**INTRODUCTION**

This project is an Expense Tracker application designed to help users manage their personal finances by tracking their expenses and categorizing them. The application allows users to add, modify, delete, and view expenses, as well as manage expense categories. The data is stored in CSV files for persistence.

By achieving these objectives, the Expense Tracker aims to help users gain better control over their spending habits and improve their financial management skills.

**FEATURES**

CATEGORY MANAGEMENT:

[C] User should be able to define categories for expenses

[C] Users should be able to use predefined categories like groceries, utilities, entertainment

[R] User should be able to view categories available

EXPENSE MANAGEMENT:

[C] User should be able to add an expense by specifying details such as the amount, date, category, and description

[U] Users should be able to modify an expense

[D] User should be able to delete an expense

[R] User should be able to view expenses as a list of all recorded expenses in a clear and organized format

[R] User should be able to view expenses for specific categories

FILE HANDLING:

Implement file handling for saving expenses to a CSV file and loading them upon starting the application

Ensure CSV file persists between sessions

ERROR HANDLING – FILE OPERATION HANDLING:

Handle errors related to file reading and writing operations, ensuring that invalid file paths or formats don’t crash the application

ERROR HANDLING - INPUT VALIDATION:

Implement robust error handling for user input errors (e.g., invalid dates, negative  
amounts and category). Ensure that invalid inputs do not crash the application and that users receive appropriate error messages

**FUTURE ENHANCEMENTS**

EXPENSE CREATION: Add option to specific date when creating expense

REPORT GENERATION: User should be able to generate reports showing total expenses for specific periods (daily, weekly, monthly)

**ARCHITECTURE**

**Main Module (**main.py**)**:

This is the entry point of the application. It provides a text-based user interface for interacting with the expense tracker. It handles user input and calls appropriate methods on an ExpenseTracker instance.

**Expense Tracker Module (**expensetracker.py**)**:

Contains the ExpenseTracker class, which manages the list of expenses and categories. Provides methods to add, modify, delete, and view expenses and categories.

**Expense Module (**expense.py**)**:

Contains the Expense class, which represents an individual expense. Each Expense instance has attributes for amount, category, and description.

**Category Module (**category.py**)**:

Contains the Category class, which represents an expense category. Each Category instance has a name attribute.

**File Handling Module (**filehandling.py**)**:

Provides functions to read and write expenses and categories to and from CSV and text files. Ensures data persistence between application runs.

**CLASS INTERACTIONS**

**ExpenseTracker Class**: Manages a list of Expense and Category instances. Provides methods to add, modify, delete, and view expenses and categories. Interacts with the Expense and Category classes to create and manage expense and category objects.

Expense**Class**: Represents an individual expense with attributes for amount, category, and description. Interacts with the Category class to associate an expense with a category.

Category**Class**: Represents an expense category with a name attribute. Used by the Expense class to categorize expenses.

**File Handling Functions**:

* read\_expenses(tracker): Reads expenses from a CSV file and populates the ExpenseTracker instance.
* read\_categories(tracker): Reads categories from a text file and populates the ExpenseTracker instance.
* save\_expenses(tracker): Writes the current list of expenses to a CSV file.
* save\_categories(tracker): Writes the current list of categories to a text file.

**EXAMPLE WORKFLOW**

1. **Initialization**:
   * The [main.py](vscode-file://vscode-app/c:/Users/Jay%20Parikh/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-sandbox/workbench/workbench.html) script initializes an ExpenseTracker instance.
   * It reads existing categories and expenses from files using read\_categories(tracker) and read\_expenses(tracker).
2. **User Interaction**:
   * The user is presented with a menu to add, modify, delete, or view expenses and categories.
   * Based on user input, the main function calls appropriate methods on the ExpenseTracker instance.
3. **Adding an Expense**:
   * The user selects the option to add an expense.
   * The main function collects input for amount, category, and description.
   * An [Expense](vscode-file://vscode-app/c:/Users/Jay%20Parikh/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-sandbox/workbench/workbench.html) instance is created and added to the ExpenseTracker instance using [tracker.add\_expense(expense)](vscode-file://vscode-app/c:/Users/Jay%20Parikh/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-sandbox/workbench/workbench.html" \o ").
4. **Saving Data**:
   * When the user chooses to quit, the main function calls save\_categories(tracker) and save\_expenses(tracker) to persist the data to files.