**Client Android App:**

**MainActivity.java**

package com.example.csio;

import android.graphics.drawable.Drawable;

import android.os.Bundle;

import android.support.v7.app.AppCompatActivity;

import android.text.Editable;

import android.text.TextWatcher;

import android.view.View;

import android.widget.Button;

import android.widget.EditText;

import android.widget.Toast;

import java.util.regex.Matcher;

import java.util.regex.Pattern;

public class MainActivity extends AppCompatActivity {

private boolean isHVACOn = false;

private boolean isAlarmOn = false;

private boolean isDoorOn = false;

private boolean isStart = false;

//buttons

Button HVAC;

Button alarm;

Button door;

Button ipButton;

//variable

private int icon;

private String text;

//text view

EditText ipEditText;

EditText portEditText;

//instance to client

PrimeRun ad;

private int valueToPass;

//ip validator

private static final Pattern IP\_ADDRESS =

Pattern.compile("^((25[0-5]|2[0-4][0-9]|[0-1][0-9]{2}|[1-9][0-9]|[0-9])\\.){0,3}"+

"((25[0-5]|2[0-4][0-9]|[0-1][0-9]{2}|[1-9][0-9]|[0-9])){0,1}$");

/\*

private TextView HVACView;

private TextView alarmView;

private TextView doorView;

private TextView lightView;

\*/

public MainActivity()

{

isHVACOn = isDoorOn = isAlarmOn = false;

icon = 0;

}

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_main);

//assign id and initialize button

HVAC = (Button) findViewById(R.id.HVAC);

door = (Button) findViewById(R.id.door);

alarm = (Button) findViewById(R.id.alarm);

ipButton = (Button) findViewById(R.id.ipButton);

ipEditText = (EditText) findViewById(R.id.ip);

portEditText = (EditText) findViewById(R.id.port);

/\*

//assign id and initialize textView

lightView = (TextView) findViewById(R.id.light);

HVACView = (TextView) findViewById(R.id.HVAC);

doorView = (TextView) findViewById(R.id.door);

alarmView = (TextView) findViewById(R.id.alarm);

\*/

ipEditText.addTextChangedListener(new TextWatcher() {

@Override

public void onTextChanged(CharSequence s, int start, int before, int count) {

}

@Override

public void beforeTextChanged(CharSequence s,int start,int count,int after) {

}

private String mPreviousText = "";

@Override

public void afterTextChanged(Editable s) {

if(IP\_ADDRESS.matcher(s).matches()) {

mPreviousText = s.toString();

} else {

Toast.makeText(getApplicationContext(),"Your IP is InCorrect, Try adding . e.g 192.168.1.14",Toast.LENGTH\_SHORT).show();

s.replace(0, s.length(), mPreviousText);

}

}

});

ipButton.setOnClickListener(new View.OnClickListener()

{

public void onClick(View v)

{

Matcher matcher = IP\_ADDRESS.matcher(ipEditText.getText());

String ip = ipEditText.getText().toString();

String portNum = portEditText.getText().toString();

PrimeRun con = new PrimeRun("connect");

if(ip.length() == 0)

{

Toast.makeText(getApplicationContext(),"Assign IP First",Toast.LENGTH\_SHORT).show();

isStart=false;

}

else

{

if(matcher.matches()) {

PrimeRun.setIp(ip);

Toast.makeText(getApplicationContext(),"Your Ip \t"+ip+"\t is Assigned",Toast.LENGTH\_SHORT).show();

isStart=true;

if(portNum.length()==0) {

PrimeRun.setPortNumber("4444");

isStart=true;

}

else

{

PrimeRun.setPortNumber(portNum);

Toast.makeText(getApplicationContext(),"Your Port Number \t"+portNum+"\t is Assigned",Toast.LENGTH\_SHORT).show();

isStart = true;

}

}

else

{

Toast.makeText(getApplicationContext(),"Your IP is InCorrect, Try adding . e.g 192.168.1.14",Toast.LENGTH\_LONG).show();

isStart =false;

}

if(isStart)

{

}

}

}

}

);

}

public void HVACEvent(View v)

{

//for HVAC on the value will be 1 and for HVAC off the value will be 2

if(ipEditText.getText().length() == 0)

{

Toast.makeText(getApplicationContext(),"Assign IP First",Toast.LENGTH\_LONG).show();

}

else {

if (isHVACOn) {

isHVACOn = false;

icon = R.drawable.fan\_on;

text = "HVAC On";

valueToPass=1;

} else {

isHVACOn = true;

icon = R.drawable.fan\_off;

text = "HVAC Off";

valueToPass = 2;

}

Drawable top = getResources().getDrawable(icon);

HVAC.setCompoundDrawablesWithIntrinsicBounds(null, top, null, null);

ad = new PrimeRun("sendData");

ad.setDataToTransfer(valueToPass);

ad.start();

//send data to server

HVAC.setText(text);

}

}

public void alarmEvent(View v)

{

//for alaram on the value will be 3 and for alarm off the value will be 4

if(ipEditText.getText().length() == 0)

{

Toast.makeText(getApplicationContext(),"Assign IP First",Toast.LENGTH\_LONG).show();

}

else {

if (isAlarmOn) {

isAlarmOn = false;

text = "Alarm On";

icon = R.drawable.alarm\_on;

valueToPass = 3;

} else {

isAlarmOn = true;

text = "Alarm Off";

icon = R.drawable.alarm\_off;

valueToPass = 4;

}

Drawable top = getResources().getDrawable(icon);

alarm.setCompoundDrawablesWithIntrinsicBounds(null, top, null, null);

ad = new PrimeRun("sendData");

ad.setDataToTransfer(getValueToPass());

ad.start();

//send data to server

alarm.setText(text);

}

}

public void doorEvent(View v)

{

//for door open the value will be 5 and for close the value will be 6

if(ipEditText.getText().length() == 0)

{

Toast.makeText(getApplicationContext(),"Assign IP First",Toast.LENGTH\_LONG).show();

}

else {

if (isDoorOn) {

isDoorOn = false;

text = "Door Open";

icon = R.drawable.door\_open;

valueToPass = 5;

} else {

isDoorOn = true;

text = "Door Close";

icon = R.drawable.door\_close;

valueToPass =6;

}

Drawable top = getResources().getDrawable(icon);

door.setCompoundDrawablesWithIntrinsicBounds(null, top, null, null);

ad = new PrimeRun("sendData");

ad.setDataToTransfer(getValueToPass());

ad.start();

//send data to server

door.setText(text);

}

}

public void changePort(View v)

{

}

public int getValueToPass() {

return valueToPass;

}

public void setValueToPass(int valueToPass) {

this.valueToPass = valueToPass;

}

}

**PrimeRun.java**

package com.example.csio;

import android.app.Activity;

import android.content.Context;

import android.util.Log;

import android.widget.Toast;

import java.io.BufferedWriter;

import java.io.IOException;

import java.io.OutputStreamWriter;

import java.net.Socket;

public class PrimeRun extends Activity implements Runnable {

private static String IP;

private static String portNumber ;

private int port;

private static final String debugString = "debug";

private Socket socket = null;

private Thread t = null;

private String threadName;

private Context context;

public int dataToTransfer;

private boolean threadComplete;

private boolean connection;

private boolean threadSuspended;

BufferedWriter bw;

public PrimeRun()

{

}

public PrimeRun(String thName)

{

this.threadName = thName;

}

public void start()

{

if(t==null)

{

t = new Thread(this,threadName);

t.start();

}

}

@Override

public void run() {

String debugString = "debug";

Socket socket = null;

t= Thread.currentThread();

try

{

t.sleep(2000);

synchronized(this) {

while (threadSuspended)

wait();

}

} catch (InterruptedException e){

}

if (t.toString().contains("sendData"))

{

if(connection)

{

synchronized (t) {

t.setName("connect");

t.start();

}

}

if (getIP().equals("") && getPortNumber().equals("")) {

Toast.makeText(getApplicationContext(), "Your IP or Port Number has issues", Toast.LENGTH\_LONG).show();

} else {

try {

Log.i(debugString, "Attempting to connect to server");

port = Integer.parseInt(portNumber);

socket = new Socket(IP, port);

connection = true;

Log.i(debugString, "Connection Establish");

bw = new BufferedWriter(new OutputStreamWriter(socket.getOutputStream()));

if (dataToTransfer == 1) {

bw.write("HVAC On\t" + dataToTransfer +"\n");

} else if (dataToTransfer == 2) {

bw.write("HVAC OFF\t" + dataToTransfer+"\n");

} else if (dataToTransfer == 3) {

bw.write("Alarm On\t" + dataToTransfer+"\n");

} else if (dataToTransfer == 4) {

bw.write("Alarm Off\t" + dataToTransfer+"\n");

} else if (dataToTransfer == 5) {

bw.write("Door open\t" + dataToTransfer+"\n");

} else if (dataToTransfer == 6) {

bw.write("Door close\t" + dataToTransfer+"\n");

}

bw.newLine();

bw.flush();

} catch (IOException e) {

System.out.println(e);

connection = false;

} finally {

threadComplete = true;

}

}

}

else if (t.toString().contains("connect"))

{

if (connection) {

try {

bw = new BufferedWriter(new OutputStreamWriter(socket.getOutputStream()));

if (dataToTransfer == 1) {

bw.write("HVAC On\t" + dataToTransfer +"\n");

} else if (dataToTransfer == 2) {

bw.write("HVAC OFF\t" + dataToTransfer+"\n");

} else if (dataToTransfer == 3) {

bw.write("Alarm On\t" + dataToTransfer+"\n");

} else if (dataToTransfer == 4) {

bw.write("Alarm Off\t" + dataToTransfer+"\n");

} else if (dataToTransfer == 5) {

bw.write("Door open\t" + dataToTransfer+"\n");

} else if (dataToTransfer == 6) {

bw.write("Door close\t" + dataToTransfer+"\n");

}

bw.newLine();

bw.flush();

} catch (IOException e) {

e.printStackTrace();

}

}

else

{

if (getIP().equals("") && getPortNumber().equals("")) {

Toast.makeText(getApplicationContext(), "Your IP or Port Number has issues", Toast.LENGTH\_LONG).show();

} else {

try {

Log.i(debugString, "Attempting to connect to server");

// Toast.makeText(null,"Attempting to connect",Toast.LENGTH\_SHORT).show();

port = Integer.parseInt(portNumber);

socket = new Socket(IP, port);

connection = true;

bw = new BufferedWriter(new OutputStreamWriter(socket.getOutputStream()));

bw.write("\t" + dataToTransfer);

bw.newLine();

bw.flush();

} catch (IOException ee) {

System.out.println(ee);

connection = false;

} finally {

threadComplete = true;

}

}

//send message to server

}

}

}

public String getThreadName() {

return threadName;

}

public int getDataToTransfer() {

return dataToTransfer;

}

public void setDataToTransfer(int dataToTransfer) {

this.dataToTransfer = dataToTransfer;

}

public static String getIP() {

return IP;

}

public static void setIp(String IP) {

PrimeRun.IP = IP;

}

public static String getPortNumber() {

return portNumber;

}

public static void setPortNumber(String portNumber) {

PrimeRun.portNumber = portNumber;

}

}

**Activity\_main.xml**

<?xml version="1.0" encoding="utf-8"?>

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"

xmlns:app="http://schemas.android.com/apk/res-auto"

xmlns:tools="http://schemas.android.com/tools"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

tools:context="com.example.csio.MainActivity"

android:orientation="vertical"

android:layout\_weight="1">

<RelativeLayout

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

android:layout\_weight="2">

<EditText

android:id="@+id/ip"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:hint="@string/ipAddress"

android:inputType="phone"/>

<EditText

android:id="@+id/port"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:layout\_marginTop="@dimen/small"

android:layout\_below="@+id/ip"

android:hint="@string/portNum"

android:inputType="phone"/>

<Button

android:id="@+id/ipButton"

android:layout\_width="@dimen/socket\_width"

android:layout\_height="wrap\_content"

android:layout\_below="@+id/port"

android:layout\_marginTop="@dimen/small"

android:layout\_centerInParent="true"

android:text="@string/Socket" />

</RelativeLayout>

<RelativeLayout

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

android:layout\_weight="1"

android:clickable="false">

<Button

android:id="@+id/HVAC"

android:layout\_width="@dimen/B\_width"

android:layout\_height="@dimen/B\_height"

android:layout\_alignParentLeft="true"

android:layout\_alignParentStart="true"

android:layout\_marginLeft="@dimen/top"

android:layout\_marginStart="@dimen/top"

android:layout\_marginTop="@dimen/topMore"

android:drawableTop="@drawable/fan\_off"

android:duplicateParentState="true"

android:onClick="HVACEvent"

android:text="@string/fan"

android:textColor="@color/textColor"

/>

<Button

android:id="@+id/door"

android:layout\_width="@dimen/B\_width"

android:layout\_height="@dimen/B\_height"

android:layout\_below="@+id/HVAC"

android:layout\_centerHorizontal="true"

android:layout\_marginLeft="@dimen/top"

android:layout\_marginStart="@dimen/top"

android:layout\_marginTop="@dimen/top"

android:drawableTop="@drawable/door\_close"

android:duplicateParentState="true"

android:onClick="doorEvent"

android:text="@string/door"

/>

<Button

android:id="@+id/alarm"

android:layout\_width="@dimen/B\_width"

android:layout\_height="@dimen/B\_height"

android:layout\_marginLeft="28dp"

android:layout\_marginStart="28dp"

android:drawableTop="@drawable/alarm\_off"

android:duplicateParentState="true"

android:onClick="alarmEvent"

android:text="@string/alarm"

android:textColor="@color/textColor"

android:layout\_above="@+id/door"

android:layout\_toRightOf="@+id/HVAC"

android:layout\_toEndOf="@+id/HVAC" />

</RelativeLayout>

</LinearLayout>

**AndriodManifest.xml**

<?xml version="1.0" encoding="utf-8"?>

<manifest xmlns:android="http://schemas.android.com/apk/res/android"

package="com.example.csio">

<uses-permission android:name="android.permission.INTERNET"></uses-permission>

<application

android:allowBackup="true"

android:icon="@mipmap/ic\_launcher"

android:label="@string/app\_name"

android:roundIcon="@mipmap/ic\_launcher\_round"

android:supportsRtl="true"

android:theme="@style/AppTheme">

<activity android:name=".MainActivity">

<intent-filter>

<action android:name="android.intent.action.MAIN" />

<category android:name="android.intent.category.LAUNCHER" />

</intent-filter>

</activity>

<activity android:name=".PrimeRun"></activity>

</application>

</manifest>

**String.xml**

<resources>

<string name="app\_name">So Client</string>

<string name="Socket">Set Values</string>

<string name="fan">HVAC Off</string>

<string name="door">Door Close</string>

<string name="alarm">Alarm Off</string>

<string name="change\_port">Change Port</string>

<string name="ipAddress">IP Address</string>

<string name="portNum">Default Port: 4444,Click to Change</string>

</resources>

**Dimen.xml**

<?xml version="1.0" encoding="utf-8"?>

<resources>

<dimen name="B\_width">130sp</dimen>

<dimen name="B\_height">130sp</dimen>

<dimen name="search\_width">200sp</dimen>

<dimen name="socket\_width">120sp</dimen>

<dimen name="top">20sp</dimen>

<dimen name="topMore">30sp</dimen>

<dimen name="small">8sp</dimen>

</resources>

**Colors.xml**

<?xml version="1.0" encoding="utf-8"?>

<resources>

<color name="colorPrimary">#3F51B5</color>

<color name="colorPrimaryDark">#303F9F</color>

<color name="colorAccent">#FF4081</color>

<color name="textColor">#ffffff</color>

</resources>

**Java Server**

**JavaServerGui.java**

/\*

This class is responsible for the creation of Gui of java server

\*/

import java.awt.BorderLayout;

import java.awt.Color;

import java.awt.event.ActionEvent;

import javax.swing.JButton;

import javax.swing.JFrame;

import javax.swing.JOptionPane;

import javax.swing.JPanel;

import javax.swing.JTextArea;

public class JavaServerGui extends JFrame{

public static JTextArea clientMessage = new JTextArea();

public JavaServerGui()

{

initUI();

}

private void initUI()

{

setBounds(0,0,600,400);

setDefaultCloseOperation(EXIT\_ON\_CLOSE);

JPanel panel = new JPanel();

panel.setLayout(new BorderLayout());

panel.setForeground(Color.red);

add(panel);

JButton startServer = new JButton("Start Server");

JButton exit = new JButton("Stop and exit");

JButton changePort = new JButton("Change Port");

clientMessage.append("Message From Client\n");

clientMessage.setFont(clientMessage.getFont().deriveFont(13f));

clientMessage.setForeground(Color.white);

clientMessage.setBackground(Color.GRAY);

panel.add(startServer,BorderLayout.WEST);

panel.add(exit,BorderLayout.EAST);

panel.add(changePort,BorderLayout.NORTH);

panel.add(clientMessage,BorderLayout.CENTER);

setVisible(true);

startServer.addActionListener((ActionEvent e) -> {

Threads t = new Threads("start\_server");

t.start();

});

exit.addActionListener((ActionEvent e) -> {

Threads t = new Threads("stop\_server");

t.start();

});

changePort.addActionListener((ActionEvent e) -> {

Threads t = new Threads("change\_port");

t.start();

});

}

public void printTextField(String text)

{

clientMessage.append(text);

}

public static void main(String[] args) {

JavaServerGui j = new JavaServerGui();

}

}

**JavaServer.java**

/\*

The javaServer class will act as a server and will recieve message from the JavaServer

\*/

import java.io.IOException;

import java.net.ServerSocket;

import javax.swing.JOptionPane;

public class JavaServer {

private static int portNumber = 4444;

private String message;

int number = 0;

ServerSocket serverSocket;

public JavaServer()

{

try

{

System.out.println("server Starting at port Number "+ portNumber);

// JOptionPane.showMessageDialog(rootPane, "server Starting at port Number \t"+ portNumber);

// serverSocket = new ServerSocket(portNumber);

ServerSocket serverSocket = new ServerSocket(portNumber);

//client connecting

System.out.println("Waiting for the client to connect");

// JOptionPane.showMessageDialog(rootPane, "Waiting for the client to connect");

try

{

while(true)

{

new Threads(serverSocket.accept(),number).start();

}

}

finally

{

serverSocket.close();

}

// JOptionPane.showMessageDialog(rootPane, "Client Connected");

//recieve message from the client

// JOptionPane.showMessageDialog(rootPane, "Server has ended");

}

catch(IOException io)

{

System.out.println(io);

}

System.out.println("Server has ended");

System.exit(0);

}

public JavaServer(int portNum)

{

portNumber = portNum;

try

{

System.out.println("server Starting at port Number "+ portNumber);

// JOptionPane.showMessageDialog(rootPane, "server Starting at port Number \t"+ portNumber);

// serverSocket = new ServerSocket(portNumber);

ServerSocket serverSocket = new ServerSocket(portNumber);

//client connecting

System.out.println("Waiting for the client to connect");

// JOptionPane.showMessageDialog(rootPane, "Waiting for the client to connect");

try

{

while(true)

{

new Threads(serverSocket.accept(),number).start();

}

}

finally

{

serverSocket.close();

}

// JOptionPane.showMessageDialog(rootPane, "Client Connected");

//recieve message from the client

// JOptionPane.showMessageDialog(rootPane, "Server has ended");

}

catch(IOException io)

{

System.out.println(io);

JOptionPane.showMessageDialog(null,io);

}

System.out.println("Server has ended");

}

/\*\*

\* @return the portNumber

\*/

public int getPortNumber() {

return portNumber;

}

/\*\*

\* @param portNumber the portNumber to set

\*/

public void setPortNumber(int portNumber) {

this.portNumber = portNumber;

}

/\*\*

\* @return the message

\*/

public String getMessage() {

return message;

}

/\*\*

\* @param message the message to set

\*/

public void setMessage(String message) {

this.message = message;

}

}

**Threads.java**

/\*

The start server will be run in thread so that all other option still accessible to user like stoping server

\*/

import java.io.BufferedReader;

import java.io.IOException;

import java.io.InputStreamReader;

import java.net.Socket;

import static java.rmi.server.LogStream.log;

import javax.swing.JOptionPane;

public class Threads implements Runnable {

private String threadName;

private Thread t;

private int no;

private Socket socket;

private String message;

public Threads()

{

}

public void start()

{

if(t == null)

{

t = new Thread(this,threadName);

t.start();

}

}

public Threads(String name)

{

threadName = name;

}

public Threads(Socket socket, int clientNumber) {

this.socket = socket;

this.no = clientNumber;

this.threadName = Integer.toString(clientNumber);

}

@Override

public void run()

{

if(t.toString().contains("stop\_server"))

{

System.exit(0);

}

else if(t.toString().contains("change\_port"))

{

String portNumber;

portNumber = JOptionPane.showInputDialog( null, "Enter the value to change the defult port " ,JOptionPane.OK\_CANCEL\_OPTION);

if (portNumber!=null)

{

if(JOptionPane.OK\_OPTION ==JOptionPane.OK\_OPTION)

{

JavaServer jj = new JavaServer(Integer.parseInt(portNumber));

JOptionPane.showMessageDialog(null, "Port Number change.Start Server again by clicking start server" );

}

else

{

JOptionPane.showMessageDialog(null, "No new Port assign,Listening on default Port" );

}

}

else

{

JOptionPane.showMessageDialog(null, "No new Port assign,Listening on default Port" );

}

}

else if(t.toString().contains("start\_server"))

{

JavaServer jj = new JavaServer();

}

try

{

BufferedReader br = new BufferedReader(new InputStreamReader(socket.getInputStream()));

while(true)

{

message = br.readLine();

if(message == null||message.equals("."))

{

break;

}

System.out.println(message);

JavaServerGui.clientMessage.append("\n"+message);

}

}

catch(IOException e)

{

log("Error handling client# " + no + ": " + e);

}

}

public String getThreadName() {

return threadName;

}

}