# **Expense Tracker Application Documentation**

## **Introduction**

The Expense Tracker Application is a CLI-based tool developed as part of a midcourse project for Fynd Academy. The application helps users manage their daily expenses by allowing them to add, categorize, view, and delete expenses. It also generates expense reports and saves data persistently in a CSV file, allowing the application to be resumed between sessions.

The project demonstrates key concepts such as Python programming, object-oriented design, file handling, and error management.

## **Architecture**

The application is divided into several modules to maintain modularity and separation of concerns:

1. **Expense**:
   1. Defines the Expense class, representing an individual expense.
   2. Attributes include amount, date, category, and description.
2. **Category**:
   1. Defines the Category class, which manages different expense categories.
   2. Categories can be added, and expenses are categorized to enable filtering.
3. **ExpenseTracker**:
   1. The core class that manages expenses and categories.
   2. Contains methods for adding, deleting, viewing, filtering expenses, and saving/loading from CSV.
4. **main.py**:
   1. The main entry point for the application.
   2. Provides the command-line interface for user interaction and connects with the ExpenseTracker class to execute core functionalities.

### **Class Interaction Overview**

* **ExpenseTracker** aggregates instances of Expense and Category. It is responsible for managing operations such as adding expenses, categorizing them, and generating reports.
* **main.py** acts as the user interface layer, calling methods from ExpenseTracker based on user inputs.

## **Features**

### **1. Expense Management**

* **Add Expense**: Users can add an expense by entering the amount, date, category, and an optional description.
* **View All Expenses**: Displays a list of all expenses recorded so far in an organized format.
* **Delete Expense**: Users can remove expenses by specifying their index in the list.
* **Update Expense**: Users can modify an existing expense’s details (if implemented).

### **2. Category Management**

* **Add Category**: Users can define their own categories (e.g., groceries, utilities, etc.).
* **Filter by Category**: Users can filter expenses based on specific categories to see relevant data.

### **3. CSV File Handling**

* **Persistence**: Expenses are saved in a CSV file, allowing data to be stored and accessed between application sessions.
* **Load Data**: When the application starts, it loads all existing expenses from the CSV file.

## **Testing**

### **Manual Testing Strategy**

The following scenarios were manually tested to ensure the application functions as expected:

* **Add Expense**: Verified that expenses are correctly added to the system and saved to the CSV file.
* **View Expenses**: Confirmed that the application can correctly display a list of all expenses.
* **Delete Expense**: Tested deletion by index and verified that the correct record is removed.
* **Filter by Category**: Ensured that expenses can be filtered by category and the correct results are displayed.
* **CSV File Operations**: Verified that data is saved to and loaded from the CSV file properly.
* **Error Handling**: Tested input validation to ensure the application does not crash due to invalid inputs (e.g., non-numeric amounts or invalid dates).

## **Challenges**

Several challenges arose during the development of the Expense Tracker:

* **Error Handling**: Implementing robust error handling was crucial, especially when dealing with invalid user inputs and file operations. Ensuring that invalid data did not crash the application required special attention.
* **Data Persistence**: Ensuring data persistence between sessions was a key challenge. Proper CSV handling was essential to maintain data integrity and prevent data loss when the application is restarted.
* **User Input Validation**: Validating user input, especially for dates and amounts, required additional logic to ensure only valid inputs were accepted.

## **Future Enhancements**

Here are some potential future enhancements that could be made to improve the application:

1. **Recurring Expenses**: Implement a system to handle recurring expenses, which would automatically add certain expenses (e.g., monthly rent or subscriptions) at regular intervals.
2. **Graphical User Interface (GUI)**: Move from a CLI-based application to a graphical interface using a library such as Tkinter for enhanced user experience.
3. **Enhanced Reporting**: Introduce more advanced reporting options, such as year-to-date reports or percentage breakdowns of expense categories.
4. **Budget Tracking**: Add a feature that allows users to set budgets for different categories and receive alerts when they are nearing their budget limits.

## **Conclusion**

The Expense Tracker Application provides users with an efficient and user-friendly way to manage and track their expenses. By leveraging Python's object-oriented capabilities and file handling, the application offers persistent storage, categorization, and expense reporting, meeting the project objectives for Fynd Academy.