EXPENSE TRACKER PROGRAM – PROJECT REPORT

Project Title: Simple Expense Tracker Program

Student Name: Rehab Hussain (25BSA10162)

Course: Fundamentals in AL & ML

Introduction

Managing daily expenses is an essential activity for students, professionals, and households. Small, untracked expenses often accumulate and lead to poor budgeting decisions. The Simple Expense Tracker Program is a Python-based console application designed to help users record, view, and analyze their expenses. It provides category-wise breakdowns, visualization using pie charts, and a minimal, user-friendly workflow suitable for beginners.

Problem Statement

Users struggle to keep track of their daily spending, which leads to ineffective budgeting and poor financial awareness. There is a need for a simple and easy-to-use software application that:

- Records expenses with names, amounts, and categories
- Displays all expenses clearly
- Generates summary reports and visual insights
- Helps users identify their highest spending category

This project addresses these needs by implementing a basic but functional expense tracking tool.

Functional Requirements

Core Functional Modules

1. **Add Expense**
    - Input: Expense name, amount, category
    - Output: Confirmation and storage of the expense
2. **View All Expenses**
    - Display all stored expenses with index, name, amount, and category
3. **Generate Expense Report**
    - Calculate total spending
    - Compute category-wise totals
    - Identify the category with the highest spending

- Display a pie chart summarizing the expenses

Input/Output Structure

Function	Input	Output
Add Expense	name, amount, category	expense entry saved
View Expenses	—	list of expenses
Expense Report	—	totals + analysis + pie chart

Workflow Overview

- User launches the program
- Selects an option from the menu
- Performs add/view/report actions
- Program continues until user chooses Exit

Non-Functional Requirements

Performance

- The system should capture inputs instantly and generate reports with minimal delay.

Security

- No external storage; data stored temporarily in memory during program execution.

Usability

- Menu-driven approach for simple navigation.
- Clear prompts and error messages.

Reliability

- Handles invalid numeric input safely using try/except.

Scalability

- Can be extended to support databases, more categories, multi-user access, etc.

Maintainability

- Modular functions make it easy to update and modify.

Error Handling Strategy

- Validation for numeric input

- Default category assignment for invalid selections
- Safe handling of empty expense lists

System Architecture

The system follows a **simple procedural architecture**:

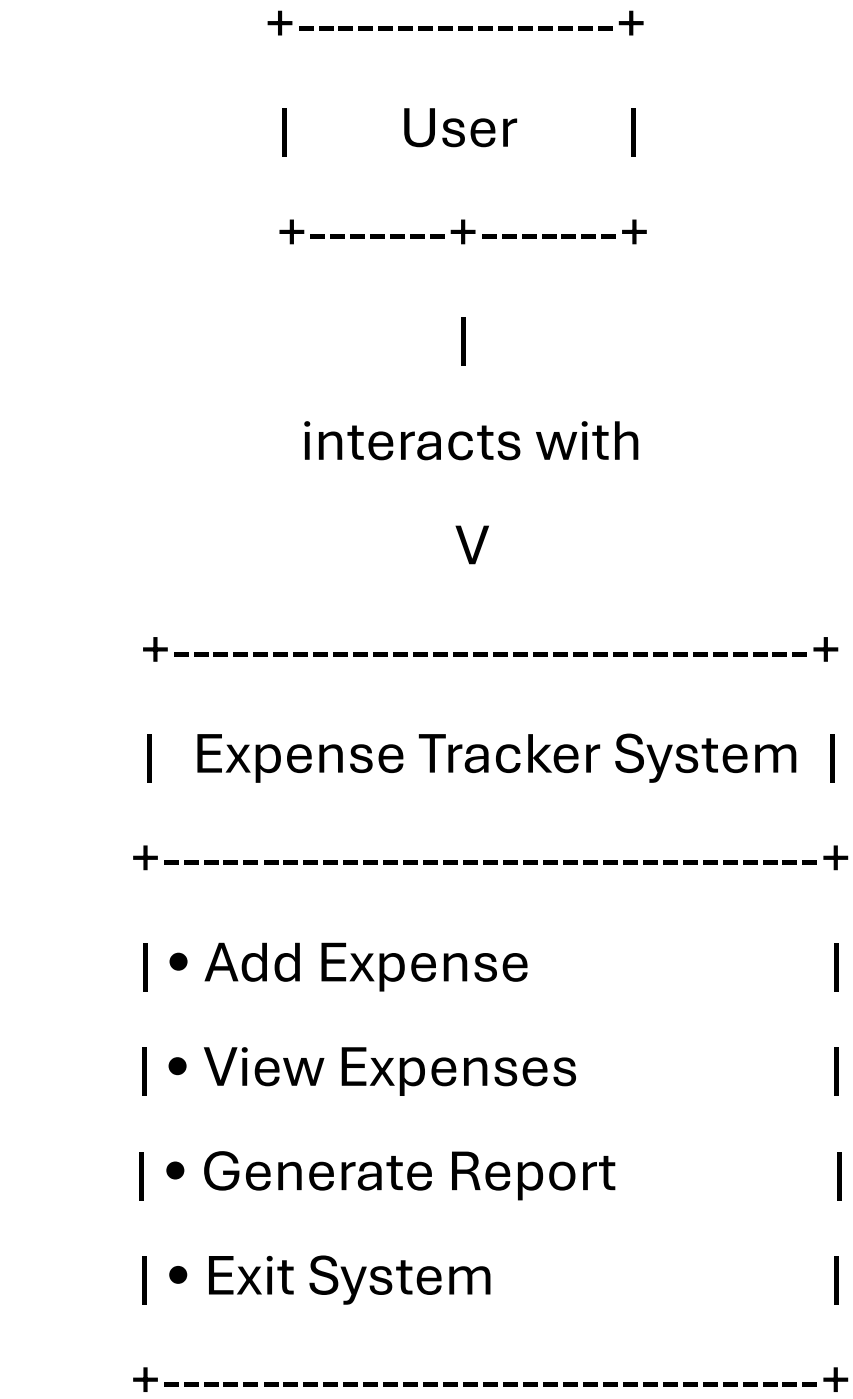
User Input → Menu Logic → Function Calls → Expense List → Report/Visualization

Core components:

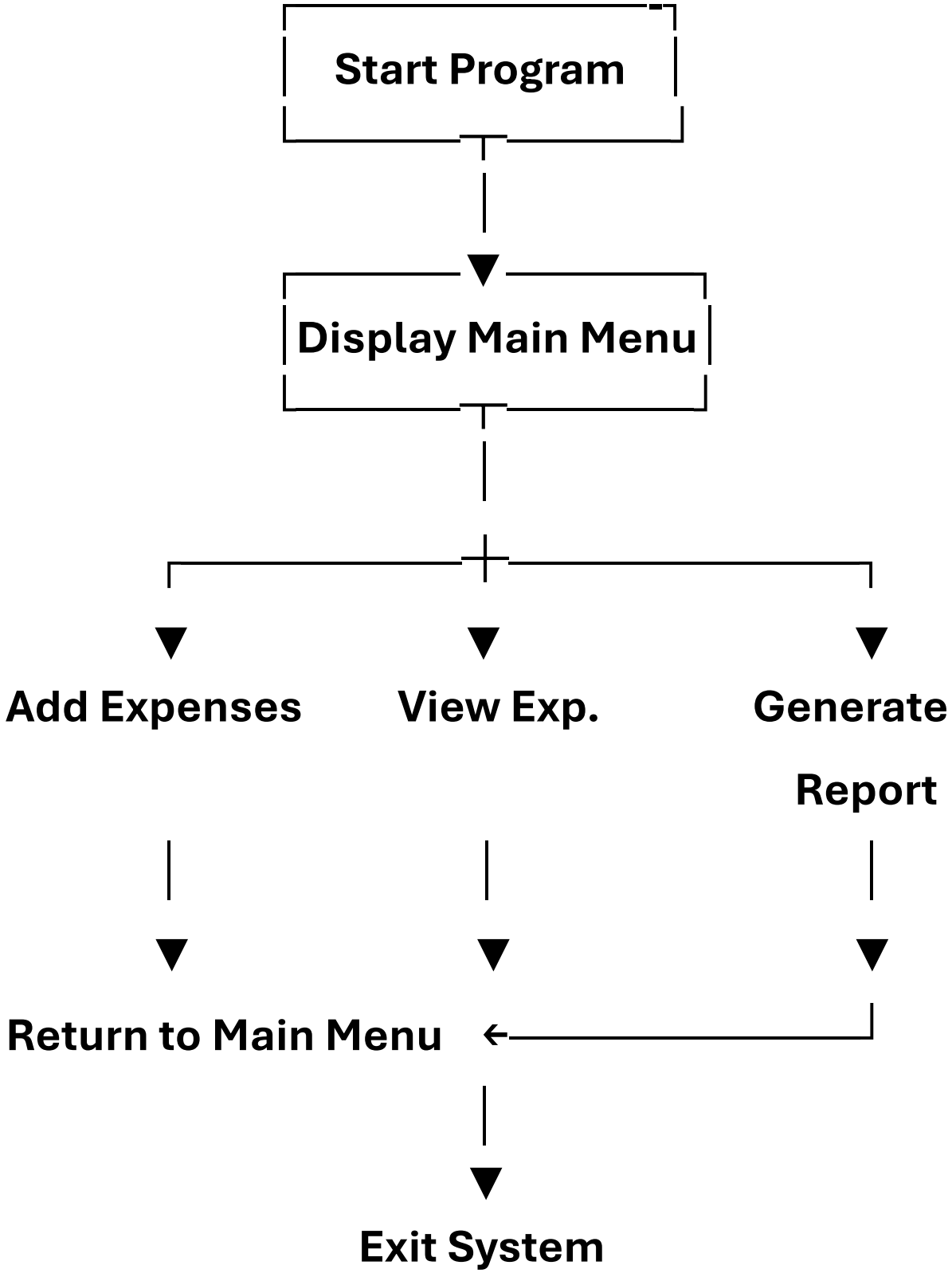
- UI Layer (Console Menu)
- Processing Layer (Functions for operations)
- Data Layer (List storing expenses)
- Visualization Layer (Matplotlib Chart)

Design Diagrams

a) Case Diagram



b) Workflow Diagram



c) Sequence Diagram

~ User → Main Menu : Select option

~ Main Menu → Add Expense : If option = 1

~ Add Expense → User : Ask for name,
amount, category

~ Add Expense → Expense List : Store entry

~ Main Menu → View Expense : If option = 2

~ View Expense → User : Display all
expenses

~ Main Menu → Report : If option = 3

~ Report → User : Display totals & chart

~ User → Main Menu : Repeat or exit

d) Class / Component Diagram

(Although program uses procedural style, logical components are shown)

Design Decisions & Rationale

- **Lists used for data storage** because they are simple and require no database setup.
- **Console-based UI** chosen for ease of implementation and accessibility.
- **Matplotlib for visualization** to provide instant visual insights.
- **Module-based functions** for maintainability and readability.
- **Category validation** ensures consistent reporting.

Implementation Details

- Language: Python 3
- Libraries used:

  ~ matplotlib.pyplot for charts

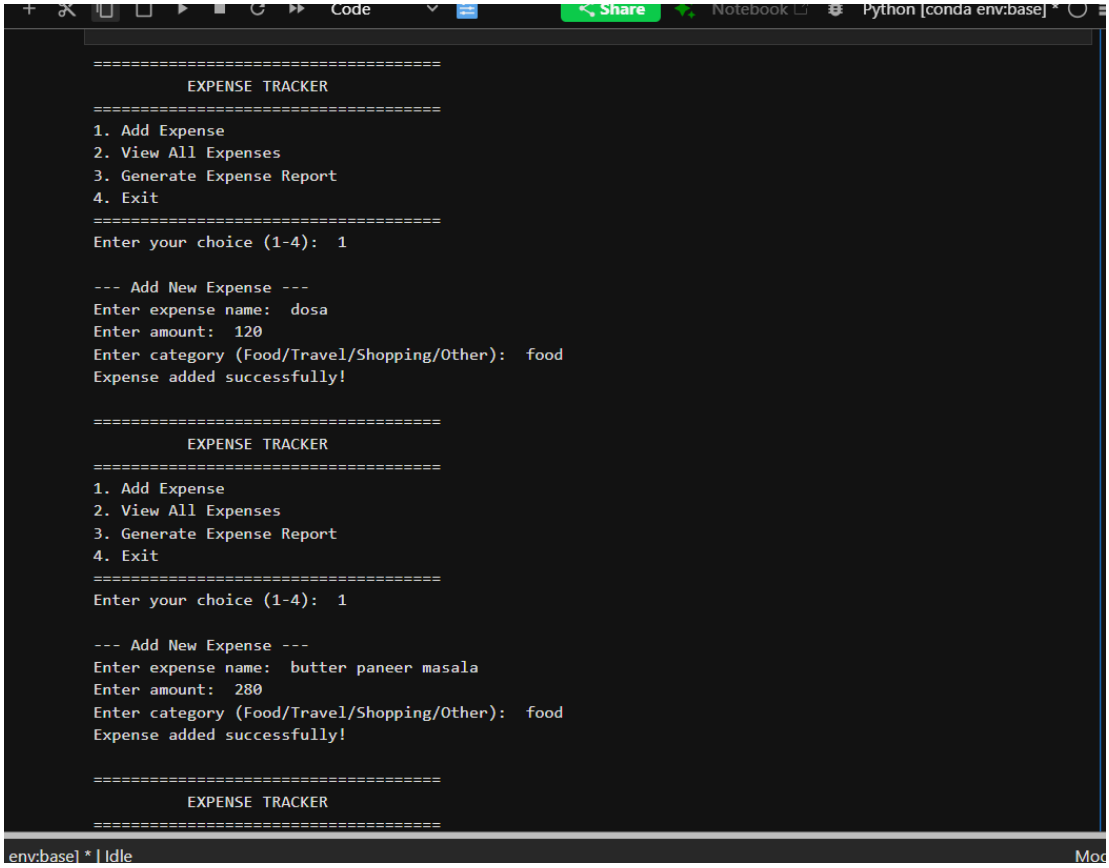
- Program flow:

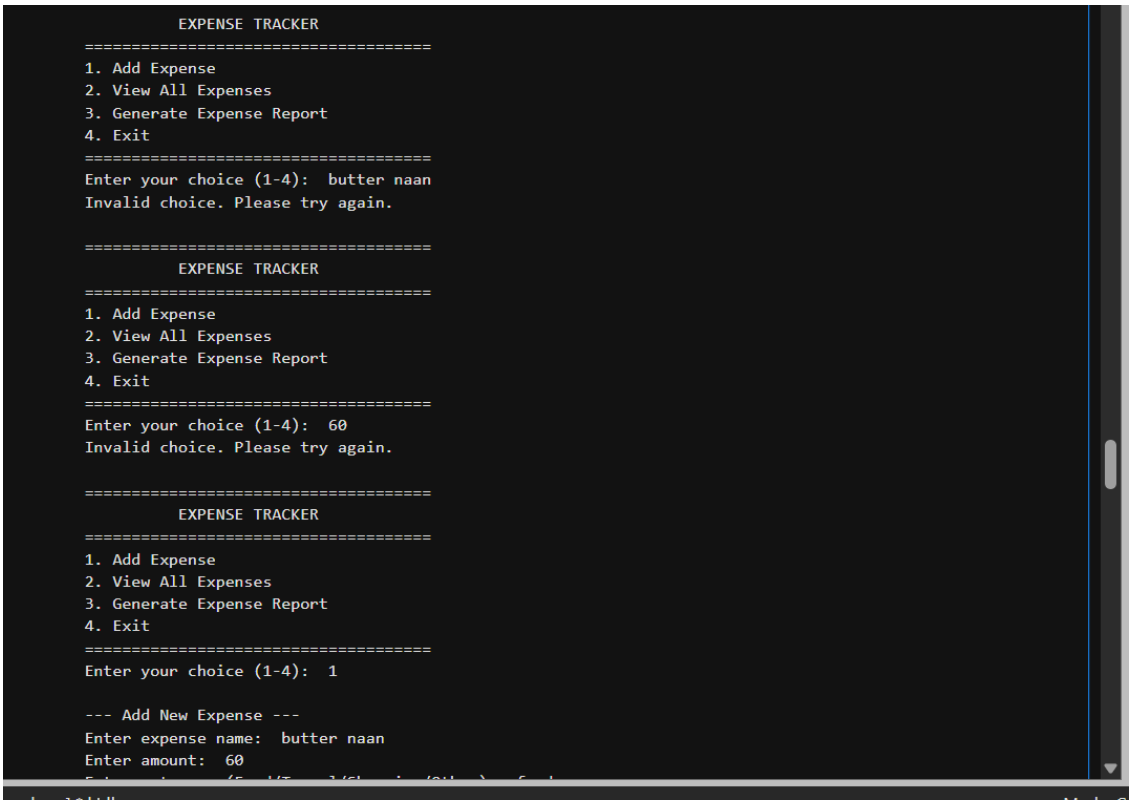
  ~Main menu triggers functions based on user input

  ~Data stored temporarily in a list of dictionaries

  ~Report function calculates totals and generates pie chart

Screenshots / Results



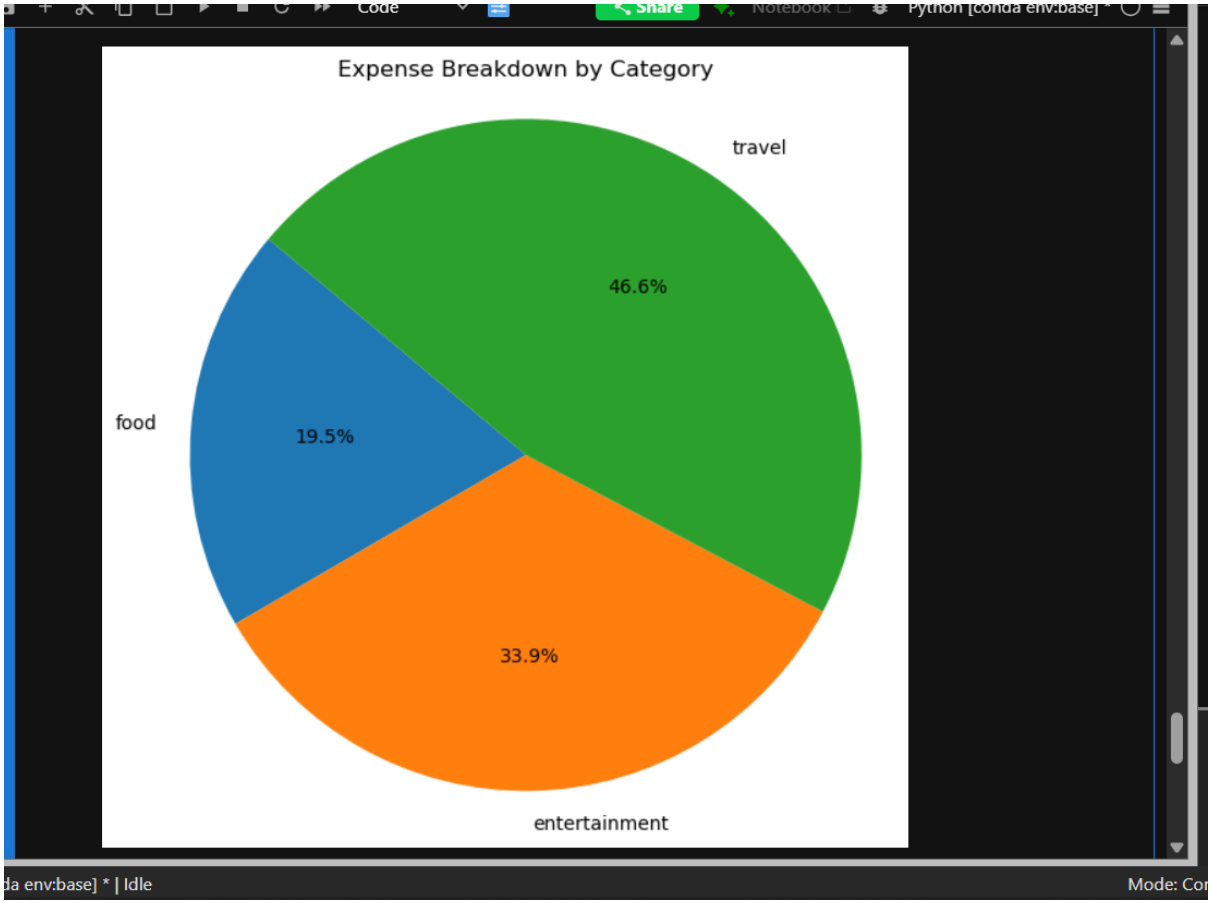


```
--- Add New Expense ---
Enter expense name:  butter naan
Enter amount:  60
Enter category (Food/Travel/Shopping/Other):  food
Expense added successfully!

====================================
          EXPENSE TRACKER
====================================
1. Add Expense
2. View All Expenses
3. Generate Expense Report
4. Exit
====================================
Enter your choice (1-4):  1

--- Add New Expense ---
Enter expense name:  movies
Enter amount:  600
Enter category (Food/Travel/Shopping/Other):  entertainment
Expense added successfully!

====================================
          EXPENSE TRACKER
====================================
1. Add Expense
2. View All Expenses
3. Generate Expense Report
4. Exit
====================================
Enter your choice (1-4):  1

--- Add New Expense ---
Enter expense name:  games
Enter amount:  200
```

```
--- Add New Expense ---
Enter expense name:  games
Enter amount:  200
Enter category (Food/Travel/Shopping/Other):  entertainment
Expense added successfully!

====================================
          EXPENSE TRACKER
====================================
1. Add Expense
2. View All Expenses
3. Generate Expense Report
4. Exit
====================================
Enter your choice (1-4):  1

--- Add New Expense ---
Enter expense name:  bus
Enter amount:  100
Enter category (Food/Travel/Shopping/Other):  travel
Expense added successfully!

====================================
          EXPENSE TRACKER
====================================
1. Add Expense
2. View All Expenses
3. Generate Expense Report
4. Exit
====================================
Enter your choice (1-4):  1

--- Add New Expense ---
Enter expense name:  train
Enter amount:  1000
```

```
--- Add New Expense ---
Enter expense name:  train
Enter amount:  1000
Enter category (Food/Travel/Shopping/Other):  travel
Expense added successfully!

====================================
          EXPENSE TRACKER
====================================
1. Add Expense
2. View All Expenses
3. Generate Expense Report
4. Exit
====================================
Enter your choice (1-4):  2

===== ALL EXPENSES =====
1. dosa            | ₹ 120.00 | Category: food
2. butter paneer masala | ₹ 280.00 | Category: food
3. butter naan     | ₹  60.00 | Category: food
4. movies          | ₹ 600.00 | Category: entertainment
5. games           | ₹ 200.00 | Category: entertainment
6. bus             | ₹ 100.00 | Category: travel
7. train           | ₹1000.00 | Category: travel
------------------------------

====================================
          EXPENSE TRACKER
====================================
1. Add Expense
2. View All Expenses
3. Generate Expense Report
4. Exit
====================================
Enter your choice (1-4):  3
```

```
Enter your choice (1-4):  3

===== EXPENSE REPORT =====
Total Expenses: ₹2360.00

Category-wise Breakdown (With Percentages):
- food        : ₹460.00 (19.49%)
- entertainment: ₹800.00 (33.90%)
- travel      : ₹1100.00 (46.61%)

===== FEEDBACK =====
ℹ️ 46.6% of total expenses went to travel. Watch this category.
--------------------------------
```




```
=====================================
          EXPENSE TRACKER
=====================================
1. Add Expense
2. View All Expenses
3. Generate Expense Report
4. Exit
=====================================
Enter your choice (1-4):  4
Exiting program... Goodbye!
```

Testing Approach

Manual Testing

- Tested input validation with:
  - Correct numeric values
  - Incorrect values (text, empty input, symbols)
- Tested category selection
- Tested empty list behavior for:
  - View expenses
  - Report generation

Functional Testing

- Verified all menu options produce correct output.

Boundary Testing

- Very large amount values

- Repeated entries
- Only one category type

Challenges Faced

- Ensuring robust input validation for numeric values
- Implementing category-wise calculations manually
- Handling cases with no expenses entered
- Integrating matplotlib pie chart in a console application

Learnings & Key Takeaways

- Gained understanding of modular programming in Python
- Improved ability to validate and sanitize user input
- Learned how to generate charts with Matplotlib
- Understood how to structure simple software requirements and architecture
- Experienced designing user-friendly console workflows

Future Enhancements

- Add permanent storage (JSON, CSV, or database)
- Add authentication for different users
- Add monthly/yearly reports
- Export reports to PDF
- Build a GUI version using Tkinter or PyQt
- Add advanced categories and analytics

References

- Python official documentation: https://docs.python.org/
- Matplotlib documentation: https://matplotlib.org/
- Online Python tutorials and learning resources
- Class lecture notes and assignments