

# Project Report

## Personal Expense Tracker

### 1. Introduction

The **Personal Expense Tracker** is a Python-based mini project designed to help users manage their daily expenses effectively.

It allows users to:

- Log expenses with category and date.
- View summaries by total, category, and over time (daily/monthly).
- Save and load expense data using **file handling** (JSON format).
- Optionally delete an expense.

This project demonstrates the use of **functions, dictionaries, file handling, and basic data analysis** in Python.

### 2. Objectives

- To apply **Python programming concepts** in a real-world problem.
- To build a tool that helps track personal financial habits.
- To implement **data persistence** using file handling.

### 3. Features Implemented

- Add expenses (amount, category, date).
- View summary:
  - Total overall spending.
  - Spending by category.
  - Daily and monthly spending trends.
- Save data permanently in **expenses.json**.
- User-friendly **menu-driven interface**.
- Bonus: Delete an expense by selecting entry number.

## 4. System Requirements

- Python 3.x
- Any text editor/IDE (VS Code, PyCharm, IDLE, etc.)
- JSON library (already included in Python standard library)

## 5. Project Structure

PersonalExpenseTracker/

├──	expense_tracker.py	# Main program file
├──	expenses.json	# Data file (created automatically after first run)
└──	README.md	# Documentation (this report)

## 6. Code Explanation

### 6.1 Adding an Expense

- User enters **amount**, **category**, and **date** (or defaults to today).
- Data is stored in a **list of dictionaries**.
- Saved to **expenses.json**.

**Example:**

```
expense = {"amount": amount, "category": category, "date": date}
expenses.append(expense)
save_expenses(expenses)
```

### 6.2 Viewing Summary

- **Total Spending** → sums all expenses.
- **Category-wise Spending** → groups by category using **defaultdict**.
- **Time-based Spending** → groups by daily (**YYYY-MM-DD**) or monthly (**YYYY-MM**).

## 6.3 File Handling

- Uses JSON to **store data persistently**.
- On startup, loads existing records.

### Example:

```
def load_expenses():  
    if os.path.exists(FILE_NAME):  
        with open(FILE_NAME, "r") as file:  
            return json.load(file)  
    return []
```

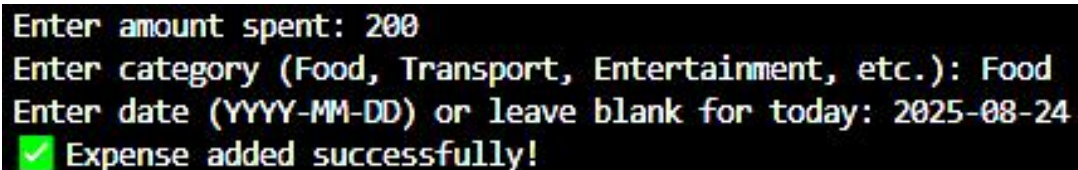
## 6.4 Menu System

```
===== Personal Expense Tracker =====  
1. Add Expense  
2. View Summary  
3. Delete Expense  
4. Exit
```

Each option calls the respective function.

## 7. Sample Execution Screenshots

### 7.1 Adding an Expense



```
Enter amount spent: 200  
Enter category (Food, Transport, Entertainment, etc.): Food  
Enter date (YYYY-MM-DD) or leave blank for today: 2025-08-24  
✅ Expense added successfully!
```

## 7.2 Viewing Summary

```
---- Expense Summary ----
1. Total Spending
2. Spending by Category
3. Spending over Time (daily/monthly)
Choose an option: 2

📊 Spending by Category:
Food: $200.00
Transport: $100.00
```

## 7.3 Spending Over Time

```
View by (daily/monthly): Daily

📅 Spending (Daily):
2025-08-24: $300.00
```

## 7.4 Deleting an Expense

```
---- Expenses List ----
1. 2025-08-24 | Food | $200.00
2. 2025-08-24 | Transport | $100.00

Enter the expense number to delete: 2
🗑 Deleted: {'amount': 100.0, 'category': 'Transport', 'date': '2025-08-24'}
```

## 8. Results & Conclusion

This project successfully demonstrates:

- **Data structures** (lists, dictionaries, defaultdict).
- **File handling** (JSON read/write).
- **Menu-driven program design**.
- **Practical application** of Python in managing finances.

The **Personal Expense Tracker** helps users monitor spending habits and improves financial awareness.

## 9. Future Enhancements (Optional Ideas)

- Add **graphical charts** (using **matplotlib**) for expense visualization.
- Export data to **CSV or Excel**.
- Add **search & filter** (e.g., show all “Food” expenses in a date range).
- Build a **GUI** version using Tkinter/PyQt.

## 10. Deliverables:

- **expense\_tracker.py** (code file)
- **expenses.json** (auto-generated data file)
- Project Report (this README/documentation)