Sai Rithwik Kukunuri

CSE 205 Object Oriented Programming and Data Structures

Professor Ryan Meuth

Personal Finance Manager (Budgeting and Expense Tracker)

INTRODUCTION

The Personal Finance Manager Application is a desktop-based Java application designed to assist users in managing their personal expenses. This application allows users to track their daily expenses, set budgets, and store financial records in CSV format. It provides a user-friendly graphical interface, and simplifies the process of managing expenses, analyzing budgets, and exporting financial data.

PROBLEM STATEMENT

In today's fast-paced world, individuals face increasing complexity in managing their day-to-day financial transactions. The challenge is to create a user-friendly platform that simplifies personal finance manager by focusing on essential features such as expense tracking, budget monitoring, and export expenses. The goal is to provide users with an intuitive solution that empowers them to take control of their finances without being overwhelmed by unnecessary complexity.

Steps Taken:

- Identified key issues with existing tools, such as complexity and lack of customization.
- Defined the primary goal to create a desktop application that helps users track expenses, monitor budgets, and export/import data with ease.

RESEARCH AND GATHER INFORMATION

I have analyzed the existing tools and reviewed studies on personal finance manager systems. Over my research, popular tools like Mint often face concerns around data security and the cost of premium features. In contrast, my application is completely free to use, with no hidden fees. I have focused on data security by ensuring that all user information remains safe and unaffected, as it operates entirely within the local environment. The application is also simple and easy to use, making it accessible to anyone, regardless of their technical skills.

BRAINSTORMING

Simplified GUI Layout

- A single window interface with a dynamic expense table.
- Buttons for key actions like adding expenses, viewing budgets, and exporting data.

Expense Model Design

- Attributes: Amount, Category, Date, and Description.
- Stored as a memory list for easy access.

CSV File Handling

- Methods for importing expenses from existing records.
- Export functionality for saving expense data in a standardized format.

Final Concept

A Java Swing application with an user friendly interface, seamless data handling, and budget tracking features.

DESIGN

User Interface Overview

The user interface is designed to provide a clear way for users to manage their finances. The layout includes

- **Buttons** for adding expenses, setting budgets, clearing expenses, exporting , and loading data.
- **Remaining Budget Display**, which dynamically updates based on user inputs.
- **Table** that lists all recorded expenses, including details like the amount, category, date, and description.

This design ensures users can easily interact with the system while maintaining a clear overview of their financial status.

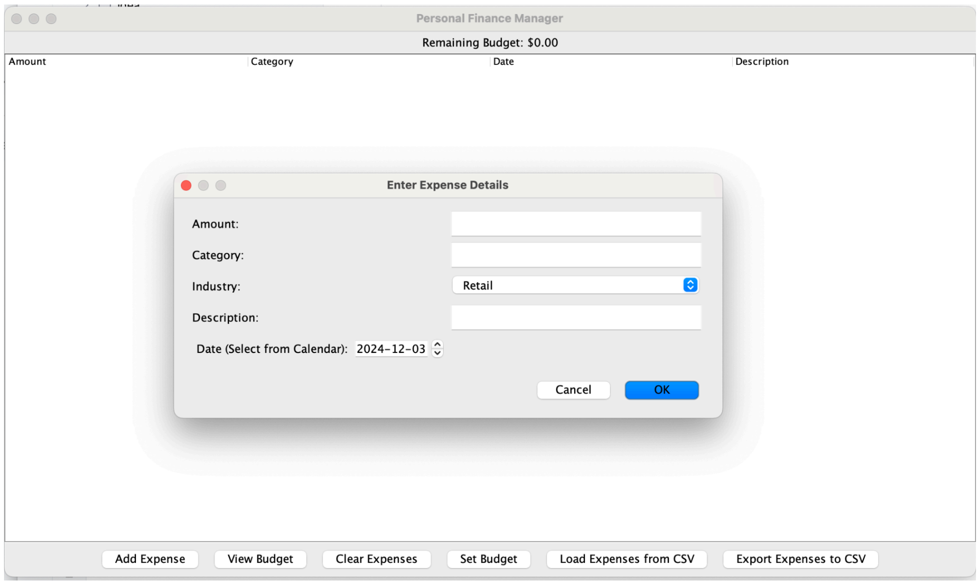

FIGURE: 1 User Interface of the Personal Finance Project

It defines the relationships between key components such as expense management, budget handling, and file operations, while showing the responsibilities of each class in terms of attributes and methods. This diagram serves as a blueprint for the system's implementation

Diagram Overview

Key Components and Classes:

1. MainApp
    - This is the primary class that acts as an entry point for my application. It handles user interface interactions , including the updating the budget, managing expenses, and exporting or loading data.
    - Key methods:
        - **setBudget():** Allows users to set a new budget.
        - **addExpense():** Adds an expense to the tracker
        - **exportCSV() and loadExpensesFromCSV():** Handle file operations for saving and loading data

2. ExpenseHandler
    - Manages a collection of expenses and performs operations like adding, clearing, and calculating totals.
    - Attributes
        - **expenses:** a list of Expense object
    - Methods
        - **getTotalSpent():** Calculates the total expenses.
        - **addExpense(Expense):** Adds a new expense to the list.

3. BudgetHandler
    - Handles operations related to budget calculations and updates.
    - Key methods:
        - **getRemainingBudget(double, double):** Calculates the remaining budget based on expenses.
        - **getTotalSpent(List<Expense>):** Summarizes all expenses for a given period.

4. ExpenseTrackerService
    - Serves as a central service that interacts with other classes to manage expenses, budgets, and file operations.

- Methods
    - **getTotalSpent():** Calculates the total expenses.
    - **getRemainingBudget(double):** Returns the updated remaining budget.
    - **exportExpensesToCSV():** Calls the CSVHandler to export data.

5. Expense
    - Represents an individual financial transaction with attributes like description, category, date, and amount.
    - Attributes:
        - **description**: A description of the expense
        - **category**: The type of expense
        - **amount**: The amount spent
        - **date**: The date of expense
    - Methods:
        - **toCSV():** Converts the expense data into a CSV-compatible format.
        - **getAmount():** Returns the amount spent.
6. CVSHandler
    - Handles reading and writing expense data from and to CSV files.
    - Methods:
        - **exportExpensesToCSV(String, List<Expense>):** Exports the list of expenses to a file.
        - **loadExpensesFromCSV(String):** Loads expense data from a file.

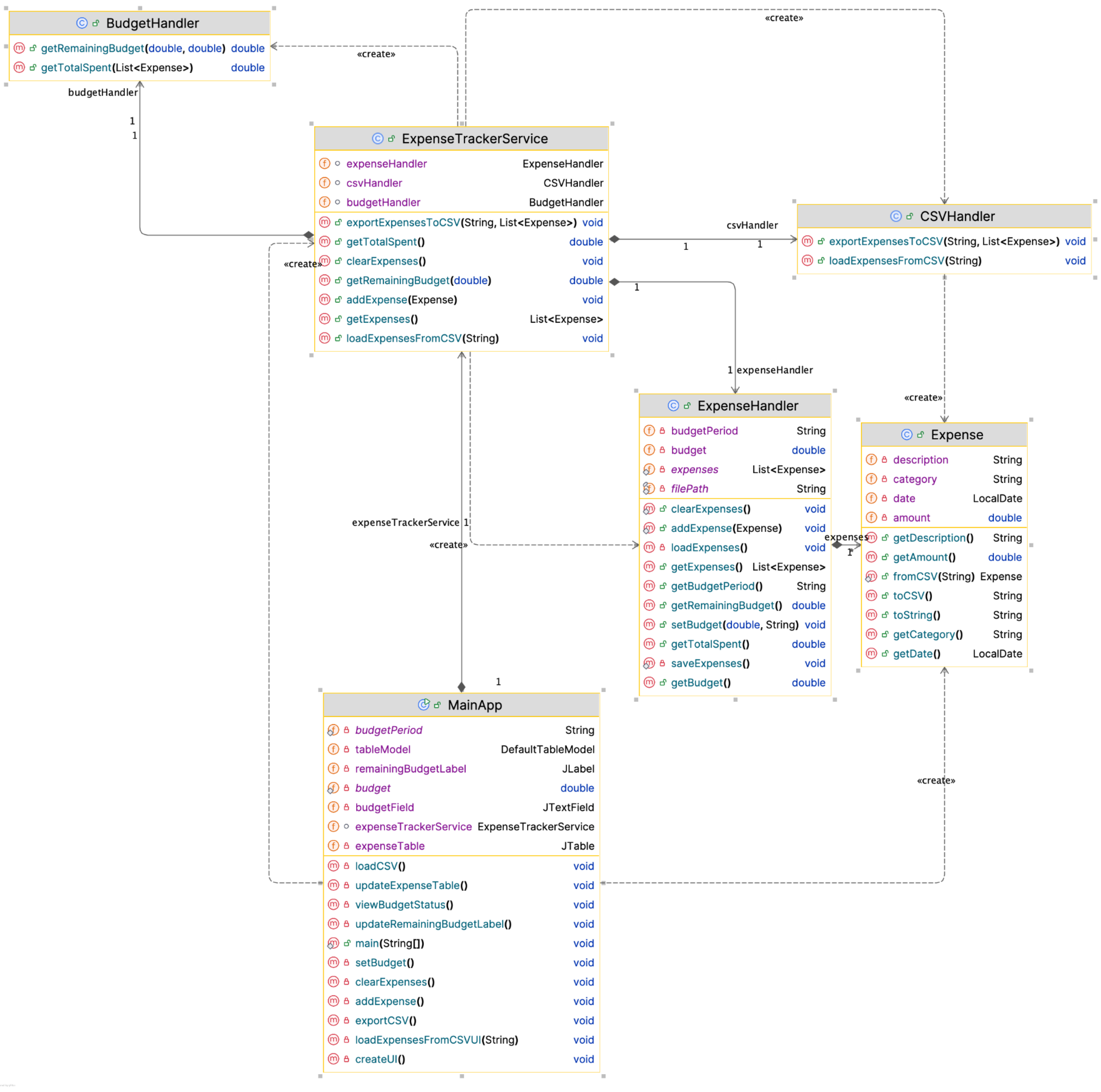

FIGURE: 2 UML Class Diagram for Personal Finance Project

TECHNOLOGIES

1. **Java (JDK 8 or Higher):**

   Java was used as the primary programming language for this project. Java's features and libraries make it great for building desktop applications. Along with that, I used Java’s Input/Output (I/O) libraries to handle reading and writing CSV files.

Components Used:

- BufferedReader: For reading data from a file.
- BufferedWriter: For writing data to a file.
- FileReader & FileWriter: For file operation

2. **Swing (For Graphical User Interface)**

   Swing is part of the Java Standard Library, providing a set of GUI components for creating interactive user interfaces.
   Components Used:

   - JButton: For action buttons
   - JTable: To display a list of expenses.
   - JTextField, JLabel, JPanel: For text inputs and labels.

DEVELOP AND PROTOTYPE SOLUTIONS
Implementation Process:

1. Week 1-2:
   - Designed the Expense class to model individual expenses.
   - Created the main GUI layout using the Java Swing.
2. Week 3-4
   - Added functionality to track expenses, including dynamic table updates.
   - Implemented budget-setting features with remaining budget calculations
3. Week 5-6
   - Integrated file handling for CSV import/export.
   - Added error handling for invalid inputs and file operations.
   - Testing the Application

Prototype Highlights:

- Fully functional GUI with a real-time budget tracker.
- Ability to add, clear, import, and export expense data.
- Remaining budget displayed dynamically based on recorded expenses.

TEST AND EVALUATE

The development process integrated rigorous testing to ensure the functionality, and seamless user experience. Various testing methods were employed:

1. Unit Testing:
    - Verified the functionality of individual components, including critical methods like addExpense, loadCSV , and exportCSV
    - Ensured operations worked correctly without external dependencies.
2. Integration Testing
    - Assessed the interaction between the graphical user interface (GUI) components and the backend logic.
    - Validated data flows from user inputs through the application's layers to ensure consistency.
3. User Testing
    - Conducted the hands-on sessions with real users.
    - Gathered feedback on usability, design intuitiveness, and feature improvements.
    - Identified and resolved minor bugs based on user feedback.

Key Test cases

The following scenarios were evaluated to ensure robust application behavior:

Scenario	**Expected Result**	**Outcome**
Add a valid expense	Expense is added and displayed in the table	Passed
Import a valid CSV file	Expenses are populated in the application	Passed
Set a valid budget	Budget is updated and displayed accurately	Passed
Export expenses to CSV	CSV file is saved with accurate expense details.	Passed
Import an invalid CSV file	An error message is displayed to the user.	Passed

RESULTS

- The application successfully passed all major test cases.
- Ensured full compliance with functional requirements, establishing the application's ease of use.
- User feedback during testing confirmed that the application's core features were effective.

The application successfully passed all major test cases. The complete codebase is available in the GitHub repository: https://github.com/YB-Yottabyte/personal-finance, and a demo of my Project: https://youtu.be/EAWhkgJs6cg

FUTURE IMPROVEMENTS

To further enhance the desktop application and expand its capabilities, the following improvements are planned:

1. Advanced Data Visualization: Integration of Chart.js or D3.js to offer user interactive charts to analyze spending trends and budget performance. Along with that, I will add a feature where users receive monthly reports with in depth insights into their spending, and recommendations on how to optimize their budget.
2. Spending Predictions: The app uses AI-based algorithms to predict the future spending based on historical data. If the user is new, the app can ask a series of questions to better understand their spending pattern and preferences, which then allows the system to make more accurate predictions.

This app allows users to take full control of their finances with personalized, and proactive tools. Unlike existing solutions, it's designed to provide continuous immediate adjustments to ensure users stay within their budget and achieve their financial goals more easily.