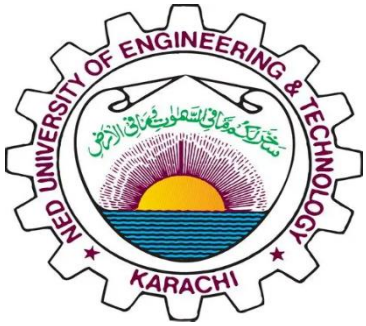


CCP-Report

PF (CT-175)

Group Members:

- Eshba (CT-25052)
- Noman Khursheed (CT-25093)

Discipline: BCIT

Teachers:

- Sir Abdullah

Table of Contents

Project Title

Byte Rails: A Railway Reservation System

Abstract

This report presents a command-line railway reservation system, "Byte Rails," developed in C. The system is designed to manage basic railway booking operations, including booking seats, canceling seats, and displaying the status of all available seats. It utilizes core C programming concepts such as arrays for data storage, standard input/output functions for user interaction, and conditional logic to handle different user choices. The system demonstrates fundamental programming principles in a practical, menu-driven application.

1. Introduction

Managing railway reservations efficiently is a common logistical challenge. Manual systems are prone to errors and are inefficient. Automating this process, even at a basic level, can significantly improve accuracy and user experience. The objective of this project is to develop a robust, menu-driven C program that simulates a railway reservation system. This system allows a user to perform the three primary functions of a booking system: booking a new seat, canceling an existing booking, and viewing the availability of all seats, distinguishing between Business and Economy classes.

2. Technical Terms and Concepts

- **`#include <stdio.h>`:** Standard input/output library used for functions like `printf`, `fgets`, and `sscanf`.
- **`#include <string.h>`:** String library providing functions for string manipulation, such as `strcasecmp` (case-insensitive comparison) and `strcspn` (find first character in a set).
- **Arrays:** The core data structure used to store the booking status of all seats. `businessSeats[20]` and `economySeats[80]` use an integer (`0` or `1`) to

represent 'Empty' or 'Booked'.

- **`while` Loop:** The main program loop that keeps the menu running and allows the user to perform multiple actions until they choose to exit.
- **`switch` Statement:** The primary control structure used to navigate the user's menu choice (1-4) and execute the corresponding functionality.
- **`fgets` / `sscanf`:** A secure input-handling pair. `fgets` reads a full line of text, preventing buffer overflow errors. `sscanf` is then used to parse the required data (like an integer) from that string, allowing for robust input validation.
- **Boolean Flag:** A variable (`int running = 1`) is used to control the state of the main `while` loop. Setting it to `0` terminates the program.

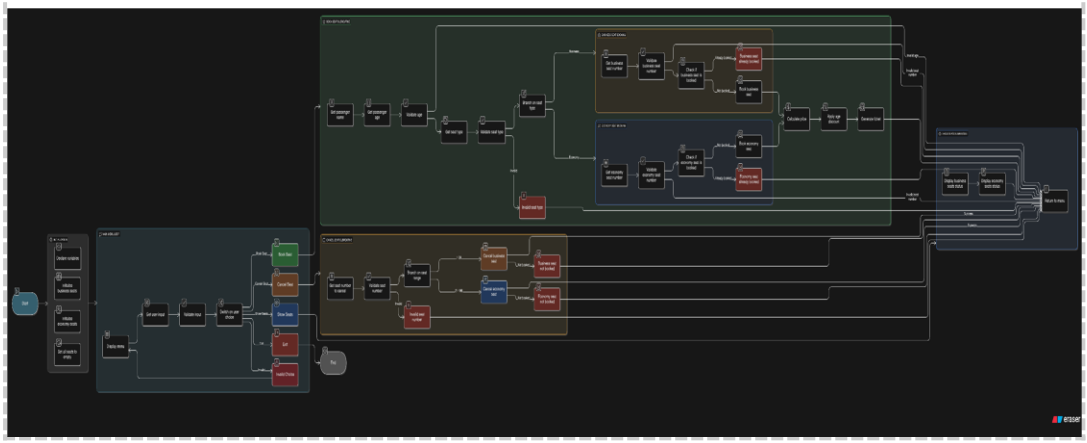
3. System Design and Implementation

3.1 Data Structures

- **`int businessSeats[20]`:** An integer array representing 20 Business class seats. The value `0` indicates the seat is 'Empty', and `1` indicates it is 'Booked'.
- **`int economySeats[80]`:** An integer array for 80 Economy class seats, following the same `0` (Empty) / `1` (Booked) logic.
- **Local Variables:** Various variables are used within the `main` function to store temporary data, such as `name` (passenger name), `age` (for discount calculation), `seatType`, `price`, and `choice` (menu selection).

3.2 Program Flow

The program initializes all seats in both arrays to `0` (Empty). It then enters a `while` loop that displays the main menu. The user's choice is captured and validated. A `switch` statement directs the program to the correct block of code.



3.3 Functional Modules (Case Logic)

- **Case 1: Book Seat**

  Prompts the user for name, age, and seat type. Validates seat type, then prompts for a seat number (1-20 for Business, 21-100 for Economy). If the seat is available, it's marked as booked, price is calculated (with discounts), and a ticket is printed.

- **Case 2: Cancel Seat**

  Prompts for a seat number (1-100). Checks the appropriate array (Business or Economy) to see if the seat is booked. If yes, it sets the status back to `0` (Empty) and confirms. Otherwise, it reports the seat was not booked.

- **Case 3: Show Seats**

  Iterates through both `businessSeats` and `economySeats` arrays, printing the status ("Booked" or "Empty") for each seat number.

- **Case 4: Exit**

  Sets the `running` flag to `0`, which terminates the main `while` loop, and the program ends.

4. Code

```
#include <stdio.h>
#include <string.h>
#include <strings.h> // For strcasecmp

int main() {
```

```
int businessSeats[20]; // 20 business class seats
int economySeats[80]; // 80 economy class seats
int i, choice, s, age;
char name[50], seatType[10];
float price;
int running = 1; // flag
char input[10]; // buffer to hold raw input as a string

// make all seats empty
for(i = 0; i < 20; i++) {
    businessSeats[i] = 0;
}
for(i = 0; i < 80; i++){
    economySeats[i] = 0;
}

while(running) {
    printf("\n--- Railway Reservation Menu ---\n");
    printf("\n1. Book Seat\n2. Cancel Seat\n3. Show Seats\n4.
    printf("Enter your choice: ");

    fgets(input, sizeof(input), stdin); // reads the entire l
    if(sscanf(input, "%d", &choice) != 1) { // checks if a va
        printf("Invalid input. Please enter a number.\n");
        continue;
    }

    switch(choice){
        case 1:
            printf("Enter passenger name: \n");
            fgets(name, sizeof(name), stdin);
            name[strcspn(name, "\n")] = '\0'; // remove newli

            printf("Enter passenger age: ");
            fgets(input, sizeof(input), stdin);
            if(sscanf(input, "%d", &age) != 1 || age < 0 || a
                printf("Invalid age.\n");
                break;
            }

            printf("Enter seat type (Business/Economy): \n");
            fgets(seatType, sizeof(seatType), stdin);
            seatType[strcspn(seatType, "\n")] = '\0'; // remo

            if(strcasecmp(seatType,  "Business")==0){
```

```
        printf("Enter seat number (1-20): ");
        fgets(input, sizeof(input), stdin);
        if(sscanf(input, "%d", &s) != 1 || s < 1 || s
            printf("Invalid seat number.\n");
            break;
        }
        s = s - 1; // converting to index

        if(businessSeats[s] == 0){
            businessSeats[s] = 1;
            price = 3000;
        } else{
            printf("Seat %d has already been booked."
            break;
        }

    } else if(strcasecmp(seatType, "Economy") == 0) {
        printf("Enter seat number (21-100): ");
        fgets(input, sizeof(input), stdin);
        if(sscanf(input, "%d", &s) != 1 || s < 21 ||
            printf("Invalid seat number.\n");
            break;
        }
        s = s - 21; // converting to index

        if(economySeats[s] == 0){
            economySeats[s] = 1;
            price = 1500;
        } else{
            printf("Seat %d has already been booked.\
            break;
        }
    } else{
        printf("Invalid seat type.\n");
        break;
    }

    // age-based discount
    if(age < 12){
        price = price * 0.5; // 50% discount
    } else if(age >= 60){
        price = price * 0.7; // 30% discount
    }

    printf("Seat %d booked successfully!\n", (strcase
```

```
        // Print Ticket
        printf("\n+----------------------------------------------------------------------------
        printf("| RAILWAY TICKET |\n");
        printf("+----------------------------------------------------------------------------
        printf("| Passenger Name : %-24.24s |\n", name);
        printf("| Age : %-24d |\n", age);
        printf("| Seat Type : %-24s |\n", seatType);
        printf("| Seat Number : %-24d |\n", (strcasecmp(s
        printf("| Booking Status : %-24s |\n", "Confirmed
        printf("| Ticket Price : %-24.2f |\n", price);
        printf("|----------------------------------------------------------------------------
        printf("| Thank you for booking with us |\n");
        printf("+----------------------------------------------------------------------------

        printf("Press Enter to return to the menu...");
        fgets(input, sizeof(input), stdin); // Pause
        break;

    case 2:
        printf("Enter seat number to cancel (1-100): ");
        fgets(input, sizeof(input), stdin);
        if(sscanf(input, "%d", &s) != 1 || s < 1 || s > 1
            printf("Invalid input. Please enter a number
            break;
        }

        if(s >= 1 && s <= 20) {
            if(businessSeats[s - 1] == 1) {
                businessSeats[s - 1] = 0;
                printf("Seat %d cancelled successfully.\n
            } else {
                printf("Seat %d was not booked.\n", s);
            }
        } else { // Seat 21-100
            if(economySeats[s - 21] == 1) {
                economySeats[s - 21] = 0;
                printf("Seat %d cancelled successfully.\n
            } else {
                printf("Seat %d was not booked.\n", s);
            }
        }
        break;

    case 3:
```

```
                printf("\nBusiness class seats status:\n");
                for(i = 0; i < 20; i++) {
                    if(businessSeats[i] == 1){
                        printf("Seat %d: Booked\n", i + 1);
                    } else{
                        printf("Seat %d: Empty\n", i + 1);
                    }
                }

                printf("\nEconomy Class Seats:\n");
                for(i = 0; i < 80; i++){
                    if(economySeats[i] == 1){
                        printf("Seat %d: Booked\n", i + 21);
                    } else{
                        printf("Seat %d: Empty\n", i + 21);
                    }
                }
                break;

            case 4:
                printf("Thank you for using the Railway Reservati
                running = 0; // Exit the loop
                break;

            default:
                printf("Invalid choice. Please select a valid opt
                break;
        }
    }

    return 0;
}
```

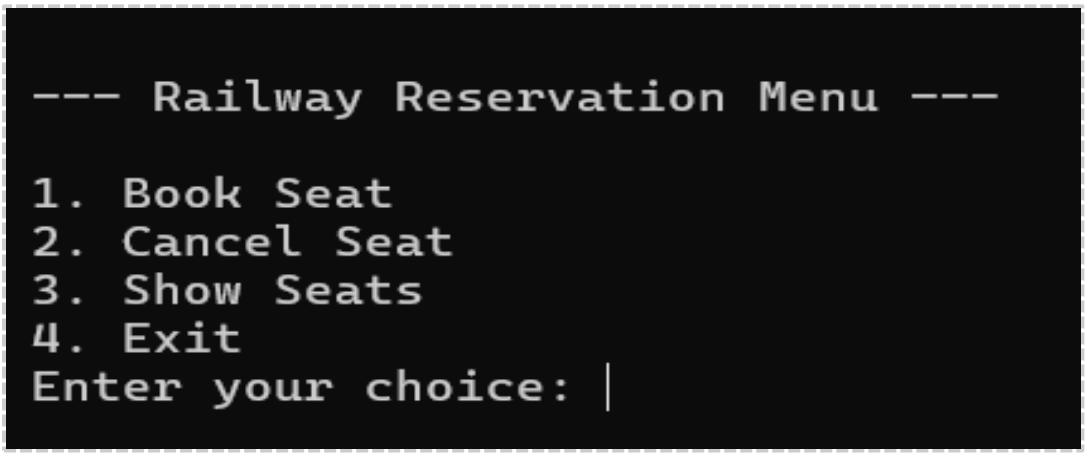
5. Sample Program Execution

The following screenshots demonstrate the program's functionality.

Main Menu (1000140458.png)

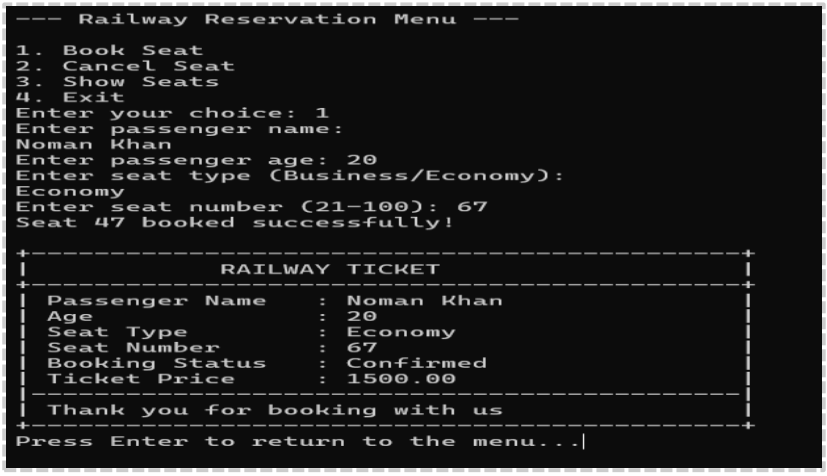
This shows the program's starting state. The main menu is displayed with four options: Book, Cancel, Show Seats, and Exit. The program is waiting for the user to "Enter your

choice:".



Booking an Economy Seat (1000140459.png)

Here, the user selected '1' to book. They entered the name "Noman Khan", age "20", and type "Economy". They chose seat "67". The system successfully booked it and generated a ticket showing the price of 1500.00 (no discount applied).

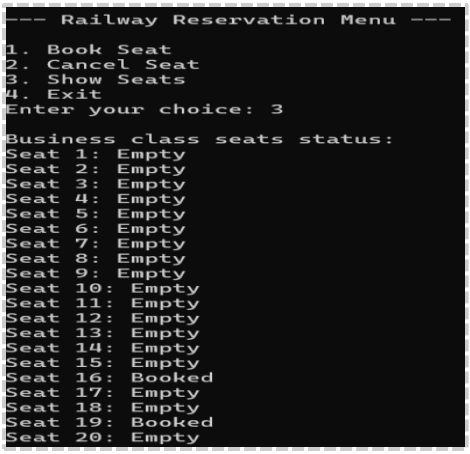


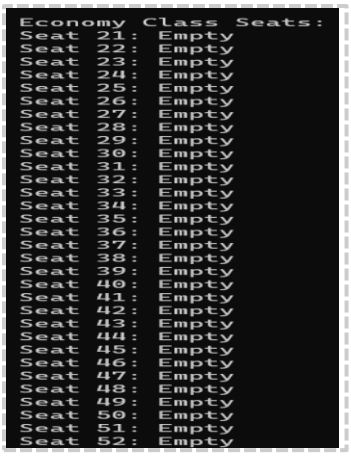
Booking a Business Seat (1000140460.png)

This snippet shows another booking. The user entered "Eshba", age "88", and type "Business", choosing seat "19". The system correctly applied a senior discount (30% off 3000.00), resulting in a final price of 2100.00, and printed the ticket.

Showing Seat Status (1000140461.png & 1000140462.png)

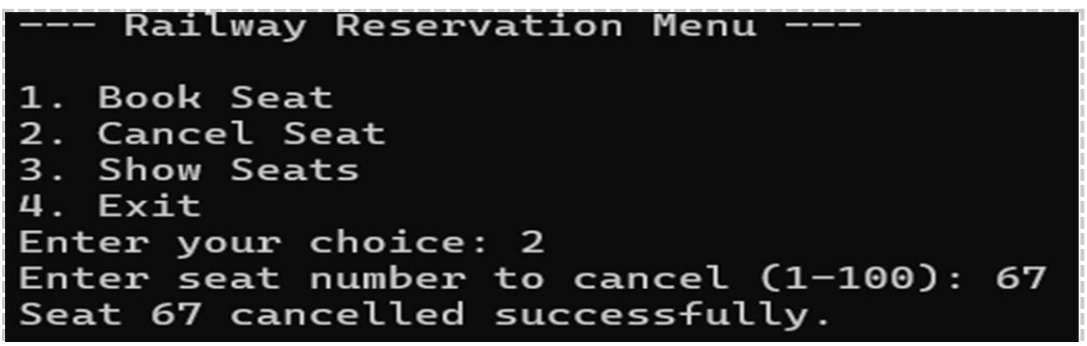
The user selected option '3'. The program first lists all Business class seats. Snippet 1000140461.png shows "Seat 19: Booked" (from Eshba's booking), while others are "Empty". Snippet 1000140462.png shows the start of the Economy list, where "Seat 67: Booked" (from Noman's booking) would appear further down.





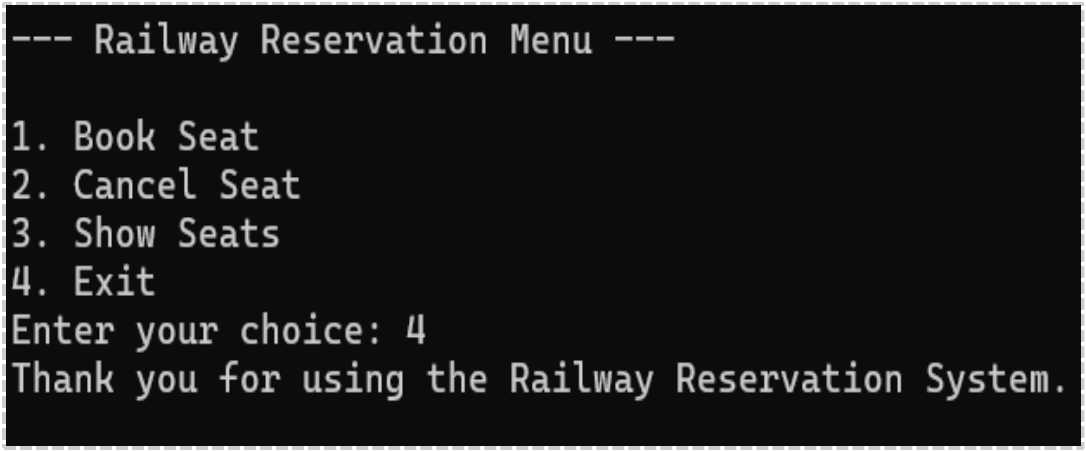
Canceling a Seat (1000140463.png)

The user chose option '2' to cancel. They entered seat number "67". The system found that this seat was booked (Noman's Economy seat) and successfully cancelled it, freeing the seat in the array.



Exiting the Program (1000140464.png)

Finally, the user selected option '4'. The program printed the exit message "Thank you for using the Railway Reservation System." and then terminated, as the `running` flag was set to 0.



6. Assumptions

- **Non-Persistent Data:** The reservation data is stored in arrays in memory. All booking information is lost when the program exits.
- **Fixed Capacity:** The system is hard-coded with a fixed capacity of 20 Business and 80 Economy seats.
- **User Input:** The system assumes users will generally follow prompts. While robust validation is in place for numeric inputs (`sscanf`), it's assumed the user won't try to maliciously break the program.
- **Simple Pricing:** The pricing model is basic, with fixed rates for two classes and two simple age-based discount rules.

7. Future Work

- **Data Persistence:** Implement file I/O using `fopen`, `fwrite`, and `fread` to save the booking status of the `businessSeats` and `economySeats` arrays to a file. This file would be read on startup, making bookings persistent.

- **Passenger-Specific Data:** Store the passenger's name and other details *with* the seat, allowing for cancellation by name or viewing a specific passenger's ticket.
- **Dynamic Seat Map:** Instead of a simple text list, create a function to display a visual grid of seats (e.g., `[B]` for booked, `[E]` for empty).
- **Graphical User Interface (GUI):** Rebuild the application's front-end using a library like GTK, Qt, or a simpler C-based GUI library for a more user-friendly, mouse-driven experience.

8. Conclusion

This project successfully demonstrates the creation of a functional, text-based railway reservation system in C. The "Byte Rails" program meets all core objectives, providing modules for booking, canceling, and viewing seats using fundamental C programming constructs like arrays, loops, and conditional logic. The program is modular in its `switch-case` design and uses secure input handling (`fgets`/`sscanf`), making it a robust console application. It serves as a strong foundation for a more complex system, with clear paths for future enhancements such as data persistence and a graphical interface.