DLITE PROJECT REPORT

PROJECT ID:CP001

PROJECT TITTLE: Bus reservation system

TEAM MEMBERS: REHAN-4MT21CS091

NANDHAKISHORE-4MT21CS092

NAVINYA SHETTY-4MT21CS093

NIHAAL AHEMMED-4MT21CS094

NIHAL SHETTY-4MT21CS095

ABSTRACT

The Bus Reservation System is a comprehensive and user-friendly software application designed to streamline and automate the bus ticket booking process. This system aims to enhance the efficiency of bus transportation services while providing convenience and flexibility to both passengers and bus operators.

The primary objective of the Bus Reservation System is to simplify the reservation process, reducing the time and effort required for passengers to book their bus tickets. It offers an online platform accessible through web and mobile applications, allowing passengers to browse available bus routes, view seat availability, and make reservations from the comfort of their homes or on the go. The system also caters to the needs of walk-in passengers by providing booking facilities at bus terminals.

INTRODUCTION

Introduction to Bus Reservation System:

The Bus Reservation System has emerged as a transformative solution to address these longstanding issues. Leveraging cutting-edge technology and user-friendly interfaces, this system empowers passengers to effortlessly plan their bus journeys, access real-time information, and secure their seats with a few clicks or taps. Gone are the days of waiting in endless lines or facing uncertainty when planning a trip.

Key components of the Bus Reservation System include a robust online platform accessible through web and mobile applications, secure payment processing, real-time seat availability updates, and integration with global positioning systems (GPS) for real-time bus tracking. Additionally, passengers can create user profiles to streamline future bookings and access a range of customer support options to address their queries and concerns.

For bus operators, the system offers a dedicated dashboard that simplifies route and schedule management, seat allocation, and provides valuable insights through reporting and analytics tools. This data-driven approach allows operators to optimize their services, improve efficiency, and enhance the overall passenger experience.

In essence, the Bus Reservation System is more than just a technological advancement; it is a catalyst for change in the bus transportation industry. It aligns with the evolving needs and expectations of modern travelers who seek convenience, reliability, and transparency in their journeys. Furthermore, it contributes to the broader digital transformation of the transportation sector, ushering in an era where bus travel is not only efficient but also an enjoyable experience for all passengers.

Background:

The Bus Reservation System has evolved as a response to the evolving demands and challenges within the transportation industry. Historically, bus ticket booking relied on manual processes and paper-based systems, which were cumbersome, time-consuming, and often resulted in inefficiencies for both passengers and bus operators. Here is a brief background outlining the key factors that led to the development of the Bus Reservation System:

1. Manual Booking Challenges: Traditional bus ticket booking involved passengers physically visiting ticket counters or agents, leading to long queues, especially during peak travel seasons. This process was prone to errors and often resulted in overbooking or underbooking of seats.

2. Limited Accessibility: Passengers faced limitations in accessing information about bus routes, schedules, and seat availability. This lack of transparency made it difficult for them to plan their journeys effectively.

3. Operational Complexities: Bus operators grappled with administrative complexities, including managing reservations, tracking payments, and optimizing bus schedules and routes. Without real-time data and tools, they struggled to streamline their operations.

4. Technological Advancements: The advancement of technology, including web development, mobile applications, secure payment gateways, and GPS tracking systems, created opportunities to revolutionize the bus reservation process.

6. Customer-Centric Approach: The focus shifted towards providing a more customer-centric approach to bus travel, with an emphasis on user-friendly interfaces, real-time updates, and responsive customer support.

Objectives:

The main objectives of a Bus Reservation System are to streamline the bus ticket booking process, enhance the overall travel experience, and improve operational efficiency for both passengers and bus operators. Here are six key objectives:

1. Efficient Booking Process: Simplify and expedite the ticket booking process for passengers by providing an easy-to-use online platform. This objective aims to reduce the time and effort required to make reservations, eliminating the need for long queues at ticket counters.

2. Real-time Availability: Ensure real-time updates on seat availability, bus schedules, and routes. Passengers should have access to accurate information, allowing them to make informed decisions and choose the most convenient options for their journeys.

3. Secure Payment Processing: Implement secure payment gateways to facilitate online transactions, enabling passengers to pay for their tickets using various payment methods, including credit/debit cards, digital wallets, and other electronic payment options.

4. Improved Customer Experience: Enhance the overall travel experience for passengers by providing user-friendly interfaces, the option to select preferred seats, and the ability to create user profiles for faster booking in the future. Additionally, offer responsive customer support to address inquiries and issues promptly.

5. Operational Efficiency: Assist bus operators in efficiently managing their services. This includes tools for route and schedule management, automated seat allocation, and access to reporting and analytics to optimize bus operations. The system should help reduce manual administrative tasks and improve resource allocation.

6. Data-driven Insights: Collect and analyze data on passenger preferences, booking patterns, and operational performance. By leveraging data analytics, bus operators can make informed decisions, adjust schedules, and tailor their services to meet passenger demand effectively.

TECHNOLOGIES USED

A Bus Reservation System relies on various technologies to provide efficient and user-friendly services to passengers and bus operators. Here are five key technologies commonly used in a Bus Reservation System:

1. Web Development Technologies: HTML, CSS, and JavaScript: These front-end web technologies are used to create the user interface of the reservation system, ensuring that passengers can access and interact with the system through web browsers.

2. Mobile Application Development: Mobile App Development Frameworks: Mobile apps for iOS and Android platforms are essential to provide on-the-go access to bus reservations. Technologies like React Native or Flutter allow for cross-platform app development, reducing development time and costs. APIs (Application Programming Interfaces): APIs facilitate communication between mobile apps and the server, enabling features like real-time seat availability checks and secure payment processing.

3. Backend Development: Programming Languages: Backend systems are typically developed using languages like Python, Ruby, Java, or Node.js to handle business logic, process bookings, manage databases, and communicate with external services. Database Management Systems: Databases (e.g., MySQL, PostgreSQL, MongoDB) are crucial for storing passenger information, reservations, schedules, and operational data securely.

4. Payment Gateways: Payment Processing APIs: Integration with payment gateways such as PayPal, Stripe, or local payment providers is essential to securely process online payments for bus reservations. This ensures passenger payment data is protected and transactions are efficient.

5. GPS and Geolocation Services: Global Positioning System (GPS): GPS technology is used to track the real-time location of buses. This information is crucial for providing passengers with live updates on bus locations and estimated arrival times. Geocoding and Mapping APIs: Services like Google Maps or Mapbox are used for geocoding bus stops, creating route maps, and displaying the location of buses and stops in the passenger interface.

SYSTEM ARCHITECTURE

Bus Reservation System Architecture typically involves a multi-tiered approach that separates various components and responsibilities within the system. Here are three common architectural layers in a Bus Reservation System:

1. Presentation Layer: User Interface (UI): This layer is the front-end of the system, responsible for presenting the system to users, including passengers and bus operators. It includes web and mobile interfaces that allow users to search for bus routes, check seat availability, make reservations, and perform other interactions. User Experience (UX): Design considerations, usability, and responsiveness are vital in creating an intuitive and user-friendly experience for passengers. The UI should also support various devices and screen sizes. Authentication and Authorization: Users need to log in or create accounts to make reservations or manage their profiles. Authentication ensures secure access to user data and functionalities, while authorization controls user permissions.

2. Application Layer: Business Logic: This layer contains the core business logic of the Bus Reservation System. It handles functions like booking management, seat allocation, payment processing, route scheduling, and bus tracking. APIs (Application Programming Interfaces): APIs enable communication between the presentation layer (UI) and the application layer (business logic). These APIs facilitate data exchange, real-time updates, and transaction processing. For example, APIs may provide access to seat availability data or payment processing services.

3. Data Layer: Database: This layer consists of one or more databases that store structured data, such as passenger information, reservations, bus schedules, and route details. It is crucial for data consistency, security, and reliability.Data Processing: Data processing components may be used to aggregate and analyze operational data, such as passenger preferences, booking trends, and bus performance metrics. This information can be valuable for business intelligence and optimization.

PROJECT MODULES

1. User Registration and Authentication Module - This module allows passengers and bus operators to create accounts and log in securely

- Features:
  - User registration and profile management.
  - Secure login and authentication mechanisms.
  - Password reset and recovery options.
  - User role management (passenger, bus operator, admin).

2. Bus Route and Schedule Management Module:

- This module deals with the management of bus routes, schedules, and seat availability.
- Features:
  - Adding, updating, and deleting bus routes.
  - Setting schedules for each route, including departure times and stops.
  - Managing seat availability and pricing.
  - Real-time updates on route and schedule changes.

3. Booking and Reservation Module:

- The heart of the system, this module handles passenger bookings and reservations.
- Features:
  - Searching for available buses and routes.
  - Selecting preferred seats.
  - Booking and reserving seats for specific journeys.

    - Payment processing and ticket generation.

    - Handling cancellations and refunds.

4. Bus Tracking and Real-time Updates Module:

   - This module provides passengers with real-time information about bus locations and estimated arrival times.

   - Features:

    - Integration with GPS or tracking systems on buses.

    - Live tracking of buses on maps.

    - Alerts and notifications for passengers about delays or changes.

5. Admin and Reporting Module:

   - This module is for system administrators and bus operators to manage the system and access analytics.

   - Features:

    - Admin dashboard for system configuration and management.

    - Reporting and analytics tools to track system performance, passenger trends, and operational data.

    - Bus operator tools for route and schedule management.

    - User management and support ticket handling.

FEATURES AND IMPLEMENTATION

Designing and implementing a Bus Reservation System involves a range of technical components and considerations. Here, I'll outline the design and implementation of four key aspects of the system:

1. User Registration and Authentication:

   - Design:

    - Create a user-friendly registration interface where users can sign up by providing essential information like name, email, contact details, and password.

    - Implement secure password hashing and storage techniques to protect user data.

- Design a login page with strong authentication mechanisms, such as multi-factor authentication (MFA) if needed.

- Implementation:

  - Use web development technologies like HTML, CSS, and JavaScript for the user interface.

  - Utilize a backend framework (e.g., Node.js, Django, Ruby on Rails) for server-side logic.

  - Store user data in a relational database (e.g., MySQL) and implement authentication using libraries like Passport.js (Node.js) or Django's built-in authentication (Django).

  - Implement security best practices to prevent common vulnerabilities like SQL injection and cross-site scripting (XSS).

2. Bus Route and Schedule Management:

- Design:

  - Create a database schema for storing bus routes, schedules, and seat availability.

  - Design an admin panel for bus operators to manage routes, schedules, and pricing.

  - Create user interfaces for passengers to search for routes and view schedules.

- Implementation:

  - Develop a database system (e.g., PostgreSQL) for storing route and schedule data.

  - Implement an admin dashboard using web development frameworks (e.g., React, Angular, or Vue.js) for bus operators to manage routes and schedules.

  - Develop a search and schedule display feature on the user interface that fetches data from the database and presents it to passengers.

3. Booking and Reservation:

- Design:

  - Create a user interface for passengers to search for available buses, select seats, and make reservations.

  - Design a payment processing system to handle various payment methods securely.

  - Develop a booking confirmation and ticket generation feature.

- Implementation:

- Implement the booking and reservation system using server-side programming languages (e.g., Python, Node.js).

- Integrate payment gateways (e.g., Stripe, PayPal) for handling secure payments.

- Develop algorithms for seat selection and availability tracking.

- Generate e-tickets with unique booking references for passengers.

TESTING

Testing is a critical phase in the development of a Bus Reservation System to ensure that it functions correctly, securely, and efficiently. Here are two important types of testing for a Bus Reservation System:

1. Functional Testing:

- Objective: Functional testing ensures that each component and feature of the Bus Reservation System performs as expected and adheres to the specified requirements.

- Test Cases:

  - Booking Process Testing: Verify that users can search for routes, view schedules, select seats, make reservations, and complete payments without errors or inconsistencies.

  - Authentication Testing: Test user registration, login, password reset, and user role management to ensure that authentication and authorization functions correctly.

  - Route and Schedule Management: Verify that bus operators can add, update, and delete routes and schedules without encountering issues.

  - Seat Availability Testing: Confirm that the system accurately tracks seat availability in real-time and prevents overbooking.

  - Bus Tracking and Real-time Updates: Test the real-time tracking feature to ensure it displays accurate bus locations and provides timely notifications of delays or route changes.

- Tools: Test management tools like TestRail, JIRA, or Excel spreadsheets can be used to document test cases and track results. Automated testing tools like Selenium may also be useful for web interfaces.

2. Security Testing:

- Objective: Security testing assesses the system's vulnerability to potential threats and ensures that user data and transactions are protected.

- Test Cases:

  - Authentication and Authorization Checks: Test for common authentication vulnerabilities like brute force attacks and password policy enforcement. Verify that users cannot access unauthorized functionalities.

  - Input Validation: Test input fields for potential security issues such as SQL injection, cross-site scripting (XSS), and other injection attacks.

  - Payment Security: Evaluate the payment processing system for security flaws and ensure that sensitive payment data is securely handled and transmitted.

  - Data Encryption: Confirm that sensitive data, including user credentials and payment information, is transmitted and stored securely using encryption protocols (e.g., HTTPS, SSL/TLS).

  - Access Control: Check whether users can access or modify data that they shouldn't have permissions for.

- Tools: Security testing tools like OWASP ZAP, Nessus, or Burp Suite can help identify vulnerabilities and weaknesses in the system. Manual testing and code reviews are also essential for uncovering security issues.

CHALLENGES FACED

Implementing file handling for data storage and retrieval.

• Ensuring data consistency and accuracy.

• Handling errors and exceptions gracefully.

• Future Enhancements

• Implementing a graphical user interface (GUI) for a more user-friendly experience.

• Adding features for generating reports and statistics.

• Implementing security measures to protect data

FUTURE ENHANCEMENT

Enhancing an employee record system using file handling in programming can be done in several ways. Here are some future enhancements you can consider:

Data Encryption: Implement data encryption to protect sensitive employee information stored in files, ensuring data security and compliance with privacy regulations.

Search and Sorting: Add advanced search and sorting functionality to quickly locate and organize employee records based on various criteria like name, department, or hire date.

Data Validation: Implement robust data validation to ensure that only valid and consistent data is entered into the system, preventing errors and inconsistencies in records.

User Authentication: Enhance security by implementing user authentication mechanisms to control access to the employee record system, ensuring only authorized personnel can view or modify data.

Audit Trails: Keep a log of all changes made to employee records, including who made the changes and when, for tracking and auditing purposes.

Reporting and Analytics: Develop tools for generating reports and analytics on employee data, allowing for better decision-making and trend analysis.

Backup and Recovery: Implement automated backup and recovery processes to prevent data loss in case of system failures or errors.

Integration: Integrate the employee record system with other HR or payroll systems to streamline data sharing and reduce redundancy.

User-Friendly Interface: Continuously improve the user interface for ease of use and accessibility, incorporating user feedback and usability testing.

CONCLUSION

The Bus Reservation System stands as a testament to the transformative power of technology in the transportation industry. This system has revolutionized the way passengers plan their bus journeys and how bus operators manage their services. Through user-friendly interfaces, real-time updates, and efficient processes, the Bus Reservation System has enhanced the overall travel experience for all stakeholders involved. In essence, the Bus Reservation System has not only modernized bus transportation but has also made it a more attractive and efficient mode of travel. It aligns with the demands of contemporary travelers who seek convenience, reliability, and transparency in their journeys. As technology continues to evolve, it is likely that the Bus Reservation System will continue to adapt and enhance the travel experience further, shaping the future of bus transportation.

APPENDIX

Appendix A : Sample Data Files Sample data files for employee used for testing the program. employee.data: Contains sample medicine records.

Appendix B : User Manual A user manual providing instructions on how to use the employee Management System, including detailed explanations of each menu option and functionality.

Appendix C : Test Cases A list of test cases and their expected results used for unit testing and integration testing of the program.

Appendix D : Flowcharts Flowcharts illustrating the flow of control and data in the program for each module (employee record).

Appendix E : Error Handling A document detailing the error handling mechanisms implemented in the program, including how errors are detected and how they are handled.

Appendix F : Future Enhancements A document outlining potential future enhancements and features that can be added to the program to improve its functionality and usability.

Appendix G : Sample Reports Sample reports generated by the program, showcasing the potential reporting capabilities of the system.

Appendix H : Data Dictionary A data dictionary providing descriptions of the data structures used in the program, including the fields and their data types.

Appendix I : Glossary A glossary of terms and abbreviations used in the program, helping users understand the terminology used in the system.

Appendix J : References A list of references and sources of information used in the development of the program.

CODE SNIPPET

```
#include<stdio.h>

#include<conio.h>

#include<stdlib.h>

#include<string.h>

typedef struct{

        char name[50];

        int bus_num;

        int num_of_seats;

}pd;

void reservation(void);

void viewdetails(void);

void printticket(char name[],int,int,float);

void specificbus(int);

float charge(int,int);

void login();

int main()


{

                system("cls");

                system("COLOR 0A");

        printf("\t\t====================================================\n");

        printf("\t\t|                                        |\n");

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

        printf("\t\t|            BUS TICKET RESERVATION            |\n");

        printf("\t\t|                  SYSTEM                  |\n");

        printf("\t\t|      ------------------------------------     |\n");
        printf("\t\t|                                    |\n");
        printf("\t\t|                                    |\n");
        printf("\t\t|                                    |\n");
        printf("\t\t|                                    |\n");
        printf("\t\t|                                    |\n");
        printf("\t\t===================================================\n\n\n");



        printf(" \n Press any key to continue:");
        getch();
system("cls");
        login();
        int menu_choice,choice_return;
        start:
        system("cls");
        printf("\n===================================\n");
        printf("     BUS RESERVATION SYSTEM");
        printf("\n===================================");
        printf("\n1>> Reserve A Ticket");
        printf("\n------------------------");
        printf("\n2>> View All Available Bus");
        printf("\n------------------------");
        printf("\n3>> Exit");
        printf("\n-->");
        scanf("%d",&menu_choice);
        switch(menu_choice)
```

```
	{
		case 1:
			reservation();
			break;
		case 2:
			viewdetails();
			printf("\n\nPress any key to go to Main Menu..");
			getch();
			break;
		case 3:
			return(0);
		default:
			printf("\nInvalid choice");
	}
	goto start;
	return(0);
}


void viewdetails(void)
{
	system("cls");
	printf("-----------------------------------------------------------------------------------");
	printf("\nBus.No\tName\t\t\tDestinations  \t\tCharges  \t\tTime\n");
	printf("-----------------------------------------------------------------------------------");
	printf("\n1001\tGodavari Travels     \tDharan to Kavre       \tRs.500    \t\t9:00  AM");
```

```
        printf("\n1002\tDevit Travels       \tKavre To Dharan      \tRs.500  \t\t12:00 PM");
        printf("\n1003\tHero Travels        \tBenighat To Pokhara   \tRs.450   \t\t1:50  PM");
        printf("\n1004\tSuper Deluxe        \tPokhara To Benigha    \tRs.450   \t\t11:00 AM");
        printf("\n1005\tSai Baba Travels    \tMaitidevi To Janakpur \tRs.400  \t\t7:05  AM");
        printf("\n1006\tShine On Travels    \tJanakpur To Maitidevi \tRs.400   \t\t9:30  AM");
   printf("\n1007\tMayur Travels       \tJumla To Humla       \tRs.350   \t\t1:00  PM");
   printf("\n1008\tShree Travels       \tHumla To Jumla       \tRs.350   \t\t4:00  PM");
   printf("\n1009\tKathmandu Express   \tKathmandu To Pokhara  \tRs.600  \t\t5:35  PM");
   printf("\n1010\tSajha Yatayat       \tPokhara To Kathmandu  \tRs.600   \t\t4:15  PM");

}



void reservation(void)
{
        char confirm;
        int i=0;
        float charges;
        pd passdetails;
        FILE *fp;
        fp=fopen("seats_reserved.txt","a");
        system("cls");

        printf("\nEnter Your Name:> ");
        fflush(stdin);
        gets(passdetails.name);
```

```
printf("\nEnter Number of seats:> ");

scanf("%d",&passdetails.num_of_seats);

printf("\n\n>>Press Enter To View Available Bus<< ");

getch();

system("cls");

viewdetails();

printf("\n\nEnter bus number:> ");

start1:

scanf("%d",&passdetails.bus_num);

if(passdetails.bus_num>=1001 && passdetails.bus_num<=1010)

{

        charges=charge(passdetails.bus_num,passdetails.num_of_seats);

        printticket(passdetails.name,passdetails.num_of_seats,passdetails.bus_num,charges);

}

else

{

        printf("\nInvalid bus Number! Enter again--> ");

        goto start1;

}


printf("\n\nConfirm Ticket (y/n):>");

start:

scanf(" %c",&confirm);

if(confirm == 'y')

{
		fprintf(fp,"%s\t\t%d\t\t%d\t\t%.2f\n",&passdetails.name,passdetails.num_of_seats,passdetails.
bus_num,charges);
			printf("==================");
			printf("\n Reservation Done\n");
			printf("==================");
			printf("\nPress any key to go back to Main menu");
		}
		else
		{
			if(confirm=='n'){
				printf("\nReservation Not Done!\nPress any key to go back to  Main menu!");
			}
			else
			{
				printf("\nInvalid choice entered! Enter again-----> ");
				goto start;
			}
		}
		fclose(fp);
		getch();
}

float charge(int bus_num,int num_of_seats)
{
```

```
if (bus_num==1001)

{
        return(500*num_of_seats);
}
if (bus_num==1002)
{
        return(500*num_of_seats);
}
if (bus_num==1003)
{
        return(450*num_of_seats);
}
if (bus_num==1004)
{
        return(450*num_of_seats);
}
if (bus_num==1005)
{
        return(400*num_of_seats);
}
if (bus_num==1006)
{
        return(400*num_of_seats);
}
if (bus_num==1007)
```

```
	{
		return(350*num_of_seats);
	}
	if (bus_num==1008)
	{
		return(350*num_of_seats);
	}
	if (bus_num==1009)
	{
		return(600*num_of_seats);
	}
	if (bus_num==1010)
	{
		return(600*num_of_seats);
	}
}

void printticket(char name[],int num_of_seats,int bus_num,float charges)
{
	system("cls");
	printf("-------------------\n");
	printf("\tTICKET\n");
	printf("-------------------\n\n");
	printf("Name:\t\t\t%s",name);
	printf("\nNumber Of Seats:\t%d",num_of_seats);
	printf("\nbus Number:\t\t%d",bus_num);
```

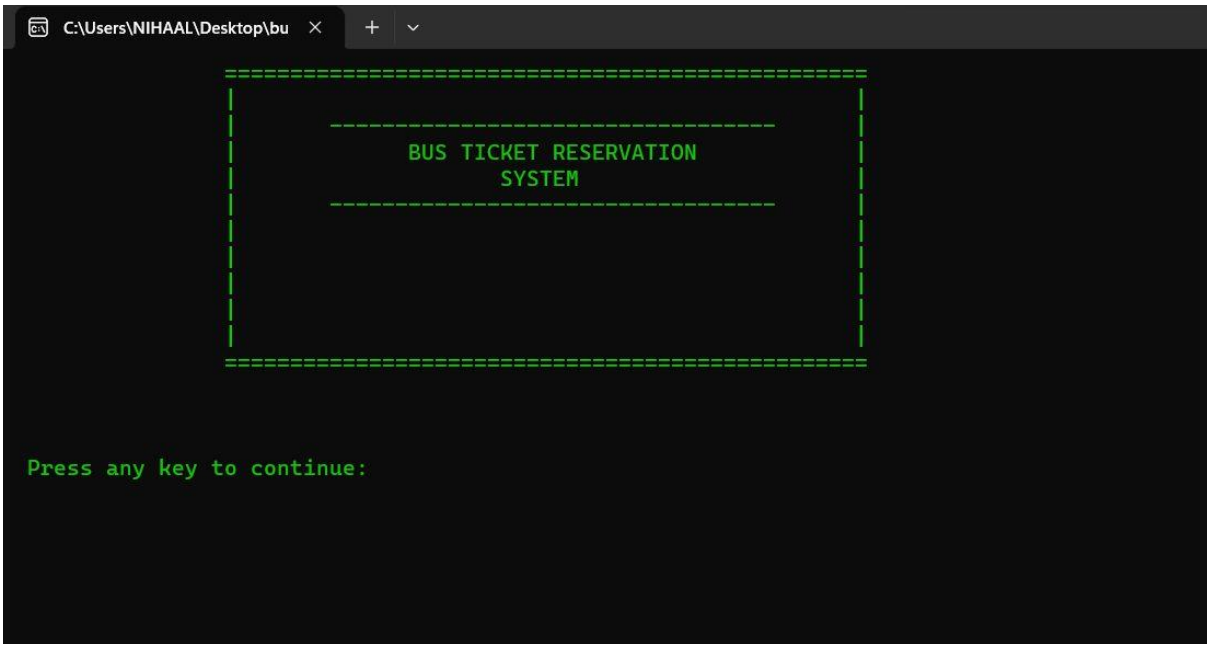
```
        specificbus(bus_num);

        printf("\nCharges:\t\t%.2f",charges);

}



void specificbus(int bus_num)

{


        if (bus_num==1001)

        {

                printf("\nbus:\t\t\tGodavari Travels ");

                printf("\nDestination:\t\tDharan to Kavre");

                printf("\nDeparture:\t\t9am ");

        }

        if (bus_num==1002)

        {

                printf("\nbus:\t\t\tDevit Travels ");

                printf("\nDestination:\t\tKavre to Dharan");

                printf("\nDeparture:\t\t12pm");

        }

        if (bus_num==1003)

        {

                printf("\nbus:\t\t\tHero Travels ");

                printf("\nDestination:\t\tBenighat to Pokhara");

                printf("\nDeparture:\t\t8am");

        }
```

```
if (bus_num==1004)
{
	printf("\nbus:\t\t\tSuper Deluxe ");
	printf("\nDestination:\t\tPokhara to Benighat");
	printf("\nDeparture:\t\t11am ");
}
if (bus_num==1005)
{
	printf("\nbus:\t\t\tSai Baba Travels ");
	printf("\nDestination:\t\tMaitidevi to Janakpur");
	printf("\nDeparture:\t\t7am");
}
if (bus_num==1006)
{
	printf("\nbus:\t\t\tShine On Travels ");
	printf("\nDestination:\t\tJanakpur to Maitidevi ");
	printf("\nDeparture:\t\t9.30am ");
}
if (bus_num==1007)
{
	printf("\nbus:\t\t\tMayur Travels");
	printf("\nDestination:\t\tHumla toJumla ");
	printf("\nDeparture:\t\t1pm ");
}
if (bus_num==1008)
{
```

```
                printf("\nbus:\t\t\tShree Travels ");

                printf("\n Destination:\t\tJumla to Humla");

                printf("\nDeparture:\t\t4pm ");

        }

        if (bus_num==1009)

        {

                printf("\nbus:\t\t\tKathmandu Express");

                printf("\nDestination:\t\tKathmandu to Pokhara");

                printf("\nDeparture:\t\t3.35pm ");

        }

        if (bus_num==1010)

        {

                printf("\nbus:\t\t\tSajha Yatayat");

                printf("\nDestination:\t\tPokhara to Kathmandu");

                printf("\nDeparture:\t\t1.15 ");

        }

}


void login()

{

        int a=0,i=0;

    char uname[10],c=' ';

    char pword[10],code[10];

    char user[10];

    char pass[10];

    do
```

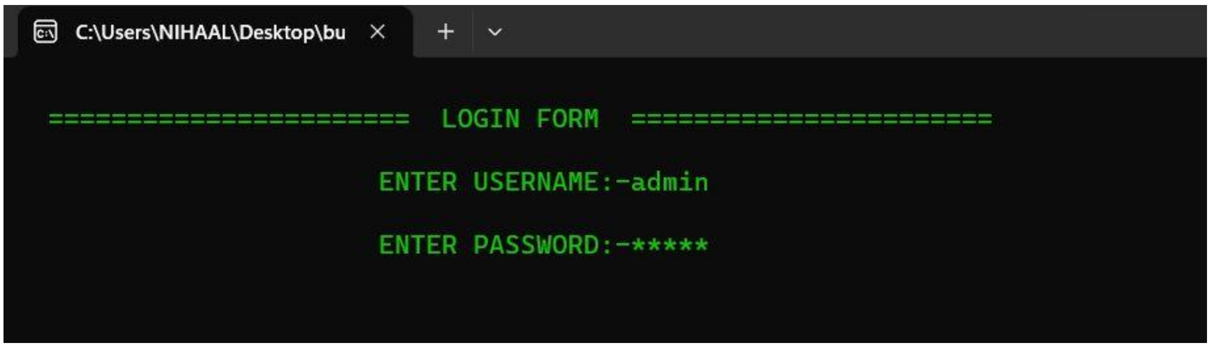
```
{


   printf("\n  =======================  LOGIN FORM  =======================\n  ");

   printf(" \n                       ENTER USERNAME:-");

         scanf("%s", &uname);

         printf(" \n                       ENTER PASSWORD:-");

         while(i<10)

         {

            pword[i]=getch();

            c=pword[i];

            if(c==13) break;

            else printf("*");

            i++;

         }

         pword[i]='\0';

         i=0;

                  if(strcmp(uname,"admin")==0 && strcmp(pword,"pass")==0)

         {

         printf("  \n\n\n       WELCOME TO OUR  BUS RESERVATION SYSTEM !! YOUR LOGIN
IS SUCCESSFUL");

         printf("\n\n\n\t\t\tPress any key to continue...");

         getch();

         break;

         }

         else

         {
                printf("\n        SORRY !!!!  LOGIN IS UNSUCESSFUL");
                a++;


                getch();
                system("cls");
        }
}
        while(a<=2);
        if (a>2)
        {
                printf("\nSorry you have entered the wrong username and password for four
times!!!");


                getch();


                }
                system("cls");
}
```

SCREENSHOT

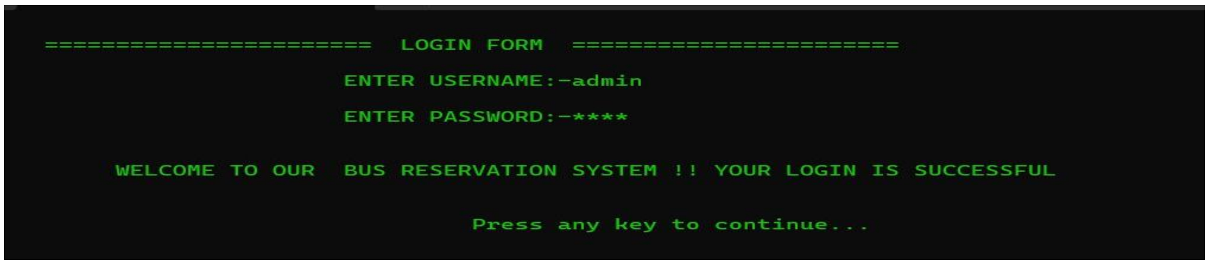


```
==================================================
|                                                |
|       ----------------------------------       |
|           BUS TICKET RESERVATION               |
|                  SYSTEM                        |
|       ----------------------------------       |
|                                                |
|                                                |
|                                                |
|                                                |
==================================================


Press any key to continue:
```

Login page:

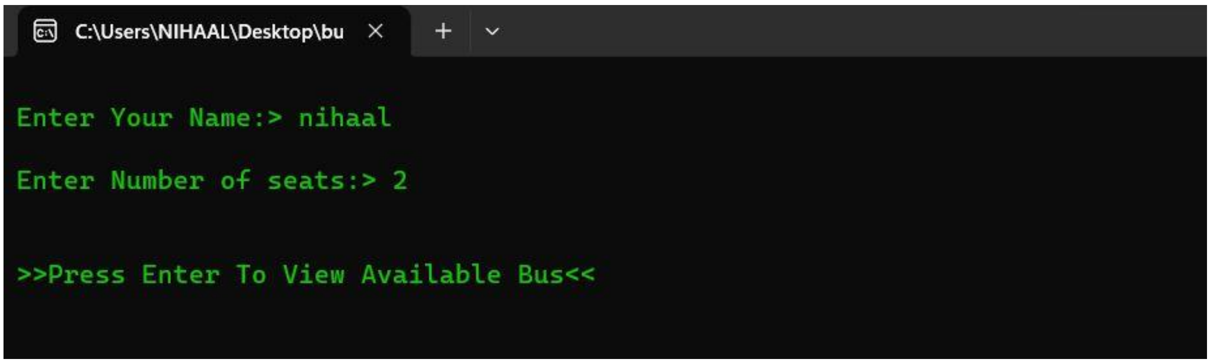


```
=======================  LOGIN FORM  =======================

                ENTER USERNAME:-admin

                ENTER PASSWORD:-*****
```



```
=======================  LOGIN FORM  =======================
                ENTER USERNAME:-admin
                ENTER PASSWORD:-****

     WELCOME TO OUR  BUS RESERVATION SYSTEM !! YOUR LOGIN IS SUCCESSFUL

                           Press any key to continue...
```

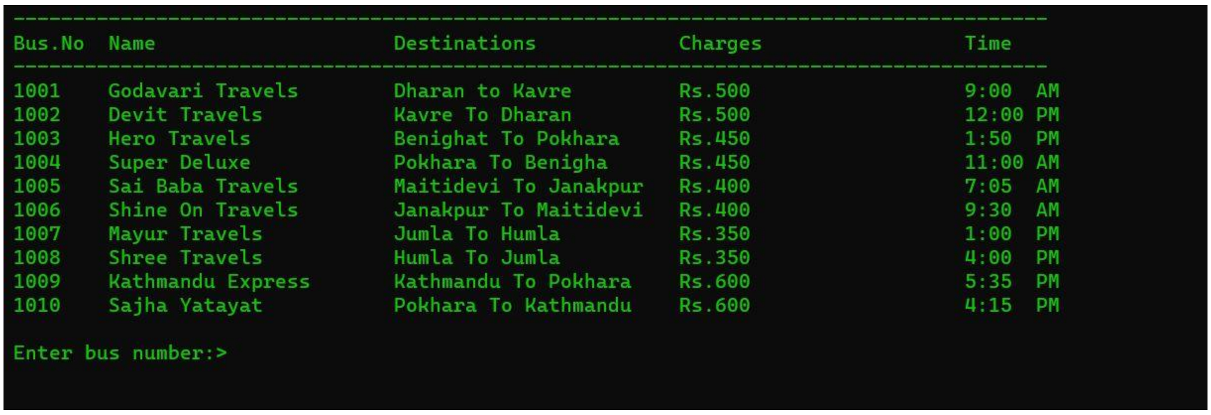
Reservation of seat:



C:\Users\NIHAAL\Desktop\bu

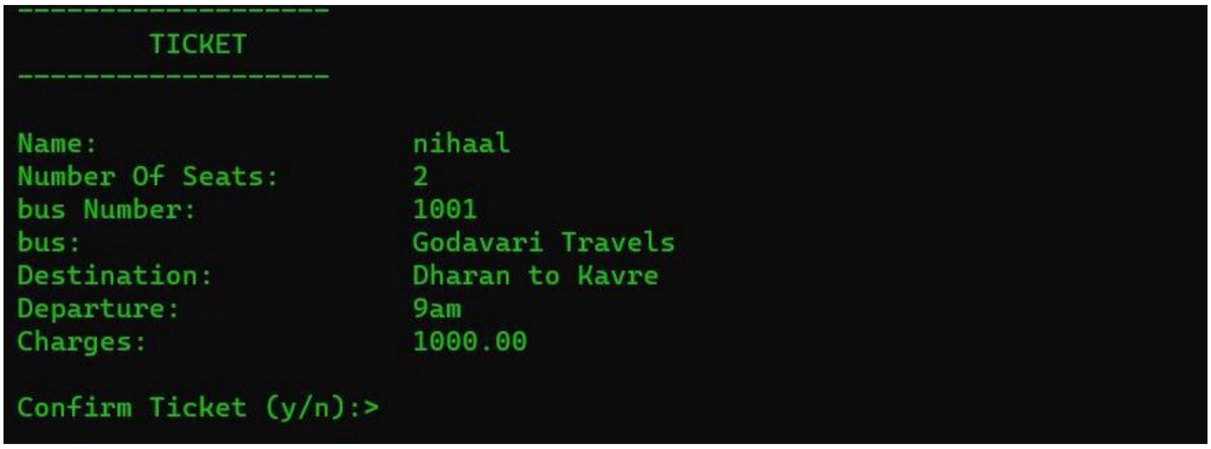
```
Enter Your Name:> nihaal

Enter Number of seats:> 2


>>Press Enter To View Available Bus<<
```

Check availability bus:



```
---------------------------------------------------------------------------------------
Bus.No  Name                    Destinations            Charges                 Time
---------------------------------------------------------------------------------------
1001    Godavari Travels        Dharan to Kavre         Rs.500                  9:00  AM
1002    Devit Travels           Kavre To Dharan         Rs.500                  12:00 PM
1003    Hero Travels            Benighat To Pokhara     Rs.450                  1:50  PM
1004    Super Deluxe            Pokhara To Benigha      Rs.450                  11:00 AM
1005    Sai Baba Travels        Maitidevi To Janakpur   Rs.400                  7:05  AM
1006    Shine On Travels        Janakpur To Maitidevi   Rs.400                  9:30  AM
1007    Mayur Travels           Jumla To Humla          Rs.350                  1:00  PM
1008    Shree Travels           Humla To Jumla          Rs.350                  4:00  PM
1009    Kathmandu Express       Kathmandu To Pokhara    Rs.600                  5:35  PM
1010    Sajha Yatayat           Pokhara To Kathmandu    Rs.600                  4:15  PM

Enter bus number:>
```

Tickets booked successfully



```
--------------------
        TICKET
--------------------

Name:                    nihaal
Number Of Seats:         2
bus Number:              1001
bus:                     Godavari Travels
Destination:             Dharan to Kavre
Departure:               9am
Charges:                 1000.00

Confirm Ticket (y/n):>
```

Reference:

1. GitHub: GitHub is a popular platform for hosting and sharing code.
2. You can use the GitHub search feature to find open-source projects and code related to virtual wallets in C. https://github.com/

3. 2. SourceForge: SourceForge hosts a variety of open-source software projects, including financial and wallet-related applications in C. https://sourceforge.net/
4. 3. CodeProject: CodeProject is a community of developers that shares code samples and projects. You may find C-based virtual wallet projects and code examples there. https://www.codeproject.com/
5. 4. Stack Overflow: Stack Overflow is a programming Q&A community where developers often share code snippets and solutions to specific problems. You can search for questions related to virtual wallets in C to find relevant code examples. https://stackoverflow.com/
6. 5. Online Coding Forums: Online programming forums like Reddit's r/C_Programming or similar communities can also be a good place to find discussions and code snippets related to virtual wallet implementations in C. https://www.reddit.com/r/C_Programming/