UNIVERSITY OF WESTMINSTER

INFORMATICS INSTITUTE OF TECHNOLOGY

Module:	Software Development (4COSC006C)
Module Leader:	Guhanathan Poravi
Assignment Type:	Individual Coursework
Student ID:	20221912
UOW ID:	W2083099
Student Name:	Joshua Fernando

Table of Contents

List of figures

Acknowledgement

We would like to express our sincere gratitude to the following individuals and organizations who have supported and assisted us throughout the completion of this report: Firstly, we would like to thank the 4COSC006C panel of lecturers: Mr. Poravi Guganathan, Dinusha Ruwan Kumara, and Mr. Sulari Fernando for their guidance, encouragement, and invaluable feedback throughout the project. Their insights and expertise have been instrumental in shaping the direction of this report. We would also like to thank the Informatics Institute of Technology, who provided us with the resources to complete this report successfully. The contributions of this institute have been crucial to the success of this project. I'm also grateful to my peers for their support and encouragement. Thank you all for your invaluable contributions.

Abstract

This report presents a Python-based Personal Finance Tracker application designed to assist individuals in managing their financial transactions effectively. The application allows users to record income and expenses, view transaction history, update or delete transactions, and generate a summary of their financial activities. Developed with simplicity and usability in mind, the tracker offers intuitive user interactions and error handling mechanisms to ensure smooth operation. By empowering users to gain insights into their financial behavior and track their spending patterns, the Personal Finance Tracker aims to promote financial awareness and responsible money management practices.

What is a Personal Finance Tracker

This Personal Finance Tracker is a Python application made to assist users in effectively managing their finances. Adding, viewing, updating, and deleting financial transactions is just one of its many important features. In addition to defining the amount, category, type, and date of each transaction, users can classify transactions as either income or expenses. Users can view their financial history across several sessions thanks to the program's data persistence feature, which stores transactions in a JSON file. It also offers a summary of net income, total expenses, and total revenue to give customers a brief picture of their financial situation. This Finance Tracker provides a straightforward yet efficient way for people to monitor and manage their finances.

How to use the Personal Finance Tracker

To use the Personal Finance Tracker program, start by running the script. Upon execution, you'll be presented with a main menu offering six options:

1. **Add Transaction**: Select this option to input a new transaction, providing details such as amount, category, type (Income or Expense), and date.

2. **View Transactions**: Choose this option to see a list of all recorded transactions, including their details.

3. **Update Transaction**: Opt for this choice to modify an existing transaction. You can select which transaction to update and then provide the new details.

4. **Delete Transaction:** Use this option to remove a transaction from the records. You'll be prompted to choose the transaction you wish to delete.

5. **Display Summary:** Select this option to view a summary of all transactions, including total income, total expenses, and net income.

6. **Exit:** Choose this option to exit the program**.**

To set up the Personal Finance Tracker program, follow these instructions

1. **Download the Python Interpreter**

Ensure you have Python installed on your computer. You can download Python from the official website and follow the installation instructions for your operating system.

2. **Download the Script**

Download the Python script containing the Personal Finance Tracker program. You can either copy the script directly from the provided source or download it as a file.

3. **Run the Script**

Navigate to the directory where you saved the script using the command line or terminal. Then, run the script by executing the command python (File_Name.py)

4. **Interact with the Program**

Once the script is running, you'll see the main menu of the Personal Finance Tracker program. Follow the prompts to add, view, update, or delete transactions, as well as to display a summary of your financial activities.

5. **Persisting Data**

The program saves transaction data to a file named transactions.json in the same directory as the script

6. **Exit the Program**

To exit the program, select the "Exit" option from the main menu. This will terminate the script and end the program's execution.

By following these easy steps, you can set up and start using the Personal Finance Tracker program to manage your financial transactions efficiently.

Feature Description

i. Transaction Management

Users can add, view, update, and delete financial transactions. Each transaction includes details such as amount, category, type (Income or Expense), and date.

ii. Persistent Storage

Transaction data is saved in a JSON file (transactions.json) in the same directory as the program.

iii. Data Validation

The program validates user inputs to maintain data integrity. It ensures that amounts are positive, dates are in the correct format (YYYY-MM-DD), and choices are within the specified given range.

iv. Summary Display

Users can view a summary of their financial activities, including total income, total expenses, and net income. This feature provides users with an overview of their financial health.

v. User-Friendly Interface

The program offers a simple and intuitive interface, making it easy for users to interact with and manage their transactions.

Functionality

Describe how each of the functions works

1. Add Transaction:

Implemented the add_transaction() function to prompt the user for transaction details, validate input, and append the new transaction to the transactions list. The updated transactions are then saved to a JSON file for persistence.

2. View Transactions:

The view_transactions() function displays all transactions stored in the transactions list. If no transactions are available, an appropriate message is printed to inform the user.

3. Update Transaction:

The update_transaction() function allows users to update existing transactions by selecting the index of the transaction to be updated. After validating user input, the specified transaction details are modified, and the updated transactions are saved to the JSON file.

4. Delete Transaction:

Implemented the delete_transaction() function to enable users to delete transactions by selecting the index of the transaction to be deleted. Upon confirmation, the selected transaction is removed from the transactions list, and the updated list is saved to the JSON file.

5. Display Summary:

The display_summary() function calculates and displays the total income, total expenses, and net income based on the transactions stored in the transactions list.

6. File Operations (Load and Save Transactions):

Defined load_transactions() and save_transactions(transactions) functions to handle loading and saving transactions to a JSON file, respectively. These functions ensure the persistence of transaction data across sessions.

7. Main Function

The main_menu() function acts as the central control mechanism of the program, presenting a menu of options to the user and executing the corresponding functionality based on their input.

Design

A nested list structure is used to manage transactions during the design phase. Each transaction is represented by a list that provides information about the transaction, including the date, amount, category, and type.

Through user prompts and input validation, the essential features adding, viewing, amending, and removing transactions are implemented.

A summary of the entire income, total expense, and balance is also shown. File operations, which load data from and save it to a JSON file, guarantee the durability of transactions. This design prioritizes clarity and simplicity, which makes it easier to implement and engage with users.

Data Structure Design

1. Each transaction will be represented as a nested list containing transaction details such as amount, category, type, and date.

2. Each transaction will be represented as a nested list with the following format [amount, category, type, date]

3. Manipulation
    - Adding a transaction: Append a new nested list representing the transaction to the main list.
    - Viewing transactions: Iterate through the main list and display each nested list representing a transaction.
    - Updating a transaction: Access the nested list representing the transaction by its index and modify the relevant elements.
    - Deleting a transaction: Remove the nested list representing the transaction from the main list based on its index.

4. Advantages
    - Simplicity: The nested list structure is straightforward and easy to understand.
    - Flexibility: Each transaction is represented as a list, allowing for easy access and manipulation of transaction details.
    - Efficiency: Lists offer efficient indexing and manipulation operations, making it suitable for managing transactions.

Pseudocode

```
Start
FUNCTION load_transactions()
    TRY
        OPEN "transactions.json" file
        RETURN transactions from the file
    EXCEPT FileNotFoundError
        PRINT "Can't find any transactions"
        RETURN empty list

FUNCTION save_transactions(transactions)
    OPEN "transactions.json" file
    WRITE transactions to the file

FUNCTION add_transaction()
    GET amount, category, transaction_type, and date
    CREATE a new transaction
    ADD the new transaction to transactions
    SAVE transactions to a file

FUNCTION view_transactions()
    IF transactions is not empty
        FOR EACH transaction in transactions
                PRINT transaction
    ELSE
        PRINT "No transactions available"s

FUNCTION update_transactions()
```

```
    IF transactions is empty
        PRINT "No transactions available"
        RETURN
    CALL view_transactions()
    GET the index of the transaction to update
    GET new amount, category, transaction_type, and date
    UPDATE the transaction at the index
    SAVE transactions to a file

FUNCTION delete_transaction()
    CALL view_transactions()
    IF transactions is not empty
        GET the index of the transaction to delete
        DELETE the transaction at the index
        PRINT "Transaction deleted successfully"
        SAVE transactions to a file
    ELSE
        PRINT "No transactions available"

FUNCTION display_summary()
    SET total_income to 0
    SET total_expenses to 0
    FOR EACH transaction in transactions
          IF transaction type is 'Income'
           ADD transaction amount to total_income
        ELSE IF transaction type is 'Expense'
           ADD transaction amount to total_expenses
     CALCULATE net_income as total_income minus total_expenses
```

```
    PRINT summary of transactions


FUNCTION input_errors(slot, message, total_transactions)
    WHILE True
        TRY
            IF slot is 'choice'
                GET and VALIDATE choice
            IF slot is 'amount'
                GET and VALIDATE amount
            IF slot is 'category'
                GET category
            IF slot is 'type'
                GET and VALIDATE transaction_type
            IF slot is 'date'
                GET and VALIDATE date
            IF slot is 'index'
                GET and VALIDATE index
        EXCEPT ValueError
            PRINT "Invalid input. Please try again."


FUNCTION main_menu()
    CALL load_transactions()
    WHILE True
          PRINT main menu options
        GET user choice
        IF choice is '1'
           CALL add_transaction()
          ELSE IF choice is '2'
```

```
        CALL view_transactions()
    ELSE IF choice is '3'
        CALL update_transactions()
    ELSE IF choice is '4'
        CALL delete_transaction()
    ELSE IF choice is '5'
        CALL display_summary()
    ELSE IF choice is '6'
        PRINT "Exiting program."
        BREAK
    ELSE
        PRINT "Invalid choice. Please try again."

IF __name__ is "__main__"
    CALL main_menu()
End
```

Test Plan Summary

The test cases were meticulously designed and executed in order to ensure the reliability and proper operation of the personal finance tracker application. All operations related to adding, examining, amending, and removing transactions were completed successfully. Testing was done extensively on both valid and invalid inputs. Robust input validation ensured that the appropriate inputs were requested from users when needed. File handling techniques, such as importing transactions from JSON files and saving them again, were assessed in order to ensure data integrity. The escape option prompted the menu to finish appropriately, and the main menu was easy to explore. Boundary cases and performance testing were also done to confirm how the application behaved with large datasets and in different settings. To ensure that the program handled exceptions in an empathetic manner.

Test Plan and Execution

1. Add Transactions Function

i. Successfully Executed

Successfully added the transaction details

Result: As Expected Status: Pass

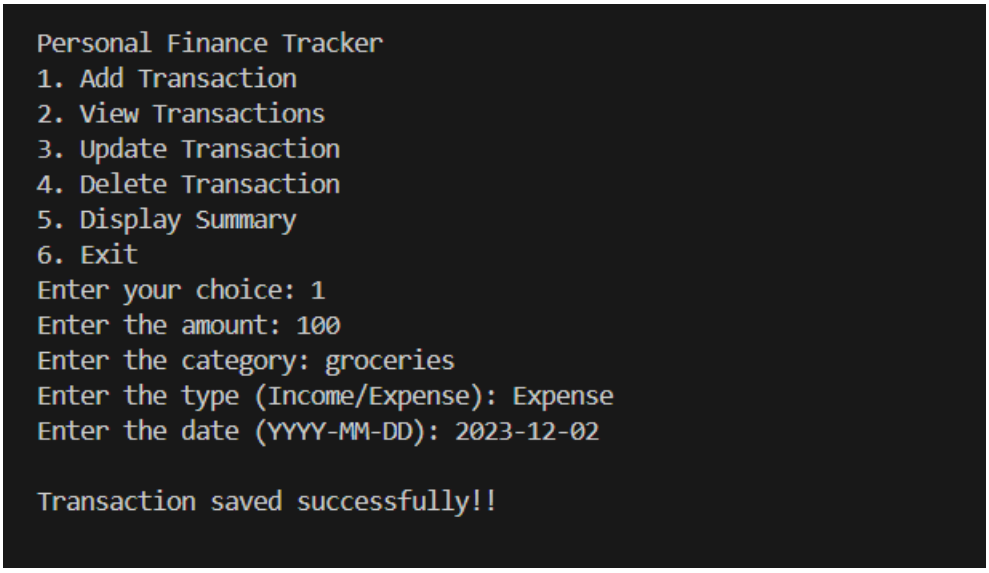

Figure 1 Add Transaction Successful

ii. JSON file where transaction data is saving

Result: As Expected Status: Pass

Figure 2 Data save to the JSON file

iii. When user input Negative amount

If user inputs a negative value for the amount, it will keep showing an error message till user inputs a correct value.

Result: As Expected Status: Pass

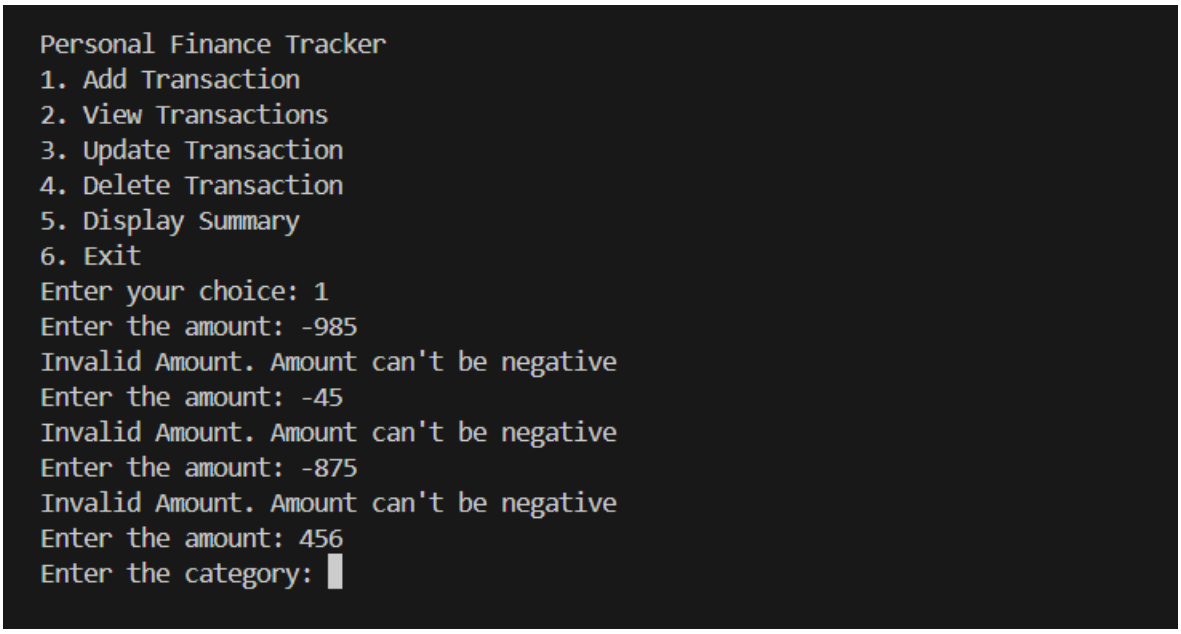

Figure 3 When user inputs a negative amount

iv. When user input Invalid inputs

If user inputs any characters or any invalid input, this will keep display an error message

Result: As Expected Status: Pass

```
Personal Finance Tracker
1. Add Transaction
2. View Transactions
3. Update Transaction
4. Delete Transaction
5. Display Summary
6. Exit
Enter your choice: 1
Enter the amount: lkh
Invalid input. Please enter a valid number for the amount.
Enter the amount: kli
Invalid input. Please enter a valid number for the amount.
Enter the amount: koi
Invalid input. Please enter a valid number for the amount.
Enter the amount: 545
Enter the category:
```

Figure 4 When user inputs an invalid input

2. View Transactions Function

i. Successfully Executed

Result: As Expected Status: Pass

```
Personal Finance Tracker
1. Add Transaction
2. View Transactions
3. Update Transaction
4. Delete Transaction
5. Display Summary
6. Exit
Enter your choice: 2
1. {'amount': 100.0, 'category': 'Groceries', 'type': 'Expense', 'date': '2024-09-05'}
```

Figure 5 Successfully Viewed

ii. When there is No Transactions available

Display an error message if there is no transactions available

Result: As Expected Status: Pass

```
Personal Finance Tracker
1. Add Transaction
2. View Transactions
3. Update Transaction
4. Delete Transaction
5. Display Summary
6. Exit
Enter your choice: 2
No transactions are available still!!.
```

Figure 6 When there is no transactions available to view

3. Update Transaction Function

i. Successfully Updated

Result: As Expected Status: Pass

```
Personal Finance Tracker
1. Add Transaction
2. View Transactions
3. Update Transaction
4. Delete Transaction
5. Display Summary
6. Exit
Enter your choice: 3
1. {'amount': 100.0, 'category': 'Tax', 'type': 'Expense', 'date': '2021-02-21'}
2. {'amount': 500.0, 'category': 'Job', 'type': 'Income', 'date': '2023-05-04'}
3. {'amount': 400.0, 'category': 'Present', 'type': 'Income', 'date': '2024-01-25'}
Enter which transaction you need to update: 2
Enter the amount: 600
Enter the category: Job
Enter the type (Income/Expense): income
Enter the date (YYYY-MM-DD): 2023-12-24

Your transaction updated successfully!!
```

Figure 7 Successfully Updated

ii. When user input Invalid index or Invalid Input

If user inputs any invalid input, this will keep display an error message

Result: As Expected Status: Pass

```
Personal Finance Tracker
1. Add Transaction
2. View Transactions
3. Update Transaction
4. Delete Transaction
5. Display Summary
6. Exit
Enter your choice: 3
1. {'amount': 100.0, 'category': 'Tax', 'type': 'Expense', 'date': '2021-02-21'}
2. {'amount': 600.0, 'category': 'Job', 'type': 'Income', 'date': '2023-12-24'}
3. {'amount': 400.0, 'category': 'Present', 'type': 'Income', 'date': '2024-01-25'}
Enter which transaction you need to update: 4
Invalid index. Please enter a number between 1 and 3.
Enter which transaction you need to update: 6
Invalid index. Please enter a number between 1 and 3.
Enter which transaction you need to update: kj
Invalid input. Please try again.
Enter which transaction you need to update: lok
Invalid input. Please try again.
Enter which transaction you need to update:
```

Figure 8 When user inputs invalid inputs

iii. No Transactions Available for Update

Display an error message if there is no transactions available

Result: As Expected Status: Pass

```
Personal Finance Tracker
1. Add Transaction
2. View Transactions
3. Update Transaction
4. Delete Transaction
5. Display Summary
6. Exit
Enter your choice: 3
Yet there are no transactions available for update
```

Figure 9 When there is are no transactions to update

4. **Delete Transaction**

i. Successfully Deleted

Result: As Expected Status: Pass

```
Personal Finance Tracker
1. Add Transaction
2. View Transactions
3. Update Transaction
4. Delete Transaction
5. Display Summary
6. Exit
Enter your choice: 4
1. {'amount': 500.0, 'category': 'Job', 'type': 'Income', 'date': '2024-01-20'}
2. {'amount': 100.0, 'category': 'Tax', 'type': 'Expense', 'date': '2024-02-24'}
Enter which transaction you want to delete: 2

Transaction deleted successfully!!
```

Figure 10 Successfully Deleted

ii. No Transactions Available

Display an error message if there is no transactions available

Result: As Expected Status: Pass

```
Personal Finance Tracker
1. Add Transaction
2. View Transactions
3. Update Transaction
4. Delete Transaction
5. Display Summary
6. Exit
Enter your choice: 4
No transactions are available still!!.
No transactions available so far!!
```

Figure 11 When there are no transactions available to delete

5. Display Summary Function

i. Successfully Executed

Result: As Expected Status: Pass

```
Personal Finance Tracker
1. Add Transaction
2. View Transactions
3. Update Transaction
4. Delete Transaction
5. Display Summary
6. Exit
Enter your choice: 5

Summary Of Transactions:
Total Income: Rs600.0
Total Expenses: Rs155.0
Net Income: Rs445.0
```

Figure 12 Successfully Displayed the summary

ii. No Transactions Available to Display

Display an error message if there is no transactions available

Result: As Expected Status: Pass

```
Personal Finance Tracker
1. Add Transaction
2. View Transactions
3. Update Transaction
4. Delete Transaction
5. Display Summary
6. Exit
Enter your choice: 5

Summary Of Transactions:
Total Income: Rs0
Total Expenses: Rs0
Net Income: Rs0
```

Figure 13 If there are no transactions available

6. Exit Function

i. Successfully Exit

Result: As Expected Status: Pass

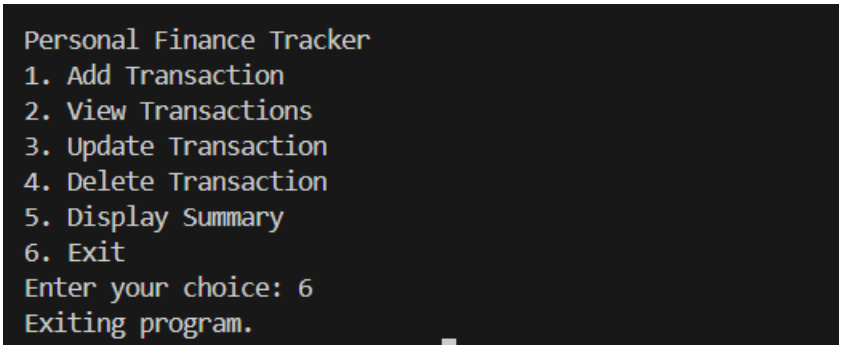

Figure 14 Successfully Exit

Test Cases

Test Case ID	Objective	Steps	Expected Output	Actual Output	Remarks
01	Adding Transaction	1. Choose option 1 from the main menu. 2. Enter valid amount, category, type, and date.	Transaction saved successfully message displayed	Display error message if user input is wrong or print a message saying the task is completed if user inputs are correct	Pass
02	View Transaction	1. Choose option 2 from the main menu.	List of transactions displayed.	Display user existing transactions. If there is no transactions available, this will display a error message saying “There is no	Pass

				transactions available”	
03	Update Transaction	1. Choose option 3 from the main menu. 2. Enter the index of the transaction to update. 3. Enter updated amount, category, type, and date.	Transaction updated successfully message displayed.	If user inputs are correct, transaction updated successfully message displayed. Else this will display various error messages describing what is the user have to do.	Pass
04	Delete Transaction	1. Choose option 4 from the main menu. 2. Enter the index of the transaction to delete.	Transaction deleted successfully message displayed.	This will delete the required transaction data set correctly. If there is no transactions made, it will display “There is no transactions” to the user	Pass

05	Display Summary	1. Choose option 5 from the main menu.	Summary of transactions displayed including total income, expenses, and net income.	This will calculate and give the exact values for Total income, Total expense and Net Income. If there is no transactions are made it will display the error message.	Pass

Conclusion

A complete solution for handling financial transactions is offered by the personal finance tracker code that is provided. Its user-friendly command-line interface makes it simple for users to add, view, change, and remove transactions and to create summaries of their financial activities. Robust error handling and input validation are incorporated into the code to ensure seamless functioning and avoid unforeseen problems. The application's usefulness is further improved by file handling functionalities that allow transaction data to be stored and retrieved. All things considered, the code is a useful resource for people who want to effectively track and handle their personal money.