Lab/Project Report

Only for course Teacher						
		Needs Improvement	Developing	Sufficient	Above Average	Total Mark
Allocate mark & Percentage		25%	50%	75%	100%	25
Problem understanding & Analysis	7					
Implementation	8					
Report writing	10					
					Total obtained mark	
Comments						

Semester: Spring 2024

Team Name: Dare to Dream

Batch: 231 **Section:40_B**

Course Code: SE133 **Course Name: Software Development Capstone Project**

Course Teacher Name: Md. Abdul Hye Zebon

Designation: Lecturer

Submission Date: 10.06.24

Bus Ticket Reservation System

Team Dare to Dream

Sl no	Name	ID
1	Md Raihan Chowdhoury	0242310005341002
2	Md Alif Khan	232-35-740
3	Romana Akter Romky	0242310005341073
4	Mst. Jannatul Ferdus Asha	0242310005341064

Source Code: Here

Introduction

The transportation industry is constantly evolving, and the demand for convenient and accessible travel booking options is ever-growing. In this dynamic landscape, traditional bus ticketing systems with physical booths and paper tickets can be cumbersome and time-consuming. The Bus Ticket Reservation System project addresses this need by offering a cutting-edge solution – a user-friendly online platform that revolutionizes the way passengers book bus tickets.

This project goes beyond simply automating the booking process. It aims to create a seamless and integrated experience for passengers, from initial route exploration to ticket management. Gone are the days of waiting in long queues or scrambling to find a ticketing booth. With the Bus Ticket Reservation System, passengers can book their bus journeys from the comfort of their homes or anywhere with an internet connection, 24/7.

The system leverages modern technology to provide passengers with real-time information on available routes, seat configurations, and pricing. This empowers them to make informed travel decisions and secure their preferred seats with ease. Additionally, the project prioritizes security by incorporating secure online payment gateways, ensuring a safe and trustworthy booking experience.

Objective

The primary objective of the Bus Ticket Reservation System project is to develop a web-based application for booking bus tickets. This system will automate various functionalities, including:

- **Search and Availability:** Passengers can easily search for available bus routes by specifying their origin, destination, and travel date. The system will display real-time seat availability for different bus schedules.
- **Online Booking:** Passengers can conveniently book tickets directly through the platform. The system will guide them through a user-friendly interface to select their desired seats, provide passenger information, and securely process payments.
- **Ticket Management:** E-tickets will be issued electronically after successful booking. Passengers can access, manage, and even cancel their bookings through their online accounts.
- **Inventory Management:** Bus companies can manage their fleet information, update seat availability, and monitor booking trends in real-time through an administrative dashboard.

Benefits

The Bus Ticket Reservation System offers numerous benefits for both passengers and bus companies:

Passenger Benefits:

- **Convenience:** 24/7 online booking from anywhere with an internet connection.
- **Efficiency:** Quick and easy search and booking process eliminates long queues.
- **Transparency:** Real-time information on seat availability and fares.
- **Security:** Secure online payment processing for a safe booking experience.

Bus Company Benefits:

- **Reduced Operational Costs:** Lower overhead expenses from eliminating physical ticketing booths.
- **Increased Efficiency:** Streamlined booking process and automated inventory management.
- **Improved Customer Service:** Faster and more convenient booking options for passengers.
- **Valuable Data Insights:** Real-time data on bookings and passenger trends for better decision-making.

Library

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

Including these header files in your C program makes the functions and definitions they contain available for your code to use. Each header file has a specific purpose, so you'll typically include only the ones you need for the functionalities your program requires. For portability, it's best to stick to standard C libraries like stdio.h, stdlib.h, string.h, and avoid non-standard headers like conio.h.

Structure Declare

```
struct booking_history
{
    int user_id;
    char name[50];
    char phn_no[50];
    char destination[50];
    char status[10];
    char time[50];
    char bus_name[50];
    int seat_no;
    int fare;
    int serial;
    int selected_seat;
    int selected_destination;
    int selected_bus;
    int selected_time;
};
struct booking_history booking_historys[100];
struct booking_history newbooking_history;
int history_count=0;
```

The provided code defines a structure named booking_history in C, which groups together related data about a booking. The structure contains the following data:

- int user_id:An integer to store the passenger’s ID.
- char name[50]:An array to store the passenger’s Name.
- char phn_no[50]:An array to store the passenger’s Phone no.
- char destination[50]:An array to store the passenger’s Destination.
- char status[10]:An array to store the Ticket Status.
- char time[50]:An array to store the Ticket Time.
- char bus_name[50]:An array to store the Bus Name.
- int seat_no:An integer to store the passenger’s seat no.
- int serial:An integer to store the Ticket Serial No.
- int selected_seat:An integer to store the passenger’s selected Seat no.
- int selected_destination:An integer to store the passenger’s Selected Destination Serial.
- int selected_bus:An integer to store the passenger’s Selected Bus Serial.
- int selected_time:An integer to store the passenger’s Selected Time Serial.
- struct booking_history booking_historys[100]: An array to store 100 history.
- struct booking_history newbooking_history: An array to store a temporary history.
- int history_count=0:A variable initialized to 0 that keeps track of the current number of history in the array.

```
struct sign_up
{
    char username[10];
    char password[10];
    int serial;
};
struct sign_up sign_ups[10];
struct sign_up new_sign_up;
int user=0;
bool sit[8][6][5][32];
```

- char username[10]:An array to store the User Name.
- char password[10]:An array to store the User Password.
- int serial:An integer to store the User Registration Serial No.
- struct sign_up sign_ups[10]: An array to store 10 Signup history.
- struct sign_up new_sign_up: An array to store Temporary Signup history.
- int user=0:A variable initialized to 0 that keeps track of the current number of Signup history in the array.
- bool sit[8][6][5][32]: A bool type Array o keep track of bus seat.

This structure allows you to create variables that can hold and manage all these related pieces of information together, simplifying data handling in your program. These variables facilitate adding, displaying, searching, updating, and deleting student records within the program.

Main Fucntion

```
int main()
{
    system("color 70");
    user_file_read();
    history_file_read();
    sit_file_read();
    welcome();
    return 0;
}
```

The provided C++ code snippet is the main function of a program. Here's a short description of what it does:

1. `system("color 70")`: This line changes the color of the console window. The parameter "70" sets the background color to gray (7) and the text color to black (0).

2. `user_file_read()`: This function call likely reads user-related data from a file.

3. `history_file_read()`: This function call probably reads historical data from a file.

4. `sit_file_read()`: This function call presumably reads situational or settings data from a file.

5. `welcome()`: This function call displays a welcome message or interface to the user.

6. `return 0`: This ends the main function and returns 0 to indicate successful execution.

Overall, the program sets up the console environment and reads necessary data files before displaying a welcome message.

The “welcome” Function

```
void welcome()
{
    system("cls");
    printf("   \xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\n\n");
    printf("   \xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\n\n");
    printf("                      Welcome to\n");
    printf("              BUS TICKET RESERVATION SYSTEM\n\n");
    printf("   \xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\n\n");
    printf("   \xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\n\n");
    printf("   Press any key to continue... ");
    getchar();
    user_list();
}
```

The `welcome` function is responsible for displaying a welcome message to the user in a formatted manner. Here's a step-by-step description of what it does:

1. `system("cls")`: This clears the console screen to provide a clean output space for the welcome message.

2. `printf("   \xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\n\n")`: This prints a horizontal line made up of extended ASCII characters (represented by `\xcd`), followed by two newline characters. The line is repeated to create a box-like structure around the welcome message.

3. `printf("                      Welcome to\n")`: This prints the text "Welcome to" with some leading spaces for centering.

4. `printf("              BUS TICKET RESERVATION SYSTEM\n\n")`: This prints the text "BUS TICKET RESERVATION SYSTEM" on the next line, with leading spaces for centering, followed by two newline characters.

5. `printf("   \xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\n\n")`: This prints another horizontal line made up of extended ASCII characters, followed by two newline characters, to complete the box structure around the message.

6. `printf("   Press any key to continue... ")`: This prints the prompt "Press any key to continue..." to the console.

7. `getchar()`: This waits for the user to press any key before proceeding, effectively pausing the program.

8. user_list(): This calls the `user_list` function, which presumably handles listing users or some related functionality after the welcome message is acknowledged by the user.

Overall, the `welcome` function provides a neatly formatted welcome screen for a bus ticket reservation system and waits for user interaction before moving on to the next part of the program.

The "user_list" Function

```
void user_list()
{
    int choice;
    while (1)
    {
        system("cls");
        printf("   \xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\n\n");
        printf("     BUS TICKET RESERVATION SYSTEM");
        printf("   \xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\n\n");
        printf("   \xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\n\n")
        printf("   [1] Admin\n\n");
        printf("   [2] Passenger\n\n");
        printf("   [3] Exit\n\n");
        printf("   \xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\n\n");
        printf("\n   ENTER YOUR CHOICE: ");
        scanf("%d",&choice);
        switch (choice)
        {
        case 1:
            login_admin();
            break;
        case 2:
            user_login_signup_menu();
            break;
        case 3:
            exit(1);
            break;
        default:
            invalid_choice();
        }
    }
}
```

The user_list function presents a menu to the user and handles their input to navigate through different parts of the bus ticket reservation system. Here's a detailed breakdown of what it does:

1. int choice;: This declares an integer variable choice to store the user's menu selection.

2. while (1): This starts an infinite loop, meaning the menu will repeatedly display until the user chooses to exit.

3. system("cls");: This clears the console screen to provide a clean slate for displaying the menu.

4. printf statements: These print various parts of the menu to the console. The menu is formatted with extended ASCII characters (\xcd) and newline characters (\n) for visual appeal. The menu options are:

   a) [1] Admin
   b) [2] Passenger
   c) [3] Exit

5. printf("\n   ENTER YOUR CHOICE: ");: This prompts the user to enter their choice.

6. scanf("%d",&choice);: This reads the user's input and stores it in the choice variable.

7. switch (choice): This evaluates the user's choice and executes the corresponding case block:

   1. case 1: Calls the login_admin function, presumably handling the admin login process.
   2. case 2: Calls the user_login_signup_menu function, likely managing the login or signup process for passengers.
   3. case 3: Calls exit(1);, which terminates the program.
   4. default: Calls invalid_choice function, which likely informs the user that their input was invalid and prompts them to try again.

The menu will continuously display and process user inputs until the user chooses the "Exit" option

The “bus_list” Function

```
void bus_list()
{
    while (1)
    {
        system("cls");
        printf("\n\n   \xcd\xcd\xcd\xcd\xcd\xcd\xcd\n");
        printf("     BUS TICKET RESERVATION SYSTEM");
        printf("\n\n   \xcd\xcd\xcd\xcd\xcd\xcd\xcd\n");
        printf("\n\n   \xcd\xcd\xcd\xcd\xcd\xcd\xcd\n");
        for (int i = 0; i < 5; i++)
        {
            printf("   [%d]    %s\n\n",i+1,bus[i]);
        }
        printf("\n\n   \xcd\xcd\xcd\xcd\xcd\xcd\xcd\n");
        if (selected_menu==1)
        {
            fflush(stdin);
            printf("\n   Press any key to continue... ");
            getchar();
            system("cls");
            if (temp==-1)
            {
                admin_menu();
            }
            user_menu();
        }
        printf("\n   ENTER YOUR CHOICE: ");
        scanf("%d", &selected_bus);
        system("cls");
          if (selected_bus >= 1 && selected_bus <= 5)
        {
            if (selected_menu==2)
            {
                strcpy(newbooking_history.bus_name,bus[selecte
                                                    d_bus-1]);
                seat_view();
                break;
            }
            break;
        }
        else
        {
            invalid_choice();
        }
    }
}
```

The bus_list function is responsible for displaying a list of available buses to the user and handling their selection. Here's a detailed explanation of the function:

1. Infinite Loop: The function operates within an infinite loop (while (1)), continuously displaying the bus list and processing user inputs until a valid choice is made.

2. Clear Console: system("cls"); clears the console screen to present a clean interface.

3. Printing the Menu:

   a. Various printf statements print the header and format the bus list menu.
   b. A for loop iterates through an array of bus names (bus) and prints each bus with an index. The buses are displayed in the format [i] bus_name, where i is the bus number (1 through 5).

4. Check Menu Source:

   - if (selected_menu==1): If selected_menu is 1, indicating an admin user, the program waits for any key press (getchar()) to continue. Then, it clears the screen again and calls either admin_menu() or user_menu() based on the value of temp. After this, the loop will restart.

5. User Choice:

   a. printf("\n   ENTER YOUR CHOICE: ");: Prompts the user to enter their choice.
   b. scanf("%d", &selected_bus);: Reads the user's input and stores it in selected_bus.

6. Validate Choice:

- if (selected_bus >= 1 && selected_bus <= 5): If the user's choice is within the valid range (1 to 5):
  - if (selected_menu==2): If selected_menu is 2, indicating a passenger user:
    - Copies the selected bus name to newbooking_history.bus_name.
    - Calls seat_view(), which likely displays the seat view and handles the seat booking process.
    - Breaks out of the loop.
  - Breaks out of the loop.
- else: If the user's choice is invalid:

- Calls invalid_choice(), which likely informs the user of the invalid input and prompts them to try again.

Overall, the bus_list function repeatedly displays a list of buses, processes the user's selection, and handles different scenarios based on whether the user is an admin or a passenger. If a valid bus is selected, it proceeds to the next step in the reservation process.

The "invalid_choice" Function

```
void invalid_choice()
{
    system("cls");
    printf("\n\n   \xcd\xcd\xcd\xcd\xcd\xcd\xcd\n");
    printf("        BUS TICKET RESERVATION SYSTEM");
    printf("\n\n   \xcd\xcd\xcd\xcd\xcd\xcd\xcd\n");
    printf("              Invalid Selection.\n\n");
    printf("\n\n   \xcd\xcd\xcd\xcd\xcd\xcd\xcd\n");
    printf("   Press any key to continue... ");
    getchar();
}
```

The invalid_choice function handles the situation when a user makes an invalid selection in the bus ticket reservation system. Here's a detailed explanation of what it does:

1. Clear Console: system("cls"); clears the console screen to ensure the message is displayed clearly without any previous content interfering.

2. Print Invalid Selection Message:

   - Various printf statements are used to format the message with extended ASCII characters (\xcd) for decorative lines and to create a clean and professional-looking output.
   - The message "Invalid Selection" is displayed prominently in the center.

3. Wait for User Interaction:

   - printf("   Press any key to continue... ");: This prompts the user to press any key to continue.
   - getchar();: This waits for the user to press any key before proceeding, effectively pausing the program until user interaction.

The invalid_choice function provides a clear and formatted message to the user when they make an invalid selection and waits for the user to acknowledge the message before allowing them to proceed. This helps to improve user experience by giving feedback and a chance to correct their input.

The “boarding_time” Function

```
void boarding_time()
{
    while (1)
    {
        system("cls");
        printf("\n\n   \xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\n");
        printf("     BUS TICKET RESERVATION SYSTEM");
        printf("\n\n   \xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\n");
        printf("\n\n   \xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\n");
        for (int i = 0; i < 6; i++)
        {
            printf("   [%d]    %s\n\n",i+1,time[i]);
        }
        printf("\n\n   \xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\n");
        printf("\n   ENTER YOUR CHOICE: ");
        scanf("%d",&selected_time);
        if (selected_time >= 1 && selected_time <= 6)
        {
            if (selected_menu==2)
            {
                    strcpy(newbooking_history.time,time[select
                                                    ed_time-1]);
            }
            bus_list();
            break;
        }
        else
        {
            invalid_choice();
        }
    }
}
```

The boarding_time function is responsible for displaying available boarding times to the user and allowing them to make a selection. Here’s a detailed breakdown of its functionality:

1. Infinite Loop: The function runs inside an infinite loop (while (1)), which continues until a valid boarding time is selected.
2. Clear Console: system("cls"); clears the console screen to provide a clean interface for displaying the boarding times.
3. Print Boarding Times Menu:
   - Several printf statements are used to format and display the boarding times menu, including decorative lines made with extended ASCII characters (\xcd).
   - A for loop iterates through an array of boarding times (time) and prints each time with an index. The boarding times are displayed in the format [i] time, where i is the time number (1 through 6).

4. Prompt User for Choice:
   - printf("\n   ENTER YOUR CHOICE: ");: Prompts the user to enter their choice.
   - scanf("%d",&selected_time);: Reads the user's input and stores it in the selected_time variable.
5. Validate Choice:
   - if (selected_time >= 1 && selected_time <= 6): If the user's choice is within the valid range (1 to 6):
     - if (selected_menu==2): If selected_menu is 2, indicating a passenger user:
       - Copies the selected boarding time to newbooking_history.time.
       - Calls bus_list(), which likely displays the bus list menu.
       - Breaks out of the loop.
   - else: If the user's choice is invalid:

Calls invalid_choice(), which likely informs the user of the invalid input and prompts them to try again.

The boarding_time function repeatedly displays a list of available boarding times, processes the user's selection, and validates it. If a valid time is selected, it updates the booking history with the selected time (if applicable) and proceeds to the bus list menu. If the selection is invalid, it provides feedback to the user and prompts them to try again.

The "bus_destination" Function

```
void bus_destination()
{
    while (1)
    {
        system("cls");
        printf("\n\n   \xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\n");
        printf("     BUS TICKET RESERVATION SYSTEM");
        printf("\n\n   \xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\n");
        printf("\n\n   \xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\n");
        for (int i = 0; i < 8; i++)
        {
            printf("   [%d]    %s\n\n",i+1,destination[i]);
        }
        printf("\n\n   \xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\n");
        printf("\n   ENTER YOUR CHOICE: ");
        scanf("%d",&selected_destination);
        system("cls");
        if (selected_destination >= 1 && selected_destination
                                                    <= 8)
        {
            if (selected_menu==2)
            {
                    strcpy(newbooking_history.destination,dest
                              ination[selected_destination-1]);
```

```
                    newbooking_history.fare=fee[selected_desti
                                                        nation-1];
            }
            boarding_time();
            break;
        }
        else
        {
            invalid_choice();
        }
    }
}
```

The bus_destination function is responsible for displaying a list of available bus destinations to the user, allowing them to make a selection, and then proceeding to the boarding time selection. Here's a detailed breakdown of its functionality:

1. Infinite Loop: The function runs inside an infinite loop (while (1)), which continues until a valid destination is selected.
2. Clear Console: system("cls"); clears the console screen to provide a clean interface for displaying the destinations.
3. Print Destinations Menu:
   - Several printf statements are used to format and display the destinations menu, including decorative lines made with extended ASCII characters (\xcd).
   - The header "DESTINATION" is printed in a formatted manner.
   - A for loop iterates through an array of destinations (destination) and prints each destination with an index. The destinations are displayed in the format [i] destination, where i is the destination number (1 through 8).
4. Prompt User for Choice:
   - printf("\n ENTER YOUR CHOICE: ");: Prompts the user to enter their choice.
   - scanf("%d",&selected_destination);: Reads the user's input and stores it in the selected_destination variable.
5. Validate Choice:
   - if (selected_destination >= 1 && selected_destination <= 8): If the user's choice is within the valid range (1 to 8):
     - if (selected_menu==2): If selected_menu is 2, indicating a passenger user:
       - Copies the selected destination to newbooking_history.destination.
       - Sets the fare for the selected destination in newbooking_history.fare.

- Calls boarding_time(), which likely displays the boarding time menu and handles the boarding time selection.
- Breaks out of the loop.

- else: If the user's choice is invalid:

Calls invalid_choice(), which likely informs the user of the invalid input and prompts them to try again.

The bus_destination function repeatedly displays a list of available destinations, processes the user's selection, and validates it. If a valid destination is selected, it updates the booking history with the selected destination and fare (if applicable) and proceeds to the boarding time menu. If the selection is invalid, it provides feedback to the user and prompts them to try again.

The "seat_view" Function

```
void seat_view()
{
    while (1)
    {
        printf("\n\n   \xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\n");
        printf("\t\t\t     BUS TICKET RESERVATION SYSTEM");
        printf("\n\n   \xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\n");
        printf("\n\n   \xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\n");
        for (int i = 1; i <= 8; i++)
        {
                if (sit[selected_destination-1][selected_time-
                                    1][selected_bus-1][i-1]==1)
                {
                    printf("      [%-2d] %-10s",i,"Booked");
                }else
                {
                    printf("      [%-2d] %-10s",i,"Available");
                }
                if (sit[selected_destination-1][selected_time-
                              1][selected_bus-1][i+8-1]==1)
                {
                    printf("   [%-2d] %-10s",i+8,"Booked");
                }else
                {
                    printf("   [%-2d] %-10s",i+8,"Available");
                }
                if (sit[selected_destination-1][selected_time-
                              1][selected_bus-1][i+16-1]==1)
                {
                    printf("      [%-2d] %-10s",i+16,"Booked");
                }else
                {
                    printf("      [%-2d] %-
                                        10s",i+16,"Available");
```

```
                }
                if (sit[selected_destination-1][selected_time-
                            1][selected_bus-1][i+24-1]==1)
                {
                    printf("   [%-2d] %-10s",i+24,"Booked");
                }else
                {
                    printf("   [%-2d] %-10s",i+24,"Available");
                }
                printf("\n");
        }
        printf("\n\n   \xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\n");
        printf("\n   ENTER SIT NUMBER: ");
        scanf("%d",&selected_seat);
        system("cls");
        if (selected_seat >= 1 && selected_seat <= 32 &&
               sit[selected_destination-1][selected_time-
               1][selected_bus-1][selected_seat-1]!=1)
        {
            if (selected_menu==2)
            {
                newbooking_history.seat_no=selected_seat;
                ask_user_info();
            }
            break;
        }
        else
        {
            invalid_choice();
        }
    }
}
```

The seat_view function is responsible for displaying the seat plan of the selected bus, allowing the user to choose a seat, and validating their selection. Here's a detailed breakdown of its functionality:

1. Infinite Loop: The function runs inside an infinite loop (while (1)), which continues until a valid seat is selected.
2. Print Seat Plan:
   - Header: Various printf statements are used to format and display the seat plan, including decorative lines made with extended ASCII characters (\xcd).
   - Seat Status: A nested loop iterates through a 3D array sit that holds the booking status of seats. The array is indexed based on the selected destination, time, and bus. Each seat is checked and printed as either "Booked" or "Available". The seats are displayed in a 4-column format for a total of 32 seats.
3. Prompt User for Seat Number:

- printf("\n ENTER SIT NUMBER: ");: Prompts the user to enter the seat number they want to book.
- scanf("%d",&selected_seat);: Reads the user's input and stores it in the selected_seat variable.
- system("cls");: Clears the console screen for the next iteration if necessary.

4. Validate Seat Selection:
   - if (selected_seat >= 1 && selected_seat <= 32 && sit[selected_destination-1][selected_time-1][selected_bus-1][selected_seat-1] != 1): Checks if the selected seat number is within the valid range (1 to 32) and if the seat is not already booked.
     - if (selected_menu==2): If selected_menu is 2, indicating a passenger user:
       - Updates newbooking_history.seat_no with the selected seat number.
       - Calls ask_user_info(), which likely collects additional user information needed for the booking.
     - Breaks out of the loop, completing the seat selection process.
   - else: If the seat number is invalid or the seat is already booked:
     - Calls invalid_choice(), which likely informs the user of the invalid input and prompts them to try again.

The seat_view function displays the seat plan, allowing the user to select a seat. It validates the seat selection to ensure it is within the valid range and not already booked. If a valid seat is selected, it updates the booking information and proceeds to the next step. If the selection is invalid, it provides feedback and prompts the user to try again.

The "ask_user_info" Function

```
void ask_user_info()
{
    char str[30],phnno[12];
    system("cls");
    printf("\n\n   \xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\n");
    printf("\t     BUS TICKET RESERVATION SYSTEM");
    printf("\n\n   \xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\n");
    printf("\n\n   \xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\n");
    printf("\n   ENTER YOUR NAME: ");
    fflush(stdin);
    gets(str);
    strcpy(newbooking_history.name,str);
    printf("   ENTER YOUR PHONE NO: ");
    fflush(stdin);
    scanf(" %s",phnno);
    strcpy(newbooking_history.phn_no,phnno);
    ticket();
}
```

The ask_user_info function is responsible for collecting the user's personal information, such as their name and phone number, and then proceeding to the ticket generation. Here's a detailed breakdown of its functionality:

1. Clear Console: system("cls"); clears the console screen to provide a clean interface for collecting user information.
2. Print Header:

Various printf statements are used to format and display the header, including decorative lines made with extended ASCII characters (\xcd).

3. Prompt User for Name:
   - printf("\n ENTER YOUR NAME: ");: Prompts the user to enter their name.
   - fflush(stdin);: Ensures the input buffer is cleared before reading the name to avoid any leftover input.
   - gets(str);: Reads the user's input and stores it in the str variable. Note: gets is unsafe due to potential buffer overflow; it is better to use fgets in practice.
   - strcpy(newbooking_history.name, str);: Copies the entered name into the name field of the newbooking_history structure.
4. Prompt User for Phone Number:
   - printf(" ENTER YOUR PHONE NO: ");: Prompts the user to enter their phone number.
   - fflush(stdin);: Clears the input buffer again before reading the phone number.
   - scanf(" %s", phnno);: Reads the user's input and stores it in the phnno variable.
   - strcpy(newbooking_history.phn_no, phnno);: Copies the entered phone number into the phn_no field of the newbooking_history structure.
5. Generate Ticket:
   - ticket();: Calls the ticket function, which likely handles the generation and display of the ticket based on the collected booking information.

The ask_user_info function collects the user's name and phone number, stores this information in the booking history, and then calls the ticket function to proceed with the ticket generation. This function ensures that necessary personal information is obtained before finalizing the booking process.

The "ticket" Function

```
void ticket()
{
    char status[]="Booked";
    system("cls");
    char a[]="Wishing you a smooth trip filled with adventures
                                                and excitement.";
    char b[]="";
```

```
    sit[selected_destination-1][selected_time-1][selected_bus-
                                        1][selected_seat-1]=1;
    printf("\n\n   \xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\n");
    printf("  |  \t\t\t  Ticket: %-25s \tSerial: %-
          2d  |",newbooking_history.bus_name,history_count+1);
    printf("\n   \xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\n");
    printf("  |  \tDestination: %-15s\t\tDeparture time: %-
     8s    |\n",newbooking_history.destination,newbooking_hist
                                        ory.time);
    fflush(stdin);
    printf("  |  \tName: %-22s\t\tPhn no: %-
     15s     |\n",newbooking_history.name,newbooking_history.p
                                        hn_no);
    printf("  |  \tSeat no: %-19d\t\tFare: BDT %-
     15d   |\n",newbooking_history.seat_no,newbooking_history.
                                                  fare);
    printf("  |    %-66s   |\n",b);
    printf("  |    %-66s   |\n",a);
    printf("   \xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\n");
    fflush(stdin);
    printf("\n   Press any key to continue... ");
    getchar();
    system("cls");
    newbooking_history.serial=history_count+1;
    strcpy(newbooking_history.status,status);
    newbooking_history.user_id=temp;
    newbooking_history.selected_destination=selected_destinati
                                                        on;
    newbooking_history.selected_bus=selected_bus;
    newbooking_history.selected_seat=selected_seat;
    newbooking_history.selected_time=selected_time;
    sit_file_write();
    booking_historys[history_count++]=newbooking_history;
    history_file_write();
    if (temp==-1)
    {
        admin_menu();
    }
    user_menu();
}
```

The ticket function is responsible for generating and displaying a ticket for the booked bus seat. Here's a detailed breakdown of its functionality:

1. Clear Console: `system("cls");` clears the console screen to provide a clean interface for displaying the ticket.
2. Define Ticket Content:
   - `char status[] = "Booked";`: Initializes a string variable status with the value "Booked". This indicates the status of the booking.

- char a[] = "Wishing you a smooth trip filled with adventures and excitement.";: Initializes a string variable a with a message for the passenger.
- char b[] = "";: Initializes an empty string variable b. It seems to be intended for additional information, but currently unused.

3. Update Seat Status:
   - sit[selected_destination-1][selected_time-1][selected_bus-1][selected_seat-1] = 1;: Updates the status of the booked seat to indicate it is now occupied.
4. Print Ticket Information:
   - Various printf statements are used to format and display the ticket details, including the bus name, destination, departure time, passenger name, seat number, fare, and a farewell message. Decorative lines made with extended ASCII characters (\xcd) are also used to enhance the appearance of the ticket.
5. Wait for User Interaction:
   - printf("\n Press any key to continue... ");: Prompts the user to press any key to continue.
   - getchar();: Waits for the user to press any key before proceeding, effectively pausing the program until user interaction.
6. Update Booking History:
   - Update newbooking_history: Updates various fields of the newbooking_history structure with relevant information about the booking, including the serial number, status, user ID, selected destination, bus, seat, and time.
   - Write to Files: Writes the updated seat and booking history information to files (sit_file_write() and history_file_write()).
7. Redirect to Menu:
   - Redirect to Menu: If the user is an admin (temp == -1), the function redirects to the admin menu using admin_menu(). Otherwise, it redirects to the user menu using user_menu().

The ticket function generates and displays a ticket for the booked bus seat, including relevant details such as destination, departure time, passenger name, seat number, fare, and a farewell message. It updates the booking history with the newly booked seat and writes the updated information to files. Finally, it redirects the user to the appropriate menu based on their role.

The “user_file_read” Function

```
void user_file_read()
{
    FILE *fp;
    int n=2;
    while (n--)
    {
```

```
        fp = fopen("user.txt", "r");
        if (fp == NULL) {
            FILE *fp = fopen("user.txt", "w");
        }
    }
    user = 0;
    while (fscanf(fp, "%[^,] ,%[^,],%d\n",
            sign_ups[user].username, sign_ups[user].password,
     &sign_ups[user].serial) != EOF) {
        user++;
    }
    fclose(fp);
}
```

The user_file_read function is designed to read user information from a file named "user.txt" and populate an array of structures (sign_ups) with this data. Here's a breakdown of its functionality:

1. File Pointer and Opening:
   - FILE *fp;: Declares a file pointer variable fp.
   - int n = 2;: Initializes a counter n with the value 2.
2. File Read Loop:
   - A while loop is used with the condition n--, which means the loop will execute twice.
   - Inside the loop:
     - fp = fopen("user.txt", "r");: Opens the "user.txt" file in read mode and assigns the file pointer to fp.
     - If the file pointer is NULL, indicating that the file doesn't exist or couldn't be opened:
       - FILE *fp = fopen("user.txt", "w");: Opens the file in write mode to create it if it doesn't exist.
       - The purpose of this loop seems to ensure that the file exists and is accessible for reading.
3. User Data Reading:
   - user = 0;: Initializes a variable user to 0, which will be used as an index to populate the sign_ups array.
   - Another loop is used to read data from the file:
     - while (fscanf(fp, "%[^,] ,%[^,],%d\n", sign_ups[user].username, sign_ups[user].password, &sign_ups[user].serial) != EOF) {: Reads data from the file until the end of the file (EOF) is reached.
       - %[^,] is a format specifier that reads characters until a comma is encountered.
       - The data read is stored in the corresponding fields of the sign_ups array.
       - The loop increments the user index after each successful read.
4. File Closure:
   - fclose(fp);: Closes the file after reading all the data.

The user_file_read function ensures the existence of the user data file "user.txt", opens it for reading, reads user information line by line, and populates an array of structures (sign_ups) with this data. It then closes the file once all data has been read.

The “user_file_read” Function

```
void user_file_write()
{
        FILE *fp = fopen("user.txt", "a");
    if (fp == NULL) {
        perror("Error opening file");
        exit(EXIT_FAILURE);
    }
        fprintf(fp, "%s,%s,%d\n", sign_ups[user].username,
          sign_ups[user].password, sign_ups[user].serial);
    fclose(fp);
}
```

The user_file_write function is responsible for writing user information from the sign_ups array to a file named "user.txt". Here's a breakdown of its functionality:

1. File Pointer and Opening:
   - FILE *fp = fopen("user.txt", "a");: Declares a file pointer fp and opens the "user.txt" file in append mode ("a"). This mode allows data to be appended to the end of the file, creating the file if it doesn't exist.
   - Error handling:
     - If the file pointer is NULL, indicating an error in opening the file:
       - perror("Error opening file");: Prints an error message describing the issue.
       - exit(EXIT_FAILURE);: Exits the program with a failure status.
2. Writing Data to File:
   - fprintf(fp, "%s,%s,%d\n", sign_ups[user].username, sign_ups[user].password, sign_ups[user].serial);: Writes user information from the sign_ups array to the file in the format username,password,serial. Each user's information is written as a separate line.
     - sign_ups[user].username, sign_ups[user].password, and sign_ups[user].serial represent the username, password, and serial number of the user, respectively.
3. File Closure:
   - fclose(fp);: Closes the file after writing the data.

The user_file_write function appends user information from the sign_ups array to the "user.txt" file. It handles errors if the file cannot be opened and ensures proper closure of the

file after writing. This function is essential for persisting user data between program executions.

The “history_file_write” Function

```
void history_file_write()
{
  FILE *fp;
  int i;
  fp = fopen("history.txt", "w");
  if (fp == NULL)
  {
    printf("Error opening file!\n");
    exit(1);
  }
  for (i = 0; i < history_count; i++)
  {
    fprintf(fp,
     "%d,%s,%s,%s,%s,%s,%s,%d,%d,%d,%d,%d,%d,%d\n",booking_his
     torys[i].user_id,booking_historys[i].name,booking_history
     s[i].phn_no,booking_historys[i].destination,booking_histo
     rys[i].status,booking_historys[i].time,booking_historys[i]
     .bus_name,booking_historys[i].seat_no,booking_historys[i].
     fare,booking_historys[i].serial,booking_historys[i].selec
     ted_seat,booking_historys[i].selected_destination,booking
     _historys[i].selected_bus,booking_historys[i].selected_ti
     me);
  }
  fclose(fp);
}
```

The history_file_write function is responsible for writing booking history information from the booking_historys array to a file named "history.txt". Here's a breakdown of its functionality:

1. File Pointer and Opening:
   - FILE *fp;: Declares a file pointer fp.
   - fp = fopen("history.txt", "w");: Opens the "history.txt" file in write mode ("w"). This mode truncates the file to zero length or creates a new file if it doesn't exist.
   - Error handling:
     - If the file pointer is NULL, indicating an error in opening the file:
       - Prints an error message.
       - Exits the program.
2. Writing Data to File:
   - A for loop iterates through the booking_historys array:
     - For each entry in the array, fprintf is used to write booking history information to the file in the specified format.

- %d is used for integer values (user_id, seat_no, fare, serial, selected_seat, selected_destination, selected_bus, selected_time).
- %s is used for string values (name, phn_no, destination, status, time, bus_name).

3. File Closure:
   - fclose(fp);: Closes the file after writing the data.

The history_file_write function writes booking history information from the booking_historys array to the "history.txt" file. It ensures proper error handling in case the file cannot be opened and closes the file after writing the data. This function is crucial for maintaining a record of booking history for future reference.

The “history_file_read” Function

```
void history_file_read()
{
    FILE *fp;
    int n=2;
    while (n--)
    {
        fp = fopen("history.txt", "r");
        if (fp == NULL) {
            fp = fopen("history.txt", "w");
        }
    }
    history_count = 0;
    while (fscanf(fp,
     "%d,%[^,],%[^,],%[^,],%[^,],%[^,],%d,%d,%d,%d,%d,%d,
     %d\n",&booking_historys[history_count].user_id,
     booking_historys[history_count].name,
     booking_historys[history_count].phn_no,
     booking_historys[history_count].destination,
     booking_historys[history_count].status,
     booking_historys[history_count].time,booking_historys[his
     tory_count].bus_name,&booking_historys[history_count].sea
     t_no,&booking_historys[history_count].fare,&booking_histo
     rys[history_count].serial,&booking_historys[history_count]
     .selected_seat,&booking_historys[history_count].selected_
     destination,&booking_historys[history_count].selected_bus,
     &booking_historys[history_count].selected_time) != EOF) {
        history_count++;
    }
    fclose(fp);
}
```

The history_file_read function reads booking history information from the "history.txt" file and populates the booking_historys array. Let's break down its functionality:

1. File Pointer and Opening:

- FILE *fp;: Declares a file pointer fp.
- int n = 2;: Initializes a counter n with the value 2.

2. File Read Loop:

- A while loop with the condition n-- is used, which means the loop will execute twice.
- Inside the loop:
  - fp = fopen("history.txt", "r");: Opens the "history.txt" file in read mode and assigns the file pointer to fp.
  - If the file pointer is NULL, indicating that the file doesn't exist or couldn't be opened:
    - fp = fopen("history.txt", "w");: Opens the file in write mode to create it if it doesn't exist.
- The purpose of this loop seems to ensure that the file exists and is accessible for reading.

3. Reading Data from File:
   - history_count = 0;: Initializes the history_count variable to 0 before reading the history.
   - Another loop is used to read data from the file:
     - while (fscanf(fp, "%d,%[^,],%[^,],%[^,],%[^,],%[^,],%d,%d,%d,%d,%d,%d,%d\n", &booking_historys[history_count].user_id, booking_historys[history_count].name, booking_historys[history_count].phn_no, booking_historys[history_count].destination, booking_historys[history_count].status, booking_historys[history_count].time, booking_historys[history_count].bus_name, &booking_historys[history_count].seat_no, &booking_historys[history_count].fare, &booking_historys[history_count].serial, &booking_historys[history_count].selected_seat, &booking_historys[history_count].selected_destination, &booking_historys[history_count].selected_bus, &booking_historys[history_count].selected_time) != EOF) {:
       - Reads data from the file until the end of the file (EOF) is reached.
       - The format string specifies the format of data to be read, including integers and strings separated by commas.
       - The data read is stored in the corresponding fields of the booking_historys array.
       - The loop increments the history_count index after each successful read.

4. File Closure:
   - fclose(fp);: Closes the file after reading all the data.

The history_file_read function ensures the existence of the booking history file "history.txt", opens it for reading, reads booking history information line by line, and populates the booking_historys array with this data. It then closes the file once all data has been read. This function is crucial for retrieving booking history information for display or further processing.

The “sit_file_write” Function

```
void sit_file_write()
{
    FILE *fp;
    int i;
    fp = fopen("seat.txt", "w");
    if (fp == NULL)
    {
        printf("Error opening file!\n");
        exit(1);
    }
    for (int i = 0; i < 8; i++)
    {
        for (int j = 0; j < 6; j++)
        {
            for (int k = 0; k < 5; k++)
            {
                for (int l = 0; l < 32; l++)
                {
                    if (l==31)
                    {
                        fprintf(fp, "%d\n",sit[i][j][k][l]);
                    }else
                    {
                        fprintf(fp, "%d,",sit[i][j][k][l]);
                    }
                }
            }
        }
    }
  fclose(fp);
}
```

The sit_file_write function is responsible for writing seat occupancy information from the sit array to a file named "seat.txt". Let's break down its functionality:

1. File Pointer and Opening:
    a) FILE *fp;: Declares a file pointer fp.
    b) fp = fopen("seat.txt", "w");: Opens the "seat.txt" file in write mode ("w"). This mode truncates the file to zero length or creates a new file if it doesn't exist.
    c) Error handling:
        i. If the file pointer is NULL, indicating an error in opening the file:
            1. Prints an error message.
            2. Exits the program.
2. Writing Data to File:
    a) Nested for loops iterate over the multi-dimensional sit array:
        i. The outermost loop iterates over the 8 destinations.
        ii. The second loop iterates over the 6 different departure times.
        iii. The third loop iterates over the 5 different buses.

iv. The innermost loop iterates over the 32 seats on each bus.
v. Within the innermost loop:
    1. If it's the last seat in the row (l == 31), fprintf writes the seat occupancy value followed by a newline character.
    2. Otherwise, it writes the seat occupancy value followed by a comma to separate values.

3. File Closure:
    a) fclose(fp);: Closes the file after writing the data.

The sit_file_write function writes seat occupancy information from the multi-dimensional sit array to the "seat.txt" file. It handles errors if the file cannot be opened and closes the file after writing the data. This function is essential for persisting seat occupancy data between program executions.

The “sit_file_read” Function

```
void sit_file_read()
{
    FILE *fp;
    int i;
    int n=2;
    while (n--)
    {
        fp = fopen("seat.txt", "r");
        if (fp == NULL)
        {
            fp = fopen("seat.txt", "w");
        }
    }
    for (int i = 0; i < 8; i++)
    {
        for (int j = 0; j < 6; j++)
        {
            for (int k = 0; k < 5; k++)
            {
                for (int l = 0; l < 32; l++)
                {
                    if (l==31)
                    {
                        fscanf(fp, "%d\n",&sit[i][j][k][l]);
                    }else
                    {
                        fscanf(fp, "%d,",&sit[i][j][k][l]);
                    }
                }
            }
        }
    }
  fclose(fp);
}
```

The sit_file_read function is responsible for reading seat occupancy information from the "seat.txt" file and populating the sit array. Here's a breakdown of its functionality:

1. File Pointer and Opening:
   a) FILE *fp;: Declares a file pointer fp.
   b) int n = 2;: Initializes a counter n with the value 2.
2. File Read Loop:
   a) A while loop with the condition n-- is used, meaning the loop will execute twice.
   b) Inside the loop:
      i. fp = fopen("seat.txt", "r");: Opens the "seat.txt" file in read mode and assigns the file pointer to fp.
      ii. If the file pointer is NULL, indicating that the file doesn't exist or couldn't be opened:
         1. fp = fopen("seat.txt", "w");: Opens the file in write mode to create it if it doesn't exist.
3. Reading Data from File:
   a) Nested for loops iterate over the multi-dimensional sit array:
      i. The outermost loop iterates over the 8 destinations.
      ii. The second loop iterates over the 6 different departure times.
      iii. The third loop iterates over the 5 different buses.
      iv. The innermost loop iterates over the 32 seats on each bus.
      v. Within the innermost loop:
         1. If it's the last seat in the row (l == 31), fscanf reads the seat occupancy value followed by a newline character.
         2. Otherwise, it reads the seat occupancy value followed by a comma to separate values.
4. File Closure:
   a) fclose(fp);: Closes the file after reading all the data.

The sit_file_read function ensures the existence of the "seat.txt" file, opens it for reading, reads seat occupancy information line by line, and populates the sit array with this data. It then closes the file once all data has been read. This function is crucial for retrieving seat occupancy information for further processing within the program.

The “login_admin” Function

```
void login_admin()
{
    char user[10];
    char username[] = "Admin";
    char password[] = "admin";
    char pass[11],ch;
    int i=0;
    while (1)
    {
```

```
        system("cls");
        printf("\n\n   \xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\n");
        printf("\t    BUS TICKET RESERVATION SYSTEM");
        printf("\n\n   \xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\n");
        printf("\n\n   \xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\n");
        fflush(stdin);
        printf("\n\n   ENTER USERNAME: ");
        fflush(stdin);
        scanf("%s", user);
        printf(" \n   ENTER PASSWORD: ");
        while (i < 11)
        {
            fflush(stdin);
            ch = getch();
            if (ch == '\r' || ch == '\n')
                break;
            else if (ch == '\b' && i > 0)
            {
                printf("\b \b");
                i--;
            } else {
                pass[i++] = ch;
                printf("*");
            }
        }
        pass[i] = '\0';
        if(strcmp(username,user)==0 &&
               strcmp(password,pass)==0)
        {
            temp=-1;
            admin_menu();
            break;
        }
    }
}
```

The login_admin function implements the login functionality for the administrator in the bus ticket reservation system. Here's a breakdown of its functionality:

1. Variable Declaration:
   a) char user[10];: Declares an array to store the username input by the user.
   b) char username[] = "Admin";: Defines the correct username for the admin.
   c) char password[] = "admin";: Defines the correct password for the admin.
   d) char pass[11], ch;: Declares an array to store the password input character by character, and a variable ch to store each character as it's entered.
   e) int i = 0;: Initializes a counter to keep track of the number of characters entered for the password.

2. Login Loop:
   a) The function runs an infinite loop (while (1)) until a valid username and password combination is entered.
   b) Within each iteration of the loop:
      i. The console screen is cleared (system("cls")) to provide a clean interface.
      ii. The user is prompted to enter their username and password.
      iii. Username and password are read from the user input, with the password characters being masked.
      iv. If the entered username and password match the predefined admin credentials:
         1. The global variable temp is set to -1 to indicate that the user is an admin.
         2. The admin_menu function is called to display the admin menu.
         3. The loop breaks to exit the login process.
3. Comparison and Validation:
   a) strcmp is used to compare the entered username and password with the predefined admin credentials.
   b) If both username and password match, the admin is successfully logged in and proceeds to the admin menu.

This function ensures that only authorized administrators with the correct username and password combination can access the admin functionalities of the system.

The "admin_menu" Function

```
void admin_menu()
{
    while (1)
    {
        system("cls");
        printf("\n\n   \xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\n");
        printf("     BUS TICKET RESERVATION SYSTEM");
        printf("\n\n   \xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\n");
        printf("\n\n   \xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\n");
        printf("   [1] View Bus Info\n\n");
        printf("   [2] Book Ticket\n\n");
        printf("   [3] Print Ticket\n\n");
        printf("   [4] Update Ticket Info\n\n");
        printf("   [5] Cancel Booking\n\n");
        printf("   [6] Booking History\n\n");
        printf("   [7] Password Recovery\n\n");
        printf("   [8] Sign out\n\n");
        printf("\n\n   \xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\n");
        printf("\n   ENTER YOUR CHOICE: ");
        scanf("%d",&selected_menu);
        system("cls");
        if (selected_menu >= 1 && selected_menu <= 8)
```

```
        {
            switch (selected_menu)
            {
            case 1:
                bus_destination();
                break;
            case 2:
                bus_destination();
                break;
            case 3:
                admin_print_ticket();
                break;
            case 4:
                admin_update();
                break;
            case 5:
                admin_cancel();
                break;
            case 6:
                admin_history();
                break;
            case 7:
                pass_recovery();
                break;
            case 8:
                user_list();
                break;
            }
        }
        else
        {
            invalid_choice();
        }
    }
}
```

The admin_menu function is designed to manage the bus ticket reservation system through a user-friendly interface for administrators. Here's a detailed explanation of how it works:

1. Continuous Menu Display:
   a) The function runs in an infinite loop to keep the menu available for the administrator until they decide to sign out or exit the system.
2. Screen Management:
   a) The screen is cleared at the beginning of each loop iteration to provide a fresh display for the menu options.
3. Menu Options:
   a) The menu presents eight options for the administrator to choose from:

i. View Bus Info: Allows the admin to view information about buses.
ii. Book Ticket: Enables the admin to book tickets for users.
iii. Print Ticket: Provides functionality to print a booked ticket.
iv. Update Ticket Info: Allows updating of existing ticket information.
v. Cancel Booking: Facilitates the cancellation of a ticket booking.
vi. Booking History: Displays the history of all bookings.
vii. Password Recovery: Offers a way to recover or reset passwords.
viii. Sign Out: Signs out the admin and exits the menu.

4. User Input Handling:
   a) The function prompts the admin to enter their choice and validates the input to ensure it is within the range of available options (1 to 8).
5. Action Execution:

   a) Based on the admin's choice, the corresponding function is called to perform the selected action. For example:
      i. Viewing bus information.
      ii. Booking a ticket.
      iii. Printing a ticket.
      iv. Updating ticket information.
      v. Cancelling a booking.
      vi. Viewing booking history.
      vii. Recovering passwords.
      viii. Signing out.
6. Invalid Input Management:
   a) If the admin enters an invalid choice (a number outside 1 to 8), the function calls another function to handle the invalid input, usually by displaying an error message and prompting the admin to try again.

The admin_menu function serves as a central control hub for administrators in the bus ticket reservation system. It continuously displays a menu of options, handles user input, and calls appropriate functions to perform various administrative tasks. This setup ensures that the admin can efficiently manage bookings, view information, and perform other necessary actions in a streamlined and user-friendly manner.

The “admin_print_ticket” Function

```
void admin_print_ticket()
{
    while (1)
    {
```

```
int x;
char a;
char b[]="Canceled";
char m[]="Booked";
system("cls");
printf("\n\n   \xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\n");
printf("     BUS TICKET RESERVATION SYSTEM");
printf("\n\n   \xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\n");
printf("\n\n   \xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\n");
printf("\n   ENTER TICKET SERIAL NO: ");
scanf("%d",&x);
if (x > history_count || x<1)
{
    system("cls");
    invalid_choice();
}else
{
    if ((strcmp(booking_historys[x-1].status,m))==0)
    {
        system("cls");
        printf("\n\n   \xcd\xcd\xcd\xcd\xcd\xcd\n");
        printf("\t\t\t\t\t\t\t\t           BUS TICKET
            RESERVATION SYSTEM\n");
        printf("\n\n   \xcd\xcd\xcd\xcd\xcd\xcd\n");
        printf("\n\n   \xcd\xcd\xcd\xcd\xcd\xcd\n");
        printf("\n\n   \xcd\xcd\xcd\xcd\xcd\xcd\n");
        printf("  |    %-10s    |    %-15s    |    %-
            25s    |    %-8s    |    %-8s    |    %-
            22s    |    %-12s    |    %-
            8s    |\n","Serial
            no","Destination","Bus","Time","Seat
            no","Passenger name","Phone no","Status");
        printf("   ------------------------------\n");
        printf("  |    %-10d    |    %-15s    |    %-
            25s    |    %-8s    |    %-8d    |    %-
            22s    |    %-12s    |    %-8s    |\n",
            booking_historys[x-1].serial,
            booking_historys[x-1].destination,
            booking_historys[x-1].bus_name,
            booking_historys[x-
            1].time,booking_historys[x-
            1].seat_no,booking_historys[x-
            1].name,booking_historys[x-
            1].phn_no,booking_historys[x-1].status);
        printf("   ------------------------------\n");
        printf("   Press y to Print Ticket, any other
            key to go to menu... ");
        fflush(stdin);
        scanf("%c",&a);
        if (a=='y')
        {
```

```
            system("cls");
            char status[]="Booked";
            system("cls");
            char e[]="Wishing you a smooth trip filled
                 with adventures and excitement.";
            char f[]="";
            printf("   ---------------------------\n");
            printf("  |  \t\t\t  Ticket: %-25s
                 \tSerial: %-
                 2d  |",booking_historys[x-
                 1].bus_name,booking_historys[x-
                 1].serial);
            printf("   ---------------------------\n");
            printf("  |  \tDestination: %-
                 15s\t\tDeparture time: %-8s    |\n",
                 booking_historys[x-1].destination,
                 booking_historys[x-1].time);
            fflush(stdin);
            printf("  |  \tName: %-22s\t\tPhn no: %-
                 15s     |\n", booking_historys[x-
                 1].name, booking_historys[x-
                 1].phn_no);
            printf("  |  \tSeat no: %-19d\t\tFare:
                 BDT %-15d   |\n", booking_historys[x-
                 1].seat_no, booking_historys[x-
                 1].fare);
            printf("  |    %-66s   |\n",f);
            printf("  |    %-66s   |\n",e);
            printf("   ---------------------------\n");
            fflush(stdin);
            printf("\n   Press any key to continue...
                 ");
            getchar();
            admin_menu();
            system("cls");
        }
        system("cls");
        admin_menu();
        break;
        break;
    }else
    {
        system("cls");
        printf("\n\n   \xcd\xcd\xcd\xcd\xcd\xcd\n");
        printf("        BUS TICKET RESERVATION
            SYSTEM");
        printf("\n\n   \xcd\xcd\xcd\xcd\xcd\xcd\n");;);
        printf("              Ticket Canceled\n\n");
        printf("\n\n   \xcd\xcd\xcd\xcd\xcd\xcd\n");
        printf("   Press any key to continue... ");
        getchar();
```

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

The admin_print_ticket function is part of the Bus Ticket Reservation System and is designed to allow an administrator to print tickets based on the ticket serial number. Here's an explanation of how it works:

1. Continuous Operation:
   a) The function operates within an infinite loop, continuously prompting the admin until a valid action is completed.
2. Screen Management:
   a) At the beginning of each loop iteration, the screen is cleared to provide a fresh display.
3. Ticket Serial Number Input:
   a) The admin is prompted to enter a ticket serial number.
   b) The entered serial number is validated to ensure it is within the range of existing ticket serial numbers. If the serial number is invalid, the screen is cleared, and an error message is displayed.
4. Ticket Status Check:
   a) The function checks if the ticket status is "Booked". If it is "Canceled", an appropriate message is displayed, and the admin is prompted to press a key to continue.
5. Ticket Details Display:
   a) If the ticket is booked, the function displays the ticket details, including:
      i. Serial number
      ii. Destination
      iii. Bus name
      iv. Departure time
      v. Seat number
      vi. Passenger name
      vii. Phone number
      viii. Status
6. Printing Option:
   a) The admin is prompted to press 'y' to print the ticket or any other key to return to the menu.
   b) If 'y' is pressed, the ticket is printed in a formatted manner, including a wish message for the passenger.
   c) After printing, the admin is returned to the admin menu.
7. Return to Admin Menu:
   a) The admin is given an option to continue with other tasks in the admin menu after printing or viewing the ticket.

The admin_print_ticket function allows the administrator to print and view detailed information about a specific ticket based on its serial number. The function handles user input validation, displays ticket details if the ticket is booked, and provides an option to print the ticket. If the ticket is canceled, it notifies the admin and returns to the menu. This ensures the administrator can efficiently manage and print tickets as needed.

The “admin_update” Function

```
void admin_update()
{
    while (1)
    {
        int x;
        int y;
        char a;
        char str[30], phnno[12];
        char m[]="Booked";
        system("cls");
        printf("\n\n   \xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\n");
        printf("     BUS TICKET RESERVATION SYSTEM");
        printf("\n\n   \xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\n");
        printf("\n\n   \xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\n");
        printf("\n   ENTER TICKET SERIAL NO: ");
        scanf("%d",&x);
        if (x > history_count || x<1)
        {
            system("cls");
            invalid_choice();
        }else
        {
            if ((strcmp(booking_historys[x-1].status, m))==0)
            {
                system("cls");
                printf("   \xcd\xcd\xcd\xcd\xcd\xcd\xcd\n");
                printf("\t\t\t\t\t\t\t\t           BUS TICKET
                                        RESERVATION SYSTEM\n");
                printf("   \xcd\xcd\xcd\xcd\xcd\xcd\xcd\n");
                printf("   \xcd\xcd\xcd\xcd\xcd\xcd\xcd\n");
                printf("   \xcd\xcd\xcd\xcd\xcd\xcd\xcd\n");
                printf("  |    %-10s    |    %-15s    |    %-
                    25s    |    %-8s    |    %-8s    |    %-
                    22s    |    %-12s    |    %-
                    8s    |\n","Serial
                    no","Destination","Bus","Time","Seat
                    no","Passenger name","Phone no","Status");
                printf("   -------------------------------\n");
                printf("  |    %-10d    |    %-15s    |    %-
                    25s    |    %-8s    |    %-8d    |    %-
                    22s    |    %-12s    |    %-
                    8s    |\n",booking_historys[x-
```

```
    1].serial,booking_historys[x-
    1].destination,booking_historys[x-
    1].bus_name,booking_historys[x-
    1].time,booking_historys[x-
    1].seat_no,booking_historys[x-
    1].name,booking_historys[x-
    1].phn_no,booking_historys[x-1].status);
printf("   ------------------------------\n");
printf("   Press y to continue updating, any
    other key to go to main menu... ");
fflush(stdin);
scanf("%c",&a);
if (a=='y')
{
    system("cls");
    printf("\n\n   \xcd\xcd\xcd\xcd\xcd\n");
    printf("     BUS TICKET RESERVATION
         SYSTEM");
    printf("\n   \xcd\xcd\xcd\xcd\xcd\n\n");
    printf("   \xcd\xcd\xcd\xcd\xcd\xcd\n\n");
    printf("   [1] Update Name\n\n");
    printf("   [2] Update Phone No.\n\n");
    printf("   \xcd\xcd\xcd\xcd\xcd\xcd\n");
    printf("\n   ENTER YOUR CHOICE: ");
    scanf("%d",&y);
    if (y >= 1 && y <= 2)
    {
        switch (y)
        {
            case 1:
                system("cls");
                printf("\n\n   \xcd\xcd\n");
                printf("\t     BUS TICKET
                   RESERVATION SYSTEM");
                printf("\n   \xcd\xcd\xcd\n");
                printf("   \xcd\xcd\xcd\xc\n");
                printf("\n ENTER YOUR NAME: ");
                fflush(stdin);
                gets(str);
                strcpy(booking_historys[x-
                   1].name, str);
                break;
            case 2:
                system("cls");
                printf("\n\n   \xcd\xcd\n");
                printf("\t     BUS TICKET
                        RESERVATION SYSTEM");
                printf("\n   \xcd\xcd\xcd\n");
                printf("   \xcd\xcd\xcd\n");
                printf("\n   ENTER YOUR PHONE
                   NO: ");
```

```
                                fflush(stdin);
                                gets(str);
                                strcpy(booking_historys[x-
                                   1].phn_no,str);
                                break;
                            }
                        }
                        system("cls");
                        history_file_write();
                        printf("\n\n  \xcd\xcd\xcd\xcd\xcd\xcd\n");
                        printf("        BUS TICKET RESERVATION
                                                    SYSTEM");
                        printf("\n  \xcd\xcd\xcd\xcd\xcd\xcd\n\n");
                        printf("            Updated
                             Successfully.\n\n");
                        printf("   \xcd\xcd\xcd\xcd\xcd\xcd\n");
                        printf("   Press any key to continue... ");
                        getchar();
                        history_file_write();
                        admin_menu();
                        system("cls");
                    }
                    system("cls");
                    admin_menu();
                    break;
                    break;
                }else
                {
                    system("cls");
                    printf("\n\n   \xcd\xcd\xcd\xcd\xcd\xcd\n");
                    printf("        BUS TICKET RESERVATION
                        SYSTEM");
                    printf("\n  \xcd\xcd\xcd\xcd\xcd\xcd\xcd\n\n");
                    printf("           Ticket canceled
                        already.\n\n");
                    printf("   \xcd\xcd\xcd\xcd\xcd\xcd\xcd\n");
                    printf("   Press any key to continue... ");
                    getchar();
                    getchar();
                    system("cls");
                }
            }
        }
    }
```

The admin_update function is part of a Bus Ticket Reservation System and allows an administrator to update specific details of a booked ticket based on the ticket's serial number. Below is a detailed explanation of how this function operates:

1. Continuous Operation:

a) The function operates within an infinite loop, allowing continuous updates until the admin chooses to return to the main menu.

2. Screen Management:
   a) At the beginning of each loop iteration, the screen is cleared to provide a fresh display.
3. Ticket Serial Number Input:
   a) The admin is prompted to enter a ticket serial number.
   b) The entered serial number is validated to ensure it falls within the range of existing ticket serial numbers. If the serial number is invalid, the screen is cleared, and an error message is displayed.
4. Ticket Status Check:
   a) The function checks if the ticket status is "Booked". If the status is "Canceled", an appropriate message is displayed, and the admin is prompted to press a key to continue.
5. Display Ticket Details:
   a) If the ticket is booked, the function displays the ticket details, including:
      i. Serial number
      ii. Destination
      iii. Bus name
      iv. Departure time
      v. Seat number
      vi. Passenger name
      vii. Phone number
      viii. Status
6. Update Confirmation:
   a) The admin is prompted to press 'y' to proceed with updating the ticket details or any other key to return to the main menu.
7. Update Options:
   a) If the admin chooses to update, the function provides two options:
      i. Update the passenger's name
      ii. Update the passenger's phone number
8. Update Execution:
   a) Depending on the admin's choice, the function prompts for the new name or phone number.
   b) The respective details in the booking history are updated accordingly.
9. Write Updated Details to File:
   a) The updated ticket details are written back to the history file to ensure data persistence.
10. Confirmation and Return to Menu:
   a) A success message is displayed upon updating, and the admin is prompted to press a key to continue.
   b) The admin is then returned to the main menu.

The admin_update function enables the administrator to update the passenger name or phone number for a booked ticket. It handles user input validation, displays current ticket details, allows specific updates, writes the updated information to a file, and confirms the update. If the ticket is canceled or an invalid serial number is entered, appropriate messages are displayed, and the admin is prompted to retry or return to the main menu.

The “admin_cancel” Function

```
void admin_cancel()
{
    while (1)
    {
        int x;
        char a;
        char b[]="Canceled";
        char m[]="Booked";
        system("cls");
        printf("\n\n   \xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\n");
        printf("     BUS TICKET RESERVATION SYSTEM");
        printf("\n   \xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\n\n");
        printf("   \xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\n\n");
        printf("\n   ENTER TICKET SERIAL NO: ");
        scanf("%d",&x);
        if (x > history_count || x<1)
        {
            system("cls");
            invalid_choice();
        }else
        {
            if ((strcmp(booking_historys[x-1].status,m))==0)
            {
                system("cls");
                printf("  \xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\n");
                printf("\t\t\t\t\t\t\t\t           BUS TICKET
                    RESERVATION SYSTEM\n");
                printf("  \xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\n");
                printf("  \xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\n");
                printf("  \xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\n");
                printf("  |    %-10s    |    %-15s    |    %-
                    25s    |    %-8s    |    %-8s    |    %-
                    22s    |    %-12s    |    %-
                    8s    |\n","Serial
                    no","Destination","Bus","Time","Seat
                    no","Passenger name","Phone no","Status");
                printf("   ------------------------------\n");
                printf("  |    %-10d    |    %-15s    |    %-
                    25s    |    %-8s    |    %-8d    |    %-
                    22s    |    %-12s    |    %-
                    8s    |\n",booking_historys[x-
                    1].serial,booking_historys[x-
```

```
            1].destination,booking_historys[x-
            1].bus_name,booking_historys[x-
            1].time,booking_historys[x-
            1].seat_no,booking_historys[x-
            1].name,booking_historys[x-
            1].phn_no,booking_historys[x-1].status);
        printf("   -------------------------------\n");
        printf("   Press y to confirm cancellation,
            any other key to go to menu... ");
        fflush(stdin);
        scanf("%c",&a);
        if (a=='y')
        {
            strcpy(booking_historys[x-1].status,b);
            system("cls");
            printf("\n\n  \xcd\xcd\xcd\xcd\xcd\xcd\n");
            printf("        BUS TICKET RESERVATION
                 SYSTEM");
            printf("\n   \xcd\xcd\xcd\xcd\xcd\xcdn\n");
            printf("          Ticket Successfully
                 Canceled.\n\n");
            printf("   \xcd\xcd\xcd\xcd\xcd\xcd\n");
            sit[(booking_historys[x-
                 1].selected_destination)-
                 1][(booking_historys[x-
                 1].selected_time)-  `
                 1][(booking_historys[x-
                 1].selected_bus)-
                 1][(booking_historys[x-
                 1].selected_seat)-1]=0;
            sit_file_write();
            printf("   Press any key to continue...");
            getchar();
            getchar();
            history_file_write();
            user_menu();
            system("cls");
        }
        system("cls");
        user_menu();
        break;
        break;
    }else
    {
        system("cls");
        printf("\n\n  \xcd\xcd\xcd\xcd\xcd\xcd\xcd\n");
        printf("        BUS TICKET RESERVATION
            SYSTEM");
        printf("\n  \xcd\xcd\xcd\xcd\xcd\xcd\xcd\n\n");
        printf("              Already Canceled.\n\n");
        printf("   \xcd\xcd\xcd\xcd\xcd\xcd\xcd\n");
```

```
                printf("   Press any key to continue... ");
                getchar();
                getchar();
                system("cls");
            }
        }
    }
}
```

The admin_cancel function is designed to allow an administrator to cancel a booked bus ticket in the Bus Ticket Reservation System. Here's a detailed explanation of how the function works:

1. Continuous Operation:
   a) The function runs in an infinite loop, allowing continuous cancellation operations until the administrator decides to return to the main menu.
2. Screen Management:
   a) The screen is cleared at the beginning of each iteration to provide a clean display for the admin.
3. Ticket Serial Number Input:
   a) The admin is prompted to enter the ticket serial number.
   b) The input is validated to ensure it falls within the range of existing ticket serial numbers. If the serial number is invalid, the screen is cleared, and an error message is displayed.
4. Ticket Status Check:
   a) The function checks if the ticket status is "Booked". If the ticket status is "Canceled", an appropriate message is displayed, and the admin is prompted to press a key to continue.
5. Display Ticket Details:
   a) If the ticket is booked, the function displays the ticket details, including:
      i. Serial number
      ii. Destination
      iii. Bus name
      iv. Departure time
      v. Seat number
      vi. Passenger name
      vii. Phone number
      viii. Status
6. Cancellation Confirmation:
   a) The admin is prompted to press 'y' to confirm the cancellation or any other key to return to the main menu.
7. Update Ticket Status:
   a) If the admin confirms the cancellation, the ticket status is updated to "Canceled".
   b) The system reflects this change by updating the seat availability and writing the changes back to the appropriate files.

8. Confirmation and Return to Menu:
   a) A success message is displayed, confirming that the ticket has been canceled.
   b) The admin is prompted to press a key to continue, after which the admin is returned to the user menu.

The admin_cancel function allows an administrator to cancel a booked ticket by verifying the ticket's serial number and status. If the ticket is already booked, it displays the details, asks for confirmation, and upon confirmation, updates the ticket status to "Canceled". If the ticket is already canceled or the serial number is invalid, appropriate messages are displayed. The function ensures that the seat availability is updated and written back to the relevant files, maintaining the integrity of the reservation system.

The “admin_history” Function

```
void admin_history()
{
    if (history_count==0)
    {
        printf("\n\n   \xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\n");
        printf("        BUS TICKET RESERVATION SYSTEM");
        printf("\n  \xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\n\n");
        printf("              No History Found!\n\n");
        printf("  \xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\n");
        printf("   Press any key to continue... ");
        getchar();
        getchar();
        system("cls");
        user_menu();
    }
    system("cls");
    printf("\n\n  \xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\n");
    printf("\t\t\t\t\t\t\t\t\t\t         BUS TICKET
          RESERVATION SYSTEM\n");
    printf("  \xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\n");
    printf("  \xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\n");
    printf("  \xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\n");
    printf("  |    %-10s    |    %-10s    |    %-
          15s    |    %-25s    |    %-8s    |    %-
          8s    |    %-22s    |    %-12s    |    %-
          8s    |\n","Serial no","User
          id","Destination","Bus","Time","Seat no","Passenger
          name","Phone no","Status");
    printf("   ----------------------------------------\n");
    for (int i = 0; i < history_count; i++)
    {
           printf("  |    %-10d    |    %-10s    |    %-
          15s    |    %-25s    |    %-8s    |    %-
          8d    |    %-22s    |    %-12s    |    %-
     8s    |\n",booking_historys[i].serial,sign_ups[booking_hi
```

```
     storys[i].user_id].username,booking_historys[i].destinati
     on,booking_historys[i].bus_name,booking_historys[i].time,
     booking_historys[i].seat_no,booking_historys[i].name,book
     ing_historys[i].phn_no,booking_historys[i].status);
            printf("   ---------------------------------\n");
    }
        printf("   Press any key to continue... ");
        getchar();
        getchar();
        system("cls");
}
```

The admin_history function is designed to allow an administrator to view the history of all bus ticket reservations. Here's a detailed explanation of its functionality:

1. Check for Empty History:
   a) The function first checks if the history count is zero, indicating that there are no ticket reservations in the history.
   b) If the history is empty, it prints a message stating "No History Found!" and prompts the user to press any key to continue. After the key press, it clears the screen and calls the user_menu function.
2. Clear Screen and Display Header:
   a) If there is history, the screen is cleared.
   b) A header is displayed with the title "BUS TICKET RESERVATION SYSTEM".
3. Display Table Header:
   a) The function prints a formatted header for the history table, showing columns for serial number, user ID, destination, bus, time, seat number, passenger name, phone number, and status.
4. Loop through History Records:
   a) The function uses a for-loop to iterate over all the records in the booking history.
   b) For each record, it prints the details in a formatted row. The details include:
      i. Serial number
      ii. User ID
      iii. Destination
      iv. Bus name
      v. Departure time
      vi. Seat number
      vii. Passenger name
      viii. Phone number
      ix. Status
5. Prompt to Continue:
      i. After displaying all records, the function prompts the user to press any key to continue.
      ii. Upon key press, it clears the screen.

This function is a part of a bus ticket reservation system, designed to be used by an administrator to monitor and manage booking histories. It provides a comprehensive view of all past reservations, including crucial details for each booking. This feature is essential for administrative tasks such as reviewing bookings, auditing, and managing customer service issues related to ticket reservations.

The “pass_recovery” Function

```
void pass_recovery()
{
    char a[10];
    bool flag=1;
    while (1)
    {
        system("cls");
        printf("\n\n   \xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\n");
        printf("\t    BUS TICKET RESERVATION SYSTEM");
        printf("\n   \xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\n\n");
        printf("   \xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd");
        fflush(stdin);
        printf("\n\n   ENTER USERNAME: ");
        fflush(stdin);
        scanf("%s", a);
        for (int i = 0; i < user; i++)
        {
            if ((strcmp(sign_ups[i].username,a)==0))
            {
                flag=0;
                printf("\n\n   PASSWORD
                    IS: %s",sign_ups[i].password);
                break;
            }
        }
        if (flag)
        {
            printf("\n\n   USERNAME NOT FOUND");
        }
        fflush(stdin);
        printf("\n\n   Press any key to continue... ");
        getchar();
        system("cls");
        admin_menu();
    }
}
```

The pass_recovery function is a crucial part of the bus ticket reservation system, providing users with a way to retrieve their password if they forget it. Here's a breakdown of its

1. Initialization:

   a) The function initializes a character array a to store the username input.
   b) It sets a boolean flag flag to 1, indicating that the username has not been found yet.
2. Main Loop:
   a) The function operates within an infinite loop, ensuring continuous interaction until the user successfully recovers their password.
   b) At the start of each iteration, the screen is cleared for a clean interface.
   c) The header "BUS TICKET RESERVATION SYSTEM" is displayed to maintain consistency and provide context.
3. Username Search:
   a) Users are prompted to enter their username.
   b) The function scans the entered username and compares it with the usernames stored in the sign_ups array.
   c) If a match is found, the flag is set to 0, indicating that the username exists, and the corresponding password is displayed.
   d) If no match is found after checking all usernames, the flag remains 1, indicating that the username was not found.
4. Handling Username Not Found:
   a) If the flag is still 1 after checking all usernames, the function informs the user that the entered username was not found by displaying a "USERNAME NOT FOUND" message.
5. User Prompt:
   a) After the search process, the user is prompted to press any key to continue.
   b) Upon keypress, the screen is cleared, and control is returned to the admin menu by calling the admin_menu function.

By providing users with a straightforward way to recover their password, the pass_recovery function enhances the user experience and ensures smooth operation of the bus ticket reservation system.

The “user_login_signup_menu” Function

```
void user_login_signup_menu()
{
    int choice;
    while (1)
    {
        system("cls");
        printf("\n\n   \xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\n");
        printf("     BUS TICKET RESERVATION SYSTEM");
        printf("\n   \xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\n\n");
        printf("   \xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd  User
Zone  \xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\n\n");
```

```
        printf("   [1] Sign up\n\n");
        printf("   [2] Log in\n\n");
        printf("  \xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\n");
        printf("\n   ENTER YOUR CHOICE: ");
        scanf("%d",&choice);
        switch (choice)
        {
        case 1:
            user_signup();
            break;
        case 2:
            login_user();
            break;
        default:
            system("cls");
            invalid_choice();
        }
        system("cls");
    }
}
```

The user_login_signup_menu function provides users with options to either sign up for a new account or log in to an existing account within the bus ticket reservation system. Here's an overview of its functionality:

1. Initialization:
   a) The function initializes an integer variable choice to store the user's menu selection.
2. Main Loop:
   a) The function operates within an infinite loop, ensuring continuous interaction with the user until an appropriate choice is made.
   b) At the beginning of each iteration, the screen is cleared for a clean interface.
   c) The header "BUS TICKET RESERVATION SYSTEM" is displayed, providing branding and context for the user.
3. User Zone Display:
   a) The function displays a menu specifically for user-related actions under the section "User Zone."
   b) Two options are presented:
      i. Option [1]: Sign up - Allows new users to create an account.
      ii. Option [2]: Log in - Enables existing users to access their accounts.
4. User Input and Processing:
   a) The user is prompted to enter their choice by selecting an option.
   b) The function reads the user's choice using scanf.
5. Menu Navigation:
   a) Based on the user's selection, the function directs control to the appropriate action:

```
      i. If the user chooses to sign up (option 1), the
         user_signup function is called to handle the signup
         process.
      ii. If the user chooses to log in (option 2), the
         login_user function is invoked to facilitate the
         login procedure.
      iii. If the user enters an invalid choice, the screen is
         cleared, and an "Invalid Choice" message is displayed.
6. Loop Continuation:
   a) After processing the user's choice, the screen is
      cleared, and the loop continues, allowing the user to
      perform further actions or make another selection.
```

Overall, the user_login_signup_menu function serves as the entry point for users to either create a new account or access their existing account, providing a streamlined interface for user authentication and management within the bus ticket reservation system.

The "user_Exists" Function

```
bool user_Exists(const char *username) {
    bool flag=0;
    for (int i = 0; i < user; i++)
    {
        if (strcmp(sign_ups[i].username,username)==0)
        {
            flag=1;
            break;
        }
    }
    if (flag)
    {
        return true;
    }else
    {
        return false;
    }
}
```

The user_Exists function checks whether a user with a given username already exists in the system. Here's an overview of its functionality:

```
1. Input:
   a) The function takes a pointer to a character array (const
      char *username) as input, representing the username to
      be checked.
2. Initialization:
   a) A boolean variable flag is initialized to false (0).
      This flag will be used to indicate whether the username
      exists or not.
3. Username Check:
```

   a) The function iterates through the sign_ups array, which presumably contains user information.
   b) For each user in the array, it compares the username with the provided username using the strcmp function.
   c) If a match is found, indicating that the username already exists, the flag is set to true (1), and the loop is terminated using break.
4. Return Statement:
   a) After checking all usernames, the function checks the value of the flag.
   b) If flag is true, indicating that the username exists, the function returns true.
   c) If flag is false, indicating that the username does not exist, the function returns false.

Overall, the user_Exists function efficiently determines whether a given username already exists in the system's user database, providing a valuable utility for user management and authentication processes.

The "user_signup" Function

```
void user_signup() {
    char username[10];
    char password[10];
    char confirm[10];
    while (1)
    {
        system("cls");
        printf("\n\n   \xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\n");
        printf("\t    BUS TICKET RESERVATION SYSTEM");
        printf("\n   \xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\n\n");
        printf("   \xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd");
        printf("\n\n   ENTER USERNAME: ");
        fflush(stdin);
        fgets(username, 10, stdin);
        username[strcspn(username, "\n")] = '\0';
        if (user_Exists(username)) {
            system("cls");
            printf("\n\n  \xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\n");
            printf("     BUS TICKET RESERVATION SYSTEM");
            printf("\n  \xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\n\n");
            printf("   \xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\n\n");
            printf("          Username already exists.\n");
            printf("     Please choose a different
               username.\n\n");
            printf("   \xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\n");
            printf("   Press any key to continue... ");
            getchar();
            system("cls");
```

```
    user_signup();
    return;
}
strcpy(new_sign_up.username,username);
printf("   ENTER PASSWORD: ");
int i = 0;
char ch;
while (1) {
    fflush(stdin);
    ch = getch();
    if (ch == '\r') {
        password[i] = '\0';
        break;
    } else if (ch == 8 && i > 0) {
        printf("\b \b");
        i--;
    } else if (ch >= 32 && ch <= 126) {
        printf("*");
        password[i] = ch;
        i++;
    }
}
printf("\n");

printf("   ENTER CONFIRM PASSWORD: ");
i = 0;
while (1) {
    fflush(stdin);
    ch = getch();
    if (ch == '\r') {
        confirm[i] = '\0';
        break;
    } else if (ch == 8 && i > 0) {
        printf("\b \b");
        i--;
    } else if (ch >= 32 && ch <= 126) {
        printf("*");
        confirm[i] = ch;
        i++;
    }
}
printf("\n");

if (strcmp(password, confirm) != 0) {
    system("cls");
    printf("\n\n   \xcd\xcd\xcd\xcd\xcd\xcd\xcd\n");
    printf("     BUS TICKET RESERVATION SYSTEM");
    printf("\n  \xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\n\n");
    printf("   \xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\n\n");
    printf("           Password doesn't match\n\n");
    printf("   \xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\n");
```

```
            printf("   Press any key to continue...");
            getchar();
            system("cls");
        }else
        {
            strcpy(new_sign_up.password,confirm);
            new_sign_up.serial=user;
            sign_ups[user]=new_sign_up;
            user_file_write();
            user++;
            break;
        }
    }
    system("cls");
    printf("\n\n   \xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\n");
    printf("     BUS TICKET RESERVATION SYSTEM");
    printf("\n   \xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\n\n");
    printf("   \xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\n\n");
    printf("              Signup Successful\n\n");
    printf("   \xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\n");
    printf("   Press any key to continue... ");
    getchar();
    system("cls");
    user_login_signup_menu();
}
```

The user_signup function allows users to sign up for the bus ticket reservation system by providing a username and password. Here's an overview of its functionality:

1. Input Handling:
   a) The function initializes character arrays username, password, and confirm to store user input.
   b) It enters a loop to handle the signup process.
2. Username Input:
   a) The function prompts the user to enter a username.
   b) It ensures that the entered username does not already exist in the system by calling the user_Exists function.
3. Password Input:
   a) After successfully entering a unique username, the function prompts the user to enter a password.
   b) It conceals the password input by displaying asterisks instead of characters.
   c) The entered password is stored in the password array.
4. Password Confirmation:
   a) The function asks the user to confirm the password by entering it again.
   b) Similar to the password input, the confirmation process conceals the input and stores it in the confirm array.
5. Validation:

   a) It compares the entered password with the confirmation to ensure they match.
   b) If the passwords do not match, it displays an error message and prompts the user to try again.
6. Signup Process:
   a) If the passwords match, the function stores the username and password in the new_sign_up structure.
   b) It assigns a serial number to the new signup based on the current number of users.
   c) The signup information is then added to the sign_ups array.
   d) The user_file_write function is called to update the user data in the file.
   e) The function increments the total number of users.
7. Success Message:
   a) Upon successful signup, the function displays a confirmation message.
   b) It prompts the user to press any key to continue.
8. Navigation:
   a) After signup, the function clears the screen and returns to the user login/signup menu.

Overall, the user_signup function provides a user-friendly interface for users to create an account in the bus ticket reservation system and handles the necessary validation steps to ensure data integrity.

The "login_user" Function

```
void login_user()
{
    char username[10];
    char password[10];
    char confirm[10];
    bool flag=0;
    while (1)
    {
        system("cls");
        printf("\n\n   \xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\n");
        printf("\t    BUS TICKET RESERVATION SYSTEM");
        printf("\n   \xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\n\n");
        printf("   \xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd");
        printf("\n\n   ENTER USERNAME: ");
        fflush(stdin);
        fgets(username, 10, stdin);
        username[strcspn(username, "\n")] = '\0';
        printf("   ENTER PASSWORD: ");
        int i = 0;
        char ch;
        while (1) {
            fflush(stdin);
```

```
    ch = getch();
    if (ch == '\r') {
        password[i] = '\0';
        break;
    } else if (ch == 8 && i > 0) {
        printf("\b \b");
        i--;
    } else if (ch >= 32 && ch <= 126) {
        printf("*");
        password[i] = ch;
        i++;
    }
}
printf("\n");

for (int i = 0; i < user; i++)
{
    if ((strcmp(username,sign_ups[i].username) == 0)
  && (strcmp(password,sign_ups[i].password) ==0)) {
        system("cls");
        printf("\n\n  \xcd\xcd\xcd\xcd\xcd\xcd\xcd\n");
        printf("     BUS TICKET RESERVATION SYSTEM");
        printf("\n  \xcd\xcd\xcd\xcd\xcd\xcd\xcd\n\n");
        printf("   \xcd\xcd\xcd\xcd\xcd\xcd\xcd\n\n");
        printf("              Login Successful\n\n");
        printf("   \xcd\xcd\xcd\xcd\xcd\xcd\xcd\n");
        temp=i;
        printf("   Press any key to continue... ");
        getchar();
        system("cls");
        flag=1;
        user_menu();
        break;
    }
}
if (flag)
{
    break;
}else
{
    system("cls");
    printf("\n\n   \xcd\xcd\xcd\xcd\xcd\xcd\xcd\n");
    printf("        BUS TICKET RESERVATION SYSTEM");
    printf("\n   \xcd\xcd\xcd\xcd\xcd\xcd\xcd\n\n");
    printf("   \xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcdn\n");
    printf("         Wrong Username or Password\n\n");
    printf("   \xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\n");
    // printf("%s%s",username,password);
    printf("   Press any key to continue... ");
    getchar();
    system("cls");
```

```
        }
    }
}
```

The login_user function handles the user login process for the bus ticket reservation system. Here's a breakdown of its functionality:

1. Input Handling:
   a) The function initializes character arrays username and password to store user input.
   b) It clears the screen and displays the system header.
   c) The user is prompted to enter their username and password.
2. Password Input:
   a) The password input is concealed by displaying asterisks instead of characters.
   b) The entered password is stored in the password array.
3. Login Validation:
   a) The function iterates through the existing user accounts stored in the sign_ups array.
   b) It checks if the entered username and password match any user account in the system.
   c) If a match is found, the login is considered successful, and the user is redirected to the user menu.
   d) If no match is found, the function displays an error message indicating incorrect username or password.
4. Success Message:
   a) If the login is successful, the function displays a confirmation message.
   b) It prompts the user to press any key to continue.
5. Loop Continuation:
   a) If the login is unsuccessful, the function clears the screen and prompts the user to try again.
   b) The process continues until a successful login is achieved.

Overall, the login_user function provides a user-friendly interface for users to log in to the bus ticket reservation system, ensuring the security of user accounts through password concealment and validation checks.

The “user_menu” Function

```
void user_menu()
{
    while (1)
    {
        system("cls");
        printf("\n\n   \xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\n");
        printf("     BUS TICKET RESERVATION SYSTEM");
        printf("\n   \xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\n\n");
```

```
        printf("   \xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\n\n");
        printf("   [1] View Bus Info\n\n");
        printf("   [2] Book Ticket\n\n");
        printf("   [3] Print Ticket\n\n");
        printf("   [4] Update Ticket Info\n\n");
        printf("   [5] Cancel Booking\n\n");
        printf("   [6] Booking History\n\n");
        printf("   [7] Sign out\n\n");
        printf("   \xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\n");
        printf("\n   ENTER YOUR CHOICE: ");
        scanf("%d",&selected_menu);
        system("cls");
        if (selected_menu >= 1 && selected_menu <= 7)
        {
            switch (selected_menu)
            {
            case 1:
                bus_destination();
                break;
            case 2:
                bus_destination();
                break;
            case 3:
                print_ticket();
                break;
            case 4:
                update();
                break;
            case 5:
                cancel();
                break;
            case 6:
                history();
                break;
            case 7:
                user_list();
                break;
            }
        }
        else
        {
            invalid_choice();
        }
    }
}
```

The user_menu function is the main menu for registered users in the bus ticket reservation system. Here's a breakdown of its functionality:

1. Menu Display:

   a) Clears the screen and displays the header for the bus ticket reservation system.
   b) Lists various options for the user to choose from, including viewing bus information, booking tickets, printing tickets, updating ticket information, canceling bookings, viewing booking history, and signing out.
2. User Input:
   a) Prompts the user to enter their choice by selecting a number corresponding to the menu options.
3. Input Validation and Menu Navigation:
   a) Checks if the entered choice is within the valid range of menu options (1 to 7).
   b) If the choice is valid, the function executes the corresponding action based on the selected menu option.
   c) If the choice is invalid, it displays an error message indicating an invalid choice.
4. Menu Actions:
   a) The function uses a switch-case statement to execute the appropriate action based on the user's choice.
   b) Each case corresponds to a specific action:
      i. Case 1 and Case 2: View Bus Info and Book Ticket both lead to the bus_destination function.
      ii. Case 3: Print Ticket leads to the print_ticket function.
      iii. Case 4: Update Ticket Info leads to the update function.
      iv. Case 5: Cancel Booking leads to the cancel function.
      v. Case 6: Booking History leads to the history function.
      vi. Case 7: Sign out leads to the user_list function, presumably for logging out the user.
5. Loop Continuation:
   a) The function runs in an infinite loop until the user chooses to sign out, ensuring that the user can continuously interact with the system's functionalities.

Overall, the user_menu function provides a user-friendly interface for registered users to navigate through different options and perform various actions within the bus ticket reservation system.

The “print_ticket” function

```
void print_ticket()
{
    while (1)
    {
        int x;
        char a;
        char b[]="Canceled";
        char m[]="Booked";
```

```
        system("cls");
        printf("\n\n   \xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\n");
        printf("     BUS TICKET RESERVATION SYSTEM");
        printf("\n   \xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\n\n");
        printf("   \xcd\xcd\xcd\xcd\xcd\xcd\xcd\n\n");
        printf("\n   ENTER TICKET SERIAL NO: ");
        scanf("%d",&x);
        if (x > history_count || x<1)
        {
            system("cls");
            invalid_choice();
        }else
        {
            if (temp==booking_historys[x-1].user_id &&
               (strcmp(booking_historys[x-1].status,m))==0)
            {
                system("cls");
                printf("\n\n   \xcd\xcd\xcd\xcd\xcd\xcd\n");
                printf("\t\t\t\t\t\t\t\t      BUS TICKET
RESERVATION SYSTEM\n");
                printf("   \xcd\xcd\xcd\xcd\xcd\xcd\xcd\n");
                printf("   \xcd\xcd\xcd\xcd\xcd\xcd\xcd\n");
                printf("   \xcd\xcd\xcd\xcd\xcd\xcd\xcd\n");
                printf("  |    %-10s    |    %-15s    |    %-
                    25s    |    %-8s    |    %-8s    |    %-
                    22s    |    %-12s    |    %-
                    8s    |\n","Serial
                    no","Destination","Bus","Time","Seat
                    no","Passenger name","Phone no","Status");
                printf("   ------------------------------\n");
                printf("  |    %-10d    |    %-15s    |    %-
                    25s    |    %-8s    |    %-8d    |    %-
                    22s    |    %-12s    |    %-
                    8s    |\n",booking_historys[x-
                    1].serial,booking_historys[x-
                    1].destination,booking_historys[x-
                    1].bus_name,booking_historys[x-
                    1].time,booking_historys[x-
                    1].seat_no,booking_historys[x-
                    1].name,booking_historys[x-
                    1].phn_no,booking_historys[x-1].status);
                printf("   ----------------------------\n");
                printf("   Press y to Print Ticket, any other
                    key to go to menu... ");
                fflush(stdin);
                scanf("%c",&a);
                if (a=='y')
                {
                    system("cls");
                    char status[]="Booked";
                    system("cls");
```

```
            char e[]="Wishing you a smooth trip filled
                 with adventures and excitement.";
            char f[]="";
            printf("\n\n   \xcd\xcd\xcd\xcd\xcd\n");
            printf("  |  \t\t\t  Ticket: %-25s
                 \tSerial: %-
                 2d  |",booking_historys[x-
                 1].bus_name,booking_historys[x-
                 1].serial);
            printf("\n   \xcd\xcd\xcd\xcd\xcd\n");
            printf("  |  \tDestination: %-
                 15s\t\tDeparture time: %-
                 8s    |\n",booking_historys[x-
                 1].destination,booking_historys[x-
                 1].time);
            fflush(stdin);
            printf("  |  \tName: %-22s\t\tPhn no: %-
                 15s     |\n",booking_historys[x-
                 1].name,booking_historys[x-1].phn_no);
            printf("  |  \tSeat no: %-19d\t\tFare:
                 BDT %-15d   |\n",booking_historys[x-
                 1].seat_no,booking_historys[x-
                 1].fare);
            printf("  |    %-66s   |\n",f);
            printf("  |    %-66s   |\n",e);
            printf("   \xcd\xcd\xcd\xcd\xcd\xcd\n");
            fflush(stdin);
            printf("\n Press any key to continue... ");
            getchar();
            user_menu();
            system("cls");
        }
        system("cls");
        user_menu();
        break;
        break;
    }else
    {
        system("cls");
        printf("\n\n  \xcd\xcd\xcd\xcd\xcd\xcd\xcd\n");
        printf("        BUS TICKET RESERVATION
            SYSTEM");
        printf("\n  \xcd\xcd\xcd\xcd\xcd\xcd\xcd\n\n");
        printf("              Canceled or No
            access.\n\n");
        printf("   \xcd\xcd\xcd\xcd\xcd\xcd\xcd\n");
        printf("   Press any key to continue... ");
        getchar();
        getchar();
        system("cls");
    }
```

```
        }
    }
}
```

The print_ticket function allows users to print their booked tickets by entering the serial number of the ticket they want to print. Here's how it works:

1. Initialization:
   a) Initializes variables x and a.
   b) Defines strings b and m for representing "Canceled" and "Booked" statuses, respectively.
2. Menu Display and Input:
   a) Clears the screen and displays the header for the bus ticket reservation system.
   b) Prompts the user to enter the ticket serial number.
3. Validation:
   a) Checks if the entered serial number is valid (within the range of available tickets).
   b) If the serial number is invalid, displays an error message and returns to the main menu.
4. Ticket Status Check:
   a) Verifies if the ticket corresponding to the entered serial number is booked by the current user.
   b) If the ticket is booked by the current user, displays the ticket details and prompts the user to confirm printing by pressing 'y'.
   c) If the ticket is not booked by the current user or is canceled, displays an appropriate message and returns to the main menu.
5. Printing Ticket:
   a) If the user confirms printing by pressing 'y', displays the ticket details in a formatted manner.
   b) Displays a message wishing the user a pleasant trip.
   c) Waits for user input to continue and returns to the main menu.
6. Loop Continuation:
   a) The function runs in an infinite loop until the user successfully prints a ticket or returns to the main menu after encountering an error or cancelation.

Overall, the print_ticket function provides users with the ability to view and print their booked tickets, enhancing their experience within the bus ticket reservation system.

The “update” Function

```
void update()
{
    while (1)
    {
        int x;
```

```
int y;
char a;
char str[30],phnno[12];
char m[]="Booked";
system("cls");
printf("\n\n   \xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\n");
printf("     BUS TICKET RESERVATION SYSTEM");
printf("\n   \xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\n\n");
printf("   \xcd\xcd\xcd\xcd\xcd\xcd\xcd\n\n");
printf("\n   ENTER TICKET SERIAL NO: ");
scanf("%d",&x);
if (x > history_count || x<1)
{
    system("cls");
    invalid_choice();
}else
{
    if ((strcmp(booking_historys[x-1].status,m))==0)
    {
        system("cls");
        printf("\n\n   \xcd\xcd\xcd\xcd\xcd\xcd\n");
        printf("\t\t\t\t\t\t\t\t      BUS TICKET
            RESERVATION SYSTEM\n");
        printf("   \xcd\xcd\xcd\xcd\xcd\xcd\xcd\n");
        printf("   \xcd\xcd\xcd\xcd\xcd\xcd\xcd\n");
        printf("   \xcd\xcd\xcd\xcd\xcd\xcd\xcd\n");
        printf("  |    %-10s    |    %-15s    |    %-
            25s    |    %-8s    |    %-8s    |    %-
            22s    |    %-12s    |    %-
            8s    |\n","Serial
            no","Destination","Bus","Time","Seat
            no","Passenger name","Phone no","Status");
        printf("   ------------------------------\n");
        printf("  |    %-10d    |    %-15s    |    %-
            25s    |    %-8s    |    %-8d    |    %-
            22s    |    %-12s    |    %-
            8s    |\n",booking_historys[x-
            1].serial,booking_historys[x-
            1].destination,booking_historys[x-
            1].bus_name,booking_historys[x-
            1].time,booking_historys[x-
            1].seat_no,booking_historys[x-
            1].name,booking_historys[x-
            1].phn_no,booking_historys[x-1].status);
        printf("   ------------------------------\n");
        printf("   Press y to continue updating, any
            other key to go to main menu... ");
        fflush(stdin);
        scanf("%c",&a);
        if (a=='y')
        {
```

```
system("cls");
printf("\n\n   \xcd\xcd\xcd\xcd\xcd\n");
printf("     BUS TICKET RESERVATION
     SYSTEM");
printf("\n   \xcd\xcd\xcd\xcd\xcd\n\n");
printf("   \xcd\xcd\xcd\xcd\xcd\xcd\n\n");
printf("   [1] Update Name\n\n");
printf("   [2] Update Phone no\n\n");
printf("   \xcd\xcd\xcd\xcd\xcd\n");
printf("\n   ENTER YOUR CHOICE: ");
scanf("%d",&y);
if (y >= 1 && y <= 2)
{
    switch (y)
    {
        case 1:
            system("cls");
            printf("\n\n  \xcd\xcd\xcd\n");
            printf("\t     BUS TICKET
               RESERVATION SYSTEM");
            printf("\n\xcd\xcd\xcd\xcd\n");
            printf("  \xcd\xcd\xcd\xcd\n");
            printf("\n ENTER YOUR NAME: ");
            fflush(stdin);
            gets(str);
            strcpy(booking_historys[x-
               1].name,str);
            break;
        case 2:
            system("cls");
            printf("\n\n  \xcd\xcd\xcd\n");
            printf("\t     BUS TICKET
               RESERVATION SYSTEM");
            printf("\n   \xcd\xcd\xcd\n");
            printf("  \xcd\xcd\xcd\xcd\n");
            printf("\n   ENTER YOUR PHONE
               NO: ");
            fflush(stdin);
            gets(str);
            strcpy(booking_historys[x-
               1].phn_no,str);
            break;
    }
}
system("cls");
history_file_write();
printf("\n\n  \xcd\xcd\xcd\xcd\xcd\xcd\n");
printf("        BUS TICKET RESERVATION
     SYSTEM");
printf("\n   \xcd\xcd\xcd\xcd\xcd\xcd\n");
```

```
                    printf("          Updated
                         Successfully.\n\n");
                    printf("  \xcd\xcd\xcd\xcd\xcd\xcd\xcd\n");
                    printf("   Press any key to continue... ");
                    getchar();
                    history_file_write();
                    user_menu();
                    system("cls");
                }
                system("cls");
                user_menu();
                break;
                break;
            }else
            {
                system("cls");
                printf("\n\n  \xcd\xcd\xcd\xcd\xcd\xcd\xcd\n");
                printf("        BUS TICKET RESERVATION
                    SYSTEM");
                printf("\n   \xcd\xcd\xcd\xcd\xcd\xcd\xcd\n");
                printf("              Ticket already canceled
                    or no access\n\n");
                printf("   \xcd\xcd\xcd\xcd\xcd\xcd\xcd\n");
                printf("   Press any key to continue... ");
                getchar();
                getchar();
                system("cls");
            }
        }
    }
}
```

The update function allows users to update their booked ticket information such as name and phone number. Here's how it works:

1. Initialization:
   a) Initializes variables x, y, and a.
   b) Defines string arrays str and phnno for storing user input.
   c) Defines string m to represent the "Booked" status.
2. Menu Display and Input:
   a) Clears the screen and displays the header for the bus ticket reservation system.
   b) Prompts the user to enter the ticket serial number.
3. Validation:
   a) Checks if the entered serial number is valid (within the range of available tickets).
   b) If the serial number is invalid, displays an error message and returns to the main menu.
4. Ticket Status Check:

   a) Verifies if the ticket corresponding to the entered serial number is currently booked.
   b) If the ticket is booked, prompts the user to confirm updating by pressing 'y'.
   c) If the ticket is not booked, displays an appropriate message and returns to the main menu.
5. Updating Ticket Information:
   a) If the user confirms updating by pressing 'y', displays a sub-menu with options to update name or phone number.
   b) Prompts the user to choose an option and performs the corresponding update.
   c) Updates the ticket information in the booking_historys array.
   d) Writes the updated information to the history file.
   e) Displays a success message and waits for user input to continue.
   f) Returns to the main menu.
6. Loop Continuation:
   a) The function runs in an infinite loop until the user successfully updates a ticket or returns to the main menu after encountering an error or cancelation.

Overall, the update function provides users with the ability to modify their booked ticket information, enhancing their control and flexibility within the bus ticket reservation system.

The “cancel” Function

```
void cancel()
{
    while (1)
    {
        int x;
        char a;
        char b[]="Canceled";
        char m[]="Booked";
        system("cls");
        printf("\n\n  \xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\n");
        printf("     BUS TICKET RESERVATION SYSTEM");
        printf("\n   \xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\n\n");
        printf("\n   ENTER TICKET SERIAL NO: ");
        scanf("%d",&x);
        if (x > history_count || x<1)
        {
            system("cls");
            invalid_choice();
        }else
        {
            if (temp==booking_historys[x-1].user_id &&
               (strcmp(booking_historys[x-1].status,m))==0)
            {
```

```
system("cls");
printf("\n\n  \xcd\xcd\xcd\xcd\xcd\xcd\xcd\n");
printf("\t\t\t\t\t\t\t\t      BUS TICKET
    RESERVATION SYSTEM\n");
printf("  \xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\n");
printf("  \xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\n");
printf("  \xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\n");
printf("  |    %-10s    |    %-15s    |    %-
    25s    |    %-8s    |    %-8s    |    %-
    22s    |    %-12s    |    %-
    8s    |\n","Serial
    no","Destination","Bus","Time","Seat
    no","Passenger name","Phone no","Status");
printf("   ------------------------------\n");
printf("  |    %-10d    |    %-15s    |    %-
    25s    |    %-8s    |    %-8d    |    %-
    22s    |    %-12s    |    %-
    8s    |\n",booking_historys[x-
    1].serial,booking_historys[x-
    1].destination,booking_historys[x-
    1].bus_name,booking_historys[x-
    1].time,booking_historys[x-
    1].seat_no,booking_historys[x-
    1].name,booking_historys[x-
    1].phn_no,booking_historys[x-1].status);
printf("   ------------------------------\n");
printf("   Press y to confirm cancellation,
    any other key to go to menu... ");
fflush(stdin);
scanf("%c",&a);
if (a=='y')
{
    strcpy(booking_historys[x-1].status,b);
    system("cls");
    printf("\n\n  \xcd\xcd\xcd\xcd\xcd\xcd\n");
    printf("        BUS TICKET RESERVATION
         SYSTEM");
    printf("\n   \xcd\xcd\xcd\xcd\xcd\xcdn\n");
    printf("          Ticket Successfully
         Canceled.\n\n");
    printf("  \xcd\xcd\xcd\xcd\xcd\xcd\xcd\n");
    sit[(booking_historys[x-
         1].selected_destination)-
         1][(booking_historys[x-
         1].selected_time)-
         1][(booking_historys[x-
         1].selected_bus)-
         1][(booking_historys[x-
         1].selected_seat)-1]=0;
    sit_file_write();
    printf("   Press any key to continue... ");
```

```
                    getchar();
                    getchar();
                    history_file_write();
                    user_menu();
                    system("cls");
                }
                system("cls");
                user_menu();
                break;
                break;
            }else
            {
                system("cls");
                printf("\n\n  \xcd\xcd\xcd\xcd\xcd\xcd\xcd\n");
                printf("        BUS TICKET RESERVATION
                    SYSTEM");
                printf("\n  \xcd\xcd\xcd\xcd\xcd\xcd\xcd\n\n");
                printf("              Already canceled or no
                    access.\n\n");
                printf("   \xcd\xcd\xcd\xcd\xcd\xcd\xcd\n");
                printf("   Press any key to continue... ");
                getchar();
                getchar();
                system("cls");
            }
        }
    }
}
```

The cancel function allows users to cancel their booked tickets. Here's how it works:

1. Initialization:
   a) Initializes variables x and a.
   b) Defines string arrays b to represent the "Canceled" status and m to represent the "Booked" status.
2. Menu Display and Input:
   a) Clears the screen and displays the header for the bus ticket reservation system.
   b) Prompts the user to enter the ticket serial number.
3. Validation:
   a) Checks if the entered serial number is valid (within the range of available tickets).
   b) If the serial number is invalid, displays an error message and returns to the main menu.
4. Ticket Status Check:
   a) Verifies if the ticket corresponding to the entered serial number is currently booked by the same user.
   b) If the ticket is booked and belongs to the current user, prompts the user to confirm cancellation by pressing 'y'.

   c) If the ticket is not booked or does not belong to the current user, displays an appropriate message and returns to the main menu.
5. Cancellation Confirmation:
   a) If the user confirms cancellation by pressing 'y', updates the status of the ticket to "Canceled" in the booking_historys array.
   b) Displays a success message indicating that the ticket has been successfully canceled.
   c) Updates the seat availability status in the sit array and writes it to the seat file.
   d) Writes the updated booking history to the history file.
   e) Waits for user input to continue and returns to the main menu.
6. Loop Continuation:
   a) The function runs in an infinite loop until the user successfully cancels a ticket or returns to the main menu after encountering an error or cancelation.

Overall, the cancel function provides users with the ability to cancel their booked tickets, updating the system's records accordingly and ensuring accurate seat availability information.

The “history” Function

```
void history()
{
    int z=0;
    if (history_count==0)
    {
        printf("\n\n   \xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\n");
        printf("        BUS TICKET RESERVATION SYSTEM");
        printf("\n  \xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\n\n");
        printf("              No History Found!\n\n");
        printf("   \xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\n");
        printf("   Press any key to continue... ");
        getchar();
        getchar();
        system("cls");
        user_menu();
    }
    system("cls");
    printf("\n\n  \xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\n");
    printf("\t\t\t\t\t\t\t\t\t      BUS TICKET RESERVATION
SYSTEM\n");
    printf("  \xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcdn");
    printf("   \xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\n");
    printf("  \xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\n");
    printf("  |    %-10s    |    %-15s    |    %-
     25s    |    %-8s    |    %-8s    |    %-22s    |    %-
     12s    |    %-8s    |\n","Serial
```

```
     no","Destination","Bus","Time","Seat no","Passenger
     name","Phone no","Status");
    printf("   ----------------------------------------\n");
    for (int i = 0; i < history_count; i++)
    {
            if (temp==booking_historys[i].user_id)
            {
                z++;
            }
    }
    if(z==0)
    {
        system("cls");
        printf("\n\n  \xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\n");
        printf("        BUS TICKET RESERVATION SYSTEM");
        printf("\n   \xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\
xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\n\n");
        printf("              No History Found!\n\n");
        printf("   \xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\n");
    }else
    {
        for (int i = 0; i < history_count; i++)
        {
            if (temp==booking_historys[i].user_id)
            {
                printf("  |    %-10d    |    %-15s    |    %-
                    25s    |    %-8s    |    %-8d    |    %-
                    22s    |    %-12s    |    %-
     8s    |\n",booking_historys[i].serial,booking_historys[i].
     destination,booking_historys[i].bus_name,booking_historys
     [i].time,booking_historys[i].seat_no,booking_historys[i].
     name,booking_historys[i].phn_no,booking_historys[i].statu
     s);
                printf("   -------------------------------\n");
            }
        }
    }
        printf("   Press any key to continue... ");
        getchar();
        getchar();
        system("cls");
        if (selected_menu==5)
        {
            user_menu();
        }
}
```

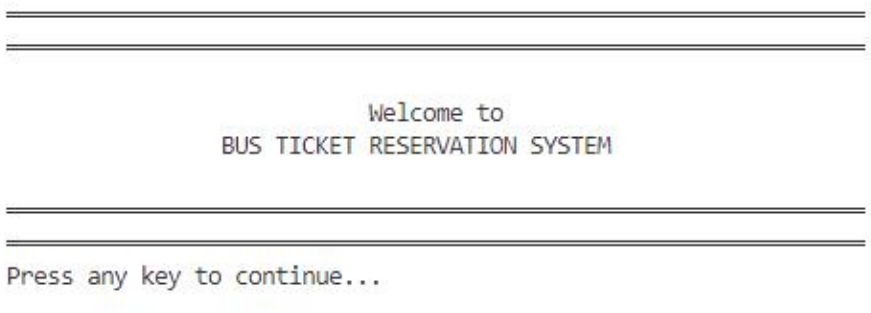
The history function is responsible for displaying the booking history of the current user. Here's a breakdown of its functionality:

1. Initialization:

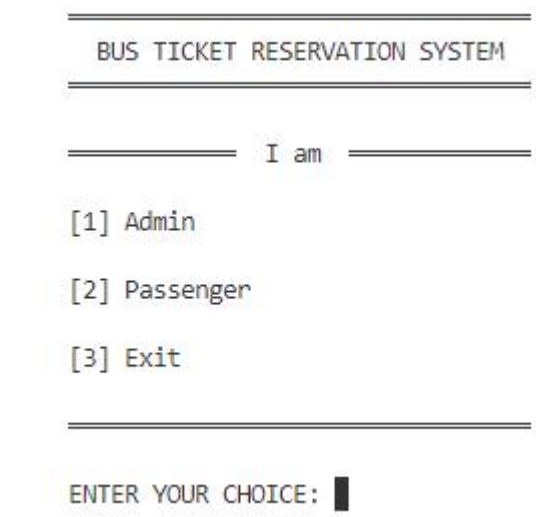
   a) Initializes the variable z to count the number of booking records associated with the current user.
2. Check for History:
   a) If history_count is zero, meaning there are no booking records in the system, it displays a message indicating no history found and returns to the main menu.
3. Display History:
   a) Clears the screen and displays the header for the bus ticket reservation system.
   b) Iterates through the booking history records.
   c) For each record, if the user ID matches the current user's ID (temp), it prints the details of the booking, including serial number, destination, bus name, time, seat number, passenger name, phone number, and status.
4. No History Found:
   a) If z is zero after the loop, it means there are no booking records associated with the current user. It displays a message indicating no history found.
5. User Interaction:
   a) Prompts the user to press any key to continue.
   b) Clears the screen.
   c) If the user accessed the history from the cancellation menu (selected_menu == 5), it returns to the main menu.

Overall, the history function provides users with the ability to view their booking history, including details of past bookings they've made. It ensures that users have access to their booking records for reference.

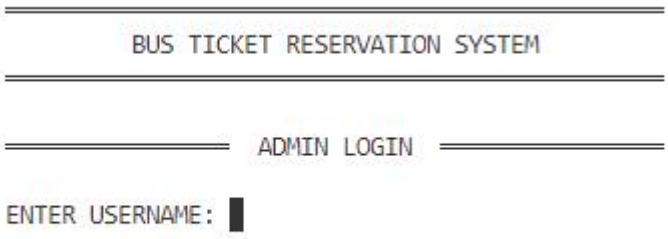
Output



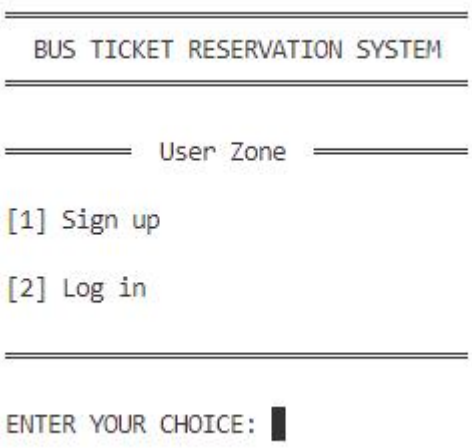
This is the Welcome screen.



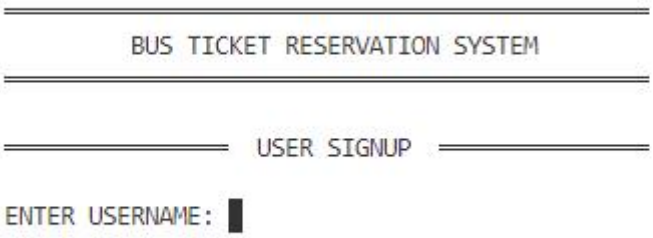
This is the user list



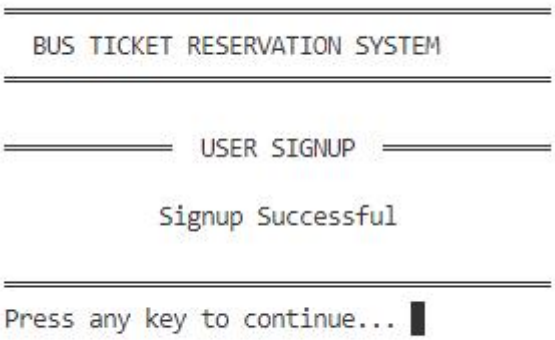
This is the Admin login Panel



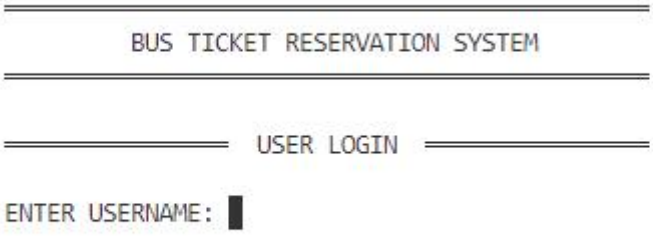
User signup_login Panel



User Signup Panel



Success Message



User Login Panel

```
═══════════════════════════════════════
  BUS TICKET RESERVATION SYSTEM
═══════════════════════════════════════

═══════════ USER LOGIN ═══════════

          Login Successful

═══════════════════════════════════════
Press any key to continue...
```

Login Success Message

```
═══════════════════════════════
  BUS TICKET RESERVATION SYSTEM
═══════════════════════════════

═══════ MAIN MENU ═══════

[1] View Bus Info

[2] Book Ticket

[3] Print Ticket

[4] Update Ticket Info

[5] Cancel Booking

[6] Booking History

[7] Sign out

═══════════════════════════════

ENTER YOUR CHOICE:
```

Main menu

```
═══════════════════════════════════════════════════════════════
                  BUS TICKET RESERVATION SYSTEM
═══════════════════════════════════════════════════════════════

═══════════════════════════ SEAT PLAN ═══════════════════════════

  [1 ] Available    [9 ] Available      [17] Available    [25] Available
  [2 ] Available    [10] Available      [18] Available    [26] Available
  [3 ] Available    [11] Available      [19] Available    [27] Available
  [4 ] Available    [12] Available      [20] Available    [28] Available
  [5 ] Available    [13] Available      [21] Available    [29] Available
  [6 ] Available    [14] Available      [22] Available    [30] Available
  [7 ] Available    [15] Available      [23] Available    [31] Available
  [8 ] Available    [16] Available      [24] Available    [32] Available


═══════════════════════════════════════════════════════════════

ENTER SIT NUMBER:
```

Bus Ticket Booking

```
                    Ticket: Hanif Enterprise                Serial: 9

    Destination: Dhaka                      Departure time: 12.00 PM
    Name: Md Raihan Chowdhoury              Phn no: 01771973004
    Seat no: 1                              Fare: BDT 200

  Wishing you a smooth trip filled with adventures and excitement.

Press any key to continue...
```

Ticket

```
                              BUS TICKET RESERVATION SYSTEM
                                          History
```

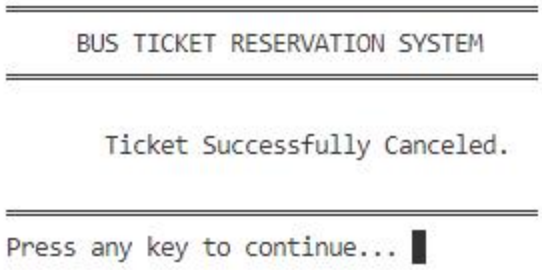
Serial no	Destination	Bus	Time	Seat no	Passenger name	Phone no	Status
1	Dhaka	Hanif Enterprice	07.00 AM	1	Raihan	01771973004	Canceled
2	Rajshahi	Greenline	10.00 AM	14	Alif	01200000000	Canceled
3	Dhaka	Hanif Enterprice	07.00 AM	4	Rumana	074776	Canceled
8	Dhaka	Hanif Enterprise	07.00 AM	9	Ratul	01958585955	Booked
9	Dhaka	Hanif Enterprise	12.00 PM	1	Md Raihan Chowdhoury	01771973004	Booked

```
Press any key to continue...
```

Booking History

```
  BUS TICKET RESERVATION SYSTEM

        MAIN MENU

[1] Update Name

[2] Update Phone no

ENTER YOUR CHOICE:
```

Update panel



Ticket Cancel

Limitations:

- User Interface: The user interface of the system is text-based and lacks graphical elements, which may not provide the best user experience compared to modern graphical user interfaces (GUIs).
- Input Validation: The system lacks comprehensive input validation, leaving it vulnerable to errors if users input unexpected data or characters.
- Security: The system does not implement robust security measures such as encryption for sensitive user data like passwords and personal information.
- Scalability: The system may face scalability issues if there is a significant increase in the number of users or booking requests due to its simplistic design.
- Error Handling: Error handling is minimal, and the system may not provide clear error messages or gracefully handle unexpected errors, leading to confusion for users.
- Single User Mode: The system appears to support only one user at a time, limiting its usability in scenarios where multiple users need to access the system simultaneously.
- Offline Functionality: It seems the system operates solely in an offline mode, lacking the capability to connect to external databases or APIs for real-time updates or integrations with other systems.

Conclusion:

Despite its limitations, the Bus Ticket Reservation System project demonstrates the foundational concepts of software development, including user input handling, data storage, and basic user authentication. It serves as a starting point for developing more robust and feature-rich systems in the future. To improve the system, considerations should be made to enhance user experience, implement stronger security measures, incorporate error handling mechanisms, and add scalability features to accommodate future growth. Additionally, integrating online functionality and enhancing the user interface could make the system more competitive in the modern market. Overall, while the project provides a functional

demonstration of a bus ticket reservation system, there is significant room for improvement to meet the demands of today's users and industry standards.