Bus Ticket Booking System

By- Mani Aggarwal
Sap ID-590027423

December 2, 2025

Abstract

This project titled “Bus Ticket Booking system” is a simple console -based application developed in C.
The main aim is to provide a basic system where users can check available buses, book tickets, view bookings, and cancel tickets. The project. Shows the use of structures, file handling, arrays, functions, and conditional statements. It helps students understand how real life systems like ticket booking can be implemented using logic and programming.

Introduction

The Bus Ticket Booking System is a small C language project that helps in booking and managing bus tickets in an easy and organised way. In daily life, bus travel is one of the most common modes of transportation, and many people use buses for going to school, work, and long-distance trips. Managing bus tickets manually often becomes confusing because it takes time to check seats, write passenger details, and update records. To reduce this difficulty, a computer-based system is needed.

This project provides a simple menu-driven program where the user can book a ticket, view booked tickets, and cancel a ticket. It uses basic C programming concepts like functions, structures, arrays, loops, conditional statements, and file handling. These features help the system store ticket data safely and allow the user to perform actions quickly.

When a user selects the booking option, the system asks for details like passenger name, bus number, seat number and travel date. The system stores all this information in a file so that the data remains saved even after closing the program. The user can also check all booked tickets, which helps in keeping track of passengers. If a passenger wants to cancel a ticket, the system deletes that entry from the file and updates the records.

This project is helpful for beginners because it explains how real-life systems work in a computerised way. It shows how data can be stored, searched and managed using simple C programming. Overall, the Bus Ticket Booking System is a useful and practical project that demonstrates the application of programming in solving

1. Problem Definition

1.1. Overview

1. The Bus Ticket Booking System is designed to replace manual ticket booking with a simple computerized system.

2. Manual booking often leads to problems like errors, missing records, slow processing, and difficulty in checking seat availability.

3. A digital system makes ticket management faster, more accurate, and more organized.

4. The project uses C programming concepts such as functions, structures, loops, and file handling.

5. The system allows users to book tickets, view booked tick✓ and cancel tickets easily.

6. File handling ensures that all passenger records remain saved even after closing the program.

7. This project shows how a basic real-world booking system can be created using simple logic and programming skills.

1.2. Objectives

1. To automate the bus ticket booking process.
2. To provide a simple menu-driven interface for easy operations.
3. To store passenger records safely using file handling.
4. To reduce manual errors and increase accuracy.
5. To save time by offering fast booking and cancellation.
6. To help students understand real-life applications of C programming.

2. System Design

2.1. System Architecture

The Bus Ticket Booking System follows a three-layer architecture. This structure helps divide the program into smaller parts so that each part has a clear role. The three layers are:

A. Presentation Layer

- This is the top layer of the system. It is responsible for interacting with the user.
- Functions of Presentation Layer:
  - Displays the main menu with options like Book Ticket, View Tickets, Cancel Ticket, Exit, etc.
  - Takes input from the user such as passenger name, seat number, bus number, and travel date.
  - Shows the output on the screen (booking confirmation, ticket list, cancellation message).
  - Ensures that the program is easy to use and understand.

In this project, the presentation layer is built using printf() and scanf() statements in C.

B. Business Logic Layer

•This is the middle layer of the architecture. It contains the main logic and rules of the system.
•Functions of Business Logic Layer:
- Processes the user input received from the presentation layer.
- Contains functions like:
bookTicket()
viewTickets()
cancelTicket()
•Verifies conditions such as:
- Whether a seat is already booked
- Whether the entered bus number is valid
- Whether the cancellation record exists
- Uses structures to manage ticket data in an organized way.
- Controls the flow of the program and decides what should happen next.

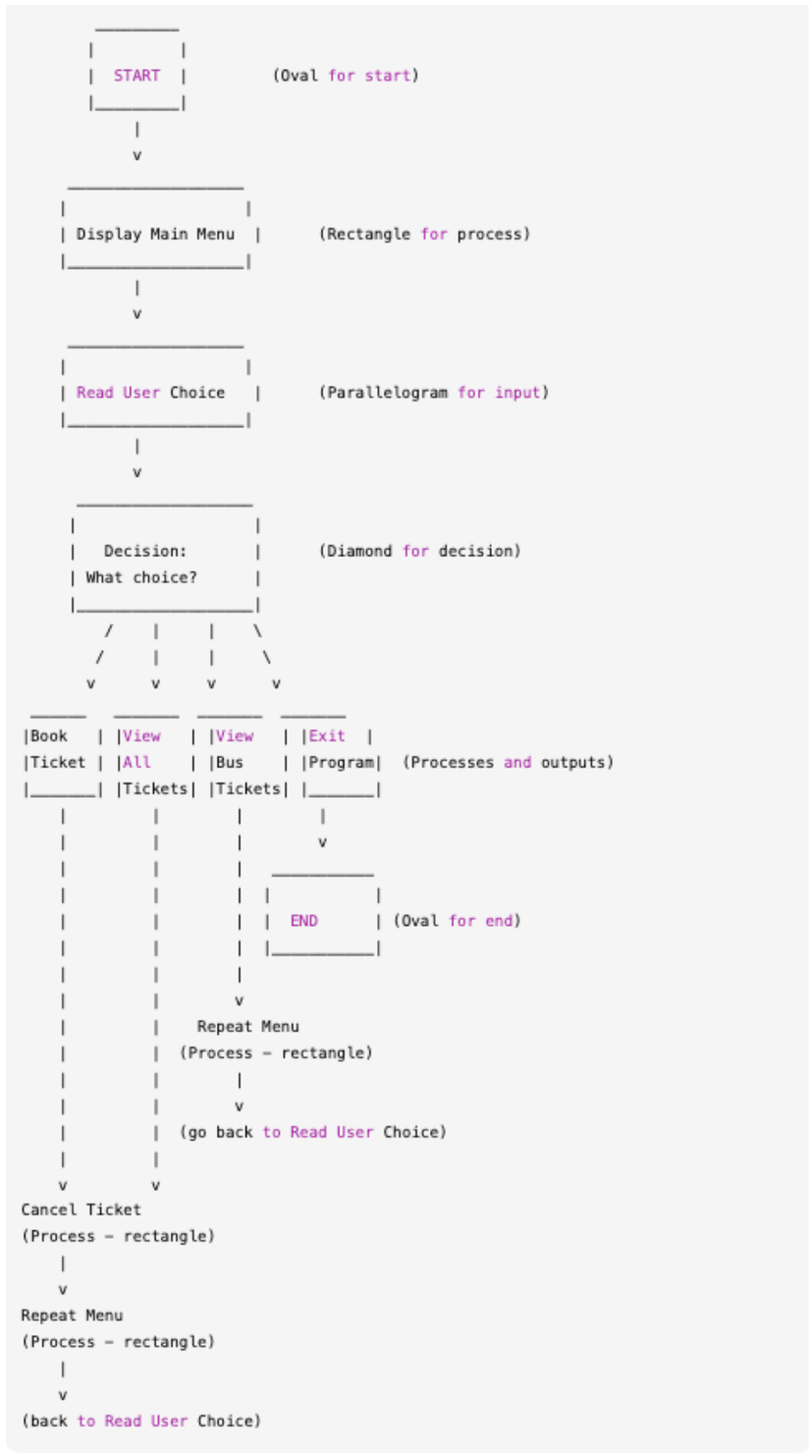
C. Data Access Layer

•This is the bottom layer of the architecture.
•It is responsible for storing and retrieving data.
•Functions of Data Access Layer:
- Uses file handling (fopen, fread, fwrite, fprintf, etc.) to store ticket details.
- Saves ticket information permanently in a file (e.g., "tickets.dat" or “tickets.txt").
- Fetches all stored details when the user wants to view booked tickets.
•Updates the file when a ticket is cancelled or modified.
•Ensures data does not get lost even after closing the program.
•This layer makes the project work like a real booking system because it stores all records safely.

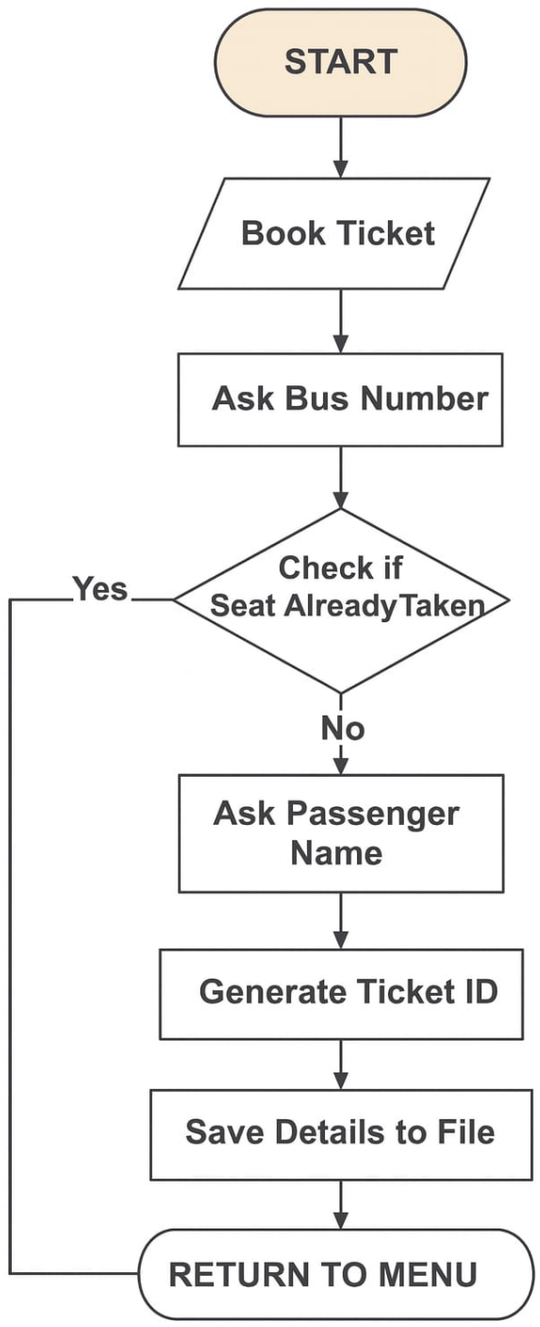
---

3. Flowchart

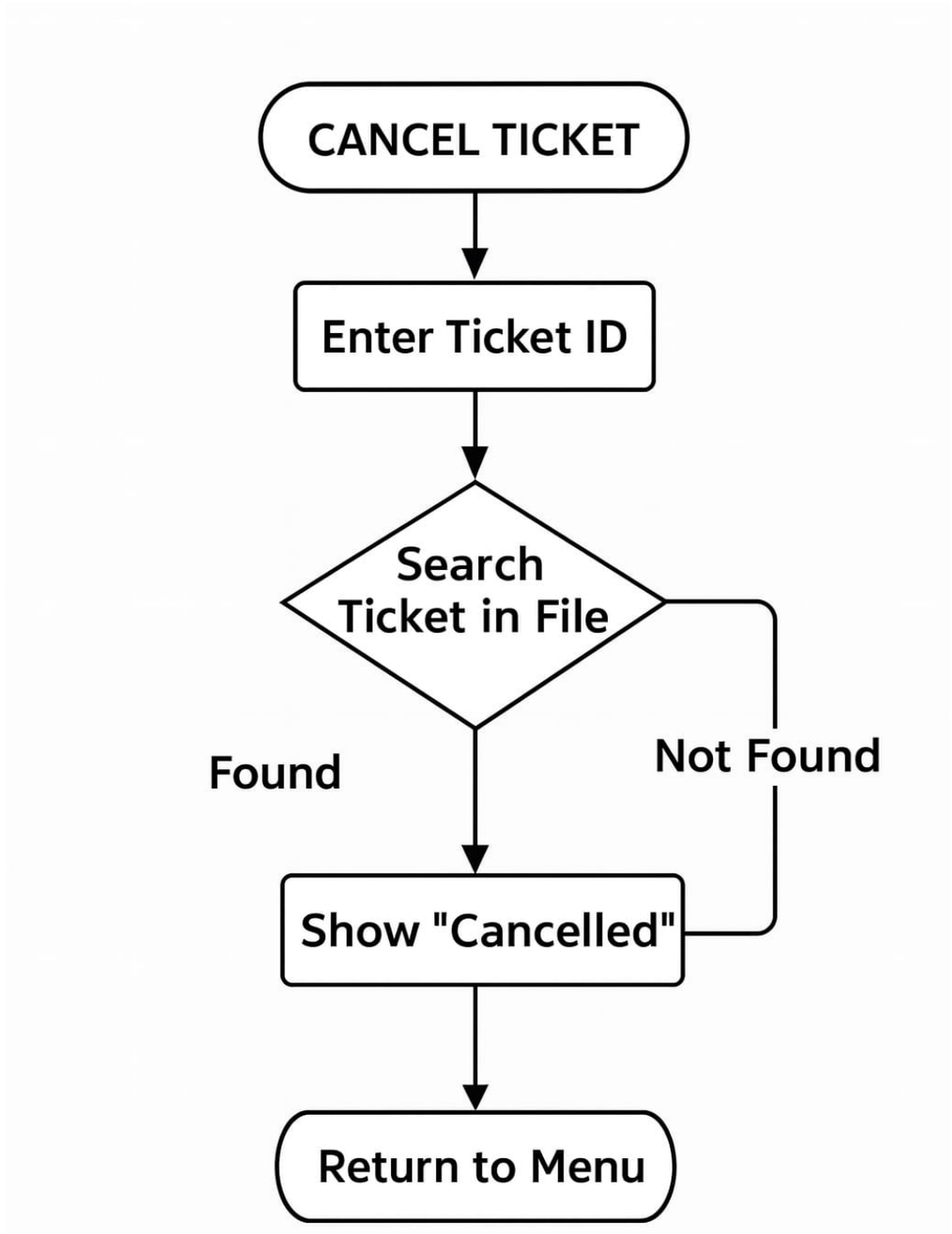
3.1. Main Menu Flowchart



3.2. Book Ticket Flowchart



3.3. Cancel Ticket Flowchart



4. Algorithms

1. **Book Ticket**

- Input Bus No, Seat No, Passenger Name.
- Check if seat is already booked → If yes, show error.
- Generate Ticket ID, save details to file, display confirmation.

2. **View Tickets**

- **All Tickets:** Read and display all records from file.
- **By Bus:** Display only tickets matching entered Bus Number.

3. **Cancel Ticket**

- Input Ticket ID → Copy all other records to a temp file.
- Replace original file with temp → Show success/failure message.

4. **Generate Ticket ID**

- Scan all records in file → Find max ID → Return max ID + 1.

---

5. Data Structures

In this project, data structures are used to store, organize, and manage the ticket information in a proper way. The main data structure used in the Bus Ticket Booking System is the structure (struct) in C language.

5.1. struct Ticket (User defined Data Structure)

To store each ticket as one unit, we use a structure named Ticket.

Structure Definition

A structure is a user-defined data type that allows us to group different types of information in one place.

In this project, each ticket has four fields:

```
typedef struct {
    int id;                 // unique ticket ID
    int busNo;              // bus number
    char name[50];          // passenger name
    int seatNo;             // seat number
} Ticket;
```

Purpose of each field:

Field	Type	Purpose
id	int	Assigns a unique ID to every ticket
busNo	int	Stores the bus number for which booking is done
name	char array	Stores passenger name
seatNo	int	Stores which seat is booked

This structure helps store all related information about a ticket in one combined record.

---

6. Implementation Details

6.1. Key Features

1. Ticket Booking

- Takes bus number, seat number, and passenger name.
- Checks if a seat is already booked.
- Generates an automatic ticket ID.
- Saves details into a binary file.

2. View All Tickets

- Displays all bookings stored in the file.
- Shows ID, Seat No, Bus No, Name in a tabular format.

3. View Tickets for a Particular Bus

- User enters a bus number.
- Only tickets of that bus are shown.

4. Cancel Ticket

- Ticket search is done by Ticket ID.
- If found, the record is deleted from file using a temporary file.
- Displays message after cancellation.

5. Unique ID Generation

- Automatically finds the next available ticket ID by scanning previous records.

6. File Handling for Permanent Storage

- Uses `"tickets.dat"` file to store all tickets.
- Data is not lost even after program closes.

6.2. Important Code Snippets

A). Structure used for tickets

```c
typedef struct {
    int id;
    int busNo;
    char name[50];
    int seatNo;
} Ticket;
```

B). Function to generate next ticket id

```c
int getNextTicketID() {
    FILE *fp = fopen("tickets.dat", "rb");
    int maxid = 0;

    if (fp == NULL)
        return 1;

    Ticket t;
    while (fread(&t, sizeof(Ticket), 1, fp)) {
        if (t.id > maxid)
            maxid = t.id;
    }

    fclose(fp);
    return maxid + 1;
}
```

C). Checking if seat already booked

```c
int seatTaken(int busNo, int seatNo) {
    FILE *fp = fopen("tickets.dat", "rb");
    if (fp == NULL) return 0;

    Ticket t;
    while (fread(&t, sizeof(Ticket), 1, fp)) {
        if (t.busNo == busNo && t.seatNo == seatNo) {
            fclose(fp);
            return 1;
        }
    }

    fclose(fp);
    return 0;
}
```

D) Booking a ticket

```c
void bookTicket() {
    Ticket t;

    printf("Enter Bus Number: ");
    scanf("%d", &t.busNo);

    printf("Enter Seat Number: ");
    scanf("%d", &t.seatNo);

    if (seatTaken(t.busNo, t.seatNo)) {
        printf("Seat already booked!\n");
        return;
    }

    printf("Enter Passenger Name: ");
    scanf("%s", t.name);

    t.id = getNextTicketID();

    FILE *fp = fopen("tickets.dat", "ab");
    fwrite(&t, sizeof(Ticket), 1, fp);
    fclose(fp);

    printf("Ticket booked successfully!\n");
}
```

E). Viewing all tickets

```c
void viewAllTickets() {
    FILE *fp = fopen("tickets.dat", "rb");

    if (fp == NULL) {
        printf("No bookings found.\n");
        return;
    }

    Ticket t;
    printf("ID   Bus   Seat   Name\n");

    while (fread(&t, sizeof(Ticket), 1, fp)) {
        printf("%d   %d   %d   %s\n", t.id, t.busNo, t.seatNo, t.name);
    }

    fclose(fp);
}
```

F). Cancel Ticket using id

```c
void cancelTicket() {
    int id;
    printf("Enter Ticket ID to cancel: ");
    scanf("%d", &id);

    FILE *fp = fopen("tickets.dat", "rb");
    FILE *temp = fopen("temp.dat", "wb");

    Ticket t;
    int found = 0;

    while (fread(&t, sizeof(Ticket), 1, fp)) {
        if (t.id == id)
            found = 1;
        else
            fwrite(&t, sizeof(Ticket), 1, temp);
    }

    fclose(fp);
    fclose(temp);

    remove("tickets.dat");
    rename("temp.dat", "tickets.dat");

    if (found)
        printf("Ticket cancelled successfully.\n");
    else
        printf("Ticket ID not found.\n");
}
```

7. Testing and Results

7.1. Test Cases

Test Case 2 – Booking an Already Taken Seat

Input:

- Bus Number: 101
- Seat Number: 5 (already booked)

Expected Output:

- "Seat already booked"

Test Case 3 – Viewing All Tickets

Input:

- Choose option 2 (View All Tickets)

Expected Output:

- All tickets should be displayed in tabular form.

Test Case 4 – Viewing Tickets for a Particular Bus

Input:

- Bus Number: 101

Expected Output:

- Only tickets belonging to Bus 101 should appear.

Test Case 5 – Cancel Ticket (Existing ID)

Input:

- Ticket ID: 1

Expected Output:

- “Ticket cancelled successfully”
- Ticket record should be removed from the file

Test Case 6 – Cancel Ticket (Wrong / Non-Existing ID)

Input:

- Ticket ID: 999

Expected Output:

- “Ticket ID not found”

Test Case 7 – Invalid Menu Input

Input:

- Alphabet or symbol instead of number

Expected Output:

- “Invalid choice” message
- Program should not crash

8. Outputs

Booking Ticket

```
vboxuser@Ubuntu1404:~/Bus1$ gcc *.c -o Bus1
vboxuser@Ubuntu1404:~/Bus1$ ./Bus1

===== BUS TICKET BOOKING SYSTEM =====
1. Book Ticket
2. View All Tickets
3. View Tickets of a Bus
4. View by Destination
5. Cancel Ticket
6. Exit
Enter choice: 1

--- Available Buses ---
1. Bus No 101 -> Delhi (Fare: 500)
2. Bus No 102 -> Jaipur (Fare: 450)
3. Bus No 103 -> Chandigarh (Fare: 400)
4. Bus No 104 -> Mumbai (Fare: 900)
Enter Bus Number: 101
Enter Seat Number: 2
Enter Passenger Name (no spaces): Mani
Ticket Booked! Your Ticket ID = 2
```

Viewing all tickets

```
===== BUS TICKET BOOKING SYSTEM =====
1. Book Ticket
2. View All Tickets
3. View Tickets of a Bus
4. View by Destination
5. Cancel Ticket
6. Exit
Enter choice: 2

--- All Tickets ---
ID:1  Bus:101  Seat:1  Name:Ohmsai  Dest:Delhi  Fare:500
ID:2  Bus:101  Seat:2  Name:Mani  Dest:Delhi  Fare:500
```

View Tickets Of A Bus

```
===== BUS TICKET BOOKING SYSTEM =====
1. Book Ticket
2. View All Tickets
3. View Tickets of a Bus
4. View by Destination
5. Cancel Ticket
6. Exit
Enter choice: 3
Enter Bus Number: 101

--- Tickets for Bus 101 ---
ID:1  Seat:1  Name:Ohmsai  Dest:Delhi  Fare:500
ID:2  Seat:2  Name:Mani  Dest:Delhi  Fare:500
```

View By Destination

```
===== BUS TICKET BOOKING SYSTEM =====
1. Book Ticket
2. View All Tickets
3. View Tickets of a Bus
4. View by Destination
5. Cancel Ticket
6. Exit
Enter choice: 4
Enter Destination (no spaces): Delhi

--- Tickets going to Delhi ---
ID:1  Bus:101  Seat:1  Name:Ohmsai  Fare:500
ID:2  Bus:101  Seat:2  Name:Mani  Fare:500
```

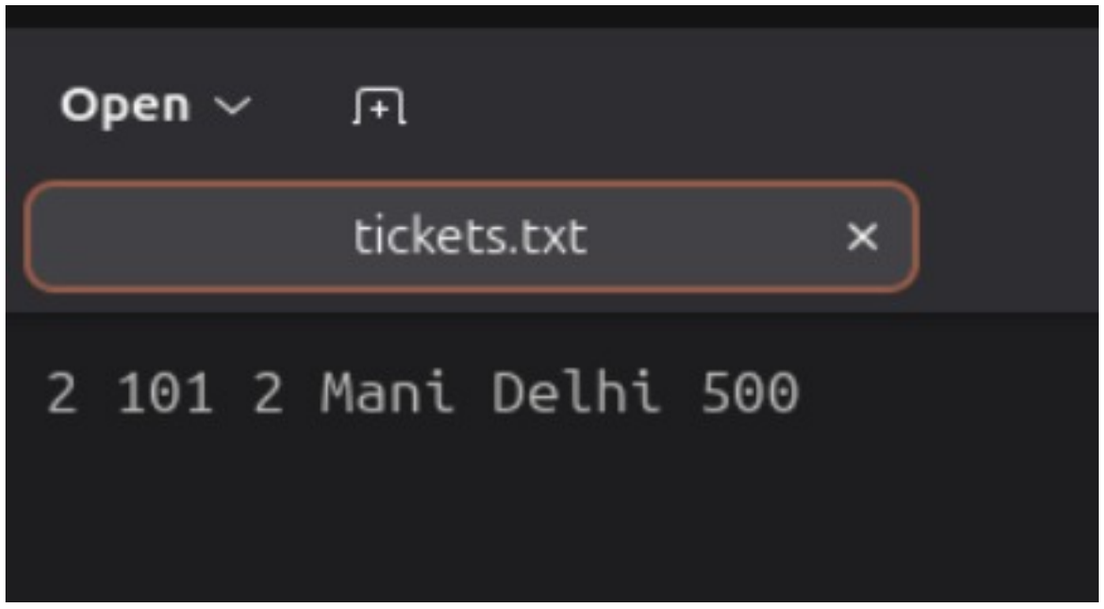
Cancel Ticket

```
===== BUS TICKET BOOKING SYSTEM =====
1. Book Ticket
2. View All Tickets
3. View Tickets of a Bus
4. View by Destination
5. Cancel Ticket
6. Exit
Enter choice: 5
Enter Ticket ID to Cancel: 1
Ticket cancelled.
```

Exit

```
===== BUS TICKET BOOKING SYSTEM =====
1. Book Ticket
2. View All Tickets
3. View Tickets of a Bus
4. View by Destination
5. Cancel Ticket
6. Exit
Enter choice: 6
Exiting...
```

Text File



9. Conclusion and Future Work

9.1. Conclusion

The Bus Ticket Booking System successfully meets its goal of providing a simple and efficient way to book, view, and cancel bus tickets. It reduces manual work and makes the ticket process faster and more organized. By using C programming concepts like structures, functions, and file handling, the project shows how real-life problems can be solved through basic programming logic. Overall, the system is easy to use, reliable for small-scale usage, and helps understand how software applications are designed and implemented.

9.2. Future work

Graphical User Interface (GUI):
Instead of a text-based console, a GUI can be created using GTK or any other C-based toolkit to make the system more attractive.

Online Booking System:
The project can be extended into a web-based application where users can book tickets online using internet connectivity.

Database Integration:
Instead of storing data in text files, a proper database like MySQL or SQLite can be used for better security and faster access.

User Login System:
Admin and user login features can be added to improve security and restrict unauthorised access.

Multiple Bus Routes & Timings:
The system can include different bus types, routes, timings, and seat categories (AC/Non-AC).

Payment Integration:
Online payment options can be introduced to make the system more realistic.

Ticket Generation with QR Code:
The system can generate a digital ticket with a QR code that can be scanned for verification.

10. References

1. Balagurusamy, E. – *Programming in ANSI C*, Tata McGraw-Hill.
2. Byron Gottfried – *Programming with C*, Tata McGraw-Hill Education.
3. Yashavant Kanetkar – *Let Us C*, BPB Publications.
4. Online Tutorials – *GeeksforGeeks (C Programming Basics and File Handling)*.
5. Online Documentation – *TutorialsPoint (C Language Concepts, Structures, and Functions)*.
6. Lecture notes provided by the faculty for C programming basics.
7. Sample reference codes from *GitHub (Open-source C projects for learning only)*.

11. Appendix

A. Sample Input/Output Screens

1. Booking a Ticket

```
Enter passenger name: Rahul
Enter age: 21
Choose seat number: 12
Ticket booked successfully!
```

2. Viewing Booked Tickets

```
--- Booked Tickets ---
Seat 12 : Rahul (21)
Seat 5  : Priya (19)
```

3. Cancelling a Ticket

```
Enter seat number to cancel: 12
Ticket cancelled successfully.
```

B. Sample Code Snippet

```
struct Ticket {
    char name[50];
    int age;
    int seat;
};
```

C. File Structure Description

- ticket.txt → Stores all booked ticket details
- main.c → Contains code for menu and functions
- functions.c → Code for booking, viewing, canceling tickets
- functions.h → Header file with function declarations

D. Tools Used

- Language: C Programming
- Compiler: GCC
- Editor: VS Code / CodeBlocks / Turbo C (any used by student)

Thank You