**INTRODUCTION**

**ARCHITECTURE**

**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 filter and view expenses for specific categories

[R] User should be able to generate reports showing total expenses for specific periods (daily, weekly, monthly) and per category

FILE HANDLING

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

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

**CORE CLASSES:**

**Class Expense:** object to hold individual expenses

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute** | **Data Type** | **Default** | **Validation** |
| amount | Float | 0 | Non-Negative Float |
| category | Class Category Object | Misc | <> |
| description | String | NA | String |

|  |  |  |  |
| --- | --- | --- | --- |
| **Method** | **Arguments** | **Return** | **Logic** |
| init | self, amount, category, description |  |  |
| get\_amount | self | amount as float |  |
| get\_category | self | category as string |  |
| get\_description | self | description as string |  |

**Class Category:** object to hold categories used to track objects

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute** | **Data Type** | **Default** | **Validation** |
| name | String | Misc |  |
| description | String | None |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Method** | **Arguments** | **Return** | **Action** |
| init | self, name, description |  |  |
| get\_name | self | name as string |  |
| get\_description | self | description as string |  |

**Class ExpenseTracker:** main class that handles overall expense tracking functionality

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute** | **Data Type** | **Default** | **Validation** |
| expenses | List |  |  |
| categories | List | “grocery”, “travel” |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Method** | **Arguments** | **Return** | **Logic** |
| add\_expense | self, expense |  |  |
| delete\_expense | self, expense |  |  |
| modify\_expense | self, expense, new\_amount, new\_category, new\_description |  |  |
| add\_category | self, category |  |  |
| view\_expenses | Self, category |  |  |

**MAIN FUNCTIONS**

**TESTING**

**CHALLENGES**

**FUTURE ENHANCEMENTS**

Add option to specific date when creating expense

# Create categories

food\_category = Category("Food")

travel\_category = Category("Travel")

# Create expenses

expense1 = Expense(100, "2023-11-22", food\_category, "Groceries")

expense2 = Expense(50, "2023-11-25", travel\_category, "Train ticket")

# Create an expense tracker

tracker = ExpenseTracker()

# Add categories and expenses to the tracker

tracker.add\_category(food\_category)

tracker.add\_category(travel\_category)

tracker.add\_expense(expense1)

tracker.add\_expense(expense2)

# View expenses

tracker.view\_expenses()

# Calculate total expenses for food category

total\_food\_expense = tracker.calculate\_total\_expenses(food\_category)

print("Total food expense:", total\_food\_expense)

**Enhanced Functionality:**

* The ExpenseTracker class can provide additional features like:
  + Filtering expenses by category or date range.
  + Generating reports in various formats (e.g., CSV, PDF).
  + Calculating total expenses for specific categories or time periods.
  + Implementing user authentication and authorization.
  + Integrating with other financial tools (e.g., budgeting apps, bank accounts).

**Expense Class Responsibilities:**

* **Represents a single expense:** Stores information about an individual expense, such as its amount, date, category, and description.
* **Provides access to expense details:** Offers methods to retrieve the specific details of an expense, including its amount, date, category, and description.

**Category Class Responsibilities:**

* **Represents a category of expenses:** Stores information about a specific category, such as its name and description.
* **Provides access to category details:** Offers methods to retrieve the name and description of a category.
* **Organizes expenses:** Categorizes expenses into different groups, making it easier to analyze spending patterns and track financial habits.