**Online HealthCare System Using Client-Server**

**Introduction**

The Online Healthcare Services System is a client-server application designed to facilitate online diagnosis and treatment of patients. The system allows clients to send symptoms to a server and receive a diagnosis in response. With the increasing demand for online healthcare services, it is essential to develop a secure, efficient, and reliable platform that can handle a large number of patients and provide accurate diagnoses. The Online Healthcare Services System aims to address this need by providing a robust and scalable solution for conducting online diagnoses.

**Objective**

The primary objective of the Online Healthcare Services System is to provide a secure, efficient, and reliable platform for conducting online diagnoses. The system aims to ensure the accuracy and integrity of the diagnosis process, while also providing a user-friendly interface for patients to send symptoms and receive diagnoses. The specific objectives of the Online Healthcare Services System are:

* To develop a secure and reliable online diagnosis system that can handle a large number of patients
* To provide a user-friendly interface for patients to send symptoms and receive diagnoses
* To ensure the accuracy and integrity of the diagnosis process
* To provide real-time updates of diagnosis results
* To handle multiple client connections concurrently using single-threaded server.

**System Requirements**

**Hardware Requirements:**

* Operating System: Linux/Unix-based
* Processor: Multi-core processor
* Memory: 4 GB RAM or more
* Storage: 10 GB or more

**Software Requirements:**

* Programming Language: C++
* Libraries: sys/socket.h, netinet/in.h, arpa/inet.h, unistd.h
* Compiler: GCC

**Functionality**

* **Symptom Sending:** Clients can send symptoms to the server.
* **Diagnosis Retrieval:** Clients can retrieve the diagnosis results from the server.
* **Real-time Updates:** The server updates the diagnosis results in real-time as new symptoms are received.
* **Concurrency:** The server handles multiple client connections concurrently using single-threaded server.

**Modules (Components)**

**Server Components:**

* **Socket Programming:** The server creates a socket and binds it to a specific address and port. It listens for incoming client connections and accepts them.
* **Single-Threaded Server:** The server uses a single thread to handle all incoming client connections.
* **Diagnosis Generation:** The server generates a diagnosis based on the symptoms received from the client.
* **Diagnosis Results:** The server maintains a map to store the diagnosis results.

**Client Components:**

* **Socket Programming:** The client creates a socket and connects to the server's socket. It sends requests to send symptoms and retrieve diagnosis results.
* **Symptom Sending:** The client sends a request to send symptoms to the server.
* **Diagnosis Retrieval:** The client retrieves the diagnosis results from the server.

**Implementation**

Here's a high-level overview of how these mechanisms are implemented in your project:

**Socket programming:**

* The server creates a socket and binds it to a specific address and port.
* Clients create sockets and connect to the server's socket.
* Both the server and clients use send () and receive () to exchange data.

**Single-Threaded Server:**

* The server uses a single thread to handle all incoming client connections.
* The server executes the handle client () function for each client connection.

**Diagnosis Generation:**

* The server generates a diagnosis based on the symptoms received from the client.
* The server updates the diagnosis results map accordingly.

**Testing:**

The Online Healthcare Services System has been tested using the following scenarios:

* **Single Client:** A single client sends symptoms and retrieves diagnosis results.
* **Multiple Clients:** Multiple clients send symptoms and retrieve diagnosis results concurrently.
* **Symptom Sending:** Clients send different symptoms and retrieve the updated diagnosis results.

**Execution / Source Code:**

**Server.cpp Code:**

#include <iostream>

#include <string>

#include <sys/socket.h>

#include <netinet/in.h>

#include <unistd.h>

#define PORT 8080

int main() {

int server\_fd, new\_socket;

struct sockaddr\_in address;

socklen\_t addrlen = sizeof(address);

char buffer[1024] = {0};

std::string message = "Welcome to online healthcare services! Please enter your symptoms:\n1. Fever\n2. Cold\n3. Cough\n";

// Create socket

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

std::cerr << "Socket creation failed" << std::endl;

return 1;

}

// Set up address

address.sin\_family = AF\_INET;

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

address.sin\_port = htons(PORT);

// Bind socket to address

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

std::cerr << "Bind failed" << std::endl;

return 1;

}

// Listen for incoming connections

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

std::cerr << "Listen failed" << std::endl;

return 1;

}

std::cout << "Server connected. Waiting for patients..." << std::endl;

while (true) {

// Accept incoming connection

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

std::cerr << "Accept failed" << std::endl;

continue;

}

std::cout << "Patient connected." << std::endl;

// Send welcome message to client

send(new\_socket, message.c\_str(), message.length(), 0);

// Receive symptoms from client

int symptom;

read(new\_socket, &symptom, sizeof(int));

// Generate diagnosis based on symptoms

std::string diagnosis;

if (symptom == 1) {

diagnosis = "You have a Fever. Take the following medicines:\n1. Paracetamol\n2. Ibuprofen\n3. Aspirin";

} else if (symptom == 2) {

diagnosis = "You have a Cold. Take the following medicines:\n1. Antihistamines\n2. Decongestants\n3. Expectorants";

} else if (symptom == 3) {

diagnosis = "You have a Cough. Take the following medicines:\n1. Cough Syrup\n2. Expectorants\n3. Bronchodilators";

} else {

diagnosis = "Unknown symptom.";

}

// Send diagnosis to client

send(new\_socket, diagnosis.c\_str(), diagnosis.length(), 0);

// Close socket

close(new\_socket);

}

return 0;

}

**Client.cpp Code:**

#include <iostream>

#include <string>

#include <sys/socket.h>

#include <arpa/inet.h>

#include <unistd.h>

#define PORT 8080

int main() {

int sock = 0;

struct sockaddr\_in serv\_addr;

// Create socket

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

std::cerr << "Socket creation failed" << std::endl;

return 1;

}

// Set up address

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

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

// Convert IPv4 address from text to binary form

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

std::cerr << "Invalid address/ Address not supported" << std::endl;

return 1;

}

// Connect to server

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

std::cerr << "Connection failed" << std::endl;

return 1;

}

// Receive welcome message from server

char buffer[1024] = {0};

read(sock, buffer, 1024);

std::cout << buffer << std::endl;

// Send symptoms to server

int symptom;

std::cout << "Enter your symptom (1 for Fever, 2 for Cold, 3 for Cough): ";

std::cin >> symptom;

send(sock, &symptom, sizeof(int), 0);

// Receive diagnosis from server

read(sock, buffer, 1024);

std::cout << "Diagnosis: " << buffer << std::endl;

// Close socket

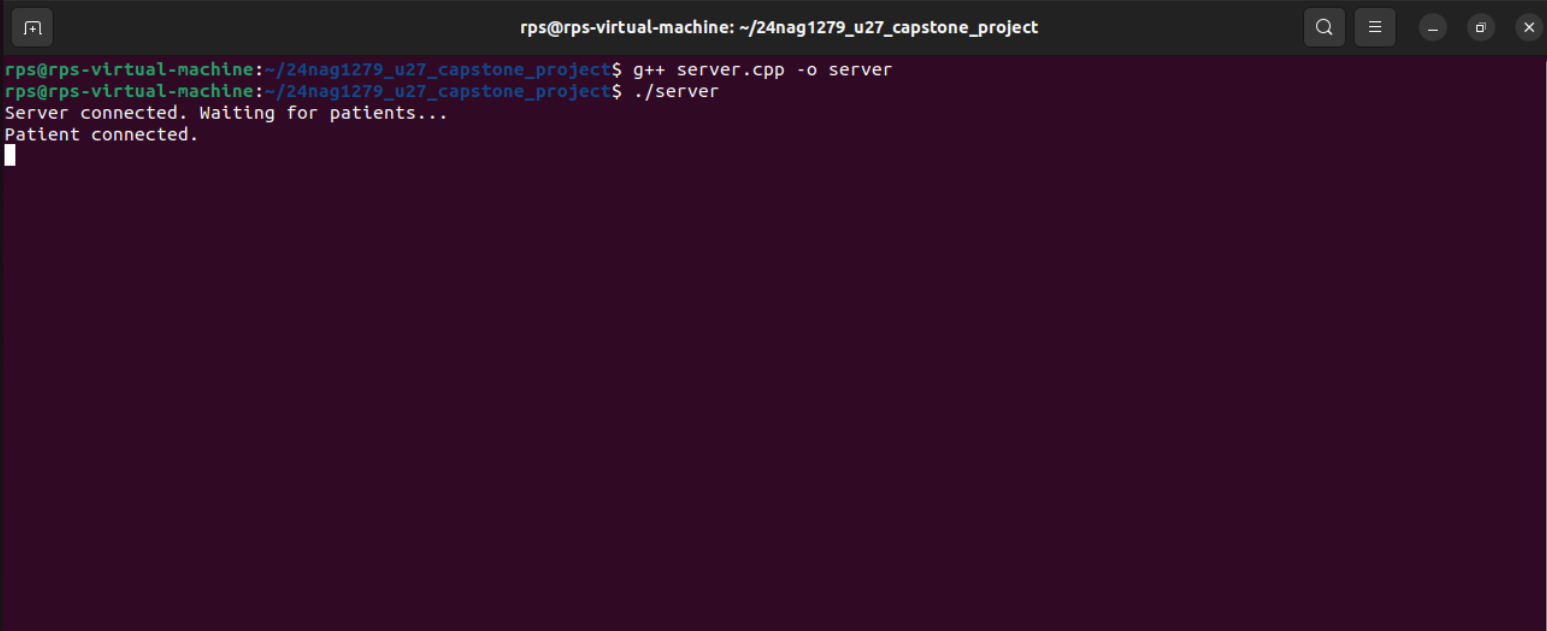
close(sock);

return 0;

}

**OUTPUTS:**

**Server Code Output:**

****

**Client Code Output:**



**Conclusion**

The Online Healthcare Services System is a secure, efficient, and reliable platform for conducting online diagnoses. It provides a user-friendly interface for patients to send symptoms and receive diagnoses, and ensures the accuracy and integrity of the diagnosis process. The system has been implemented in C++ using socket programming and single-threaded server, and has been tested using various scenarios and has proven to be robust and scalable.