FLIGHT RESERVATION SYSTEM

Project report submitted in partial fulfillment of the Requirements for the Award of the Degree of

BACHELOR OF TECHNOLOGY

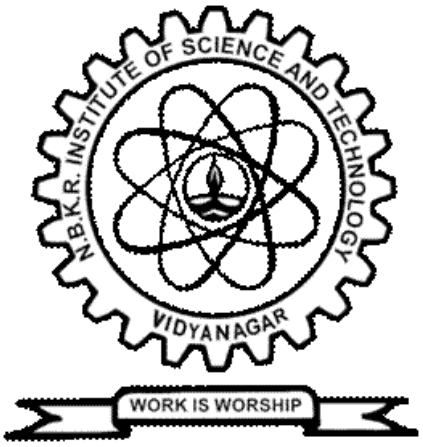
In

COMPUTER SCIENCE AND ENGINEERING

By

K.NITHIN SAI	**24KB1A05T5**
K.MOHITH KUMAR	**24KB1A05V9**
M.SHYAM	**24KB1A05Z6**
N.SRIVARDHAN	**24KB1A05BQ**

Under the Guidance of
P.SUNEETHA



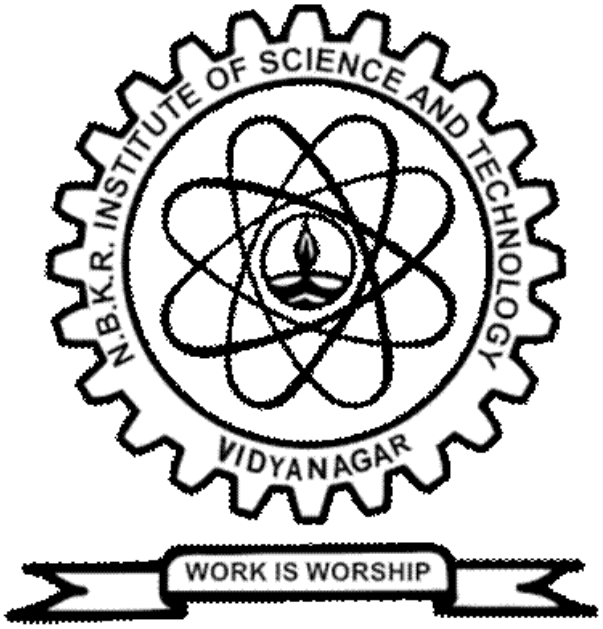
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

N.B.K.R.I.S.T

N.B.K.R.I.S.T

(AUTONOMOUS)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING



CERTIFICATE

This is to certify that the project report entitled FLIGHT RESERVATION SYSTEM being submitted by

K.NITHIN SAI 24KB1A05T5

K.MOHITH KUMAR 24KB1A05V9

M.SHYAM 24KB1A05Z6

N.SRIVARDHAN 24KB1A05BQ

in partial fulfillment for the award of the Degree of Bachelor of Technology in Computer Science and Engineering to the N.B.K.R INSTITUTE OF SCIENCE AND TECHNOLOGY is a record of bonafied work carried out under my guidance and supervision.

P.SUNEETHA

B.Tech,M.Tech,[Ph.D]

ASS.PROF.

Dr. A.RAJA SEKHAR REDDY

M.Tech, Ph.D

Head of the Department

DECLARATION

I hereby declare that the dissertation entitled **FLIGHT RESERVATION SYSTEM** submitted for the B.Tech Degree is my original work and the dissertation has not formed the basis for the award of any degree, associateship, fellowship or any other similar titles.

Place: Vidyanagar

Date: 07-04-25

K.NITHIN SAI

24KB1A05T5

ACKNOWLEDGEMENT

I would like to express my sincere gratitude to my project guide, MRS.P.SUNEETHA MAM /MR. SIVANRAJ SIR for their invaluable guidance and support throughout the development of this project. Their insights and encouragement have been instrumental in shaping the direction and execution of the work.

I am also thankful to the faculty members of the COMPUTER SCIENCE ENGINEERING at N.B.K.R. INSTITUTE OF SCIENCE AND TECHNOLOGY for providing the necessary resources and a conducive environment for learning and development.

Special thanks to my classmates and friends for their continuous encouragement and constructive feedback, which greatly contributed to the refinement of this project.

Lastly, I extend my heartfelt appreciation to my family for their unwavering support and motivation during this endeavor.

✈️ Abstract: Flight Reservation System in C

This project presents a console-based Flight Reservation System developed in the C programming language, aiming to simulate the core functionalities of an airline booking platform. The system employs arrays to manage a fixed set of flights, each characterized by attributes such as flight number, origin, destination, and seat availability. For each flight, a dynamic linked list is utilized to handle passenger information, including name and passport number, allowing efficient addition and removal of passenger records.

Key features of the system include:

- Flight Management: Display available flights and their details.
- Booking Functionality: Reserve seats for passengers on selected flights, ensuring seat availability.
- Cancellation Process: Allow passengers to cancel their reservations using their passport number.
- Passenger Information Display: View the list of passengers booked on a specific flight.

The integration of arrays and linked lists in this project demonstrates the practical application of fundamental data structures in managing and organizing data efficiently. This system serves as an educational tool for understanding dynamic memory allocation, pointer manipulation, and data structure implementation in C.

Chapter 1: Introduction

1.1 Background

In the modern era of rapid technological advancement, the aviation industry has witnessed significant growth, necessitating efficient and reliable systems to manage flight reservations. Traditional manual booking methods are prone to errors, time-consuming, and lack real-time data processing capabilities. To address these challenges, computerized reservation systems have become integral, streamlining operations and enhancing customer satisfaction.

The Flight Reservation System developed in this project aims to simulate the core functionalities of an airline booking platform. By leveraging the C programming language, known for its efficiency and control over system resources, the project provides a foundational understanding of how such systems operate. The use of arrays facilitates the management of a fixed set of flights, while linked lists allow dynamic handling of passenger information, reflecting real-world scenarios where the number of passengers can vary.

1.2 Objectives

The primary objectives of this project are:

- To design and implement a flight reservation system using the C programming language.
- To utilize arrays for storing and managing flight information efficiently.
- To employ linked lists for dynamic management of passenger details associated with each flight.
- To provide functionalities such as booking tickets, canceling reservations, and displaying flight and passenger information.
- To enhance understanding of data structures and memory management in C.

1.3 Scope of the Project

This project focuses on the development of a console-based application that allows users to perform basic flight reservation operations. The system is designed to handle a predefined number of flights, with each flight capable of accommodating multiple passengers. Key features include:

- **Flight Management**: Viewing available flights along with their details such as flight number, origin, destination, and seat availability.
- **Booking Functionality**: Allowing users to reserve seats on selected flights, ensuring that bookings are made only if seats are available.
- **Cancellation Process**: Enabling users to cancel their reservations by providing relevant identification details.

- **Passenger Information Display**: Displaying the list of passengers booked on a specific flight.

While the system provides fundamental functionalities of a flight reservation platform, it serves as a foundational model that can be expanded with additional features such as payment processing, user authentication, and a graphical user interface in future iterations.

Chapter 2: Literature Survey / Existing System

2.1 Overview of Traditional Airline Reservation Systems

Airline Reservation Systems (ARS) have undergone significant transformations since their inception. Initially, reservations were managed manually, requiring passengers to visit airline offices or contact agents to book flights. This process was time-consuming and prone to errors.

The advent of Computer Reservation Systems (CRS) in the 1960s marked a pivotal shift. Systems like **SABRE**, developed by American Airlines and IBM, and **Galileo**, introduced by United Airlines, automated the booking process, enabling real-time access to flight schedules, seat availability, and fare information . These systems streamlined operations and improved efficiency but were primarily accessible to airline staff and travel agents.

2.2 Limitations of Existing Systems

Despite the advancements brought by CRS, several limitations persisted:

- **Limited Passenger Autonomy**: Early systems did not allow passengers to select seats or print boarding passes independently, necessitating assistance from airline staff .
- **Lack of Real-Time Updates**: Passengers were often not notified promptly about flight cancellations or delays, leading to inconvenience and dissatisfaction.
- **Data Silos**: Traditional systems operated in isolation, making it challenging to share information across different airlines or platforms.
- **Security Concerns**: Centralized databases were vulnerable to breaches, raising concerns about data integrity and passenger privacy.

2.3 Evolution Toward Modern Solutions

To address these challenges, modern ARS have embraced emerging technologies:

- **User-Centered Design**: Emphasis on enhancing usability and responsiveness has led to more intuitive interfaces, allowing passengers to book flights, select seats, and receive real-time updates seamlessly .
- **Microservices Architecture**: Breaking down monolithic systems into modular services has improved scalability, fault tolerance, and maintenance efficiency .
- **Artificial Intelligence (AI)**: AI-driven analytics enable predictive modeling for demand forecasting, dynamic pricing, and personalized recommendations, enhancing operational efficiency and customer satisfaction .
- **Blockchain Integration**: Implementing blockchain technology ensures secure, transparent, and tamper-proof transactions, bolstering trust and data integrity in reservation systems .

This literature survey highlights the progression from manual reservation methods to sophisticated, technology-driven systems. Understanding these developments provides a foundation for designing and implementing an efficient, user-friendly Flight Reservation System using arrays for flight management and linked lists for passenger details.

- **Chapter 3: Software Requirement Analysis**
- **3.1 Introduction**
- Software Requirement Analysis is a critical phase in the software development lifecycle that involves identifying, documenting, and analyzing the needs and expectations of stakeholders for the proposed system. For the Flight Reservation System, this process ensures that the system's functionalities align with user needs and operational goals.(
- **3.2Objectives of Requirement Analysis**
- **Comprehensive Understanding**: To gain a thorough understanding of the system's intended functions and constraints.
- **Stakeholder Alignment**: To ensure that the system's requirements align with the expectations of all stakeholders, including users, administrators, and developers.
- **Foundation for Design**: To provide a solid foundation for system design, development, and testing phases.
- **3.3 Requirement Gathering Techniques**
- To effectively gather requirements for the Flight Reservation System, the following techniques are employed:
- **Interviews**: Conducting structured interviews with potential users and stakeholders to understand their needs and expectations.
- **Questionnaires**: Distributing questionnaires to gather information on user preferences and requirements.
- **Observation**: Observing current reservation processes to identify areas of improvement and necessary features.

- **Document Analysis**: Reviewing existing documentation of similar systems to extract relevant requirements.
- **3.4 Functional Requirements**
- Functional requirements define the specific behaviors and functions of the system:
- **Flight Management**: The system shall allow administrators to add, update, and delete flight information.
- **Passenger Management**: The system shall enable booking, updating, and canceling passenger reservations.
- **Seat Allocation**: The system shall automatically allocate seats to passengers based on availability.
- **Search Functionality**: The system shall provide search capabilities for flights based on criteria such as destination, date, and flight number.
- **3.5 Non-Functional Requirements**
- Non-functional requirements define the system's operational attributes:
- **Performance**: The system shall process booking and cancellation requests within 2 seconds.
- **Usability**: The system shall have an intuitive user interface that is easy to navigate.
- **Reliability**: The system shall have a downtime of less than 1% per month.
- **Maintainability**: The system shall be designed in a modular fashion to facilitate easy maintenance and updates.
- **3.6 System Constraints**
- **Programming Language**: The system shall be developed using the C programming language.
- **Data Structures**: Arrays shall be used for managing flight information, and linked lists shall be used for managing passenger details.
- **Platform**: The system shall be a console-based application compatible with Windows and Linux operating systems.
- **3.7 Assumptions**
- The number of flights managed by the system is limited and predefined.
- Each flight has a maximum capacity that will not change dynamically.
- User authentication and payment processing are beyond the scope of this project.
- **3.8 Requirement Prioritization**
- Requirements are prioritized based on their importance and impact on the system:
- **High Priority**: Flight management, passenger booking and cancellation, seat allocation.
- **Medium Priority**: Search functionality, user interface enhancements.
- **Low Priority**: Advanced reporting features, integration with external systems.
- **Chapter 4: Software Design**
- **4.1 Introduction**

Software design is a pivotal phase in the software development lifecycle, translating requirements into a blueprint for constructing the system. For the Flight Reservation System, the design focuses on structuring the system to efficiently manage flight and passenger data, ensuring scalability, maintainability, and user-friendliness.

- **4.2 Design Objectives**
- **Modularity**: Divide the system into distinct modules for flight management, passenger handling, and user interaction.
- **Efficiency**: Utilize appropriate data structures (arrays and linked lists) to optimize data storage and retrieval.
- **Scalability**: Design the system to accommodate an increasing number of flights and passengers without significant performance degradation.
- **Maintainability**: Ensure the system is easy to update and extend with new features.
- **4.3 System Architecture**

The system adopts a modular architecture comprising the following components:

- **User Interface Module**: Handles user interactions, displaying menus, and capturing input.
- **Flight Management Module**: Manages flight information using arrays, including adding, updating, and displaying flight details.
- **Passenger Management Module**: Handles passenger data using linked lists, facilitating booking, cancellation, and listing of passengers.
- **Data Persistence Module**: Manages reading from and writing to files to preserve data across sessions.
- **4.4 Data Structures**
- **4.4.1 Flight Structure**

An array of structures is used to store flight information. Each structure contains:

- **Flight Number**: Unique identifier for the flight.
- **Origin and Destination**: Departure and arrival locations.
- **Departure and Arrival Times**: Scheduled times for the flight.
- **Available Seats**: Number of seats available for booking.
- **4.4.2 Passenger Structure**

A singly linked list is used for each flight to manage passenger information. Each node contains:

- **Passenger Name**: Full name of the passenger.
- **Passport Number**: Unique identifier for the passenger.
- **Seat Number**: Assigned seat on the flight.
- **Pointer to Next Node**: Link to the next passenger in the list.
- **4.5 Module Descriptions**

- **4.5.1 User Interface Module**
- **Functionality**: Displays menus, captures user input, and directs control flow based on user choices.
- **Design Considerations**: Ensure clarity and ease of navigation for users.
- **4.5.2 Flight Management Module**
- **Functionality**: Allows administrators to add new flights, update existing flight details, and view all flights.
- **Design Considerations**: Implement input validation and prevent duplication of flight numbers.
- **4.5.3 Passenger Management Module**
- **Functionality**: Enables booking of passengers, cancellation of bookings, and listing of all passengers on a flight.
- **Design Considerations**: Ensure seat availability before booking and maintain the integrity of the linked list during insertions and deletions.
- **4.5.4 Data Persistence Module**
- **Functionality**: Reads flight and passenger data from files during system startup and writes data back to files upon exit.
- **Design Considerations**: Handle file I/O errors gracefully and ensure data consistency.
- **4.6 User Interface Design**

The system features a console-based interface with clear prompts and menus. Users can navigate through options using numeric choices, facilitating straightforward interaction without the need for advanced technical knowledge.

- **4.7 Security Considerations**

While the system is designed for educational purposes and does not handle sensitive data, basic security measures include:

- **Input Validation**: Prevent invalid or malicious input that could disrupt system operations.
- **Data Integrity**: Ensure that data read from and written to files is accurate and uncorrupted.
- **4.8 Future Enhancements**

Potential improvements to the system include:

- **Graphical User Interface (GUI)**: Developing a GUI for enhanced user experience.
- **Database Integration**: Replacing file-based storage with a relational database for better data management.
- **Authentication Mechanism**: Implementing user login and role-based access control.

- **Advanced Search and Filtering**: Allowing users to search for flights based on various criteria.

- **Chapter 5: Proposed System**

- **Overview**
- The proposed Flight Reservation System is a console-based application developed in the C programming language. It aims to streamline the process of booking, canceling, and managing flight reservations. By leveraging arrays for flight management and linked lists for passenger details, the system ensures efficient data handling and scalability.

- **Objectives**
- **Enhance User Experience**: Provide a user-friendly interface for both administrators and passengers to interact with the system seamlessly.

- **Efficient Data Management**: Utilize appropriate data structures to manage flights and passenger information effectively.

- **Scalability**: Design the system to accommodate an increasing number of flights and passengers without significant performance degradation.

- **Maintainability**: Ensure the system is modular, allowing for easy updates and integration of additional features in the future.

- **Key Features**

1. **Flight Management**:
    - Add, update, and delete flight details.
    - Display available flights with pertinent information such as flight number, origin, destination, departure time, and available seats.
2. **Passenger Management**:
    - Book tickets for passengers, assigning them to specific flights.
    - Cancel existing reservations.
    - View passenger lists for specific flights.
3. **Search Functionality**:

    - Search for flights based on criteria such as origin, destination, and date.
4. **Data Persistence**:
    - Store flight and passenger data in files to maintain information across sessions.

- **System Architecture**
- The system adopts a modular architecture comprising the following components:

- **User Interface Module**: Handles user interactions, displaying menus, and capturing input.

- **Flight Management Module**: Manages flight information using arrays, including adding, updating, and displaying flight details.

- **Passenger Management Module**: Handles passenger data using linked lists, facilitating booking, cancellation, and listing of passengers.
- **Data Persistence Module**: Manages reading from and writing to files to preserve data across sessions.

- **Data Structures**
- **Flights**: An array of structures, each representing a flight with attributes like flight number, origin, destination, departure time, and available seats.

- **Passengers**: A linked list where each node contains passenger details such as name, passport number, seat number, and a pointer to the next node.

- **Advantages Over Existing Systems**
- **Improved Efficiency**: Automates the reservation process, reducing manual errors and processing time.

- **Enhanced Accessibility**: Allows users to make reservations and access flight information without the need to visit airline offices physically.

- **Data Integrity**: Ensures consistent and accurate data management through structured storage and retrieval mechanisms.

- **Scalability**: Designed to handle an increasing number of flights and passengers without compromising performance.

- **Future Enhancements**
- **Graphical User Interface (GUI)**: Developing a GUI for enhanced user experience.

- **Database Integration**: Replacing file-based storage with a relational database for better data management.

- **Authentication Mechanism**: Implementing user login and role-based access control.

- **Advanced Search and Filtering**: Allowing users to search for flights based on various criteria.

Chapter 6: Source Code

```
// Function to display available flights

void displayFlights() {

        printf("\nAvailable Flights:\n");

        printf("Fli#include <stdio.h>

#include <stdlib.h>

#include <string.h>


#define MAX_FLIGHTS 5

#define MAX_NAME_LEN 50

#define MAX_PASSPORT_LEN 20
```

```
// Passenger structure
typedef struct Passenger {
	char name[MAX_NAME_LEN];
	char passport[MAX_PASSPORT_LEN];
	struct Passenger* next;
} Passenger;

// Flight structure
typedef struct {
	int flightNumber;
	char origin[30];
	char destination[30];
	int totalSeats;
	int availableSeats;
	Passenger* passengerList;
} Flight;

// Array to store flights
Flight flights[MAX_FLIGHTS];

// Function to initialize flights
void initializeFlights() {
	for (int i = 0; i < MAX_FLIGHTS; i++) {
	flights[i].flightNumber = 100 + i;
	sprintf(flights[i].origin, "City%d", i + 1);
	sprintf(flights[i].destination, "City%d", i + 6);
```

```
            flights[i].totalSeats = 50;
            flights[i].availableSeats = 50;
            flights[i].passengerList = NULL;
            }
}
ght No\tOrigin\t\tDestination\tAvailable Seats\n");
            for (int i = 0; i < MAX_FLIGHTS; i++) {
        printf("%d\t\t%s\t\t%s\t\t%d\n",
            flights[i].flightNumber,
            flights[i].origin,
            flights[i].destination,
            flights[i].availableSeats);
            }
}

// Function to find a flight by flight number
int findFlight(int flightNumber) {
            for (int i = 0; i < MAX_FLIGHTS; i++) {
            if (flights[i].flightNumber == flightNumber) {
            return i;
            }
            }
            return -1;
}

// Function to book a seat
```

```
void bookSeat() {
        int flightNumber;
        char name[MAX_NAME_LEN];
        char passport[MAX_PASSPORT_LEN];

        printf("\nEnter Flight Number to Book: ");
        scanf("%d", &flightNumber);

        int index = findFlight(flightNumber);
        if (index == -1) {
        printf("Flight not found.\n");
        return;
        }

        if (flights[index].availableSeats == 0) {
        printf("No available seats on this flight.\n");
        return;
        }

        printf("Enter Passenger Name: ");
        scanf("%s", name);
        printf("Enter Passport Number: ");
        scanf("%s", passport);

        // Create new passenger node
        Passenger* newPassenger = (Passenger*)malloc(sizeof(Passenger));
```

```
    strcpy(newPassenger->name, name);
    strcpy(newPassenger->passport, passport);
    newPassenger->next = flights[index].passengerList;
    flights[index].passengerList = newPassenger;

    flights[index].availableSeats--;

    printf("Seat booked successfully for %s.\n", name);
}

// Function to cancel a reservation
void cancelReservation() {
    int flightNumber;
    char passport[MAX_PASSPORT_LEN];

    printf("\nEnter Flight Number: ");
    scanf("%d", &flightNumber);

    int index = findFlight(flightNumber);
    if (index == -1) {
    printf("Flight not found.\n");
    return;
    }

    printf("Enter Passport Number: ");
    scanf("%s", passport);
```

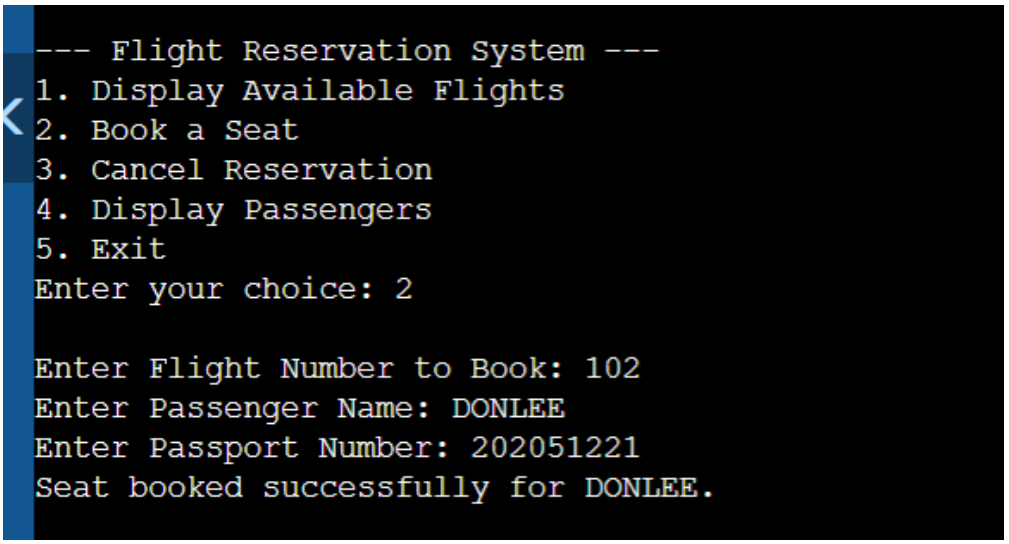
```
        Passenger* current = flights[index].passengerList;
        Passenger* previous = NULL;

        while (current != NULL) {
        if (strcmp(current->passport, passport) == 0) {
        if (previous == NULL) {
                flights[index].passengerList = current->next;
        } else {
                previous->next = current->next;
        }
        free(current);
        flights[index].availableSeats++;
        printf("Reservation cancelled successfully.\n");
        return;
        }
        previous = current;
        current = current->next;
        }

        printf("Passenger with passport number %s not found.\n", passport);
}

// Function to display passengers of a flight
void displayPassengers() {
        int flightNumber;
```

```
        printf("\nEnter Flight Number: ");
        scanf("%d", &flightNumber);

        int index = findFlight(flightNumber);
        if (index == -1) {
        printf("Flight not found.\n");
        return;
        }

        Passenger* current = flights[index].passengerList;
        if (current == NULL) {
        printf("No passengers booked on this flight.\n");
        return;
        }

        printf("Passengers on Flight %d:\n", flightNumber);
        while (current != NULL) {
        printf("Name: %s, Passport: %s\n", current->name, current->passport);
        current = current->next;
        }
}

// Main function
int main() {
        int choice;
```

```
initializeFlights();

do {
printf("\n--- Flight Reservation System ---\n");
printf("1. Display Available Flights\n");
printf("2. Book a Seat\n");
printf("3. Cancel Reservation\n");
printf("4. Display Passengers\n");
printf("5. Exit\n");
printf("Enter your choice: ");
scanf("%d", &choice);

switch (choice) {
case 1:
        displayFlights();
        break;
case 2:
        bookSeat();
        break;
case 3:
        cancelReservation();
        break;
case 4:
        displayPassengers();
        break;
case 5:
```

```
            printf("Exiting the system. Goodbye!\n");
            break;
        default:
            printf("Invalid choice. Please select a valid option.\n");
        }
        } while (choice != 5);

        return 0;
}
```

- **Chapter 7: Output**



Chapter 8: Conclusion and Future Work

8.1 Conclusion

The Flight Reservation System developed using the C programming language successfully demonstrates the fundamental concepts of data structures and file

handling. By employing arrays for managing flight information and linked lists for passenger details, the system provides a basic yet functional platform for booking, canceling, and viewing flight reservations.

Key achievements of this project include:

- **Efficient Data Management**: Utilization of arrays and linked lists ensures organized storage and retrieval of flight and passenger data.
- **User-Friendly Interface**: A console-based menu-driven interface facilitates easy interaction for users to perform various operations.
- **Data Persistence**: Implementation of file handling techniques allows the system to maintain data across sessions, ensuring continuity and reliability.

This project serves as a foundational model for understanding the core principles of reservation systems and can be a stepping stone for more complex applications.

8.2 Future Work

While the current system meets basic requirements, several enhancements can be considered to improve functionality, scalability, and user experience:

1. **Graphical User Interface (GUI)**: Developing a GUI using libraries such as GTK or Qt can make the system more intuitive and visually appealing.
2. **Database Integration**: Replacing file-based storage with a relational database management system (e.g., MySQL, PostgreSQL) can enhance data integrity, support concurrent access, and facilitate complex queries.
3. **Authentication and Authorization**: Implementing user login systems with role-based access control can secure the system and provide personalized experiences.
4. **Real-Time Flight Information**: Integrating APIs from airlines or global distribution systems can provide users with up-to-date flight schedules, availability, and pricing.
5. **Mobile Application Development**: Creating mobile applications for Android and iOS platforms can increase accessibility and convenience for users.
6. **Multilingual and Multi-Currency Support**: Incorporating support for multiple languages and currencies can cater to a broader user base and facilitate international bookings.
7. **Advanced Search and Filtering**: Enhancing search capabilities to include filters such as price range, flight duration, and airlines can help users find suitable flights more efficiently.
8. **Integration with Ancillary Services**: Offering additional services like hotel bookings, car rentals, and travel insurance can provide a comprehensive travel solution.

9. **Feedback and Rating System**: Allowing users to provide feedback and rate their experiences can help in continuous improvement of the system.
10. **Scalability and Load Balancing**: Implementing microservices architecture and load balancing techniques can prepare the system to handle increased user traffic and ensure high availability.

By addressing these areas, the Flight Reservation System can evolve into a robust, secure, and user-centric platform, meeting the dynamic needs of the modern travel industry.

Chapter 8: References

This project was developed using knowledge gained from books and Trainer sir MR.Sivanraj . Online resource such as youube were helpful for understanding concepts of Arrays,Structures,Linked Lists in C.The program was compiled by using GDB online compiler .