Project Based Evaluation

Data Structure

Project ID-(JPG09-15)

Project Report

Semester-IV (Batch-2023)

Movie Ticket Booking System

Supervised By:

Dr. Rajender kumar

Submitted By:

Karanbir , 2310990708

Kartik Jindal , 2310990709

Nahar , 2310990744

Pranav Goyal , 2310991426

Department of Computer Science and Engineering
Chitkara University Institute of Engineering & Technology , Chitkara University Punjab

INDEX

Abstract :-

This project presents a Movie Ticket Booking System with a Graphical User Interface (GUI) developed in Java using Swing. The application simplifies the ticket booking process through an intuitive interface that enables users to book and cancel movie tickets, search tickets by ID, and view booking history. The system utilizes both Array List and LinkedList data structures to efficiently manage and organize ticket information and user history—Array List is used for faster random access and updates, while LinkedList is used where frequent insertions or deletions occur, such as in maintaining booking records. The system ensures persistent storage of booked seats and coin balances between sessions and includes safeguards against booking already reserved seats. The GUI provides real-time feedback for user actions, creating a smooth and engaging experience.

Key Features :-

1. Movie and Ticket Management:

- Each movie entry stores details such as title, total seats, and available seats.
- Users can view movies, check seat availability, and make bookings through an intuitive GUI.
- Ticket search by booking ID allows users to quickly locate specific bookings.

2. Dynamic Data Handling with Hybrid Structures:

- Utilizes both Array List and LinkedList to balance performance and flexibility:
- Array List is used for efficient index-based access and updates to movie lists and bookings.
- LinkedList handles dynamic user history and operations involving frequent insertions/removals.
- Ensures seamless and real-time interaction with booking data

.3. Booking Workflow and Validation:

- Prevents double booking by validating seat availability before confirming.
- Supports cancelling bookings and refunds by updating both GUI and underlying data models.
- Each booking generates a unique ID for traceability and retrieval.

4. Graphical User Interface (GUI):

Built using Java Swing with organized layout elements like buttons, combo boxes, and labels.

Includes sections for:

- Selecting movies and seats
- Booking and cancelling tickets
- Viewing booking history

Uses responsive design principles to provide smooth feedback (e.g., button animations, alerts).

Persistent Storage and Session Continuity:

Booked seats and coin balance are saved and reloaded across sessions using file-based storage.

Guarantees user data is not lost between application runs.

5.User-Friendly Design:

- Designed to cater to general users with no technical background.
- Admin features include movie list editing and reset functionality for booked seats.

1. Introduction :-

In the modern era of digital entertainment, movie ticket booking has transitioned from manual counter-based systems to fast, user-friendly digital platforms. However, many existing systems still lack personalization, responsiveness, and intuitive interfaces, leading to inefficient booking experiences and poor customer satisfaction. To address these gaps, this project presents a Movie Ticket Booking System developed using Java Swing, offering a desktop-based graphical user interface (GUI) for seamless movie ticket booking, seat management, and booking history tracking.

1.1 Objectives :

The primary objectives of this Movie Ticket Booking System are as follows:

1.Streamlined Management of Movie Bookings:

Automate the movie ticket booking process through a centralized GUI platform.

Display real-time seat availability, enable quick booking/cancellation, and prevent double bookings via validation checks.

2. User-Friendly Interface

Graphical User Interface (GUI) for Accessibility

Offer an intuitive and organized interface using Java Swing components like combo boxes, labels, tables, and buttons.

Facilitate seamless user interaction without requiring technical expertise.

3. Efficient Data Handling Using Data Structures

Use Array List for structured and indexed storage of movies and booking records.

Use LinkedList for dynamically handling booking history and user actions.

These data structures ensure responsive performance and easy modifications as the dataset grows.

4. Persistent Data Storage

Maintain consistency of booking records and user coin balances across sessions through file-based storage mechanism. Allow retrieval of past bookings using unique ticket IDs.

5. Real-Time Feedback and Confirmation

Provide immediate confirmation of successful bookings or cancellations.

Display essential booking details like movie name, seat number, showtime, and coins used/refunded.

1.2 Significance :

The Movie Booking GUI System is more than a simple desktop booking tool—it demonstrates a scalable and user-friendly solution to modern cinema management. Its key significance lies in the following aspects:

1. Enhanced Operational Efficiency

Minimizes manual intervention by automating seat selection, booking validation, and history tracking.

Enables smoother ticketing operations, reducing queue time and human error.

2. Improved Data Accuracy and Integrity

Accurate and reliable booking validation prevents overbooking or loss of booking records.

Persistent storage ensures data continuity across sessions

3. Real-Time Booking and Seat Management

The system enables users to view real-time seat availability and book tickets instantly. This reduces booking conflicts and enhances user satisfaction by offering a seamless and responsive experience.

4. User Empowerment

The clean, responsive GUI enhances usability for non-technical users.

Features like search-by-ID and visible seat counts empower users with better control over their bookings.

5. Future-Proofing Cinema Management

The modular design and use of Java collections allow easy future expansions.

Additional features like admin movie management, multi-the support, or payment integration can be introduced without major structural changes.

2. Problem Definition and Requirements :-

2.1 Problem Definition :

In the digital age, cinema enthusiasts expect fast, reliable, and user-friendly platforms to book movie tickets. Manual ticketing systems and outdated software often fail to meet these expectations, leading to frustration among users and inefficiencies for administrators. This gap is particularly evident in local or offline cinema setups that do not utilize modern ticketing software. The Movie Booking GUI system addresses these challenges by offering a desktop-based, GUI-driven movie booking platform built in Java, designed to improve the overall user and administrative experience.

Inefficiency in Booking Management:
Manual ticketing and record-keeping are labour- intensive, time-consuming, and prone to human error. This can result in double bookings, lost reservations, and frustrated customers, ultimately affecting the reputation and revenue of threat.

Limited Accessibility and Convenience:
Traditional systems do not offer customers the flexibility to view available shows or book tickets anytime, anywhere. This lack of accessibility hinders customer satisfaction and limits the reach of cinema services.

Real-Time Data Limitations:
Without a centralized digital system, managing real-time updates on seat availability, show changes, or cancellations becomes difficult. This can lead to overbooking, miscommunication, and disrupted operations.

Lack of Analytical Tools:
The absence of integrated analytics makes it difficult for cinema owners or managers to assess booking trends, identify popular movies, or understand customer preferences. This limits opportunities for strategic planning and business growth.

Scalability Challenges:

As threat expand their operations—adding more screens, introducing new branches, or increasing showtimes—traditional systems become inadequate. They often fail to handle larger volumes of bookings or adapt to more complex scheduling needs.

2.2 Requirements :

The development of the **Movie Ticket Booking System** involves a clear set of functional and non-functional requirements to ensure the platform's effectiveness, usability, and scalability.

Functional Requirements :

1. **User Authentication:**
   The system must provide secure login functionality to ensure that only authorized users (e.g., admins, staff, or customers) can access specific functionalities.

2. **Movie and Show Management:**
   - Admins should be able to add, update, or delete movie information and showtimes.
   - Each show must include details such as movie name, date, time, screen, and available seats.

3. **Ticket Booking:**
   - Users must be able to browse available movies and showtimes.
   - Users should be able to select a show, choose seats, and confirm the booking.

4. **Ticket Cancellation:**
   - Users should be able to cancel their booked tickets.
   - The system must automatically update seat availability upon cancellation.

5. **Booking History and Status:**
   - Customers should have access to their booking history.
   - Admins should be able to view all bookings, filter by movie or date, and monitor booking trends.

6. **Search and Filter Options:**
   - Users should be able to search for movies based on name, genre, date, or language.

7. **Error Handling:**
    - The system must handle invalid inputs and unavailable bookings gracefully, with appropriate error messages and prompts.
8. **Seat Map Display:**
    - A visual seat map should be presented to users during booking to select available seats interactively.

Non-Functional Requirements

1. **Usability:**

   The interface should be user-friendly and intuitive, allowing users of all technical levels to navigate and use the system without difficulty**.**
2. **Performance:**

   The system should respond quickly to booking actions and handle multiple simultaneous users without lag or performance degradation.
3. **Scalability:**

   The architecture should support adding more theaters, screens, or advanced features (e.g., loyalty programs, mobile app integration) in the future with minimal changes.
4. **Security:**

   Sensitive information, such as user credentials and booking details, must be securely stored using encryption and secure authentication protocols.
5. **Reliability:**

   The system should have minimal downtime and must preserve data integrity in case of unexpected shutdowns or network issues.
6. **Maintainability:**

   The codebase and system architecture should follow modular design principles, allowing easy updates, debugging, and feature enhancements without disrupting the core functionality.

3. Methodology for the Smart Student Portal :-

The development of the **Movie Ticket Booking System** follows an **Agile software development methodology**, which promotes iterative progress, stakeholder collaboration, and the flexibility to adapt to changes. This approach ensures continuous improvement and efficient delivery of a functional, user-centric booking platform.

1. Requirements Gathering and Analysis

- Stakeholder Engagement:
  Identify and involve key stakeholders, such as cinema administrators, ticket agents, and end-users, to gather requirements and understand expectations.
- Data Collection:
  Conduct interviews and surveys with cinema operators and regular moviegoers to collect requirements about features such as seat selection, payment options, and cancellation policies.

2. System Design:

Architectural Design:

Develop a modular system architecture including the following components:

- **User Entity:** Handles user registration, login, and authentication.
- **Movie & Show Management Module:** Admin module for managing movie listings and showtimes.
- **Booking Engine:** Manages seat selection, booking, and ticket cancellation.
- **Mock Payment Gateway:** Simulates secure payment processing.
- **User Interface (UI):** GUI for customers and administrators.

Database Design:

Design a relational database schema to manage users, movies, shows, seats, and booking records. Ensure proper normalization for efficient querying and data integrity.

UI/UX Design:

Create wireframes and mock ups for both the user-facing and admin-facing interfaces. Focus on a clean, intuitive layout for ease of navigation and booking flow.

3. Implementation:

Development Environment Setup:

Set up the necessary tools, programming languages (e.g., Java or Python), frameworks (e.g., Java Swing or web-based technologies), and database systems.

Iterative Development (Sprints):

Implement core features in small, manageable cycles:

- Sprint 1: User registration and login.
- Sprint 2: Movie/show management by admin.
- Sprint 3: Seat selection and booking.
- Sprint 4: Booking history and cancellation.

Code Reviews:

Perform regular peer reviews to maintain code quality, ensure modularity, and adhere to design standards.

4. Testing:

Unit Testing:

Validate individual modules such as booking logic, seat availability updates, and payment processing.

Integration Testing:

Ensure smooth communication between modules (e.g., UI and database, payment and booking engine).

User Acceptance Testing (UAT):

Invite stakeholders to test the system and provide feedback on usability, navigation, and overall experience

.5. Deployment:

Outline a clear strategy for deploying the system in a live environment, which may include server setup or local installation.

- **User Training:**
  Provide manuals or demo sessions to admins and ticket agents to ensure they are comfortable with using the system.
- **Go-Live:**
  Launch the system, monitor real-time performance, and address any critical issues that arise post-deployment.

6. Maintenance and Support:

Ongoing Support:

Set up a helpdesk or ticketing system for handling user queries and technical issues.

Periodic Updates:

Plan version updates to introduce new features (e.g., loyalty points, discounts), fix bugs, and enhance performance.

Feedback Integration:

Maintain a continuous feedback loop with users to refine and optimize the system in future iterations.

4. Results :-

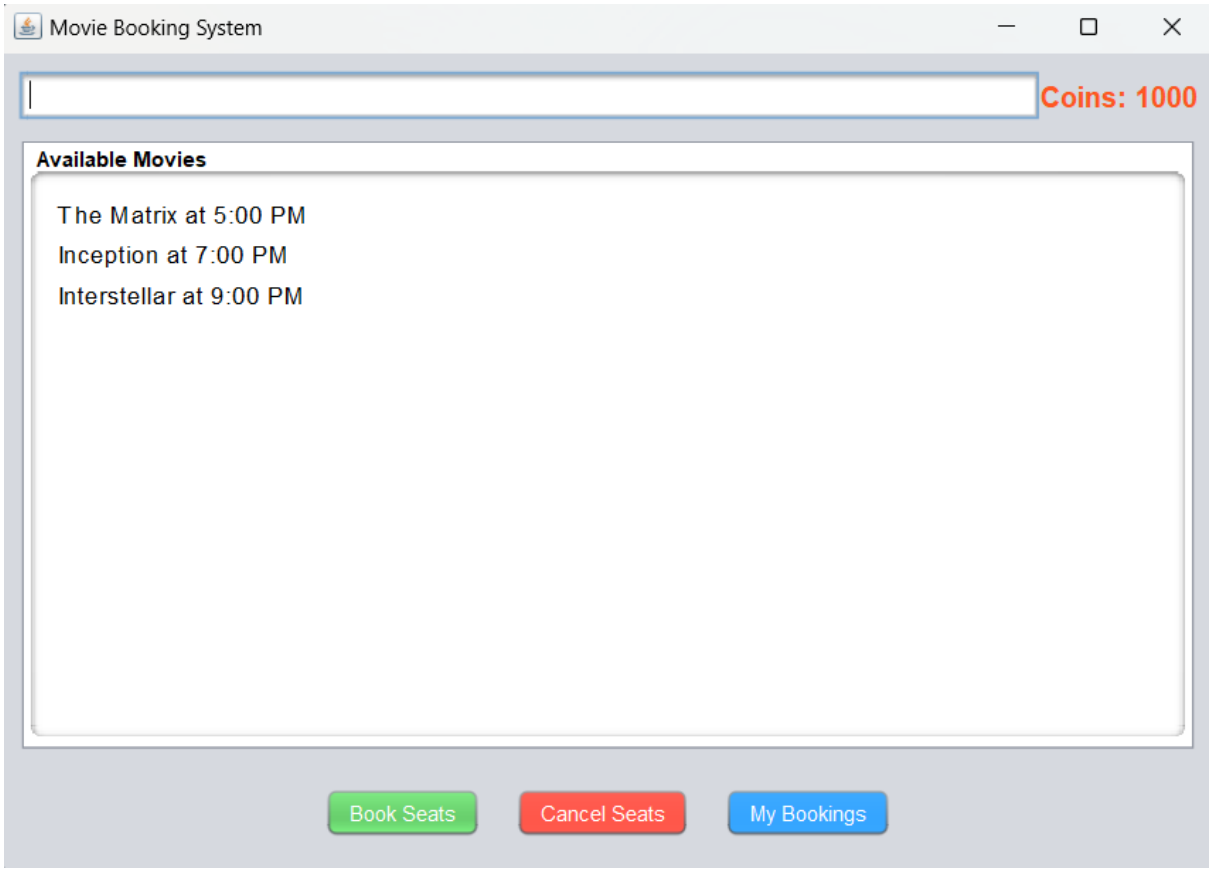

Figure 1

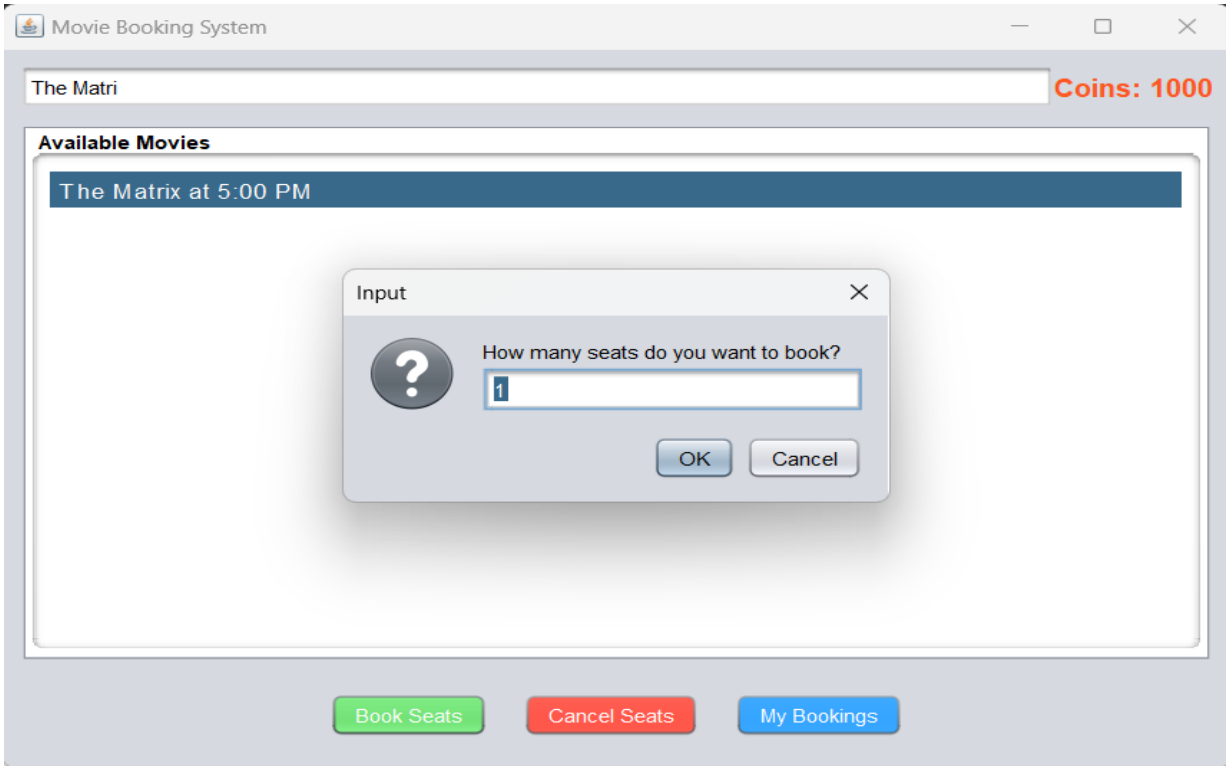

Figure 2

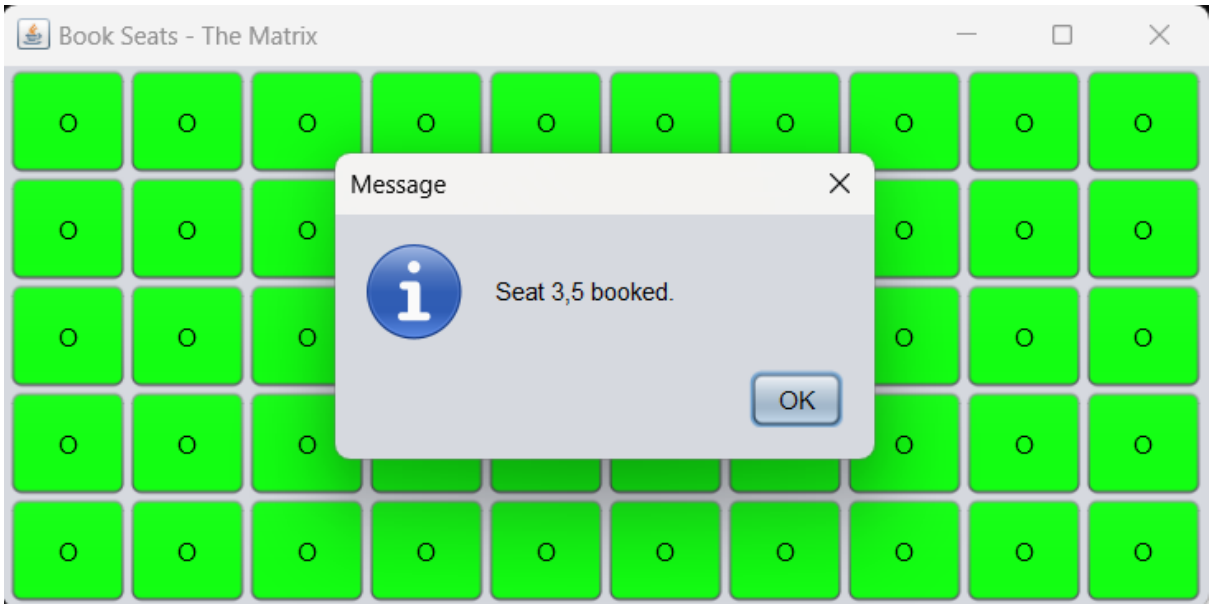

Figure 3

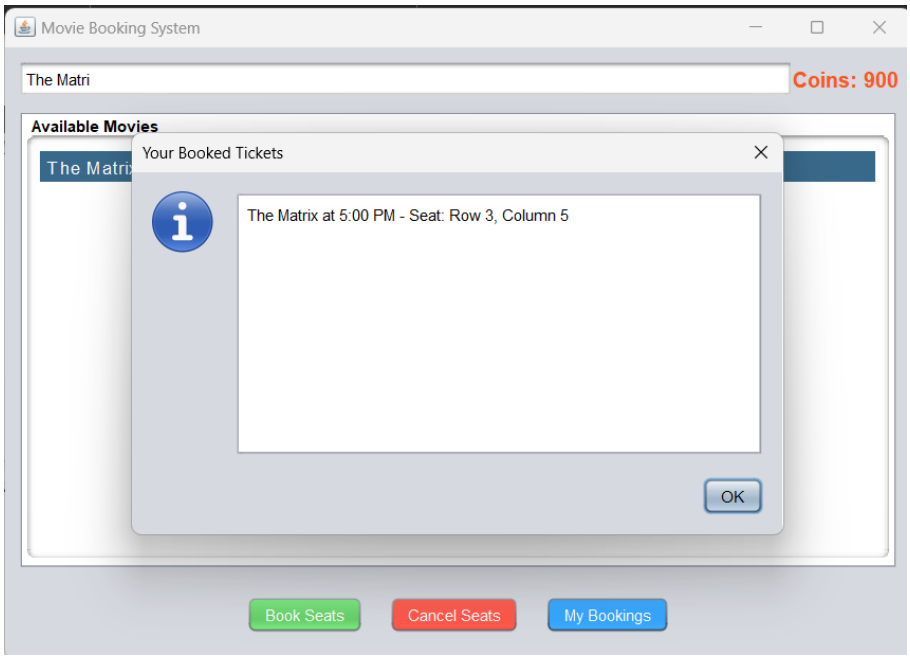

Figure 4

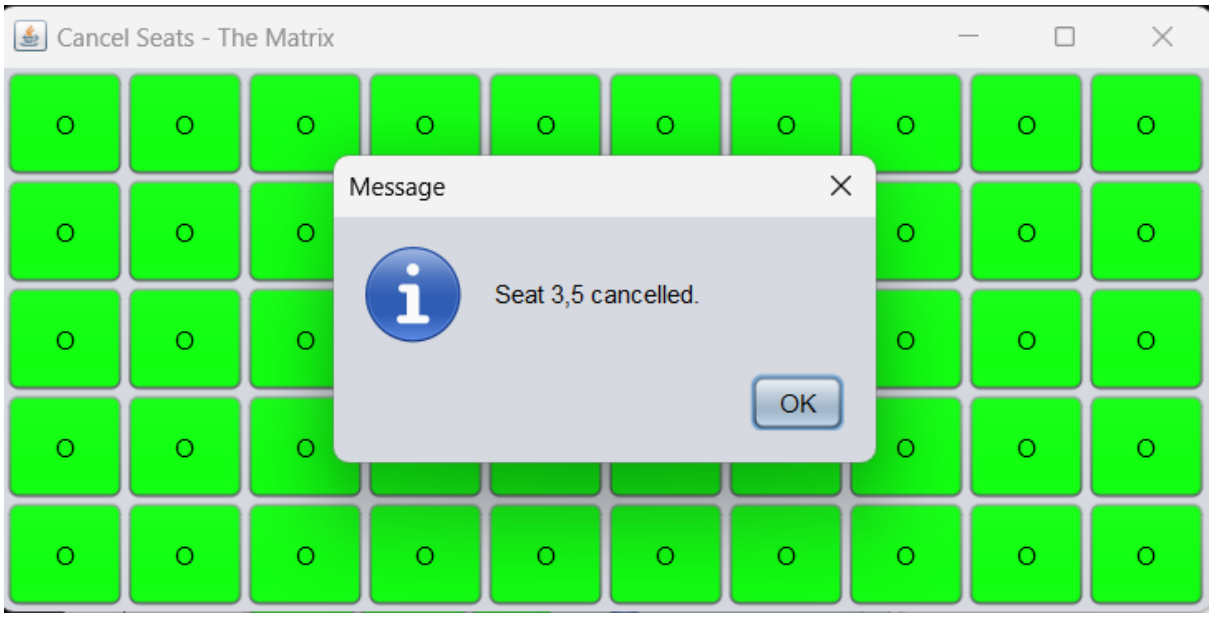

Figure 5

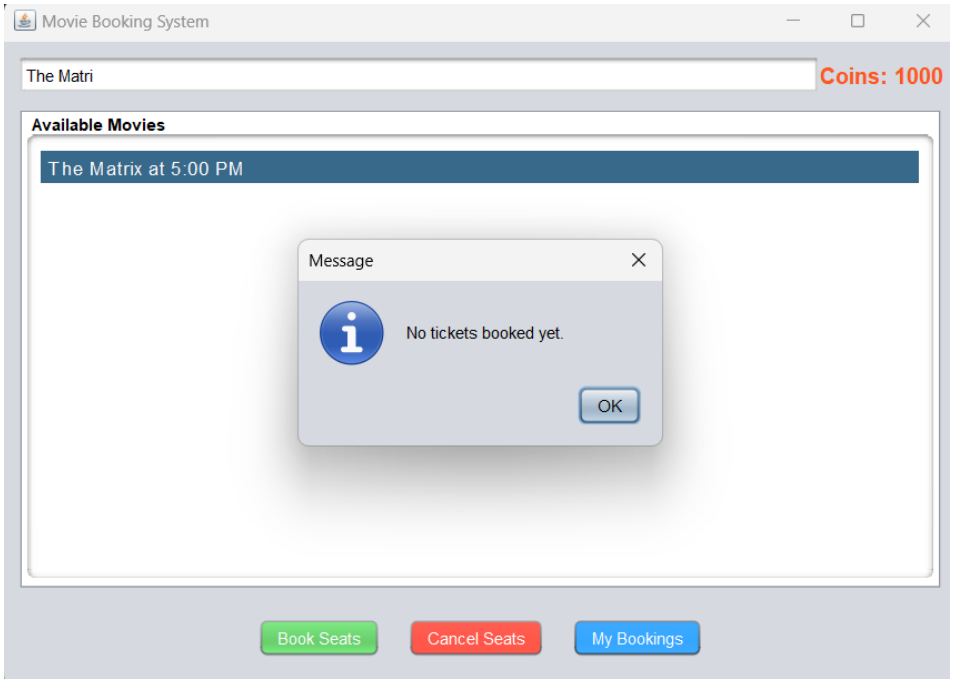

Figure 6

5. References :-

5.1 Official Documentation

Java Documentation: https://docs.oracle.com/en/java/

Swing Documentation: https://docs.oracle.com/javase/tutorial/uiswing/

5.2 Educational Platforms

GeeksforGeeks: https://www.geeksforgeeks.org/

W3Schools: https://www.w3schools.com/

Coursera: https://www.coursera.org/

5.3 Programming Communities

Stack Overflow: https://stackoverflow.com/

GitHub: https://github.com/

5. 4 Software Development Blogs

Medium: https://medium.com/

Dev.to: https://dev.to/

5.5 Code Repositories and Examples

GitHub Gists: https://gist.github.com