RAJALAKSHMI
ENGINEERING COLLEGE
An AUTONOMOUS Institution
Affiliated to ANNA UNIVERSITY, Chennai

MOVIE RESERVATION SYSTEM

CS23333 – Object Oriented Programming Using JAVA Project Report

Submitted by

SURYA M.V -231501166

SUPRAJA. R -231501164

Of

BACHELOR OF TECHNOLOGY

In

Artificial Intelligence And Machine Learning

RAJALAKSHMI ENGINEERING COLLEGE, THANDALAM
(An Autonomous Institution)

RAJALAKSHMI ENGINEERING COLLEGE

BONAFIDE CERTIFICATE

Certified that this project titled "Movie Reservation System" is the bonafide work of "SURYA M.V **(231501166)** SUPRAJA R **(231501164)**" who carried out the project work under my supervision.

SIGNATURE
Mr.Devendar Rao
COURSE INCHARGE

Rajalakshrni Engineering College
Rajalakshrni Nagar, Thandalarn
Chennai- 602105

This project is submitted for IT19341 – Introduction to Oops and Java held on

_____

INTERNAL EXAMINAR **EXTERNAL EXAMINAR**

TABLE OF CONTENTS

1.1Abstract:

The primary objective of the Expense Tracker System is to provide users with a reliable and efficient platform to manage and monitor their daily, weekly, or monthly expenses. This system leverages Java's capabilities and integrates with a MySQL database to ensure data consistency and security. The intuitive interface empowers users to categorize expenses, generate reports, and analyze spending patterns, making financial planning accessible and streamlined.

.

1.2 Introduction:

The Expense Tracker System is a modern solution aimed at transforming how users manage their finances. Traditional methods like manual bookkeeping are often cumbersome and error-prone. This project introduces automation to enhance financial management processes, offering tools for budgeting, categorization, and visualization. By integrating technology into personal finance, the system simplifies financial oversight, promoting better decision-making and financial health.

experience, marking a notable departure from traditional methods.

1.3 Purpose:

The Expense Tracker aims to:Simplify financial management for users by tracking income and expenditures.Provide insights into spending patterns through detailed reports and charts.Allow categorization of expenses for better financial planning.Offer secure and user-friendly tools to set and achieve financial goals. The purpose of this project is to create an efficient and user-friendly movie reservation system that benefits both cinema-goers and cinema administrators. The system aims to:

1.4 Scope of the Project:

The Expense Tracker System interacts seamlessly with users to log, categorize, and analyze financial transactions. Developed using Java and MySQL, the system ensures robust data handling and security. It provides real-time insights and budgeting features, reducing the effort required for manual financial tracking. The project prioritizes user-friendliness, security, and scalability to cater to a wide range of financial management needs.

1.5 Software Requirement Specification:

Introduction:

The Movie Reservation System is designed to manage all aspects of movie ticket reservations, including movie details, showtimes, and seat allocations. It serves as an automated alternative to traditional manual reservation processes, enhancing efficiency in cinema management.

Document Purpose:

This SRS document outlines the software requirements for the Movie Reservation System, covering design decisions, architectural design, and detailed design necessary for successful implementation. It offers insight into the system's structure and provides information crucial for ongoing software support.

Product Scope:

The Movie Reservation System is developed for widespread use, aiming to replace outdated paper-based reservation systems. It streamlines the movie reservation process, offering a comprehensive solution for cinema administrations. The system is versatile, providing flexibility through mechanisms for editing movie information and optimizing the overall reservation experience.

References and Acknowledgement:

[1] **https://www.javatpoint.com/java-awt**
[2] **https://www.javatpoint.com/java-swing**

Overall Description:

The Expense Tracker System is an innovative tool designed to simplify financial management by automating the tracking of income and expenditures. It provides users with a structured and organized approach to monitor their spending habits and manage budgets effectively. This system uses a client-server architecture, with a user-friendly Java-based frontend and a MySQL database backend for robust data handling

---

Product Perspective:

Utilizing a client/server architecture, the system is designed to be compatible with the Microsoft Windows Operating System. The front end is developed using Java AWT and SWING, while the backend leverages the MySQL server for efficient data management.

User and Characteristics:

Qualification: Users should have at least basic educational qualifications, such as matriculation, and be comfortable with English.
Experience: Familiarity with the university registration process is advantageous.
Technical Experience: Users are expected to have elementary knowledge of computers for optimal system interaction.

Operating Environment:

Hardware Requirements:
- Processor: Any Processor over i3
- Operating System: Windows 8, 10, 11
- Processor Speed: 2.0 GHz
- RAM: 4GB
- Hard Disk: 500GB

Software Requirements:
- Database: MySQL
- Frontend: Java (SWING, AWT)
- Technology: Java (JDBC)

Constraints:
- System access limited to administrators.
- Delete operation restricted to administrators without additional checks for simplicity.
- Administrators must exercise caution during deletion to maintain data consistency.

Assumptions and Dependencies:
- System administrators create and confidentially communicate login IDs and passwords to users.

Specific Requirements:

User Interface:

The Movie Reservation System provides user-friendly, menu-driven interfaces for:

a) Admin Register: Registering new administrators.
b) Admin Login: Logging in existing administrators.
c) Add Movie: Storing new movie details.
d) View Movie: Viewing and updating existing movie information.
e) Delete Movie: Deleting existing movies.
f) Add Reservation: Creating new reservations for users.
g) Update Reservation: Viewing and modifying existing reservations.
h) Remove Admin: Deleting specific administrator accounts.

Hardware Interface:
- Screen resolution of at least 640 x 480 or above.
- Compatible with any version of Windows 8, 10, 11.

Software Interface:

a) MS-Windows Operating System
b) Java AWT and SWING for designing the front end
c) MySQL for the backend
d) Platform: Java Language
e) Integrated Development Environment (IDE): NetBeans

Functional Requirements:

1. Log in Module (LM):

- Users (admins) access the Login Module.
- LM supports user login with a username and password.
- Passwords are masked for security.
- Successful login verification by the database administrator is required for access.

2. Registered Users Module (RUM):

- After successful login, users (admins) can navigate through the application.
- Users can view detailed information about movies, showtimes, and reservations.
- Users can update and maintain movie details, including modifying showtimes and seat allocations.

3. Administrator Module (AM)

- Upon successful login, the system displays administrative functions.
- Functions include adding and updating movie details.
- The "Add" function allows administrators to input new movie details and remove unused entries.
- The "Update" function enables administrators to modify existing movie details in the database.
- All add, update, or delete requests trigger the AM module to communicate with the Server Module (SM) for necessary database changes.

4. Server Module (SM):

- SM acts as an intermediary between various modules and the database (DB).
- Receives requests from different modules and formats pages for display.
- Validates and executes requests received from other modules.
- Handles communication with the database, ensuring data consistency and integrity, especially regarding movie details, showtimes, and reservations.

Non-functional Requirements:

Performance:
- The system must handle real-time reservation requests efficiently, ensuring a response time of less than 2 seconds for seat selection and confirmation.
- Safety-critical failures, such as payment processing errors, must be addressed instantly to ensure a smooth user experience.

Reliability:
- The system is safety-critical; in case of abnormal operation or downtime, immediate measures should be taken to resolve the issue and restore normal functionality.

Availability:
- Under normal operating conditions, user requests for movie reservations, including seat selection and payment, should be processed within 2 seconds to maintain a seamless booking experience.
- Immediate feedback on reservation status and confirmation should be communicated to users to enhance the overall booking process.

Security:
- A robust security mechanism must be in place on the server side to prevent unauthorized access, safeguard user payment information, and ensure the integrity of the reservation system.
- User privacy, including personal details, must be securely stored and managed to maintain confidentiality.

Maintainability:
- Design documents outlining software and database maintenance procedures must be available to facilitate regular updates and modifications to the movie reservation system.
- Administrative access should be provided for proper maintenance at both the front end and back end, ensuring the system's long-term functionality and adaptability. end

3. Module description:

Admin:

Register:

Admin can register with the username and password for the registration.

Login:

Admin can log in with their username and password.

After Login:

Add Movie:

In this section, admin can add details about new movies, including the title, genre, release date, and other relevant information.

View Movie:

Admin can view and update movie details such as the title, genre, and release date.

Delete Movie:

Admin can delete movie details, removing movies from the system.

Add Reservation:

Admin can add reservation details, including the movie, date, time, and number of seats.

Update Reservation:

Admin can update reservation details, such as changing the date, time, or the number of reserved seats.

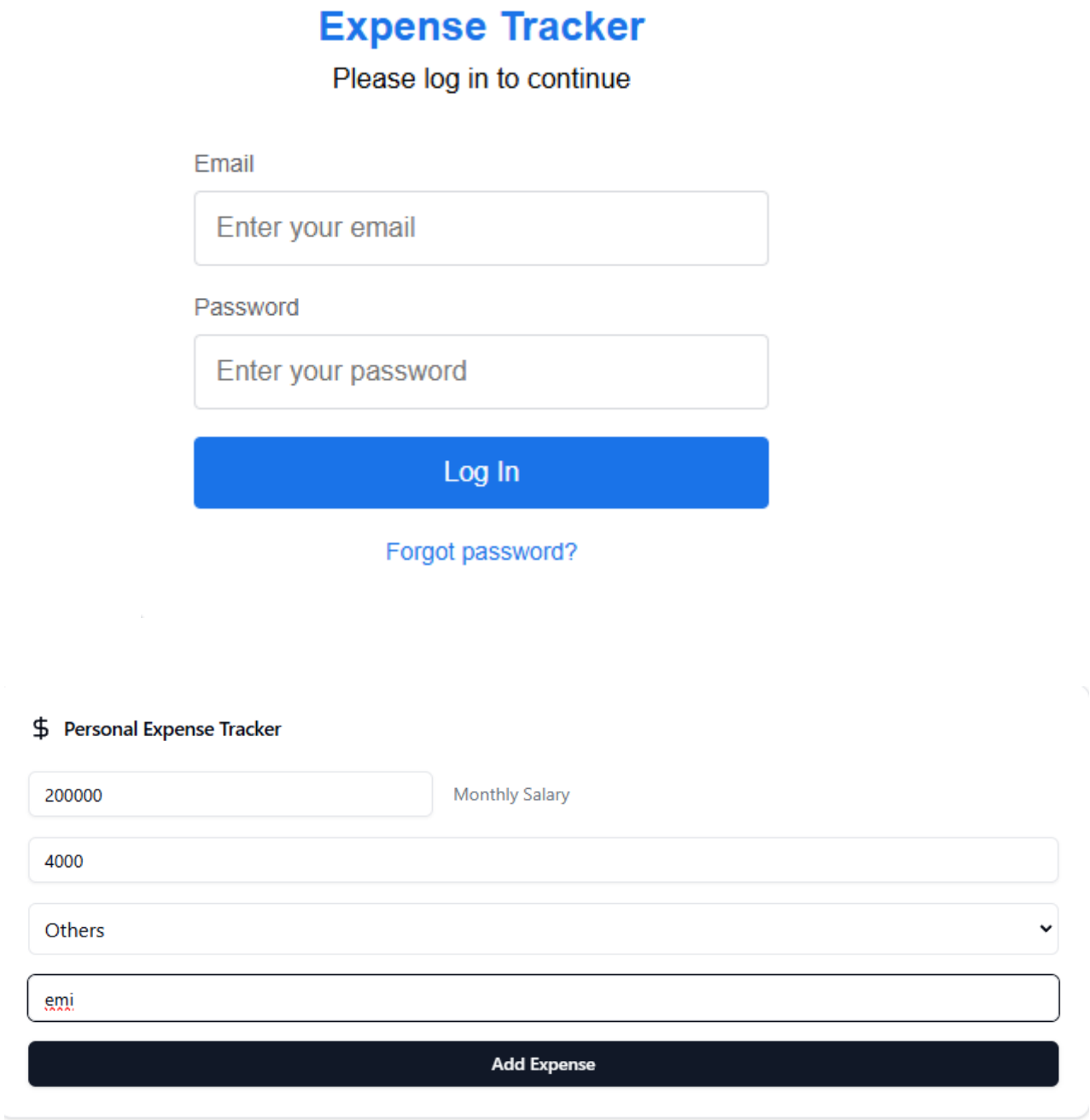
Remove Admin:

In this section, the system can allow for the deletion of admin details if needed.

4.Implementation:

4.1 Design:

Home Page:



Financial Summary

Monthly Salary
$200000.00

Total Expenses
$50000.00

Remaining Balance
$150000.00

You can spend $30000.00 per day for the remaining 5 days of this month.

Expense History

Date	Category	Amount	Description
11/25/2024	Food	$1000.00	treat
11/25/2024	Transportation	$9000.00	tour
11/25/2024	Housing	$25000.00	rent
11/25/2024	Entertainment	$5000.00	gaming center
11/25/2024	Shopping	$10000.00	house grocery

4.3 CODE:

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.Scanner;

public class ExpenseTracker {
  private static final String DB_URL = "jdbc:sqlite:expenses.db";
  private static final String CREATE_USERS_TABLE = "CREATE TABLE
IF NOT EXISTS users (id INTEGER PRIMARY KEY
AUTOINCREMENT,username TEXT UNIQUE NOT NULL,password
TEXT NOT NULL)";
  private static final String CREATE_EXPENSES_TABLE = "CREATE
TABLE IF NOT EXISTS expenses (id INTEGER PRIMARY KEY
AUTOINCREMENT,user_id INTEGER NOT NULL,category TEXT NOT
NULL,amount DECIMAL(10,2) NOT NULL,date TEXT NOT
NULL,FOREIGN KEY (user_id) REFERENCES users(id))";

  public ExpenseTracker() {
  }

  public static void main(String[] var0) {
    try {
      Class.forName("org.sqlite.JDBC");
      Connection var1 =
DriverManager.getConnection("jdbc:sqlite:expenses.db");
      Statement var2 = var1.createStatement();
```

```
var2.execute("CREATE TABLE IF NOT EXISTS users (id INTEGER
PRIMARY KEY AUTOINCREMENT,username TEXT UNIQUE NOT
NULL,password TEXT NOT NULL)");
        var2.execute("CREATE TABLE IF NOT EXISTS expenses (id
INTEGER PRIMARY KEY AUTOINCREMENT,user_id INTEGER NOT
NULL,category TEXT NOT NULL,amount DECIMAL(10,2) NOT
NULL,date TEXT NOT NULL,FOREIGN KEY (user_id) REFERENCES
users(id))");
        Scanner var3 = new Scanner(System.in);
        boolean var4 = false;
        int var5 = -1;

        while(true) {
          int var6;
          while(var4) {
            System.out.println("1. Log Expense");
            System.out.println("2. View Expenses");
            System.out.println("3. Set Budget");
            System.out.println("4. View Analytics");
            System.out.println("5. Logout");
            System.out.print("Enter your choice (1-5): ");
            var6 = var3.nextInt();
            var3.nextLine();
            switch (var6) {
              case 1:
                logExpense(var1, var5, var3);
                break;
              case 2:
```

```
viewExpenses(var1, var5);
                    break;
                case 3:
                    setBudget(var1, var5, var3);
                    break;
                case 4:
                    viewAnalytics(var1, var5);
                    break;
                case 5:
                    var4 = false;
                    var5 = -1;
                    System.out.println("Logged out successfully.");
                    break;
                default:
                    System.out.println("Invalid choice. Please try again.");
            }
        }

        System.out.println("1. Sign Up");
        System.out.println("2. Log In");
        System.out.print("Enter your choice (1 or 2): ");
        var6 = var3.nextInt();
        String var7;
        String var8;
        if (var6 == 1)
    System.out.print("Enter username: ");
            var7 = var3.nextLine();
            System.out.print("Enter password: ");
            var8 = var3.nextLine();

BorderLayout.CENTER);
```

```
 registerUser(var1, var7, var8);
            } else if (var6 == 2) {
               System.out.print("Enter username: ");
               var7 = var3.nextLine();
               System.out.print("Enter password: ");
               var8 = var3.nextLine();
               var5 = loginUser(var1, var7, var8);
               if (var5 != -1) {
                  var4 = true;
                  System.out.println("Logged in successfully!");
               }
            } else {
               System.out.println("Invalid choice. Please try again.");
            }
         }
      } catch (SQLException | ClassNotFoundException var9) {
         var9.printStackTrace();
      }
   }

   private static void registerUser(Connection var0, String var1, String
var2) throws SQLException {
      String var3 = "INSERT INTO users (username, password) VALUES
(?, ?)";
      PreparedStatement var4 = var0.prepareStatement(var3);
      var4.setString(1, var1);
      var4.setString(2, var2);
      var4.executeUpdate();
      System.out.println("User registered successfully.");
   }
```

```
private static int loginUser(Connection var0, String var1, String var2) throws
SQLException {
    String var3 = "SELECT id FROM users WHERE username = ? AND
password = ?";
    PreparedStatement var4 = var0.prepareStatement(var3);
    var4.setString(1, var1);
    var4.setString(2, var2);
    ResultSet var5 = var4.executeQuery();
    if (var5.next()) {
      return var5.getInt("id");
    } else {
      System.out.println("Invalid username or password. Please try again.");
      return -1;
    }
  }
  private static void logExpense(Connection var0, int var1, Scanner var2)
throws SQLException {
    System.out.print("Enter expense category: ");
    String var3 = var2.nextLine();
    System.out.print("Enter expense amount: ");
    double var4 = var2.nextDouble();
    var2.nextLine();
    System.out.print("Enter expense date (yyyy-MM-dd): ");
    String var6 = var2.nextLine();
    String var7 = "INSERT INTO expenses (user_id, category, amount, date)
VALUES (?, ?, ?, ?)";
    PreparedStatement var8 = var0.prepareStatement(var7);
    var8.setInt(1, var1);
    var8.setString(2, var3);
    var8.setDouble(3, var4);
    var8.setString(4, var6);
    var8.executeUpdate();
    System.out.println("Expense logged successfully.");
  }
```

```
private static void viewExpenses(Connection var0, int var1) throws
SQLException {
    String var2 = "SELECT category, amount, date FROM expenses WHERE
user_id = ?";
    PreparedStatement var3 = var0.prepareStatement(var2);
    var3.setInt(1, var1);
    ResultSet var4 = var3.executeQuery();
    System.out.println("Your expenses:");

    while(var4.next()) {
      String var5 = var4.getString("category");
      double var6 = var4.getDouble("amount");
      String var8 = var4.getString("date");
      System.out.printf("%s - $%.2f on %s%n", var5, var6, var8);
    }

  }

  private static void setBudget(Connection var0, int var1, Scanner var2) throws
 SQLException {
    System.out.println("Budget setting feature not yet implemented.");
  }

  private static void viewAnalytics(Connection var0, int var1) throws
SQLException {
    System.out.println("Analytics feature not yet implemented.");
  }
}
```

Conclusion:

The Expense Tracker System project, executed under

The Expense Tracker System is a user-centric tool designed for intuitive and efficient financial management. With a robust backend and dynamic features, it simplifies personal budgeting and financial analysis. The project addresses current requirements while being adaptable for future enhancements.
requirements while allowing for future adaptability and enhancements.
Promotes Financial Discipline: By providing a clear overview of expenses, the tracker helps users identify unnecessary spending and promotes better budgeting habits.Customizable and Scalable: The expense tracker can be expanded with additional features, such as advanced analytics, saving goals, or reminders for bill payments, to better suit user needs.User-Friendly Design: The tracker is designed with simplicity in mind, ensuring ease of use for users of all ages, regardless of their technical expertise.Data Insights for Decision-Making: With categorized data and monthly summaries, users can make informed financial decisions and plan for long-term goals effectively.Real-Time Expense Monitoring: The tracker provides up-to-date tracking of expenses, helping users maintain a real-time understanding of their financial situation.Encourages Savings: By calculating remaining balances and allowing users to visualize how much they can still spend, the tracker naturally encourages saving habits.Supports Financial Goals: Users can integrate financial objectives, such as saving for a vacation or a large purchase, and monitor their progress.Adaptable for Various Income Levels: The expense tracker can be used by individuals with different income levels, making it versatile and inclusive.Encourages Accountability: Users can hold themselves accountable for their spending habits, reducing the likelihood of overspending.Educational Tool: For users unfamiliar with budgeting, the tracker serves as a learning tool to develop personal finance management skills.Environmentally Friendly: By eliminating the need for paper-based expense recording, the tracker supports eco-friendly practices.Potential for Integration: Future iterations of the tracker can integrate with bank accounts or payment apps to automate the expense logging process.Time Efficiency: Instead of manually calculating expenses, the tracker automates the process, saving time and effort.Accessibility: The application can be accessed from multiple devices, ensuring users can manage their finances on the go.Data Security: Implementing secure storage methods ensures that users' sensitive financial data is protected, building trust in the application.

Reference links:

[1] https://www.javatpoint.com/java-awt
[2] https://www.javatpoint.com/java-swing