

**SEMESTER PROJECT (Operating System Lab) Fall- 2023**

TITLE: Linux Bus Reservation System

**Submitted by:**

**Name:** Md. Sami Alam

**ID:** 222-16-656

**Section:** A

**Subject Code:** **232L**

**Course Title:** **Operating System**

**Semester:** Fall- 2023

**Department** Of **CIS**, **DIU**

**Due Date:** 4rth November 2023

**Submitted to:**

**Name:** Ms. Sonia Nasrin

**Designation:** Lecturer

**Department:** Computing and Information System, DIU

Submission date 4th November, 2023

**Introduction**

This Bash script presents a functional Bus Reservation System, allowing users to register, log in, book tickets, check bus status, and even cancel their bookings. It manages various aspects, such as available buses, their seat limits, and real-time seat availability, enhancing the user experience. The script maintains user credentials in a file and provides secure login functionality. The booking feature allows users to choose a bus, specify the number of tickets, simulate payment, generate a unique reference number, and send email confirmations. It also enforces a time limit for ticket cancellations. The script's bus status check feature provides route and schedule information, and it simulates real-time status for each bus. Users can quickly switch between functionalities or return to the main menu. Overall, this script serves as a solid foundation for a practical bus reservation system and demonstrates fundamental Bash scripting techniques.

**Project code github link:** [**https://github.com/sami5671/Linux-Bus-Management-System**](https://github.com/sami5671/Linux-Bus-Management-System)

Motivation

In the journey of life, challenges are the stepping stones to growth. Embrace difficulties as opportunities, and setbacks as lessons. Remember that success often comes after a series of failures, and every obstacle you overcome strengthens your resilience. Stay focused on your goals, even when the path seems tough. Believe in your capabilities and keep your determination unwavering. Your dreams are worth pursuing, and your efforts will lead you to triumph. In the face of adversity, your persistence will shine through, and you'll emerge stronger, wiser, and closer to your aspirations. Keep pushing forward; your potential knows no bounds.

Objectives

The objective of this Bash-based Bus Reservation System is to provide a user-friendly platform for booking bus tickets. The system is designed to allow users to register, log in, and perform the following actions:

1. **User Registration:** Users can register with a unique username and password. The system checks for duplicate usernames.
2. **User Login:** Registered users can log in using their credentials to access the main menu.
3. **Booking Tickets:** Users can choose from available buses, select the number of tickets, and make a simulated payment. The system provides booking confirmation with a reference number.
4. **Checking Bus Status:** Users can check real-time status information for available buses, including route, departure, and arrival times.
5. **Canceling Tickets:** Users can cancel their booked tickets within a time limit (2 minutes).
6. **System Navigation:** Users can navigate between different functionalities using a clear menu interface.

**Required Tools**

Device (PC or Laptop):

Windows 11 HP EliteBook 840g5

Programming Language: Shell scripting(bash)

Development Environment:

* A Linux-based operating system (windows) for testing and execution.
* Text editor or IDE for shell scripting.

**Methodology**

The provided Bash script implements a simple Bus Reservation System. Below is the methodology of this code:

**1. Initialization:**

The script starts by initializing two associative arrays:

bus\_seat\_limits stores the seat limits for different buses.

bus\_available\_seats stores the current available seats for each bus.

Two file paths, user\_credentials\_file and booked\_tickets\_file, are set to store user credentials and booked tickets information.

**2. Pre-login Menu (pre\_login\_menu):**

A function is defined to display the pre-login menu with three options:

Login: Registered users can log in.

Register: New users can register.

Exit: To exit the system.

**3. User Registration (register Function):**

Users are prompted to enter a unique username and password.

The script checks if the username already exists in the user\_credentials\_file. If not, it saves the new user's credentials in the file.

**4. User Login (login Function):**

Registered users are prompted to enter their username and password.

The script checks if the provided username and password match the entries in the user\_credentials\_file. If the login is successful, users proceed to the main menu.

**5. Main Menu (main\_menu Function):**

After logging in, users are presented with a main menu.

The main menu offers the following options:

Book Tickets: Allows users to select and book bus tickets.

Check Bus Status: Provides real-time information about the bus status.

Cancel Tickets: Permits users to cancel their booked tickets.

Logout: Logs the user out and returns to the pre-login menu.

Exit: Exits the system.

**6. Booking Tickets (book\_tickets Function):**

Users can view available buses and their seat availability.

After selecting a bus and the number of tickets, users proceed to a simulated payment process.

Payment is simulated with card details (you can replace this with actual payment integration).

A reference number is generated for the booking.

Booking information is saved in the booked\_tickets\_file, and a confirmation email is sent to the user.

Users can return to the book tickets section or the main menu.

**7. Checking Bus Status (check\_bus\_status Function):**

Users can check real-time status information for each available bus.

The script provides details such as the bus route, departure time, and a randomly generated status (you can replace this with actual status data).

Users have the option to go back to the main menu or book tickets.

**8. Canceling Tickets (cancel\_tickets Function):**

Users can view their booked tickets and select one for cancellation.

A warning is issued that tickets cannot be canceled if more than 2 minutes have passed since booking.

Users can cancel a ticket within the time limit, which updates the canceled\_tickets.txt file.

**9. Main Program Loop:**

The script runs in a loop, repeatedly displaying the pre-login menu until the user chooses to exit the system.Users can log in, register, or exit the system.

**Overall:**

The system provides a command-line interface for bus ticket reservation and management. It includes functionalities for registration, login, booking, checking bus status, and canceling tickets. The script stores user credentials and booking information in text files. Some functionalities are simulated, like payment processing, and can be enhanced with real-world integration. It provides a basic example of a bus reservation system that can be extended and improved for practical use.

**Implementation (code + Output Screen shot)**

*#!/bin/bash*

*# Initialize available buses with their respective seat limits*

declare -A bus\_seat\_limits=( ["Bus A"]=50 ["Bus B"]=40 ["Bus C"]=30 )

*# Initialize available seats for each bus*

declare -A bus\_available\_seats=( ["Bus A"]=50 ["Bus B"]=40 ["Bus C"]=30 )

*# File to store registered user credentials*

user\_credentials\_file="user\_credentials.txt"

*# File to store booked tickets*

booked\_tickets\_file="booked\_tickets.txt"

*# Function to display the pre-login menu*

pre\_login\_menu() {

  clear

  echo "------------------->>>>>>>>>>>>>>>>>> BUS RESERVATION SYSTEM <<<<<<<<<<<<<<<<<<<----------------"

  echo "1. Login"

  echo "2. Register"

  echo "3. Exit"

}

*# Function for user registration*

register() {

  clear

  echo "------------------->>>>>>>>>>>>>>>>>> USER REGISTRATION <<<<<<<<<<<<<<<<<<<----------------"

  read -p "Enter a username: " new\_username

*# Check if the username already exists*

*if* grep -q "^$new\_username:" "$user\_credentials\_file"; *then*

    echo "Username already exists. Please choose a different username."

*else*

    read -p "Enter a password: " new\_password

*# Store the new user's credentials in the user\_credentials\_file*

    echo "$new\_username:$new\_password" >> "$user\_credentials\_file"

    echo "Registration successful! You can now log in."

*fi*

  read -n 1 -s -r -p "Press any key to continue..."

}

*# Function for user login*

login() {

  clear

  echo "------------------->>>>>>>>>>>>>>>>>> LOGIN SYSTEM <<<<<<<<<<<<<<<<<<<----------------"

  read -p "Enter your username: " username

  read -p "Enter your password: " password

*# Check if username and password are correct*

*# Add your authentication logic here*

*if* grep -q "^$username:$password" "$user\_credentials\_file"; *then*

    echo "Login successful!"

    read -n 1 -s -r -p "Press any key to continue..."

    main\_menu "$username"  *# Pass the username to the main menu*

*else*

    echo "Login failed. Invalid username or password."

    read -n 1 -s -r -p "Press any key to continue..."

*fi*

}

*# Function for the main menu*

main\_menu() {

  local username="$1"  *# Get the username passed from login*

*while* true; *do*

    clear

    echo "------------------->>>>>>>>>>>>>>>>>> BUS RESERVATION SYSTEM- WELCOME <<<-----, $username"

    echo "1. Book Tickets"

    echo "2. Check Bus Status"

    echo "3. Cancel Tickets"

    echo "4. Logout"

    echo "5. Exit"

    read -p "Enter your choice: " choice

*case* $choice *in*

      1)

        book\_tickets

        ;;

      2)

        check\_bus\_status

        ;;

      3)

        cancel\_tickets

        ;;

      4)

*return*  *# Return to the pre-login menu (logout)*

        ;;

      5)

        clear

        echo "Exiting the Bus Reservation System. Goodbye!"

        exit 0

        ;;

      \*)

        echo "Invalid choice. Please select a valid option."

        read -n 1 -s -r -p "Press any key to continue..."

        ;;

*esac*

*done*

}

*# Function for booking tickets*

book\_tickets() {

*while* true; *do*

    clear

    echo "------------------->>>>>>>>>>>>>>>>>> BOOKING OF TICKETS <<<<<<<<<<<<<<<<<<<----------------"

    echo "Available Buses with Available Seats:"

*# Loop through the available buses and display seat information*

    available\_buses=("${!bus\_seat\_limits[@]}")

*for* ((i = 0; i < ${#available\_buses[@]}; i++)); *do*

      bus="${available\_buses[i]}"

      seat\_limit="${bus\_seat\_limits["$bus"]}"

      available\_seats="${bus\_available\_seats["$bus"]}"

      echo "$(($i + 1)). $bus - Available Seats: $available\_seats/$seat\_limit"

*done*

    echo "0. Back to Main Menu"  *# Added "Back to Main Menu" option*

    read -p "Select a bus (0 to go back, 1-${#available\_buses[@]}): " bus\_choice

*if* [ "$bus\_choice" -eq 0 ]; *then*

*return*  *# Go back to the main menu*

*elif* [ "$bus\_choice" -ge 1 ] && [ "$bus\_choice" -le ${#available\_buses[@]} ]; *then*

      selected\_bus="${available\_buses[$(($bus\_choice - 1))]}"

      seat\_limit="${bus\_seat\_limits["$selected\_bus"]}"

      available\_seats="${bus\_available\_seats["$selected\_bus"]}"

      echo "You have selected: $selected\_bus"

      echo "Seat Limit: $seat\_limit"

      echo "Available Seats: $available\_seats"

      echo "One ticket costs 500 BDT."

      read -p "Enter number of tickets: " num\_tickets

*if* [ "$num\_tickets" -le "$available\_seats" ]; *then*

*# Calculate the total price*

        total\_price=$((num\_tickets \* 500))  *# Assuming 1 ticket costs 500 TK*

*# Generate a 4-digit reference number for the booking*

        reference\_number=$((1000 + RANDOM % 9000))

*# Update available seats*

        new\_available\_seats=$((available\_seats - num\_tickets))

        bus\_available\_seats["$selected\_bus"]=$new\_available\_seats

*# Capture the booking time*

        booking\_time=$(date +"%Y-%m-%d %H:%M:%S")

*# Demo card details*

        echo " Here is the Demo Card Details--------->>>>>    Credit card number: 1234, card\_expiration: 12/25, CVV: 123   <<<<<<<-------------"

*# Payment processing logic*

        read -p "Enter your credit card number: " credit\_card\_number

        read -p "Enter the card expiration date (MM/YY): " card\_expiration

        read -p "Enter the CVV: " cvv

*# Simulate payment processing (you can replace this with actual payment gateway integration)*

*if* [ "$credit\_card\_number" = "1234" ] && [ "$card\_expiration" = "12/25" ] && [ "$cvv" = "123" ]; *then*

*# Payment successful*

          echo "Payment successful!"

*# Display the total price*

          echo "Total Price: $total\_price BDT"

*# Prompt the user for their email address*

          read -p "Enter your email address for booking confirmation: " user\_email

*# Save booking information to a text file*

          booking\_info="Reference Number: $reference\_number, Bus: $selected\_bus, Tickets: $num\_tickets, Total Price: $total\_price BDT, Booking Time: $booking\_time"

          echo "$booking\_info" >> "$booked\_tickets\_file"

          echo "Tickets booked successfully! Your reference number is $reference\_number."

*# Send an email to the user with payment details and booking time*

          email\_subject="Bus Ticket Booking Confirmation"

          email\_body="Dear user,\n\nYour bus ticket booking has been confirmed with the following details:\n\n$booking\_info\n\nThank you for choosing our service!\n\nBest regards,\nThe Bus Reservation Team"

*# Use the mail command to send the email*

          echo -e "$email\_body" | mail -s "$email\_subject" "$user\_email"

          echo "An email with booking details has been sent to $user\_email."

          read -n 1 -s -r -p "Press any key to continue..."

*else*

*# Payment failed*

          echo "Payment failed. Please check your payment details and try again."

          read -n 1 -s -r -p "Press any key to continue..."

*fi*

*# Provide an option to go back to the book bus section*

        echo "1. Back to Book Bus Section"

        read -p "Enter your choice (1 to go back): " back\_choice

*if* [ "$back\_choice" -eq 1 ]; *then*

*continue*  *# Go back to the book bus section*

*fi*

*else*

        echo "Sorry, the requested number of tickets exceeds the available seats for $selected\_bus."

        read -n 1 -s -r -p "Press any key to continue..."

*fi*

*else*

      echo "Invalid choice. Please select a valid bus or 0 to go back."

      read -n 1 -s -r -p "Press any key to continue..."

*fi*

*done*

}

*# Function for checking bus status and seat availability*

check\_bus\_status() {

*while* true; *do*

    clear

    echo "------------------->>>>>>>>>>>>>>>>>>>>>> CHECKING BUS STATUS <<<<<<<<<<<<<<<<<<<<<<<<-----------------"

*# Define bus information, including route and schedule*

    declare -A bus\_info=(

      ["Bus A"]="Route: Route A, Departure: 08:00 AM, Arrival: 10:30 AM"

      ["Bus B"]="Route: Route B, Departure: 09:00 AM, Arrival: 11:30 AM"

      ["Bus C"]="Route: Route C, Departure: 10:00 AM, Arrival: 12:30 PM"

    )

*# Fetch real-time status for each bus*

*for* bus *in* "${!bus\_info[@]}"; *do*

*# Generate a random status (you can replace this with actual data)*

      bus\_statuses=("On Time" "Delayed" "Cancelled" "Arriving Soon" "Boarding")

      random\_index=$((RANDOM % ${#bus\_statuses[@]}))

      status="${bus\_statuses[$random\_index]}"

      echo "Bus: $bus"

      echo "${bus\_info["$bus"]}"

      echo "Status: $status"

      echo "------------------------"

*done*

*# Provide options to go back to the main menu or book tickets*

    echo "0. Back to Main Menu"

    echo "B. Book Tickets"

    read -p "Enter your choice (0/B): " choice

*if* [ "$choice" = "0" ]; *then*

*return*  *# Go back to the main menu*

*elif* [ "$choice" = "B" ] || [ "$choice" = "b" ]; *then*

      book\_tickets  *# Call the book\_tickets function*

*else*

      echo "Invalid choice. Please select '0' to go back to the main menu or 'B' to book tickets."

      read -n 1 -s -r -p "Press any key to continue..."

*fi*

*done*

}

*# Function for canceling tickets*

cancel\_tickets() {

*while* true; *do*

    clear

    echo "------------------->>>>>>>>>>>>>>>>>>>>>> CANCEL TICKETS <<<<<<<<<<<<<<<<<<<<<<<<-----------------"

    echo "Booked Bus Tickets:"

    echo "----------------->>>>>>>>>>> WARNING!!!! YOU CANNOT CANCEL TICKETS. IF TIME LIMIT EXCEED TO 2 MINUTES <<<<<<<<<<<<<----"

    cat "$booked\_tickets\_file"

    read -p "Enter booking reference number (4 digits) or 0 to go back to the Main Menu: " reference\_number

*if* [ "$reference\_number" -eq 0 ]; *then*

*return*  *# Go back to the main menu*

*fi*

*# Check if the reference number exists in the booked\_tickets.txt file*

*if* grep -q "Reference Number: $reference\_number" "$booked\_tickets\_file"; *then*

*# Find the canceled ticket entry*

      canceled\_ticket\_info=$(grep "Reference Number: $reference\_number" "$booked\_tickets\_file")

*# Extract the booking time from the canceled ticket info*

      booking\_time=$(grep "Reference Number: $reference\_number" "$booked\_tickets\_file" | awk '{print $NF}')

*# Calculate the current time*

      current\_time=$(date +"%s")

*# Convert the booking time to seconds since epoch*

      booking\_time\_seconds=$(date -d "$booking\_time" +"%s")

*# Calculate the time difference in seconds*

      time\_diff=$((current\_time - booking\_time\_seconds))

*# Check if the time difference is within the cancellation limit (2 minutes = 120 seconds)*

*if* [ "$time\_diff" -le 120 ]; *then*

*# Remove the booking entry from booked\_tickets.txt*

        sed -i "/Reference Number: $reference\_number/d" "$booked\_tickets\_file"

*# Add a header with a timestamp for the canceled ticket in canceled\_tickets.txt*

        cancel\_timestamp=$(date +"%Y-%m-%d %H:%M:%S")

        echo "Cancellation Timestamp: $cancel\_timestamp" >> canceled\_tickets.txt

        echo "$canceled\_ticket\_info" >> canceled\_tickets.txt

        echo "" >> canceled\_tickets.txt  *# Add an empty line for separation*

        echo "Ticket with reference number $reference\_number canceled successfully!"

*else*

        echo "Ticket cannot be canceled. Time limit for cancellation has expired."

*fi*

*else*

      echo "Invalid reference number. No matching booking found."

*fi*

    read -n 1 -s -r -p "Press any key to continue..."

*done*

}

*# Main loop for the program*

*while* true; *do*

  pre\_login\_menu

  read -p "Enter your choice: " pre\_login\_choice

*case* $pre\_login\_choice *in*

    1)

      login

      ;;

    2)

      register

      ;;

    3)

      clear

      echo "Exiting the Bus Reservation System. Goodbye!"

      exit 0

      ;;

    \*)

      echo "Invalid choice. Please select a valid option."

      read -n 1 -s -r -p "Press any key to continue..."

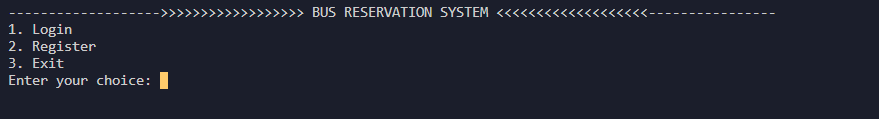
      ;;

*esac*

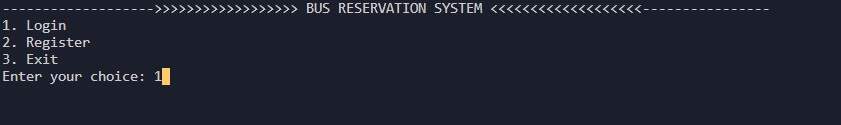
*done*

**Output**

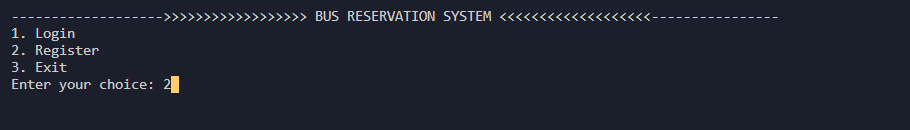
Output 1:



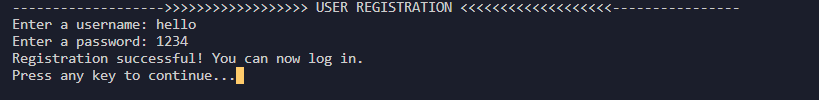
Output 2:



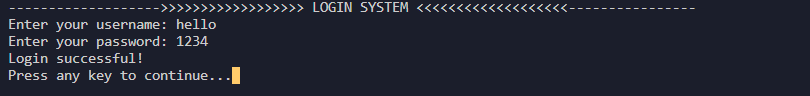
Output 3:



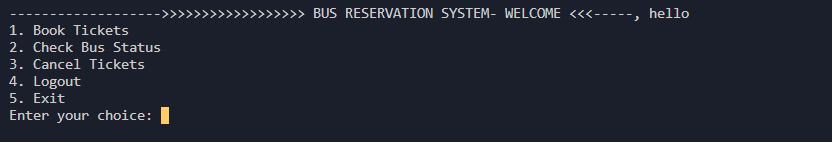
Output 4:



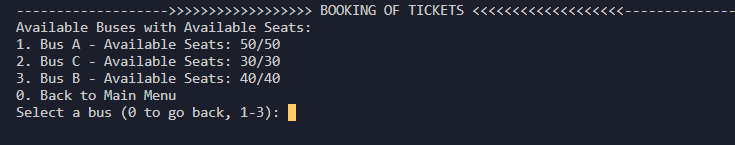
Output 5:



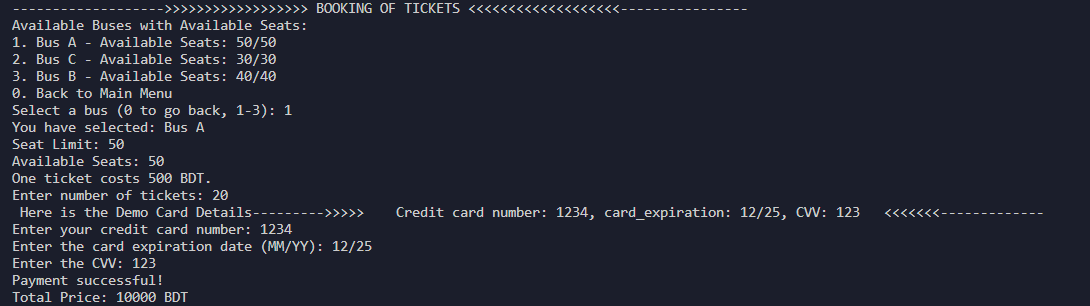
Output 6:



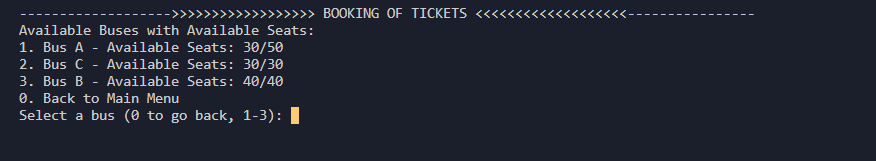
Output 7:



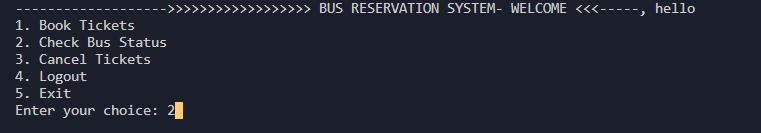
Output 8:



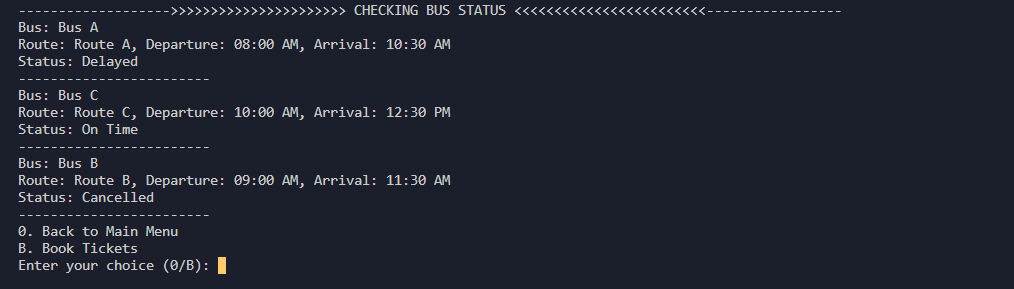
Output 9: (bus ticket of Bus A has now 30 tickets available)



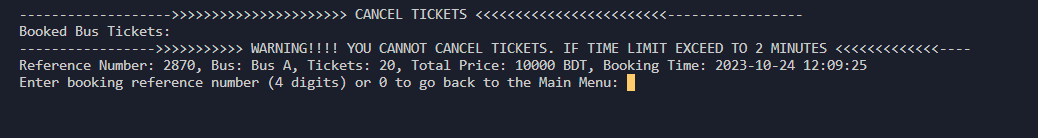
Output 10:



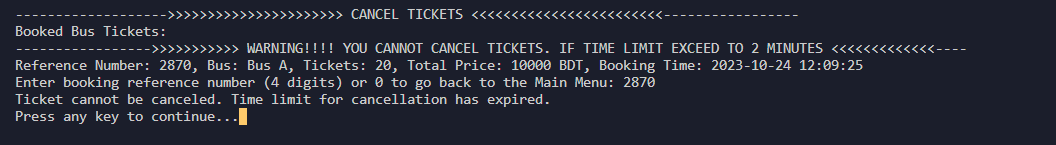
Output 11:



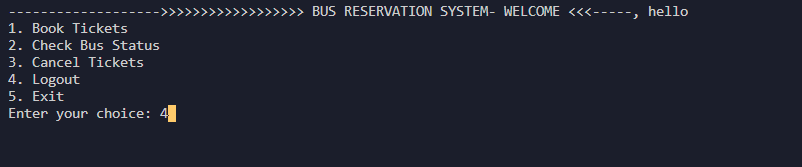
Output 12: (Cancel tickets )



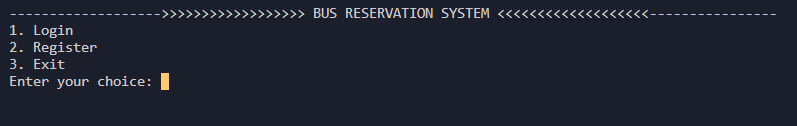
Output 13: (ticket cannot be cancel because cancellation time limit exceed)



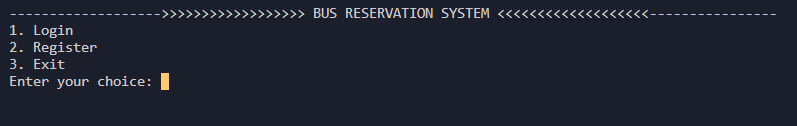
Output 14: (for logout)

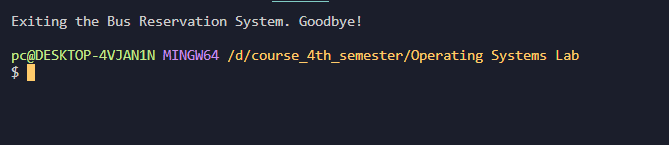


Output 15: (After logout)



Output: 16 (Exit the System if press 3)





**Conclusion**

In conclusion, the Bash script presents a basic Bus Reservation System that allows user registration, login, ticket booking, status checks, and cancellations. It utilizes data structures for managing bus seat availability and user credentials, with corresponding data stored in text files. The code includes a simulated payment process and confirmation email. While serving as a functional example of a reservation system, it requires substantial improvements for production use, including enhanced security, data validation, and comprehensive error handling. Nevertheless, it offers a fundamental understanding of a reservation system's core components, serving as a valuable starting point for more robust implementations.

**Reference**

**THANK YOU FOR YOUR PATIENCE**