one completion cycle for the habit. Each specific habit needs to be completed as many times in a day as the pre-defined daily goal. Once the user has hit the daily goal for any specific habit, the “Done” button for that habit will change to “Completed” and will be disabled. To visualize the data and insights gathered click the “H STAST!” button.

**Manage My Habits Panel:**

* This panel displays the ALL habits defined by the user. The user can check all parameters for every single active habit. The parameter displayed are: Time, Habit Name, Daily Goal, and Frequency. Each habit listed has a “Delete” button. If clicked, the user will permanently delete all data associated with the selected habit.

**Clocked Habits Panel:**

* This panel is used to track habits within a timed session. The user can user the “Set Habit” button to define new clocked habits. To start a session, click on the “Start Session” button. Ness will push a notification every time the stopwatch hits the targeted frequency for each active habit. To user can then select to either skip the habit or mark it as completed. To end the session, click the “End Session” button. To visualize the data gathered from the last 10 sessions, click the “C STAST!” button.

**New Habits Panel**

* This panel allows the user to interactively define and create a new habit. The user must enter the habit’s name, its targeted daily frequency (i.e., the number of times the habits should be done on a daily base), the days of the week the user wishes to do the habit, and a general time frame for completion (i.e., morning, afternoon, evening, all time). Once all parameters are defined, the user can proceed to press the “Add” button and the habit will be added to the database.

# Challenges and Pitfalls

Through the challenges and learning experiences encountered during the development of the Habit Tracker application, my understanding of application development and problem-solving has significantly deepened. The initial and most daunting phase involved conceptualizing and architecting the application. Every conceivable variable and potential obstacle had to be foreseen and meticulously addressed. This entailed decision-making on the requisite number of classes, defining their functionalities, and establishing seamless interactions between them.

The application was ultimately constructed around key classes, each dedicated to specific functionalities: database management, habit definition, and graphical user interface management. Selecting the appropriate database was another pivotal choice. JSON was selected for its straightforwardness and seamless integration with Python. Nevertheless, structuring the tables and columns within the database demanded thoughtful consideration. As the project advanced, the necessity for supplementary columns emerged to effectively store and retrieve all required data. The refinement of the database schema was an iterative process, necessitating frequent adjustments, which proved to be an unforeseen challenge in the project's early stages.

Looking back, certain challenges and obstacles that arose during the project were unforeseen at its outset, ultimately resulting in an extension of the project timeline. Nonetheless, these hurdles served as invaluable learning opportunities and played a pivotal role in delivering a fully functional and dependable application. Addressing these challenges and implementing solutions contributed significantly to enhancing my proficiency in Python, JSON, Tkinter, and the broader application development process.

# Achievements

During the development of the Habit Tracker application, numerous features were integrated that I am particularly proud of. These functionalities not only augment the application's utility but also enrich the user experience, rendering habit tracking a seamless and user-friendly endeavor.

1. **Adaptable Habit Scheduling:** A notable feature of the Habit Tracker application is its versatile habit scheduling. This feature enables users to customize their habits according to various frequencies, including daily or weekly. Such adaptability accommodates diverse habits and individual schedules, ensuring a highly personalized experience for each user.
2. **Engaging GUI:** The graphical user interface (GUI) of the Habit Tracker application stands as a notable accomplishment. Despite encountering challenges while working with Tkinter, the ultimate outcome was a sleek, uncluttered, and dynamic interface. The GUI offers an intuitive means for users to engage with the application, rendering habit tracking visually appealing and uncomplicated.
3. **Habit Notifications:** A standout feature that I take pride in is the implementation of daily habit reminders. The application prominently showcases habits scheduled for completion on the current day, using a highlighted section within the window to provide a clear and unmistakable reminder for users. This functionality guarantees users won't overlook their daily habits, thereby assisting in the cultivation of consistent and positive habits.
4. **Efficient Data Management:** Implementing efficient data management using SQLite or JSON for storing habit data, ensuring reliability and performance even with large datasets.
5. **Reminder System:** Developing a reminder system that notifies users of upcoming habits or overdue tasks, helping them stay on track with their goals.

# CONCLUSION

# Users can chart their behaviours and track their advancement over time using the Habit Tracker program in a straightforward and efficient manner. Users can easily download and start the application, create new habits, inspect current habits, and analyse habit analytics. In this application, we have created the application for Habit management. Users can effortlessly download and initiate the application, establish new routines, inspect current habits, and analyze habit status.

A user-friendly graphical user interface (GUI) developed with Tkinter, a robust database system, and the ability to calculate habit analytics such as completion rate, shortest streak, and longest streak are crucial features of the Habit Tracker application. These functionalities enable users to gain deeper insights into their habits, empowering them to make informed decisions aimed at improving their daily routines.The application's functionality is well-supported by the selection of Python as the programming language, database as Json file, and Tkinter as the GUI library. The application's functionality is effectively bolstered by the choice of Python as the programming language, utilizing a JSON file as the database, and employing Tkinter as the GUI framework.