ABSTRACT

A personalized expense tracker is a digital tool designed to help individuals monitor, manage, and analyze their spending habits. It integrates with various financial accounts, categorizes expenses automatically, and provides insights through customizable reports and visualizations. Users can set budget goals, receive alerts for overspending, and track progress towards financial objectives. Enhanced with features like receipt scanning, expense tagging, and secure data encryption, the tracker aims to simplify personal finance management, promote financial discipline, and support informed decision-making. Its user-friendly interface and adaptive learning capabilities ensure a tailored experience that evolves with the user's financial behavior.

ABSTRACT WITH POs AND PSOs MAPPING

ABSTRACT	**POs MAPPED**	**PSOs MAPPED**
A personalized expense tracker is a digital tool designed to help individuals monitor, manage, and analyze their spending habits. It integrates with various financial accounts, categorizes expenses automatically, and provides insights through customizable reports and visualizations. Users can set budget goals, receive alerts for overspending, and track progress towards financial objectives. Enhanced with features like receipt scanning, expense tagging, and secure data encryption, the tracker aims to simplify personal finance management, promote financial discipline, and support informed decision-making. Its user-friendly interface and adaptive learning capabilities ensure a tailored experience that evolves with the user's financial behavior	**PO1 - 1** **PO2 - 1** **PO3 - 1** **PO12 - 2**	**PSO1 - 3**

Note: 1- Low, 2-Medium, 3- High

SUPERVISOR **HEAD OF THE DEPARTMENT**

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

1.1 Objective

A personalized expense tracker aims to help individuals manage their finances effectively by providing a clear overview of income, expenses, and savings. It enables users to categorize spending, set budgets, and track financial goals in real time. With customizable features tailored to individual needs, the tracker simplifies decision-making, encourages mindful spending, and promotes financial discipline. Its user-friendly design ensures seamless data entry and insightful analytics, fostering better control over personal finances while paving the way for long-term financial stability and growth.

1.2 Overview

A personalized expense tracker is a financial management tool designed to help users monitor and control their spending habits effectively. It provides an organized platform to record income and expenses, categorize transactions, and analyze spending patterns. With features like budget setting, goal tracking, and real-time updates, it empowers users to make informed financial decisions. By offering customization options, such as tailored categories and periodic reminders, the tracker adapts to individual financial needs. Its intuitive interface and insightful visualizations promote financial awareness, enabling users to achieve better financial health and long-term stability.

1.3 Python Programming Concepts

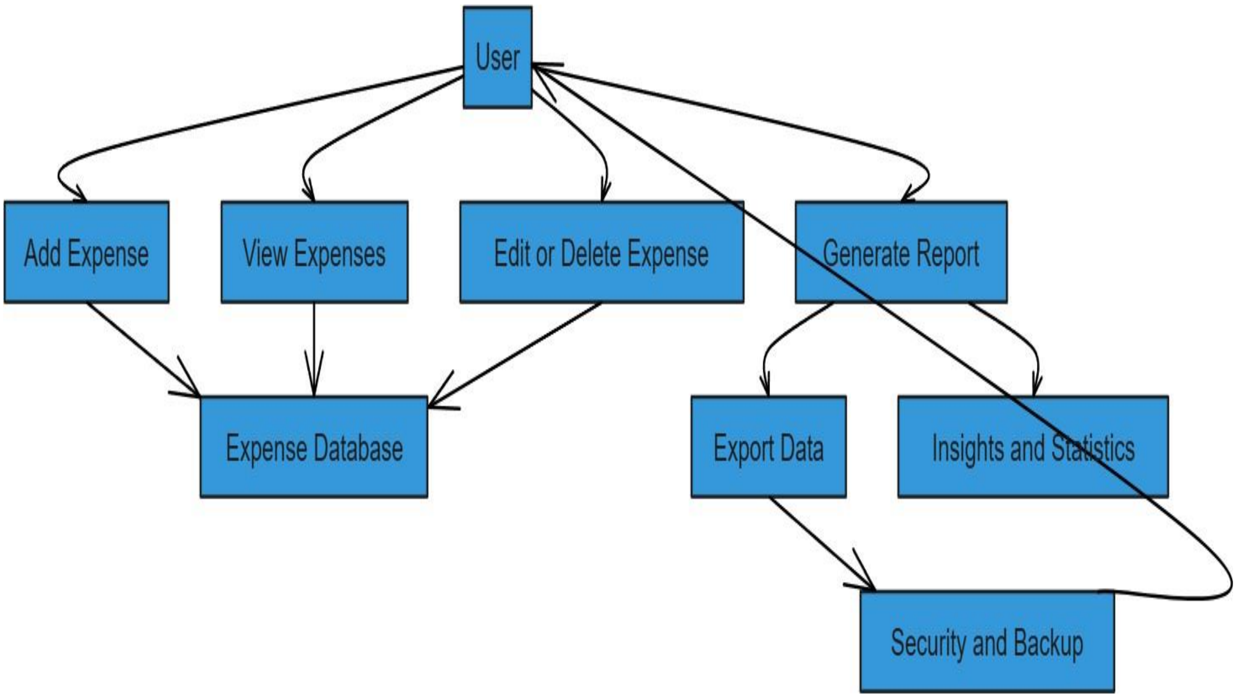
1. **Variables and Data Types:** To store user details, income, expenses, and categories as integers, floats, or strings.
2. **Control Structures:** Conditional statements (if, elif, else) for categorizing expenses or checking budget thresholds, and loops (for, while) for iterating through user inputs or data records.
3. **Functions:** Modularizing the code by creating functions for adding expenses, generating reports, calculating totals, and setting budgets.
4. **File Handling**: Saving and retrieving user data in files (e.g., CSV, JSON) for persistence.
5. **Data Structures:** Using lists for transaction records, dictionaries for categorizing expenses, and tuples for immutable budget settings.
6. **Object-Oriented Programming (OOP):** Encapsulating data and functionality in classes, such as User, Expense, and BudgetTracker, for a structured approach.
7. **Modules and Libraries:** Employing libraries like datetime for date tracking, matplotlib or seaborn for visualizing spending patterns, and pandas for data manipulation.
8. **Regular Expressions:** Validating user inputs like dates or currency formats.

CHAPTER 2
PROJECT METHODOLOGY

2.1 Proposed Work

1. **User Input Module:** Write Python functions to capture user inputs for income, expenses, and categories using simple input() statements.
2. **Data Storage:** Use dictionaries or lists to store transaction details (e.g., amount, category, date). For example, a list of dictionaries can hold each transaction as a record.
3. **Expense Categorization:** Implement logic using conditional statements (if, elif, else) to categorize transactions based on user input.
4. **Total and Balance Calculation**: Write functions to calculate total expenses and remaining balance by iterating over transaction records using loops.
5. **Budget Alerts:** Add a function to compare total expenses against a predefined budget and display a warning message if the limit is exceeded.
6. **Data Persistence:** Use file handling with JSON or CSV to save and retrieve transaction data for future use.
7. **Simple Analytics:** Provide basic insights, such as total expenses per category, using dictionary operations or comprehensions.
8. **Report Generation:** Generate and display a summary report (e.g., using formatted strings) to show income, total expenses, balance, and category-wise spending.

2.2 Block Diagram



CHAPTER 3
MODULE DESCRIPTION

3.1 Module 1 : Add Expense

Function Name: `def add_expense()`

Description: The add_expense() function is likely used within a budgeting or expense tracking program. Its purpose is to add a new expense record to the system. This function would typically require some input parameters, such as the amount of the expense, the category it belongs to (e.g., groceries, rent, entertainment), and possibly a date or description. After receiving this information, the function would process it and store the new expense data in a suitable format within the program.

Steps:

- Amount: The amount of the expense, expected to be a decimal number (float).
- Category: The category of the expense, represented by a string (e.g., "Groceries", "Transportation").
- Description (optional): An optional description of the expense as a string. By default, it's an empty string ("").

3.2 Module 2 : Remove Expense

Function Name : ‘**def remove_expense()**’

Description:- Removes an expense from a data structure updates financial records or budget after expense removal.

Steps:

- Method Definition: Define the method remove_expense within the ExpenseTracker class. The method will take self and description as parameters.
- Iterate Over Expenses: Loop through each expense in the expenses list to find the one that matches the given description.
- Check Description: Inside the loop, compare the description attribute of the current expense with the provided description.
- Remove Expense: If a match is found, remove the expense from the expenses list using the remove method.

3.3 Module 3 : Display Expenses

Function Name : **‘def display_expense()’**

Description:- display_expense() is used to display the expense after the above add the expenses and the removal of expenses.

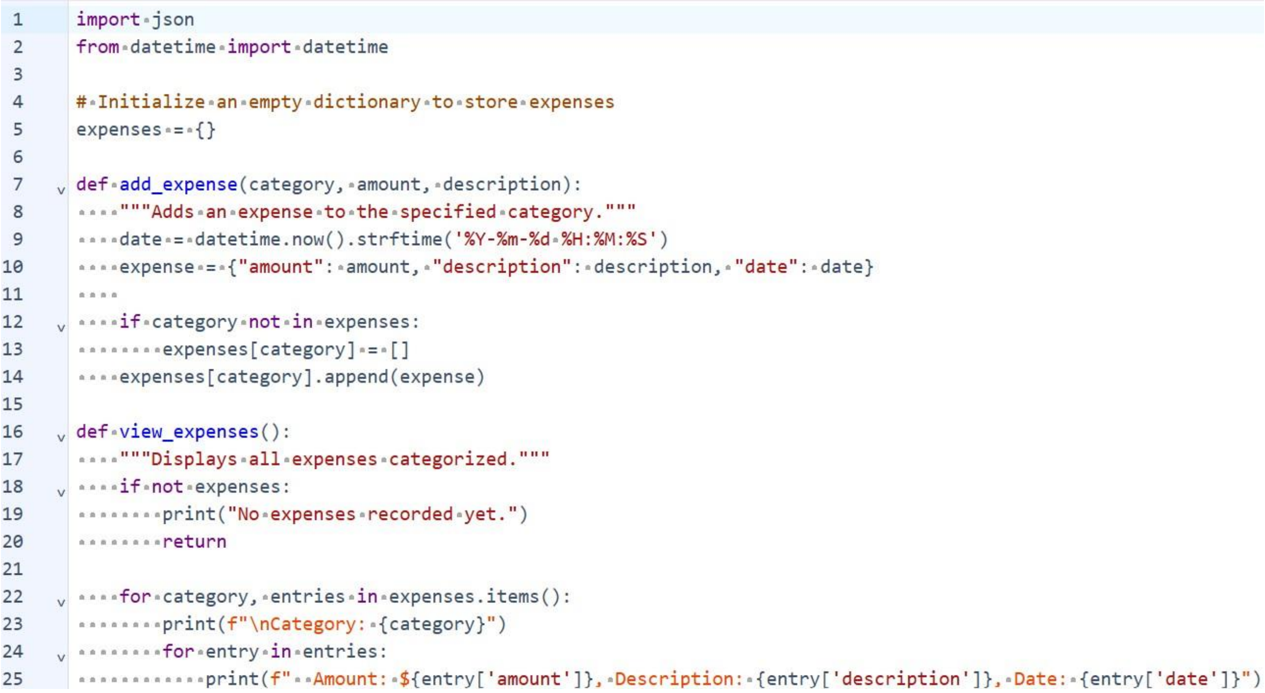
Steps:

- Function Definition: Define the function display_expenses that takes a list of expenses as an argument.
- Check for Empty List: Check if the list is empty and print a message if there are no expenses.
- Iterate Over Expenses: Loop through each expense in the list.
- Print Details: Print the details of each expense in a formatted manner.

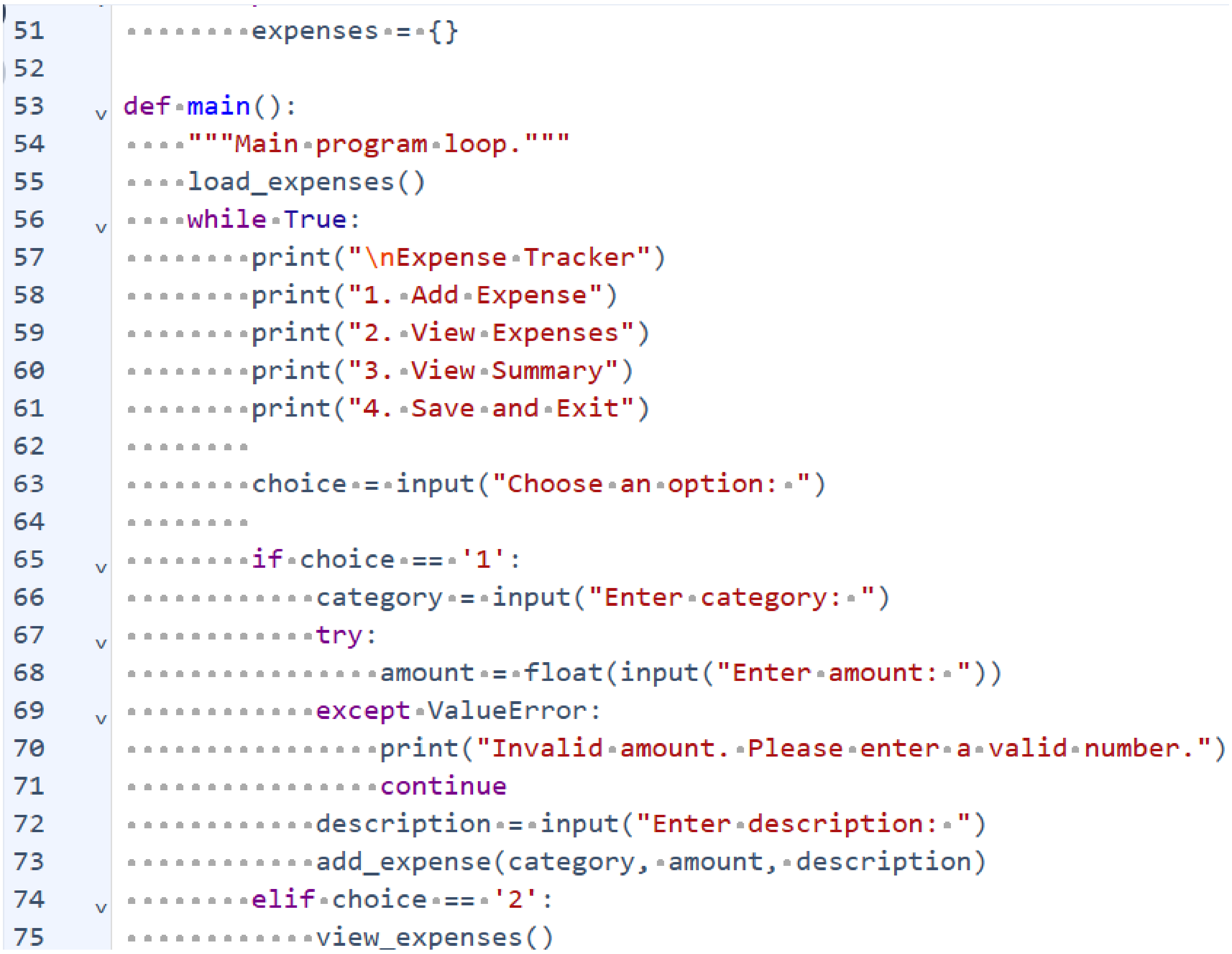
CHAPTER 4

RESULTS AND DISCUSSION

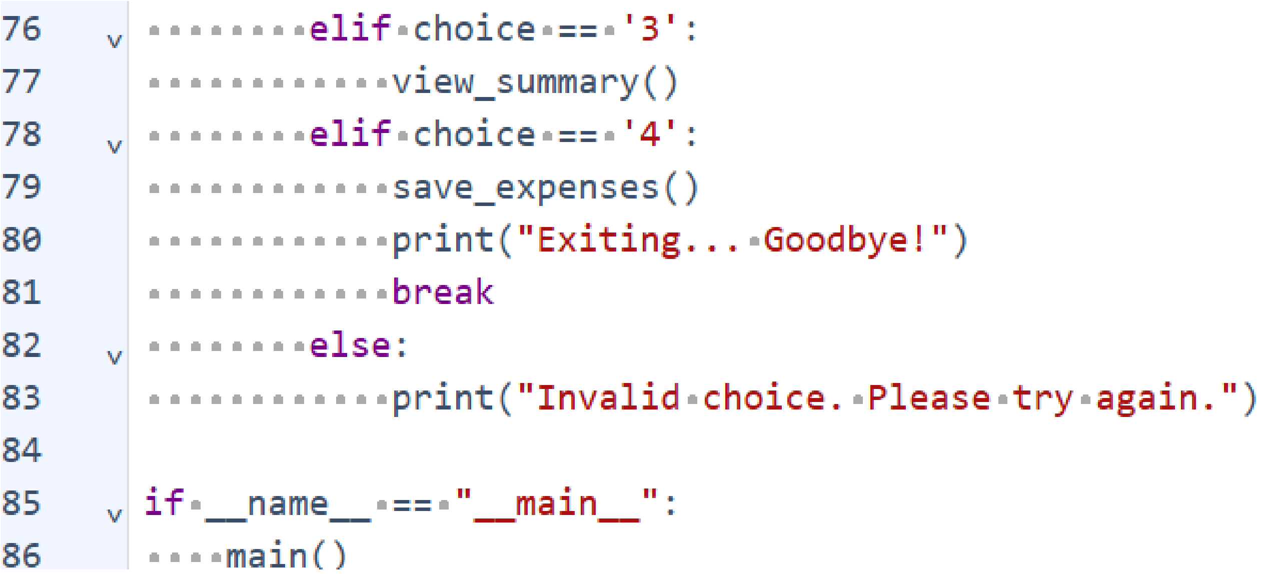
PROGRAM



```python
import json
from datetime import datetime

# Initialize an empty dictionary to store expenses
expenses = {}

def add_expense(category, amount, description):
    """Adds an expense to the specified category."""
    date = datetime.now().strftime('%Y-%m-%d %H:%M:%S')
    expense = {"amount": amount, "description": description, "date": date}
    
    if category not in expenses:
        expenses[category] = []
    expenses[category].append(expense)

def view_expenses():
    """Displays all expenses categorized."""
    if not expenses:
        print("No expenses recorded yet.")
        return

    for category, entries in expenses.items():
        print(f"\nCategory: {category}")
        for entry in entries:
            print(f"  Amount: ${entry['amount']}, Description: {entry['description']}, Date: {entry['date']}")
```

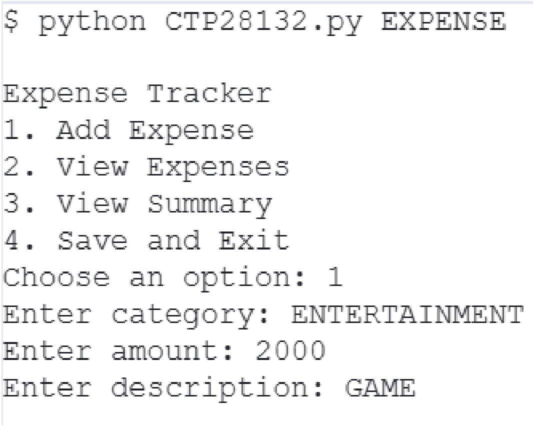


```
        expenses = {}

def main():
    """Main program loop."""
    load_expenses()
    while True:
        print("\nExpense Tracker")
        print("1. Add Expense")
        print("2. View Expenses")
        print("3. View Summary")
        print("4. Save and Exit")
        
        choice = input("Choose an option: ")
        
        if choice == '1':
            category = input("Enter category: ")
            try:
                amount = float(input("Enter amount: "))
            except ValueError:
                print("Invalid amount. Please enter a valid number.")
                continue
            description = input("Enter description: ")
            add_expense(category, amount, description)
        elif choice == '2':
            view_expenses()
```

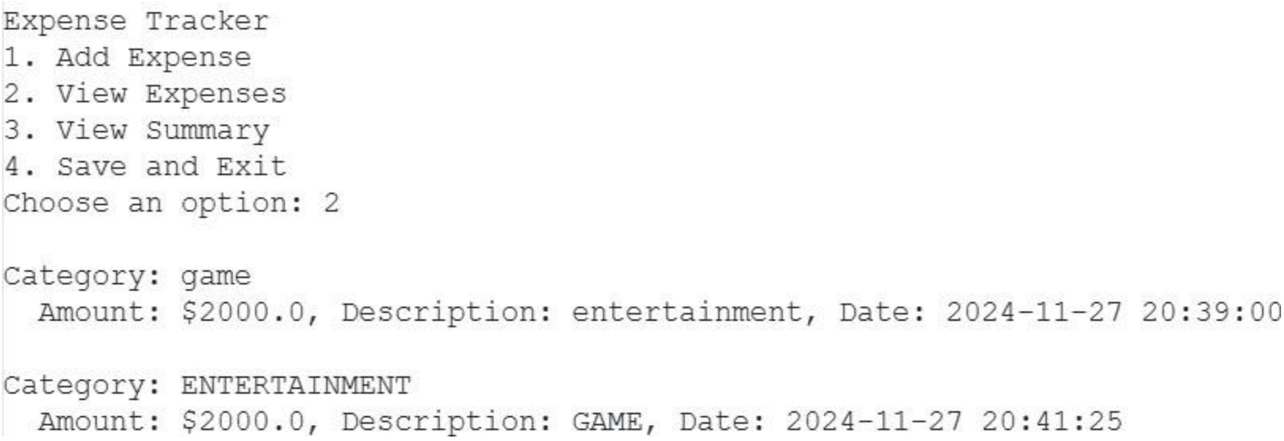


```
        elif choice == '3':
            view_summary()
        elif choice == '4':
            save_expenses()
            print("Exiting... Goodbye!")
            break
        else:
            print("Invalid choice. Please try again.")

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

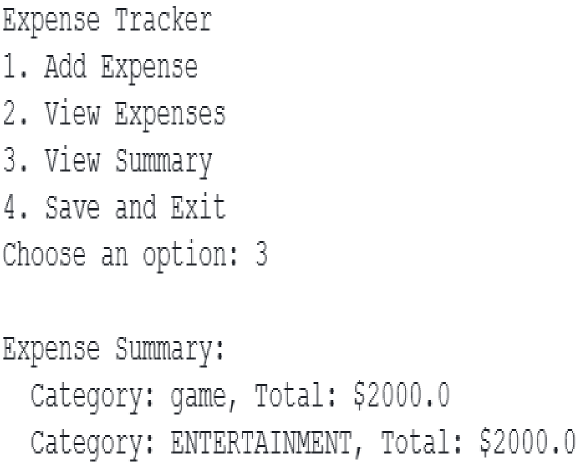
OUTPUT



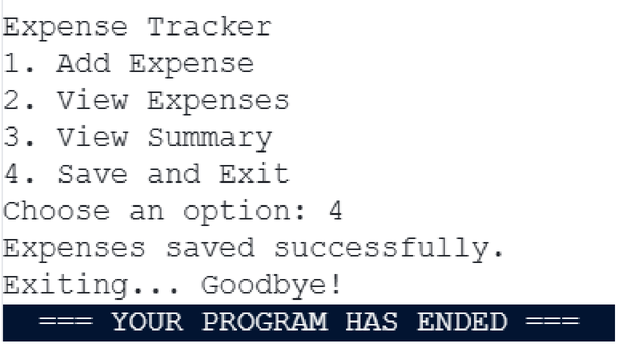
```
$ python CTP28132.py EXPENSE

Expense Tracker
1. Add Expense
2. View Expenses
3. View Summary
4. Save and Exit
Choose an option: 1
Enter category: ENTERTAINMENT
Enter amount: 2000
Enter description: GAME
```



```
Expense Tracker
1. Add Expense
2. View Expenses
3. View Summary
4. Save and Exit
Choose an option: 2

Category: game
  Amount: $2000.0, Description: entertainment, Date: 2024-11-27 20:39:00

Category: ENTERTAINMENT
  Amount: $2000.0, Description: GAME, Date: 2024-11-27 20:41:25
```



```
Expense Tracker
1. Add Expense
2. View Expenses
3. View Summary
4. Save and Exit
Choose an option: 3

Expense Summary:
  Category: game, Total: $2000.0
  Category: ENTERTAINMENT, Total: $2000.0
```



```
Expense Tracker
1. Add Expense
2. View Expenses
3. View Summary
4. Save and Exit
Choose an option: 4
Expenses saved successfully.
Exiting... Goodbye!
  === YOUR PROGRAM HAS ENDED ===
```

CHAPTER 5
CONCLUSION

The personalized expense tracker is an efficient and user-friendly tool designed to help individuals manage their finances by providing streamlined tracking of income, expenses, and budgets. With features like categorization, data visualization, and budget alerts, it simplifies financial decision-making while promoting discipline and savings. Built using Python, it leverages essential programming concepts like data structures, file handling, and visualization libraries to offer a robust and scalable solution. This project demonstrates the power of Python in creating practical applications, paving the way for future enhancements like real-time integrations and predictive analytics to further enhance its utility.

REFERENCES:

1. Automate the Boring Stuff with Python" by Al Sweigart.
2. Python for Finance: Analyze Big Financial Data" by Yves Hilpisch.
3. Python Programming for Beginners: An Introduction to the Python Computer Language and Computer Programming" by Jason Cannon.
4. https//:w3schools.com

APPENDIX

(Coding)

```
import json
print("\nExpense Summary:")
for category, total in summary.items():
print(f" Category: {category}, Total: ${total}")

def save_expenses(file_path='expenses.json'):
"""Saves expenses to a file."""
with open(file_path, 'w') as file:
json.dump(expenses, file)
print("Expenses saved successfully.")

def load_expenses(file_path='expenses.json'):
"""Loads expenses from a file."""
global expenses
try:
with open(file_path, 'r') as file:
expenses = json.load(file)
except FileNotFoundError:
expenses = {}

def main():
"""Main program loop."""
load_expenses()
while True:
print("\nExpense Tracker")
print("1. Add Expense")
print("2. View Expenses")
print("3. View Summary")
print("4. Save and Exit")
choice = input("Choose an option: ")
if choice == '1':
category = input("Enter category: ")
try:
amount = float(input("Enter amount: "))
except ValueError:
print("Invalid amount. Please enter a valid number.")
continue
description = input("Enter description: ")
add_expense(category, amount, description)
elif choice == '2':
view_expenses()
elif choice == '3':
view_summary()
elif choice == '4':
save_expenses()
print("Exiting... Goodbye!")
break
else:
print("Invalid choice. Please try again.")

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

APPENDIX

(Output)

```
Expense Tracker
1. Add Expense
2. View Expenses
3. View Summary
4. Save and Exit
Choose an option: 1
Enter category: entertainment
Enter amount: 2000
Enter description: games

Expense Tracker
1. Add Expense
2. View Expenses
3. View Summary
4. Save and Exit
Choose an option: 2
Category: entertainment
  Amount: $2000.0, Description: games, Date: 2024-12-01 21:05:14
```

```
Expense Tracker
1. Add Expense
2. View Expenses
3. View Summary
4. Save and Exit
Choose an option: 3
Expense Summary:
  Category: entertainment, Total: $2000.0

Expense Tracker
1. Add Expense
2. View Expenses
3. View Summary
4. Save and Exit
Choose an option: 4
Expenses saved successfully.
Exiting... Goodbye!
  === YOUR PROGRAM HAS ENDED ===
```