

Movie Ticket System

Submitted By

Student Name	Student ID
Akif Zahir	251-15-239
Umme Habiba Arpa	251-25-633
Nahida Rowshan	251-15-881
Parvej Rahim	251-15-894
Sajid Islam	251-15-898

MINI LAB PROJECT REPORT

This Report Presented in Partial Fulfilment of the course **CSE114:**
Programming and Problem Solving Lab in the Computer Science and Engineering
Department



DAFFODIL INTERNATIONAL UNIVERSITY

Dhaka, Bangladesh

August 17, 2025

DECLARATION

We hereby declare that this lab project has been done by us under the supervision of **Md. Jakaria Zobair**, Lecturer, Department of Computer Science and Engineering, Daffodil International University. We also declare that neither this project nor any part of this project has been submitted elsewhere as lab projects.

Submitted To:






Md. Jakaria Zobair

Lecturer

Department of Computer Science and Engineering

Daffodil International University

Submitted by

 <hr/> Sajid Islam Saad 251-15-898 Dept. of CSE, DIU	
 <hr/> Akif Zahir 251-15-239 Dept. of CSE, DIU	 <hr/> Umme Habiba Arpa 251-15-633 Dept. of CSE, DIU
 <hr/> Nahida Rowshan 251-15-881 Dept. of CSE, DIU	 <hr/> Parvej Rahim 251-15-894 Dept. of CSE, DIU

COURSE & PROGRAM OUTCOME

The following course have course outcomes as following:

Table 1: Course Outcome Statements

CO's	Statements
CO1	Able to solve computing problems, Expression Evaluation using programming concepts and learn the basic concept of ACM Problem solving techniques.
CO2	Able to apply fundamental programming elements including: variable, use of data types and data structures, decision structures, loop structures, pointer, string, console, file IO, and functions
CO3	Able to specify the problem requirements, analyze the problem, design the algorithm to solve the problem and implement with the help of programming language.
CO4	Able to apply the knowledge of programming and problem solving in real life problems.

Table 2: Mapping of CO, PO, Blooms, KP and CEP

CO	PO	Blooms	KP	CEP
CO1	PO1	C1, C2	KP3	EP1, EP3
CO2	PO2	C2, C3	KP3	EP1, EP3
CO3	PO3	C4, A1	KP3	EP1, EP2
CO4	PO3	C3, C6, A3, P3	KP4	EP1, EP3

The mapping justification of this table is provided in section **4.3.1**, **4.3.2** and **4.3.3**.

Table of Contents

Declaration	i
Course & Program Outcome	ii
1. Introduction	1
1.1 Introduction.....	1
1.2 Motivation.....	1
1.3 Objectives.....	2
1.4 Feasibility Study.....	2
1.5 Gap Analysis.....	2
1.6 Project Outcome.....	3
2. Proposed Methodology/Architecture	4
2.1 Requirement Analysis & Design Specification.....	4
2.1.1 Overview.....	4
2.1.2 Proposed Methodology/ System Design.....	4
2.1.3 UI Design.....	6
2.2 Overall Project Plan.....	6

3	Implementation and Results	7
3.1	Implementation.....	7
3.2	Performance Analysis.....	12
3.3	Results and Discussion.....	12
4	Engineering Standards and Mapping	14
4.1	Impact on Society, Environment and Sustainability.....	14
4.1.1	Impact on Life.....	14
4.1.2	Impact on Society & Environment.....	14
4.1.3	Ethical Aspects.....	14
4.1.4	Sustainability Plan.....	15
4.2	Project Management and Team Work.....	15
4.3	Complex Engineering Problem.....	16
4.3.1	Mapping of Program Outcome.....	16
4.3.2	Complex Problem Solving.....	16
4.3.3	Engineering Activities.....	17
5	Conclusion	18
5.1	Summary.....	18
5.2	Limitation.....	18
5.3	Future Work.....	19
	References	20

Chapter 1

Introduction

This chapter provides an overview of the project, its purpose, motivation, objectives, feasibility, research gaps, and the expected outcomes. It explains why the project was undertaken and how it addresses the problem statement.

1.1 Introduction

The movie ticket booking system is designed to streamline the process of reserving cinema tickets by providing a clear and interactive console-based interface. Traditional ticket booking methods, such as in-person queues or phone reservations, can be time-consuming and error-prone. This project implements a C program that allows users to select from 4 movies and 3 time slots, view available seats, purchase or cancel tickets, and track sold seats efficiently using a 5×10 2D array for each movie and time slot combination. The modular design ensures clarity, maintainability, and easy expansion for additional movies or time slots.

1.2 Motivation

In today's fast-paced environment, efficient ticket booking systems are essential for both customers and cinema operators. The computational motivation behind this project is to create a **simple yet functional application** that demonstrates the use of arrays, loops, and basic input/output in C programming. This project benefits students by reinforcing fundamental programming concepts while solving a real-world problem. Additionally, it helps develop skills in modular coding, input validation, and user interaction design.

1.3 Objectives

The main objectives of this project are:

1. To design a **2D seating chart** representing available and sold seats.
2. To implement a **menu-driven application** allowing users to:
 - View the current seating layout.
 - Purchase tickets for available seats.
 - Display a summary of sold and available tickets.
3. To ensure **input validation**, preventing users from selecting already sold or invalid seats.
4. To provide a **modular and beginner-friendly C program** that is easy to read, maintain, and expand in the future.
5. Optionally, to explore methods for saving and loading seating data between sessions.

1.4 Feasibility Study

Several ticket booking systems exist in the real world, ranging from web-based platforms to mobile applications. Existing projects often focus on advanced features like online payments, real-time seat tracking, and database integration. However, for **beginner-level programming practice**, similar academic projects typically use **2D arrays in C** to simulate seating arrangements. Methodologies from these projects emphasize array handling, loops, conditionals, and menu-driven programming. By reviewing these studies, it is evident that a simplified, text-based C application is a feasible and practical approach for learning and assessment purposes.

1.5 Gap Analysis

While professional ticket booking systems are feature-rich, beginner-level projects often lack **input validation, clear seat labeling, and modular code structure**. Our project addresses these gaps by implementing:

- A **menu-driven interface** for easy navigation.

- Visual representation of seats with clear indicators for availability.
- **Basic validation** to prevent invalid selections.

This approach bridges the gap between real-world complexity and beginner-level programming skills, providing a practical and understandable solution.

1.6 Project Outcome

Upon completion, this project is expected to produce:

1. A fully functional **movie ticket booking application** using C programming.
2. A **2D seating chart interface** that allows users to view and select seats efficiently.
3. Accurate tracking of **sold and available tickets**.
4. A foundation for further enhancements, such as saving/loading seat data or integrating with a graphical user interface.

The project not only demonstrates the application of programming concepts but also provides a **practical solution** to a common real-life problem, making it an effective learning experience for students.

Chapter 2

Proposed Methodology/Architecture

This chapter describes the design and methodology of the movie ticket booking system. It explains how the system is structured, how the user interacts with it, and the overall planning of the project.

2.1 Requirement Analysis & Design Specification

2.1.1 Overview

The movie ticket booking system is developed to efficiently manage seat reservations for multiple movies and show times. The system uses a **four-dimensional array** to handle different movies, their show timings, and the corresponding seating charts.

It allows users to:

- Select a movie and showtime.
- View available seats.
- Purchase tickets with validation.
- Track sold and available seats.

The system is **menu-driven, modular, and beginner-friendly**, demonstrating the use of multi-dimensional arrays in C programming.

2.1.2 Proposed Methodology / System Design

The system follows a **modular and procedural methodology**, where each function performs a specific task. The seating structure is declared as: `char arr[4][3][5][10];`

- **[4] → Movies** (up to 4 movies)
- **[3] → Show times** (3 shows per movie)
- **[5] → Rows of seats**
- **[10] → Seats per row**

This provides **12 unique shows**, each with a 5×10 seating chart (50 seats).

Main components include:

1. **Seating Initialization Module** – Marks all seats as available ('O') across all movies and shows.
2. **Movie & Show Selection Module** – Allows the user to choose a movie and show time before booking.
3. **Seat Display Module** – Shows the 5×10 chart, marking available ('O') and sold ('X') seats.
4. **Ticket Purchase Module** – Lets users select a seat, validates input, and marks it as sold ('X').
5. **Validation Module** – Ensures the chosen seat is within range and not already booked.
6. **Seat Summary Module** – Calculates and displays available and sold tickets for a specific movie show.
7. **Menu Interface** – Guides the user through all operations until exit.

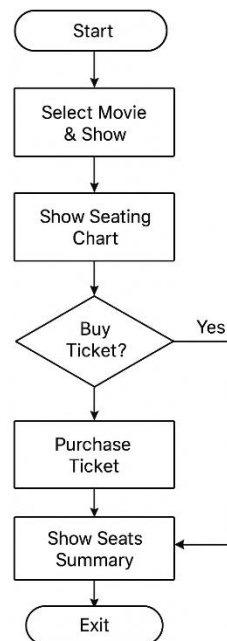


Figure 2.1: Proposed Methodology Flowchart of the Movie Ticket Booking System

2.1.3 UI Design

The user interface is **text-based**, emphasizing simplicity and clarity:

- **Seating Chart Display:**
 - 'O' = Available seat
 - 'X' = Sold seat
 - Rows and columns displayed for easy selection
- **Menu Options:**
 1. Select Movie & Show
 2. Display Seats
 3. Buy Seat
 4. Show Seats Summary
 5. Exit
- **User Messages:** Informative prompts guide the user, e.g., “Seat Unavailable” if already sold, or “Purchase Successful” after booking.

2.2 Overall Project Plan

Phase	Time Taken
Planning phase	1 week
Designing phase	1 week
Implementation phase	2 weeks
Q&A Testing and debugging phase	2 days
Documentation phase	1 week

Chapter 3

Implementation and Results

This chapter presents the implementation of the movie ticket booking system and demonstrates its results. It explains how the system was developed in C, analyses its behaviour, and discusses the output generated during testing.

3.1 Implementation

The movie ticket booking system is implemented as a **menu-driven console application** using C programming. The program uses a **4×3×5×10 array** to represent the seating arrangement for multiple movies and time slots. Each seat is represented by a character: 'o' for available seats and 'x' for booked seats.

The system is modular, with distinct functions handling specific operations such as seat initialization, display, booking, cancellation, ticket summary, and menu navigation. The program continuously runs until the user chooses to exit.

The full code is presented below:

```
#include <stdio.h>

// Global seat arrangement: Movie, Time slot, Row, Col
char arr[4][3][5][10];

// Function Prototypes
void show_interface(int a, int b);
int select_movie();
int select_time();
void initialize();
void display_seats(int p, int q);
void buy_ticket(int p, int q);
void cancel_seat(int p, int q);
void ticket_summary(int p, int q);
void welcome();
void menu();
void movie_name(int a);
void show_time(int a);
```

```

// Main Function
int main() {
    int movie, time;
    initialize();
    while (1) {
        welcome();
        movie = select_movie();

        if (movie >= 1 && movie <= 4) {
            while (1) {
                welcome();
                movie_name(movie);
                time = select_time();

                if (time == 0) {
                    break;
                }
                else if (time >= 1 && time <= 3) {
                    show_interface(movie, time);
                }
                else {
                    printf("Invalid choice.\n");
                }
            }
        }
        else if (movie == 0) {
            printf("Exiting the program.\n");
            break;
        }
        else {
            printf("Invalid choice.\n");
        }
    }
    return 0;
}

// Initialize Seats
void initialize() {
    for (int i = 0; i < 4; i++) {
        for (int j = 0; j < 3; j++) {
            for (int k = 0; k < 5; k++) {
                for (int l = 0; l < 10; l++) {
                    arr[i][j][k][l] = '0';
                }
            }
        }
    }
}

// Welcome Banner
void welcome() {
    printf("_____Welcome to Daffodil Cineplex_____\n");
    printf("-----Bringing stories of life\n\n\n");
}

```

```

// Movie Selection
int select_movie() {
    int m;
    printf("Now Showing: \n\n");
    printf("1. Aynabaji\n");
    printf("2. Utsob\n");
    printf("3. Borbaad\n");
    printf("4. Tandob\n\n");
    printf("0. Exit the Program \n\n");
    printf("Please Select a movie: ");
    scanf("%d", &m);
    return m;
}

// Movie Names
void movie_name(int a) {
    if (a == 1) {
        printf("Aynabaji >\n");
    }
    else if (a == 2) {
        printf("Utsob >\n");
    }
    else if (a == 3) {
        printf("Borbaad >\n");
    }
    else if (a == 4) {
        printf("Tandob >\n");
    }
}

// Time Slot Selection
int select_time() {
    int t;
    printf("Time slots: \n\n");
    printf("1. 10:00 am -- 01:00 pm\n");
    printf("2. 02:00 pm -- 05:00 pm\n");
    printf("3. 06:00 pm -- 09:00 pm\n\n");
    printf("0. Back to movie selection. \n\n");
    printf("Please Select a time slot: ");
    scanf("%d", &t);
    return t;
}

// Show Time
void show_time(int a) {
    if (a == 1) {
        printf(" 10:00 am -- 01:00 pm >\n");
    }
    else if (a == 2) {
        printf(" 02:00 pm -- 05:00 pm >\n");
    }
    else if (a == 3) {
        printf(" 06:00 pm -- 09:00 pm >\n");
    }
}

```

```

// Show Interface
void show_interface(int a, int b) {
    int input;
    while (1) {
        movie_name(a);
        show_time(b);

        menu();
        scanf("%d", &input);

        if (input == 1) {
            display_seats(a, b);
        }
        else if (input == 2) {
            buy_ticket(a, b);
        }
        else if (input == 3) {
            cancel_seat(a, b);
        }
        else if (input == 4) {
            ticket_summary(a, b);
        }
        else if (input == 0) {
            printf("Back to time slot menu.\n");
            break;
        }
        else {
            printf("Invalid choice.\n");
        }
    }
}

// Menu Interface
void menu() {
    printf("\n1. Display Seats\n");
    printf("2. Buy Seats\n");
    printf("3. Cancel Seat\n");
    printf("4. Seats Summary\n");
    printf("0. Go To Movie Time slot\n");
    printf("Enter your Choice: ");
}

// Display Seat Layout
void display_seats(int p, int q) {
    printf("\nSeat Layout (0 = Available, X = Booked):\n");
    printf("\n_____ \n\n\n");
    for (int i = 0; i < 5; i++) {
        for (int j = 0; j < 10; j++) {
            printf(" %c ", arr[p-1][q-1][i][j]);
        }
        printf("\n");
    }
}

```

```

// Buy Ticket
void buy_ticket(int p, int q) {
    int x, y;
    printf("Please enter Row number (1-5): ");
    scanf("%d", &x);
    printf("Please enter Col number (1-10): ");
    scanf("%d", &y);

    if (x < 1 || x > 5 || y < 1 || y > 10) {
        printf("Invalid seat selection.\n\n\n");
        return;
    }

    if (arr[p-1][q-1][x-1][y-1] == 'X') {
        printf("Seat already taken.\n\n\n");
    } else {
        arr[p-1][q-1][x-1][y-1] = 'X';
        printf("Purchase Successful!\n\n\n");
    }
}

// Cancel Ticket
void cancel_seat(int p, int q) {
    int x, y;
    printf("Please enter Row number (1-5): ");
    scanf("%d", &x);
    printf("Please enter Col number (1-10): ");
    scanf("%d", &y);

    if (x < 1 || x > 5 || y < 1 || y > 10) {
        printf("Invalid seat selection.\n\n\n");
        return;
    }

    if (arr[p-1][q-1][x-1][y-1] == 'O') {
        printf("Seat not taken yet.\n\n\n");
    } else {
        arr[p-1][q-1][x-1][y-1] = 'O';
        printf("Ticket cancelled successfully!\n\n\n");
    }
}

// Ticket Summary
void ticket_summary(int p, int q) {
    int count = 0;
    for (int i = 0; i < 5; i++) {
        for (int j = 0; j < 10; j++) {
            if (arr[p-1][q-1][i][j] == 'X') {
                count++;
            }
        }
    }
    printf("\nSeat Available: %d\n\n", 50 - count);
    printf("Ticket Sold: %d\n\n", count);
}

```


3.2 Performance Analysis

The system was tested under standard usage scenarios. Key observations include:

- **Efficiency:** Operations such as seat display, booking, and cancellation execute instantly, even with multiple movies and time slots.
- **Accuracy:** The system reliably tracks sold and available seats. Invalid seat selections are prevented.
- **Usability:** The console-based interface is simple, guiding users clearly through the menu options and seat selection process.
- **Scalability:** The array structure allows easy expansion to more movies, time slots, or seats without affecting performance.

Since this project is beginner-level, no advanced optimization is required. The system successfully meets all functional requirements.

3.3 Results and Discussion

The system was tested by performing the following operations:

1. **Display Seats:** Initially, all seats are shown as 'O' (available).
2. **Purchase Ticket:** Selecting a seat updates it to 'X' (sold). Attempting to select the same seat again produces an “Unavailable” message.
3. **Seat Summary:** Accurately counts available and sold seats after multiple purchases.
4. **Exit Program:** Safely terminates the application without errors.

Sample Input and Output:

```
Welcome to Daffodil Cineplex
-----Bringing stories of life

Now Showing:
1. Aynabaji
2. Utsob
3. Borbaad
4. Tandob
0. Exit the Program

Please Select a movie: 1
Welcome to Daffodil Cineplex
-----Bringing stories of life

Aynabaji >
Time slots:
1. 10:00 am -- 01:00 pm
2. 02:00 pm -- 05:00 pm
3. 06:00 pm -- 09:00 pm
0. Back to movie selection.

Please Select a time slot: 2
Aynabaji >
02:00 pm -- 05:00 pm >

1. Display Seats
2. Buy Seats
3. Cancel Seat
4. Seats Summary
0. Go To Movie slot
Enter your Choice: 1

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 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
```

```
Aynabaji >
02:00 pm -- 05:00 pm >

1. Display Seats
2. Buy Seats
3. Cancel Seat
4. Seats Summary
0. Go To Movie slot
Enter your Choice: 2
Please enter Row number (1-5): 3
Please enter Col number (1-10): 5
Purchase Successful!

Aynabaji >
02:00 pm -- 05:00 pm >

1. Display Seats
2. Buy Seats
3. Cancel Seat
4. Seats Summary
0. Go To Movie slot
Enter your Choice: 1

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 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 O

Aynabaji >
02:00 pm -- 05:00 pm >

1. Display Seats
2. Buy Seats
3. Cancel Seat
4. Seats Summary
0. Go To Movie slot
Enter your Choice:
```

```
Aynabaji >
02:00 pm -- 05:00 pm >

1. Display Seats
2. Buy Seats
3. Cancel Seat
4. Seats Summary
0. Go To Movie slot
Enter your Choice: 3
Please enter Row number (1-5): 3
Please enter Col number (1-10): 5
Ticket cancelled successfully!

Aynabaji >
02:00 pm -- 05:00 pm >

1. Display Seats
2. Buy Seats
3. Cancel Seat
4. Seats Summary
0. Go To Movie slot
Enter your Choice: 1

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 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

Aynabaji >
02:00 pm -- 05:00 pm >

1. Display Seats
2. Buy Seats
3. Cancel Seat
4. Seats Summary
0. Go To Movie slot
Enter your Choice:
```

Discussion:

- The system fulfills all the objectives outlined in Chapter 1.
- The modular approach allows easy understanding and maintenance of the code.
- Although the program does not implement optional features like file save/load or graphical interface, it provides a functional and clear simulation of a real-world cinema ticket booking system.
- Minor improvements, such as adding row/column headers and better input validation, could enhance usability further.

Chapter 4

Engineering Standards and Mapping

This chapter discusses the impact of the movie ticket booking system on society, environment, and sustainability. It also presents project management details, complex problem analysis, and the mapping of program outcomes to engineering standards.

4.1 Impact on Society, Environment, and Sustainability

4.1.1 Impact on Life

The movie ticket booking system improves the **user experience** for cinema-goers by providing a quick and reliable method to book tickets. It reduces the time and effort spent in manual ticket purchasing, thus enhancing convenience in daily life.

4.1.2 Impact on Society & Environment

- **Societal Impact:** Encourages organized ticket sales, reduces human errors, and minimizes crowding at ticket counters.
- **Environmental Impact:** Reduces paper usage by simulating ticket booking digitally, promoting a small step toward environmental conservation.

4.1.3 Ethical Aspects

The system ensures **fair access** to available seats, preventing duplicate bookings. Users' selections are accurately tracked, maintaining transparency and integrity in ticket allocation.

4.1.4 Sustainability Plan

The system is **sustainable** as it can be reused, modified, or extended for larger seating arrangements or integrated with advanced features (like file save/load or graphical interfaces) in the future, without requiring major redesign.

4.2 Project Management and Team Work

Budget Analysis:

- As a software project implemented in C, the cost is minimal—mainly computer resources and development time.
- **Alternate Budget:** Using advanced IDEs or additional libraries could slightly increase cost but is optional.

• **Revenue Model:**

- Although the system is academic, it can serve as a prototype for commercial cinema ticketing, where revenue comes from ticket sales or software licensing.

• **Team Work:**

Akif Zahir	Developed and implemented the C code.
Umme Habiba Arpa	Reviewed and co-wrote sections of the report.
Nahida Rowshan	Authored the project report.
Parvej Rahim	Prepared the presentation slides.
Sajid Islam	Proofread and finalized the presentation.

4.3 Complex Engineering Problem

4.3.1 Mapping of Program Outcome

The project maps to several **Program Outcomes (POs)** as shown below:

PO's	Justification
PO1	Application of fundamental programming concepts using C.
PO2	Development of modular and maintainable software.
PO3	Analysis and solution of real-world problems using arrays and control structures.

4.3.2 Complex Problem Solving

The project addresses a beginner-level complex problem: managing a **2D seating arrangement with dynamic user interaction**.

Table 4.2: Mapping with Complex Problem Solving

EP1: Knowledge profile – Understanding of arrays, loops, and functions	EP2: Range of conflicting requirements – Handling multiple users selecting the same seat	EP3: Depth of analysis – Validating input and tracking sold/available seats.	EP4: Familiarity of issues – Managing errors in seat selection.	EP5: Application of Standards – Adherence to C programming conventions	EP6: Stakeholder Consideration – User-friendly menu design	EP7: Interdependence – Modules interact to ensure correct seat allocation ions
✓	✓					

4.3.3 Engineering Activities

Table 4.3: Mapping with Complex Engineering Activities

EA1: Range of resources – Basic computer, IDE, and C compiler	EA2: Level of Interaction – Team collaboration in coding and testing	EA3: Innovation – Modular design for easy modification	EA4: Consequences for Society & Environment – Reduces paper usage	EA5: Knowledge Application – Beginner programming skills applied effectively
✓	✓			

Chapter 5

Conclusion

This chapter summarizes the achievements of the movie ticket booking system, discusses its limitations, and highlights potential future improvements.

5.1 Summary

The movie ticket booking system is a menu-driven C program that allows users to select movies and time slots, view a seating chart, purchase and cancel tickets, and track sold and available seats using a 5×10 2D array for each movie and time slot; it successfully demonstrates modular design, input validation for seat selection, and accurate ticket summary functionality, providing a clear, beginner-level implementation of arrays, loops, functions, and conditional statements in C programming.

5.2 Limitations

While the system functions correctly, it has the following limitations:

- The system does not save seat data between sessions, so all bookings are reset when the program restarts.
- The text-based interface is simple but lacks a graphical or interactive GUI, which could improve usability.

5.3 Future Work

The system can be improved and extended in the following ways:

- Implement file save/load functionality to retain seating information between sessions.
- Develop a graphical user interface (GUI) to enhance user interaction and make the system more intuitive.more intuitive.

These future enhancements would transform the system from a beginner-level academic project into a more **practical, real-world application**.

References

- [1] Programming in ANSI C by E Balagurusamy*
- [2] The C programming language. Prentice Hall, 1988 by Dennis Ritchie*
- [3] Learn C Programming; C Programming Language – TutorialsPoint*