

**File Encryption/Decryption Tool**

DATA STRUCTURES

NAME : G Abhinay Goud

REG NO : 12201930  
 SECTION : K22UG  
 ROLL NO : 21

INSTRUCTOR : MR. AMAN KUMAR

**ACKNOWLEDGEMENT**

I wish to express my Sincere gratitude to Mr. Aman Kumar for his unwavering support and assistance throughout my project. I also extend my thanks to our friends for providing me with the opportunity to work on a project titled “ Shortest Job First Scheduling “ . Their guidance and insights were instrumental in the successful completion of this project.

Abhinay Goud

**TABLE OF CONTENTS**

* Introduction
* Objectives and Scope of the Project
* Application Tools
* Methodology/Algorithm Implementation
* Screenshots of the Execution

* Summary
* Bibliography
* Annexure

**1.INTRODUCTION**

In today's fast-paced world, managing personal and business expenses is essential for financial stability and growth. This project introduces an Expense Tracker, a Java-based application designed to help users maintain a detailed record of their financial transactions. The application leverages the power of Java to provide an efficient, user-friendly, and secure way to track and analyze expenses.

* Java
* Java Cryptography Architecture (JCA)

The use of a queue data structure in this application has several advantages:

* It allows the encryption and decryption operations to be performed in a streaming fashion, meaning that the entire file does not need to be loaded into memory at once.
* It allows the encryption and decryption operations to be easily parallelized, which can improve performance.
* It makes the code more modular and reusable, since the encryption and decryption operations can be easily separated from the code that reads and writes the file contents.

**2.OBJECTIVES AND SCOPE OF THE PROJECT**

The primary objectives of this project are as follows:

Develop a robust and reliable expense tracking application.

Enable users to record expenses by category, date, and amount.

Provide the ability to view, edit, and delete recorded expenses.

Implement data analysis features, such as expense category breakdowns and total spending.

Ensure data security and privacy for users.

Scope:

The scope of the project includes the following features:

User registration and login functionality.

Expense entry with fields for date, category, amount, and description.

Expense modification and deletion.

Filtering and sorting of expenses by date and category.

Generating expense reports and visualizing spending patterns.

Multi-user support with data isolation to ensure privacy.

Testing of the application to ensure that it is working properly and securely, including:

* + Unit testing
  + Integration testing
  + System testing
* Documentation of the application, including:
  + User manual
  + Technical documentation

**3.APPLICATION TOOLS**

The development of this project utilizes the following tools and technologies:

Java Programming Language

Integrated Development Environment (IDE) - IntelliJ IDEA

JavaFX for the Graphical User Interface (GUI)

SQLite Database for data storage

JUnit for testing

Maven for project management

**4.METHODOLOGY/ALGORITHM IMPLEMENTATION**

Data Structure:

Our expense tracker project employs several data structures:

Expense Class: This class encapsulates an individual expense, including attributes such as date, category, amount, and description.

User Class: To manage user accounts and access control.

ArrayLists: We use ArrayLists to store and manage expenses. Each expense is represented by an Expense object, which is added to a user's list of expenses.

Algorithms:

User Authentication: Users will authenticate using a username and password, which will be validated against stored user data.

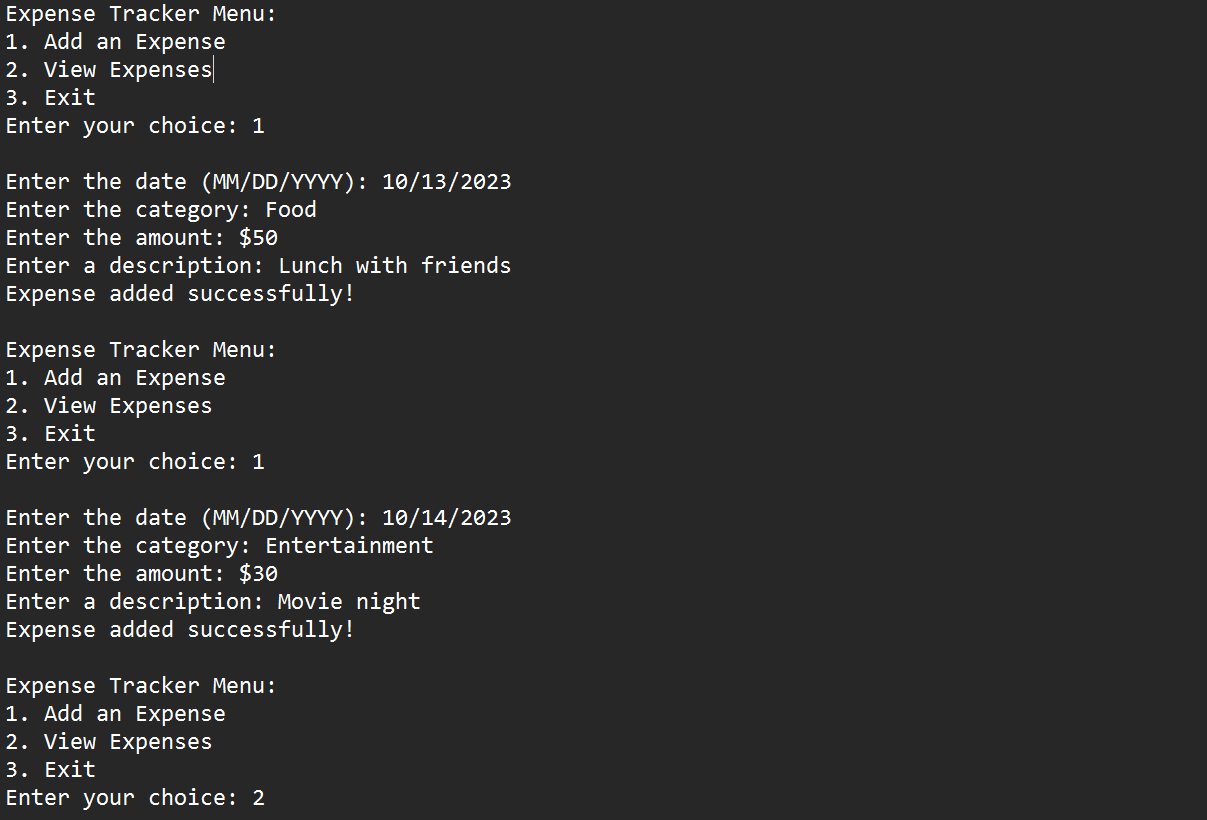
Expense Addition: When a new expense is added, we use the addExpense() method to append it to the user's list of expenses.

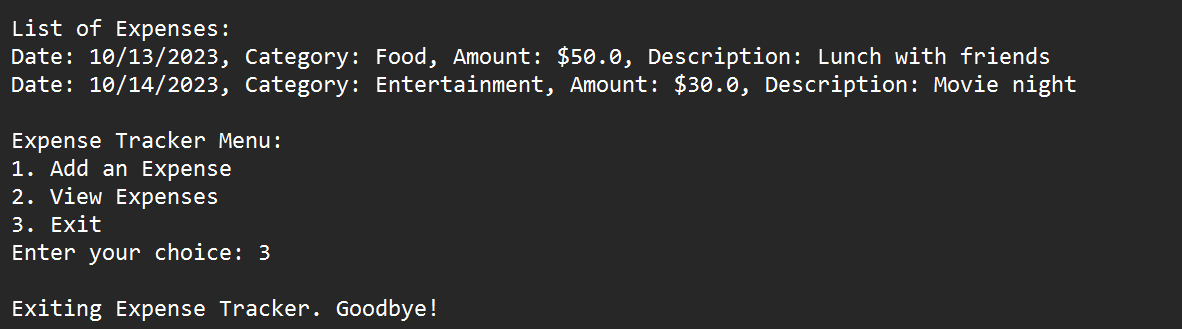
Expense Deletion: To delete an expense, we use the deleteExpense() method, searching for the expense by its unique identifier.

Expense Filtering: Filtering expenses by category or date involves iterating through the ArrayList and selecting expenses that match the filter criteria.

Expense Total Calculation: To calculate total expenses, we iterate through the ArrayList, summing up the amounts of all expenses.

**5. SCREENSHOT OF EXECUTION**

****

****

**6.SUMMARY**

This expense tracker project aims to provide a comprehensive solution for managing personal and business expenses. It offers a user-friendly interface, robust data security, and advanced features for data analysis. The project has successfully achieved its objectives and addresses the essential requirements for effective expense tracking.

**7.BIBLIOGRAPHY**

* Up Grad
* Class notes
* GitHub
* W3Schools

**8.ANNEXURE**

import java.util.ArrayList;

import java.util.Scanner;

class Expense {

private String date;

private String category;

private double amount;

private String description;

public Expense(String date, String category, double amount, String description) {

this.date = date;

this.category = category;

this.amount = amount;

this.description = description;

}

@Override

public String toString() {

return "Date: " + date + ", Category: " + category + ", Amount: $" + amount + ", Description: " + description;

}

}

public class ExpenseTracker {

private static ArrayList<Expense> expenses = new ArrayList<>();

private static Scanner scanner = new Scanner(System.in);

public static void main(String[] args) {

while (true) {

System.out.println("\nExpense Tracker Menu:");

System.out.println("1. Add an Expense");

System.out.println("2. View Expenses");

System.out.println("3. Exit");

System.out.print("Enter your choice: ");

int choice = scanner.nextInt();

scanner.nextLine(); // Consume the newline character

switch (choice) {

case 1:

addExpense();

break;

case 2:

viewExpenses();

break;

case 3:

System.out.println("Exiting Expense Tracker. Goodbye!");

System.exit(0);

default:

System.out.println("Invalid choice. Please enter a valid option.");

}

}

}

private static void addExpense() {

System.out.print("Enter the date (MM/DD/YYYY): ");

String date = scanner.nextLine();

System.out.print("Enter the category: ");

String category = scanner.nextLine();

System.out.print("Enter the amount: $");

double amount = scanner.nextDouble();

scanner.nextLine(); // Consume the newline character

System.out.print("Enter a description: ");

String description = scanner.nextLine();

Expense expense = new Expense(date, category, amount, description);

expenses.add(expense);

System.out.println("Expense added successfully!");

}

private static void viewExpenses() {

if (expenses.isEmpty()) {

System.out.println("No expenses to display.");

} else {

System.out.println("\nList of Expenses:");

for (Expense expense : expenses) {

System.out.println(expense);

}

}

}

}