# SEAT RESERVATION SYSTEM USING C

# INTRODUCTION

The main goal behind this project is to provide a user-friendly and efficient system for seat reservation. It enables users to select and reserve seats based on availability in various scenarios, such as transport systems, theaters, or events. The tool simplifies the process by offering an interactive interface to choose the desired seat and confirm the booking. It ensures accurate tracking of reservations and prevents double bookings, making it a practical solution for both users and organizers. This system is especially beneficial for managing seating arrangements seamlessly in real time.

# PROGRAM FLOW

## Start Program

1.Initialize total seats and seat status array.
2.Display menu options:

- View Seats
- Book a Seat
- Cancel a Seat
- Exit

1.Based on user choice:
Display seat status.
Book or cancel a seat.
Exit the program.

4.Loop back to menu until user exits

# PROGRAM EXPLANATION 1

## Initialization

- **Purpose:** The system is designed to handle seat reservation operations with
- logic and structure.
- **Variables:**
  1. totalSeats: Defines the number of seats available in the system (10 in
  2. seats[10]: Maintains the booking status of each seat using an array.
     - 0 = Available
     - 1 = Booked
- **Choice Handling:** A while loop ensures the menu keeps displaying until th
- to exit.
- **User-Friendly Menu:**
  Options are clearly defined for actions:
  1. View Seats
  2. Book a Seat
  3. Cancel a Seat
  4. Exit

```
=== Seat Reservation System ===
1. View Seats
2. Book a Seat
3. Cancel a Seat
4. Exit
==============================
Enter your choice: 1

Seat Status:
Seat 1: Available
Seat 2: Available
Seat 3: Available
Seat 4: Available
Seat 5: Available
Seat 6: Available
Seat 7: Available
Seat 8: Available
Seat 9: Available
Seat 10: Available
```

## Purpose:

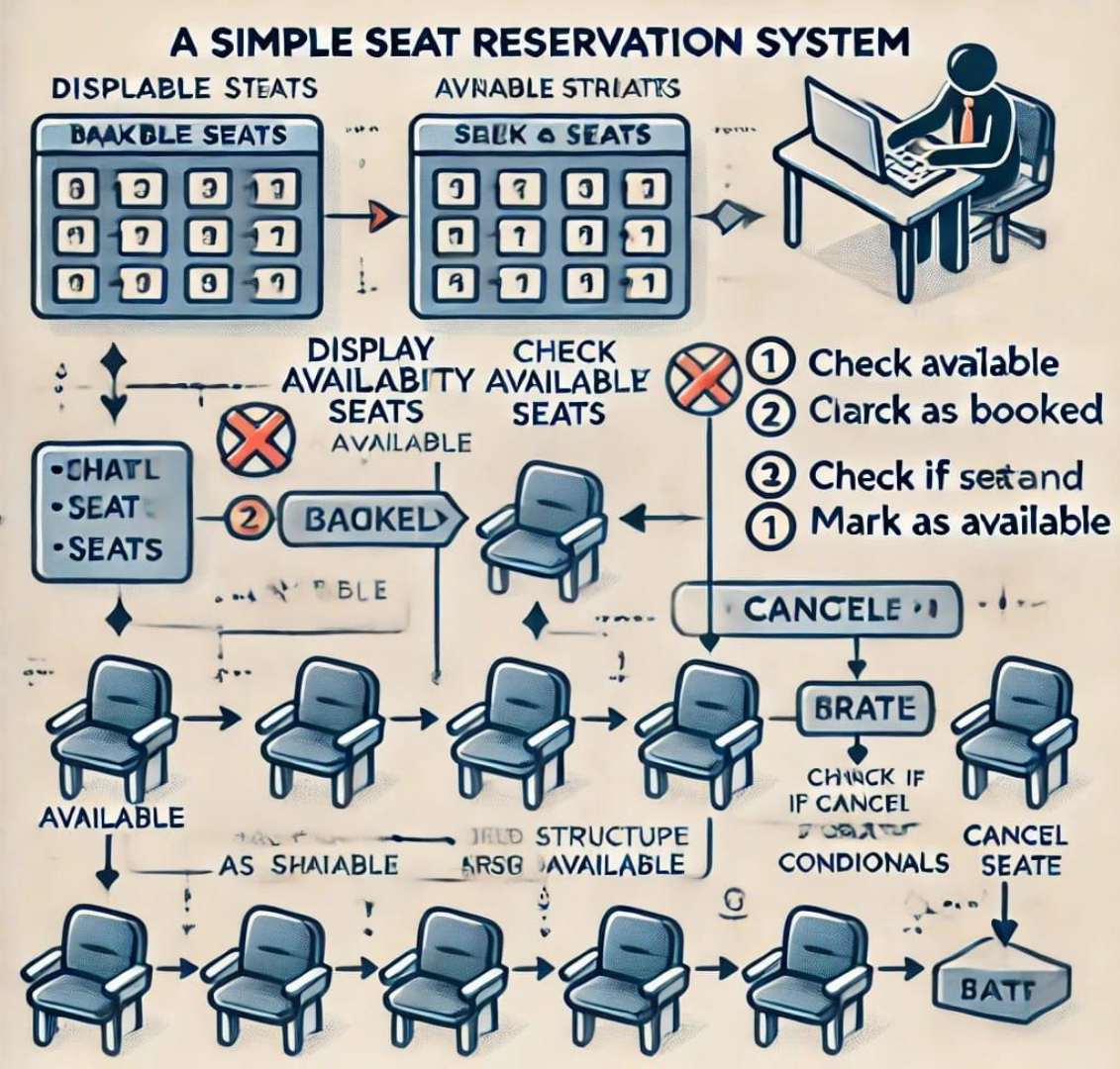
- The purpose of this project is to build a simple seat reservation system that allows users to manage bookings and cancellations efficiently. The system will help track seat availability and ensure easy management of seat reservations.

## Key Features:

- Seat Viewing: Users can check if seats are available or already booked.
- Booking Seats: Users can select and book available seats.
- Cancelling Seats: Users can cancel their seat bookings, making the seats available again.
- Simple Text-Based Interface: The system offers a user-friendly, text-based interface for easy interaction.

# Enhancement and Future Scope

1. Persistent Data Storage:Store bookings in files or databases to retain data even after the program is restarted. This ensures that bookings are saved permanently and can be retrieved later.
2. Graphical User Interface (GUI):Implement a more interactive interface using libraries like Qt or GTK, making it easier for users to interact with the system through visual elements instead of a text-based interface.
3. 3. Multi-User Support:Allow multiple users to book and cancel seats simultaneously, enabling real-time updates and bookings in a multi-user environment.4. Dynamic Pricing:Integrate seat pricing that can adjust based on demand, location, or seat type, adding flexibility and scalability to the reservation system.




Here is the visual flowchart representing the process of the seat reservation system. It shows how seats are displayed, booked, and canceled, along with how arrays are used to store the seat statuses. The flowchart also highlights key programming concepts such as modular programming, array manipulation, input validation, error handling, and decision-making using conditionals.

# FUTURE ENHANCEMENT

1. Graphical User Interface (GUI):Implement a GUI using libraries like Qt or GTK to provide a more interactive and user-friendly interface, replacing the text-based interaction with buttons, menus, and dynamic seat displays.
2. Multi-User Support:Enable simultaneous seat bookings and cancellations for multiple users, allowing real-time updates and preventing conflicts. This could be achieved using server-client architecture or multi-threading.
3. Persistent Data Storage:Integrate a database or file storage system (e.g., SQLite, CSV, or JSON files) to store seat reservation data permanently. This ensures that bookings are retained even after the program is closed or restarted.
4. Dynamic Pricing:Implement dynamic pricing based on seat demand, time, or location, allowing the system to adjust seat prices accordingly.
5. Notification System:Add email or SMS notifications for users when their booking is confirmed or canceled.These enhancements will improve the system's functionality, scalability, and user experience.

# PROGRAM EXPLANATION 2

```
=== Seat Reservation System ===
1. View Seats
2. Book a Seat
3. Cancel a Seat
4. Exit
==============================
Enter your choice: 2
Enter seat number to book (1-10): 5
Seat 5 booked successfully!
```

```
=== Seat Reservation System ===
1. View Seats
2. Book a Seat
3. Cancel a Seat
4. Exit
==============================
Enter your choice: 2
Enter seat number to book (1-10): 5
Seat 5 is already booked.
```

**Book Seat**

- **Function:** bookSeat()
- **Purpose:** Books a seat if it is available.
- **Steps:**
  1. **Input Validation:** The user inputs a seat number to book.
  2. If the seat is already booked or the number is invalid,
  3. the system shows an error.
  4. **Seat Availability Check:**
     - If the seat is available (seats[seatNumber - 1] == 0), it is marked as booked (seats[seatNumber - 1] = 1).
     - If the seat is already booked, a message informs the user.
  5. **Output:** Displays success or failure messages based on the seat status.

# PROGRAM EXPLANATION 3

```
=== Seat Reservation System ===
1. View Seats
2. Book a Seat
3. Cancel a Seat
4. Exit
==============================
Enter your choice: 3
Enter seat number to cancel (1-10): 5
Seat 5 cancellation successful!
```

**Cancel Seat**

- **Function:** cancelSeat()
- **Purpose:** Cancels a booking for a previously booked seat.
- **Steps:**
  1. **Input Validation:** The user is prompted to enter the seat number they wish to cancel. Invalid numbers or unbooked seats are flagged.
  2. **Seat Cancellation:**
     - If the seat is booked (seats[seatNumber - 1] == 1), it is marked as available (seats[seatNumber - 1] = 0).
     - If the seat is not booked, an error message is shown.
  3. **Output:** Displays success or failure messages after cancellation attempt.

# FULL PROGRAM

```c
#include <stdio.h>
#include <stdlib.h>

// Function prototypes
void displaySeats(int *seats, int totalSeats);
void bookSeat(int *seats, int totalSeats);
void cancelSeat(int *seats, int totalSeats);
void menu();

int main() {
    int totalSeats = 10; // Total number of seats
    int seats[10] = {0}; // Array to store seat status (0 = free, 1 = booked)
    int choice;

    while (1) {
        menu();
        printf("Enter your choice: ");
        scanf("%d", &choice);

```

```c
    switch (choice) {
        case 1:
            displaySeats(seats, totalSeats);
            break;
        case 2:
            bookSeat(seats, totalSeats);
            break;
        case 3:
            cancelSeat(seats, totalSeats);
            break;
        case 4:
            printf("Exiting the program. Thank you!\n");
            exit(0);
        default:
            printf("Invalid choice. Please try again.\n");
    }
}
```

```c
    return 0;
}

// Function to display seat status
void displaySeats(int *seats, int totalSeats) {
    printf("\nSeat Status:\n");
    for (int i = 0; i < totalSeats; i++) {
        printf("Seat %d: %s\n", i + 1, seats[i] == 0 ? "Available" : "Booked");
    }
    printf("\n");
}

// Function to book a seat
void bookSeat(int *seats, int totalSeats) {
    int seatNumber;
    printf("Enter seat number to book (1-%d): ", totalSeats);
    scanf("%d", &seatNumber);
```

```
55
56      if (seatNumber < 1 || seatNumber > totalSeats) {
57          printf("Invalid seat number. Please try again.\n");
58          return;
59      }
60
61      if (seats[seatNumber - 1] == 0) {
62          seats[seatNumber - 1] = 1;
63          printf("Seat %d booked successfully!\n", seatNumber);
64      } else {
65          printf("Seat %d is already booked.\n", seatNumber);
66      }
67  }
68
69  // Function to cancel a seat booking
70  void cancelSeat(int *seats, int totalSeats) {
71      int seatNumber;
72      printf("Enter seat number to cancel (1-%d): ", totalSeats);
73      scanf("%d", &seatNumber);
74
```

```
75      if (seatNumber < 1 || seatNumber > totalSeats) {
76          printf("Invalid seat number. Please try again.\n");
77          return;
78      }
79
80      if (seats[seatNumber - 1] == 1) {
81          seats[seatNumber - 1] = 0;
82          printf("Seat %d cancellation successful!\n", seatNumber);
83      } else {
84          printf("Seat %d is not booked yet.\n", seatNumber);
85      }
86  }
87
88  // Function to display the menu
89  void menu() {
90      printf("\n=== Seat Reservation System ===\n");
91      printf("1. View Seats\n");
92      printf("2. Book a Seat\n");
93      printf("3. Cancel a Seat\n");
94      printf("4. Exit\n");
95      printf("==============================\n");
96  }
```