Finance Tracker Application Report

**1. Introduction**

**a. What is your application?**

The Finance Tracker is a command-line application designed to help users manage and track their expenses. It allows users to add, remove, and view expenses, set daily/weekly/monthly deposit limits, and handle group deposits for multiple users. The application also supports saving and loading expense data to and from CSV files.

**b. How to run the program?**

Ensure Python is installed: The program is written in Python, so you need Python installed on your machine.

Save the program: Save the provided code into a file, for example, finance\_tracker.py.

Run the program: Open your command line interface (CLI), navigate to the directory where the finance\_tracker.py file is located, and run the following command:



Run the tests: To run the unit tests, use the following command:



**c. How to use the program?**

Start the program: Run the program using the instructions above. The main menu will be displayed.

Main Menu Options:

1. Add Expense: Enter the amount, description, user, and category of the expense.

2. Remove Expense: View the list of expenses and enter the index of the expense to remove.

3. View Expenses: Display the list of all recorded expenses.

4. Save Expenses to File: Enter a filename to save the current list of expenses to a CSV file.

5. Load Expenses from File: Enter a filename to load expenses from a CSV file.

6. Set Limits: Set daily, weekly, and monthly deposit limits.

7. Add User: Add a new user to the application.

8. Remove User: Remove an existing user from the application.

9. List Users: Display the list of all users.

10. Exit: Exit the program.

**2. Body/Analysis**

**Explain how the program covers (implements) functional requirements**

**1. The 4 OOP Pillars (Polymorphism, Abstraction, Inheritance, Encapsulation)**

**Polymorphism:** Implemented through the Expense class and its subclasses FoodExpense and TransportExpense. Each subclass can be used interchangeably with the base class.

A screenshot of a computer

Description automatically generated

**Abstraction:** The FinanceTracker class provides a simple interface for managing expenses without exposing the underlying complexity.

A screenshot of a computer program

Description automatically generated .

**Inheritance:** FoodExpense and TransportExpense inherit from the Expense class.

A screenshot of a computer screen

Description automatically generated

**Encapsulation:** Data members like expenses and limits in the FinanceTracker class are encapsulated and accessed through methods.

A black screen with white text

Description automatically generated

**2. Use 2 Design Patterns (Singleton and Decorator)**

**Singleton Pattern:** Ensures only one instance of FinanceTracker exists.





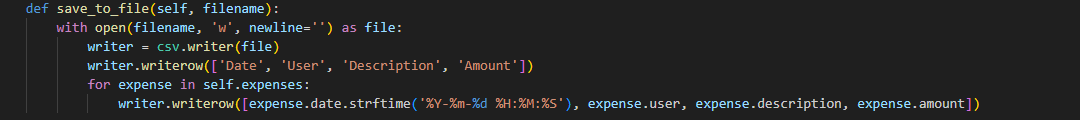
**Decorator Pattern:** Adds a print statement before executing any decorated function.

A computer screen shot of text

Description automatically generated

**3. Read from file & write to file (Choose a file type (e.g., TXT, CSV) and implement functions to import and export data)**

**Save to CSV:** Saves expenses to a CSV file.



**Load from CSV:** Loads expenses from a CSV file.

A screen shot of a computer code

Description automatically generated

**4. Testing (Core functionality should be covered with unit tests (using unittest framework))**

**Unit Tests:** Validate core functionalities like adding, removing, setting limits, and file operations.

A screen shot of a computer program

Description automatically generated5.

**3. Results and Summary**

**Results**

The program successfully tracks expenses for multiple users.

The file operations for saving and loading expense data work as intended.

The user management system allows for adding, removing, and listing users.

Deposit limits can be set for daily, weekly, and monthly tracking.

**Challenges Faced**

Designing a flexible system to handle expenses for multiple users.

Ensuring data integrity when saving and loading from CSV files.

Implementing and testing the Singleton and Decorator design patterns.

**4. Conclusions**

**Key Findings and Outcomes**

The Finance Tracker application meets the defined functional requirements and provides a robust system for managing personal and group expenses.

The implementation demonstrates the effective use of OOP principles and design patterns to create a maintainable and scalable codebase.

**Future Prospects**

The application can be extended to include more detailed financial reports and visualizations.

Integration with external APIs for automatic expense tracking and categorization.

Adding a graphical user interface (GUI) for a more user-friendly experience.

**In conclusion:** this coursework has demonstrated the application of advanced programming concepts to build a practical finance tracking application. The Finance Tracker provides a solid foundation for further development and feature enhancement.