A
Mini Project Report
On

Movie Booking System

Presented by

SACHIN RAVINDRA CHOUDHARY
SY[CSE]

Computer Science and Engineering
(2025 – 2026)

Guided By

Ms. N. L. Pariyal
(Department of Computer Science and Engineering)

Submitted to

MGM's College of Engineering, Nanded

Under

Dr. Babasaheb Ambedkar Technological University, Lonere

Certificate

This is to certify that the mini project entitled

Movie Booking System

Submitted By

SACHIN RAVINDRA CHOUDHARY

In satisfactory manner as a partial fulfillment of
SY[CSE] in Computer Science and Engineering

To

MGM's College of Engineering, Nanded

Under
Dr. Babasaheb Ambedkar Technological University, Lonere
has been carried out under my guidance,

Ms. N. L. Pariyal
Mini Project Guide

Dr. Mrs. A. M. Rajurkar	**Dr. Mrs. G. S. Lathkar**
H.O.D	**Director**
Computer Science & Engineering	MGM's College of Engg., Nanded

1.Abstract :-

The **Movie Booking System** is a Java-based console application that simulates an online movie ticket reservation process. It allows users to select a movie, view seat availability, and book seats using a simple text-based interface. The system updates seat status in real time to prevent duplicate bookings and supports booking multiple seats in one session. Developed using core Java concepts such as arrays, loops, conditional statements, methods, and the Scanner class, this mini project is beginner-friendly and helps students understand real-world booking systems in a practical way

2.Problem Statement:-

In many small cinemas, theatres, and educational demonstrations, movie ticket booking is still managed manually using counters, paper registers, or verbal communication. This traditional approach leads to several issues such as difficulty in tracking seat availability, chances of double booking, long waiting times, and poor customer experience. As the number of movies and viewers increases, managing seat arrangements manually becomes more confusing and inefficient.

Manual booking systems do not provide real-time seat availability, automatic booking updates, or a clear visual representation of seating layouts. As a result, customers may face problems like booking already occupied seats, lack of clarity about available seats, and delays during the booking process. In an educational environment, explaining how an online movie booking system works using only theory does not give students a clear practical understanding of real-world booking operations.

In today's digital era, even basic entertainment services require a simple computer-based booking system that improves efficiency, accuracy, and user interaction. However, many existing movie booking applications are complex, costly, or require advanced technologies, making them unsuitable for beginners and academic practice.

Thus, the problem addressed in this project is:

How to design and implement a simple, fast, and user-friendly Movie Booking System using core Java concepts that can manage movie selection, display seat availability, prevent duplicate bookings, and provide a clear practical understanding of real-world cinema booking systems?

3. Objective of the Project :-

• To develop a **simple Movie Booking System using Java**.
• To allow users to **select a movie and book seats** efficiently.
• To display the **seat layout and real-time seat availability**.
• To prevent **duplicate seat booking** through validation logic.
• To provide a **user-friendly, console-based interface**.

• To help beginners understand **core Java concepts** through a real-life booking application.

---

4. Scope of the Project :-

This project focuses on developing a **simple Movie Booking System using Java**. It allows users to select a movie, view seat availability, and book seats using a two-dimensional array. The system automatically updates seat status after booking. It is **console-based, lightweight, and easy to use**, making it suitable for **beginners and academic practice**.

---

5. Proposed System :-

The proposed system is a **computer-based Movie Booking System developed in Java**. It overcomes the limitations of manual ticket booking by providing **real-time seat display, automatic booking updates, and prevention of double booking**. The system uses **arrays (2D arrays)** to manage seat layouts and provides a simple interactive console interface.

---

6. Existing System

Limitations of the Existing System

1. Manual Seat Management:
Traditional movie ticket booking relies on counters or registers, which can lead to errors and confusion in seat allocation.

2. Time-Consuming Process:
Manually checking seat availability and issuing tickets takes more time and becomes inefficient during peak hours.

---

6. Hardware Requirements :-

The system requires a **minimum 1 GHz processor, 2 GB RAM, and about 100 MB free disk space**. A keyboard and basic monitor are sufficient to run this console-based application. No high-end hardware is required.

Comparison of Traditional vs Computer-Based Movie Booking System

Feature	Traditional Movie Booking System	Computer-Based Movie Booking System (Our Project)
Speed	Slow and manual operations	Fast and automated booking
Accuracy	Prone to human errors	High accuracy using program logic
Seat	Paper-based tracking	Digital seat layout using 2D

Management		arrays
Availability Check	Manual checking	Real-time seat availability
Booking	Manual ticket issue	Automatic seat booking
User Convenience	Difficult to manage	Easy-to-use, console-based system

Table: Comparison of Existing System and Proposed System

Table: Movie Booking Data Parameters

Parameter	Description	Example
Movie List	Available movies for selection	Avengers
Movie Choice	User-selected movie	Option 3
Seat Layout	Seat arrangement matrix	5 × 5
Seat Status	Availability indicator	O / X
Row Number	Seat row selected	2
Column Number	Seat column selected	4
Booking Status	Seat confirmation	Booked
Repeat Booking	Multiple seat option	Yes

---

7. Feasibility Study :-

The **Movie Booking System** is **technically feasible** as it uses core Java concepts and runs on any Java-enabled system. It is **operationally feasible** due to its simple input-based interaction, and **economically feasible** because it uses free software tools and requires no special hardware

8. System Flow :-

(START)

↓

[Display Available Movies]

↓

[User Selects a Movie]

↓

[Display Seat Layout]

↓

[User Enters Row and Column]

↓

[Check Seat Availability]

IF Available → Book Seat

```
    ELSE → Show Error Message
    ↓
[ Update Seat Status ]
    ↓
[ Book Another Seat? ]
    Yes → Repeat
    No  → Continue
    ↓
[ Display Exit Message ]
    ↓
   ( END )
```

SYSTEM DESIGN

9. Use Case Diagram :-

Use Cases:
Select movie, view seat layout, book seat, book multiple seats, exit system.
Description:
The diagram shows how the user interacts with the **Movie Booking System** to select movies, check seat availability, and book seats using a simple console-based interface.

VARIABLE	PURPOSE	DATA TYPE
movies[]	Stores movie names	String[]
seats[][]	Stores seat layout	char[][]
row	Stores selected row	int
col	Stores selected column	int
choice	Stores repeat booking option	int
sc	Accepts user input	Scanner

Table 4.1: Program Variables and Their Purpose

10. Class Diagram :-

The project uses **one main Java class** to manage the Movie Booking system.
• **Attributes:** Arrays for movie list and 2D array for seats.
• **Input Handling:** Uses Scanner to accept user input.
• **Logic Section:** Implements seat checking and booking validation.

• **Method Block:** Displays seat layout and handles booking.
• **The main method** controls program flow and execution.

---

11. Sequence Diagram :-

The application is started by the user.
The list of movies is displayed.
The user selects a movie.
The seat layout is displayed.
The user selects row and column for booking.
The system checks seat availability and updates booking status.
The program continues until the user chooses to exit.

IMPLEMENTATION

12. Introduction to Implementation :-

The implementation phase focuses on developing a **console-based Movie Booking System using Java** by converting the system design into a functional program. It involves storing movie names in arrays, managing seat layouts using two-dimensional arrays, handling seat booking logic, and updating seat status. Core Java concepts such as **arrays, loops, conditional statements, methods, and the Scanner class** are used.

---

13. Program Structure :-

The project is divided into two main components.
The **main class** manages movie selection, seat display, and user input.
The **processing logic** handles seat validation, booking updates, and repeated booking operations.
This structure makes the system **simple, efficient, and easy to understand**.

---

14. Code Snippet :-

```
import java.util.*;

public class MovieBookingSystem {

    static void displaySeats(char[][] seats) {
        for (int i = 0; i < seats.length; i++) {
            for (int j = 0; j < seats[i].length; j++) {
                System.out.print(seats[i][j] + " ");
            }
            System.out.println();
        }
    }
```

```
public static void main(String[] args) {
    Scanner sc = new Scanner(System.in);

    String[] movies = {
        "Dhurandhar",
        "Toxic",
        "Ramayan",
        "Maddock Horror Comedy Universe: Bhediya 2",
        "YRF Alpha",
        "Avengers: Doomsday",
        "Spiderman: Brand New Day"
    };

    char[][] seats = new char[5][5];

    for (int i = 0; i < seats.length; i++) {
        for (int j = 0; j < seats[i].length; j++) {
            seats[i][j] = 'O';
        }
    }

    System.out.println("Welcome to Movie Booking System!");
    System.out.println("Available Movies:");
    for (int i = 0; i < movies.length; i++) {
        System.out.println((i + 1) + ". " + movies[i]);
    }

    System.out.print("Select a movie (1-" + movies.length + "): ");
    int movieChoice = sc.nextInt();

    if (movieChoice < 1 || movieChoice > movies.length) {
        System.out.println("Invalid movie selection. Exiting...");
        return;
    }

    System.out.println("You selected: " + movies[movieChoice - 1]);

    int choice;
    do {
        System.out.println("\nSeat Layout (O = Available, X = Booked):");
        displaySeats(seats);

        System.out.print("Enter row (1-5): ");
        int row = sc.nextInt() - 1;
        System.out.print("Enter column (1-5): ");
        int col = sc.nextInt() - 1;

        if (row >= 0 && row < 5 && col >= 0 && col < 5) {
            if (seats[row][col] == 'O') {
                seats[row][col] = 'X';
                System.out.println("Seat booked successfully!");
            } else {
                System.out.println("Seat already booked. Try another.");
            }
        } else {
            System.out.println("Invalid seat selection.");
        }

        System.out.print("Do you want to book another seat? (1 = Yes, 0 = No): ");
        choice = sc.nextInt();
    } while (choice == 1);

    System.out.println("Thank you for booking!");
  }
}
```

---

15. Output Sample :-

Case1:Invaild Movie Selection :

```
Welcome to Movie Booking System!
Available Movies:
1. Dhurandhar
2. Toxic
3. Ramayan
4. Maddock Horror Comedy Universe: Bhediya 2
5. YRF Alpha
6. Avengers: Doomsday
7. Spiderman: Brand New Day
Select a movie (1-7): 9
Invalid movie selection. Exiting...
```

Case 2: Valid Movie, Seat Available → Booked Successfully

```
Select a movie (1-7): 2
You selected: Toxic

Seat Layout (O = Available, X = Booked):
O O O O O
O O O O O
O O O O O
O O O O O
O O O O O
Enter row (1-5): 2
Enter column (1-5): 3
Seat booked successfully!
Do you want to book another seat? (1 = Yes, 0 = No): 0
Thank you for booking!
```

Case 3: Valid Movie, Seat Already Booked

```
Select a movie (1-7): 3
You selected: Ramayan

Seat Layout (O = Available, X = Booked):
O O O O O
O O O O O
O O O O O
O O O O O
O O O O O
Enter row (1-5): 1
Enter column (1-5): 1
Seat booked successfully!
Do you want to book another seat? (1 = Yes, 0 = No): 1

Seat Layout (O = Available, X = Booked):
X O O O O
O O O O O
O O O O O
O O O O O
O O O O O
Enter row (1-5): 1
Enter column (1-5): 1
Seat already booked. Try another.
Do you want to book another seat? (1 = Yes, 0 = No): 0
Thank you for booking!
```

Case 4: Invalid Seat Selection (out of range)

```
Select a movie (1-7): 5
You selected: YRF Alpha

Seat Layout (O = Available, X = Booked):
O O O O O
O O O O O
O O O O O
O O O O O
O O O O O
Enter row (1-5): 6
Enter column (1-5): 2
Invalid seat selection.
Do you want to book another seat? (1 = Yes, 0 = No): 0
Thank you for booking!
```

Case 5: Multiple Seats Booked Successfully

```
Select a movie (1-7): 7
You selected: Spiderman: Brand New Day

Seat Layout (O = Available, X = Booked):
```

```
O O O O O
O O O O O
O O O O O
O O O O O
O O O O O
Enter row (1-5): 1
Enter column (1-5): 1
Seat booked successfully!
Do you want to book another seat? (1 = Yes, 0 = No): 1

Seat Layout (O = Available, X = Booked):
X O O O O
O O O O O
O O O O O
O O O O O
O O O O O
Enter row (1-5): 5
Enter column (1-5): 5
Seat booked successfully!
Do you want to book another seat? (1 = Yes, 0 = No): 0
Thank you for booking!
```

---

16. Line-by-Line Explanation :-

Main Class Description

The project uses one main Java class to manage the Movie Booking system.

Input Section

Accepts movie choice, seat row, and column from the user.

Processing & Logic Section

• Check seat availability
• Prevent duplicate booking
• Update seat layout

Output Section

• Displays movie list
• Shows seat layout
• Displays booking confirmation or error messages

Description of Modules

Module 1: Input Module
Accepts user movie and seat inputs.
Module 2: Seat Display Module
Displays current seat layout.
Module 3: Booking Module
Handles seat validation and booking.
Module 4: Output Module
Displays confirmations and messages.
Module 5: Control Module
Controls booking repetition and exit.

---

17. Working Procedure :-
The user runs the program and selects a movie.
The seat layout is displayed.
The user selects a seat.
The system validates and books the seat.
The process repeats until the user exits.

---

18. Summary :-
This project demonstrates the use of **core Java concepts** such as arrays, loops, and conditional logic. It provides a **simple Movie Booking System** with real-time seat booking and validation. The program is easy to understand and suitable for academic use.

CONCLUSION
The **Movie Booking System** successfully demonstrates how basic Java programming concepts can be applied to solve a real-world problem. The project provides an automated way to manage movie selection and seat booking, reducing errors and saving time compared to manual booking methods.
The project effectively uses **arrays, loops, conditional statements, methods, and user input handling** to build a simple yet practical console-based application. The clear program structure makes it easy for beginners to understand how Java logic works in real applications.
Overall, the project fulfills its objectives and serves as a valuable learning tool for students, proving that even simple Java programs can create meaningful and practical solutions.