UNIVERSITY OF WESTMINSTER

INFORMATICS INSTITUTE OF TECHNOLOGY

UNIVERSITY OF WESTMINSTER

Software Development I

4COSC006C

Name – S. Promodi Kaushani Silva

UOW Number – 20827539

IIT Number – 20230105

Module Leader - Mr. Guhanathan Poravi

Assignment Type – Individual course work

Course work name – Enhanced Personal Finance Tracker (GUI Implementation with Tkinter and OOP)

I.ABSTRACT

This project enhances the Personal Finance Tracker by incorporating a graphical user interface (GUI) using Tkinter and object-oriented programming (OOP) concepts. The GUI displays information from a JSON file, implements search and sorting functionalities, and ensures user-friendliness and robustness. It aims to deepen understanding of Python, dictionaries, file I/O, and GUI development, reinforcing essential concepts in data manipulation and file management.

II.ACKNOWLEDGEMENT

I would like to express my sincere appreciation to Module leader Mr. Guhanathan Poravi and my tutorial instructors Miss. Sandunika Rasanjalee for their guidance and support throughout the duration of this project. Their expertise and insights have been invaluable in shaping the direction and quality of my work. I am grateful to my colleagues for their valuable feedback and encouragement, which have been instrumental in refining my ideas and approaches. Lastly, I am thankful to my family and friends for their unwavering support and understanding during these endeavors. This project would not have been possible without the contributions and support of these individuals and organizations. Thank you all for your invaluable assistance.

TABLE OF CONTENTS

III.Table of Figures

1. INTRODUCTION.

The enhanced Personal Finance Tracker project aims to create a user-friendly graphical interface using Tkinter and object-oriented programming (OOP) concepts in Python. This version will allow users to manage and analyze financial transactions more effectively by incorporating features such as loading transactions from a JSON file, a search function, and a sorting feature.

The GUI will be designed with classes and objects to encapsulate components, ensuring a clear and intuitive interface with labels and buttons for various functionalities. Upon invocation, the GUI will load transactions from the provided JSON file, integrating the JSON file format for data handling.

The search functionality will enable users to search for transactions based on attributes like date, amount, or type of expense, with filtering to display only matching transactions. Additionally, a sorting feature will be implemented, allowing users to click on a column heading in the transaction display table to sort the data based on that column, toggling between ascending and descending order.

Overall, the project aims to enhance the user experience by providing a robust and user-friendly application for managing personal finances.

2. DESIGN DECISIONS.

- Transactions Dictionary: The main logic of the application is based on a dictionary called transactions. This dictionary stores transactions categorized by their expense type. For each expense type, a list of transactions is maintained, where each transaction is represented as a dictionary with keys for the amount and date.
- File Handling: The application provides functionality for loading transactions from a JSON file (transactions.json) using the load_transactions() function and saving transactions to the same file using the save_transactions() function. Additionally, the read_bulk_transactions_from_file() function allows the user to read bulk transactions from a text file and add them to the transactions dictionary.
- User Input Validation: The code includes input validation for various user inputs, such as ensuring a valid date format is entered when adding or updating transactions.
- CRUD Operations: The application provides functionality for creating (adding) new transactions, reading (viewing) existing transactions, updating existing transactions, and deleting transactions.
- Summary Display: The display_summary() function calculates and displays the total expenses, as well as the category-wise expenses.
- Transaction Search: The search_transactions() function allows the user to search for transactions based on the category.

3.CLASS STRUCTURES.

- Transaction Class **:** This class represents a single transaction. It has attributes for the date, transaction type, description, and amount. The class is used to create transaction objects, which can be easily manipulated and displayed in the GUI.

```
class Transaction:
    def __init__(self, date, transaction_type, description, amount):
        self.date = date
        self.transaction_type = transaction_type
        self.description = description
        self.amount = amount
```

Figure 1 Transaction Class

- FinanceTrackerGUI Class**:** This class is responsible for creating and managing the graphical user interface using the tkinter library. It inherits from the tk.Tk class and contains methods for creating the GUI elements, loading and displaying transactions, searching transactions, sorting transactions, and handling various events

  The FinanceTrackerGUI class has various methods for creating GUI elements (e.g., create_widgets()), loading and displaying transactions (load_transactions(), display_transactions()), searching transactions (search_transactions()), sorting transactions (sort_by_column(), sort_transactions()), and handling other events.

```
class FinanceTrackerGUI:
    def __init__(self, root):
        self.root = root
        self.root.title("Personal Finance Tracker")
        self.create_widgets()
        self.transactions = self.load_transactions("transactions.json")
        self.display_transactions(self.transactions)
        self.sort_column = None
        self.sort_reverse = False
```

Figure 2 FinanceTrackerGUI Class

4.FUNCTIONALITY OF THE GUI.

- Main Title: Displays a welcoming title for the application.
- Table and Scrollbar: Displays transactions in a table format with a vertical scrollbar for navigation.
- Search Bar and Button: Allows users to search for transactions by category or amount.
- Sort Combobox and Button: Allows users to select and apply sorting criteria for transactions.
- Refresh Button: Refreshes the transaction table to display the latest data.
- Column Headings: Clicking on column headings (Category, Date, Amount) sorts the transactions based on that column in ascending or descending order.

5.RUNNING THE APPLICATION.

- Install the required dependencies by running the following command: pip install tkinter
- Save both cw part B.py and sample_code_1.py files in the same directory.
- Run the cw part B.py script using python cw part B.py.
- The application will prompt you to choose an input method: 'B' for bulk input, 'M' for the main menu, or 'G' for opening the GUI.
- If you choose 'B', you will be prompted to provide the name of a text file containing bulk transactions in the format expense_type, amount, date (one transaction per line).
- If you choose 'M', the main menu will be displayed with various options for managing transactions.

- If you choose 'G', the GUI will open, allowing you to view, search, sort, and manage transactions through a graphical interface.
- Dependencies: The application requires the `tkinter`, `ttk`, `messagebox`, `json`, and `datetime` modules, which are included in the standard Python library.
- Running the Script: You can run the script by executing it using a Python interpreter. For example, if your script is saved as `finance_tracker.py`, you can run it from the command line or terminal using the command `python finance_tracker.py`.
- Initializing the GUI: Upon running the script, a main Tkinter window (`root`) is created, and an instance of `FinanceTrackerGUI` is created, passing the main window as an argument. This instance sets up the GUI layout and widgets.
- Loading Transactions: The `load_transactions` method is called to load transactions from a JSON file (`transactions.json`). If the file exists, it loads the transactions into the application. If the file does not exist or there is an error loading it, an empty dictionary is returned, indicating that there are no transactions yet.
- Displaying Transactions: The `display_transactions` method is called to populate the transaction table (`transactions_tree`) with the loaded transactions. Each transaction is displayed as a row in the table, showing the category, date, and amount.
- User Interaction: Users can interact with the GUI by entering search queries in the search bar, clicking the search button to filter transactions, selecting a sort criteria from the dropdown combobox, and clicking the sort button to sort transactions based on the selected criteria.
- Refreshing Transactions: Clicking the refresh button reloads the transactions from the JSON file and updates the transaction table to reflect any changes.

- Closing the GUI: When the user closes the GUI window, the `mainloop()` function exits, and the script execution ends.

  In summary, running the script opens a user-friendly GUI that allows users to manage their personal finances by adding, viewing, searching, and sorting transactions.