## **Bus Reservation System**

### **Introduction**

This project focuses on developing a comprehensive bus reservation system that allows users to book and manage bus tickets efficiently. The system provides features for user authentication, ticket booking, cancellation, checking bus status, and managing user credentials. By leveraging a client-server architecture, the project aims to deliver a reliable and interactive application that can handle various operations related to bus reservations.

### **Project Objective**

The main objective of this project is to create a robust bus reservation system with the following features:

* **User Authentication:** Login, and manage user credentials.
* **Ticket Management:** Book and cancel tickets, and check bus availability.
* **Bus Status:** Retrieve and display bus details, including available seats and fare.
* **User Management:** Change and reset passwords. The system is designed to handle multiple user requests and provide real-time updates on bus status.

### **System Design**

The system is designed with a client-server architecture:

* **Client:** The user interface allows interaction with the system, including login, booking, and managing tickets. It sends requests to the server and displays responses.
* **Server:** Handles client requests, processes operations such as booking and cancelling tickets, and manages bus and user data. It communicates with the client to provide status updates and handle requests.

**Modules:**

1. **Authentication Module:** Manages user registration, login, and credential updates.
2. **Ticket Management Module:** Handles booking and cancellation of tickets.
3. **Bus Status Module:** Provides information about bus routes, seats, and fares.
4. **User Management Module:** Allows users to change and reset passwords.

**Communication:**

* **Sockets:** Used for client-server communication to send and receive data.

### **Source Code**

The source code is structured into multiple files, including:

* **Client Code :** Manages user interactions, sends requests to the server, and processes responses.
* **Server Code :** Handles client requests, manages data, and performs operations related to ticketing and user management.

**Client Code**

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <unistd.h>

#include <arpa/inet.h>

#define PORT 8080

#define MAXLINE 1024

void loginUser(int sock);

void bookTicket(int sock);

void cancelTicket(int sock);

void checkBusStatus(int sock);

void changePassword(int sock);

void forgotPassword(int sock);

int main() {

int sock = 0;

struct sockaddr\_in serv\_addr;

if ((sock = socket(AF\_INET, SOCK\_STREAM, 0)) < 0) {

printf("\n Socket creation error \n");

return -1;

}

serv\_addr.sin\_family = AF\_INET;

serv\_addr.sin\_port = htons(PORT);

if (inet\_pton(AF\_INET, "127.0.0.1", &serv\_addr.sin\_addr) <= 0) {

printf("\nInvalid address/ Address not supported \n");

return -1;

}

if (connect(sock, (struct sockaddr \*)&serv\_addr, sizeof(serv\_addr)) < 0) {

printf("\nConnection Failed \n");

return -1;

}

printf("Socket created\nConnected\n");

while (1) {

printf("\n=== Bus Reservation System ===\n");

printf("1. Login\n");

printf("2. Exit\n");

printf("Enter your choice: ");

int choice;

scanf("%d", &choice);

switch (choice) {

case 1:

loginUser(sock);

break;

case 2:

printf("Exiting the program.\n");

close(sock);

exit(0);

default:

printf("Invalid choice. Please try again.\n");

}

}

return 0;

}

void loginUser(int sock) {

char username[50], password[50];

char buffer[MAXLINE] = {0};

printf("Enter Username: ");

scanf("%s", username);

printf("Enter Password: ");

scanf("%s", password);

snprintf(buffer, MAXLINE, "LOGIN %s %s", username, password);

send(sock, buffer, strlen(buffer), 0);

read(sock, buffer, MAXLINE);

printf("%s\n", buffer);

if (strstr(buffer, "successful")) {

while (1) {

printf("\n=== User Menu ===\n");

printf("1. Book a Ticket\n");

printf("2. Cancel a Ticket\n");

printf("3. Check Bus Status\n");

printf("4. Change Password\n");

printf("5. Forgot Password\n");

printf("6. Logout\n");

printf("Enter your choice: ");

int choice;

scanf("%d", &choice);

switch (choice) {

case 1:

bookTicket(sock);

break;

case 2:

cancelTicket(sock);

break;

case 3:

checkBusStatus(sock);

break;

case 4:

changePassword(sock);

break;

case 5:

forgotPassword(sock);

break;

case 6:

printf("Logging out.\n");

return;

default:

printf("Invalid choice. Please try again.\n");

}

}

}

}

void bookTicket(int sock) {

char busNumber[10], name[50], age[5];

char buffer[MAXLINE] = {0};

printf("Enter Bus Number: ");

scanf("%s", busNumber);

printf("Enter Name: ");

scanf("%s", name);

printf("Enter Age: ");

scanf("%s", age);

snprintf(buffer, MAXLINE, "BOOK %s %s %s", busNumber, name, age);

send(sock, buffer, strlen(buffer), 0);

read(sock, buffer, MAXLINE);

printf("%s\n", buffer);

}

void cancelTicket(int sock) {

char name[50];

char buffer[MAXLINE] = {0};

printf("Enter Name to Cancel Ticket: ");

scanf("%s", name);

snprintf(buffer, MAXLINE, "CANCEL %s", name);

send(sock, buffer, strlen(buffer), 0);

read(sock, buffer, MAXLINE);

printf("%s\n", buffer);

}

void checkBusStatus(int sock) {

char busNumber[10];

char buffer[MAXLINE] = {0};

printf("Enter Bus Number to Check Status: ");

scanf("%s", busNumber);

snprintf(buffer, MAXLINE, "STATUS %s", busNumber);

send(sock, buffer, strlen(buffer), 0);

read(sock, buffer, MAXLINE);

printf("%s\n", buffer);

}

void changePassword(int sock) {

char username[50], oldPassword[50], newPassword[50];

char buffer[MAXLINE] = {0};

printf("Enter Username: ");

scanf("%s", username);

printf("Enter Old Password: ");

scanf("%s", oldPassword);

printf("Enter New Password: ");

scanf("%s", newPassword);

snprintf(buffer, MAXLINE, "CHANGE\_PASSWORD %s %s %s", username, oldPassword, newPassword);

send(sock, buffer, strlen(buffer), 0);

read(sock, buffer, MAXLINE);

printf("%s\n", buffer);

}

void forgotPassword(int sock) {

char username[50], newPassword[50];

char buffer[MAXLINE] = {0};

printf("Enter Username: ");

scanf("%s", username);

printf("Enter New Password: ");

scanf("%s", newPassword);

snprintf(buffer, MAXLINE, "FORGOT\_PASSWORD %s %s", username, newPassword);

send(sock, buffer, strlen(buffer), 0);

read(sock, buffer, MAXLINE);

printf("%s\n", buffer);

}

### **Server Code**

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <unistd.h>

#include <arpa/inet.h>

#define PORT 8080

#define MAXLINE 1024

typedef struct {

char username[50];

char password[50];

} User;

typedef struct {

int busNumber;

char source[50];

char destination[50];

int totalSeats;

int availableSeats;

float fare;

} Bus;

User users[] = {

{"user1", "password1"},

{"user2", "password2"}

};

Bus buses[] = {

{101, "City A", "City B", 50, 50, 25.00}

};

void handleClient(int new\_socket);

int authenticateUser(char \*username, char \*password);

void bookTicket(int new\_socket, char \*busNumber, char \*name, char \*age);

void cancelTicket(int new\_socket, char \*name);

void checkBusStatus(int new\_socket, char \*busNumber);

void changePassword(int new\_socket, char \*username, char \*oldPassword, char \*newPassword);

void forgotPassword(int new\_socket, char \*username, char \*newPassword);

int main() {

int server\_fd, new\_socket;

struct sockaddr\_in address;

int opt = 1;

int addrlen = sizeof(address);

if ((server\_fd = socket(AF\_INET, SOCK\_STREAM, 0)) == 0) {

perror("socket failed");

exit(EXIT\_FAILURE);

}

if (setsockopt(server\_fd, SOL\_SOCKET, SO\_REUSEADDR | SO\_REUSEPORT, &opt, sizeof(opt))) {

perror("setsockopt");

exit(EXIT\_FAILURE);

}

address.sin\_family = AF\_INET;

address.sin\_addr.s\_addr = INADDR\_ANY;

address.sin\_port = htons(PORT);

if (bind(server\_fd, (struct sockaddr \*)&address, sizeof(address)) < 0) {

perror("bind failed");

exit(EXIT\_FAILURE);

}

if (listen(server\_fd, 3) < 0) {

perror("listen");

exit(EXIT\_FAILURE);

}

printf("Server is running and waiting for connections...\n");

while (1) {

if ((new\_socket = accept(server\_fd, (struct sockaddr \*)&address, (socklen\_t\*)&addrlen)) < 0) {

perror("accept");

exit(EXIT\_FAILURE);

}

handleClient(new\_socket);

}

return 0;

}

void handleClient(int new\_socket) {

char buffer[MAXLINE] = {0};

int valread;

char \*token;

char \*command;

char \*username, \*password, \*oldPassword, \*newPassword, \*busNumber, \*name, \*age;

while ((valread = read(new\_socket, buffer, MAXLINE)) > 0) {

buffer[valread] = '\0';

command = strtok(buffer, " ");

if (strcmp(command, "LOGIN") == 0) {

username = strtok(NULL, " ");

password = strtok(NULL, " ");

if (authenticateUser(username, password)) {

send(new\_socket, "Login successful. Welcome!", strlen("Login successful. Welcome!"), 0);

} else {

send(new\_socket, "Login failed. Please check your username and password.", strlen("Login failed. Please check your username and password."), 0);

}

} else if (strcmp(command, "BOOK") == 0) {

busNumber = strtok(NULL, " ");

name = strtok(NULL, " ");

age = strtok(NULL, " ");

bookTicket(new\_socket, busNumber, name, age);

} else if (strcmp(command, "CANCEL") == 0) {

name = strtok(NULL, " ");

cancelTicket(new\_socket, name);

} else if (strcmp(command, "STATUS") == 0) {

busNumber = strtok(NULL, " ");

checkBusStatus(new\_socket, busNumber);

} else if (strcmp(command, "CHANGE\_PASSWORD") == 0) {

username = strtok(NULL, " ");

oldPassword = strtok(NULL, " ");

newPassword = strtok(NULL, " ");

changePassword(new\_socket, username, oldPassword, newPassword);

} else if (strcmp(command, "FORGOT\_PASSWORD") == 0) {

username = strtok(NULL, " ");

newPassword = strtok(NULL, " ");

forgotPassword(new\_socket, username, newPassword);

}

memset(buffer, 0, MAXLINE);

}

close(new\_socket);

}

int authenticateUser(char \*username, char \*password) {

for (int i = 0; i < sizeof(users) / sizeof(users[0]); i++) {

if (strcmp(users[i].username, username) == 0 && strcmp(users[i].password, password) == 0) {

return 1;

}

}

return 0;

}

void bookTicket(int new\_socket, char \*busNumber, char \*name, char \*age) {

int busNum = atoi(busNumber);

for (int i = 0; i < sizeof(buses) / sizeof(buses[0]); i++) {

if (buses[i].busNumber == busNum && buses[i].availableSeats > 0) {

buses[i].availableSeats--;

char response[MAXLINE];

snprintf(response, MAXLINE, "Ticket booked successfully! Seat Number: %d", buses[i].totalSeats - buses[i].availableSeats);

send(new\_socket, response, strlen(response), 0);

return;

}

}

send(new\_socket, "Failed to book ticket.", strlen("Failed to book ticket."), 0);

}

void cancelTicket(int new\_socket, char \*name) {

// Simulate cancellation

for (int i = 0; i < sizeof(buses) / sizeof(buses[0]); i++) {

if (buses[i].availableSeats < buses[i].totalSeats) {

buses[i].availableSeats++;

send(new\_socket, "Ticket canceled successfully!", strlen("Ticket canceled successfully!"), 0);

return;

}

}

send(new\_socket, "Failed to cancel ticket.", strlen("Failed to cancel ticket."), 0);

}

void checkBusStatus(int new\_socket, char \*busNumber) {

int busNum = atoi(busNumber);

for (int i = 0; i < sizeof(buses) / sizeof(buses[0]); i++) {

if (buses[i].busNumber == busNum) {

char response[MAXLINE];

snprintf(response, MAXLINE, "Bus Number: %d\nSource: %s\nDestination: %s\nTotal Seats: %d\nAvailable Seats: %d\nFare: %.2f",

buses[i].busNumber, buses[i].source, buses[i].destination, buses[i].totalSeats, buses[i].availableSeats, buses[i].fare);

send(new\_socket, response, strlen(response), 0);

return;

}

}

send(new\_socket, "Bus not found.", strlen("Bus not found."), 0);

}

void changePassword(int new\_socket, char \*username, char \*oldPassword, char \*newPassword) {

for (int i = 0; i < sizeof(users) / sizeof(users[0]); i++) {

if (strcmp(users[i].username, username) == 0 && strcmp(users[i].password, oldPassword) == 0) {

strcpy(users[i].password, newPassword);

send(new\_socket, "Password changed successfully.", strlen("Password changed successfully."), 0);

return;

}

}

send(new\_socket, "Failed to change password.", strlen("Failed to change password."), 0);

}

void forgotPassword(int new\_socket, char \*username, char \*newPassword) {

for (int i = 0; i < sizeof(users) / sizeof(users[0]); i++) {

if (strcmp(users[i].username, username) == 0) {

strcpy(users[i].password, newPassword);

send(new\_socket, "Password reset successfully.", strlen("Password reset successfully."), 0);

return;

}

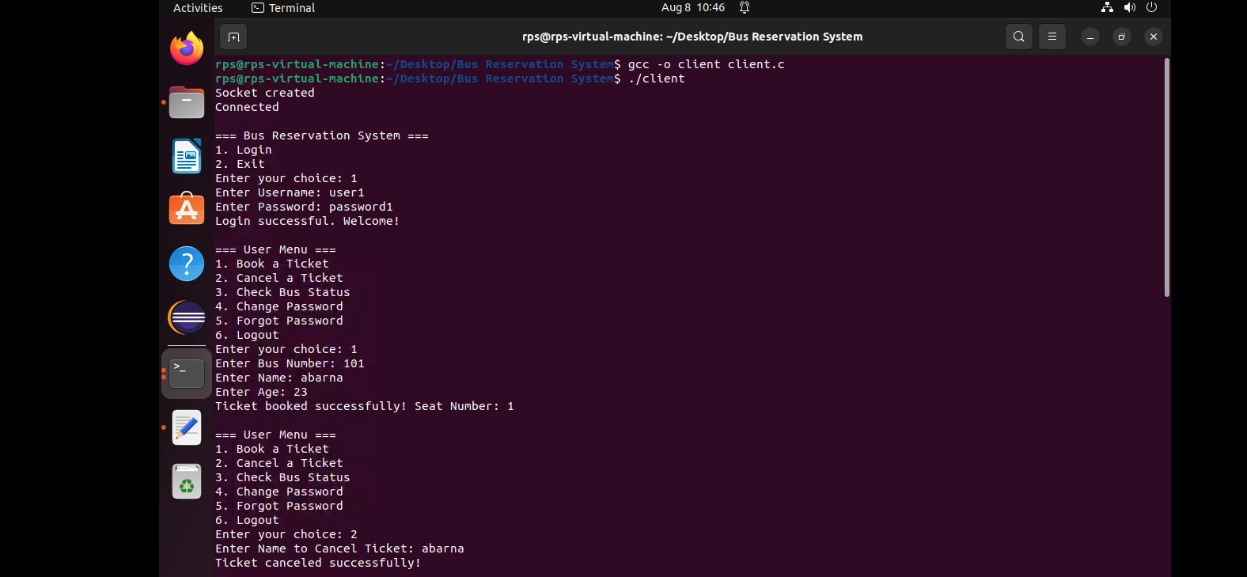
}

send(new\_socket, "Failed to reset password.", strlen("Failed to reset password."), 0);

}

### **Output**

The system output includes:

****

### 

### 

### 

### 



### 

### 

### 

### 

### 

### 

### **Future Enhancements**

1. **Handling Increased Load:**
   * **Challenge:** The system may need to accommodate higher traffic and data volumes.
   * **Enhancement:** Implement load balancing, optimise data handling, and consider asynchronous processing to improve scalability and performance.
2. **Extended Features:**
   * **Challenge:** Users may request additional functionalities.
   * **Enhancement:** Add features such as advanced reporting, support for additional transportation modes, and integration with external services.

### **Conclusion**

The bus reservation system project demonstrates the ability to develop a functional and interactive application with real-time capabilities. It provides essential features for managing bus reservations and user accounts, while also offering opportunities for future improvements and extensions. The project serves as a practical example of client-server architecture and system design, showcasing the integration of various software components and technologies.