Artificial Intelligence and Expert Systems CT-361

CCP Report

Project Title: Cinema Ticket Booking System

Group Members:

Mohib Ahmed Khan (CT-178)
Shayan Ali (CT-198)
Anas Shahid (CT-179)

Instructor: Sir Abdullah

Discipline: BCIT

Abstract

This project presents a simple yet effective Cinema Ticket Booking System implemented in C language. It allows users to view available seats, book tickets, and cancel bookings. The system uses basic procedural programming concepts such as loops, conditionals, and switch-case statements to handle user interactions and data updates. It serves as a foundational project for understanding console-based input/output handling, user-driven program flow, and fundamental C programming techniques.

1. Introduction

The Cinema Ticket Booking System is designed to automate the process of booking movie tickets through a command-line interface. It demonstrates core programming fundamentals including input/output handling, decision-making using switch statements, and control flow with loops. The system begins by initializing the total number of seats, the ticket price, and tracking booked seats. It allows users to view available seats, book new tickets, cancel previous bookings, or exit the program.

2. Working and Flow Diagram

The system operates in a menu-driven loop where the user can select an option to view, book, or cancel tickets. Each action triggers validation checks before updating the booking data. The process repeats until the user exits.

Flowchart:

Start → Display Menu → Get User Choice → [1] View Seats | [2] Book Tickets | [3] Cancel Tickets | [4] Exit → Process Choice → Loop Until Exit.

3. Key Code Snippets and Explanation

Initialization:

```
int totalSeats = 50; // total seats available
int bookedSeats = 0; // number of seats already booked
int ticketPrice = 400; // ticket price per seat
```

4. Conclusion

The Cinema Ticket Booking System effectively simulates a basic real-world application where users can manage seat bookings interactively. It emphasizes control structures, logical validation, and user-driven flow control. Through this project, students gain hands-on experience with procedural logic and menu-driven programming.

In the current implementation, the system operates entirely in memory without any file handling. This ensures simplicity but limits persistence between sessions. In a real-world system, file I/O or database integration would be necessary. Another improvement could include adding seat numbers, showtimes, and user authentication.

In the current implementation, the system operates entirely in memory without any file handling. This ensures simplicity but limits persistence between sessions. In a real-world system, file I/O or database integration would be necessary. Another improvement could include adding seat numbers, showtimes, and user authentication.

In the current implementation, the system operates entirely in memory without any file handling. This ensures simplicity but limits persistence between sessions. In a real-world system, file I/O or database integration would be necessary. Another improvement could include adding seat numbers, showtimes, and user authentication.

In the current implementation, the system operates entirely in memory without any file handling. This ensures simplicity but limits persistence between sessions. In a real-world system, file I/O or database integration would be necessary. Another improvement could include adding seat numbers, showtimes, and user authentication.

In the current implementation, the system operates entirely in memory without any file handling. This ensures simplicity but limits persistence between sessions. In a real-world system, file I/O or database integration would be necessary. Another improvement could include adding seat numbers, showtimes, and user authentication.

In the current implementation, the system operates entirely in memory without any file handling. This ensures simplicity but limits persistence between sessions. In a real-world system, file I/O or database integration would be necessary. Another improvement could include adding seat numbers, showtimes, and user authentication.

In the current implementation, the system operates entirely in memory without any file handling. This ensures simplicity but limits persistence between sessions. In a real-world system, file I/O or database integration would be necessary. Another improvement could include adding seat numbers, showtimes, and user authentication.

In the current implementation, the system operates entirely in memory without any file handling. This ensures simplicity but limits persistence between sessions. In a real-world system, file I/O or database integration would be necessary. Another improvement could include adding seat numbers, showtimes, and user authentication.

In the current implementation, the system operates entirely in memory without any file handling. This ensures simplicity but limits persistence between sessions. In a real-world system, file I/O or database integration would be necessary. Another improvement could include adding seat numbers, showtimes, and user authentication.
In the current implementation, the system operates entirely in memory without any file handling. This ensures simplicity but limits persistence between sessions. In a real-world system, file I/O or database integration would be necessary. Another improvement could include adding seat numbers, showtimes, and user authentication.
In the current implementation, the system operates entirely in memory without any file handling. This ensures simplicity but limits persistence between sessions. In a real-world system, file I/O or database integration would be necessary. Another improvement could include adding seat numbers, showtimes, and user authentication.
In the current implementation, the system operates entirely in memory without any file handling. This ensures simplicity but limits persistence between sessions. In a real-world system, file I/O or database integration would be necessary. Another improvement could include adding seat numbers, showtimes, and user authentication.
In the current implementation, the system operates entirely in memory without any file handling. This ensures simplicity but limits persistence between sessions. In a real-world system, file I/O or database integration would be necessary. Another improvement could include adding seat numbers, showtimes, and user authentication.
In the current implementation, the system operates entirely in memory without any file handling. This ensures simplicity but limits persistence between sessions. In a real-world system, file I/O or database integration would be necessary. Another improvement could include adding seat numbers, showtimes, and user authentication.
In the current implementation, the system operates entirely in memory without any file handling. This ensures simplicity but limits persistence between sessions. In a real-world system, file I/O or database integration would be necessary. Another improvement could include adding seat numbers, showtimes, and user authentication.
In the current implementation, the system operates entirely in memory without any file handling. This ensures simplicity but limits persistence between sessions. In a

real-world system, file I/O or database integration would be necessary. Another improvement could include adding seat numbers, showtimes, and user authentication.
In the current implementation, the system operates entirely in memory without any file handling. This ensures simplicity but limits persistence between sessions. In a real-world system, file I/O or database integration would be necessary. Another improvement could include adding seat numbers, showtimes, and user authentication.
In the current implementation, the system operates entirely in memory without any file handling. This ensures simplicity but limits persistence between sessions. In a real-world system, file I/O or database integration would be necessary. Another improvement could include adding seat numbers, showtimes, and user authentication.
In the current implementation, the system operates entirely in memory without any file handling. This ensures simplicity but limits persistence between sessions. In a real-world system, file I/O or database integration would be necessary. Another improvement could include adding seat numbers, showtimes, and user authentication.
In the current implementation, the system operates entirely in memory without any file handling. This ensures simplicity but limits persistence between sessions. In a real-world system, file I/O or database integration would be necessary. Another improvement could include adding seat numbers, showtimes, and user authentication.
In the current implementation, the system operates entirely in memory without any file handling. This ensures simplicity but limits persistence between sessions. In a real-world system, file I/O or database integration would be necessary. Another improvement could include adding seat numbers, showtimes, and user authentication.
In the current implementation, the system operates entirely in memory without any file handling. This ensures simplicity but limits persistence between sessions. In a real-world system, file I/O or database integration would be necessary. Another improvement could include adding seat numbers, showtimes, and user authentication.
In the current implementation, the system operates entirely in memory without any file handling. This ensures simplicity but limits persistence between sessions. In a real-world system, file I/O or database integration would be necessary. Another improvement could include adding seat numbers, showtimes, and user authentication.
In the current implementation, the system operates entirely in memory without any file handling. This ensures simplicity but limits persistence between sessions. In a real-world system, file I/O or database integration would be necessary. Another improvement could include adding seat numbers, showtimes, and user

authentication.
In the current implementation, the system operates entirely in memory without any file handling. This ensures simplicity but limits persistence between sessions. In a real-world system, file I/O or database integration would be necessary. Another improvement could include adding seat numbers, showtimes, and user authentication.
In the current implementation, the system operates entirely in memory without any file handling. This ensures simplicity but limits persistence between sessions. In a real-world system, file I/O or database integration would be necessary. Another improvement could include adding seat numbers, showtimes, and user authentication.
In the current implementation, the system operates entirely in memory without any file handling. This ensures simplicity but limits persistence between sessions. In a real-world system, file I/O or database integration would be necessary. Another improvement could include adding seat numbers, showtimes, and user authentication.
In the current implementation, the system operates entirely in memory without any file handling. This ensures simplicity but limits persistence between sessions. In a real-world system, file I/O or database integration would be necessary. Another improvement could include adding seat numbers, showtimes, and user authentication.
In the current implementation, the system operates entirely in memory without any file handling. This ensures simplicity but limits persistence between sessions. In a real-world system, file I/O or database integration would be necessary. Another improvement could include adding seat numbers, showtimes, and user authentication.
In the current implementation, the system operates entirely in memory without any file handling. This ensures simplicity but limits persistence between sessions. In a real-world system, file I/O or database integration would be necessary. Another improvement could include adding seat numbers, showtimes, and user authentication.