UNIVERSITY OF WESTMINSTER

INFORMATICS INSTITUTE OF TECHNOLOGY

INFORMATIC INSTITUTE OF TECHNOLOGY.

4COSC006C Software Development I

Module Code & Module Name : 4COSC006C.2 Software Development I.

Module Leader : Mr. Guhanathan Poravi.

Issue Date : 6th April 2024

Student Details:

Student Name	IIT ID	UOW ID
S.R Walakuluarachchi	20230436	2083527

TABLE OF CONTENT

LIST OF FIGURES

ACKNOWLEDGMENT

I would like to express my sincere gratitude to Mr. Poor Avi Guganathan, and our tutorial lecturer Mr. Lakshan Costa for their exceptional teaching during the programming module. Their dedication, clarity, and passion for the subject made the learning experience enjoyable and insightful.

Lectures guidance played a pivotal role in shaping my understanding of programming concepts and improving my coding skills. I appreciate their commitment to fostering a positive and conducive learning environment.

Thank you, Mr. Pooravi Guganathan, and Mr. Lakshan Costa for being an inspiring and supportive instructor throughout this module.

I would also like to thank my friends who helped me to complete this assignment.

01.PROBLEM STATEMENT

The advanced version of the Personal Finance Tracker will be developed with a graphical user interface (GUI) using Tkinter and object-oriented programming (OOP) concepts. Tkinter will facilitate the creation of the GUI, with components organized into classes to adhere to OOP principles. The application will read financial transactions from a JSON file and display them in the GUI, possibly represented as rows in a table or a list. Users will have access to a search function allowing them to query specific transactions based on criteria like date, amount, or category. Additionally, the application will feature a sorting capability enabling users to organize transactions by attributes such as date, amount, or category. This version aims to provide a more intuitive and efficient way for users to manage and analyze their financial data.

02.PSEUDOCODE OF THE MAIN PROGRAM

```
BEGIN
# Import necessary libraries
import json
from datetime import datetime
import Coursework_03_SD1_20230436  # Import GUI module

# Global dictionary for storing transactions
transactions = {}

# Function for loading transactions from file
function load_transactions():
    try:
        Open 'transactions.json' file for reading
        Load transactions from file into global 'transactions' dictionary using json.load()
    except FileNotFoundError:
        Set 'transactions' to an empty dictionary

# Function for saving transactions to file
function save_transactions():
    Open 'transactions.json' file for writing
    Write transactions dictionary to file in JSON format using json.dump()

# Function for reading bulk transactions from file
function read_bulk_transactions(file_name):
    try:
        Open 'file_name' for reading
```

```
        For each line in the file:
            Parse line into category, amount, date
            Add transaction to global 'transactions' dictionary
        Print "Bulk transactions added successfully."
        Call save_transactions() to save transactions to file
    except FileNotFoundError:
        Print "File 'file_name' not found."

# Function for adding a single transaction
function add_transactions():
    Repeat until user finishes adding transactions:
        Input transaction category
        Input transaction amount
        Input transaction date
        Add transaction to global 'transactions' dictionary
        Ask user if they want to add more transactions
    Print "Transaction added successfully!"
    Call save_transactions() to save transactions to file

# Function for viewing all transactions
function view_transactions():
    If 'transactions' dictionary is empty:
        Print "No transactions found."
    Else:
        For each category and its transactions in 'transactions':
            Print category
            For each transaction in transactions:
                Print amount and date
```

```
# Function for updating a transaction
function update_transactions():
    Call view_transactions() to display all transactions
    If 'transactions' dictionary is empty:
        Return
    Input transaction category to update
    If category exists in 'transactions' dictionary:
        Display transactions for that category
        Input field to update (amount or date)
        Input index of transaction to update
        Input new value for the chosen field
        Update transaction
        Print "Field updated successfully!"
    Else:
        Print "No transactions found for this category."

# Function for deleting a transaction
function delete_transactions():
    Call view_transactions() to display all transactions
    If 'transactions' dictionary is empty:
        Return
    Input transaction category to delete from
    If category exists in 'transactions' dictionary:
        Display transactions for that category
        Input index of transaction to delete
        Delete transaction
        Print "Transaction deleted successfully!"
```

```
    Else:
        Print "No transactions found for this category."

# Function for displaying a summary
function display_summary():
    Call view_transactions() to display all transactions
    Initialize max_amount to 0
    For each transaction in 'transactions':
        If transaction amount is greater than max_amount:
            Update max_amount to transaction amount
    Print "The total of the expenses are: ", max_amount

# Main menu function
function main_menu():
    Call load_transactions() to load transactions from file
    Repeat until user chooses to exit:
        Display main menu options
        Input user choice
        Perform corresponding action based on user choice
        If user chooses to exit, print "Exiting program." and break

# Main program
if __name__ == "__main__":
    Call main_menu() to start the program
END.
```

03.PYTHON CODE OF THE MAIN PROGRAM

```
import json
from datetime import datetime
import sys
import Coursework_03_SD1_20230436

# Global dictionary for storing transactions
transactions = {}

# Functions for file handling
def load_transactions():
    global transactions
    try:
        with open('transactions.json', 'r') as file:
            transactions = json.load(file)
    except FileNotFoundError:
        transactions = {}

def save_transactions():
    with open('transactions.json', 'w') as file:
        json.dump(transactions, file, indent=4)

def read_bulk_transactions(file_name):
    global transactions
    try:
        with open(file_name, 'r') as file:
            for line in file:
                parts = line.strip().split(',')
                if len(parts) == 3:
                    category, amount, date = map(str.strip, parts)
                    category = category.capitalize()
                    amount = float(amount)
                    transactions.setdefault(category, []).append({"Amount":
amount, "Date": date})
                else:
                    print("Invalid format in line:", line)
            print("Bulk transactions added successfully.")
            save_transactions()
    except FileNotFoundError:
        print(f"File '{file_name}' not found.")

# Function for adding a single transaction
def add_transactions():
```

```
    while True:
        category = input("\nEnter transaction category: ").capitalize()
        while True:
            try:
                amount = float(input("Enter the amount: "))
                break
            except ValueError:
                print("Invalid amount. Please enter a valid number.")
        while True:
            try:
                date = input(f"Enter the date for {category} (YYYY-MM-DD): ")
                datetime.strptime(date, "%Y-%m-%d")
                break
            except ValueError:
                print("Invalid date format. Please enter as YYYY-MM-DD.")
        transactions.setdefault(category, []).append({"Amount": amount, "Date":
date})
        choice = input("Do you want to add more transactions? (Y/N): ").upper()
        if choice != "Y":
            break
    print("\nTransaction added successfully!")
    save_transactions()

# Function for viewing all transactions
def view_transactions():
    if not transactions:
        print("No transactions found.")
    else:
        for category, details in transactions.items():
            print(f"\nCategory: {category}")
            for idx, trans in enumerate(details, start=1):
                print(f"{idx}. Amount: {trans['Amount']}, Date: {trans['Date']}")

# Function for updating a transaction
def update_transactions():
    view_transactions()
    if not transactions:
        return
    while True:
        category = input("\nEnter the transaction category to update:
").capitalize()
        if category in transactions:
            print(f"{category} transactions: {transactions[category]}")
            break
        else:
```

```
            print("No transactions found for this category.")
    while True:
        field = input("\nEnter the field to update (Amount/Date): ").capitalize()
        if field in ["Amount", "Date"]:
            idx = int(input("Enter the index of the transaction to update: "))
            if 0 < idx <= len(transactions[category]):
                new_value = input(f"Enter the new {field}: ")
                transactions[category][idx - 1][field] = new_value
                print(f"{field} updated successfully!")
                break
            else:
                print("Invalid transaction index.")
        else:
            print("Invalid field. Please enter 'Amount' or 'Date'.")

# Function for deleting a transaction
def delete_transactions():
    view_transactions()
    if not transactions:
        return
    while True:
        category = input("Enter the transaction category to delete from:
").capitalize()
        if category in transactions:
            print(f"{category} transactions: {transactions[category]}")
            break
        else:
            print("No transactions found for this category.")
    while True:
        idx = int(input("Enter the index of the transaction to delete: "))
        if 0 < idx <= len(transactions[category]):
            transactions[category].pop(idx - 1)
            print("Transaction deleted successfully!")
            break
        else:
            print("Invalid transaction index.")

# Function for displaying a summary
def display_summary():
    view_transactions()
    max_amount = 0
    for category, details in transactions.items():
        for trans in details:
            if trans["Amount"] > max_amount:
                max_amount = trans["Amount"]
```

```
    print("\nThe total of the expenses are: ", max_amount)

# Main menu function
def main_menu():
    load_transactions()
    while True:
        print("\nPersonal Finance Tracker")
        print("1. Add Transaction")
        print("2. View Transactions")
        print("3. Update Transactions")
        print("4. Delete Transactions")
        print("5. Read Bulk Transactions")
        print("6. Display Summary")
        print("7. Open GUI")
        print("8. Exit")
        choice = input("\nEnter your choice: ")
        if choice == '1':
            add_transactions()
        elif choice == '2':
            view_transactions()
        elif choice == '3':
            update_transactions()
        elif choice == '4':
            delete_transactions()
        elif choice == '5':
            file_name = input("Enter the file name to read bulk transactions
from: ")
            read_bulk_transactions(file_name)
        elif choice == '6':
            display_summary()
        elif choice == '7':
            Coursework_03_SD1_20230436.main()
        elif choice == '8':
            print("Exiting program.")
            break
        else:
            print("Invalid choice. Please try again.")

if __name__ == "__main__":
    main_menu()
```

04. TEST CASES OF THE MAIN PROGRAM

Test Component	Test No	Test Input	Expected Result	Actual Result	Pass / Fail
Main Menu	1	None	Displaying the main menu with options and asking choice.	Displaying the main menu with options and asking choice.	Pass
Add Transactions	2.0	**Valid Input:** Category: Salary Amount: 1000 Date: 2024-03-17	Display "Transaction Added Successfully"	Display "Transaction Added Successfully"	Pass
	2.1	**Invalid Input:** Amount : abc	Display "Invalid amount, Please enter a valid amount"	Display "Invalid amount, Please enter a valid amount"	Pass
View Transactions	3.0	**View transactions when there are no transactions:** Transactions[]	Display "No Transactions Found"	Display "No Transactions Found"	Pass
	3.1	**View transactions when there are existing transactions:** Category: Salary Amount: 1000 Date: 2024-03-17	Category: Salary 1. Amount: 1000.0, Date: 2023-03-17	Category: Salary 1. Amount: 1000.0, Date: 2023-03-17	Pass
Update Transactions	4.0	**Valid Input:** Update an existing Transaction	Category: Salary 1. Amount: 1000.0, Date: 2023-03-17 Enter the transaction category to update: salary	Category: Salary 1. Amount: 1000.0, Date: 2023-03-17 Enter the transaction category to update: salary	Pass

			Salary transactions: [{'Amount': 1000.0, 'Date': '2023-03-17'}] Enter the field to update (Amount/Date): amount Enter the index of the transaction to update: 1 Enter the new Amount: 2500 Amount updated successfully!	Salary transactions: [{'Amount': 1000.0, 'Date': '2023-03-17'}] Enter the field to update (Amount/Date): amount Enter the index of the transaction to update: 1 Enter the new Amount: 2500 Amount updated successfully!!	
	4.1	**Invalid Input:** Index of transaction to update: 5	Displaying "Invalid transaction index"	Displaying "Invalid transaction index"	Pass
Delete Transaction.	5.0	**Valid input: Delete an existing transaction**. Index of transaction to delete: 1	Delete the selected transaction and Display "Transaction delete successfully."	Delete the selected transaction and Display "Transaction delete successfully."	Pass
Display Summary	6.0	**Display summary when there are existing transactions.** Category: Salary 1. Amount: 1000.0, Date: 2023-03-17 Category: Grocery 1. Amount: 500.0, Date: 2023-05-05	The total of the expenses are: 1000.0	The total of the expenses are: 1000.0	Pass
Exit	7.0	Option: 8	Exiting From the program	Exiting From the program	Pass

Read Bulk Transactions	8.0	Option: 5	Display bulk transactions added successfully.	Display bulk transactions added successfully.	Pass
Linking GUI to the Program	9.0	Option: 7	Opening the GUI	Opening the GUI	Pass

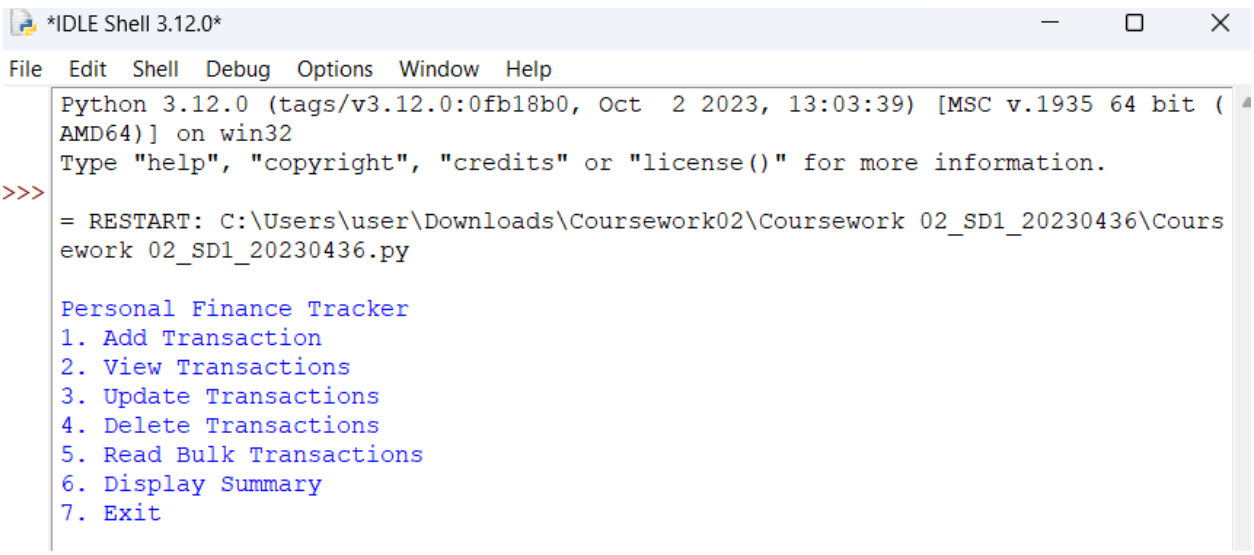

Figure 1: Test No 1

```
*IDLE Shell 3.12.0*
File  Edit  Shell  Debug  Options  Window  Help
Python 3.12.0 (tags/v3.12.0:0fb18b0, Oct  2 2023, 13:03:39) [MSC v.1935 64 bit (
AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:\Users\user\Downloads\Coursework02\Coursework 02_SD1_20230436\Cours
ework 02_SD1_20230436.py

Personal Finance Tracker
1. Add Transaction
2. View Transactions
3. Update Transactions
4. Delete Transactions
5. Read Bulk Transactions
6. Display Summary
7. Exit

Enter your choice: 1

Enter transaction category: salary
Enter the amount: 1000
Enter the date for Salary (YYYY-MM-DD): 2023-03-17
Do you want to add more transactions? (Y/N): n

Transaction added successfully!
```

Figure 2: Test No 2.0

```
Personal Finance Tracker
1. Add Transaction
2. View Transactions
3. Update Transactions
4. Delete Transactions
5. Read Bulk Transactions
6. Display Summary
7. Exit

Enter your choice: 1

Enter transaction category: salary
Enter the amount: abc
Invalid amount. Please enter a valid number.
Enter the amount:
                                                    Ln: 38  Col: 18
```

Figure 3:Test No 2.1

```
Enter your choice: 1

Enter transaction category: salary
Enter the amount: 1000
Enter the date for Salary (YYYY-MM-DD): 2023-03-17
Do you want to add more transactions? (Y/N): n

Transaction added successfully!

Personal Finance Tracker
1. Add Transaction
2. View Transactions
3. Update Transactions
4. Delete Transactions
5. Read Bulk Transactions
6. Display Summary
7. Exit

Enter your choice: 2

Category: Salary
1. Amount: 1000.0, Date: 2023-03-17

Personal Finance Tracker
1. Add Transaction
2. View Transactions
3. Update Transactions
4. Delete Transactions
5. Read Bulk Transactions
6. Display Summary
7. Exit
```

Figure 4: Test No 3.1

```
Personal Finance Tracker
1. Add Transaction
2. View Transactions
3. Update Transactions
4. Delete Transactions
5. Read Bulk Transactions
6. Display Summary
7. Exit

Enter your choice: 3

Category: Salary
1. Amount: 1000.0, Date: 2023-03-17

Enter the transaction category to update: salary
Salary transactions: [{'Amount': 1000.0, 'Date': '2023-03-17'}]

Enter the field to update (Amount/Date): amount
Enter the index of the transaction to update: 1
Enter the new Amount: 2500
Amount updated successfully!
```

Figure 5: Test No 4.0

```
Personal Finance Tracker
1. Add Transaction
2. View Transactions
3. Update Transactions
4. Delete Transactions
5. Read Bulk Transactions
6. Display Summary
7. Exit

Enter your choice: 3

Category: Salary
1. Amount: 2500, Date: 2023-03-17

Enter the transaction category to update: salary
Salary transactions: [{'Amount': '2500', 'Date': '2023-03-17'}]

Enter the field to update (Amount/Date): amount
Enter the index of the transaction to update: 5
Invalid transaction index.
```

Figure 6: Test No 4.1

```
Enter your choice: 1

Enter transaction category: salary
Enter the amount: 1000
Enter the date for Salary (YYYY-MM-DD): 2023-03-17
Do you want to add more transactions? (Y/N): n

Transaction added successfully!

Personal Finance Tracker
1. Add Transaction
2. View Transactions
3. Update Transactions
4. Delete Transactions
5. Read Bulk Transactions
6. Display Summary
7. Exit

Enter your choice: 4

Category: Salary
1. Amount: 1000.0, Date: 2023-03-17
Enter the transaction category to delete from: salary
Salary transactions: [{'Amount': 1000.0, 'Date': '2023-03-17'}]
Enter the index of the transaction to delete: 1
Transaction deleted successfully!
```

Figure 7: Test No 5.0

```
Personal Finance Tracker
1. Add Transaction
2. View Transactions
3. Update Transactions
4. Delete Transactions
5. Read Bulk Transactions
6. Display Summary
7. Exit

Enter your choice: 6

Category: Salary
1. Amount: 1000.0, Date: 2023-03-17

Category: Grocery
1. Amount: 500.0, Date: 2023-05-05

The total of the expenses are:  1000.0

Personal Finance Tracker
1. Add Transaction
2. View Transactions
3. Update Transactions
4. Delete Transactions
5. Read Bulk Transactions
6. Display Summary
7. Exit
```

Figure 8: Test No 6.0

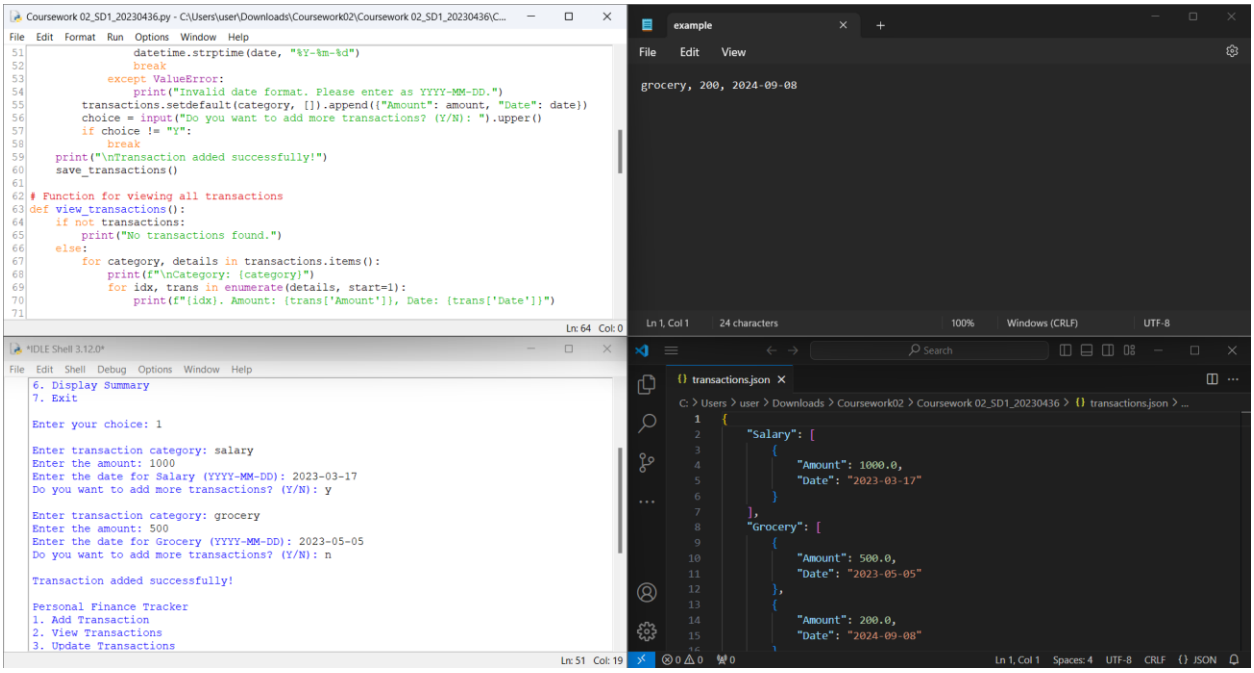

Figure 9: Test No 8.0

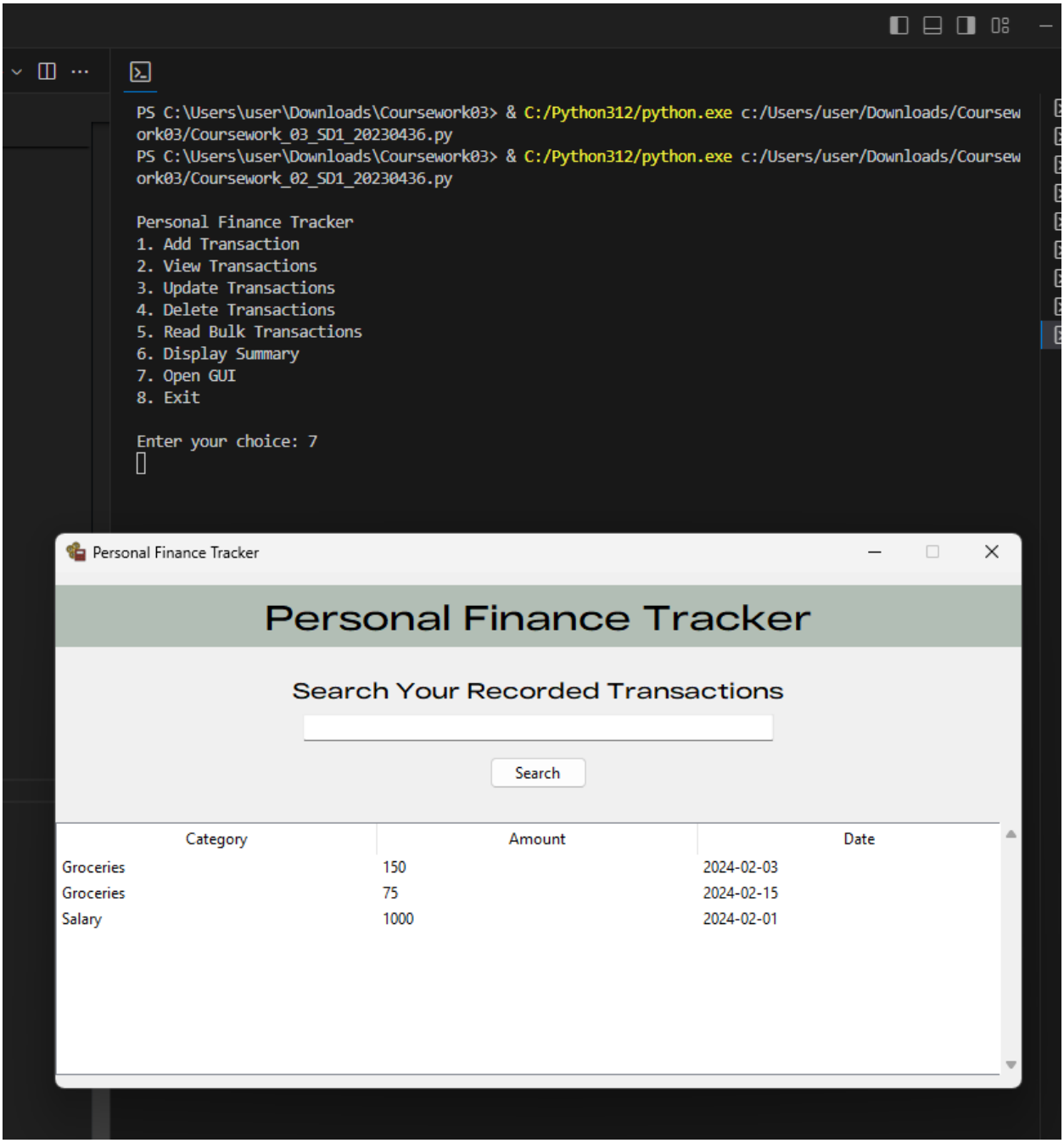

Figure 10: Test No 9.0

05. PSEUDOCODE OF THE GUI PROGRAM

```
BEGIN
Class FinanceTrackerGUI:
    Function __init__(root):
        Initialize root window
        Set window title and icon
        Set window geometry and make it not resizable
        Load transactions from JSON file
        Create GUI widgets
        Display transactions

    Function create_widgets():
        Create frame for table and scrollbar
        Create labels
        Create Treeview for displaying transactions
        Create scrollbar for Treeview
        Create search bar and button

    Function load_transactions(filename):
        Try to open the file
            Load transactions from JSON file
            Return transactions
        If file not found, return an empty dictionary

    Function display_transactions(transactions):
        Clear existing data in the table
        Loop through transactions
```

```
        Insert transaction data into the table

    Function search_transactions():
        Get search text from the entry
        Initialize an empty dictionary for searched transactions
        Loop through transactions
            If search text matches any transaction data
                Add matching transactions to searched dictionary
        Display searched transactions

    Function sort_by_column(col):
        Get current items in the treeview
        Sort the data
        Move items in the treeview according to sorted data
        Change heading to reflect sort

Function main():
    Create root window
    Initialize FinanceTrackerGUI instance
    Call display_transactions
    Start the main event loop

If __name__ == "__main__":
    Call main function
END.
```

06.PYTHON CODE OF THE GUI PROGRAM

```
import tkinter as tk
from tkinter import ttk
import json

class FinanceTrackerGUI:
    def __init__(self, root):
        self.root = root
        self.root.title("Personal Finance Tracker")
        self.root.iconbitmap('icon.ico') # icon
        root.geometry('750x400') # width x height
        root.resizable(False,False) # using this for avoiding resize the GUI
        self.create_widgets()
        self.transactions = self.load_transactions("transactions.json")

    def create_widgets(self):
        # Frame for table and scrollbar

        self.tree_frame1 = tk.Frame(self.root,background='#B2BEB5') # This line
creates a Frame widget named self.tree_frame within the self.root window
        self.tree_frame1.pack(fill = "x",pady=10) # Pady puts some space between
the button widgets and the borders of the frame and the borders of the root
window
        # fill="x" means the frame will fill the available horizontal space
within the root window.

        label1 = tk.Label(self.tree_frame1,text= "Personal Finance
Tracker",font=("rf dewi expanded semibold",20),background='#B2BEB5')
        label1.pack(ipady=5)

        self.tree_frame2 = tk.Frame(self.root)
        self.tree_frame2.pack(fill="x",pady=10)

        label2 = tk.Label(self.tree_frame2,text="Search Your Recorded
Transactions",font=("rf dewi expanded semibold",13))
        label2.pack(padx=10)


        # Treeview for displaying transactions

        self.table = ttk.Treeview(self.root, columns=('Category', 'Amount',
'Date'), show='headings')
```

```
        self.table.column('Amount', width=100)
        self.table.column('Date', width=100)
        self.table.column('Category', width=100)

        self.table.heading('Amount', text="Amount", command=lambda:
self.sort_by_column("Amount"))
        self.table.heading('Date', text="Date", command=lambda:
self.sort_by_column("Date"))
        self.table.heading('Category', text="Category", command=lambda:
self.sort_by_column("Category"))


        self.table.pack(fill="x", pady=10)


        # Scrollbar for the Treeview

        scrollbar = ttk.Scrollbar(self.table, orient="vertical",
command=self.table.yview)
        self.table.configure(yscrollcommand=scrollbar.set)

        # Pack the Treeview and scrollbar widgets

        self.table.pack(side="left", fill="both", expand=True)
        scrollbar.pack(side="right", fill="y")


        # Search bar and button

        self.search_text=tk.StringVar()
        self.search_option = ttk.Entry(self.tree_frame2,
width=60,textvariable=self.search_text)

        self.search_click = ttk.Button(self.tree_frame2,
text="Search",command=self.search_transactions)

        self.search_option.pack(padx=6, pady=6)
        self.search_click.pack(padx=6, pady=6)

        pass


    def load_transactions(self, filename):
        try:
            with open(filename,'r') as file:
```

```
                transactions = json.load(file)
                return transactions
        except FileNotFoundError:
                return{}

        except FileNotFoundError:
            return {}

    def display_transactions(self, transactions):
        self.display=transactions
        for data in self.table.get_children():
            self.table.delete(data)

        for self.keys, self.values in self.display.items():
            for self.data in self.values:
                self.table.insert("", index='end', values=(f'{self.keys}',
                                                            self.data['Amount'],
                                                            self.data['Date']))

    def search_transactions(self):
        search_text = self.search_text.get().lower()
        searched = {}

        for category, category_data in self.transactions.items():
            filtered_category_transactions = []
            for transaction in category_data:
                if any(
                    search_text in str(value).lower()
                    for value in [category, transaction["Amount"],
transaction["Date"]]
                ):
                    filtered_category_transactions.append(transaction)
            if filtered_category_transactions:
                searched[category] = filtered_category_transactions

        self.display_transactions(searched)

    def sort_by_column(self, col):
        # Get the current items in the treeview
        data = [(self.table.set(child, col), child) for child in
self.table.get_children('')]

        # Sort the data
        data.sort()
```

```
        for index, (val, child) in enumerate(data):
            self.table.move(child, '', index)

        # Change the heading to reflect the sort
        self.table.heading(col, command=lambda: self.sort_by_column(col))

def main():
    root = tk.Tk()
    app = FinanceTrackerGUI(root)
    app.display_transactions(app.transactions)
    root.mainloop()

if __name__ == "__main__":
    main()
```

07. TEST CASES OF THE GUI PROGRAM

Test Component	Test No	Test Input	Expected Result	Actual Result	Pass / Fail
View of the Personal Finance Tracker	10.0	None	Getting the basic structure of the output	Getting the basic structure of the output.	Pass
Searching the transactions in the GUI	11.0	"Groceries" in search bar	Display Transactions related to "Groceries"	Display Transactions related to "Groceries"	Pass
	11.1	"Salary" in search bar	Display Transactions related to "Salary"	Display Transactions related to "Salary"	Pass

Searching Invalid Transaction	12.0	**abcd**	Not Showing Anything	Not Showing Anything	Pass
Sorting	13.0	**Clicking the Amount Column**	Sorting the amount in Ascending order	Sorting the amount in Ascending order	Pass
	13.1	**Clicking the Category Column**	Sorting the Category in Ascending order	Sorting the Category in Ascending order	Pass
	13.2	**Clicking the Date Column**	Sorting the Date in Ascending order	Sorting the Date in Ascending order	Pass

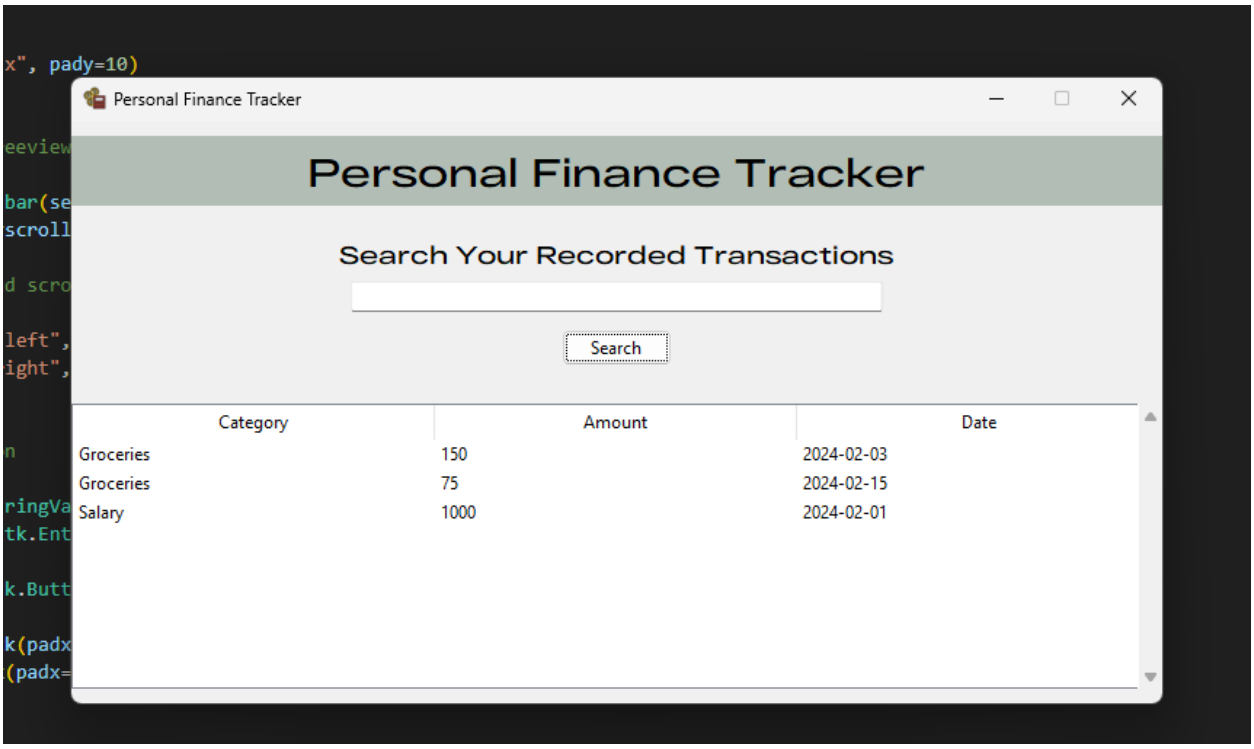

Figure 11: Test No 10.0

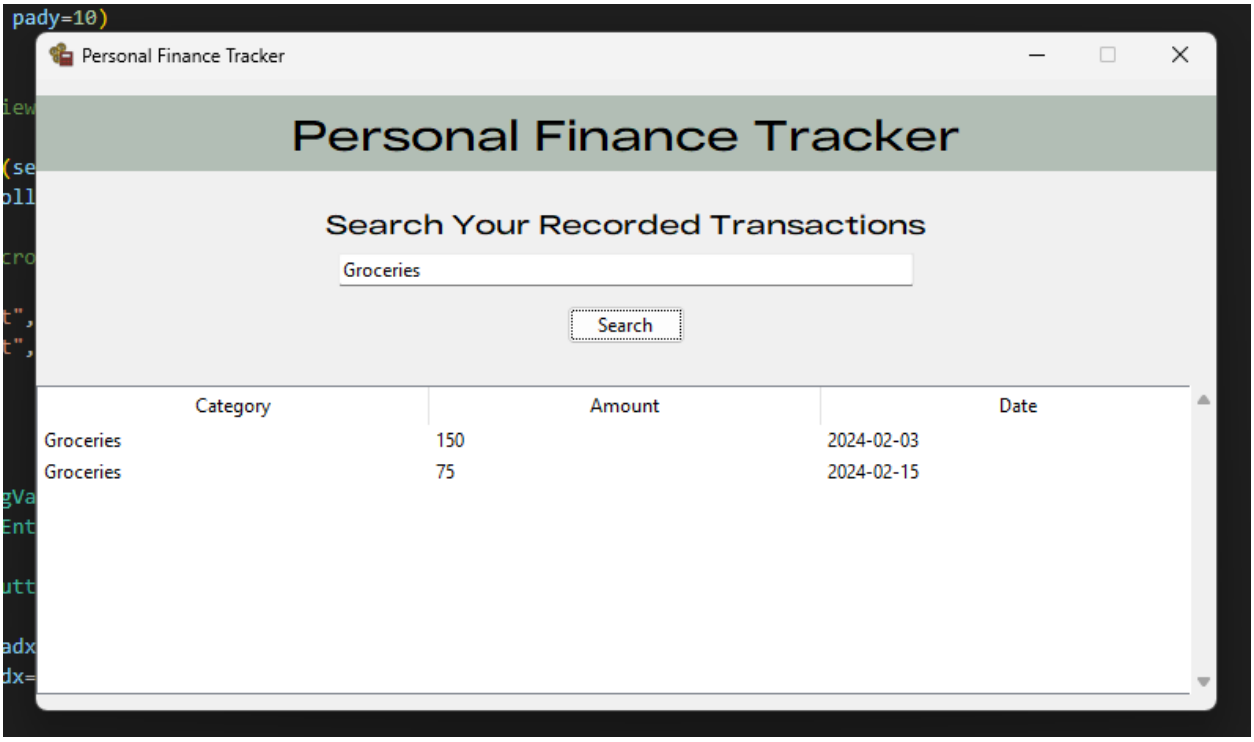

Figure 12: Test No 11.0

Figure 13: Test No 11.1

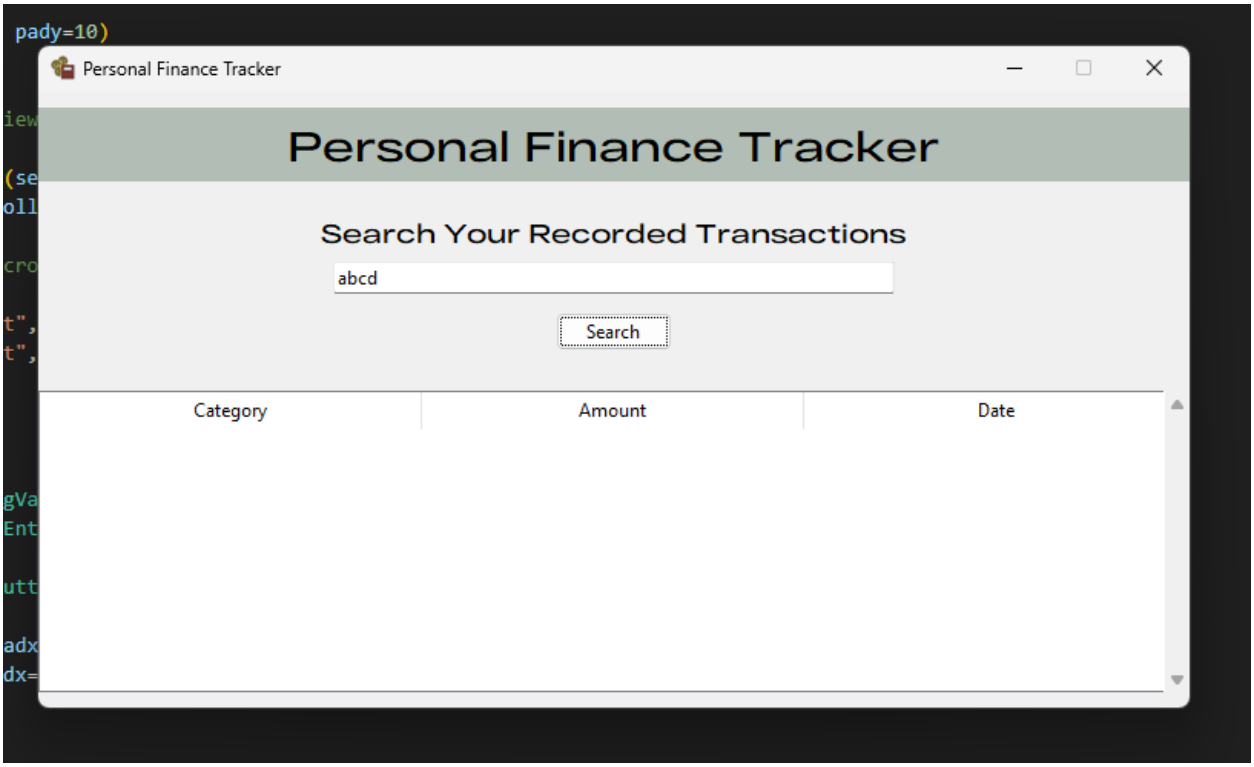

Figure 14: Test No 12.0

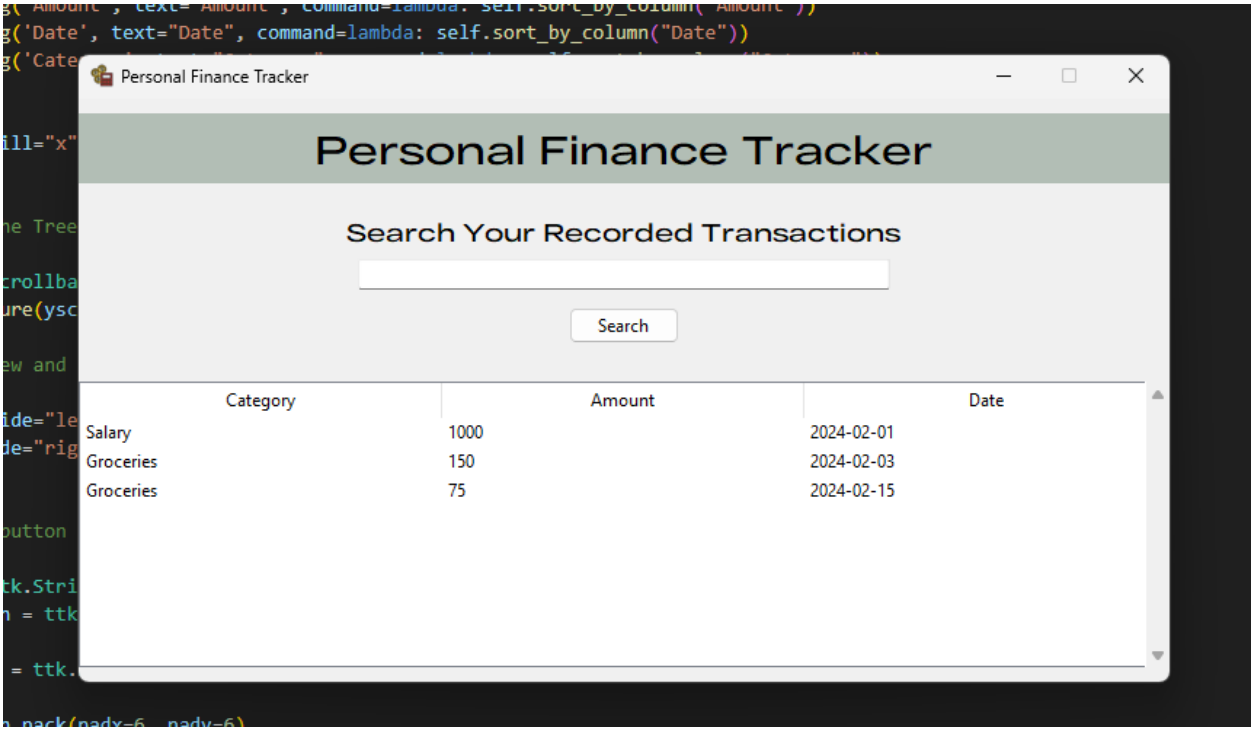

Figure 15: Test No 13.0

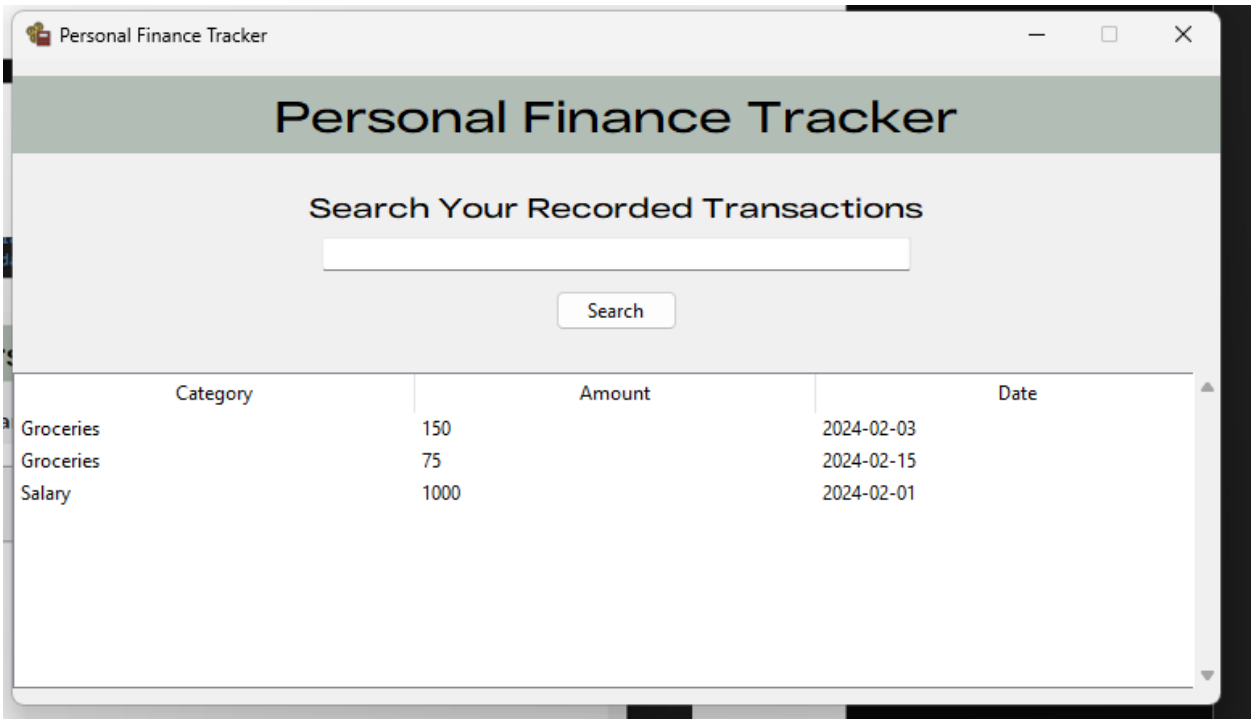

Figure 16: Test No 13.1

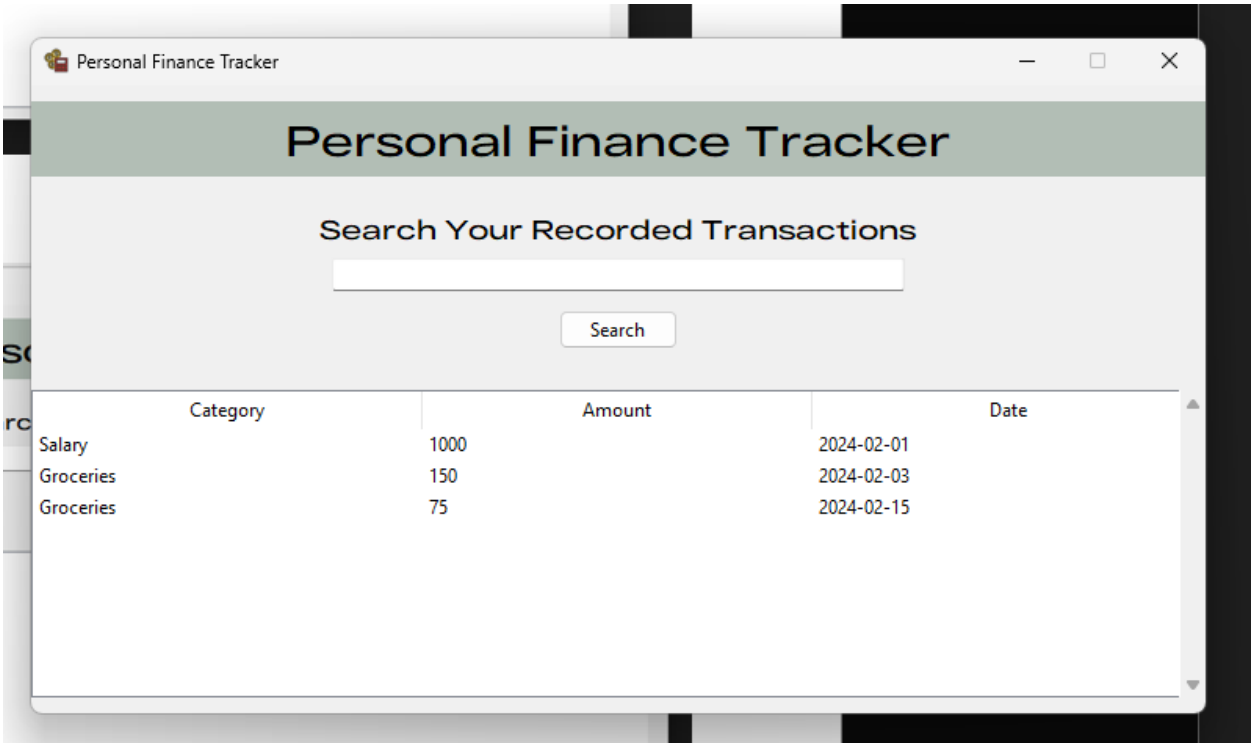

Figure 17: Test No 13.2