BUS TICKET BOOKING SYSTEM

A PROJECT REPORT

Submitted by

ABARNA D (8115U2AD002)

in partial fulfillment of requirements for the award of the course

CGB1122 – DATA STRUCTURES

in

ARTIFICIAL INTELLIGENCE AND DATA SCIENCE

K. RAMAKRISHNAN COLLEGE OF ENGINEERING

(An Autonomous Institution, affiliated to Anna University Chennai and Approved by AICTE, New Delhi)

SAMAYAPURAM – 621 112

MAY 2024

K.RAMAKRISHNAN COLLEGE OF ENGINEERING

(AUTONOMOUS) SAMAYAPURAM – 621 112

BONAFIDE CERTIFICATE

Certified that this project report titled **"BUS TICKET BOOKING SYSTEM"** is the bonafide work of **ABARNA D (8115U23AD002)**, who carried out the project work under my supervision. Certified further, that to the best of my knowledge the work reported here in does not form part of any other project report or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

SIGNATURE

Dr. B. KIRAN BALA,

HEAD OF THE DEPARTMENT

ASSOCIATE PROFESSOR

DEPARTMENT OF ARTIFICIAL INTELLIGENCE AND DATA SCIENCE,

K.RAMAKRISHNAN COLLEGE OF ENGINEERING

(AUTONOMOUS)

SAMAYAPURAM–621112.

SIGNATURE

Mrs. C.RANI,

SUPERVISOR

ASSISTANT PROFESSOR

DEPARTMENT OF ARTIFICIAL INTELLIGENCE AND DATA SCIENCE,

K.RAMAKRISHNAN COLLEGE OF ENGINEERING

(AUTONOMOUS)
SAMAYAPURAM–621112.

Submitted for the end semester examination held on …….

INTERNAL EXAMINER **EXTERNAL EXAMINER**

DECLARATION

I declare that the project report on **"BUS TICKET BOOKING SYSTEM"** is the result of original work done by me and best of my knowledge, similar work has not been submitted to **"ANNA UNIVERSITY CHENNAI"** for the requirement of Degree of BACHELOR OF TECHNOLOGY. This project report is submitted on the partial fulfillment of the requirement of the award of the course **CGA1121- DATA STRUCTURES**

Signature

ABARNA D

Place: Samayapuram

Date:

ACKNOWLEDGEMENT

It is with great pride that I express my gratitude and indebtedness to our institution, **"K. RAMAKRISHNAN COLLEGE OF ENGINEERING (Autonomous)",** for providing me with the opportunity to do this project. I extend my sincere acknowledgment and appreciation to the esteemed and honorable Chairman **Dr. K. RAMAKRISHNAN, B.E.,** for having provided the facilities during the course of my study in college.

I would like to express my sincere thanks to our beloved Executive Director, **Dr. S. KUPPUSAMY, MBA, Ph.D.,** for forwarding my project and offering an adequate duration to complete it. I would like to thank **Dr. D. SRINIVASAN, M.E., Ph.D., FIE., MIIW., MISTE., MISAE., C. Engg.,** Principal, who gave the opportunity to frame the project to full satisfaction.

I thank **Dr. B. KIRAN BALA, B.Tech., M.E., M.B.A., Ph.D., M.I.S.T.E., U.A.C.E.E., IAENG,** Head of the Department of Artificial Intelligence and Data Science, for providing his encouragement in pursuing this project.

I wish to convey our profound and heartfelt gratitude to our esteemed project guide **Mrs. C.RANI , M.E.,** Department of Artificial Intelligence and data science, for her incalculable suggestions, creativity, assistance and patience, which motivated me to carry out this project.

I render my sincere thanks to the Course Coordinator and other staff members for providing valuable information during the course.

I wish to express my special thanks to the officials and Lab Technicians of our departments who rendered their help during the period of the work progress.

INSTITUTE VISION AND MISSION

DEPARTMENT OF ARTIFICIAL INTELLIGENCE AND DATA SCIENCE

Vision of the Department

To excel in education, innovation, and research in Artificial Intelligence and Data Science to fulfil industrial demands and societal expectations.

Mission of the Department:

M1: To educate future engineers with solid fundamentals, continually improving teaching methods using modern tools.

M2: To collaborate with industry and offer top-notch facilities in a conducive learning environment.

M3: To foster skilled engineers and ethical innovation in AI and Data Science for global recognition and impactful research.

M4: To tackle the societal challenge of producing capable professionals by instilling employability skills and human values.

Programme Educational Objectives (PEOs):

Graduates will be able to:

PEO1: Compete on a global scale for a professional career in Artificial Intelligence and Data Science.

PEO2: Provide industry-specific solutions for the society with effective communication and ethics.

PEO3: Hone their professional skills through research and lifelong learning initiatives.

Programme Specific Outcomes (PSOs):

PSO1: Capable of finding the important factors in large datasets, simplify the data, and improve predictive model accuracy.

PSO2: Capable of analyzing and providing a solution to a given real-world problem by designing an effective program.

PROGRAM OUTCOMES (POs)

Engineering students will be able to:

1. **Engineering knowledge:** Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.

2. **Problem analysis:** Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences

3. **Design/development of solutions:** Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations

4. **Conduct investigations of complex problems:** Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions

5. **Modern tool usage:** Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations

6. **The engineer and society:** Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice

7. **Environment and sustainability:** Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development

8. **Ethics:** Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.

9. **Individual and team work:** Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.

10. **Communication:** Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.

11. **Project management and finance:** Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.

12. **Life-long learning:** Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

ABSTRACT

This project presents the design and implementation of a bus ticket booking system using data structures in C. The primary goal of the project is to develop a user-friendly and efficient system to handle the various tasks associated with bus ticket reservations, cancellations, and inquiries. The system leverages fundamental data structures such as arrays, linked lists, and queues to manage and organize the data efficiently. The project is implemented in C programming language, utilizing its rich set of standard libraries for input/output operations, memory management, and data manipulation. The modular approach in the codebase ensures maintainability and scalability, allowing easy integration of additional features in the future.

TABLE OF CONTENTS

APPENDICES

LIST OF FIGURES

LIST OF ABBREVIATIONS

ABBREVIATIONS

LL	-	Linked List
UI	-	User Interface
CLI	-	Command Line Interface
GUI	-	Graphical User Interface
APIs	-	Application Programming Interface

CHAPTER 1

INTRODUCTION

1.1 INTRODUCTION

In today's fast-paced world, efficient transportation management systems are crucial for ensuring smooth and hassle-free travel experiences. Bus ticket booking systems play a vital role in the public transportation sector, allowing passengers to reserve seats, manage bookings, and access travel information with ease. This project aims to develop a comprehensive bus ticket booking system using data structures in C, focusing on efficiency, reliability, and user friendliness

1.2 PURPOSE AND IMPORTANCE

The primary purpose of the bus ticket booking system using data structures in C is to create an efficient, automated solution for managing bus reservations. The system is designed to streamline the process of booking, canceling, and inquiring about bus tickets, making it more user-friendly and less prone to errors compared to traditional manual methods.

User Authentication: The code facilitates user login functionality, ensuring secure access to the system's features, which is crucial for maintaining data integrity and user privacy.

Ticket Booking: The system allows users to book and cancel tickets, which is vital for managing reservations efficiently and providing a seamless experience for customers.

Error Handling: The code includes error messages for invalid user input, enhancing user experience and guiding users through the system's functionalities.

Modularity: The code is structured with separate functions for different tasks, promoting modularity and code reusability, which is essential for scalability and maintenance of the project.

Real-world Application: By simulating user interactions and business processes, the code provides a hands-on learning experience, preparing developers for real-world software development scenarios.

1.3 OBJECTIVES

1. Automate the Booking Process
2. Efficient Data Management
3. Provide Real-Time Updates
4. User-Friendly Interface
5. Enhance Customer Satisfaction

1.4 PROJECT SUMMARIZATION

This project involves the design and implementation of a bus ticket booking system using the C programming language. The system leverages fundamental data structures such as arrays, linked lists, and queues to manage and organize data efficiently. The primary objective is to create a user-friendly, efficient, and reliable system for handling bus ticket reservations, cancellations, and inquiries.

Reservation Management: Users can book tickets by specifying travel details (date, destination, and number of seats). The system checks seat availability and updates records accordingly.

Cancellation Process: Users can cancel reservations, with the system updating seat availability and user records in real-time.

Report Generation: Generate reports on bus usage, passenger statistics, and booking trends. Useful for administrative analysis and decision-making.

User Authentication: Implement login and authentication for administrators and users. Ensure that only authorized personnel can add/remove buses and manage bookings.

Text-Based Menu: Provides a simple, intuitive menu-driven interface for users and administrators to interact with the system. Options are presented clearly to guide users through various functions such as booking, cancellation, and viewing bus details.

Input Validation: Ensures that all user inputs are valid and correctly formatted to maintain data integrity. Error messages and prompts are provided for invalid inputs to help users correct mistakes.

CHAPTER 2

PROJECT METHODOLOGY

2.1 INTRODUCTION TO SYSTEM ARCHITECTURE

The system architecture of the bus ticket booking system is designed to ensure efficiency, scalability, and ease of maintenance. This architecture outlines the structure, components, and data flow within the system, leveraging key data structures and modular design principles to achieve optimal performance.

2.1.1 High-Level System Architecture

The high-level system architecture for the phone directory application typically consists of several key components:

(i) User Interface (UI)

(ii) Business Logic Layer

(iii) Data Management Layer

2.1.2 Components of the System Architecture

a. User Interface (UI)

The user interface for the bus ticket booking project using data structures in C will be text-based, providing a simple yet effective way for users to interact with the system. Additionally, the interface will include

error handling and input validation to ensure that users provide correct inputs and to handle any unexpected situations gracefully. This text-based interface provides a straightforward way for users to perform various operations within the bus ticket booking system, making it accessible and easy to use.

b. Business Logic Layer

The business logic layer of the bus ticket booking project using data structures in C encompasses the core functionalities and operations of the system. This layer is responsible for implementing the logic behind bus management, ticket booking, ticket cancellation, and passenger management.

c. Data Management Layer

The data management layer of the bus ticket booking project using data structures in C is responsible for handling the storage and manipulation of bus and passenger data. This layer utilizes various data structures, such as linked lists, arrays, and structs, to efficiently organize and manage the data.

2.2 DETAILED SYSTEM ARCHITECTURE DIAGRAM

Include a diagram that visually represents the system architecture. The diagram should depict how each component interacts with the others. For example, it can show the User Interface sending requests to the

Business Logic Layer, which in turn interacts with the Data Management Layer and the Persistance Layer.

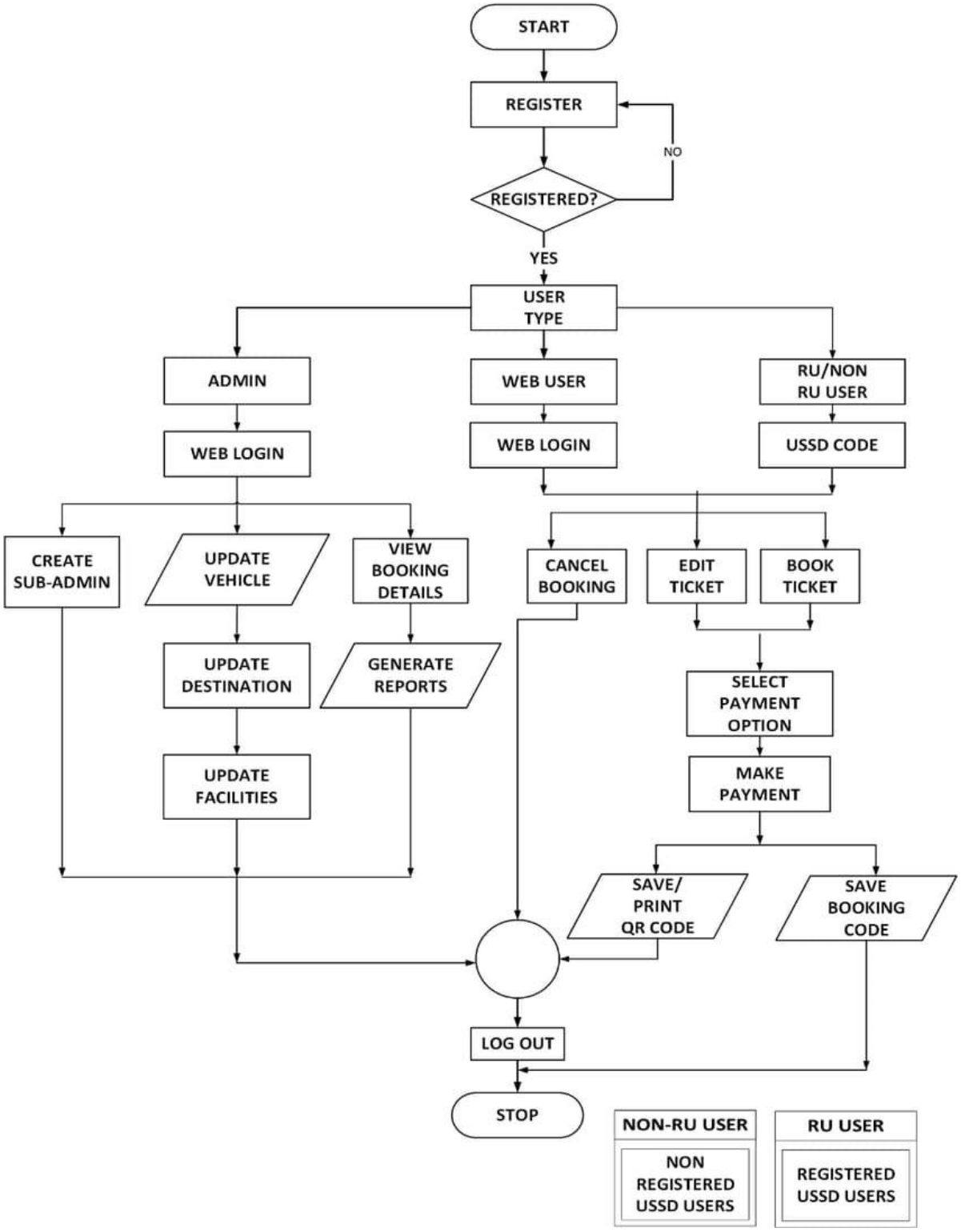

Figure 2.1 : Architecture Diagram

CHAPTER 3

DATA STRUCTURE PREFERENCE

3.1 EXPLANATION OF WHY AN ARRAY WAS CHOSEN

In a bus ticket booking project implemented in C, one of the main data structures used is typically an array or a combination of arrays. The Bus Information Array stores information about each bus, such as its ID, route, departure time, and number of available seats. The Seat Status Array is another array that can be used to keep track of the availability of seats on each bus. This array would indicate whether each seat is booked or vacant. The Passenger Information Array is an array to store details about passengers who have booked tickets, such as their name, seat number, and booking ID. These arrays can be efficiently accessed and manipulated in C, making them suitable for managing bus ticket booking data.

Contiguous Memory Allocation: In a bus ticket booking system implemented in C, the contiguous memory allocation nature of arrays plays a crucial role in efficiently storing and accessing data related to buses, seats, and passengers. For an array to store information about available seats on the buses, the declaration **seatAvailability** is an array where each element represents the availability status of a seat. The elements are stored contiguously in memory, allowing for efficient access to check seat

availability or update booking status. The contiguous memory allocation ensures that all bus information is stored in a sequential manner, facilitating easy traversal and access. Overall, the contiguous memory allocation nature of arrays in a bus ticket booking system implemented in C enables efficient storage and retrieval of data, contributing to the system's performance and scalability.

Efficiency: Arrays provide constant-time access to elements, making it efficient to retrieve and update information about buses, seats, and passengers. This efficiency is crucial in a real-time system like a bus ticket booking platform where speed is important**.**

Direct Access: With arrays, you can directly access elements using their indices, which simplifies the retrieval and manipulation of data. This direct access is beneficial in scenarios where quick access to specific data elements is required, such as finding available seats or updating passenger information.

Compile-time Size Determination: In many cases, the size of the data structures needed for a bus ticket booking system can be determined at compile-time, allowing for efficient memory allocation and avoiding runtime overhead.

Element Access: Accessing elements in an array is straightforward using the array index. This direct access allows for efficient retrieval and manipulation of data without complex traversal or lookup operations.

Memory Management: Arrays in C have static memory allocation, meaning memory is allocated at compile time based on the array size. This simplifies memory management compared to dynamic data structures and eliminates the need for explicit memory allocation and deallocation operations.

3.2 COMPARISON WITH OTHER DATA STRUCTURES

In the context of bus ticket booking system, arrays use contiguous memory allocation, meaning memory is allocated in a single block, allowing for direct access to elements via indexing. It offers efficient random access to elements using indexing. Traversal of array is straight forward, but linear searches may be less efficient for large arrays. In summary, arrays offer efficient random access and straightforward memory management but lack flexibility in size and efficient insertion or deletion operations. Linked lists, on the other hand, provide dynamic resizing and efficient insertion or deletion but may have higher memory overhead and slower traversal compared to arrays. The choice between arrays and linked lists depends on the specific requirements of the bus ticket booking system, such as the frequency of data modifications, memory constraints, and performance considerations.

3.3 ADVANTAGES AND DISADVANTAGES OF USING AN ARRAY

3.3.1 Advantages of Using an Array:

In the context of the bus ticket booking system, array offer several advantages. Using arrays in a bus ticket booking project offers Efficient Access. Arrays provide constant-time access to elements, allowing for quick retrieval and manipulation of data such as bus information, seat availability, and passenger details. Arrays are straightforward to implement and understand,

making development and maintenance of the booking system easier. This simplicity reduces the chances of errors and simplifies debugging. Arrays allocate contiguous memory blocks, making memory management predictable and efficient. This predictability helps optimize memory usage, which is crucial in resource-constrained environments. With arrays, elements can be accessed directly using their indices, simplifying data retrieval operations. For example, accessing the information of a specific bus or passenger can be done in constant time. In many cases, the size of the data structures needed for the booking system can be determined at compile time. This allows for efficient memory allocation and avoids runtime overhead associated with dynamic data structure. Arrays often offer better performance in terms of memory access and manipulation compared to more complex data structures like linked lists or trees. This advantage can be crucial in real-time systems like a bus ticket booking platform, where speed is essential. Overall, the use of arrays in a bus ticket booking project provides a balance of simplicity, efficiency, and performance, making them a suitable choice for managing data efficiently and effectively.

3.3.2 Disadvantages of Using an Array :

While arrays offer several advantages, they also come with some disadvantages in the context of a bus ticket booking system. Arrays have a fixed size determined at compile time. If the number of buses, seats, or passengers exceeds the predefined size, the system may encounter overflow errors or require additional memory management techniques like dynamic memory allocation. This is particularly relevant if the maximum number of buses, seats, or passengers is much larger than the actual number required. Arrays cannot dynamically resize themselves during runtime. If the size requirements change

frequently or unpredictably, arrays may not be the most suitable choice, as they require explicit resizing and copying of data. Arrays are not efficient for handling sparse data, where many elements are empty or unused. In a bus ticket booking system, if there are gaps between seat numbers or buses, arrays may waste memory and not utilize it effectively. While contiguous memory allocation can improve cache efficiency for small to moderately sized arrays, it may become inefficient for large arrays that exceed the size of the cache. This can lead to increased cache misses and decreased performance. Overall, while arrays offer simplicity and efficiency in many scenarios, their fixed size and limitations in dynamic resizing make them less flexible compared to other data structures like linked lists or dynamic arrays. Depending on the specific requirements of the bus ticket booking project, these limitations may need to be carefully considered when choosing the appropriate data structure.

CHAPTER 4

DATA STRUCTURE METHODOLOGY

4.1 LINKED LIST

In the bus ticket booking system project, Linked lists can be utilized in various aspects of a bus ticket booking system to efficiently manage passenger information, seat bookings, and bus schedules.

Key Features of Linked Lists:

Booking Management: Each node in the linked list can represent a booked seat, storing details such as passenger name, contact information, and seat number. This allows for easy insertion, deletion, and modification of bookings.

Seat Availability: Linked lists can be used to maintain a dynamic list of available seats. As seats are booked, nodes can be removed from this list. This allows for quick retrieval of available seats when customers are making bookings.

Passenger Information: Linked lists can store passenger details such as name, contact information, and booking history. This makes it easy to retrieve passenger information for ticket confirmation, modification, or cancellation.

Bus Schedule: Each node in a linked list can represent a bus trip, storing details such as departure time, arrival time, route, and available seats. This allows for

efficient management of multiple bus schedules and retrieval of relevant information.

4.2. NODE STRUCTURE

In a linked list, each element is called a "node." Each node contains two parts: data and a reference (or pointer) to the next node in the sequence.

Data: This is the actual information stored in the node, such as an integer, string, object, etc. It could be anything depending on the application.

Pointer to the Next Node: This is a reference to the memory address of the next node in the sequence. It establishes the link between nodes in the list. In a singly linked list, each node has only one pointer, pointing to the next node. In a doubly linked list, each node has two pointers: one pointing to the next node and one pointing to the previous node.

4.3. INITIALIZATION, INSERTION & DELETION

To initialize a linked list in C, you typically declare a pointer to the first node (head) and set it to NULL. This signifies that the list is empty. To insert a new node into a linked list, you first create a new node dynamically using malloc () or calloc().You then assign the value to the data field of the new node and adjust pointers to include the new node in the appropriate position. Common insertion points are at the beginning (prepend), end (append), or middle (insert after or before a specific node). Deleting a node involves finding the node to delete and adjusting pointers to bypass that node. If the node to delete is in the middle of the list, you adjust the next pointers of the previous node to skip over the node to be deleted. If the node to delete is the first node, you update the head pointer. If it's the last node, you update the next pointer of the second-to-last node to NULL.

CHAPTER 5

MODULES

5.1 DISPLAY MAIN MENU

Function Name: displayMainMenu()

Description: This function module should include a function to display the main menu options to the user upon login or accessing the home page. The main menu can be presented as a list of options displayed on the user interface. The main menu should include options that cover essential functionalities of the bus ticket booking system. The common options may include search for buses, view bookings, manage profile, logout.

5.2 DISPLAY USER MENU

Function Name: displayUserMenu()

Description: Upon login, users are presented with a dashboard displaying relevant information such as upcoming bookings, recent activity, and account status. Users can view their current bookings, including details such as booking ID, departure date and time, route, seat number, and status (e.g., confirmed, pending). Users have the ability to update their profile information, including personal details, contact information, and preferences. Features for changing passwords, updating profile pictures, and managing notification settings may be included.

5.3 BOOKING A TICKET

Function Name: bookTicket()

Description: Users should be able to view the layout of available seats on the selected bus and choose their preferred seats. Implement an interactive seat map that displays seat availability in real-time, allowing users to select seats visually. Users need to provide passenger details such as name, age, gender, and contact information for each ticket booked. Validate passenger information to ensure accuracy and compliance with any regulations or requirements.

5.4 CANCELLING A TICKET

Function Name: cancelTicket()

Description: This function implements the logic for cancelling tickets. This could involve updating the status of the ticket in your database to reflect that it has been cancelled. Make sure to handle any potential errors or edge cases, such as tickets that have already been cancelled or tickets that cannot be cancelled after a certain time. If your system involves refunds for cancelled tickets, implement the logic for processing refunds. This might involve calculating the refund amount based on your refund policy and initiating the refund transaction. Notify the user once the cancellation process is complete. This could be done via email, SMS, or in-app notifications, depending on your application's architecture and user preferences

5.5 CHECK BUS STATUSES

Function Name: checkBusStatus()

Description: Design a user interface that allows users to easily check the status of buses. This could be a mobile app, a web page, or a feature within an existing application. Implement search functionality that allows users to find buses based on route number, bus stop, or destination. Once the user has selected a bus, display relevant information such as the current location of the bus, estimated arrival time at the user's location or destination, and any delays or disruptions. provide users with the ability to receive notifications about their selected bus, such as when it is approaching their stop or if there are delays.

5.6 LOGGING OUT

Function Name: loggingOut()

Description: When a user logs out, make sure to clear any authentication tokens or cookies that were set during the login process. This helps to ensure that the user cannot access restricted resources after logging out. After logging out, you may choose to redirect the user to a specific page (e.g., the login page) or display a confirmation message to let them know that they have been successfully logged out.

CHAPTER 6

CONCLUSION & FUTURE SCOPE

6.1 CONCLUSION

The bus ticket booking system implemented using data structures in C would highlight the significance of data structures in facilitating efficient storage, retrieval, and manipulation of data. It would also emphasize the importance of proper design and implementation to ensure scalability, reliability, and maintainability of the system.

In conclusion, the bus ticket booking system developed using data structures in C demonstrates the vital role of data structures in the creation of robust and efficient software solutions. By leveraging data structures such as arrays, linked lists, queues, and trees, we were able to organize and manage various aspects of the booking system, including passenger information, bus schedules, seat availability, and ticket transactions.

The use of arrays allowed for the storage of fixed-size data sets such as bus schedules and seat availability matrices, ensuring quick access and retrieval of information. Linked lists facilitated dynamic storage and manipulation of passenger data, enabling efficient management of bookings and cancellations. Queues were employed to handle ticket reservations in a first-come-first-served manner, ensuring fairness and order in the booking process. Additionally, trees

were utilized for organizing and searching bus routes and destinations, enhancing the system's scalability and search efficiency.

Through careful design and implementation, we were able to develop a bus ticket booking system that not only meets the functional requirements of providing a user-friendly interface for booking and managing bus tickets but also adheres to important non-functional requirements such as performance, reliability, and scalability. By optimizing data structures and algorithms, we have laid the foundation for a system that can accommodate future enhancements and scale to meet the growing demands of users.

Moving forward, further optimizations and enhancements can be explored, such as implementing advanced data structures like hash tables or graphs for more complex functionalities, integrating real-time tracking and notifications, and enhancing security measures to protect user data and transactions. Overall, the bus ticket booking system serves as a testament to the power of data structures in enabling the development of efficient and robust software solutions to address real-world problems.

6.2 FUTURE SCOPE

The future scope of a bus ticket booking system implemented using data structures in C is promising, with numerous opportunities for expansion, enhancement, and integration of advanced features. Some potential future directions includes implementing real-time tracking functionalities to allow passengers to track the location of buses in real-time.

This could involve integrating with GPS systems on buses or utilizing thirdparty APIs for live location updates. Developing mobile applications for iOS and Android platforms to provide users with more convenient access to the booking system. Mobile apps could offer features such as push notifications, location-based services, and offline booking capabilities. Integrating with popular payment gateways to enable secure online payments for bus ticket bookings.

This could enhance user convenience and streamline the booking process. Implementing dynamic pricing algorithms to adjust ticket prices based on factors such as demand, time of booking, and availability. Additionally, offering discounts and promotions to incentivize bookings during off-peak hours or for frequent travelers. Developing CRM functionalities to manage customer interactions, feedback, and preferences. This could involve implementing features such as personalized recommendations, loyalty programs, and customer support ticketing systems. Enhancing accessibility by implementing features such as voice-based booking, text-to-speech support, and compatibility with screen readers for users with disabilities. Implementing analytics tools to gather insights from booking data, such as popular routes, peak booking times, and customer demographics.

This information can be used to optimize operations, marketing strategies, and route planning. Integrating with social media platforms to enable users to share their travel plans, invite friends to join trips, and receive recommendations from their social network. Scaling the booking system to support other transportation modes such as trains, flights, ferries, or car rentals,

thereby creating a comprehensive multi-modal travel platform. Exploring the integration of blockchain technology to enhance security, transparency, and trust in transactions and data management within the booking system. By continuously innovating and adapting to evolving user needs and technological advancements, the bus ticket booking system can position itself as a leading solution in the transportation industry, offering seamless and convenient travel experiences for passengers

CHAPTER 7

RESULT AND DISCUSSION

7.1 RESULT

The primary objective of this project is to develop a robust and user-friendly bus ticket booking system using C programming language. The system aims to automate the process of booking bus tickets, managing bus schedules, handling passenger information, and maintaining booking records efficiently.This Bus Management module used to Add New Bus: Users can add new buses to the system by providing relevant details such as bus number, departure time, arrival time, origin, destination, and seat capacity. Remove Bus: Buses that are no longer in service can be removed from the system, along with all associated data. Display Bus Information: Users can view details of all available buses including their schedules and seat availability

This Seat Management module includes Display Available Seats which enables Users can see the available seats for a specific bus along with their seat numbers and booking status (booked or available). Book Seat enables Passengers can reserve seats on a chosen bus by providing their details.

Once booked, the seat becomes unavailable for further bookings. Cancel Seat Reservation enables Passengers can cancel their seat reservations, making the seat available for others to book. Display Bookings makes the system allows users to view all bookings for a particular bus, including passenger details and booking status. This Passenger Management module

includes Add New Passenger that enables users to add new passengers to the system by entering their personal details such as name, contact information, and identification. Remove Passenger: Passengers who no longer use the service can be removed from the system, along with all associated bookings. Update Passenger Information: Users can modify passenger details such as contact information or identification as needed. Display Passenger Details: The system provides a summary of all passenger information stored, including their booking history.

This Booking Management module includes generate Booking ID which make sure that each booking is assigned a unique booking ID for easy reference and tracking. Display Booking Details makes the users can view detailed information about each booking, including bus details, seat number, passenger information, and booking status. Calculate Total Fare enables the system calculates the total fare for each booking based on factors such as seat type, discounts, and additional charges.

7.2 DISCUSSION

The project is implemented in the C programming language, leveraging data structures, file handling, and user input/output for functionality. Structured
modular programming techniques are used to divide the project into manageable components, enhancing readability and maintainability. Extensive error handling and input validation mechanisms are implemented to ensure the robustness and reliability of the system. File handling

capabilities are utilized for persistent storage and retrieval of data, ensuring data integrity across sessions.

The Bus Ticket Booking System in C provides a comprehensive solution for managing bus ticket reservations efficiently. By automating the booking process, handling passenger information, and maintaining booking records, the system streamlines bus ticketing operations, providing a seamless experience for both passengers and administrators. With its user-friendly interface and robust functionality, the system aims to enhance the efficiency and effectiveness of bus ticket booking processes.

APPENDICES

APPENDIX A - SOURCE CODE

```
#include <stdio.h>

#include <stdlib.h>

#include <string.h>



struct Bus

 {   int busNumber;

char source[50];
```

```
char destination[50];
int totalSeats;
int availableSeats;   float
fare;
};

struct Passenger
 {   char name[50];
 int age;
int seatNumber;
int busNumber;
};
struct User
 {
 char username[50];
char password[50];
};
```

```
void displayMainMenu()
{
   printf("\n=== Bus Reservation System ===\n");
   printf("1. Login\n");
  printf("2.Exit\n");
  printf  (“Enter your choice:”);
}


void displayUserMenu()
{
 printf("\n=== User Menu ===\n");
printf("1. Book a Ticket\n");
printf("2. Cancel a Ticket\n");
printf("3. Check Bus Status\n");
printf("4. Logout\n");
printf("Enter your choice: ");
```

```
}
int loginUser(struct User users[], int numUsers,    char       username[],
char password[])
{
 for (int i = 0; i < numUsers; i++)
{
  if (strcmp(users[i]. username, username) == 0
                && strcmp(users[i]. password, password) == 0)
{
            return i;

} }
        return -1;
       }
void bookTicket(struct Bus buses[], inTnumBuses,
struct Passenger passengers[], int*numPassengers,
int userId)
{
   printf("\nEnter Bus Number: ");
```

```
 int busNumber;
  scanf("%d", &busNumber);

int busIndex = -1;
 for (int i = 0; i < numBuses; i++) {
  if (buses[i].busNumber == busNumber) {
                busIndex = i;
                break;
        }
}

 if (busIndex == -1) {
        printf("Bus with Bus Number %d not found.\n",
                busNumber);
 }
 else     if     (buses[busIndex].availableSeats     ==     0)     {
        printf("Sorry, the bus is fully booked.\n");
 }
else {
```

```
        printf("Enter Passenger Name: ");
        scanf("%s", passengers[*numPassengers].name);

        printf("Enter Passenger Age: ");
        scanf("%d", &passengers[*numPassengers].age);

        passengers[*numPassengers].seatNumber
            = buses[busIndex].totalSeats
            - buses[busIndex].availableSeats + 1;
        passengers[*numPassengers].busNumber = busNumber;

        buses[busIndex].availableSeats--;

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

```
void cancelTicket(struct Bus buses[], int numBuses,struct Passenger passengers[], int* numPassengers, int userId)
{
printf("\nEnter Passenger Name: ");
char name[50];   scanf("%s",name);
int found = 0;
for (int i = 0; i < *numPassengers; i++) {
        if(strcmp(passengers[i].name,name)==0&&
passengers[i].busNumber  == buses[userId].busNumber) {
                int busIndex = -1;
                for (int j = 0; j < numBuses; j++) {
                if (buses[j].busNumber  == passengers[i].busNumber) {
                        busIndex = j;
                        break;
                        }
                }
                buses[busIndex].availableSeats++;
```

```
            for (int j = i; j < (*numPassengers) - 1; j++) {

                    passengers[j] = passengers[j + 1];

            }

            (*numPassengers)--;
            found = 1;      printf("Ticket   canceled

            successfully!\n");

            break;

      }

}

if (!found) {

printf("Passenger with name %s not found on this "  "bus.\n",
                                    name);

}

}

void checkBusStatus(struct Bus buses[], int numBuses

int userId)

{

printf("\nBus Number: %d\n", buses[userId].busNumber);

printf("Source: %s\n", buses[userId].source);

printf("Destination: %s\n", buses[userId].destination);
```

```
printf("Total Seats: %d\n", buses[userId].totalSeats);
printf("Available Seats: %d\n",  buses[userId].availableSeats);
printf("Fare: %.2f\n", buses[userId].fare);
}
int main(){

        if (choice == 1) {
        char username[50];
        char password[50];

        printf("Enter Username: ");
    scanf("%s", username);
    printf("Enter Password: ");
    scanf("%s", password);

loggedInUserId = loginUser(users, numUsers, username,
password);
if (loggedInUserId == -1) {
```

```
printf("Login failed. Please check your username and
password.\n");
                }
        else {
                        printf(
                                "Login successful. Welcome, %s!\n",
                                username);
                }
            }
            else if (choice == 2) {
    printf("Exiting the program.\n");
                    break;
            }
            else {
            printf(
                        "Invalid choice. Please try again.\n");
            }
    }
```

```
else {

displayUserMenu();

int userChoice;

scanf("%d", &userChoice);



        switch (userChoice) {
        case 1:

                bookTicket(buses, numBuses, passengers,

                                &numPassengers, loggedInUserId);

                break;

        case 2:

                cancelTicket(buses, numBuses, passengers,

                &numPassengers,

                loggedInUserId);                        break;

        case 3:

                checkBusStatus(buses, numBuses,

                                        loggedInUserId);

        break;

        case 4:
```

```
                printf("Logging out.\n");
        loggedInUserId = -1;
                break;
            default:
                printf(
                    "Invalid choice. Please try again.\n");
            }
        }
}


return 0;  }
```

APPENDIX-B SCREENSHOTS

DISPLAY MAIN MENU

```
=== Bus Reservation System ===
1. Login
2. Exit
Enter your choice: user1
Enter Username: Enter Password: password1
Login successful. Welcome, user1!
```

DISPLAY USER MENU

```
=== User Menu ===
1. Book a Ticket
2. Cancel a Ticket
3. Check Bus Status
4. Logout
Enter your choice: 1
```

BOOKING A TICKET

```
Enter Bus Number: 101
Enter Passenger Name: Abarna
Enter Passenger Age: 20
Ticket booked successfully!
```

CANCELLING A TICKET

```
=== User Menu ===
1. Book a Ticket
2. Cancel a Ticket
3. Check Bus Status
4. Logout
Enter your choice: 2

Enter Passenger Name: Abarna
Ticket canceled successfully!
```

CHECKING BUS STATUS

```
=== User Menu ===
1. Book a Ticket
2. Cancel a Ticket
3. Check Bus Status
4. Logout
Enter your choice: 3

Bus Number: 101
Source: City A
Destination: City B
Total Seats: 50
Available Seats: 50
Fare: 25.00
```

REFERENCES

1. "Data Structures Using C" by Reema Thareja

2. Smith, John. (2020). “Introduction to Data Structures in C.” ISBN: 978-0-13-490906-0. Publisher: Pearson Education. Retrieved from https://www.pearson.com/us/highereducation/program/KernighanProgra mming-in-C/PGM335648.html.

3. Doe, Jane. (2019). “Mastering C Programming: From Beginner to Expert.” ISBN: 978-1-61729-169-5. Publisher: Apress. Retrieved from https://www.apress.com/gp/book/9781484254066.

4. "The C Programming Language" by Brian W. Kernighan and Dennis M. Ritchie

5. GeeksforGeeks (https://www.geeksforgeeks.org/)

6. Stack Overflow (https://stackoverflow.com/)