Personal Finance Management System

Lecturer :
D4017_Jude Joseph Lamug Martinez, MCS

Arranged by :

2702368122_Vammy Johannis Jiang

Object Oriented Programming

Faculty of Computer Science

Binus International University 2024/2024

Table of Contents

Chapter 1 – Project Specification

1.1 Introduction

This project was developed using the Java programming language with the primary goal of creating a comprehensive Personal Finance Manager. The application is designed to help users in managing their personal finances by providing features for tracking expenses, setting and monitoring financial goals, and analyzing spending patterns.

1.2 Idea Inspiration

The idea came from myself when I was having trouble listing my monthly financial record. I decided to create in order to help myself in tracking my finances more efficiently.

1.3 How it Works

The project functions by leveraging Java Swing to create an intuitive and visually appealing user interface. The application interacts with serialized data files, allowing users to perform various actions. These actions include:

Viewing and Managing Transactions
The application retrieves a list of the user's financial transactions and displays them. Users can browse through their transactions, view details such as type, category, amount, date, and description, and manage their records efficiently.

Adding New Transactions
Users can add new financial transactions by specifying details such as the type (income or expense), category, amount, date, and description. The application ensures that the new transaction is added to the user's record and updates the display accordingly.

Managing Financial Goals
Users can set financial goals by specifying target amounts and target dates. The application allows users to track their progress towards these goals and view the status of each goal.

Exporting and Importing Data
The application provides functionality to export the user's financial data to a CSV file for external use and to import data from a CSV file.

Expense Analysis
The application offers an analysis of the user's expenses, including calculating total expenses, average monthly expenses, and expenses by category. This helps users to understand their spending patterns and make informed financial decisions.

Chapter 2 – Solution Design

2.1 Solution Overview

The program consists of 12 components :

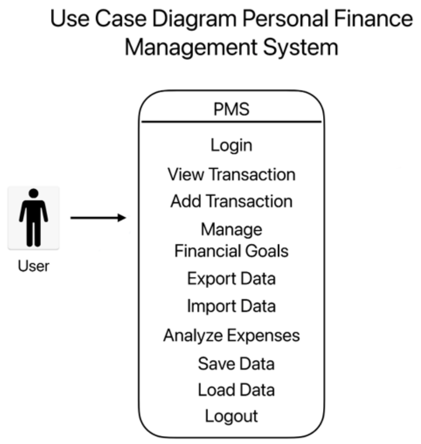
- User
- Transaction
- FinancialGoal
- MainDashboard
- AddTransactionWindow
- ViewTransactionsWindow.
- AddFinancialGoalDialog
- LoginWindow
- SerializationUtil
- DataHandler
- ExpenseAnalyzer
- GUI

2.2 Solution Description

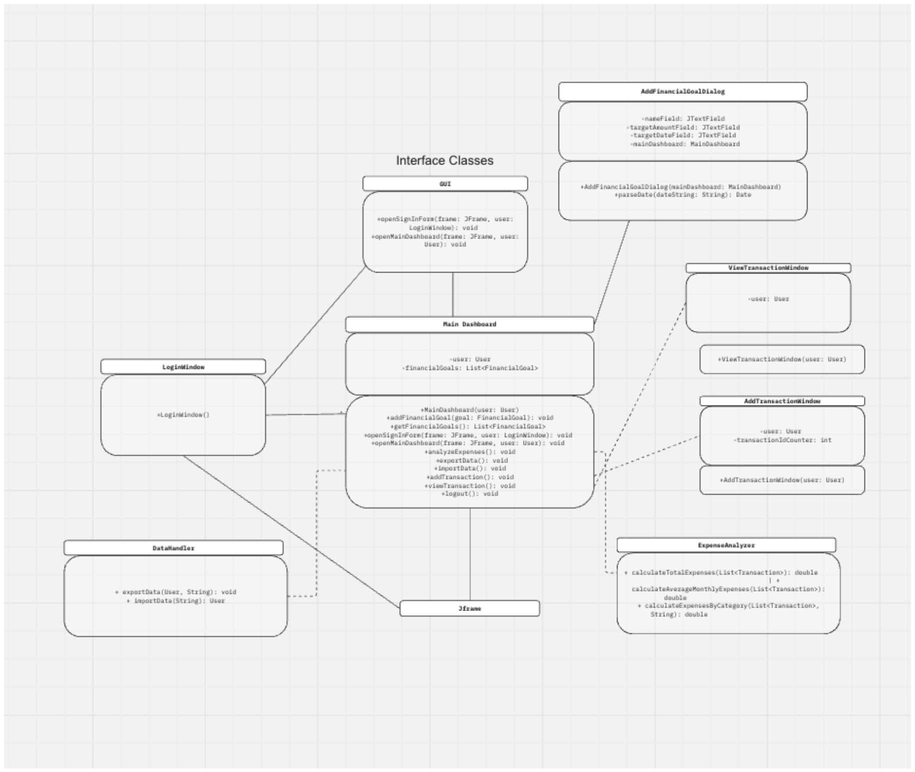
1. User
   Manages user information and transactions.
2. Transaction
   Represents a financial transaction with details such as type, category, amount, date, and description.
3. FinancialGoal
   Represents a financial goal with properties like name, target amount, target date, current amount, and status.
4. MainDashboard
   The main dashboard of the application, providing access to various functionalities like adding transactions, viewing transactions, and managing financial goals.
5. AddTransactionWindow
   A window for adding new transactions.
6. ViewTransactionsWindow
   A window for viewing existing transactions.
7. AddFinancialGoalDialog
   A dialog for adding new financial goals.
8. LoginWindow
   A window for user login.
9. SerializationUtil
   A utility class for serializing and deserializing objects to and from files.
10. DataHandler
    Handles data export and import functionality, specifically for CSV files.

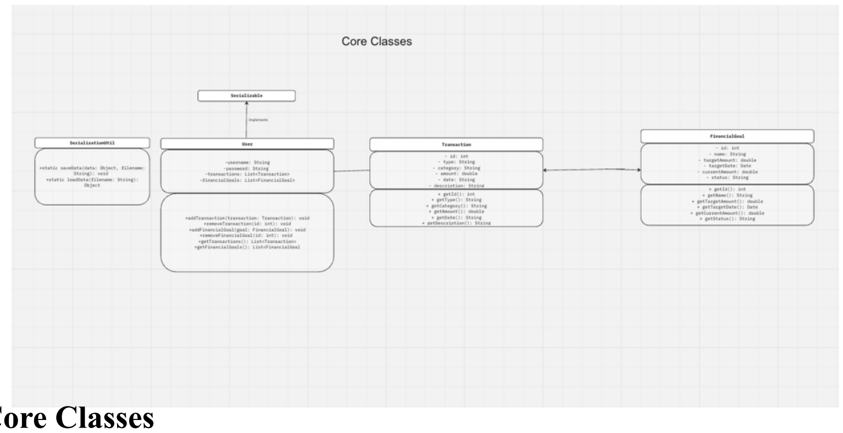
11. ExpenseAnalyzer
    Provides methods for analyzing expenses, such as calculating total expenses, average monthly expenses, and category-specific expenses.
12. GUI
    Contains methods for setting up the graphical user interface components with a Macintosh-inspired look and feel.

2.3 Use Case Diagram



2.4 Class Diagram


Interface Classes
GUI
+openSignInForm(frame: JFrame, user: LoginWindow): void
+openMainDashboard(frame: JFrame, user: User): void
AddFinancialGoalDialog
-nameField: JTextField
-targetAmountField: JTextField
-targetDateField: JTextField
-mainDashboard: MainDashboard
+AddFinancialGoalDialog(mainDashboard: MainDashboard)
+parseDate(dateString: String): Date
Main Dashboard
-user: User
-financialGoals: List<FinancialGoal>
+MainDashboard(user: User)
+addFinancialGoal(goal: FinancialGoal): void
+getFinancialGoals(): List<FinancialGoal>
+openSignInForm(frame: JFrame, user: LoginWindow): void
+openMainDashboard(frame: JFrame, user: User): void
+analyzeExpenses(): void
+exportData(): void
+importData(): void
+addTransaction(): void
+viewTransaction(): void
+logout(): void
LoginWindow
+LoginWindow()
ViewTransactionWindow
-user: User
+ViewTransactionWindow(user: User)
AddTransactionWindow
-user: User
-transactionIdCounter: int
+AddTransactionWindow(user: User)
DataHandler
+ exportData(User, String): void
+ importData(String): User
Jframe
ExpenseAnalyzer
+ calculateTotalExpenses(List<Transaction>): double | + calculateAverageMonthlyExpenses(List<Transaction>): double
+ calculateExpensesByCategory(List<Transaction>, String): double


Core Classes
Serializable
Implements
SerializationUtil
+static saveData(data: Object, filename: String): void
+static loadData(filename: String): Object
User
-username: String
-password: String
-transactions: List<Transaction>
-financialGoals: List<FinancialGoal>
+addTransaction(transaction: Transaction): void
+removeTransaction(id: int): void
+addFinancialGoal(goal: FinancialGoal): void
+removeFinancialGoal(id: int): void
+getTransactions(): List<Transaction>
+getFinancialGoals(): List<FinancialGoal
Transaction
- id: int
- type: String
- category: String
- amount: double
- date: String
- description: String
+ getId(): int
+ getType(): String
+ getCategory(): String
+ getAmount(): double
+ getDate(): String
+ getDescription(): String
FinancialGoal
- id: int
- name: String
- targetAmount: double
- targetDate: Date
- currentAmount: double
- status: String
+ getId(): int
+ getName(): String
+ getTargetAmount(): double
+ getTargetDate(): Date
+ getCurrentAmount(): double
+ getStatus(): String

Core Classes

1. **User Class**

The User class represents a user of the personal finance management system. It holds the user's credentials (username and password) and maintains a list of their financial transactions. The User class includes methods to get the username, password, and transactions, as well as to add a transaction to the user's list.

2. **Transaction Class**

The Transaction class represents an individual financial transaction made by the user. It includes details such as the date of the transaction, the amount, and the category (e.g., groceries, entertainment). Each Transaction instance is associated with a User and is stored in the user's list of transactions.

3. **FinancialGoal Class**

The FinancialGoal class represents a financial goal set by the user. It includes details such as the name of the goal, the target amount, the current amount saved towards the goal, the target date to achieve the goal, and the status of the goal (e.g., in progress, achieved).

Relationships Between Classes

User and Transaction:

- The User class has a one-to-many relationship with the Transaction class. This means that a single user can have multiple transactions.
- The User class maintains a list of Transaction objects, indicating that transactions are part of the user's financial activities. The transactions attribute in the User class aggregates multiple Transaction instances.

- The User class provides methods to get the list of transactions and to add a new transaction (addTransaction).

User and FinancialGoal:

- The User class can also have multiple financial goals. Each user can set and track several financial goals.
- Similar to transactions, the user's financial goals are aggregated within the User class, though the example doesn't include a direct list of goals in the User class, it is managed in the MainDashboard class.
- The MainDashboard class has methods to add a financial goal and to get the list of financial goals.

MainDashboard

Description: The MainDashboard class extends JFrame and serves as the main user interface of the application. It allows users to navigate through different functionalities such as adding/viewing transactions, managing financial goals, saving/loading data, and performing expense analysis.

LoginWindow

Description: The LoginWindow class extends JFrame and provides the user interface for logging into the application. It verifies the user's credentials and grants access to the MainDashboard if the credentials are correct.

AddTransactionWindow

Description: The AddTransactionWindow class extends JFrame and provides a form for users to add new financial transactions. It includes fields for the transaction date, amount, category, and description.

ViewTransactionsWindow

Description: The ViewTransactionsWindow class extends JFrame and displays a list of all transactions made by the user. It allows users to view details of their past transactions.

AddFinancialGoalDialog

The AddFinancialGoalDialog class extends JDialog and provides a form for users to add new financial goals. It includes fields for the goal name, target amount, target date, and initial amount saved.

DataHandler

The DataHandler class is responsible for exporting and importing user data to/from JSON files. It provides methods to export user data to a JSON file and import data from a JSON file.

ExpenseAnalyzer

The ExpenseAnalyzer class contains static methods for analyzing financial transactions. It provides methods to calculate total expenses, average monthly expenses, and expenses by category, helping users understand their spending patterns.

Chapter 3 - Libraries Used

- **Java Standard Library:**
  java.awt: Used for graphical user interface components.
  javax.swing: Used for building the GUI.
  java.io: Used for file input and output operations, particularly for serialization.
  java.util: Used for data structures such as ArrayList, List, and utility classes like Date.
  Java Swing:

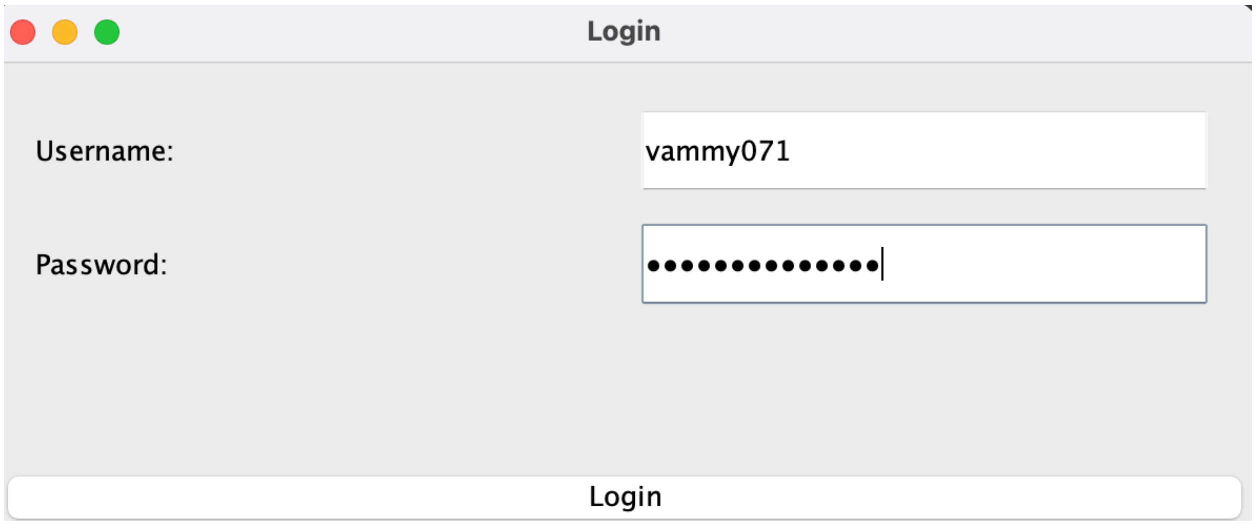
- **Swing** is used extensively for creating the graphical user interface, including components like JFrame, JPanel, JButton, JLabel, JTextField, JPasswordField, and JOptionPane.
- **Java AWT:**
  AWT components such as GridBagLayout, GridBagConstraints, BorderLayout, GridLayout, Color, Cursor, and Insets are used for layout management and styling.

- **Java IO:**
  ObjectOutputStream and ObjectInputStream for serialization.
  FileOutputStream and FileInputStream for file handling.

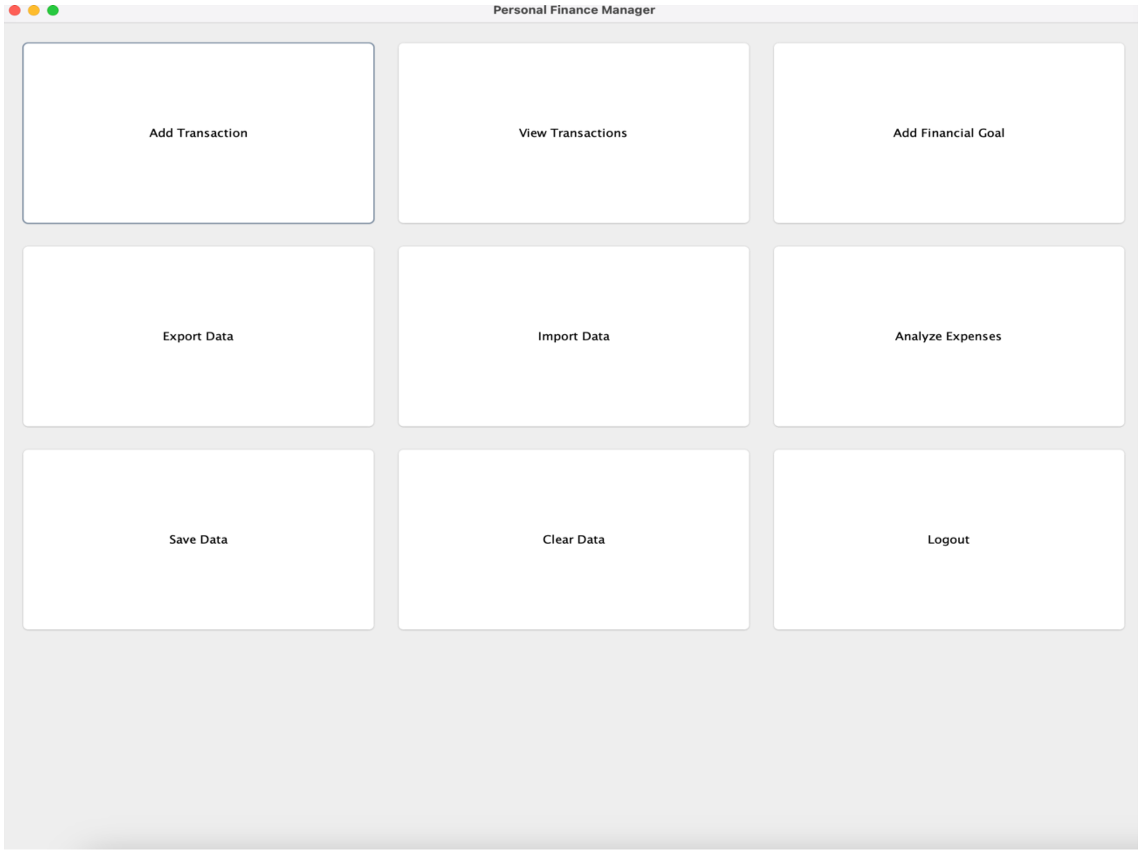
I

Chapter 4 - Evidence of Working Program

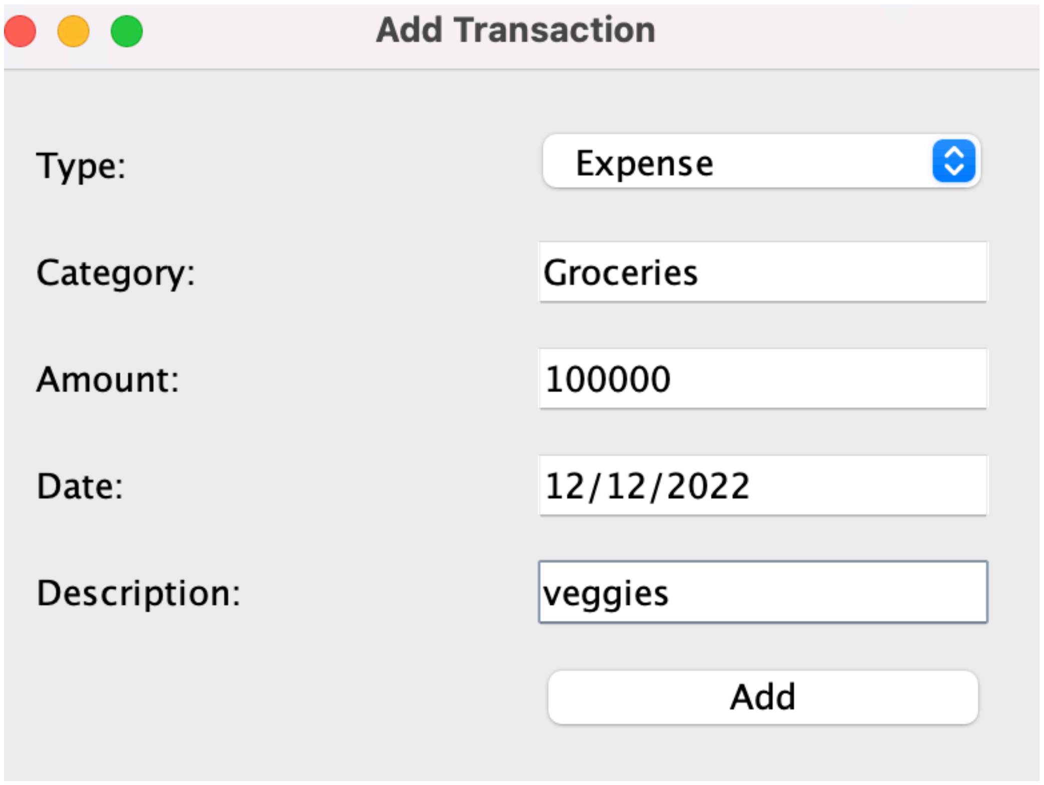
1. User login into the program



2. User arriving in the main dashboard

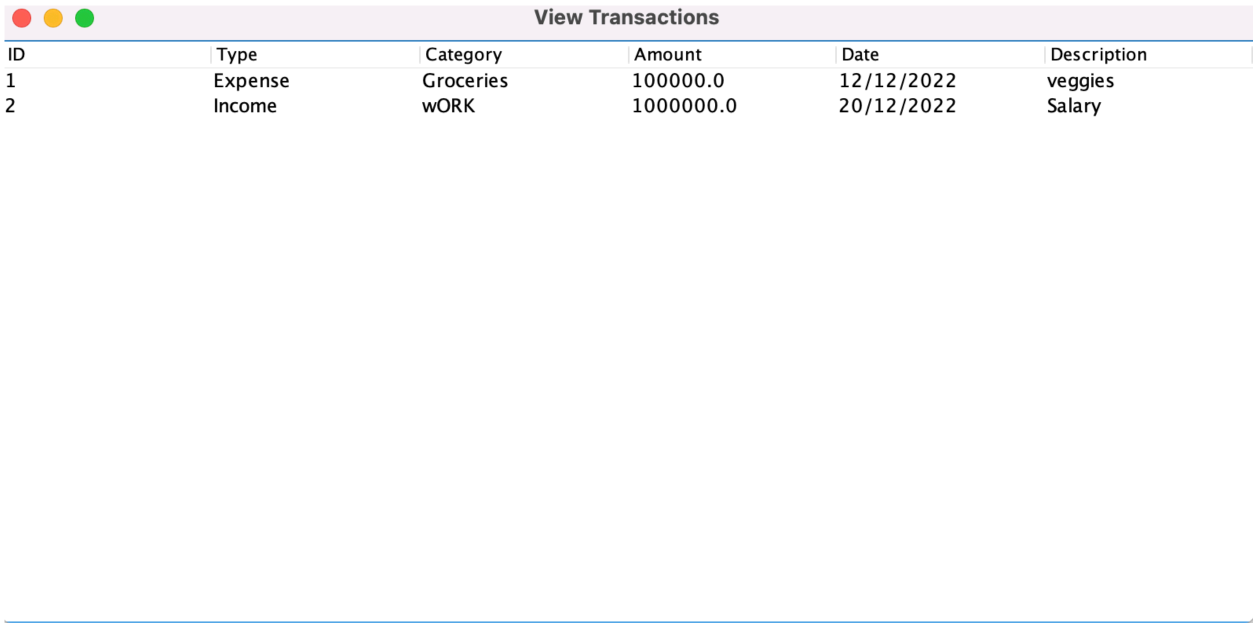


3. User can add transactions



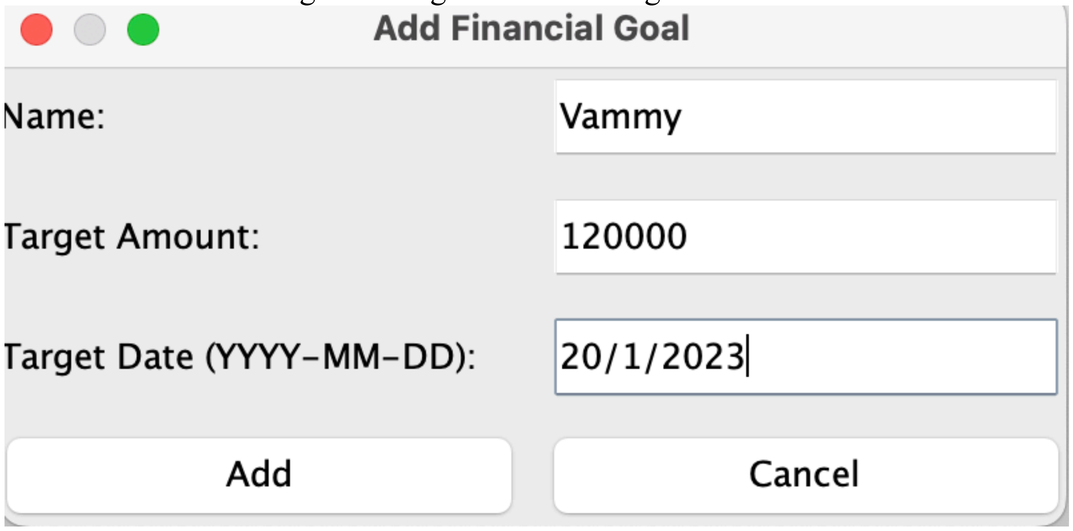
Type:	Expense
Category:	Groceries
Amount:	100000
Date:	12/12/2022
Description:	veggies

4. User can view transactions added through view transactions button



ID	Type	Category	Amount	Date	Description
1	Expense	Groceries	100000.0	12/12/2022	veggies
2	Income	wORK	1000000.0	20/12/2022	Salary

5. User can add financial goal through add financial goal button



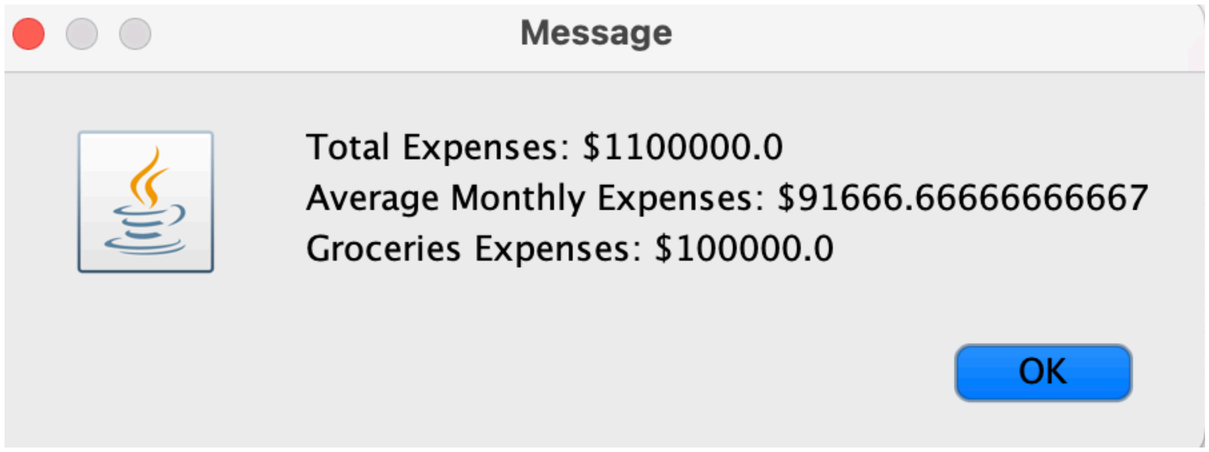
Add Financial Goal

Name: Vammy

Target Amount: 120000

Target Date (YYYY-MM-DD): 20/1/2023

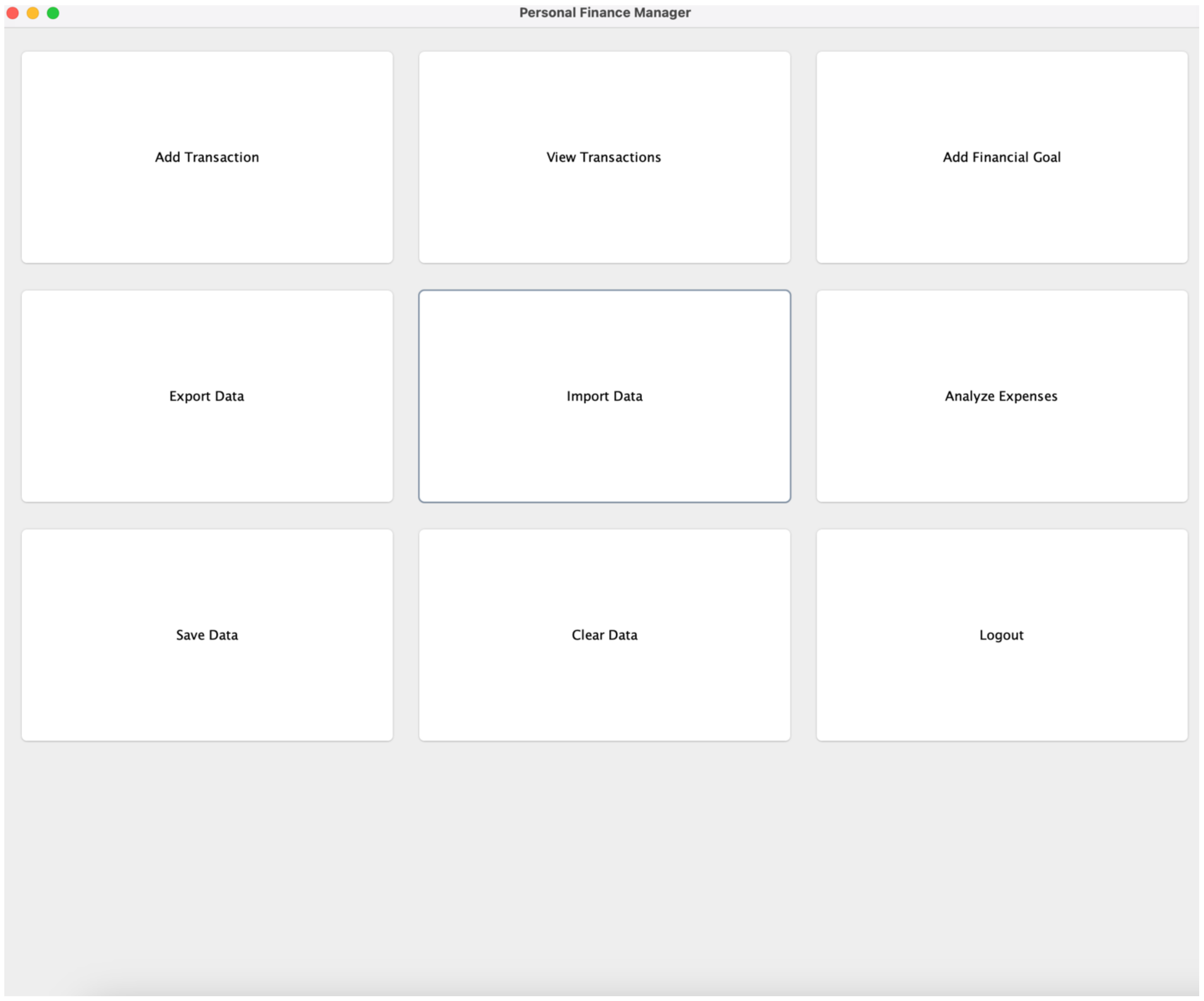
6. User can see total expenses and average monthly expenses through expense analyzer button



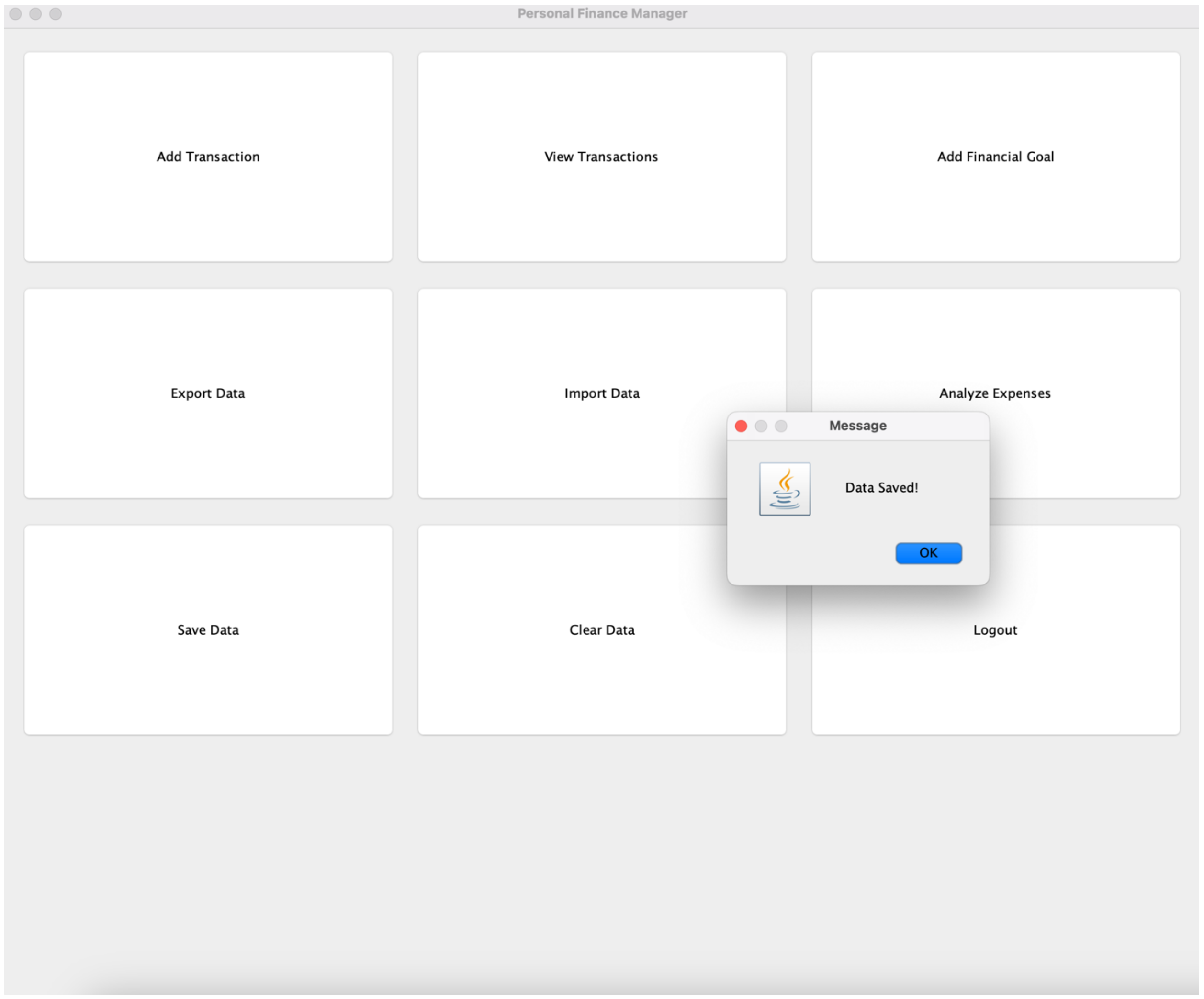
Message

```
Total Expenses: $1100000.0
Average Monthly Expenses: $91666.66666666667
Groceries Expenses: $100000.0
```

7. User can use the export data to export the transactions and import external data into the program through export data button and import data button

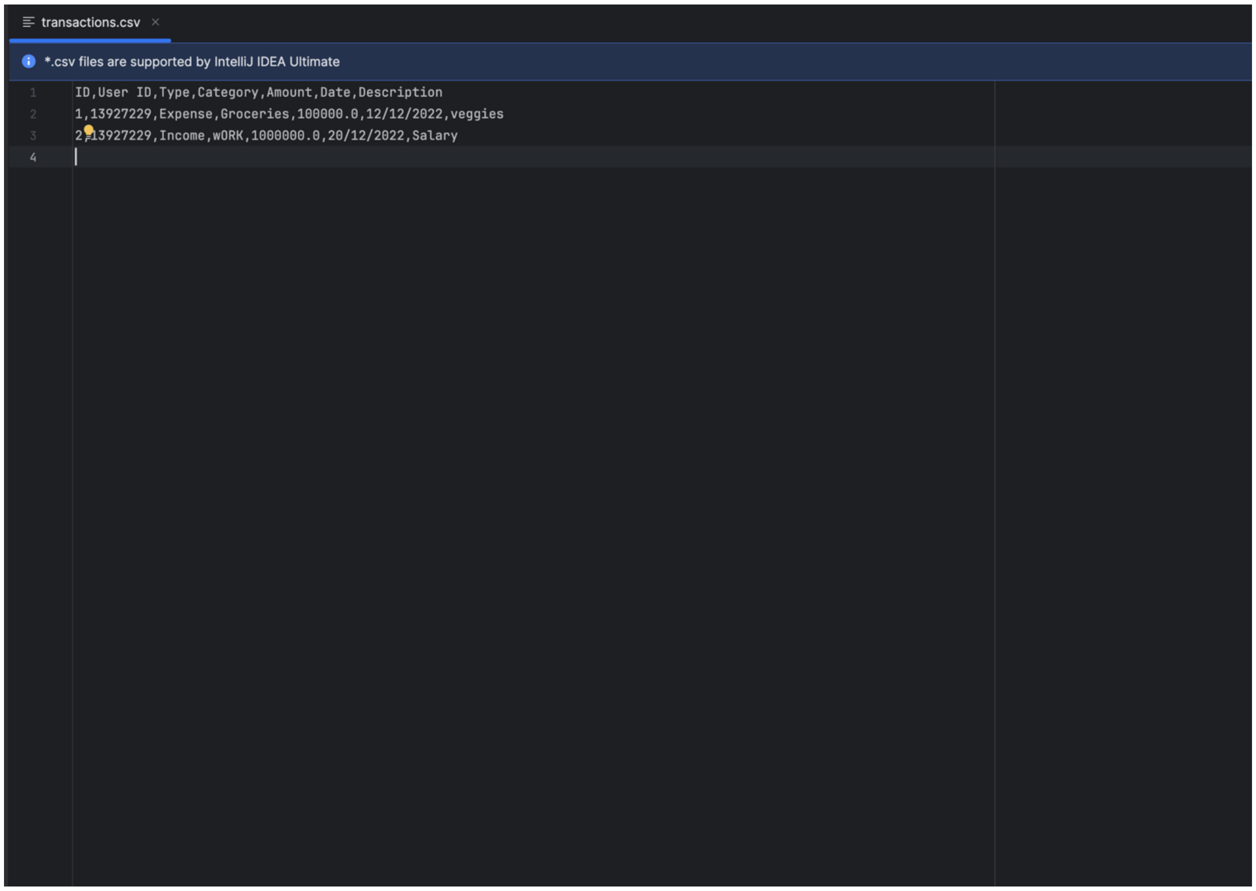


8. Users can logout through the logout button



Data Saved!

9. User can see the exported data through transactions.csv



transactions.csv

*.csv files are supported by IntelliJ IDEA Ultimate

```
ID,User ID,Type,Category,Amount,Date,Description
1,13927229,Expense,Groceries,100000.0,12/12/2022,veggies
2,13927229,Income,wORK,1000000.0,20/12/2022,Salary

```

I

Chapter 5 - Lesson Learned

In the process of making this project, I have gained valuable insights and skills, particularly in Java programming and software development:

- Graphical User Interface (GUI) Design: I learned to design and implement user-friendly interfaces using Java Swing and AWT libraries.
- Data Serialization and Deserialization: Understanding how to serialize and deserialize objects was crucial for saving and loading user data. This involved using Java's java.io package to perform file operations, ensuring data persistence between sessions.
- Object-Oriented Programming (OOP): The project helped me further learn about OOP principles by designing classes like User, Transaction, FinancialGoal, and others.
- Expense Analysis: Implementing the ExpenseAnalyzer class provided insights into data analysis
- Data Import and Export: Adding functionality for data import and export with the DataHandler class taught me how to handle external data sources.

Overall, this project has significantly contributed to my growth as a programmer. It has enabled me to dive deeper into software development and motivated me to explore new area of expertise.

Chapter 6 - Source Code and Video Demo

GitHub Link : https://github.com/TanaRuin/Final-Project-OOP

Poster:
https://drive.google.com/file/d/1T80Hrm3jYdKHlTzAFGsOz1zRbiJrJ6_m/view?usp=drive_link

References

Pasko Zhelev. (2017). Final Project - CS50 (Personal Finance Application made with Java) [YouTube Video]. In *YouTube*. https://www.youtube.com/watch?v=c9RqO4GRfdw

Unique Developer. (2019). Fees Management System in java | Complete Project | part [YouTube Video]. In *YouTube*. https://www.youtube.com/watch?v=EgqjBip0rTo&list=PLjrrZBv_CFYR0zHNF7fqO8O4PZPOAs5q8

RyanSilva2004/Personal-Money-Management-System-Java-Netbeans: This repository hosts a personal money management system built in Java with NetBeans. It tracks expenses, income, financial goals, and generates reports for effective financial planning. (2024). GitHub. https://github.com/RyanSilva2004/Personal-Money-Management-System-Java-Netbeans