BUS RESERVATION SYSTEM

A PROJECT REPORT

Submitted by

NAVEENA R(2303811710422104)

in partial fulfillment of requirements for the award of the course

CGA1121 – DATA STRUCTURES

in

COMPUTER SCIENCE AND ENGINEERING

K. RAMAKRISHNAN COLLEGE OF TECHNOLOGY

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

SAMAYAPURAM – 621 112

May, 2024

K. RAMAKRISHNAN COLLEGE OF TECHNOLOGY

(AUTONOMOUS)

SAMAYAPURAM – 621 112

BONAFIDE CERTIFICATE

Certified that this project report titled **"BUS RESERVATION SYSTEM"** is the bonafide work of **NAVEENA R (2303811710422104),** 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.

[illegible]

SIGNATURE

Mrs.A.Delphin Carolina Rani, M.E.,Ph.D.,

HEAD OF THE DEPARTMENT

PROFESSOR

Department of CSE

K. Ramakrishnan College of Technology (Autonomous)

Samayapuram–621112.

[illegible]

SIGNATURE

Mr. D.P.DEVAN B.E., M.E.,

SUPERVISOR

ASSISTANT PROFESSOR

Department of CSE

K. Ramakrishnan College of Technology (Autonomous)

Samayapuram–621112.

Submitted for the viva-voce examination held on 15/06/2024

[illegible]

INTERNAL EXAMINER

DECLARATION

I declare that the project report on **“BUS RESERVATION SYSTEM”** is the result of original work done by us and best of our knowledge, similar work has not been submitted to **“ANNA UNIVERSITY CHENNAI”** for the requiremen2t of Degree of **BACHELOR OF ENGINEERING**. This project report is submitted on the partial fulfillment of the requirement of the award of the course **CGA1121- DATA STRUCTURES.**

Signature

[illegible]

____

NAVEENA R

Place: Samayapuram

Date: 15/06/2024

ACKNOWLEDGEMENT

It is with great pride that I express our gratitude and indebtedness to our institution, "**K. Ramakrishnan College of Technology (Autonomous)**", for providing us with the opportunity to do this project.

I extend our sincere acknowledgment and appreciation to the esteemed and honorable Chairman, **Dr. K. RAMAKRISHNAN**, **B.E.,** for having provided the facilities during the course of our study in college.

I would like to express our sincere thanks to our beloved Executive Director, **Dr. S. KUPPUSAMY, MBA, Ph.D.,** for forwarding our project and offering an adequate duration to complete it.

I would like to thank **Dr. N. VASUDEVAN, M.TECH., Ph.D.,**

Principal, who gave the opportunity to frame the project to full satisfaction.

I thank **Dr. A. DELPHIN CAROLINA RANI M.E., Ph.D.,** Head of the Department of COMPUTER SCIENCE AND ENGINEERING, for providing her encouragement in pursuing this project.

I wish to convey our profound and heartfelt gratitude to our esteemed project guide **Mr. D.P.DEVAN B.E.,M.E.,** Department of **COMPUTER SCIENCE AND ENGINEERING,** for her incalculable suggestions, creativity, assistance and patience, which motivated us to carry out this project.

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

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

VISION OF THE INSTITUTION

To emerge as a leader among the top institutions in the field of technical education.

MISSION OF THE INSTITUTION

- ➢ Produce smart technocrats with empirical knowledge who can surmount the global challenges.
- ➢ Create a diverse, fully-engaged, learner-centric campus environment to provide quality education to the students.
- ➢ Maintain mutually beneficial partnerships with our alumni, industry, and Professional associations.

VISION OF DEPARTMENT

To be a center of eminence in creating competent software professionals with research and innovative skills.

MISSION OF DEPARTMENT

M1: Industry Specific: To nurture students in working with various hardware and software platforms inclined with the best practices of industry.

M2: Research: To prepare students for research-oriented activities.

M3: Society: To empower students with the required skills to solve complex technological problems ofsociety.

PROGRAM EDUCATIONAL OBJECTIVES

1. PEO1: Domain Knowledge

To produce graduates who have strong foundation of knowledge and skills in the field of Computer Science and Engineering.

2. PEO2: Employability Skills and Research

To produce graduates who are employable in industries/public sector/research organizations or work as an entrepreneur.

3. PEO3: Ethics and Values

To develop leadership skills and ethically collaborate with society to tackle real-world challenges.

PROGRAM SPECIFIC OUTCOMES (PSOs)

PSO 1: Domain Knowledge

To analyze, design and develop computing solutions by applying foundational concepts of Computer Science and Engineering.

PSO 2: Quality Software

To apply software engineering principles and practices for developing quality software for scientific and business applications.

PSO 3: Innovation Ideas

To adapt to emerging Information and Communication Technologies (ICT) to innovate ideas and solutions to existing/novel problems

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

The Bus Reservation System is a comprehensive software solution developed in C language, designed to automate the process of booking bus tickets. This system aims to improve the efficiency, accuracy, and convenience of managing bus reservations. It offers a user-friendly interface for customers to seamlessly search for available buses, select seats, and book tickets. The system reduces the dependency on manual processes, thereby minimizing human errors and enhancing the overall operational workflow.Key functionalities of the system include secure user registration and authentication, real-time seat availability checks, and detailed booking management. Users can view available buses on specific routes, make reservations, and cancel bookings with ease. For administrators, the system provides robust tools to manage bus schedules, routes, and monitor booking activities. This ensures that all operations are streamlined and data is consistently up-to-date. The Bus Reservation System leverages efficient data structures and file handling mechanisms to maintain an accurate database of buses, routes, users, and bookings. It is designed to handle a high volume of transactions, ensuring reliability and performance. The system also prioritizes security, protecting user data and maintaining the integrity of booking records.By automating the reservation process, the Bus Reservation System offers significant advantages including reduced operational costs, improved customer satisfaction, and enhanced data accuracy. This project not only simplifies the reservation process but also provides valuable insights through detailed reporting and analytics for better decision-making.

TABLE OF CONTENTS

CHAPTER	TITLE	PAGENO.
	ABSTRACT	viii
	LIST OF FIGURES	x
	LIST OF ABBREVIATIONS	xi
1	**INTRODUCTION**	
	1.1 Introduction	1
	1.2 Purpose And Importance	1
	1.3 Objectives	2
	1.4 Project Summarization	2
2	**PROJECT METHODOLOGY**	
	2.1. Introduction to System Architecture	3
	2.2 Detailed System Architecture Diagram	4
3	**DATA STRUCTURE PREFERANCE**	
	3.1 Explanation of why a Queue was chosen	5
	3.2 Comparison with Other Data Structures	6
	3.3 Advantages and disadvantages of using a Queue	6
4	**DATA STRUCTURE METHODOLOGY**	
	4.1. Arrays	7
	4.2. Linked List	7
	4.3. Stacks	8
5	**MODULES**	
	5.1 login Management	9
	5.2 Bus Schedule Management	9
	5.3 Booking Management	9
	5.5 Generate Reports	9
6	**CONCLUSION & FUTURE SCOPE**	
	6.1 Conclusion	10
	6.2 Future Scope	10
	APPENDICES	
	Appendix A-Source code	12
	Appendix B -Screen shots	17

LIST OF FIGURES

FIGURE NO	TITLE	PAGENO.
2.1	Architecture Diagram	4

LIST OF ABBREVIATIONS

ABBREVIATIONS

1. BRS - Bus Reservation System
2. CRS - Central Reservation System
3. PNR - Passenger Name Record
4. ET - Electronic Ticket
5. ETA - Estimated Time of Arrival
6. ETD - Estimated Time of Departure
7. ID - Identification
8. OTP - One-Time Password
10. UI - User Interface
11. UX - User Experience

CHAPTER 1

INTRODUCTION

1.1 INTRODUCTION TO PROJECT

The Bus Reservation System project is like an online platform for booking bus tickets. It helps users find buses, choose seats, book tickets, and get their tickets easily. The system manages bus details, routes, available seats, and bookings. Users can search for buses by date, time, and destination, book tickets, cancel bookings, and view their booking information. It's a convenient way to handle bus ticket reservations for both users and administrators.

1.2 PURPOSE AND IMPORTANCE OF THE PROJECT

The purpose and importance of the Bus Reservation System project in C is to streamline the process of booking bus tickets for users and administrators. It simplifies the ticket booking process by providing an online platform where users can easily check bus availability, select seats, and book tickets without the need to visit a physical ticket counter.

This project is essential because it saves time and effort for travelers by allowing them to make reservations from the comfort of their homes or on the go. It also benefits bus companies and administrators by automating the ticketing process, managing seat availability, and keeping track of bookings efficiently. Overall, the project enhances the user experience, increases operational efficiency, and improves the overall management of bus reservations. This project is not only important for simplifying the booking process but also for improving the overall efficiency and effectiveness of bus services. It helps in reducing manual work for bus companies, streamlining operations, and ensuring better customer service. Additionally, it can lead to increased revenue for bus companies by attracting more customers through the convenience of online booking. Overall, the project has a positive impact on the travel industry by enhancing user experience, optimizing resources, and driving business growth.

1.3 OBJECTIVES

1. Streamline Ticket Booking Process
2. Enhance User Experience
3. Improve Operational Efficiency
4. Provide Real-time Updates
5. Secure Data Management

1.4 PROJECT SUMMARIZATION

The Bus Reservation System is a C-based application designed to automate the process of booking bus tickets. It provides a user-friendly interface for customers to check real-time seat availability, book, and cancel tickets. Administrators can manage bus schedules, routes, and bookings efficiently. The system employs various data structures: queues for handling booking requests, single linked lists for managing user data, and doubly linked lists for managing bus routes. This ensures dynamic data handling, scalability, and secure data management, ultimately enhancing operational efficiency and user satisfaction and some of the benefits are

1. **Increased Efficiency:**

   Automates manual booking processes, significantly reducing time and effort.

2. **Enhanced User Experience:**

   Provides a simple and intuitive interface for booking and managing reservations.

3. **Improved Operational Accuracy:**

   Minimizes human errors through automated data handling and management.

4. **Real-time Seat Availability:**

   Ensures up-to-date information on seat availability, preventing overbooking and related issues.

CHAPTER 2

PROJECT METHODOLOGY

2.1 INTRODUCTION TO SYSTEM ARCHITECTURE

The system architecture for the bus reservation system is basically the blueprint that shows how all the parts of the booking system fit together. It includes software, hardware, and databases to ensure smooth online booking for travelers and bus companies. It's like the foundation that keeps everything running smoothly.

2.11 High-Level System Architecture

The high-level system architecture for the bus reservation system typically consists of several key components:

(i) User Interface (UI)

(ii) Data Structure

(iii) Functional Module

2.1.2 Components of the System Architecture

a. Functional Module

These modules contain the core functionality of the bus reservation system. They handle tasks such as managing buses, routes, reservations, payments, and other related operations. Each module typically focuses on a specific aspect of the system's functionality, making the system modular and easier to maintain.

b. Data Structure

This component defines the data models and structures used throughout the system. It includes entities such as buses, routes, reservations, users, and any other relevant information. Data structures may be implemented using classes, databases, or other data storage mechanisms, depending on the system's requirements.

c.User Interface

Provides a way for users to interact with the system by displaying menus, prompts, and messages, and reading user input.

2.2 DETAILED SYSTEM ARCHITECTURE DIAGRAM

The system architecture of the Bus Reservation System consists of four main layers. The User Interface (UI) Layer provides a user-friendly interface for both customers and administrators, facilitating interactions with the system. The functionalities of Functional Module such as booking, cancellation, and seat availability checks, enforcing business rules and handling user requests. The Data Management Layer handles data storage and retrieval, employing queues for request management, single linked lists for user data, and doubly linked lists for bus routes. Lastly, the Security Layer ensures secure login and data management, protecting user information and maintaining data integrity. This layered approach ensures a robust, efficient, and secure system for managing bus reservations.

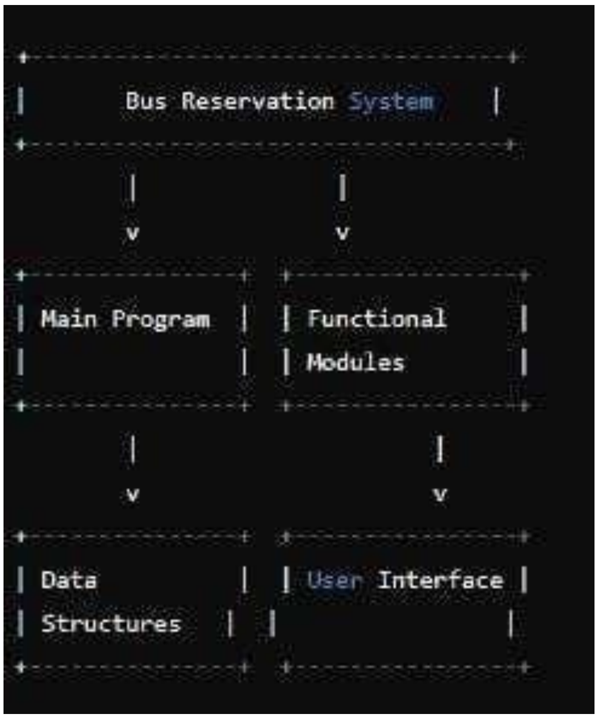


Fig 2.1 : Architecture Diagram (Sample)

CHAPTER 3

DATA STRUCTURE PREFERANCE

3.1 EXPLANATION OF WHY A QUEUE WAS CHOSEN

The In the Bus Reservation System, a queue is chosen for request management due to several reasons specific to the nature of the system:

1. **First-Come**, **First-Serve** (**FCFS**): **Basis**: Queues naturally follow the FCFS principle, where the first request to arrive is the first one to be served. In a bus reservation system, this ensures fairness in processing requests for booking, cancellation, or seat availability checks. Customers expect their requests to be handled in the order they submit them, and a queue facilitates this without any bias.

2. **Handling Concurrent Requests**: A busreservation system typically experiences a high volume of requests, especially during peak travel times. Queues provide an organized way to handle concurrent requests without risking data integrity issues. Each request enters the queue and is processed sequentially, preventing conflicts and ensuring that resources are utilized efficiently.

3. **Scalability**: As the system grows and more users interact with it, the queue can scale accordingly to accommodate increasing demand. Whether there are ten or ten thousand requests in the queue, the system can manage them efficiently, maintaining performance and responsiveness.

4. **Fault Tolerance**: In case of system failures or temporary downtimes, a queue acts as a buffer, storing pending requests until the system is operational again. This ensures that no requests are lost and that users' actions are not disrupted due to unforeseen technical issues.

5. **Load Balancing**: Queues can help in distributing the load evenly across different components of the system. By managing the rate at which requests are dequeued from the queue, the system can adjust to fluctuations in demand, preventing overload on specific resources and ensuring a smooth user experience.

3.2 COMPARISON WITH OTHER DATA STRUCTURES

In comparison to other data structures, queues operate on the principle of First In, First Out (FIFO), distinguishing them from stacks, which adhere to Last In, First Out (LIFO) logic. While stacks are useful for applications like function calls and backtracking, queues excel in scenarios requiring orderly processing, such as task scheduling and breadth-first traversal of trees. Linked lists serve as a foundation for queue implementations, offering dynamic memory allocation for efficient insertion and deletion operations. Moreover, queues find utility in conjunction with other structures, such as trees, facilitating breadth-first search algorithms. This succinctly underscores how queues, with their FIFO behavior, provide a structured approach to managing data that needs to be processed in a sequential manner.

3.3 ADVANTAGES AND DISADVANTAGES OF USING QUEUE

3.3.1 Advantages of Using a Queue:

One advantage of queues is their ability to ensure orderly processing based on the First In, First Out (FIFO) principle. This property makes queues particularly useful in scenarios where tasks or requests need to be handled in a systematic and fair manner, enhancing efficiency and maintaining data integrity. One more advantage of queues is their ability to decouple producers and consumers, allowing them to operate independently at their own pace. This asynchronous communication enhances system robustness and flexibility, enabling efficient handling of varying workloads without the risk of bottlenecks. Overall, integrating a queue mechanism into the bus reservation system enhances reliability, fairness, and efficiency, ultimately enhancing the overall user experience.

Disadvantages of Using a Queue:

One disadvantage of queues is the potential for increased waiting times, especially during peak periods, which can lead to delays and customer dissatisfaction. Additionally, queues may lack flexibility in prioritizing urgent tasks or requests, potentially resulting in inefficiencies during high-demand situations. One disadvantage of queues is their vulnerability to congestion, particularly when the queue becomes too long or when there are system failures. This congestion can lead to delays in processing tasks or requests, impacting overall system user experience.

CHAPTER -4
DATA STRUCTURE METHODOLOGY

4.1 ARRAY

Arrays are integral components of a bus reservation system, providing a structured approach to managing crucial data such as seat availability, passenger information, and bus schedules. By representing seats as elements in an array, the system can efficiently track which seats are booked and available, facilitating smooth booking processes. Additionally, arrays store passenger details, including names, contact information, and booking statuses, ensuring accurate and up-to-date records for efficient reservation management. Moreover, arrays organize bus schedules, allowing for easy access to information on departure times, destinations, and available seats across different routes. Overall, the utilization of arrays optimizes the system's performance, enhancing user experience and operational efficiency in bus reservation processes.

4.1.1 Key features of a Array:

1. Storage of Multiple Elements: Arrays can store multiple elements of the same data type in contiguous memory locations.

2. Indexed Access: Elements in arrays can be accessed using indices, allowing for quick retrieval and modification based on their position within the array.

3. Fixed Size: Arrays have a fixed size determined at the time of declaration, which helps manage memory allocation efficiently.

4. Versatility: Arrays support various operations such as iteration, searching, sorting, and merging, making them versatile data structures for a wide range of applications.

5. Multi-dimensional Capability: Arrays can be multi-dimensional, enabling the representation of complex data structures such as matrices and tables.

6. Efficient Memory Usage: Arrays use contiguous memory allocation, which ensures efficient memory usage and fast access times for elements.

4.2 NODE STRUCTURE

- **Main File (app.js or index.js)**: This is the entry point of your application where you initialize your Node.js server and set up your routes.
- **Controllers**: Implement the logic for each route in separate controller files. This keeps your route handlers clean and modular. For example, you might have controllers for user authentication, bus booking, etc.
- **Models**: Define data models to represent your application's data. For a bus reservation system, you'll likely have models for users, buses, bookings, etc. You can use an ORM (Object-Relational Mapping) library like Sequelize or Mongoose to interact with your database.
- **Middleware**: Middleware functions can be used for tasks like authentication, validation, error handling, etc. They can be applied globally or to specific routes as needed.
- **Services (Optional)**: If your application has complex business logic, you can encapsulate it in service modules. For example, a booking service might handle tasks like checking seat availability, calculating fares, etc.
- **Database Configuration**: Set up a connection to your database. Depending on your choice of database (SQL or NoSQL), this configuration might differ.
- **Configuration Files**: Store configuration settings such as database credentials, API keys, etc., in separate files to keep them separate from your code.

4.3 INITIALIZATION, INSERTION & DELETION

In array data structure, initialization involves creating an array with a fixed size and storing elements sequentially. Insertion typically involves adding an element at a specified position within the array, which may require shifting existing elements to accommodate the new one. Deletion entails removing an element from the array, which might also involve shifting elements to fill the gap left by the deleted element.

CHAPTER-5
MODULES

- **LOGIN MANAGEMENT**

  Function Name: addlogin**()`**

  Description: Login management is the process of controlling access to digital systems. It involves verifying users' identities, determining their permissions, managing active sessions securely, and ensuring passwords are handled safely. This safeguards the system from unauthorized access and protects user data.

- **BUS SCHEDULE MANAGEMENT**

  Function Name: addschedule()

  Description: "Bus Schedule Management" encompasses planning,coordination, and fine-tuning of bus timetables to optimize routes, minimize waiting times, and enhance overall passenger experience. It involves factors like analyzing demand, adjusting routes, and ensuring timely updates to meet evolving needs while maintaining operational efficiency.

- **BOOKING MANAGEMENT**

  Function name: addbooking()

  Description: Booking management involves the organization and coordination of reservations, appointments, or arrangements for various services, events, or resources. It typically includes tasks such as scheduling, confirming bookings, handling cancellations or modifications, and ensuring resources are available to fulfill the bookings. Efficient booking management systems often utilize technology to streamline processes, manage availability, and provide convenient options for customers to make, change, or cancel bookings. The goal is to optimize resource utilization, enhance customer satisfaction.

- **GENERATE REPORT:**

Function name:addreport()

Description: The generateReport() function in the Bus Reservation System compiles and presents detailed reports based on specified criteria, such as report type, date range, and filters. It collects data, filters and aggregates it, and formats it into readable tables or charts. Reports can include metrics like total bookings, cancellations, revenue, popular routes, user activity, and bus utilization rates. This function aids administrators in monitoring performance, identifying trends, making data-driven decisions, and enhancing operational transparency

CHAPTER 6
CONCLUSION & FUTURE SCOPE

6.1 CONCLUSION

The Bus Reservation System is a comprehensive solution designed to streamline the process of booking bus tickets. By leveraging a layered architecture, it efficiently manages user interactions, booking processes, data storage, and security. The system's user-friendly interface simplifies tasks for both passengers and administrators, while the application logic layer ensures accurate and efficient booking and cancellation operations. The use of robust data structures enhances data management and retrieval, supporting dynamic and scalable operations. Overall, the system improves operational efficiency, enhances user satisfaction, and ensures secure and reliable handling of all booking-related activities, making it a vital tool for modern transportation management.

6.2 FUTURE SCOPE

The The future scope of the Bus Reservation System includes developing a mobile app for greater convenience, integrating real-time GPS tracking for better bus location monitoring, and implementing advanced analytics for improved decision-making. Adding multi-language support and multiple payment gateways will enhance accessibility and transaction security. Introducing AI-powered chatbots will improve customer service, while dynamic pricing strategies can optimize revenue. A feedback and rating system will help improve service quality, and enhanced security features will ensure better protection of user data and transactions. Additionally, the system can incorporate features like automated schedule adjustments based on real-time traffic conditions, support for inter-modal transportation planning, partnerships with other travel services for comprehensive travel packages, and eco-friendly options like carbon footprint tracking and recommendations for sustainable travel. These enhancements will make the system more versatile, user-friendly, and aligned with trends.

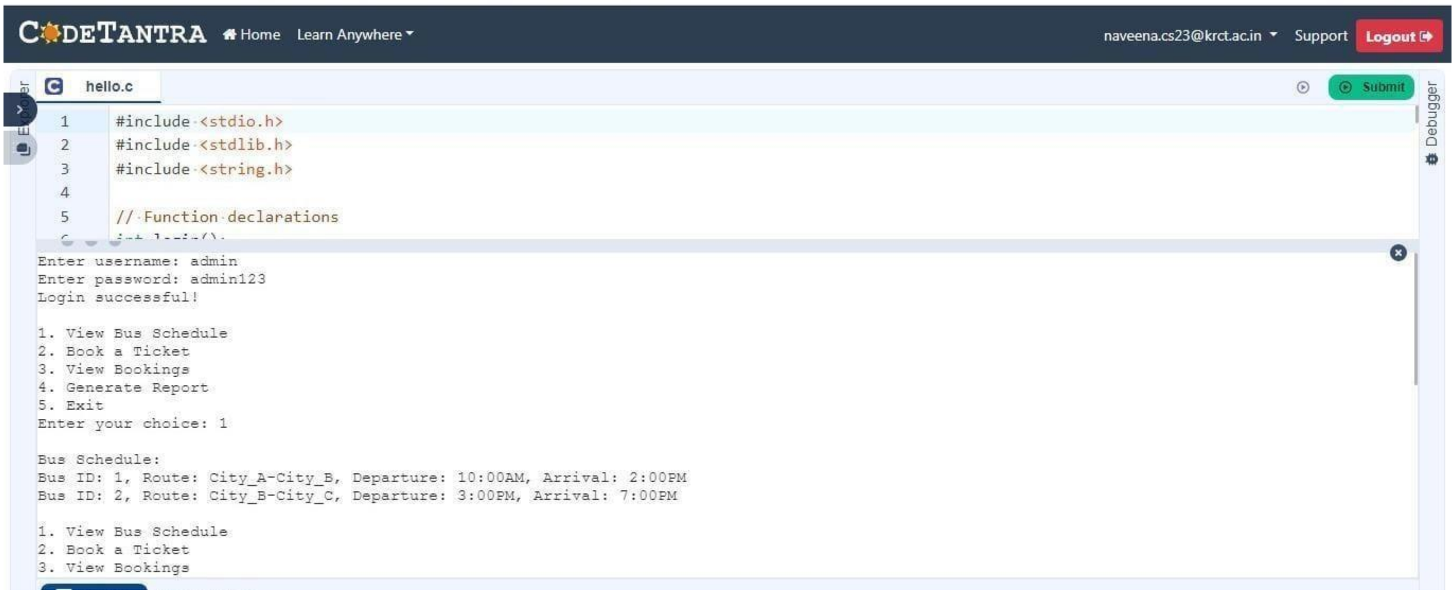
APPENDICES

APPENDIX A-SOURCE CODE

```
#include   <stdio.h>
#include  <stdlib.h>
#include <string.h>

// Function declarations
int login();
void view_schedule();
void book_ticket();
void view_bookings();
void generate_report();

// User credentials
typedef struct {
    char username[50];
    char password[50];
} User;

// Bus schedule
typedef struct {
   char id[10];
    char route[100];
    char departure[20];
    char arrival[20];
} Bus;
```

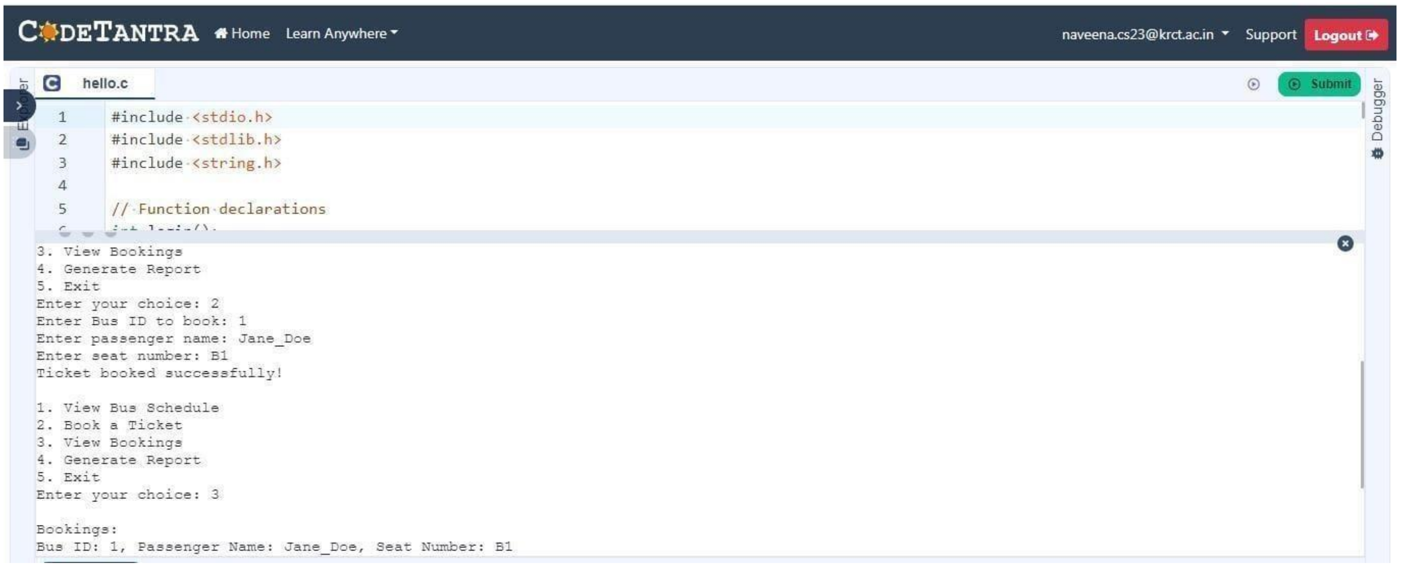
```
// Booking information
typedef struct {
    char bus_id[10];
    char passenger_name[50];
    char seat_number[10];
} Booking;

// Sample data
User users[] = {
    {"admin", "admin123"},
    {"user1", "password1"}
};

Bus buses[] = {
    {"1", "City_A-City_B", "10:00AM", "2:00PM"},
    {"2", "City_B-City_C", "3:00PM", "7:00PM"}
};

Booking bookings[100];
int booking_count = 0;

int main() {
    if (login()) {
        while (1) {
            printf("\n1. View Bus Schedule\n");
            printf("2. Book a Ticket\n");
            printf("3. View Bookings\n");
            printf("4. Generate Report\n");
            printf("5. Exit\n");
            int choice;
            printf("Enter your choice: ");
            scanf("%d", &choice);

            switch (choice) {
                case 1:
                    view_schedule();
                    break;
                case 2:
                    book_ticket();
                    break;
                case 3:
                    view_bookings();
                    break;
                case 4:
                    generate_report();
                    break;
                case 5:
                    exit(0);
                default:
                    printf("Invalid choice. Please try again.\n");
            }
        }
    }
    return 0;
}

// Login management
int login() {
    char username[50], password[50];
    printf("Enter username: ");
    scanf("%s", username);
    printf("Enter password: ");
    scanf("%s", password);

    for (int i = 0; i < sizeof(users) / sizeof(users[0]); i++) {
        if (strcmp(username, users[i].username) == 0 && strcmp(password, users[i].password)
== 0) {
            printf("Login successful!\n");
            return 1;
        }
    }

    printf("Invalid credentials. Please try again.\n");
    return 0;
}

// Bus schedule management
void view_schedule() {
    printf("\nBus Schedule:\n");
    for (int i = 0; i < sizeof(buses) / sizeof(buses[0]); i++) {
        printf("Bus ID: %s, Route: %s, Departure: %s, Arrival: %s\n",
               buses[i].id, buses[i].route, buses[i].departure, buses[i].arrival);
    }
}

// Booking management
void book_ticket() {
    if (booking_count >= 100) {
        printf("Booking limit reached.\n");
        return;
    }

    printf("Enter Bus ID to book: ");
    scanf("%s", bookings[booking_count].bus_id);
    printf("Enter passenger name: ");
    scanf("%s", bookings[booking_count].passenger_name);
    printf("Enter seat number: ");
    scanf("%s", bookings[booking_count].seat_number);

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

void view_bookings() {
    printf("\nBookings:\n");
    for (int i = 0; i < booking_count; i++) {
        printf("Bus ID: %s, Passenger Name: %s, Seat Number: %s\n",
               bookings[i].bus_id, bookings[i].passenger_name, bookings[i].seat_number);
    }
}

// Report generation
void generate_report() {
    printf("\nBooking Report:\n");
    for (int i = 0; i < booking_count; i++) {
        printf("Bus ID: %s, Passenger Name: %s, Seat Number: %s\n",
               bookings[i].bus_id, bookings[i].passenger_name, bookings[i].seat_number);
    }
}
```

APPENDIX B- SCREENSHOTS
OUTPUT



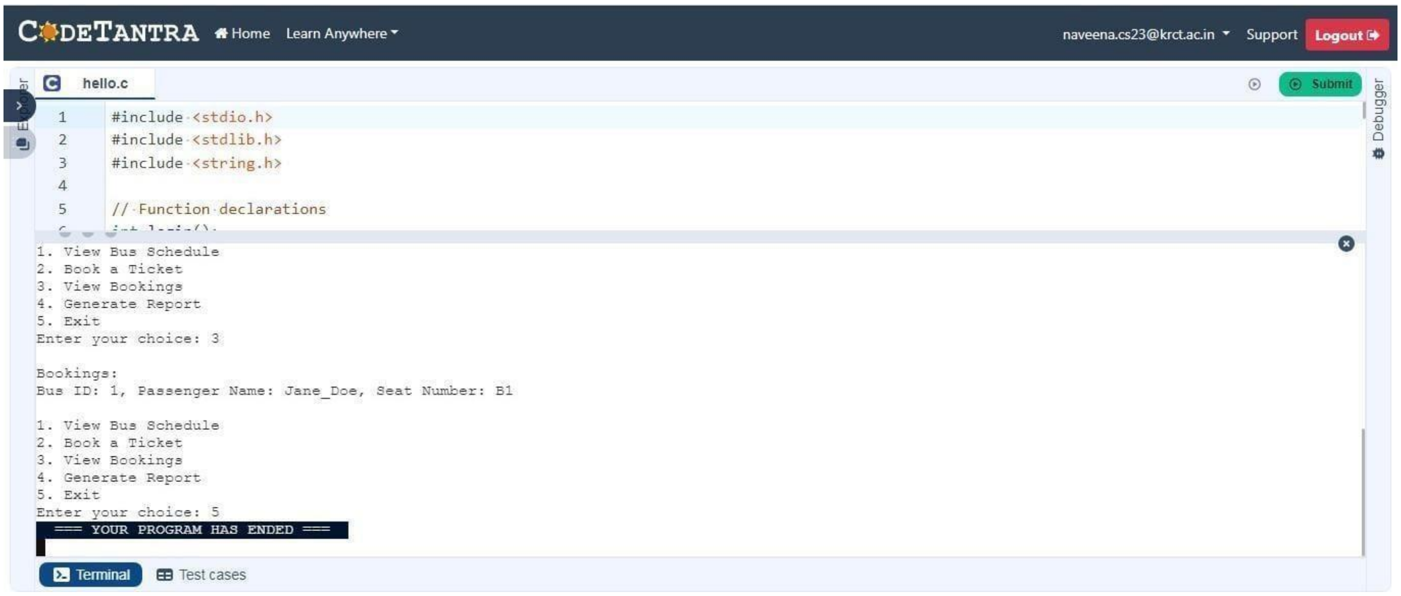
```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

// Function declarations
```

```
Enter username: admin
Enter password: admin123
Login successful!

1. View Bus Schedule
2. Book a Ticket
3. View Bookings
4. Generate Report
5. Exit
Enter your choice: 1

Bus Schedule:
Bus ID: 1, Route: City_A-City_B, Departure: 10:00AM, Arrival: 2:00PM
Bus ID: 2, Route: City_B-City_C, Departure: 3:00PM, Arrival: 7:00PM

1. View Bus Schedule
2. Book a Ticket
3. View Bookings
```



```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

// Function declarations
```

```
3. View Bookings
4. Generate Report
5. Exit
Enter your choice: 2
Enter Bus ID to book: 1
Enter passenger name: Jane_Doe
Enter seat number: B1
Ticket booked successfully!

1. View Bus Schedule
2. Book a Ticket
3. View Bookings
4. Generate Report
5. Exit
Enter your choice: 3

Bookings:
Bus ID: 1, Passenger Name: Jane_Doe, Seat Number: B1
```



```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

// Function declarations
```

```
1. View Bus Schedule
2. Book a Ticket
3. View Bookings
4. Generate Report
5. Exit
Enter your choice: 3

Bookings:
Bus ID: 1, Passenger Name: Jane_Doe, Seat Number: B1

1. View Bus Schedule
2. Book a Ticket
3. View Bookings
4. Generate Report
5. Exit
Enter your choice: 5
 === YOUR PROGRAM HAS ENDED ===
```