Smart Console-Based Expense Tracker (Core Java Project)

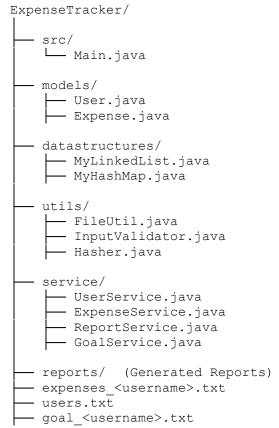
Objective

This project is a **console-based Expense Tracker built entirely in Core Java**. It does not use Java Collections or external libraries and involves implementing custom data structures, file-based persistence, and real-world expense tracking features.

Features Implemented

- 1. User Registration & Login with simple password hashing.
- 2. Add Expenses with amount, category, description, date, recurring flag.
- 3. Generate Monthly Reports (category totals, total expense, highest expense).
- 4. Category Insights (most used category, average per category).
- 5. Savings Goal Feature with goal comparison.
- 6. Recurring Expense Tracking.
- 7. Sorting of Expenses by Amount and Date (using bubble sort).
- 8. Data Persistence in .txt files.

Folder Structure



Custom Data Structures

1. MyLinkedList<T> (<T> called generics)

Since I could not use Java's built-in ArrayList or LinkedList, I created my own linked list from scratch. This structure is used to manage lists of Expenses, Users, and file data lines.

How does it work?

Node class: It has a data field and a pointer next to the next node.

The MyLinkedList<T> manages a chain of these nodes.

Supports:

- $-add(T data) \rightarrow Adds$ element at the end.
- -get(int index) \rightarrow Access element by index.
- -set(int index, T data) \rightarrow Update data at a specific index.
- -remove(int index) \rightarrow Remove element at index.
- $-size() \rightarrow Returns total number of elements.$
- -forEach() \rightarrow For manual iteration.

Where is it used?

- -UserService: To store & search registered users.
- -ExpenseService: To load and store all expens
- -FileUtil: To read file lines into a custom list.

2. MyHashMap<K, V>

Needed a key-value mapping structure to track:

- Total amount per category
- Count of expenses per category
- Custom "get or default" logic.

How does it work?

Array of linked lists (buckets) to handle collisions (Separate Chaining). Simple hash function based on key's hashcode.

Supports:

- -put(K key, V value) \rightarrow Insert or update key.
- $-get(K \text{ key}) \rightarrow \text{Retrieve value by key}.$
- -getOrDefault(K key, V defaultValue) → If key not found, return default.
- -containsKey(K key) \rightarrow Check key presence.
- -keySet() \rightarrow Get all keys as a String array.

Where is it used?

- -ReportService: For category-wise totals & counts.
- -Category Insights: To find most-used category.
- -Savings Goal Comparison: Stores user goals (optional).

Approach

- All expense and user data are persisted in .txt files.
- Passwords are hashed using a simple multiplication-based hashing algorithm.
- No Java Collections are used. All lists and maps are custom-built.
- Recurring expenses are marked with a flag and shown during monthly reports.
- Savings goals are stored per user and checked during report generation.
- Sorting is implemented using a custom bubble sort on MyLinkedList.



Sample Input/Output

User Flow Example:

1. Register

Username: Subhu05 Password: Subhu05

2. Login

Username: Subhu05 Password: Subhu05

```
3. Add Expense
Amount: 1500
Category: Food
Description: Lunch at Café
Date: 15-07-2025
Recurring: No

4. Generate Monthly Report
> Report saved as reports/monthly_Subhu05_07-2025.txt

5. View Category Insights
> Most Used Category: Food
> Average Spending: ₹1500.00

6. Set Savings Goal
> Goal set to ₹10000
```

Report File Example:

```
==== Monthly Expense Summary ====
User: Subhu05
Month: 07-2025

Category-wise Breakdown:
- Food: Rs.1500.0

Total Monthly Expense: Rs.1500.0

Highest Expense: Rs.1500.0 (Food)

Expense Details:
- Rs.1500.0 | Food | Lunch at Café | 15-07-2025

Savings Goal: Rs.10000.0

You are within your savings goal! Saved Rs. 8500.0
```

Bonus Features Implemented:

- Recurring Expenses
- Savings Goal Feature
- Custom Sorting (Amount/Date)

Conclusion:

This project simulates a real-world expense tracking application without using Java's built-in collection frameworks, ensuring a deep understanding of data structures and application architecture.