VELAGAPUDI RAMAKRISHNA SIDDHARTHA ENGINEERING COLLEGE

(AUTONOMOUS - AFFILIATED TO JNTU-K, KAKINADA)

Approved by AICTE & Accredited by NBA

KANURU, VIJAYAWADA-7

ACADEMIC YEAR

(2022-2023)

20IT5301 – Computer Networks

Project

Submitted To:

Dr.N.Neelima

Assistant professor

Submitted By:

A. Charan (208W1A1268)

Md.Rizwanullah (208W1A1291)

Dept of IT **N.Ajay kumar varma**
(208W1A12A1)

AIM:

Developing an Chat Application using Bluetooth

PROBLEM:

Our primary goal is to create a chat with the people who are connected to Bluetooth of our device

DESCRIPTION:

Two devices are able to use Bluetooth to set up a secure peer-to-peer connection. It's the same technology when you use Bluetooth to transfer music or images from one device to another. The maximum range between devices is about 300 feet, or 100 meters.

PROCEDURE:

Step 1. Setting up the Project

Let's begin with setting up the project.

Step 1.1. Download Android Studio

- You can download Android studio for your platform by clicking on this link. Then install the latest sdk's for better development pace

Step 1.2. Create a Empty project

- Open Android Studio an select New project in the pop up displayed
- Then a window containing different projects will be displayed then select the Empty Project

Step 1.3. Get Bluetooth Service

- We need the Android Bluetooth service for this tutorial to work. In order to use Bluetooth service, declare **BLUETOOTH** permission in manifest file.

- Add the following code in android manifest.xml file <uses-permission android:name="android.permission.BLUETOOTH" />
- Now to initiate device discovery and to access Bluetooth setting declare **BLUETOOTH_ADMIN** permission.
- Add the following code in android manifest.xml file

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

Step 2. Bluetooth Adapter Class

Now to check whether Bluetooth is supported on device or not, we use object of BluetoothAdapter class.

BluetoothAdapter bluetoothAdapter = BluetoothAdapter.getDefaultAdapter();

If this method returns null, then it means that Bluetooth is not supported on the device and so we will close the application

Step3: isEnable() method:

To check Bluetooth is enabled or not, we will use **isEnabled()** method on object of BluetoothAdapter class.

If Bluetooth is disabled then we request the user to enable it. And we perform this action by calling **startActivityForResult()** with **REQUEST_ENABLE_BT** action. This will open dialog to enable Bluetooth on the devic**e.**

If the user clicks Allow then the **onActivityResult()** method receives RESULT_OK and if the user clicks Deny (or due to internal problem in device), **RESULT_CANCELED** is received. If returned value RESULT_OK is received then we will initiate the chat service.

Step4: Discover Bluetooth

Now in android, device is not discoverable by default. To make device discoverable, call **startActivityForResult()** with BluetoothAdapter.**ACTION_REQUEST_DISCOVERABLE** action. By default device is discoverable for 120 seconds. To set discoverable duration, add **EXTRA_DISCOVERABLE_DURATION** in intent extra. The maximum value for duration is 360 seconds.

Code:

```
Intent discoverableIntent = new Intent(

BluetoothAdapter.ACTION_REQUEST_DISCOVERABLE);

discoverableIntent.putExtra(

BluetoothAdapter.EXTRA_DISCOVERABLE_DURATION, 300);

startActivity(discoverableIntent);
```

Step5: Bluetooth Connection

To start the chat, we first need to establish connection with the desired device. And before starting scanning for available devices, we usually get paired devices first in the list.

Code:

```
BluetoothAdapter bluetoothAdapter = BluetoothAdapter.getDefaultAdapter();

Set<BluetoothDevice> pairedDevices = bluetoothAdapter.getBondedDevices();
```

Above code will return a set of BluetoothDevice objects. The object of BluetoothDevice class gives required information about remote device which is used to establish connection.

To start scanning, call the **startDiscovery()** method of BluetoothAdapter class. The activity which starts scanning must register receiver with BluetoothDevice.**ACTION_FOUND** action. After completing discovery, system will broadcast BluetoothDevice.**ACTION_FOUND** intent. This Intent contains extra fields **EXTRA_DEVICE** and **EXTRA_CLASS,** representing a BluetoothDevice and a BluetoothClass, respectively.

Code:

```
private final BroadcastReceiver discoveryFinishReceiver = new BroadcastReceiver() {

@Override

public void onReceive(Context context, Intent intent) {

String action = intent.getAction();



if (BluetoothDevice.ACTION_FOUND.equals(action)) {



BluetoothDevice device = intent

.getParcelableExtra(BluetoothDevice.EXTRA_DEVICE);



if (device.getBondState() != BluetoothDevice.BOND_BONDED) {

newDevicesArrayAdapter.add(device.getName() + "\n"+ device.getAddress());
```

```
}

} else if (BluetoothAdapter.ACTION_DISCOVERY_FINISHED

.equals(action)) {

setProgressBarIndeterminateVisibility(false);

setTitle(R.string.select_device);

if (newDevicesArrayAdapter.getCount() == 0) {

String noDevices = getResources().getText(

R.string.none_found).toString();

newDevicesArrayAdapter.add(noDevices);

}

}

}

};
```

Step 6: Pairing Devices

To connect two devices, we must implement server side and client side mechanism. One device shall open the server socket and another should initiate the connection. Both are connected when BluetoothSocket is connected on the same RFCOMM channel. During connection procedure android framework automatically shows pairing dialog.

Connection as Server:

Make object of BluetoothServerSocket by calling
the listenUsingRfcommWithServiceRecord().

Listening for connection requests by calling accept().

Release server socket by calling close().

Code:

```
private class AcceptThread extends Thread {

        private final BluetoothServerSocket serverSocket;

    private String socketType;



    public AcceptThread(boolean secure) {

       BluetoothServerSocket tmp = null;

       socketType = secure ? "Secure" : "Insecure";



            try {

          if (secure) {

             tmp = bluetoothAdapter.listenUsingRfcommWithServiceRecord(

                 NAME_SECURE, MY_UUID_SECURE);

          } else {

             tmp = bluetoothAdapter
```

```
                .listenUsingInsecureRfcommWithServiceRecord(
                        NAME_INSECURE, MY_UUID_INSECURE);
            }
        } catch (IOException e) {
        }
        serverSocket = tmp;
    }

    public void run() {
        setName("AcceptThread" + socketType);

        BluetoothSocket socket = null;

            while (state != STATE_CONNECTED) {
            try {

                socket = serverSocket.accept();
            } catch (IOException e) {
                break;
```

```
}



if (socket != null) {
    synchronized (ChatService.this) {
        switch (state) {
        case STATE_LISTEN:
        case STATE_CONNECTING:
                                connected(socket, socket.getRemoteDevice(),
                socketType);
            break;
        case STATE_NONE:
        case STATE_CONNECTED:


            try {
                socket.close();
            } catch (IOException e) {
            }
            break;
```

```
                }

            }

        }

    }

}



    public void cancel() {

        try {

            serverSocket.close();

        } catch (IOException e) {

        }

    }

}
```

Step 7: Connection as Client

1. Create object of BluetoothSocket by calling createRfcommSocketToServiceRecord(UUID) on BluetoothDevice object.
2. Initiate connection by calling connect().

Code:

```
private class ConnectThread extends Thread {
    private final BluetoothSocket socket;
    private final BluetoothDevice device;
    private String socketType;

    public ConnectThread(BluetoothDevice device, boolean secure) {
        this.device = device;
        BluetoothSocket tmp = null;
```

```
        socketType = secure ? "Secure" : "Insecure";



        try {
            if (secure) {
                tmp = device
                        .createRfcommSocketToServiceRecord(MY_UUID_SECURE);
            } else {
                tmp = device

.createInsecureRfcommSocketToServiceRecord(MY_UUID_INSECURE);
            }
        } catch (IOException e) {
        }
        socket = tmp;
    }

    public void run() {
        setName("ConnectThread" + socketType);

                bluetoothAdapter.cancelDiscovery();

                try {

            socket.connect();
        } catch (IOException e) {

            try {
                socket.close();
            } catch (IOException e2) {
            }
            connectionFailed();
            return;
        }


        synchronized (ChatService.this) {
            connectThread = null;
        }

                connected(socket, device, socketType);
    }

    public void cancel() {
        try {
            socket.close();
        } catch (IOException e) {
        }
    }
```

```
    }
```

Step8: Read And Write Data

1.After establishing connection successfully, each device has connected BluetoothSocket.

2. Now one can Read and write data to the streams using read(byte[]) and write(byte[]).

Code:

```
private class ConnectedThread extends Thread {
        private final BluetoothSocket bluetoothSocket;
        private final InputStream inputStream;
        private final OutputStream outputStream;

        public ConnectedThread(BluetoothSocket socket, String socketType) {
            this.bluetoothSocket = socket;
            InputStream tmpIn = null;
            OutputStream tmpOut = null;

                        try {
                tmpIn = socket.getInputStream();
                tmpOut = socket.getOutputStream();
            } catch (IOException e) {
            }

            inputStream = tmpIn;
            outputStream = tmpOut;
        }

        public void run() {
            byte[] buffer = new byte[1024];
            int bytes;

                        while (true) {
                try {

                    bytes = inputStream.read(buffer);

                                        handler.obtainMessage(MainActivity.MESSAGE_READ, bytes, -1,
                            buffer).sendToTarget();
                } catch (IOException e) {
                    connectionLost();
                                        ChatService.this.start();
                    break;
                }
            }
        }
```

```
    public void write(byte[] buffer) {
        try {
            outputStream.write(buffer);


            handler.obtainMessage(MainActivity.MESSAGE_WRITE, -1, -1,
                    buffer).sendToTarget();
        } catch (IOException e) {
        }
    }

    public void cancel() {
        try {
            bluetoothSocket.close();
        } catch (IOException e) {
        }
    }
}
```

Result: