Project Report

On

TICKET BOOKING SYSTEM

Submitted in partial fulfilment of the requirements for the award of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE & ENGINEERING

(Artificial Intelligence & Machine Learning)

by

Ms. A.INDHU	**-**	**23WH1A6658**
Ms. L.ABHISATHVIKA	**-**	**23WH1A6659**
Ms. G.VAISHNAVI	**-**	**23WH1A6660**
Ms. B.ABHIGNA	**-**	**23WH1A6662**

Under the esteemed guidance of

Ms. S Annapoorna

Assistant Professor, CSE(AI&ML)

BVRIT HYDERABAD College of Engineering for Women

(UGC Autonomous Institution | Approved by AICTE | Affiliated to JNTUH)

(NAAC Accredited - A Grade | NBA Accredited B.Tech. (EEE, ECE, CSE and IT)

Bachupally, Hyderabad – 500090

2024-25

Department of Computer Science & Engineering

(Artificial Intelligence & Machine Learning)

BVRIT HYDERABAD COLLEGE OF ENGINEERING FOR WOMEN

(Approved by AICTE, New Delhi and Affiliated to JNTUH, Hyderabad)

Accredited by NBA and NAAC with A Grade

Bachupally, Hyderabad – 500090

2024-25

VISHNU
UNIVERSAL LEARNING
BVRIT H
Estd. 2012

CERTIFICATE

This is to certify that the Project Report entitled **“Ticket Booking System ”** is a bonafide work carried out by **Ms. A.Indhu (23WH1A6658), Ms. L.Abhisathvika (23WH1A6659), Ms. G.Vaishnavi (23WH1A6660), Ms. B.Abhigna (23WH1A6662)** in partial fulfillment for the award of
B. Tech degree in **Computer Science & Engineering (AI&ML), BVRIT HYDERABAD College of Engineering for Women, Bachupally, Hyderabad**, affiliated to Jawaharlal Nehru Technological University Hyderabad, Hyderabad under my guidance and supervision. The results embodied in the project work have not been submitted to any other University or Institute for the award of any degree or diploma

Supervisor	Head of the Department
Ms.S.Annapoorna	Dr. B. Lakshmi Praveena
Assistant Professor	HOD & Professor
Dept of CSE(AI&ML)	Dept of CSE(AI&ML)

ACKNOWLEDGEMENT

We would like to express our sincere thanks to **Dr. K. V. N. Sunitha, Principal, BVRIT HYDERABAD College of Engineering for Women**, for her support by providing the working facilities in the college

Our sincere thanks and gratitude **to Dr. B. Lakshmi Praveena, Head of the Department, Department of CSE(AI&ML), BVRIT HYDERABAD College of Engineering for Women,** for all timely support and valuable suggestions during the period of our project.

We are extremely thankful to our Internal Guide, **Ms. S.Annapoorna, Assistant Professor CSE(AI&ML), BVRIT HYDERABAD College of Engineering for Women,** for her constant guidance and encouragement throughout the project.

A.Indhu(23wh1a6658)
L.Abhisathvika(23wh1a6659)
G.Vaishnavi(23wh1a6660)
B.Abhigna (23wh1a6662)

DECLARATION

We hereby declare that the work presented in this project entitled "**Ticket Booking System** " submitted towards completion of Project work in II Year of B.Tech of CSE(AI&ML) at **BVRIT HYDERABAD College of Engineering for Women, Hyderabad** is an authentic record of our original work carried out under the guidance of **Ms. S.Annapoorna, Assistant Professor, Department of CSE(AI&ML).**

Sign with Date:

A.Indhu

(23WH1A6658)

Sign with Date:

L.Abhisathvika

(23WH1A6659)

Sign with Date:

G.Vaishnavi

(23WH1A6660)

Sign with Date:

B.Abhigna

(23WH1A6662)

TICKET BOOKING SYSTEM

AIM :

To develop a multi-threaded ticket booking system in C that simulates multiple users booking seats simultaneously, using semaphores to handle synchronization and prevent race conditions, ensuring that no two users can book the same seat at the same time.

ALGORITHM :

1 . Initialize System:

- Define the seat grid size (ROWS x COLS) and initialize all seats to 0 (indicating available seats).
- Initialize a semaphore (seat_semaphore) to manage access to shared resources (seats).

2 . Display Seat Layout:

- Print the current status of the seats, where:
  0 = Available seat
  1 = Booked seat

3 . For Each User (Thread):

Input:

- Get user input for seat row and column (coordinates of the seat the user wants to book).
- Critical Section (Access Control):
- Use sem_wait() to lock the critical section and prevent other threads from accessing the seat grid simultaneously.

4 . Seat Booking:

- Check if the selected seat is available (0).
- If available (0), change the seat status to booked (1) and display a success message.
- If not available (1), display a failure message indicating the seat is already booked.

5 . End Critical Section:

- Use sem_post() to release the lock on the semaphore.

6 . Exit:

- Once all users have booked or attempted to book their seats, display the final seat layout.
- Destroy the semaphore and free allocated memory.

PROCEDURE :

1 . Initialize System:

- Define the seat grid size (ROWS x COLS).
- Initialize a 2D array seats[ROWS][COLS] to 0 (all seats are available).
- Initialize a semaphore seat_semaphore with value 1 to manage access to seats.

2 . Display Seat Layout:

- Create a function display_seats() to print the current seat availability: 0 for available, 1 for booked.
- For Each User (Thread):
- Get user input for the seat row and column.
- Validate the input to ensure it's within grid bounds.

3 . Critical Section:

- Use sem_wait(&seat_semaphore) to lock access to the seat grid.
- Check if the selected seat is available:
- If available (0), book the seat by setting it to 1 and print a success message.
- If not available (1), print a failure message.

4 . End Critical Section:

- Use sem_post(&seat_semaphore) to release the lock.
- Repeat for All Users:
- Repeat the booking process for all users (threads).

5 . Final Output:

- Display the final seat grid showing booked and available seats.

6 . Cleanup:

- Destroy the semaphore with sem_destroy(&seat_semaphore) and free any allocated memory.

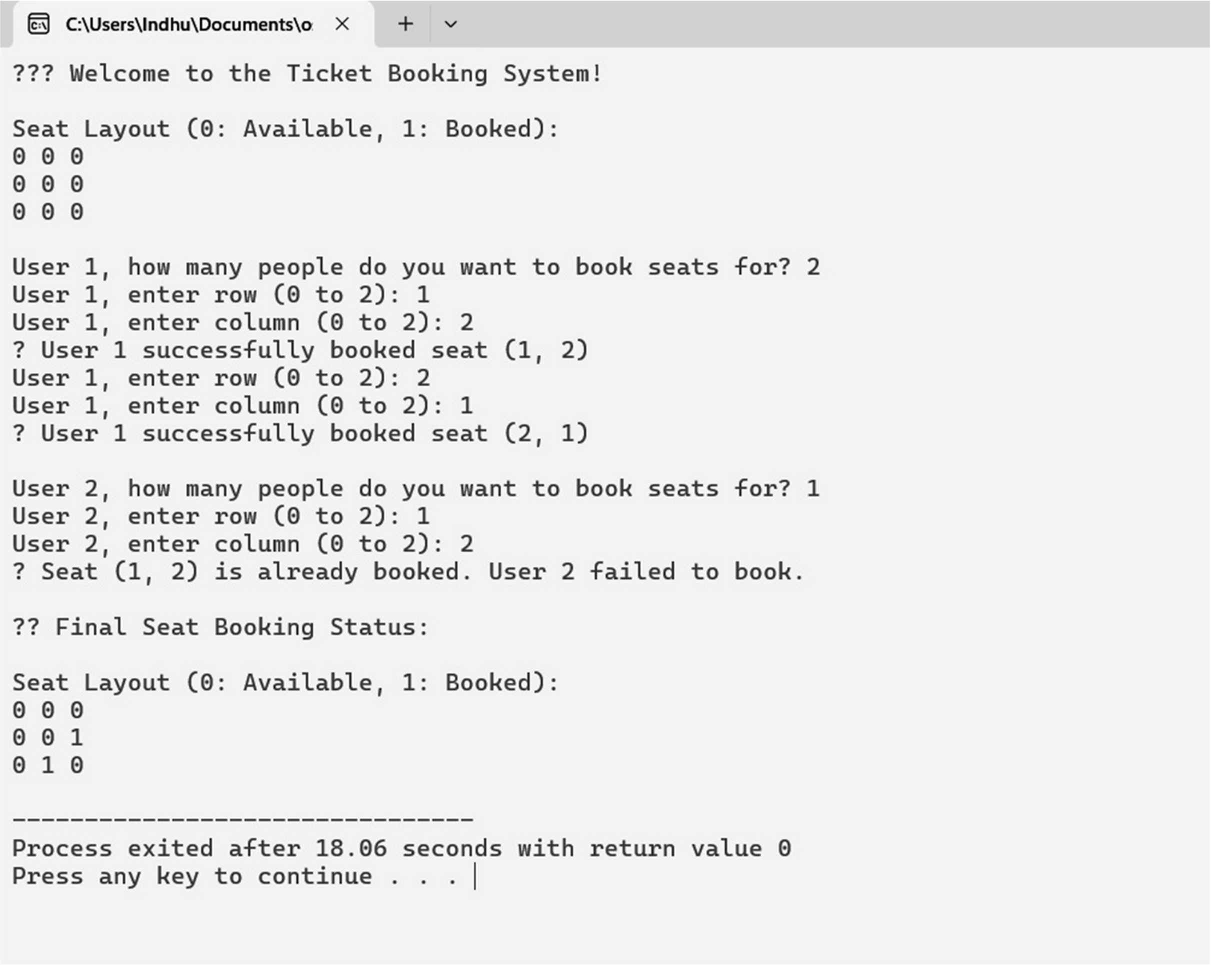
SOURCE CODE :

```
 #include <stdio.h>
#include <stdlib.h>
#include <pthread.h>
#include <unistd.h>
#include <semaphore.h>

int **seats;
int ROWS, COLS, NUM_USERS;
sem_t seat_semaphore;
int total_seats;

void display_seats() {
    int i, j;
    printf("\nSeat Layout (0: Available, 1: Booked):\n");
    for (i = 0; i < ROWS; i++) {
        for (j = 0; j < COLS; j++) {
            printf("%d ", seats[i][j]);
        }
        printf("\n");
    }
}

void* book_seat(void* arg) {
    int user_id = *((int*)arg);
    int num_people, i;
    int row, col;

    sem_wait(&seat_semaphore); // critical section begins

    if (total_seats == 0) {
        printf("\n?? All seats are already booked!\n");
        sem_post(&seat_semaphore); // critical section ends
        free(arg);
        return NULL;
    }

    // Ask the user how many people they want to book seats for
    printf("\nUser %d, how many people do you want to book seats for? ", user_id);
    scanf("%d", &num_people);

    // Check if enough seats are available
    if (num_people > total_seats) {
        printf("? Not enough seats available. You can book up to %d seats.\n", total_seats);
        sem_post(&seat_semaphore); // critical section ends
        free(arg);
        return NULL;
    }
    // Book the required number of seats for the user
    for (i = 0; i < num_people; i++) {
        printf("User %d, enter row (0 to %d): ", user_id, ROWS - 1);
        scanf("%d", &row);


      printf("User %d, enter column (0 to %d): ", user_id, COLS - 1);
        scanf("%d", &col);

        // Validate if the seat is within bounds
        if (row >= 0 && row < ROWS && col >= 0 && col < COLS) {
            if (seats[row][col] == 0) { // Check if seat is available
                seats[row][col] = 1; // Book the seat
                total_seats--; // Decrease available seats
                printf("? User %d successfully booked seat (%d, %d)\n", user_id, row, col);
            } else {
                printf("? Seat (%d, %d) is already booked. User %d failed to book.\n", row, col,
user_id);
            }
        } else {
            printf("?? Invalid seat selected by User %d\n", user_id);
        }
    }

    sem_post(&seat_semaphore); // critical section ends

    free(arg);
    return NULL;
}

int main() {
    int i, j;

    // Get user input for grid size and number of users
    printf("Enter number of rows: ");
    scanf("%d", &ROWS);
    printf("Enter number of columns: ");
    scanf("%d", &COLS);
    printf("Enter number of users: ");
    scanf("%d", &NUM_USERS);

    // Calculate total seats available
    total_seats = ROWS * COLS;

    // If the number of users exceeds the total seats, adjust users
    if (NUM_USERS > total_seats) {
        printf("?? Warning: Number of users exceeds available seats. Adjusting the number of
users to available seats (%d).\n", total_seats);
        NUM_USERS = total_seats;
    }
```

```
// Allocate memory for seats dynamically
seats = (int**)malloc(ROWS * sizeof(int*));
if (seats == NULL) {
    printf("Memory allocation failed for seats array!\n");
    exit(1);
}

for (i = 0; i < ROWS; i++) {
    seats[i] = (int*)malloc(COLS * sizeof(int));
    if (seats[i] == NULL) {

        printf("Memory allocation failed for row %d!\n", i);


        exit(1)
}

    for (j = 0; j < COLS; j++) {
        seats[i][j] = 0; // Initialize all seats to 0 (available)
    }
}

pthread_t threads[NUM_USERS];
if (sem_init(&seat_semaphore, 0, 1) != 0) {
    printf("Semaphore initialization failed!\n");
    exit(1);
}

printf("\n??? Welcome to the Ticket Booking System!\n");
display_seats();

// Create threads for each user
for (i = 0; i < NUM_USERS; i++) {
    int* user_id = malloc(sizeof(int));
    *user_id = i + 1;
    if (pthread_create(&threads[i], NULL, book_seat, user_id) != 0) {
        printf("Thread creation failed for user %d\n", i + 1);
        exit(1);
    }
}

// Wait for all threads to finish
for (i = 0; i < NUM_USERS; i++) {
    pthread_join(threads[i], NULL);
}

printf("\n?? Final Seat Booking Status:\n");
display_seats();
```

```
// Free dynamically allocated memory
 for (i = 0; i < ROWS; i++) {
    free(seats[i]);
 }
 free(seats);

 sem_destroy(&seat_semaphore);
 return 0;
```

OUTPUT:



```
??? Welcome to the Ticket Booking System!

Seat Layout (0: Available, 1: Booked):
0 0 0
0 0 0
0 0 0

User 1, how many people do you want to book seats for? 2
User 1, enter row (0 to 2): 1
User 1, enter column (0 to 2): 2
? User 1 successfully booked seat (1, 2)
User 1, enter row (0 to 2): 2
User 1, enter column (0 to 2): 1
? User 1 successfully booked seat (2, 1)

User 2, how many people do you want to book seats for? 1
User 2, enter row (0 to 2): 1
User 2, enter column (0 to 2): 2
? Seat (1, 2) is already booked. User 2 failed to book.

?? Final Seat Booking Status:

Seat Layout (0: Available, 1: Booked):
0 0 0
0 0 1
0 1 0

--------------------------------
Process exited after 18.06 seconds with return value 0
Press any key to continue . . .
```