Project Report

Programming in java

(CSE2006)

Submitted by:

NAME: Piyush Aggarwal

REG No: 24BCE10387

Bachelor of Technology

In

Computer Science & Engineering

School of Computer Science and Engineering

VIT Bhopal University

Introduction

This project is a lightweight Expense Tracker built using Java Swing. It allows users to manage income and expenses locally.

Problem Statement

Users need an offline, easy-to-use desktop tool to track daily finances without relying on cloud services.

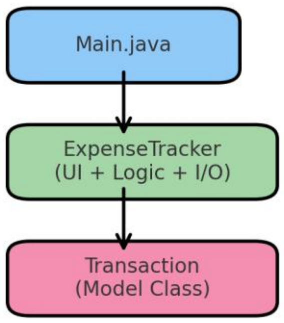
Functional Requirements

- Add Transaction
- Remove Transaction
- View Transactions
- Save CSV
- Load CSV
- Generate Report

Non-functional Requirements

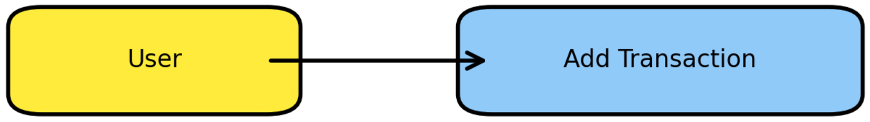
- Usability
- Reliability
- Maintainability
- Performance

System Architecture

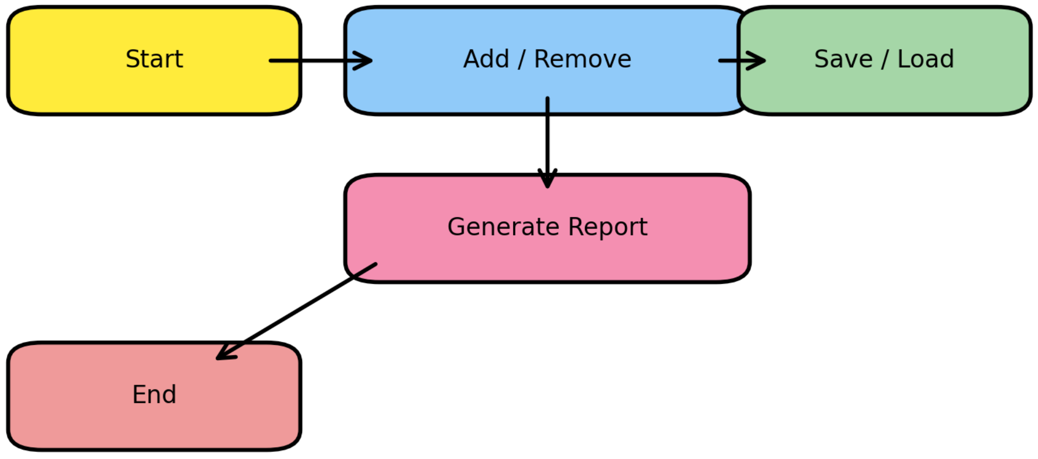


Design Diagrams

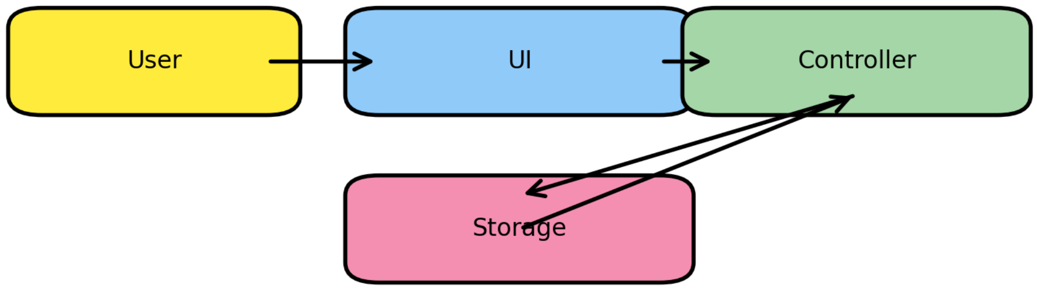
Use Case Diagram



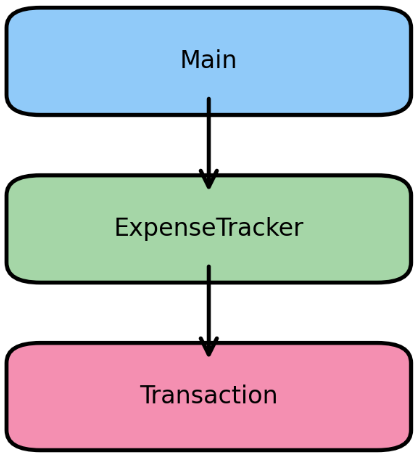
Workflow Diagram



Sequence Diagram

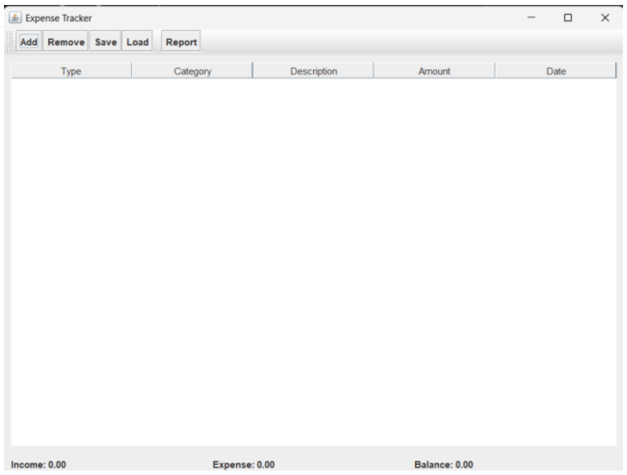


Class Diagram



Screenshots / Results

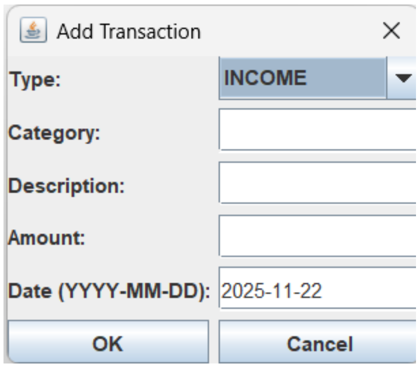
Main UI Screenshot



Type	Category	Description	Amount	Date

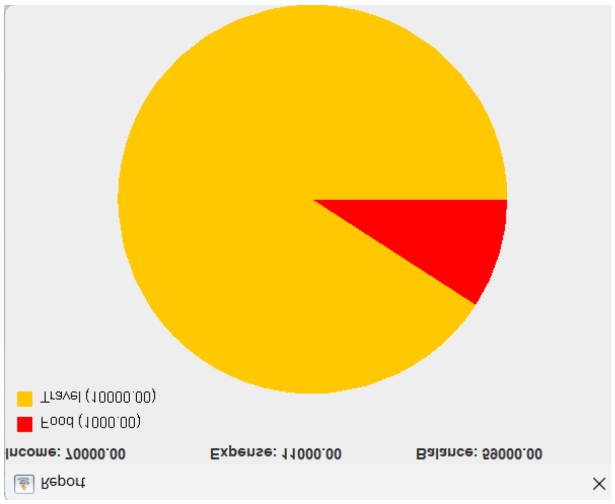
Income: 0.00 Expense: 0.00 Balance: 0.00

Add Transaction Dialog



Type:	INCOME
Category:	
Description:	
Amount:	
Date (YYYY-MM-DD):	2025-11-22

Report Window



Income: 70000.00 Expense: 11000.00 Balance: 59000.00

Design Decisions & Rationale

The project intentionally follows a simplified architecture to ensure clarity, ease of maintenance, and suitability for academic demonstration. Rather than distributing responsibilities across numerous classes, the UI, controller logic, and CSV-based storage were combined into a single primary class (*ExpenseTracker*). This decision reduces structural complexity and minimizes overhead while still clearly separating the core data model through the dedicated *Transaction* class.

The use of Swing was chosen due to its native support in Java, portability across platforms, and the absence of dependency requirements. Storing data in CSV format was selected over databases to maintain transparency, allow easy inspection of saved data, and support lightweight usage without external drivers or setup.Implementation Details.

The app uses Swing for UI and CSV for storage.

Testing Approach

Testing relied on manual, scenario-based validation to ensure that all functional requirements were met. Several test cases were executed, including:

- Adding income and expense entries with valid and invalid inputs.
- Attempting numeric conversions for amount fields.
- Entering invalid dates and observing fallback behavior.
- Saving data to CSV and validating its structure
- Loading CSV files with correct, missing, or corrupted lines..
- Opening the report window with zero, few, and many expense entries.

The UI was also evaluated for responsiveness, error-handling behavior, and stability during rapid actions such as repeated entry additions or loading files in succession

Challenges Faced

One notable challenge involved avoiding a naming conflict with Window.getType(), requiring a redesign of the accessor method in the dialog class. Implementing the pie-chart manually using Graphics2D also required careful handling of arc calculations, color assignment, and layout management.

Managing the interaction between table updates, status summaries, and CSV synchronization required attention to ensure consistency and prevent data mismatches.

Learnings & Key Takeaways

he project strengthened understanding of Java Swing event-driven programming, custom component rendering, and multi-class coordination. Hands-on experience in file I/O reinforced concepts related to data serialization, input validation, and error handling.

Additionally, designing UI workflows improved insight into user experience considerations, layout planning, and creating intuitive interfaces in desktop applications.

Future Enhancements

Several improvements can be incorporated to extend the application:

- An **Edit Transaction** feature to modify existing entries.
- Category- or date-based filters to allow monthly and custom reporting.
- Migration from CSV to **SQLite** for more robust data management.
- Support for exporting detailed reports in **PDF** format.
- Improved themes and interface customization.
- More detailed analytics, including bar charts and multi-category comparisons.

These enhancements would significantly expand functionality while maintaining the simplicity of the current design.

References

- Oracle Java Documentation.
- Java Swing Official Guides.
- TutorialsPoint – Java Swing.
- Oracle I/O and File Handling Documentation.
- Stack Overflow developer discussions.
- Reddit programming and Java communities (for conceptual clarifications and best practices).
- GitHub repositories and examples (for reviewing common project structures and design patterns).