## Department of Information Science and Engineering

**A DS Mini Project Report On**

**“SIMPLE AIRLINE TICKET BOOKING**

**SYSTEM”**

submitted in partial fulfillment for the award of degree

of

Bachelor of Engineering

in

**INFORMATION SCIENCE AND ENGINEERING**

by

**Under the guidance of**

Assistant professor, Dept. Of Information Science and Engineering

|  |
| --- |
| **The National Institute of Engineering**  (An Autonomous Institute under Visvesvaraya Technological University, Belagavi)  Manandavadi Road, Mysuru–570008, Karnataka, India  Recognized by AICTE, New Delhi, Accredited by National Board of Accreditation, New Delhi  Grant-in-Aid by Government of Karnataka |

**Department of Information Science & Engineering**

**CERTIFICATE**

Certified that the Data Structure Mini-Project work entitled **“Simple Airline Ticket Booking System”** carried out by Mr. **USN:** is a bonafide student of mester ‘’ section is submitted in partial fulfillment for the award of Bachelor of Engineering Degree in Information Science and Engineering of**,** an autonomous institute under Visvesvaraya Technological University, Belagavi during the A.Y.2023-2024. It is certified that all suggestions/ corrections suggested during Internal Assessment have been incorporated in the report deposited in the departmental library. The mini project report has been approved as it satisfies the academic requirements in respect of work prescribed for the award of the said Degree.

**Name & Signature of Guide Name & Signature of HoD**

**ACKNOWLEDGEMENT**

We are extremely thankful, Principal, , Mysuru, for providing us the academic ambiance and laboratory facilities to work, and everlasting motivation to carry out this work and shape our careers.

We express our sincere gratitude tofor his stimulating guidance, continuous encouragement, and motivation throughout the course of the present work.

We extend our gratitude to our Guide Professor Information Science for providing relevant information, guidance and encouragement to complete this mini project.

We take this opportunity to thank all our friends, classmates who always stood by us in difficult situations and helped us in some technical aspects

We express our gratitude to our parents who were a constant source of encouragement and stood by us as a pillar of strength for completing this work and course successfully.

**Yours Sincerely,**

**ABSTRACT**

The airline industry relies heavily on efficient booking systems to manage reservations, ticket sales, and passenger information. Traditional methods often involve complex data structures and algorithms to handle the dynamic nature of bookings. This abstract proposes a novel approach to streamline the process by utilizing linked lists for airline ticket booking.

Linked lists offer several advantages over traditional data structures in terms of memory allocation, flexibility, and ease of manipulation. In the context of airline ticket booking, linked lists provide a scalable solution that can accommodate fluctuating passenger demands and optimize resource utilization.

At the core of our system is a doubly linked list, where each node represents a booking record containing essential information such as passenger details, flight information, seat allocation, and payment status. This structure allows for efficient insertion, deletion, and traversal operations, essential for managing real-time bookings in a dynamic environment.

The booking process begins with the creation of a new node to represent the reservation. Passenger details are collected and stored within the node, along with relevant flight information. The node is then inserted into the linked list based on criteria such as departure time, destination, or seat availability.

As bookings are made, the system dynamically adjusts the linked list, ensuring optimal seat allocation and resource utilization. Linked list operations such as insertion and deletion are performed in constant time, resulting in fast and responsive booking transactions.

Furthermore, the use of linked lists facilitates easy integration with other components of the booking system, such as payment processing and seat assignment algorithms. By leveraging the inherent flexibility of linked lists, our system can adapt to changing requirements and scale efficiently to handle large volumes of bookings.

In conclusion, the proposed approach offers a robust and efficient solution for airline ticket booking systems. By harnessing the power of linked lists, we can streamline the booking process, optimize resource utilization, and enhance the overall passenger experience. Future work may explore enhancements such as optimization techniques and integration with emerging technologies to further improve system performance and functionality.

**TABLE OF CONTENTS**

**Chapter Page No**

1. Introduction

About Data Structures 01

Advantages of Data Structures 01

Introduction to “**SIMPLE AIRLINE TICKETS BOOKING**” 02

About Linked List 03

1. Requirement Specification 04

Hardware Requirement:

* Any modern computer with sufficient

processing power and memory (e.g.,2GB RAM)

* Adequate storage space for development

tools and project files

Software Requirement:

* C Compiler (e.g., GCC, Clang)
* Text Editor/IDE (e.g., Visual Studio Code, Sublime Text)

1. Implementation: 05

Code snippets 07

1. Screenshots 13

Conclusion 15

Bibliography 16

# Chapter 1 Introduction

## About Data Structures

Data structures form the backbone of efficient data organization and manipulation in computer science. They are essential for storing, managing, and accessing data effectively. Data structures provide various ways to arrange and store data in memory, each with its own strengths and weaknesses. Examples include arrays, linked lists, stacks, queues, trees, graphs, and hash tables. Choosing the right data structure for a specific problem is crucial for optimizing time and space complexity in algorithms.

Different data structures excel in different scenarios. For instance, arrays offer fast access to elements by index but may be inefficient for inserting or deleting elements in the middle. Linked lists, on the other hand, provide efficient insertion and deletion operations at any position but may have slower access times. Understanding the properties and performance characteristics of each data structure empowers programmers to make informed decisions when designing algorithms and applications, ultimately leading to more robust and efficient software solutions.

## ****Advantages of**** ****Data Structures****

Data structures provide several advantages in software development:

* Efficient Data Organization: Data structures provide efficient ways to organize and store data, optimizing memory usage and facilitating easy access and retrieval.
* Fast Retrieval and Search: Certain data structures like hash tables and binary search trees offer fast retrieval and search operations, enabling quick access to data even with large datasets.
* Modularity and Reusability: Data structures promote modularity and reusability in programming by encapsulating data and operations, making it easier to manage and maintain code.
* Improved Algorithm Efficiency: Efficient data structures are crucial for implementing algorithms with optimal time and space complexity, enhancing overall program performance.
* Dynamic Memory Allocation: Dynamic data structures like linked lists and dynamic arrays allow for dynamic memory allocation, enabling flexibility in managing data size during runtime.
* Versatility: Different data structures serve different purposes, providing versatility in solving various types of problems and accommodating diverse application requirements.

## 

* Concurrency Support: Some advanced data structures are designed to support concurrent access and modification, facilitating multi-threaded programming and parallel processing.
* Facilitates Data Manipulation: Data structures offer built-in operations for manipulating data, such as insertion, deletion, sorting, and traversal, simplifying complex data manipulation tasks.
* Memory Efficiency: Efficient data structures minimize memory overhead and fragmentation, optimizing memory usage and reducing resource consumption.
* Scalability: Scalable data structures can handle growing datasets and workload demands without significant performance degradation, ensuring applications remain responsive and efficient as they scale.

## ****Introduction To “Simple Airline Ticket Booking System”****

## The provided code is a C program that implements an online quiz game using Linked Lists data structure to manage quiz questions and user profiles. Here's a brief overview of the code:

## Header Files: The header section includes essential C libraries that are used throughout the program. These libraries facilitate input/output operations, memory allocation, string manipulation, and boolean data type support. Additionally, macros are defined to specify constants such as the maximum number of flights and seats per flight, and the number of business and economy class seats.

## Data Structures: Two structures are defined: Flight and Passenger. The Flight structure represents information about each flight, including its number, departure and arrival cities, departure and arrival times, and available seats in both business and economy classes. The Passenger structure stores details about each passenger, such as name, age, seat class, seat number, and the flight they are booked on. Both structures include pointers to facilitate the creation of linked lists.

## Function section: The function section contains implementations for various functionalities of the reservation system. Functions are provided to create flights, display available flights, display ticket details, book seats, delete booked tickets, and check login credentials. Memory allocation functions are also included to allocate memory for flights and passengers dynamically. Additionally, helper functions are defined to free memory allocated for flights and passengers when the program terminates.

## Input section: The input section handles user interactions by presenting a menu-driven interface. Users are prompted to enter their credentials for authentication. Once logged in, they can choose from a range of options, including displaying available flights, booking seats, displaying ticket details, deleting booked tickets, or exiting the program. Input validation is implemented to ensure that user inputs are within valid ranges and formats.

## Execution section: The execution section contains the main function, which orchestrates the execution flow of the program. It initializes flight information, prompts users for login credentials, and presents the main menu for interacting with the reservation system. The program continues to execute until the user chooses to exit.

## Conclusion: In conclusion, the provided code demonstrates the implementation of a basic Airline Ticket Reservation System using linked lists in C. It offers essential functionalities for managing flight bookings efficiently, providing users with a simple yet effective interface for reserving seats and managing tickets.

## ****Introduction To Linked Lists****

A linked list is a fundamental data structure commonly used in computer science and programming. Unlike arrays, which store elements in contiguous memory locations, a linked list consists of a series of nodes, each containing a data element and a reference (or pointer) to the next node in the sequence. This structure allows for dynamic memory allocation, making linked lists versatile and efficient for various applications.

The core concept of a linked list revolves around its nodes. Each node contains two components: the data it holds and a pointer to the next node in the sequence. This pointer essentially establishes the connection between nodes, forming a chain-like structure. Because nodes are linked through pointers, they can be scattered throughout the computer's memory, allowing for efficient memory utilization and dynamic allocation.

One of the key advantages of linked lists is their flexibility in terms of insertion, deletion, and traversal operations. Unlike arrays, which require shifting elements to accommodate insertions or deletions, linked lists only need to update pointers, resulting in constant-time complexity for these operations in many cases. This property makes linked lists particularly suitable for scenarios where frequent data manipulation is required.

Linked lists come in various forms, with singly linked lists being the simplest. In a singly linked list, each node has a pointer to the next node, forming a unidirectional sequence. Another common variant is the doubly linked list, where each node has pointers to both the next and previous nodes, enabling bidirectional traversal. These variations offer different trade-offs in terms of memory overhead and operation complexity, allowing developers to choose the most appropriate type based on specific requirements.

In summary, linked lists are dynamic data structures that offer efficient memory utilization and support for dynamic operations such as insertion, deletion, and traversal. By linking nodes through pointers, linked lists provide flexibility and versatility in managing data, making them indispensable in many programming scenarios.

**Requirements**

**2.1 Hardware Requirements**

For running the Airline Ticket Booking,

* a standard modern computer with adequate processing power and memory is sufficient.
* A minimum of 2GB of RAM is recommended to ensure smooth operation of the program.
* Additionally, the computer should have sufficient storage space available for storing development tools, project files, and any additional resources required for running the software.

**2.2 Software Requirements**

To compile and execute the C code of the online quiz game,

* A ‘C’ compiler is necessary. Popular choices include GCC (GNU Compiler Collection) and Clang, both of which are widely used and freely available.
* Additionally, a text editor or integrated development environment (IDE) is required for writing, editing, and managing the source code.
* Examples of text editors/IDEs suitable for C programming include Visual Studio Code, Sublime Text, and many others.
* These software tools provide essential features such as syntax highlighting, code completion,
* and debugging capabilities, facilitating the development.

**3.1 Implementation**

1. **Header and library:**

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <stdbool.h>

#include <time.h>

This section includes necessary header files required for input/output operations, dynamic memory allocation, string manipulation, boolean values, and time-related functions.

1. **Definitions:**

#define MAX\_FLIGHTS 3

#define MAX\_SEATS\_PER\_FLIGHT 50

#define BUSINESS\_CLASS\_SEATS 10

#define ECONOMY\_CLASS\_SEATS 10

Defines constants for maximum number of flights, maximum seats per flight, and available seats in business and economy class.

1. **Data structures:**

typedef struct Flight {

char flightNumber[10];

char departureCity[50];

char arrivalCity[50];

char departureTime[20];

char arrivalTime[20];

int availableBusinessSeats;

int availableEconomySeats;

struct Flight \*next;

} Flight;

typedef struct Passenger {

char name[50];

int age;

char seatClass;

int seatNumber;

char flightNumber[10];

struct Passenger \*next;

} Passenger;

Defines structures for flights and passengers. Flight structure contains flight details and available seats. Passenger structure contains passenger details and seat allocation.

1. **Function Definitions:**

createFlight: Creates a new flight node.

displayFlights: Displays available flights.

displayTicketDetails: Displays booked ticket details.

bookSeat: Books a seat for a passenger on a flight.

deleteTicket: Deletes a booked ticket.

checkLogin: Checks login credentials**.**

1. **Main function:**

int main() {

// Initialization of variables and structures

// Creation of flights

// Login functionality

// Menu-driven interface for user interaction

// Memory deallocation

return 0;}

This is the entry point of the program. It initializes variables, creates flights, prompts for login, and provides a menu-driven interface for users to interact with the system.

**6. Memory deallocation:**

void freeFlights(Flight \*head);

void freePassengers(Passenger \*head);

The provided code implements an Airline Ticket Reservation System using C. It consists of functionalities to manage flights, book seats, display ticket details, and delete booked tickets. The code is organized into sections according to their functionalities, making it modular and easier to maintain.

**3.2 Code Snippet**

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <stdbool.h>

#include <time.h>

#define MAX\_FLIGHTS 3

#define MAX\_SEATS\_PER\_FLIGHT 50

#define BUSINESS\_CLASS\_SEATS 10

#define ECONOMY\_CLASS\_SEATS 10

typedef struct Flight {

char flightNumber[10];

char departureCity[50];

char arrivalCity[50];

char departureTime[20];

char arrivalTime[20];

int availableBusinessSeats;

int availableEconomySeats;

struct Flight \*next;

} Flight;

typedef struct Passenger {

char name[50];

int age;

char seatClass;

int seatNumber;

char flightNumber[10];

struct Passenger \*next;

} Passenger;

Flight\* createFlight(char \*flightNumber, char \*departureCity, char \*arrivalCity, char \*departureTime, char \*arrivalTime) {

Flight \*newFlight = (Flight\*)malloc(sizeof(Flight));

if (newFlight == NULL) {

printf("Memory allocation failed!\n");

exit(1);

}

strcpy(newFlight->flightNumber, flightNumber);

strcpy(newFlight->departureCity, departureCity);

strcpy(newFlight->arrivalCity, arrivalCity);

strcpy(newFlight->departureTime, departureTime);

strcpy(newFlight->arrivalTime, arrivalTime);

newFlight->availableBusinessSeats = BUSINESS\_CLASS\_SEATS;

newFlight->availableEconomySeats = ECONOMY\_CLASS\_SEATS;

newFlight->next = NULL;

return newFlight;

}

void displayFlights(Flight \*head) {

if (head == NULL) {

printf("No available flights to display.\n");

return;

}

Flight \*current = head;

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

while (current != NULL) {

printf("Flight Number: %s | Departure City: %s | Arrival City: %s | Departure Time: %s | Arrival Time: %s | Business Seats: %d | Economy Seats: %d\n",

current->flightNumber, current->departureCity, current->arrivalCity, current->departureTime, current->arrivalTime, current->availableBusinessSeats, current->availableEconomySeats);

current = current->next;

}

printf("\n");

}

void displayTicketDetails(Passenger \*head, Flight \*flights) {

if (head == NULL) {

printf("No booked tickets to display.\n");

return;

}

Passenger \*current = head;

printf("\nTicket Details:\n");

while (current != NULL) {

Flight \*currentFlight = flights;

while (currentFlight != NULL && strcmp(currentFlight->flightNumber, current->flightNumber) != 0) {

currentFlight = currentFlight->next;

}

if (currentFlight != NULL) {

printf("Flight Number: %s | Departure City: %s | Arrival City: %s | Departure Time: %s | Arrival Time: %s | ",

currentFlight->flightNumber, currentFlight->departureCity, currentFlight->arrivalCity, currentFlight->departureTime, currentFlight->arrivalTime);

}

printf("Name: %s | Age: %d | Seat Class: %c | Seat Number: %d\n",

current->name, current->age, current->seatClass, current->seatNumber);

current = current->next;

}

printf("\n");

}

void bookSeat(Flight \*head, Passenger \*passenger, char \*flightNumber, char seatClass) {

Flight \*currentFlight = head;

while (currentFlight != NULL) {

if (strcmp(currentFlight->flightNumber, flightNumber) == 0) {

if (seatClass == 'B' && currentFlight->availableBusinessSeats > 0) {

passenger->seatClass = 'B';

passenger->seatNumber = BUSINESS\_CLASS\_SEATS - currentFlight->availableBusinessSeats + 1;

currentFlight->availableBusinessSeats--;

} else if (seatClass == 'E' && currentFlight->availableEconomySeats > 0) {

passenger->seatClass = 'E';

passenger->seatNumber = ECONOMY\_CLASS\_SEATS - currentFlight->availableEconomySeats + 1;

currentFlight->availableEconomySeats--;

} else {

printf("No available seats in the specified class.\n");

return;

}

strcpy(passenger->flightNumber, flightNumber);

printf("Payment methods available:\n");

printf("1. GPay\n");

printf("2. PhonePe\n");

printf("3. Paytm\n");

printf("4. Debit Card\n");

printf("5. Credit Card\n");

printf("6. Online Banking\n");

int paymentMethod;

printf("Choose payment method (1-6): ");

scanf("%d", &paymentMethod);

switch(paymentMethod) {

case 1:

printf("Paid using GPay.\n");

break;

case 2:

printf("Paid using PhonePe.\n");

break;

case 3:

printf("Paid using Paytm.\n");

break;

case 4:

printf("Paid using Debit Card.\n");

break;

case 5:

printf("Paid using Credit Card.\n");

break;

case 6:

printf("Paid using Online Banking.\n");

break;

default:

printf("Invalid payment method.\n");

return;

}

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

return;

}

currentFlight = currentFlight->next;

}

printf("Invalid flight number.\n");

}

void deleteTicket(Passenger \*\*headRef, char \*flightNumber, int seatNumber) {

Passenger \*current = \*headRef;

Passenger \*prev = NULL;

while (current != NULL && (strcmp(current->flightNumber, flightNumber) != 0 || current->seatNumber != seatNumber)) {

prev = current;

current = current->next;

}

if (current != NULL) {

if (prev == NULL) {

\*headRef = current->next;

} else {

prev->next = current->next;

}

free(current);

printf("Ticket deleted successfully.\n");

} else {

printf("Ticket not found.\n");

}

}

bool checkLogin(char \*username, char \*password) {

return strcmp(username, "admin") == 0 && strcmp(password, "password") == 0;

}

void freeFlights(Flight \*head) {

while (head != NULL) {

Flight \*temp = head;

head = head->next;

free(temp);

}

}

void freePassengers(Passenger \*head) {

while (head != NULL) {

Passenger \*temp = head;

head = head->next;

free(temp);

}

}

int main() {

char username[50];

char password[50];

Flight \*flights = NULL;

Passenger \*passengers = NULL;

flights = createFlight("AI101", "Chennai", "Sri lanka", "09:00", "12:00");

flights->next = createFlight("AI202", "Bengaluru", "Mumbai", "13:00", "16:00");

flights->next->next = createFlight("AI303", "Gujarat", "Delhi", "18:00", "21:00");

printf("Welcome to Airline Ticket Reservation System\n");

printf("--------------------------------------------\n");

printf("Please login to continue.\n");

printf("Username: ");

scanf("%s", username);

printf("Password: ");

scanf("%s", password);

if (!checkLogin(username, password)) {

printf("Invalid credentials. Exiting program.\n");

freeFlights(flights);

return 1;

}

int choice;

do {

printf("\n1. Display Available Flights\n");

printf("2. Book a Seat\n");

printf("3. Display Ticket Details\n");

printf("4. Delete a Ticket\n");

printf("5. Exit\n");

printf("Enter your choice: ");

scanf("%d", &choice);

switch (choice) {

case 1:

displayFlights(flights);

break;

case 2:

{

char flightNumber[10];

char seatClass;

Passenger \*newPassenger = (Passenger\*)malloc(sizeof(Passenger));

printf("Enter Flight Number: ");

scanf("%s", flightNumber);

printf("Enter Passenger Name: ");

scanf("%s", newPassenger->name);

printf("Enter Passenger Age: ");

scanf("%d", &newPassenger->age);

printf("Enter Seat Class (B for Business, E for Economy): ");

scanf(" %c", &seatClass);

bookSeat(flights, newPassenger, flightNumber, seatClass);

newPassenger->next = passengers;

passengers = newPassenger;

}

break;

case 3:

displayTicketDetails(passengers, flights);

break;

case 4:

{

int seatNumber;

char flightNumber[10];

printf("Enter Flight Number: ");

scanf("%s", flightNumber);

printf("Enter Seat Number to delete ticket: ");

scanf("%d", &seatNumber);

deleteTicket(&passengers, flightNumber, seatNumber);

}

break;

case 5:

printf("Exiting program.\n");

break;

default:

printf("Invalid choice. Please try again.\n");

}

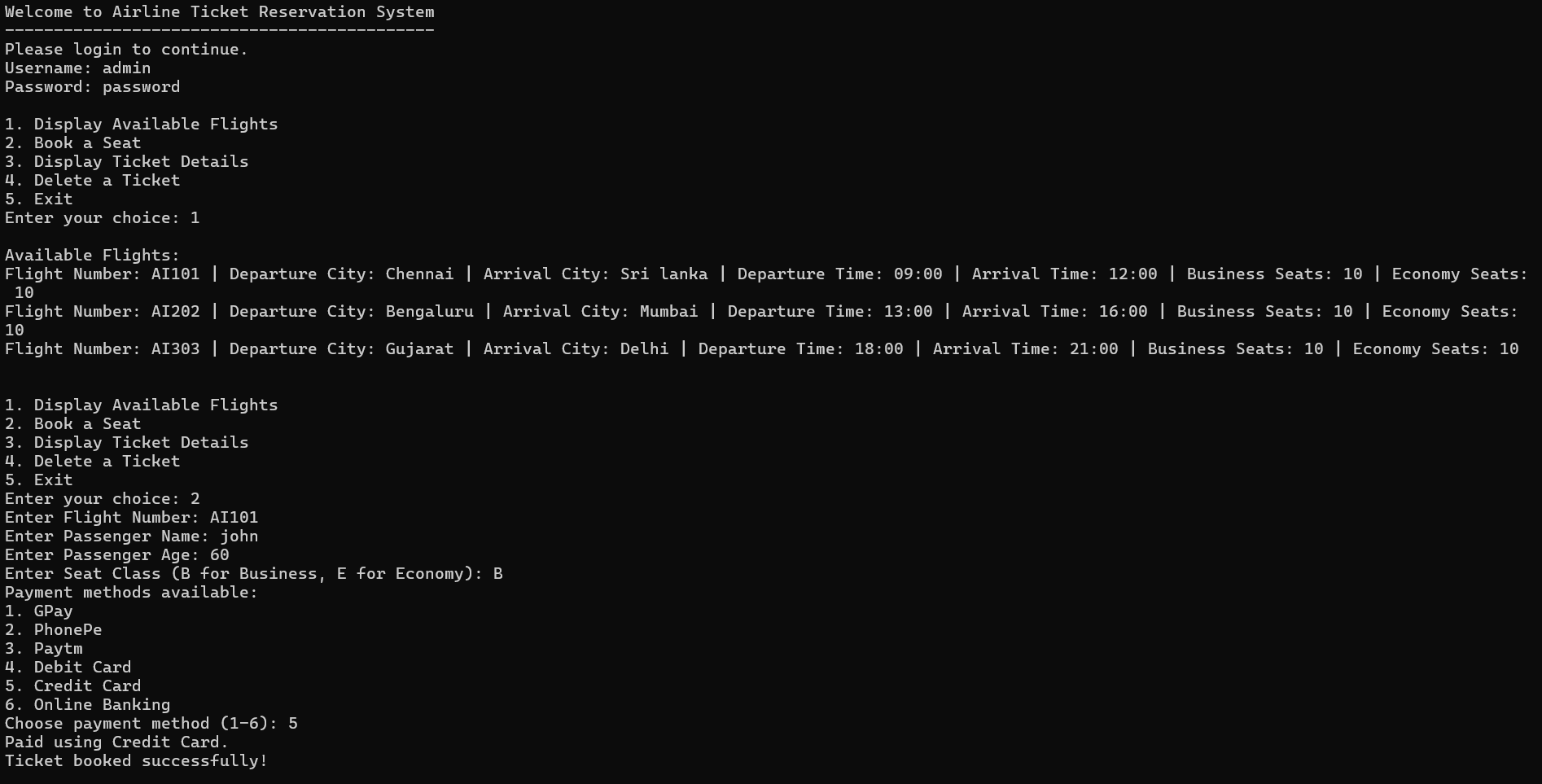
} while (choice != 5);

freeFlights(flights);

freePassengers(passengers);

return 0;

}

1. ** Screenshots**



**Conclusion**

* Modular Structure: The code is organized into functions, each responsible for a specific task such as creating flights, booking seats, displaying flight information, and managing passengers.
* Dynamic Memory Allocation: The code dynamically allocates memory for flights and passengers using malloc, ensuring efficient memory usage and flexibility in managing data structures.
* Data Structures: Two custom data structures, Flight and Passenger, are defined using struct to encapsulate relevant information about flights and passengers, respectively.
* Linked List Implementation: The code employs linked lists to manage flights and passengers, allowing for easy insertion, deletion, and traversal of data elements.
* User Input Handling: User input is captured using scanf, enabling interactive interactions such as logging in, choosing options from a menu, and providing necessary details for booking seats or deleting tickets.
* Error Handling: The code includes error handling mechanisms such as checking for memory allocation failures and invalid input to ensure robustness and prevent unexpected crashes.
* Encapsulation and Information Hiding: Information about flights and passengers is encapsulated within their respective data structures, promoting data integrity and abstraction.
* User Authentication: The program verifies user credentials (username and password) before granting access to the system, enhancing security and restricting unauthorized usage.
* Interactive Interface: The program offers a user-friendly interface with a menu-driven approach, allowing users to perform various operations such as viewing available flights, booking seats, displaying ticket details, and deleting tickets.
* Resource Management: The code appropriately frees dynamically allocated memory for flights and passengers using free to prevent memory leaks and ensure efficient resource management.

Overall, the code demonstrates basic functionalities of an airline ticket reservation system, showcasing principles of data structures, memory management, user interaction, and error handling in a C programming context.

**Bibliography**

1. Data Structures using C Second Edition. Author: Rema Thareja
2. GeeksforGeeks using time library function.

Link: <https://www.geeksforgeeks.org/time-function-in-c/>

1. Official C documentation link: https://devdocs.io/c/