#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <unistd.h>

#include <pthread.h>

#include <arpa/inet.h>

#include <time.h>

// Function declarations

void establish\_connection();

void display\_main\_menu();

void send\_signup\_request();

void send\_login\_request();

void display\_post\_login\_menu();

void send\_process\_cdr\_request();

void display\_billing\_info\_menu();

void display\_customer\_billing\_menu();

void display\_interoperator\_billing\_menu();

void send\_search\_msisdn\_request();

void send\_search\_operator\_request();

void logout();

void close\_connection();

void log\_message(const char \*level, const char \*message);

// Global variables

int sockfd;

struct sockaddr\_in server\_addr;

FILE \*log\_file;

// Function to establish connection to the server

void establish\_connection() {

sockfd = socket(AF\_INET, SOCK\_STREAM, 0);

if (sockfd < 0) {

log\_message("FATAL", "Failed to create socket");

exit(EXIT\_FAILURE);

}

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

server\_addr.sin\_port = htons(6789);

server\_addr.sin\_addr.s\_addr = inet\_addr("127.0.0.1");

if (connect(sockfd, (struct sockaddr \*)&server\_addr, sizeof(server\_addr)) < 0) {

log\_message("FATAL", "Failed to connect to server");

exit(EXIT\_FAILURE);

}

log\_message("INFO", "Connected to server");

}

// Function to display the main menu

void display\_main\_menu() {

printf("\nMain Menu:\n");

printf("1. SignUp\n");

printf("2. Login\n");

printf("3. Exit\n");

printf("Choice: ");

}

// Function to send signup request to the server

void send\_signup\_request() {

char username[50], password[50];

printf("Enter username: ");

scanf("%s", username);

printf("Enter password: ");

scanf("%s", password);

// Send the signup request to the server

char request[100];

snprintf(request, sizeof(request), "SIGNUP|%s|%s", username, password);

send(sockfd, request, strlen(request), 0);

}

// Function to send login request to the server

void send\_login\_request() {

char username[50], password[50];

printf("Enter username: ");

scanf("%s", username);

printf("Enter password: ");

scanf("%s", password);

// Send the login request to the server

char request[100];

snprintf(request, sizeof(request), "LOGIN|%s|%s", username, password);

send(sockfd, request, strlen(request), 0);

}

// Function to display post-login menu

void display\_post\_login\_menu() {

printf("\nPost Login Menu:\n");

printf("1. Process CDR file\n");

printf("2. Print/Search Billing Information\n");

printf("3. Logout\n");

printf("Choice: ");

}

// Function to send a CDR processing request to the server

void send\_process\_cdr\_request() {

char file\_name[50];

printf("Enter CDR file name to process: ");

scanf("%s", file\_name);

// Send the CDR processing request to the server

char request[100];

snprintf(request, sizeof(request), "PROCESS\_CDR|%s", file\_name);

send(sockfd, request, strlen(request), 0);

}

// Function to display billing info menu

void display\_billing\_info\_menu() {

printf("\nBilling Information Menu:\n");

printf("1. Customer Billing\n");

printf("2. Interoperator Settlement Billing\n");

printf("Choice: ");

}

// Function to display customer billing menu

void display\_customer\_billing\_menu() {

printf("\nCustomer Billing Menu:\n");

printf("1. Search by MSISDN\n");

printf("2. Dump all customer billing data to CB.txt\n");

printf("Choice: ");

}

// Function to display interoperator billing menu

void display\_interoperator\_billing\_menu() {

printf("\nInteroperator Settlement Billing Menu:\n");

printf("1. Search by Brand Name/Operator ID\n");

printf("2. Dump all interoperator billing data to IOSB.txt\n");

printf("Choice: ");

}

// Function to send MSISDN search request for customer billing

void send\_search\_msisdn\_request() {

char msisdn[50];

printf("Enter MSISDN to search: ");

scanf("%s", msisdn);

// Send the MSISDN search request to the server

char request[100];

snprintf(request, sizeof(request), "SEARCH\_MSISDN|%s", msisdn);

send(sockfd, request, strlen(request), 0);

}

// Function to send search request for interoperator billing by operator

void send\_search\_operator\_request() {

char operator\_id[50];

printf("Enter Operator ID to search: ");

scanf("%s", operator\_id);

// Send the operator search request to the server

char request[100];

snprintf(request, sizeof(request), "SEARCH\_OPERATOR|%s", operator\_id);

send(sockfd, request, strlen(request), 0);

}

// Function to logout and close the connection

void logout() {

send(sockfd, "LOGOUT", 6, 0);

log\_message("INFO", "Logged out");

}

// Function to close the connection to the server

void close\_connection() {

close(sockfd);

log\_message("INFO", "Connection closed");

}

// Function to log messages with different severity levels

void log\_message(const char \*level, const char \*message) {

log\_file = fopen("client\_log.txt", "a");

if (log\_file == NULL) {

printf("Error opening log file\n");

return;

}

time\_t now;

time(&now);

fprintf(log\_file, "[%s] [%s] %s\n", ctime(&now), level, message);

fclose(log\_file);

}

// Main function

int main() {

int choice;

establish\_connection();

while(1)

{

display\_main\_menu();

scanf("%d", &choice);

switch (choice) {

case 1:

send\_signup\_request();

break;

case 2:

send\_login\_request();

break;

case 3:

close\_connection();

exit(0);

default:

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

break;

}

// After login, show post-login menu

display\_post\_login\_menu();

scanf("%d", &choice);

switch (choice) {

case 1:

send\_process\_cdr\_request();

break;

case 2:

display\_billing\_info\_menu();

scanf("%d", &choice);

if (choice == 1) {

display\_customer\_billing\_menu();

scanf("%d", &choice);

if (choice == 1) {

send\_search\_msisdn\_request();

} else if (choice == 2) {

// Logic for dumping to CB.txt

}

} else if (choice == 2) {

display\_interoperator\_billing\_menu();

scanf("%d", &choice);

if (choice == 1) {

send\_search\_operator\_request();

} else if (choice == 2) {

// Logic for dumping to IOSB.txt

}

}

break;

case 3:

logout();

break;

default:

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

break;

}

}

return 0;

}