SARASWATI Education Society's

SARASWATI College of Engineering

Learn Live Achieve and Contribute

Kharghar, Navi Mumbai - 410 210.

NAAC Accredited

Department of Computer Engineering

Name: Shaikh Adnan Shaukat Ali

Roll No.: 76

Subject code: CSL603

Subject Name: Mobile Computing Lab

Class/Sem: T.E. / SEM-VI

Year: 2021-2022

SARASWATI Education Society's
SARASWATI College of Engineering
Learn Live Achieve and Contribute
Kharghar, Navi Mumbai - 410 210.
NAAC Accredited

Department of Computer Engineering

Subject: Mobile Computing Lab **Class/Sem: TE/VI**

Name of Laboratory: Network Lab **Year: 2021-2022**

LIST OF EXPERIMENTS

Expt. No.	Name of the Experiment	Date	Page No.
1.	Implementation of Bluetooth Network with application as transfer of a file from one device to another.	24-01-22	4-12
2.	To implement a basic function of Code Division Multiple Access (CDMA).	31-01-22	13-19
3.	Implementation of GSM Security Algorithm(A3/A5/A8).	08-02-22	20-23
4.	Develop and application that uses GUI components (Registration Form).	15-02-22	24-29
5.	To implement a Paint Application in Java.	22-02-22	30-35
6.	To implement a basic function of Global Positioning System (GPS).	08-03-22	36-42
7.	To implement a basic function of Android Notification.	15-03-22	43-47
8.	To implement a program in DBMS using Android Studio.	21-03-22	48-51
9	Write a program to implement EMI calculation using Android Studio.	28-03-22	52-56
10	Implementation of Hidden Terminal Problem	18-04-22	57-65

Dr. Anjali Dadhich **Prof. Sujata Bhairnallykar**

Subject In-charge **HOD**

SARASWATI Education Society's

SARASWATI College of Engineering

Learn Live Achieve and Contribute

Kharghar, Navi Mumbai - 410 210.

NAAC Accredited

Department of Computer Engineering

Subject: Mobile Computing **Class/Sem: TE/VI**

Name of Laboratory: Network Lab **Year: 2021-2022**

ASSIGNMENT INDEX

SR No.	Name of Assignment	Date	Page No.
1	Assignment-1	02-03-22	66-73

Prof. Anjali Dadhich **Prof. Sujata Bhairnallykar**

Subject In-charge **H.O.D**

Experiment No. 1

Aim: To implement a Bluetooth network with application as transfer of a file from one device to another.

Requirements: Windows/MAC/Linux O.S, Compatible version of Android Studio and Android device to test the application.

Theory:

Bluetooth network is an application of wireless PAN (Personal Area Network). It is used to share data between two or more devices (advance in technology allowed connection of more than 2 devices through Bluetooth). It requires both party to make connection before transfer of data, one party shares the data and other party accept it if they want that data or they can reject it. Connection between parties terminated through forceful mean if one party disconnect or close the connection other party will automatically loose the connection.

It takes very low power and achieved it by embedded low-cost transceivers into the devices. It supports the frequency band of 2.45GHz and can support up to 721KBps along with three voice channels. This frequency band has been set aside by international agreement for the use of industrial, scientific, and medical devices (ISM).rd-compatible with 1.0 devices.

It can connect up to "**eight devices"** simultaneously and each device offers a unique 48-bit address from the IEEE 802 standard with the connections being made a point to point or multipoint.

Code:

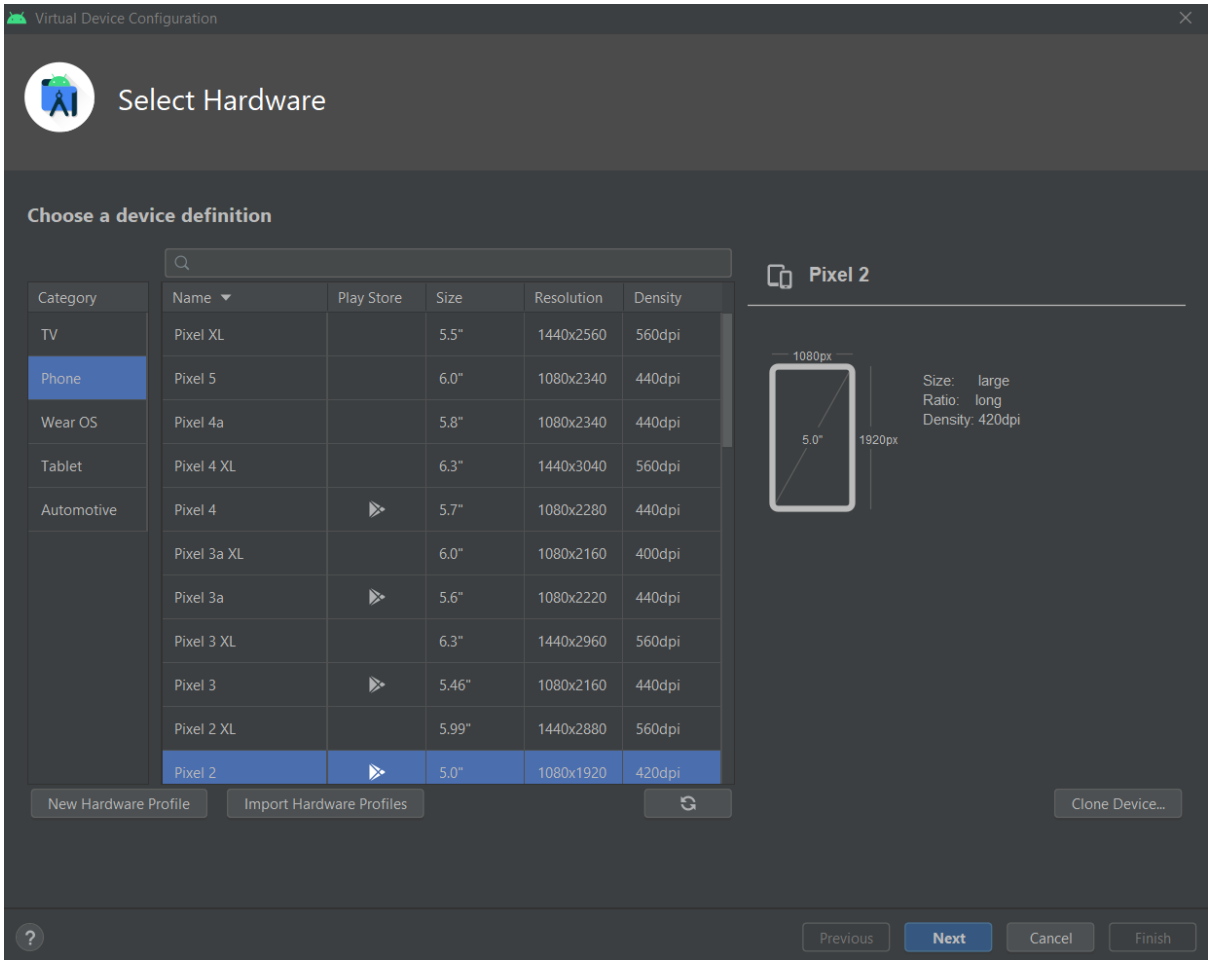
```
public class MainActivity extends AppCompatActivity {
   //Create Objects----------------------------------------------------
   Button buttonopenDailog, buttonUp, send;
   TextView textFolder;
   EditText dataPath;
   static final int CUSTOM_DIALOG_ID = 0;
   ListView dialog_ListView;
   File root, fileroot, curFolder;
   private List<String> fileList = new ArrayList<String>();
   private static final int DISCOVER_DURATION = 300;
   private static final int REQUEST_BLU = 1;
   BluetoothAdapter btAdatper = BluetoothAdapter.getDefaultAdapter();
   //------------------------------------------------------------------
   @Override
   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_main);
      dataPath=(EditText)findViewById(R.id.FilePath);
      buttonopenDailog= (Button) findViewById(R.id.opendailog);
      send=(Button)findViewById(R.id.sendBtooth);
      buttonopenDailog.setOnClickListener(new View.OnClickListener() {
         @Override
         public void onClick(View v) {
            dataPath.setText("");
            showDialog(CUSTOM_DIALOG_ID);
         }
      });

      root = new File(Environment.getExternalStorageDirectory().getAbsolutePath());
      curFolder = root;
      send.setOnClickListener(new View.OnClickListener() {
         @Override
         public void onClick(View v) {
            sendViaBluetooth();
         }
      });
   }

   @Override
   protected Dialog onCreateDialog(int id) {
      Dialog dialog = null;
      switch (id) {
         case CUSTOM_DIALOG_ID:
            dialog = new Dialog(MainActivity.this);
            dialog.setContentView(R.layout.dailoglayout);
            dialog.setTitle("File Selector");
            dialog.setCancelable(true);
            dialog.setCanceledOnTouchOutside(true);
            textFolder = (TextView) dialog.findViewById(R.id.folder);
            buttonUp = (Button) dialog.findViewById(R.id.up);
```

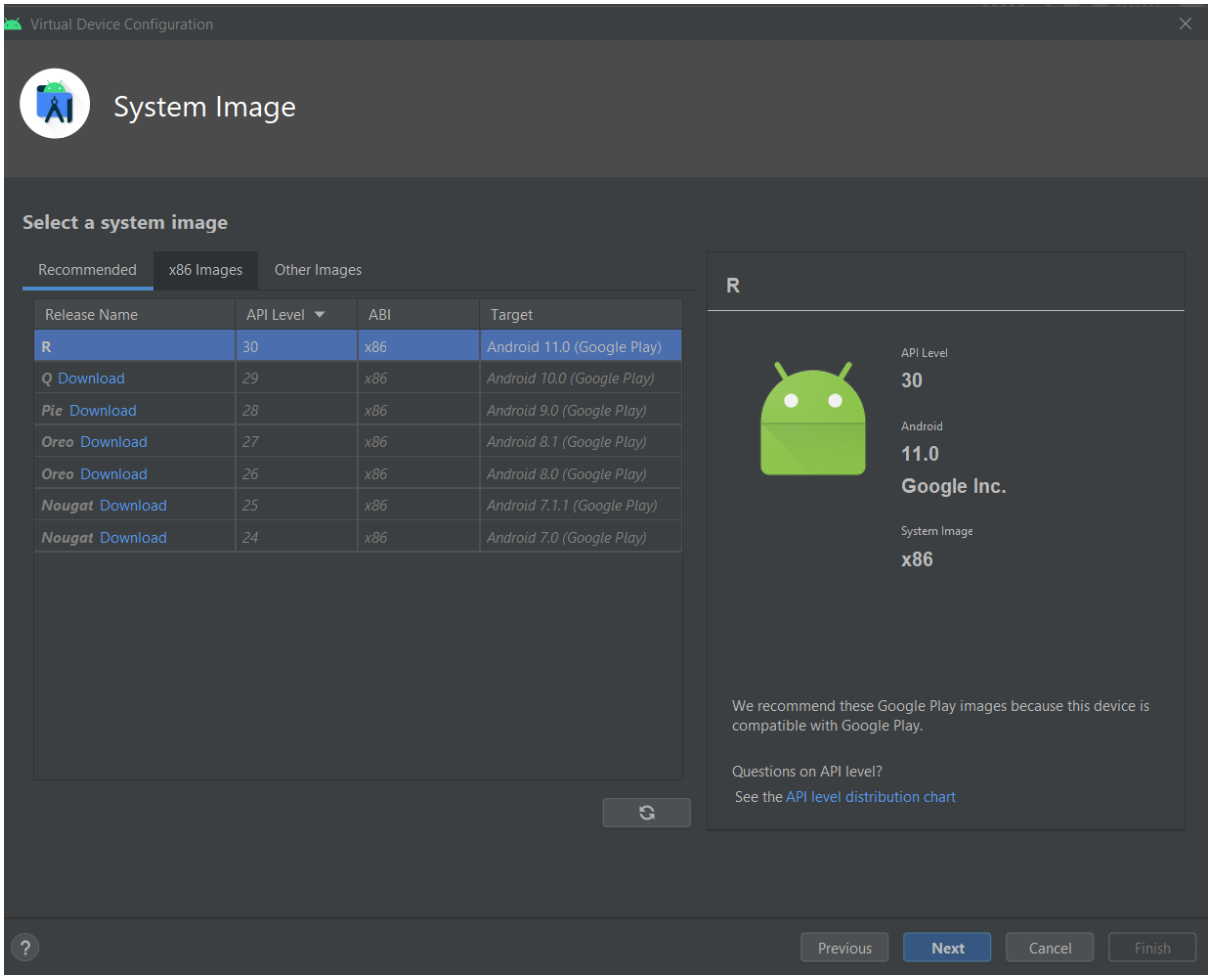
```
            buttonUp.setOnClickListener(new View.OnClickListener() {
               @Override
               public void onClick(View v) {
                  ListDir(curFolder.getParentFile());
               }
            });
            dialog_ListView = (ListView) dialog.findViewById(R.id.dialoglist);
            dialog_ListView.setOnItemClickListener(new AdapterView.OnItemClickListener() {
               @Override
               public void onItemClick(AdapterView<?> parent, View view, int position, long id) {
                  File selected = new File(fileList.get(position));
                  if (selected.isDirectory()) {
                     ListDir(selected);
                  } else if (selected.isFile()) {
                     getselectedFile(selected);
                  } else {
                     dismissDialog(CUSTOM_DIALOG_ID);
                  }
               }
            });
            break;
      }
      return dialog;
   }

   @Override
   protected void onPrepareDialog(int id, Dialog dialog) {
      super.onPrepareDialog(id, dialog);
      switch (id) {
         case CUSTOM_DIALOG_ID:
            ListDir(curFolder);
            break;
      }
   }

   public void getselectedFile(File f){
      dataPath.setText(f.getAbsolutePath());
      fileList.clear();
      dismissDialog(CUSTOM_DIALOG_ID);
   }


   public void ListDir(File f) {
      if (f.equals(root)) {
         buttonUp.setEnabled(false);
      } else {
         buttonUp.setEnabled(true);
      }
      curFolder = f;
      textFolder.setText(f.getAbsolutePath());
      dataPath.setText(f.getAbsolutePath());
```

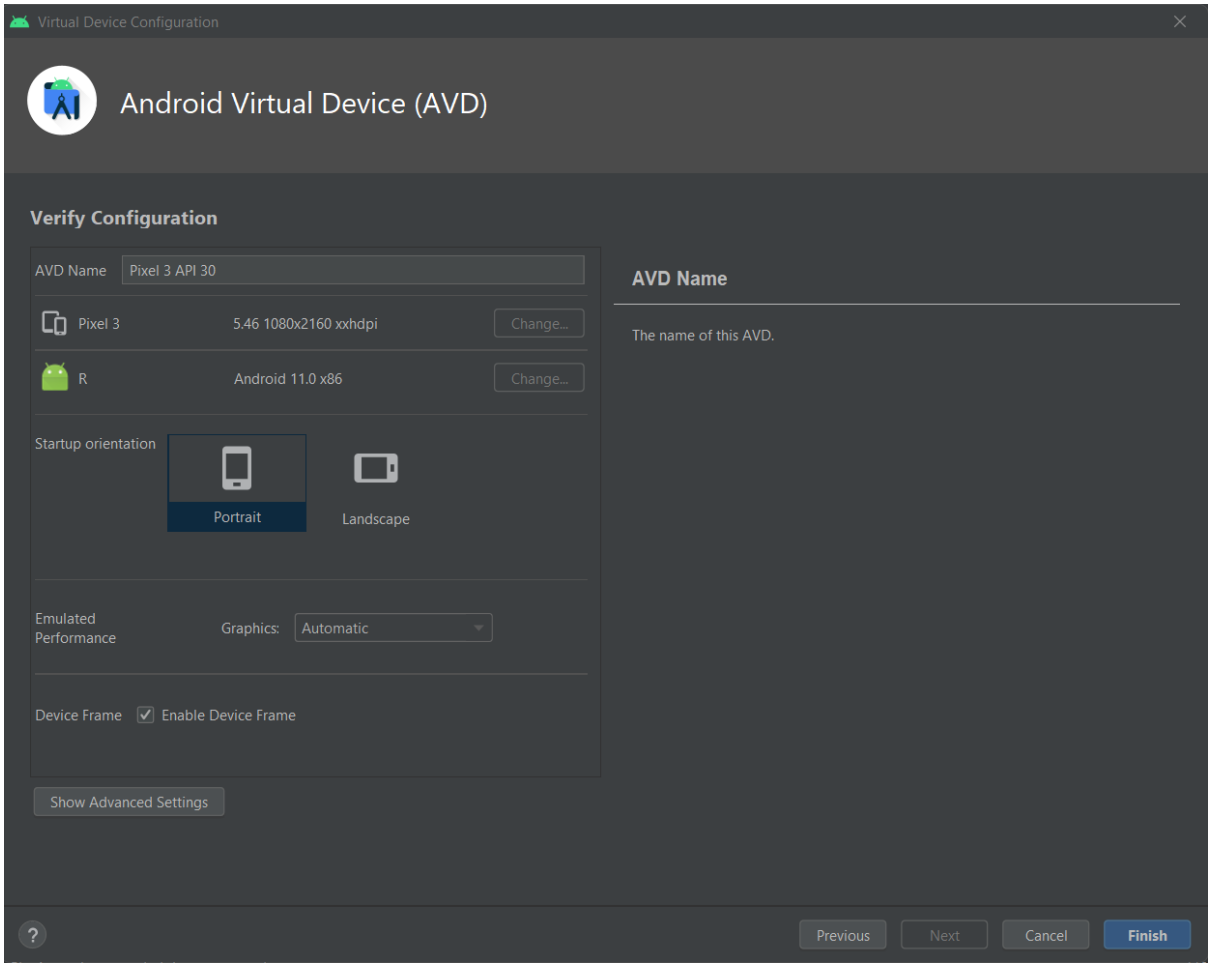
```
        File[] files = f.listFiles();
        fileList.clear();

        for (File file : files) {
          fileList.add(file.getPath());
        }
        ArrayAdapter<String> directoryList = new ArrayAdapter<String>(this,
android.R.layout.simple_list_item_1, fileList);
        dialog_ListView.setAdapter(directoryList);
    }

    //exit to application-------------------------------------------------------------------------
    public void exit(View V) {
        btAdatper.disable();
        Toast.makeText(this,"*** Now Bluetooth is off... Thanks. ***",Toast.LENGTH_LONG).show();
        finish(); }

    //Method for send file via bluetooth---------------------------------------------------------
    public void sendViaBluetooth() {
        if(!dataPath.equals(null)){
        if (btAdatper == null) {
            Toast.makeText(this, "Device not support bluetooth", Toast.LENGTH_LONG).show();
        } else {
            enableBluetooth();
        }
    }else{
            Toast.makeText(this,"Please select a file.",Toast.LENGTH_LONG).show();
        }
    }

    public void enableBluetooth() {
        Intent discoveryIntent = new Intent(BluetoothAdapter.ACTION_REQUEST_DISCOVERABLE);
        discoveryIntent.putExtra(BluetoothAdapter.EXTRA_DISCOVERABLE_DURATION,
DISCOVER_DURATION);
        startActivityForResult(discoveryIntent, REQUEST_BLU);
    }

    //Override method for sending data via bluetooth availability--------------------------
    @Override
    protected void onActivityResult(int requestCode, int resultCode, Intent data) {
        if (resultCode == DISCOVER_DURATION && requestCode == REQUEST_BLU) {
            Intent i = new Intent();
            i.setAction(Intent.ACTION_SEND);
            i.setType("*/*");
            File file = new File(dataPath.getText().toString());

            i.putExtra(Intent.EXTRA_STREAM, Uri.fromFile(file));

            PackageManager pm = getPackageManager();
            List<ResolveInfo> list = pm.queryIntentActivities(i, 0);
            if (list.size() > 0) {
```

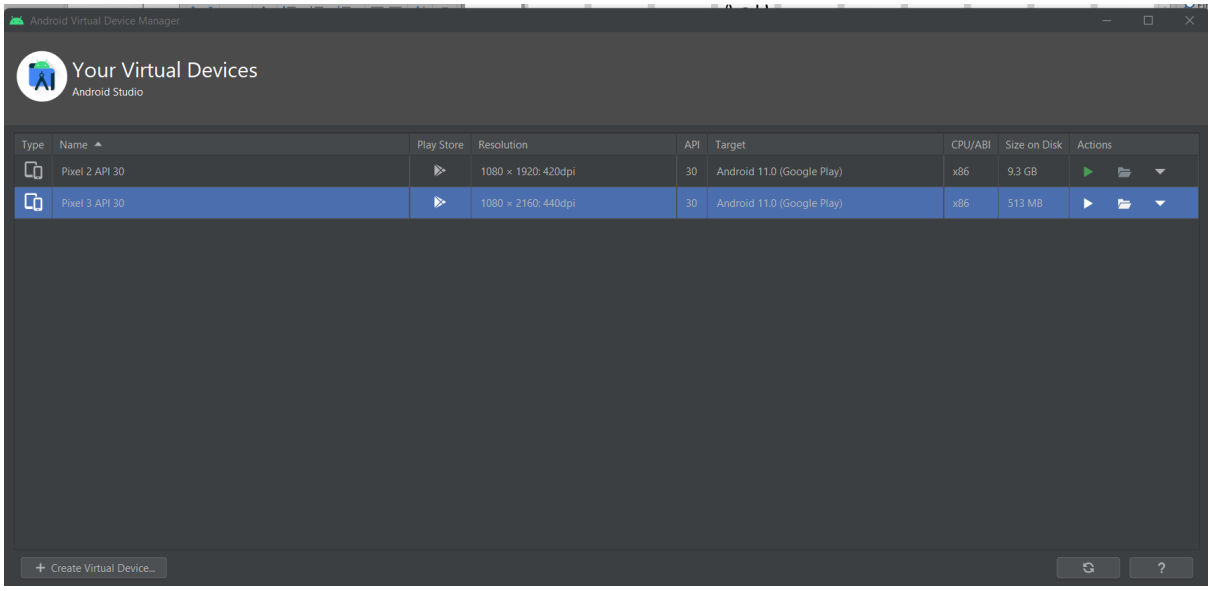
```
            String packageName = null;
            String className = null;
            boolean found = false;

            for (ResolveInfo info : list) {
               packageName = info.activityInfo.packageName;
               if (packageName.equals("com.android.bluetooth")) {
                  className = info.activityInfo.name;
                  found = true;
                  break;
               }
            }
            //CHECK BLUETOOTH available or not----------------------------------------------
            if (!found) {
               Toast.makeText(this, "Bluetooth not been found", Toast.LENGTH_LONG).show();
            } else {
               i.setClassName(packageName, className);
               startActivity(i);
            }
         }
      } else {
         Toast.makeText(this, "Bluetooth is cancelled", Toast.LENGTH_LONG).show();
      }
   }

   @Override
   public boolean onCreateOptionsMenu(Menu menu) {
      // Inflate the menu; this adds items to the action bar if it is present.
      getMenuInflater().inflate(R.menu.menu_main, menu);
      return true;
   }

   @Override
   public boolean onOptionsItemSelected(MenuItem item) {
      // Handle action bar item clicks here. The action bar will
      // automatically handle clicks on the Home/Up button, so long
      // as you specify a parent activity in AndroidManifest.xml.
      int id = item.getItemId();

      //noinspection SimplifiableIfStatement
      if (id == R.id.action_settings) {
         Toast.makeText(this, "****************\nDeveloper: Santosh Kumar Singh\nContact:
superssingh@gmail.com\n****************", Toast.LENGTH_LONG).show();
         return true;
      }
      return super.onOptionsItemSelected(item);
   }
}
```

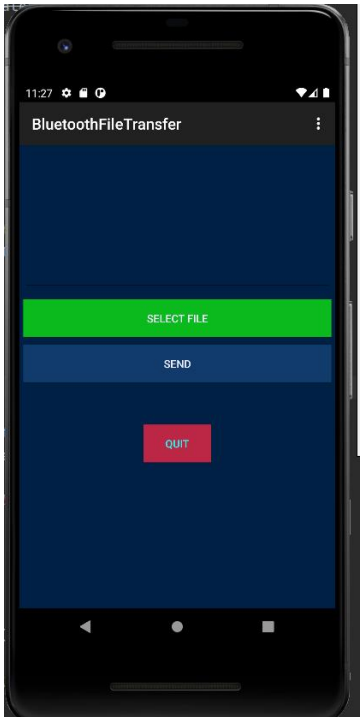
Output:

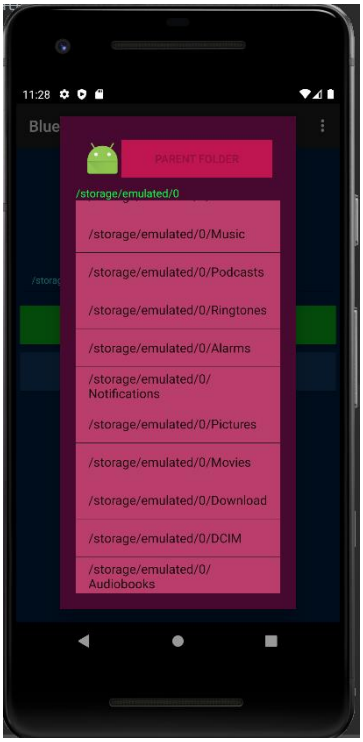

Virtual Device Configuration
Select Hardware
Choose a device definition
Category
TV
Phone
Wear OS
Tablet
Automotive
Name
Play Store
Size
Resolution
Density
Pixel XL 5.5" 1440x2560 560dpi
Pixel 5 6.0" 1080x2340 440dpi
Pixel 4a 5.8" 1080x2340 440dpi
Pixel 4 XL 6.3" 1440x3040 560dpi
Pixel 4 5.7" 1080x2280 440dpi
Pixel 3a XL 6.0" 1080x2160 400dpi
Pixel 3a 5.6" 1080x2220 440dpi
Pixel 3 XL 6.3" 1440x2960 560dpi
Pixel 3 5.46" 1080x2160 440dpi
Pixel 2 XL 5.99" 1440x2880 560dpi
Pixel 2 5.0" 1080x1920 420dpi
Pixel 2
1080px
5.0"
1920px
Size: large
Ratio: long
Density: 420dpi
New Hardware Profile
Import Hardware Profiles
Clone Device...
Previous
Next
Cancel
Finish

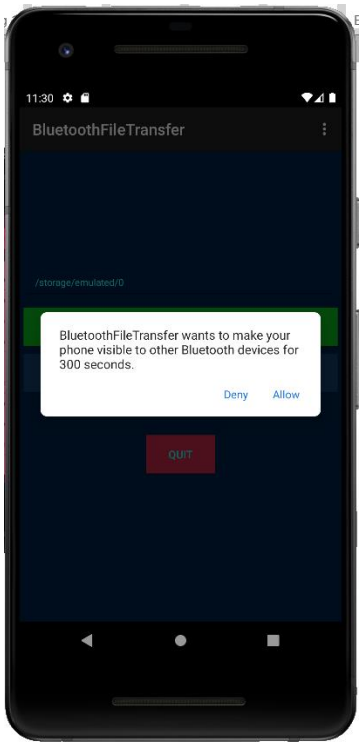

Virtual Device Configuration
System Image
Select a system image
Recommended
x86 Images
Other Images
Release Name
API Level
ABI
Target
R 30 x86 Android 11.0 (Google Play)
Q Download 29 x86 Android 10.0 (Google Play)
Pie Download 28 x86 Android 9.0 (Google Play)
Oreo Download 27 x86 Android 8.1 (Google Play)
Oreo Download 26 x86 Android 8.0 (Google Play)
Nougat Download 25 x86 Android 7.1.1 (Google Play)
Nougat Download 24 x86 Android 7.0 (Google Play)
R
API Level
30
Android
11.0
Google Inc.
System Image
x86
We recommend these Google Play images because this device is compatible with Google Play.
Questions on API level?
See the API level distribution chart
Previous
Next
Cancel
Finish


Virtual Device Configuration
Android Virtual Device (AVD)
Verify Configuration
AVD Name
Pixel 3 API 30
Pixel 3
5.46 1080x2160 xxhdpi
Change...
R
Android 11.0 x86
Change...
Startup orientation
Portrait
Landscape
Emulated Performance
Graphics:
Automatic
Device Frame
Enable Device Frame
Show Advanced Settings
AVD Name
The name of this AVD.
Previous
Next
Cancel
Finish


Android Virtual Device Manager
Your Virtual Devices
Android Studio
Type
Name
Play Store
Resolution
API
Target
CPU/ABI
Size on Disk
Actions
Pixel 2 API 30
1080 × 1920: 420dpi
30
Android 11.0 (Google Play)
x86
9.3 GB
Pixel 3 API 30
1080 × 2160: 440dpi
30
Android 11.0 (Google Play)
x86
513 MB
Create Virtual Device...







Conclusion: We have successfully implemented Bluetooth network with application in Android Studio using Java and tested the application using virtual devices presents in Android Studio.

Experiment No. 2

Aim: To implement CDMA in Java.

Requirements: Compatible version of JDK.

Theory: Code division multiple access (CDMA) is a form of spread spectrum communications that is being used for, among other applications, some second-generation cellular phone systems (others use TDMA). The method is also known as S-CDMA, where the S stands for *synchronous*. S-CDMA is one of the advanced upstream options specified for DOCSIS 2.0.

Figure A illustrates the principle of CDMA transmission. The basic modulation is often QAM[1] of some level. The data to be modulated on each axis is exclusive ORed with a higher-speed pseudorandom bit sequence (PRBS), which is also known as a *spreading code* or *spreading sequence*. The signal is then modulated, normally using m-ary QAM. The effect is to spread out the carrier over a wider bandwidth than necessary for carriage of the information.

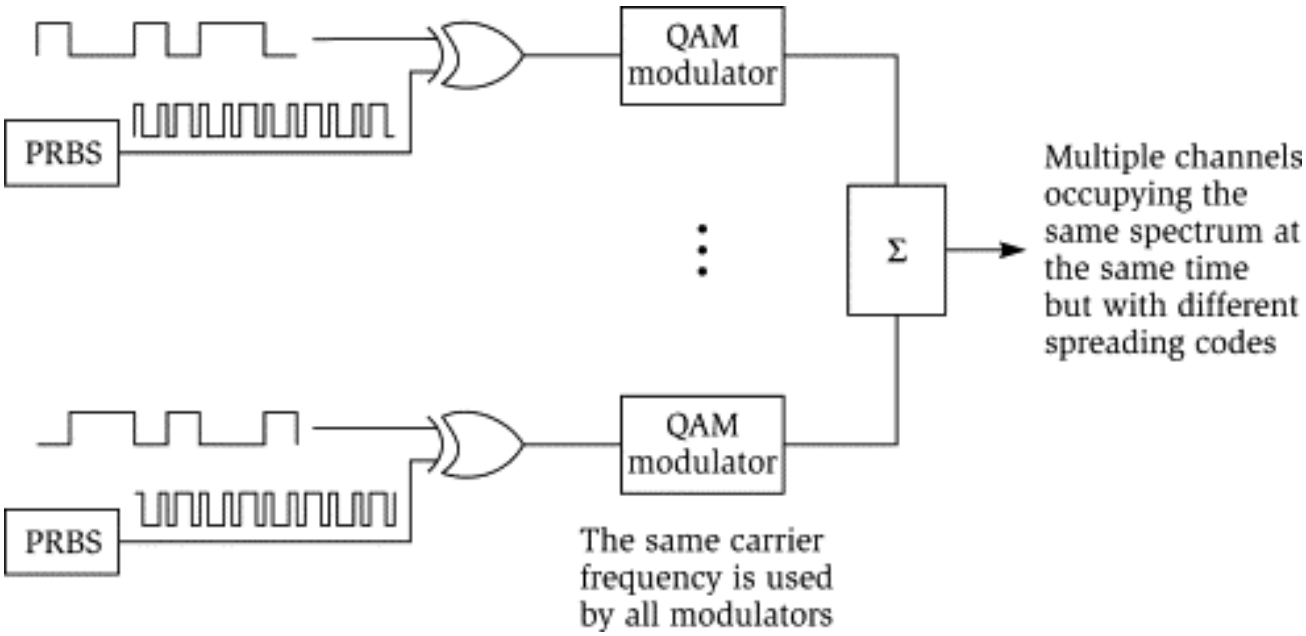


Figure A. Principle of CDMA Transmission.

QAM[1]: Quadrature Amplitude Modulation implements combination of Amplitude and Phase shift keying, standard QAM uses three different amplitudes and 12 different phases

At the receiver, the demodulated signal is again exclusive ORed (EXOR) with the same PRBS. The result is a collapse of the transmission bandwidth back to that required to convey the information signal without the PRBS. Recovery of the original signal can then proceed. In the process, any narrowband interference that is added to the signal during transmission is spread, most of it out of the receive channel. In addition, if an impairment, such as a suck-out, affects part of the channel, the signal quality is minimally degraded, because most of the energy lies outside of the affected portion of the passband. (For the benefit of data communications engineers, the term *suck-out* is used in the cable television industry to mean a narrow frequency-selective reduction in the response of a transmission path.)

Figure A illustrates that many intentional transmissions may occupy the same frequency band simultaneously without interfering with one another, so long as each uses a different spreading sequence. All signals may be received simultaneously, with each having its own unique *despreading* sequence applied to separate it from the other signals. As each signal is separated with its despreading sequence, power in the other signals is spread out as if it were background noise. All spreading codes used must have the mutual (i.e., shared) property of *orthogonality*. That is, they must be noninterfering with each other. The practical limitation on the number of signals that can be transmitted in the same bandwidth is the number of orthogonal spreading codes available. In turn, the number of spreading codes depends on the number of bits in the spreading sequence: The longer the sequence, the more orthogonal codes exist. Spreading code design is a complex subject.

Example: Consider, there are 4 stations using CDMA and they all send the data simultaneously

$Station_1$: $D_1 = 0$, $Station_2$: $D_2 = 0$, $Station_3$: $D_3 = 1$, $Station_4$: $D_4 = 1$.

Since, there are 4 stations we will need 4 orthogonal codes of length 4. We will use Walsh table to generate orthogonal codes.

$W_1 = [0],\ W_{2(1)} = \begin{bmatrix} W_1 & W_1 \\ W_1 & W_1' \end{bmatrix},\quad W_{2(2)} = W_4 = \begin{bmatrix} W_2 & W_2 \\ W_2 & W_2' \end{bmatrix}$ ----- $\{W'$ represents complement of $W\}$.

Using this recursive procedure we will prepare Walsh table of desired length. We need Walsh table of length 4 i.e. W_4.

$$W_4 = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 \end{bmatrix}$$

In our case we will replace 0 with 1 and 1 with -1 (0 can be replaced with -1 and 1 stay as it is). $[0,1] \rightarrow [1, -1]$

$$W_4 = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{bmatrix}$$

$Station_1$: $D_1 = 1$, $Station_2$: $D_2 = 1$, $Station_3$: $D_3 = -1$, $Station_4$: $D_4 = -1$.

Providing row entries of table as a chipping code to station:

$Station_1$: $C_1 = [1 \quad 1 \quad 1 \quad 1]$, $Station_2$: $C_2 = [1 \quad -1 \quad 1 \quad -1]$,

$Station_3$: $C_3 = [1 \quad 1 \quad -1 \quad -1]$, $Station_4$: $C_4 = [1 \quad -1 \quad -1 \quad 1]$.

If all stations send data together the channel with have sum of scalar multiple of data and corresponding code.

$D_1C_1 + D_2C_2 + D_3C_3 + D_4C_4 = [1 \quad 1 \quad 1 \quad 1] + [1 \quad -1 \quad 1 \quad -1] + [-1 \quad -1 \quad 1 \quad 1] + [-1 \quad 1 \quad 1 \quad -1]$. – (1)

Consider, if $Station_3$ wants the data of $Station_1$ then it will just take dot product of equation (1) and code C_1.

$D_1 = C_1 \cdot (D_1C_1 + D_2C_2 + D_3C_3 + D_4C_4) = 4 + 0 + 0 + 0 = 4$. Normalizing by length 4 (since, we consider code of length 4) we get, $D_1 = 4/4 = 1 = 0$ (Back to binary form). Value of operation is indeed the data of D_1.

Implementation:

```
import java.util.*;

public class CDMA {

   private int[][] wtable;
   private int[][] copy;
   private int[] channel_sequence;

   public int[] setUp(int[] data, int num_stations)
   {

      wtable = new int[num_stations][num_stations];
      copy = new int[num_stations][num_stations];

      buildWalshTable(num_stations, 0, num_stations - 1, 0,
                              num_stations - 1, false);

      showWalshTable(num_stations);

      for (int i = 0; i < num_stations; i++) {

         for (int j = 0; j < num_stations; j++) {

            // Making a copy of walsh table
            // to be used later
            copy[i][j] = wtable[i][j];

            // each row in table is code for one station.
            // So we multiply each row with station data
            wtable[i][j] *= data[i];
         }
      }

      channel_sequence = new int[num_stations];

      for (int i = 0; i < num_stations; i++) {

         for (int j = 0; j < num_stations; j++) {
            // Adding all sequences to get channel sequence
            channel_sequence[i] += wtable[j][i];
         }
      }
```

```
   return channel_sequence;
}

public void listenTo(int sourceStation, int num_stations)
{
   int innerProduct = 0;

   for (int i = 0; i < num_stations; i++) {

      // multiply channel sequence and source station code
      innerProduct += copy[sourceStation][i] * channel_sequence[i];
   }
   int data = innerProduct / num_stations;
   if(data == -1) data = 1;
   else data = 0;
   System.out.println("The data received is: " + data);
}

public int buildWalshTable(int len, int i1, int i2, int j1,
                           int j2, boolean isBar)
{
   // len = size of matrix. (i1, j1), (i2, j2) are
   // starting and ending indices of wtable.

   // isBar represents whether we want to add simple entry
   // or complement(southeast submatrix) to wtable.

   if (len == 2) {

      if (!isBar) {

         wtable[i1][j1] = 1;
         wtable[i1][j2] = 1;
         wtable[i2][j1] = 1;
         wtable[i2][j2] = -1;
      }
      else {

         wtable[i1][j1] = -1;
         wtable[i1][j2] = -1;
         wtable[i2][j1] = -1;
         wtable[i2][j2] = 1;
      }

      return 0;
   }

   int midi = (i1 + i2) / 2;
   int midj = (j1 + j2) / 2;

   buildWalshTable(len / 2, i1, midi, j1, midj, isBar);
```

```
        buildWalshTable(len / 2, i1, midi, midj + 1, j2, isBar);
        buildWalshTable(len / 2, midi + 1, i2, j1, midj, isBar);
        buildWalshTable(len / 2, midi + 1, i2, midj + 1, j2, !isBar);

        return 0;
    }

    public void showWalshTable(int num_stations)
    {

        System.out.print("\n");

        for (int i = 0; i < num_stations; i++) {
            for (int j = 0; j < num_stations; j++) {
                System.out.print(wtable[i][j] + " ");
            }
            System.out.print("\n");
        }
        System.out.println("-------------------------");
        System.out.print("\n");
    }

    // Driver Code
    public static void main(String[] args)
    {
        int num_stations = 4;
        int[] channel_sequence;
        int[] data = new int[num_stations];
        //data bits corresponding to each station
        data[0] = 0;
        data[1] = 0;
        data[2] = 1;
        data[3] = 1;

        System.out.println("D1 = "+data[0]+ " D2 = "+data[1]+" D3 = "+data[2]+" D4 = "+data[3]);
        for(int i=0;i<4;i++) if(data[i]==0) data[i] = 1; else data[i] = -1;
        CDMA channel = new CDMA();

        channel_sequence = channel.setUp(data, num_stations);
        System.out.print("Channel Seqeunce: ");
        for(int C: channel_sequence) System.out.print(C+" ");
        System.out.println();

        while(true){
            System.out.print("Enter the station no. to extract the data: ");
            // station you want to listen to
            int sourceStation = new Scanner(System.in).nextInt() - 1;
            channel.listenTo(sourceStation, num_stations);
        }
    }
}
```

Output:

```
PS C:\Users\adnan\OneDrive\Desktop\College\sem6\MC\practical2> javac CDMA.java
PS C:\Users\adnan\OneDrive\Desktop\College\sem6\MC\practical2> java CDMA
D1 = 0 D2 = 0 D3 = 1 D4 = 1

1 1 1 1
1 -1 1 -1
1 1 -1 -1
1 -1 -1 1
------------------------

Channel Seqeunce: 0 0 4 0
Enter the station no. to extract the data: 1
The data received is: 0
Enter the station no. to extract the data: 2
The data received is: 0
Enter the station no. to extract the data: 3
The data received is: 1
Enter the station no. to extract the data: 4
The data received is: 1
```

Conclusion: We have understand the concept of CDMA and implemented in JAVA.

Experiment No. 3

Aim: To implement security algorithm of GSM in python.

Requirements: Compatible version of python.

Theory:

GSM is the most secured cellular telecommunications system available today. GSM has its security methods standardized. GSM maintains end-to-end security by retaining the confidentiality of calls and anonymity of the GSM subscriber.

Temporary identification numbers are assigned to the subscriber's number to maintain the privacy of the user. The privacy of the communication is maintained by applying encryption algorithms and frequency hopping that can be enabled using digital systems and signalling.

Mobile Station Authentication:

The GSM network authenticates the identity of the subscriber through the use of a challenge-response mechanism. A 128-bit Random Number (RAND) is sent to the MS. The MS computes the 32-bit Signed Response (SRES) based on the encryption of the RAND with the authentication algorithm (A3) using the individual subscriber authentication key (Ki). Upon receiving the SRES from the subscriber, the GSM network repeats the calculation to verify the identity of the subscriber.

The individual subscriber authentication key (Ki) is never transmitted over the radio channel, as it is present in the subscriber's SIM, as well as the AUC, HLR, and VLR databases. If the received SRES agrees with the calculated value, the MS has been successfully authenticated and may continue. If the values do not match, the connection is terminated and an authentication failure is indicated to the MS.

The calculation of the signed response is processed within the SIM. It provides enhanced security, as confidential subscriber information such as the IMSI or the individual subscriber authentication key (Ki) is never released from the SIM during the authentication process.

Signalling and Data Confidentiality:

The SIM contains the ciphering key generating algorithm (A8) that is used to produce the 64-bit ciphering key (Kc). This key is computed by applying the same random number (RAND) used in the authentication process to ciphering key generating algorithm (A8) with the individual subscriber authentication key (Ki).

GSM provides an additional level of security by having a way to change the ciphering key, making the system more resistant to eavesdropping. The ciphering key may be changed at regular intervals as required. As in case of the authentication process, the computation of the ciphering key (Kc) takes place internally within the SIM. Therefore, sensitive information such as the individual subscriber authentication key (Ki) is never revealed by the SIM.

Encrypted voice and data communications between the MS and the network is accomplished by using the ciphering algorithm A5. Encrypted communication is initiated by a ciphering mode request command from the GSM network. Upon receipt of this command, the mobile station begins encryption and decryption of data using the ciphering algorithm (A5) and the ciphering key (Kc).

Subscriber Identity Confidentiality:

To ensure subscriber identity confidentiality, the Temporary Mobile Subscriber Identity (TMSI) is used. Once the authentication and encryption procedures are done, the TMSI is sent to the mobile station. After the receipt, the mobile station responds. The TMSI is valid in the location area in which it was issued. For communications outside the location area, the Location Area Identification (LAI) is necessary in addition to the TMSI.

Code:

```
import random
import math

def encryption(data_size):
    assert 8<= data_size <= 128, "Data size should be in range [8,128]"
    assert data_size & (data_size-1) == 0, "Data size must be in power of two"

    key, data = [bin(random.getrandbits(data_size))[2:].zfill(data_size) for _ in range(2)]
    key_left, key_right = key[:data_size//2], key[data_size//2:]
    data_left, data_right = data[:data_size//2], data[data_size//2:]
    a1, a2 = int(key_left,2)^int(data_right,2),int(key_right,2)^int(data_left,2)
    a3 = a1^a2
    a4 = bin(a3)[2:].zfill(data_size//2)
    a5, a6 = a4[:data_size//4], a4[data_size//4:]
    a7 = int(a5,2)^int(a6,2)

    return key, data, bin(a7)[2:].zfill(data_size//4)

data_size = int(input("Enter number in range [8,128] and in power of 2: "))
print()
try:
    key, random_bits, res_sres_ratio = encryption(data_size)
    print(f"{data_size} bit key= {key}\n\n{data_size} random bits generated=
{random_bits}\n")
    print(f"RES/SRES= {res_sres_ratio}")

except AssertionError as error:
    print(error)
```

Output:

```
Enter number in range [8,128] and in power of 2: 8

8 bit key= 10100000

8 random bits generated= 00010111

RES/SRES= 11
```

```
Enter number in range [8,128] and in power of 2: 16

16 bit key= 1101110010110101

16 random bits generated= 0111000111010101

RES/SRES= 0001
```

```
Enter number in range [8,128] and in power of 2: 32

32 bit key= 10000101101101100111101000001100

32 random bits generated= 01000010011011100000110101111000

RES/SRES= 00011100
```

```
Enter number in range [8,128] and in power of 2: 64

64 bit key= 0001111001010110011000111101110110110001010001011111001111000101

64 random bits generated= 0100011111111111110100011001000000011110001101101110010110111110

RES/SRES= 0010110011000111


Enter number in range [8,128] and in power of 2: 128

128 bit key= 11100001111001000011011010011000100011110011111010000111110011000011110101010010101101111101110100011101010000110000011011111010

128 random bits generated= 01111110011111001100010100010011100011110111101100001100011110000100100001001011001010101010000001110110101010111100001010111010

RES/SRES= 10000001001011000010000100000010
```

```
Enter number in range [8,128] and in power of 2: 4

Data size should be in range [8,128]
```

```
Enter number in range [8,128] and in power of 2: 256

Data size should be in range [8,128]
```

```
Enter number in range [8,128] and in power of 2: 100

Data size must be in power of two
```

Conclusion: We have successfully implemented GSM security algorithm in python.

Experiment No. 4

Aim: To create GUI base application in Android.

Requirements: Compatible version of Android Studio, Gradle and Java.

Theory:

User Interface

Your app's user interface is everything that the user can see and interact with. Android provides a variety of pre-built UI components such as structured layout objects and UI controls that allow you to build the graphical user interface for your app. Android also provides other UI modules for special interfaces such as dialogs, notifications, and menus.

Layouts

A layout defines the structure for a user interface in your app, such as in an activity. All elements in the layout are built using a hierarchy of View and ViewGroup objects. A View usually draws something the user can see and interact with. Whereas a ViewGroup is an invisible container that defines the layout structure for View and other ViewGroup objects, as shown in figure 1.

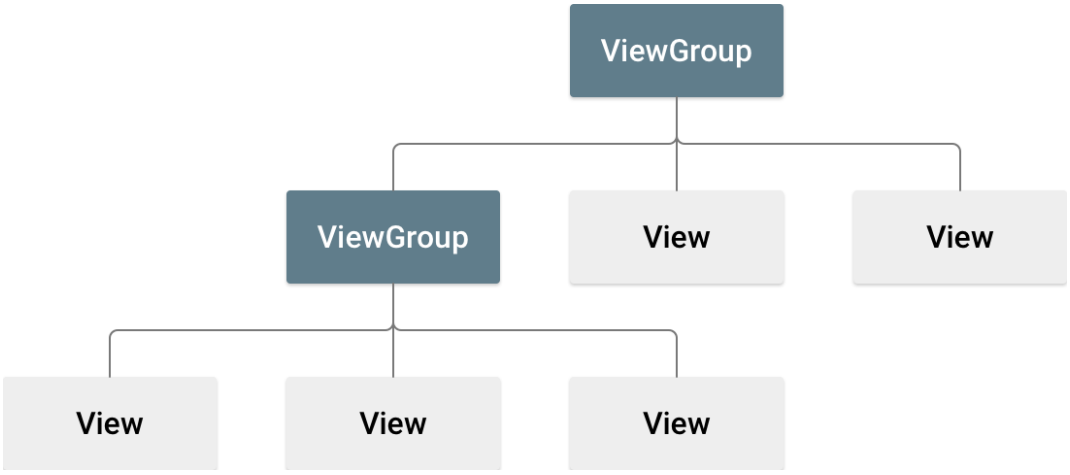


Figure 1. Illustration of a view hierarchy, which defines a UI layout

The View objects are usually called "widgets" and can be one of many subclasses, such as Button or TextView. The ViewGroup objects are usually called "layouts" can be one of many types that provide a different layout structure, such as LinearLayout or ConstraintLayout.

You can declare a layout in two ways:

- **Declare UI elements in XML**. Android provides a straightforward XML vocabulary that corresponds to the View classes and subclasses, such as those for widgets and layouts.

You can also use Android Studio's Layout Editor to build your XML layout using a drag-and-drop interface.

- **Instantiate layout elements at runtime**. Your app can create View and ViewGroup objects (and manipulate their properties) programmatically.

Declaring your UI in XML allows you to separate the presentation of your app from the code that controls its behavior. Using XML files also makes it easy to provide different layouts for different screen sizes and orientations (discussed further in Supporting Different Screen Sizes).

The Android framework gives you the flexibility to use either or both of these methods to build your app's UI. For example, you can declare your app's default layouts in XML, and then modify the layout at runtime.

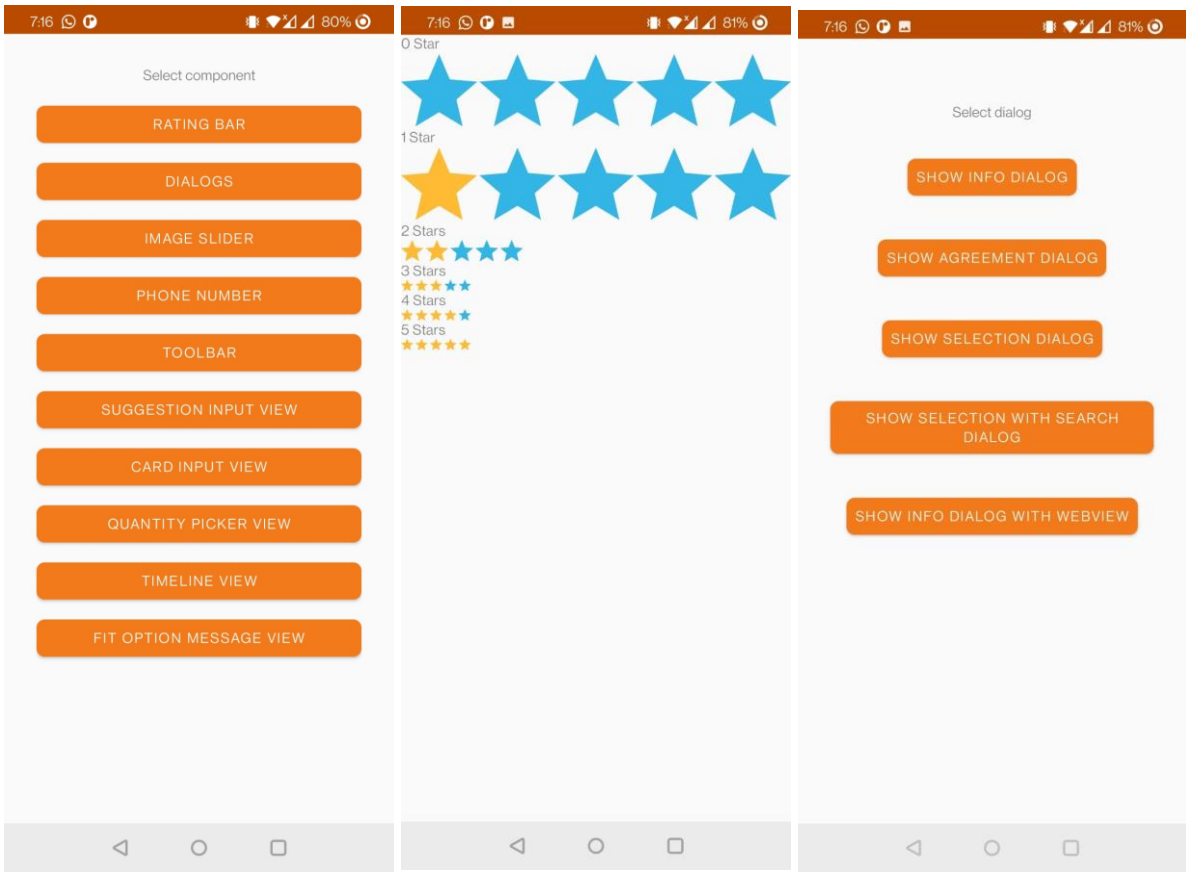
Code:

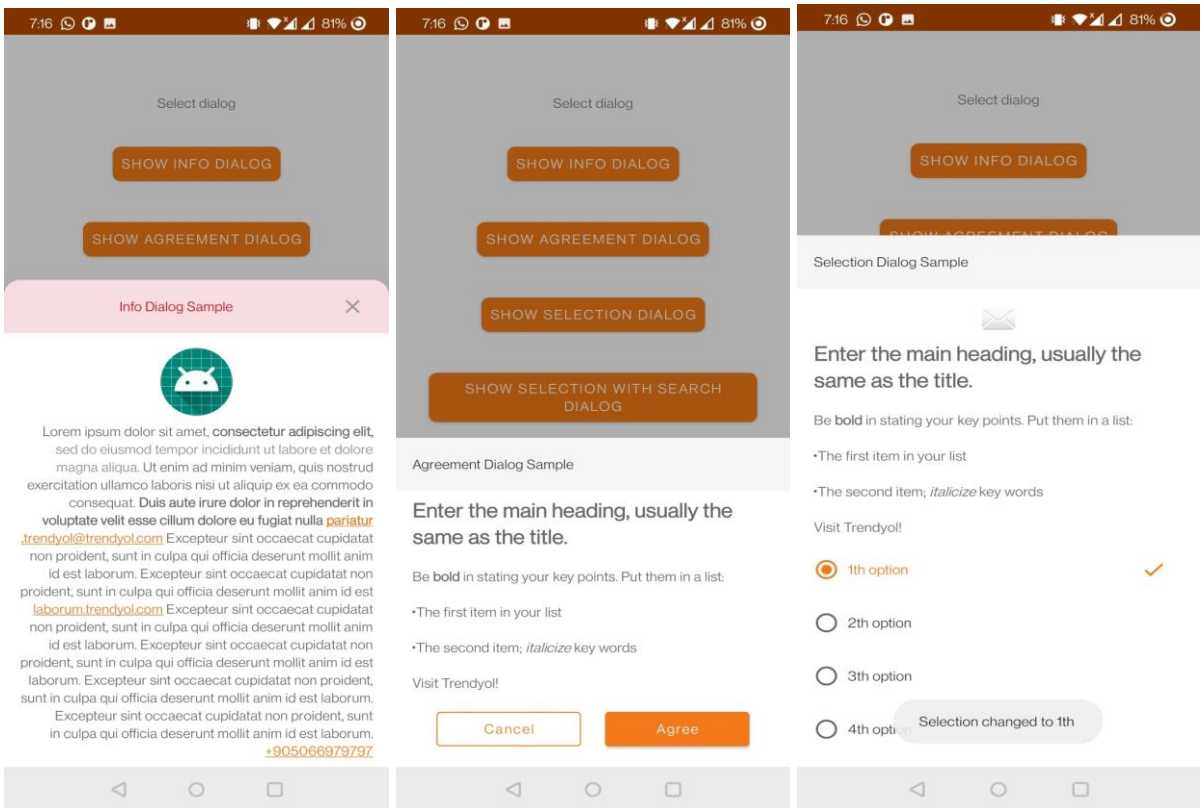
```
package com.trendyol.uicomponents
import android.content.Intent
import android.os.Bundle
import android.widget.Button
import androidx.appcompat.app.AppCompatActivity

class MainActivity : AppCompatActivity() {

    private val buttonRatingBar by lazy { findViewById<Button>(R.id.button_rating_bar) }
    private val buttonDialogs by lazy { findViewById<Button>(R.id.button_dialogs) }
```

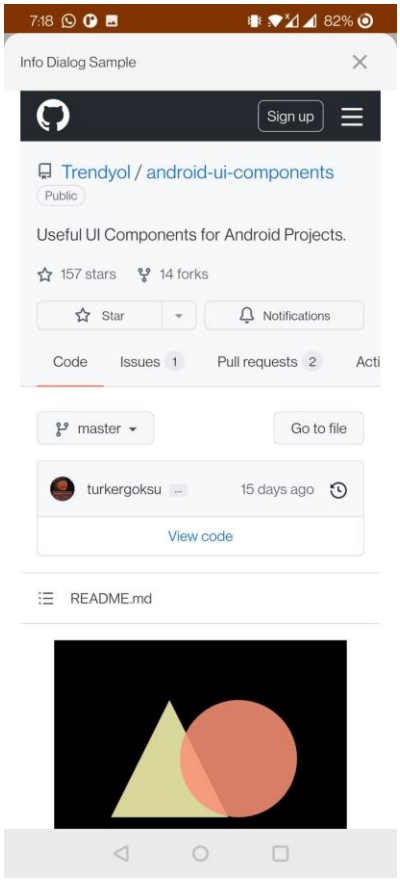
```
    private val buttonImageslider by lazy { findViewById<Button>(R.id.button_imageslider)
}
    private val buttonPhoneNumber by lazy {
findViewById<Button>(R.id.button_phone_number) }
    private val buttonToolbar by lazy { findViewById<Button>(R.id.button_toolbar) }
    private val buttonSuggestionInputView by lazy {
findViewById<Button>(R.id.button_suggestion_input_view) }
    private val buttonCardInput by lazy { findViewById<Button>(R.id.button_card_input) }
    private val buttonQuantityPicker by lazy {
findViewById<Button>(R.id.button_quantity_picker) }
    private val buttonTimelineView by lazy {
findViewById<Button>(R.id.button_timeline_view) }
    private val buttonFitOptionMessageView by lazy {
findViewById<Button>(R.id.button_fit_option_message_view) }

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)

        buttonRatingBar.setOnClickListener {
            startActivity(Intent(this, RatingBarActivity::class.java))
        }
        buttonDialogs.setOnClickListener {
            startActivity(Intent(this, DialogsActivity::class.java))
        }
        buttonImageslider.setOnClickListener {
            startActivity(Intent(this, ImageSliderActivity::class.java))
        }
        buttonPhoneNumber.setOnClickListener {
            startActivity(Intent(this, PhoneNumberActivity::class.java))
        }
        buttonToolbar.setOnClickListener {
            startActivity(Intent(this, ToolbarActivity::class.java))
        }
        buttonSuggestionInputView.setOnClickListener {
            startActivity(Intent(this, SuggestionInputViewActivity::class.java))
        }
        buttonCardInput.setOnClickListener {
            startActivity(Intent(this, CardInputViewActivity::class.java))
        }
        buttonQuantityPicker.setOnClickListener {
            startActivity(Intent(this, QuantityPickerViewActivity::class.java))
        }
        buttonTimelineView.setOnClickListener {
            startActivity(Intent(this, TimelineViewActivity::class.java))
        }
        buttonFitOptionMessageView.setOnClickListener {
            startActivity(Intent(this, FitOptionMessageViewActivity::class.java))
        }
    }
```

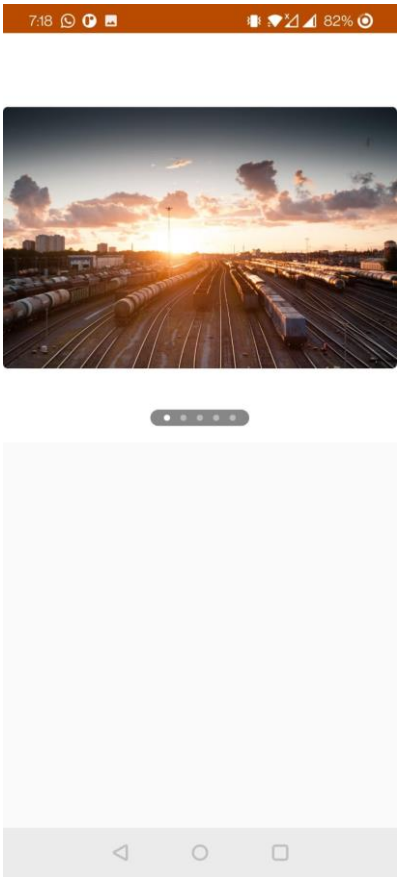
}

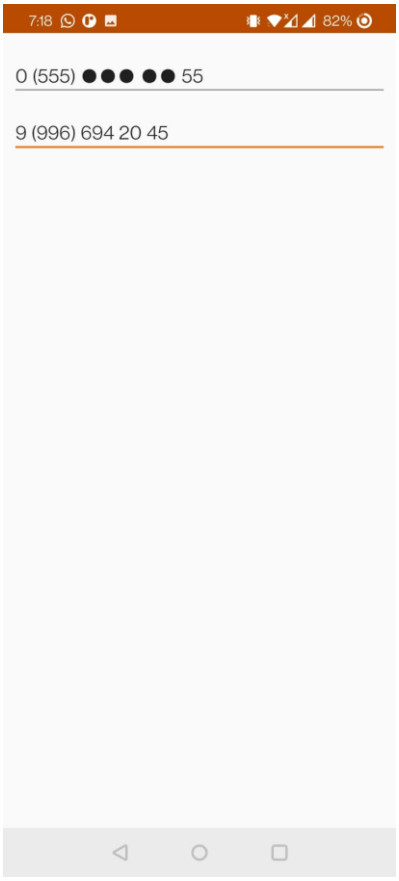
Output:

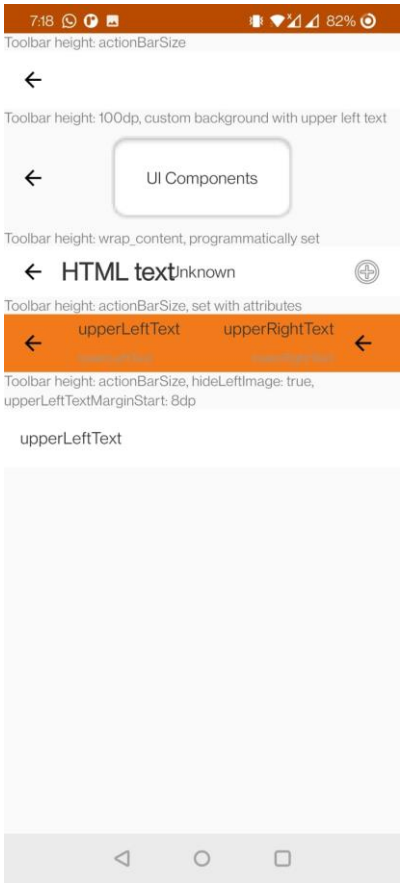

Select component
RATING BAR
DIALOGS
IMAGE SLIDER
PHONE NUMBER
TOOLBAR
SUGGESTION INPUT VIEW
CARD INPUT VIEW
QUANTITY PICKER VIEW
TIMELINE VIEW
FIT OPTION MESSAGE VIEW
0 Star
1 Star
2 Stars
3 Stars
4 Stars
5 Stars
Select dialog
SHOW INFO DIALOG
SHOW AGREEMENT DIALOG
SHOW SELECTION DIALOG
SHOW SELECTION WITH SEARCH DIALOG
SHOW INFO DIALOG WITH WEBVIEW


Select dialog
SHOW INFO DIALOG
SHOW AGREEMENT DIALOG
Info Dialog Sample
Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur .trendyol@trendyol.com Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.trendyol.com Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum. +905066979797
Select dialog
SHOW INFO DIALOG
SHOW AGREEMENT DIALOG
SHOW SELECTION DIALOG
SHOW SELECTION WITH SEARCH DIALOG
Agreement Dialog Sample
Enter the main heading, usually the same as the title.
Be bold in stating your key points. Put them in a list:
•The first item in your list
•The second item; italicize key words
Visit Trendyol!
Cancel
Agree
Select dialog
SHOW INFO DIALOG
Selection Dialog Sample
Enter the main heading, usually the same as the title.
Be bold in stating your key points. Put them in a list:
•The first item in your list
•The second item; italicize key words
Visit Trendyol!
1th option
2th option
3th option
4th option
Selection changed to 1th

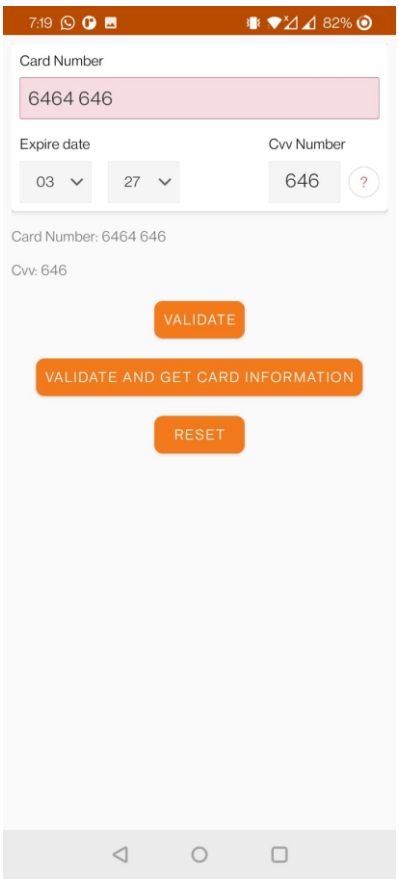

Info Dialog Sample
Sign up
Trendyol / android-ui-components
Public
Useful UI Components for Android Projects.
157 stars
14 forks
Star
Notifications
Code
Issues 1
Pull requests 2
Acti
master
Go to file
turkergoksu
15 days ago
View code
README.md

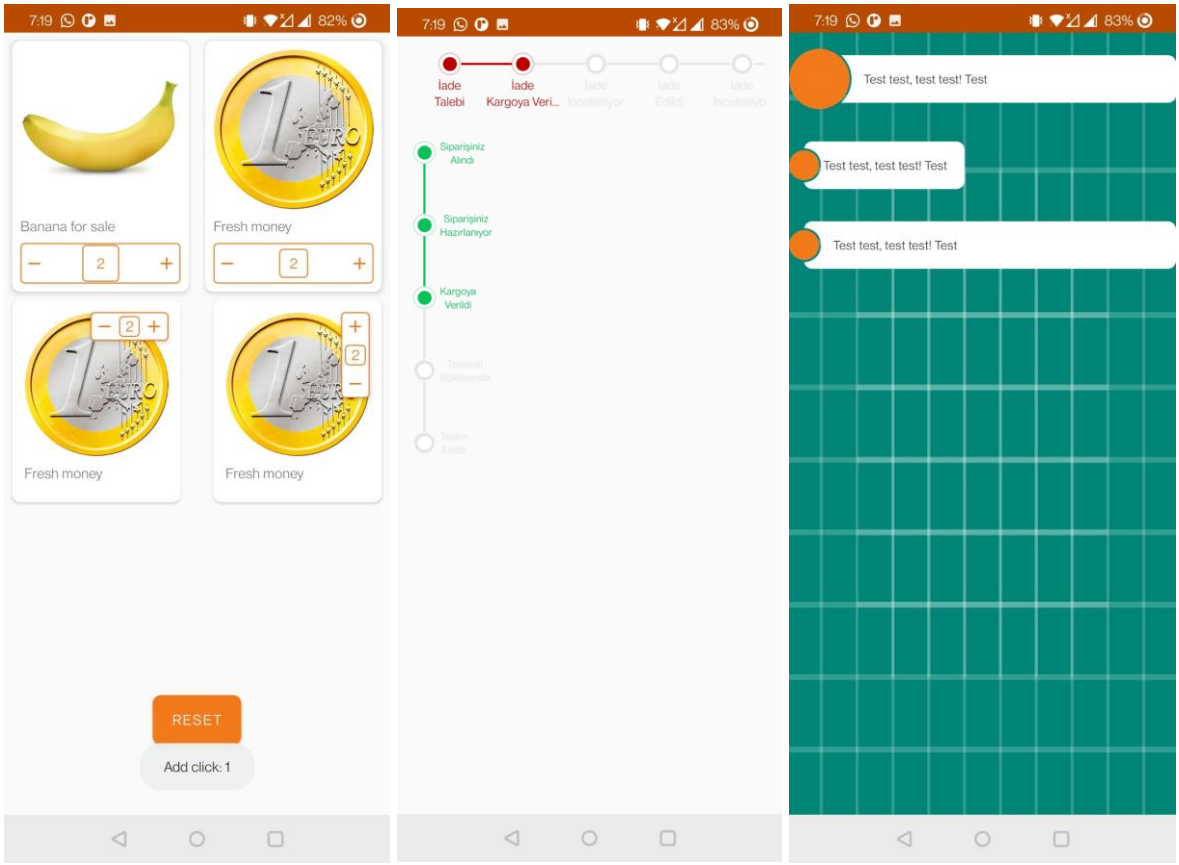



0 (555) ●●● ●● 55
9 (996) 694 20 45


Toolbar height: actionBarSize
Toolbar height: 100dp, custom background with upper left text
UI Components
Toolbar height: wrap_content, programmatically set
HTML text Unknown
Toolbar height: actionBarSize, set with attributes
upperLeftText
upperRightText
Toolbar height: actionBarSize, hideLeftImage: true, upperLeftTextMarginStart: 8dp
upperLeftText


10 ₺
20 ₺
30 ₺
Other
LOAD
Selected Text: 30 ₺
Selected Value: 30


Card Number
6464 646
Expire date
Cvv Number
03
27
646
Card Number: 6464 646
Cvv: 646
VALIDATE
VALIDATE AND GET CARD INFORMATION
RESET



Conclusion: We have successfully implemented GUI app in andorid.

Experiment No. 5

Aim: To create Drawing application in Android.

Requirements: Compatible version of Android Studio, Gradle and Java.

Theory:

A ShapeDrawable object can be a good option when you want to dynamically draw a two-dimensional graphic. You can programmatically draw primitive shapes on a ShapeDrawable object and apply the styles that your app needs.

ShapeDrawable is a subclass of Drawable. For this reason, you can use a ShapeDrawable wherever a Drawable is expected. For example, you can use a ShapeDrawable object to set the background of a view by passing it to the setBackgroundDrawable() method of the view. You can also draw your shape as its own custom view and add it to a layout in your app.

Because ShapeDrawable has its own draw() method, you can create a subclass of View that draws the ShapeDrawable object during the onDraw() event.

If you want to use the custom view in the XML layout instead, then the CustomDrawableView class must override the View(Context, AttributeSet) constructor, which is called when the class is inflated from XML. The following example shows how to declare the CustomDrawableView in the XML layout:

```
<com.example.shapedrawable.CustomDrawableView
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    />
```

The ShapeDrawable class, like many other drawable types in the android.graphics.drawable package, allows you to define various properties of the object by using public methods. Some example properties you might want to adjust include alpha transparency, color filter, dither, opacity, and color.

Code:

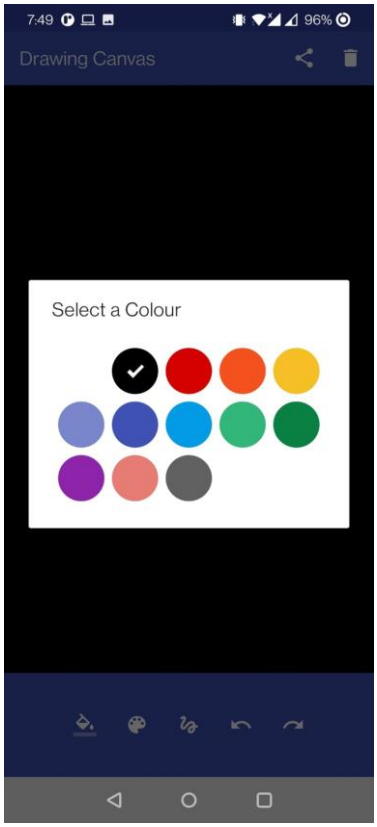
```
private void initDrawingView()
	{
		mCurrentBackgroundColor = ContextCompat.getColor(this,
android.R.color.white);
		mCurrentColor = ContextCompat.getColor(this, android.R.color.black);
		mCurrentStroke = 10;

		mDrawingView.setBackgroundColor(mCurrentBackgroundColor);
		mDrawingView.setPaintColor(mCurrentColor);
		mDrawingView.setPaintStrokeWidth(mCurrentStroke);
	}

	private void startFillBackgroundDialog()
	{
		int[] colors = getResources().getIntArray(R.array.palette);

		ColorPickerDialog dialog =
ColorPickerDialog.newInstance(R.string.color_picker_default_title,
				colors,
				mCurrentBackgroundColor,
				5,
				ColorPickerDialog.SIZE_SMALL);

		dialog.setOnColorSelectedListener(new
ColorPickerSwatch.OnColorSelectedListener()
		{

			@Override
			public void onColorSelected(int color)
			{
				mCurrentBackgroundColor = color;

	mDrawingView.setBackgroundColor(mCurrentBackgroundColor);
			}

		});

		dialog.show(getFragmentManager(), "ColorPickerDialog");
	}
```

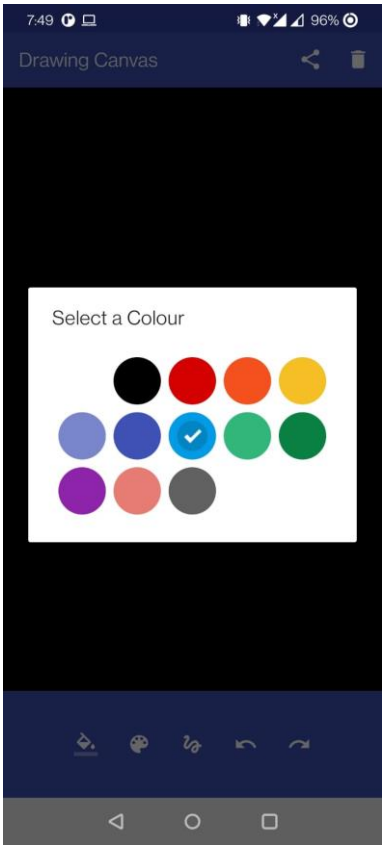
```
	private void startColorPickerDialog()
	{
		int[] colors = getResources().getIntArray(R.array.palette);

		ColorPickerDialog dialog =
ColorPickerDialog.newInstance(R.string.color_picker_default_title,
				colors,
				mCurrentColor,
				5,
				ColorPickerDialog.SIZE_SMALL);

		dialog.setOnColorSelectedListener(new
ColorPickerSwatch.OnColorSelectedListener()
		{

			@Override
			public void onColorSelected(int color)
			{
				mCurrentColor = color;
				mDrawingView.setPaintColor(mCurrentColor);
			}

		});

		dialog.show(getFragmentManager(), "ColorPickerDialog");
	}

	private void startStrokeSelectorDialog()
	{
		StrokeSelectorDialog dialog =
StrokeSelectorDialog.newInstance(mCurrentStroke, MAX_STROKE_WIDTH);

		dialog.setOnStrokeSelectedListener(new
StrokeSelectorDialog.OnStrokeSelectedListener()
		{
			@Override
			public void onStrokeSelected(int stroke)
			{
				mCurrentStroke = stroke;
				mDrawingView.setPaintStrokeWidth(mCurrentStroke);
			}
		});

		dialog.show(getSupportFragmentManager(), "StrokeSelectorDialog");
	}

	private void startShareDialog(Uri uri)
	{
		Intent intent = new Intent();
```

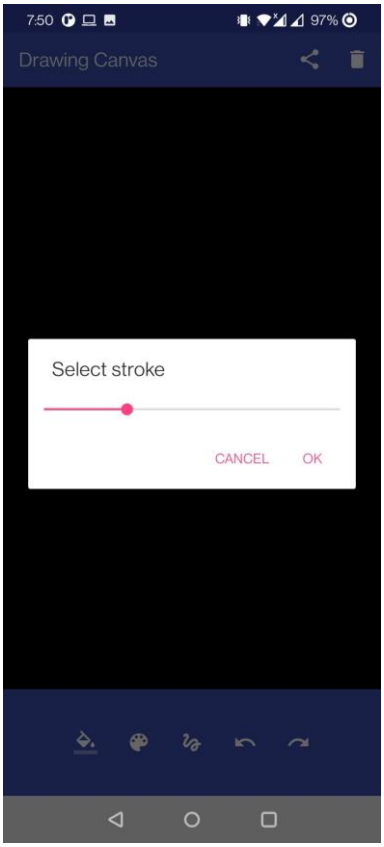
```
		intent.setAction(Intent.ACTION_SEND);
		intent.setType("image/*");

		intent.putExtra(android.content.Intent.EXTRA_SUBJECT, "");
		intent.putExtra(android.content.Intent.EXTRA_TEXT, "");
		intent.putExtra(Intent.EXTRA_STREAM, uri);
		intent.addFlags(Intent.FLAG_GRANT_READ_URI_PERMISSION);
		startActivity(Intent.createChooser(intent, "Share Image"));
	}

	private void requestPermissionsAndSaveBitmap()
	{
		if (PermissionManager.checkWriteStoragePermissions(this))
		{
			Uri uri = FileManager.saveBitmap(this, mDrawingView.getBitmap());
			startShareDialog(uri);
		}
	}

	@Override
	public void onRequestPermissionsResult(int requestCode, String[] permissions, int[]
grantResults)
	{
		super.onRequestPermissionsResult(requestCode, permissions, grantResults);
		switch (requestCode)
		{
			case PermissionManager.REQUEST_WRITE_STORAGE:
			{
				if (grantResults.length > 0 && grantResults[0] ==
PackageManager.PERMISSION_GRANTED)
				{
					Uri uri = FileManager.saveBitmap(this,
mDrawingView.getBitmap());
					startShareDialog(uri);
				} else
				{
					Toast.makeText(this, "The app was not allowed to write
to your storage. Hence, it cannot function properly. Please consider granting it this
permission", Toast.LENGTH_LONG).show();
				}
			}
		}
	}
```

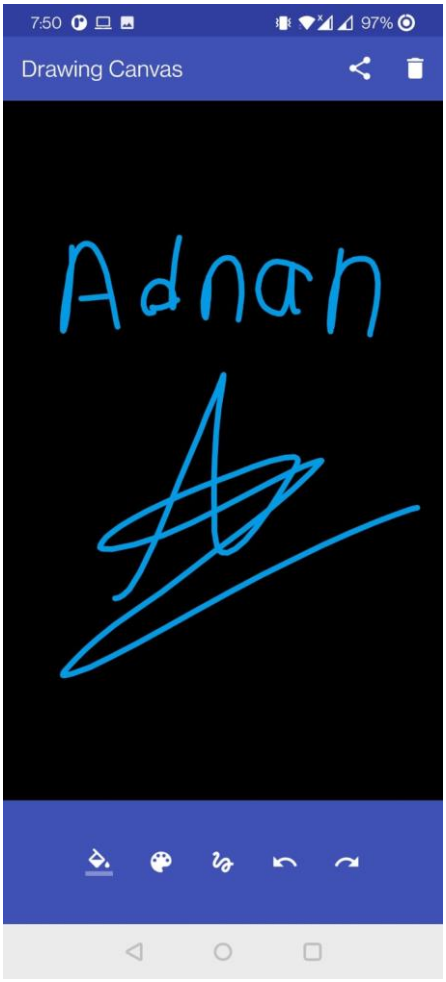
Output:


7:49
96%
Drawing Canvas


7:49
96%
Drawing Canvas
Select a Colour


7:49
96%
Drawing Canvas
Select a Colour


7:50
97%
Drawing Canvas
Select stroke
CANCEL
OK



Conclusion: We have successfully implemented Drawing App in Android.

Experiment no. 6

Aim: To implement GPS application in Android.

Requirements: Compatible version of android studio.

Theory:

Request location permissions

To protect user privacy, apps that use location services must request location permissions.

When you request location permissions, follow the same best practices as you would for any other runtime permission. One important difference when it comes to location permissions is that the system includes multiple permissions related to location. Which permissions you request, and how you request them, depend on the location requirements for your app's use case.

This page describes the different types of location requirements and provides guidance on how to request location permissions in each case.

Types of location access

Each permission has a combination of the following characteristics:

Category: Either foreground location or background location.

Accuracy: Either precise location or approximate location.

Foreground location

If your app contains a feature that shares or receives location information only once, or for a defined amount of time, then that feature requires foreground location access. Some examples include the following:

Within a navigation app, a feature allows users to get turn-by-turn directions.

Within a messaging app, a feature allows users to share their current location with another user.

The system considers your app to be using foreground location if a feature of your app accesses the device's current location in one of the following situations:

An activity that belongs to your app is visible.

Your app is running a foreground service. When a foreground service is running, the system raises user awareness by showing a persistent notification. Your app retains access when it's placed in the background, such as when the user presses the Home button on their device or turns their device's display off.

Additionally, it's recommended that you declare a foreground service type of location, as shown in the following code snippet. On Android 10 (API level 29) and higher, you must declare this foreground service type.

```
<!-- Recommended for Android 9 (API level 28) and lower. -->

<!-- Required for Android 10 (API level 29) and higher. -->

<service

    android:name="MyNavigationService"

    android:foregroundServiceType="location" ... >

    <!-- Any inner elements would go here. -->

</service>
```

You declare a need for foreground location when your app requests either the ACCESS_COARSE_LOCATION permission or the ACCESS_FINE_LOCATION permission, as shown in the following snippet:

```
<manifest ... >
  <!-- Always include this permission -->
  <uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
  <!-- Include only if your app benefits from precise location access. -->
  <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
</manifest>
```

Background location

An app requires background location access if a feature within the app constantly shares location with other users or uses the Geofencing API. Several examples include the following:

Within a family location sharing app, a feature allows users to continuously share location with family members.

Within an IoT app, a feature allows users to configure their home devices such that they turn off when the user leaves their home and turn back on when the user returns home.

The system considers your app to be using background location if it accesses the device's current location in any situation other than the ones described in the foreground location section. The background location precision is the same as the foreground location precision, which depends on the location permissions that your app declares.

On Android 10 (API level 29) and higher, you must declare the ACCESS_BACKGROUND_LOCATION permission in your app's manifest in order to request background location access at runtime. On earlier versions of Android, when your app receives foreground location access, it automatically receives background location access as well.

```
<manifest ... >
  <!-- Required only when requesting background location access on
       Android 10 (API level 29) and higher. -->
  <uses-permission
android:name="android.permission.ACCESS_BACKGROUND_LOCATION" />
</manifest>
```

Accuracy

Android supports the following levels of location accuracy:

Approximate

Provides a device location estimate. If this location estimate is from the LocationManagerService or FusedLocationProvider, this estimate is accurate to within about 3 square kilometers (about 1.2 square miles). Your app can receive locations at this level of accuracy when you declare the ACCESS_COARSE_LOCATION permission but not the ACCESS_FINE_LOCATION permission.

Precise

Provides a device location estimate that is as accurate as possible. If the location estimate is from LocationManagerService or FusedLocationProvider, this estimate is usually within about 50 meters (160 feet) and is sometimes as accurate as within a few meters (10 feet) or better. Your app can receive locations at this level of accuracy when you declare the ACCESS_FINE_LOCATION permission.

If the user grants the approximate location permission, your app only has access to approximate location, regardless of which location permissions your app declares.

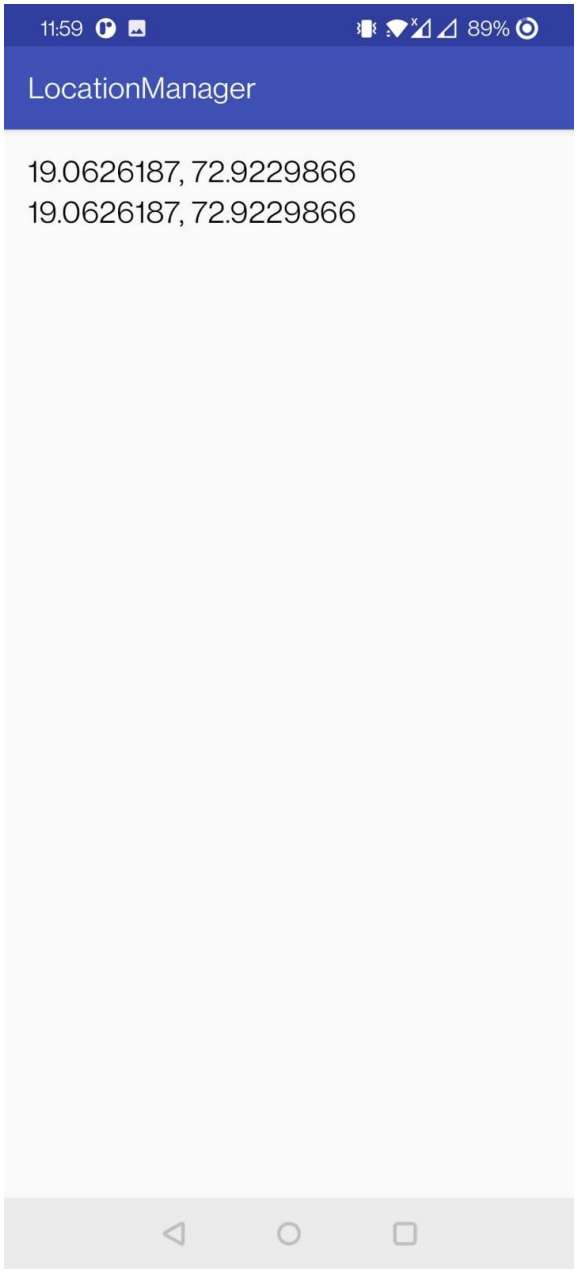
Your app should still work when the user grants only approximate location access. If a feature in your app absolutely requires access to precise location using the ACCESS_FINE_LOCATION permission, you can ask the user to allow your app to access precise location.

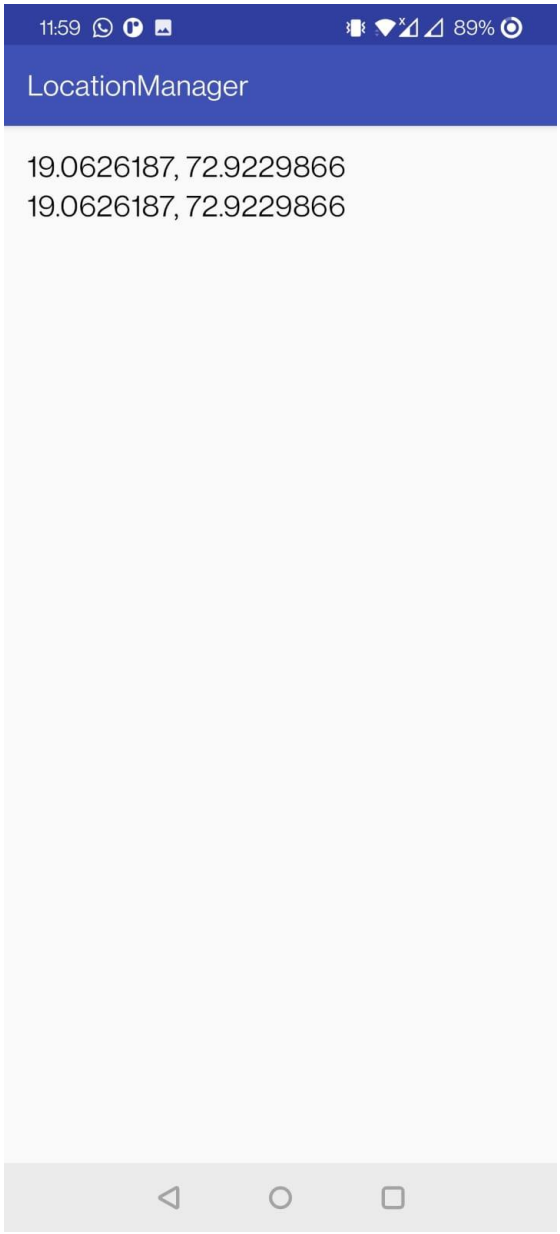
Code:

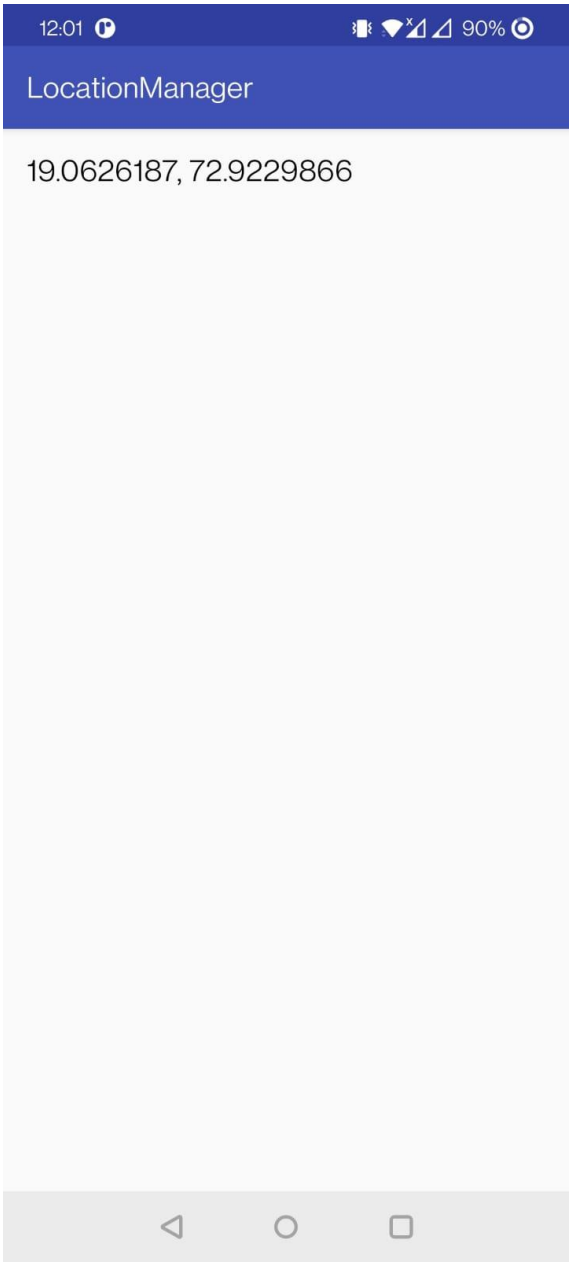
```
package com.yayandroid.locationmanager.sample;

import android.content.Intent;
import android.os.Bundle;
import android.view.View;

import androidx.annotation.Nullable;
import androidx.appcompat.app.AppCompatActivity;

import com.yayandroid.locationmanager.sample.activity.SampleActivity;
import com.yayandroid.locationmanager.sample.fragment.SampleFragmentActivity;
import com.yayandroid.locationmanager.sample.service.SampleServiceActivity;

public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(@Nullable Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }

    public void inActivityClick(View view) {
        startActivity(new Intent(this, SampleActivity.class));
    }

    public void inFragmentClick(View view) {
        startActivity(new Intent(this, SampleFragmentActivity.class));
    }

    public void inServiceClick(View view) {
        startActivity(new Intent(this, SampleServiceActivity.class));
    }
}
```

Output:


11:59
89%
LocationManager
SAMPLE IN ACTIVITY
SAMPLE IN FRAGMENT
SAMPLE IN SERVICE


11:59
89%
LocationManager
19.0626187, 72.9229866
19.0626187, 72.9229866





Conclusion: We have successfully implemented Android GPS application using Android Studio.

Experiment No. 7

Aim: To implement notification application in Java and Android.

Requirements: Compatible version of Java, Android Studio and Windows (Supports System Tray).

Theory:

Notification

A notification is a message that Android displays outside your app's UI to provide the user with reminders, communication from other people, or other timely information from your app. Users can tap the notification to open your app or take an action directly from the notification.

Notification anatomy

The design of a notification is determined by system templates—your app simply defines the contents for each portion of the template. Some details of the notification appear only in the expanded view.

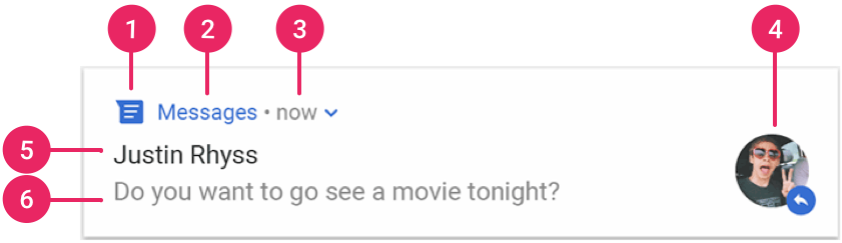


Figure 7. A notification with basic details

The most common parts of a notification are indicated in figure 7 as follows:

1. Small icon: This is required and set with setSmallIcon().
2. App name: This is provided by the system.

3. Time stamp: This is provided by the system but you can override with setWhen() or hide it with setShowWhen(false).

4. Large icon: This is optional (usually used only for contact photos; do not use it for your app icon) and set with setLargeIcon().

5. Title: This is optional and set with setContentTitle().

6. Text: This is optional and set with setContentText().

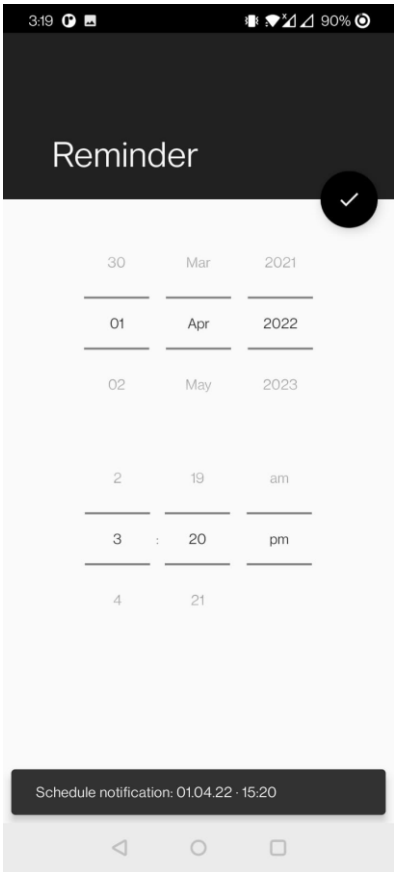
Android Implementation

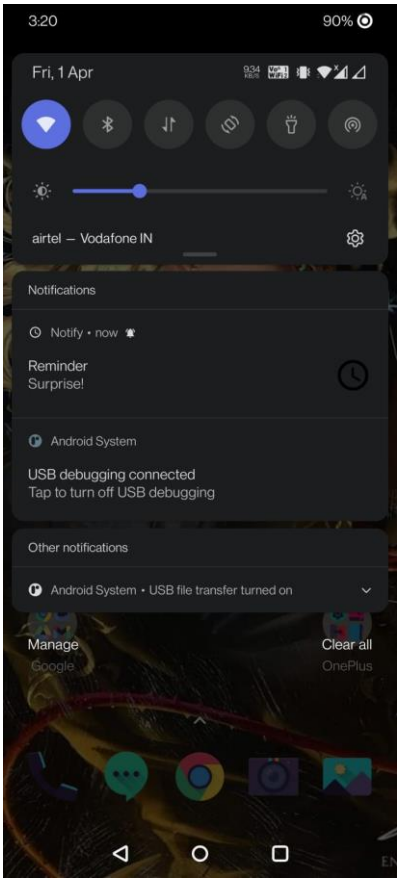
Code:

```
class MainActivity : AppCompatActivity() {

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)

        userInterface()
    }

    private fun userInterface() {
        setSupportActionBar(toolbar)

        val titleNotification = getString(R.string.notification_title)
        collapsing_toolbar_l.title = titleNotification

        done_fab.setOnClickListener {
            val customCalendar = Calendar.getInstance()
            customCalendar.set(
                date_p.year, date_p.month, date_p.dayOfMonth, time_p.hour, time_p.minute, 0
            )
            val customTime = customCalendar.timeInMillis
            val currentTime = currentTimeMillis()
            if (customTime > currentTime) {
                val data = Data.Builder().putInt(NOTIFICATION_ID, 0).build()
                val delay = customTime - currentTime
                scheduleNotification(delay, data)

                val titleNotificationSchedule = getString(R.string.notification_schedule_title)
                val patternNotificationSchedule = getString(R.string.notification_schedule_pattern)
                make(
```

```
                coordinator_l,
                titleNotificationSchedule + SimpleDateFormat(
                    patternNotificationSchedule, getDefault()
                ).format(customCalendar.time).toString(),
                LENGTH_LONG
            ).show()
        } else {
            val errorNotificationSchedule = getString(R.string.notification_schedule_error)
            make(coordinator_l, errorNotificationSchedule, LENGTH_LONG).show()
        }
    }
}

private fun scheduleNotification(delay: Long, data: Data) {
    val notificationWork = OneTimeWorkRequest.Builder(NotifyWork::class.java)
        .setInitialDelay(delay, MILLISECONDS).setInputData(data).build()

    val instanceWorkManager = WorkManager.getInstance(this)
    instanceWorkManager.beginUniqueWork(NOTIFICATION_WORK,          REPLACE,
notificationWork).enqueue()
  }
}
```

Output:



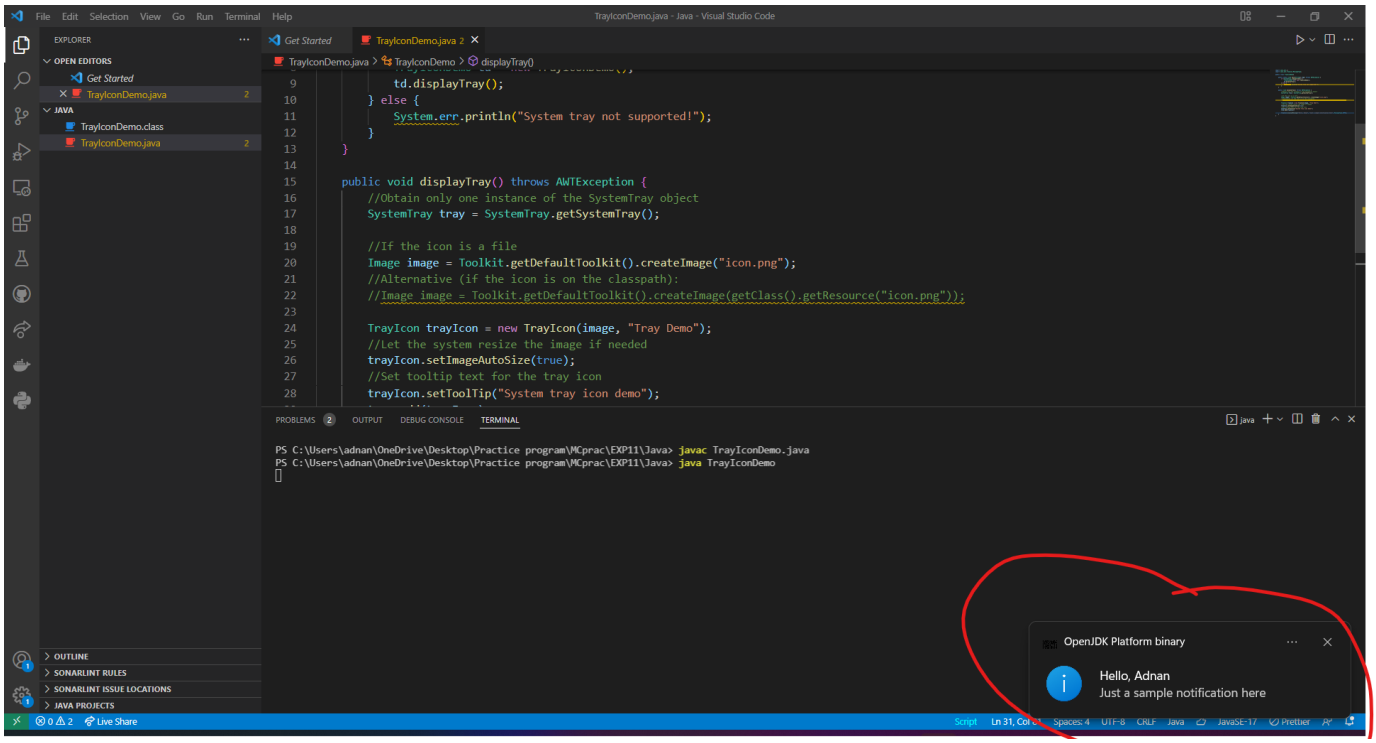


Java Implementation

Code:

```
import java.awt.*;
import java.awt.TrayIcon.MessageType;

public class TrayIconDemo{

   public static void main(String[] args) throws AWTException {
      if (SystemTray.isSupported()) {
         TrayIconDemo td = new TrayIconDemo();
         td.displayTray();
      } else {
         System.err.println("System tray not supported!");
      }
   }

   public void displayTray() throws AWTException {
      //Obtain only one instance of the SystemTray object
      SystemTray tray = SystemTray.getSystemTray();

      //If the icon is a file
      Image image = Toolkit.getDefaultToolkit().createImage("icon.png");
      //Alternative (if the icon is on the classpath):
      //Image image = Toolkit.getDefaultToolkit().createImage(getClass().getResource("icon.png"));

      TrayIcon trayIcon = new TrayIcon(image, "Tray Demo");
      //Let the system resize the image if needed
      trayIcon.setImageAutoSize(true);
      //Set tooltip text for the tray icon
      trayIcon.setToolTip("System tray icon demo");
      tray.add(trayIcon);

      trayIcon.displayMessage("Hello, Adnan", "Just a sample notification here", MessageType.INFO);
   }
}
```

Output:



Conclusion: We have successfully implemented Notification application in Java and Android Studio.

Experiment No. 8

Aim: To implement Inventory Tracking App in Android.

Requirements: Compatible version of Android Studio.

Theory:

This app is an Inventory tracking app. Demos how to add, update, sell, and delete items from the local database.

This app demonstrated

Room database

Apps that handle non-trivial amounts of structured data can benefit greatly from persisting that data locally. The most common use case is to cache relevant pieces of data so that when the device cannot access the network, the user can still browse that content while they are offline.

The Room persistence library provides an abstraction layer over SQLite to allow fluent database access while harnessing the full power of SQLite. In particular, Room provides the following benefits:

- Compile-time verification of SQL queries.
- Convenience annotations that minimize repetitive and error-prone boilerplate code.
- Streamlined database migration paths.

View Model

Architecture Components provides ViewModel helper class for the UI controller that is responsible for preparing data for the UI. ViewModel objects are automatically retained during

configuration changes so that data they hold is immediately available to the next activity or fragment instance.

Flow

A flow is very similar to an Iterator that produces a sequence of values, but it uses suspend functions to produce and consume values asynchronously. This means, for example, that the flow can safely make a network request to produce the next value without blocking the main thread.

View Binding

View binding is a feature that allows you to more easily write code that interacts with views. Once view binding is enabled in a module, it generates a *binding class* for each XML layout file present in that module. An instance of a binding class contains direct references to all views that have an ID in the corresponding layout.

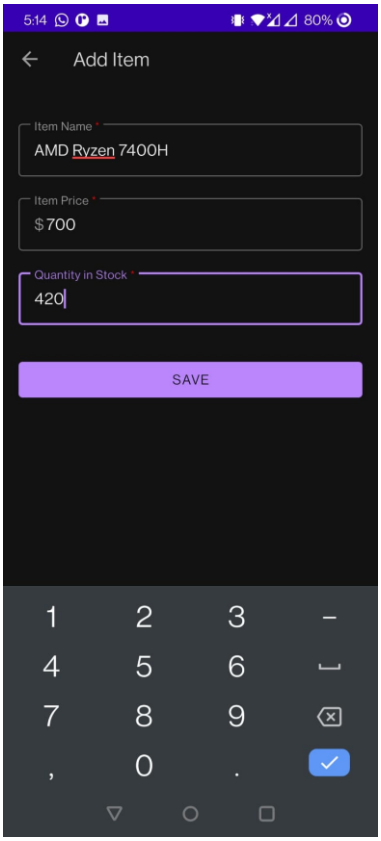
Navigation

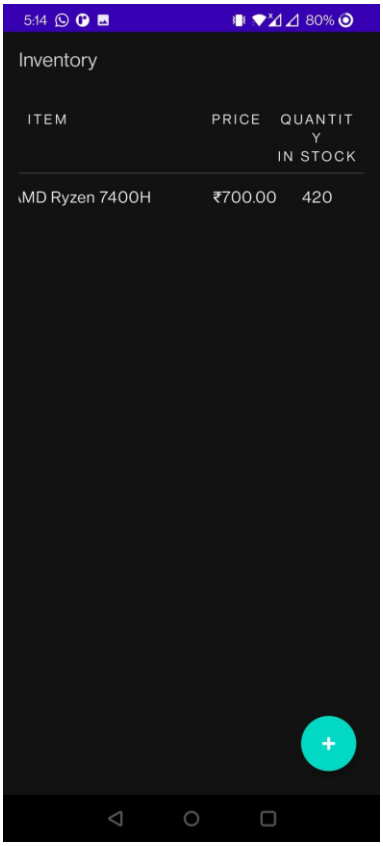
Navigation refers to the interactions that allow users to navigate across, into, and back out from the different pieces of content within your app. Android Jetpack's Navigation component helps you implement navigation, from simple button clicks to more complex patterns, such as app bars and the navigation drawer. The Navigation component also ensures a consistent and predictable user experience by adhering to an established set of principles.

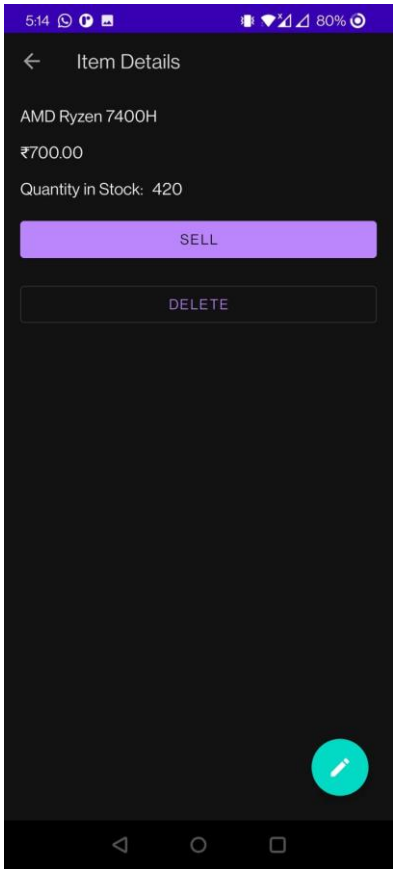
Code:

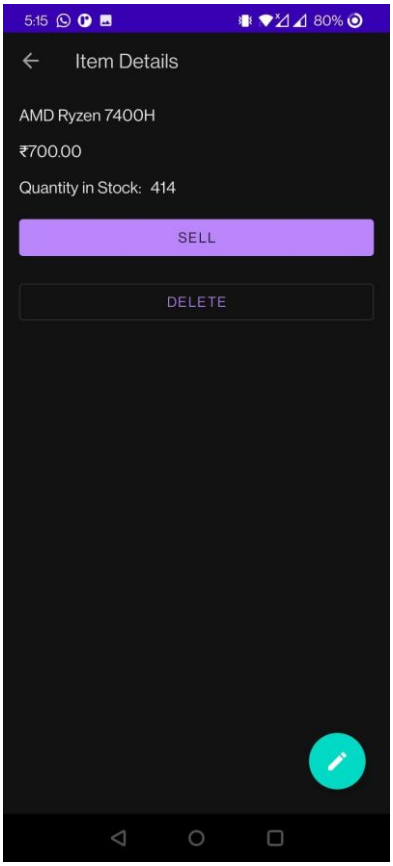
```
class MainActivity : AppCompatActivity(R.layout.activity_main) {

    private lateinit var navController: NavController

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)

        // Retrieve NavController from the NavHostFragment
        val navHostFragment = supportFragmentManager
            .findFragmentById(R.id.nav_host_fragment) as NavHostFragment
        navController = navHostFragment.navController
        // Set up the action bar for use with the NavController
        setupActionBarWithNavController(this, navController)
    }

    /**
     * Handle navigation when the user chooses Up from the action bar.
     */
    override fun onSupportNavigateUp(): Boolean {
        return navController.navigateUp() || super.onSupportNavigateUp()
    }
}
```

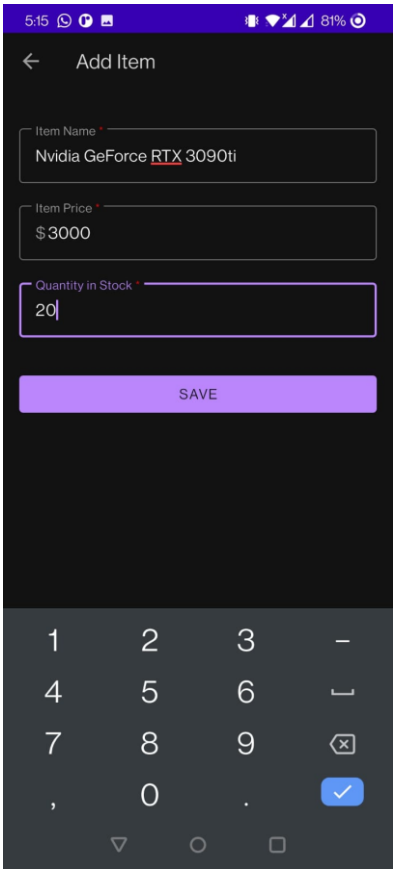
Output:

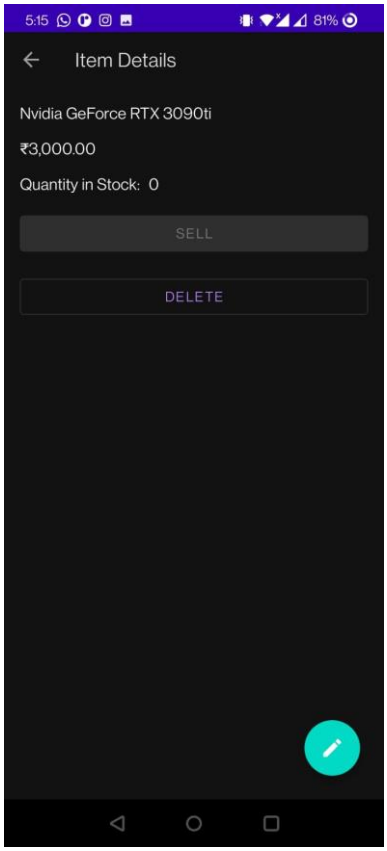












Conclusion: We have successfully implemented Inventory Tracking App in Android.

Experiment No. 9

Aim: To implement income tax and EMI calculator

Requirements: Compatible version of Android Studio, Gradle and Java.

Theory:

What is income tax?

According to the Income Tax Act, 1961, every salaried person needs to pay an amount from their salary as tax to the country. This amount of tax is called the income tax. The law consists of a lot of amendments and variations with subsections describing the details about tax payment, deductions, and computations. A lot of deductions from subsection 80C to 80U are available. The final amount after subtracting all the available tax-saving provisions and deductions is given to the government as the income tax on salary.

Income Source	Description
Income from Salary	All income you receive from your job like salary, leave encashment, allowances and so on.
Income from Property	Income from house or land (rented or self-occupied)
Income from Business/Profession	Earnings form part-time job or profession

Income from Gains	Earnings from the sale of a capital asset
Income from other sources	Residual income like earnings from the fixed deposit, gifts, pension, etc.

The EMI of a loan depends on three factors:

Loan amount – This stands for the total amount that has been borrowed the individual.

Interest rate – This stands for the rate at which the interest is charged on the amount borrowed. Tenure of loan – This stands for the agreed loan repayment time-frame between the borrower and the lender.

How is EMI calculated?

The mathematical formula to calculate EMI is:

$$EMI = \frac{P \times r \times (1 + r)n}{(1 + r)n - 1}$$

Where, $P = Loan\ amount, r = interest\ rate, n = tenure\ in\ number\ of\ months.$

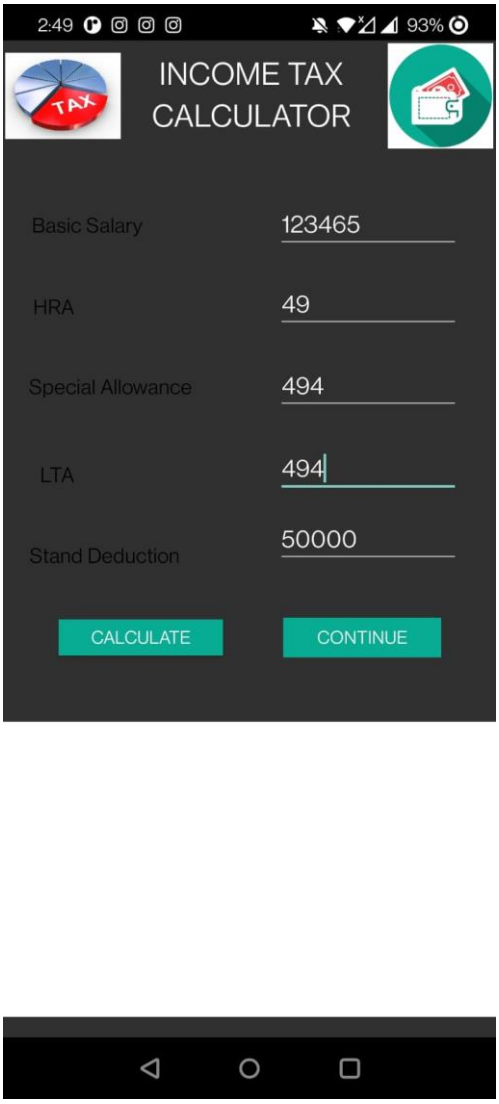
Code:

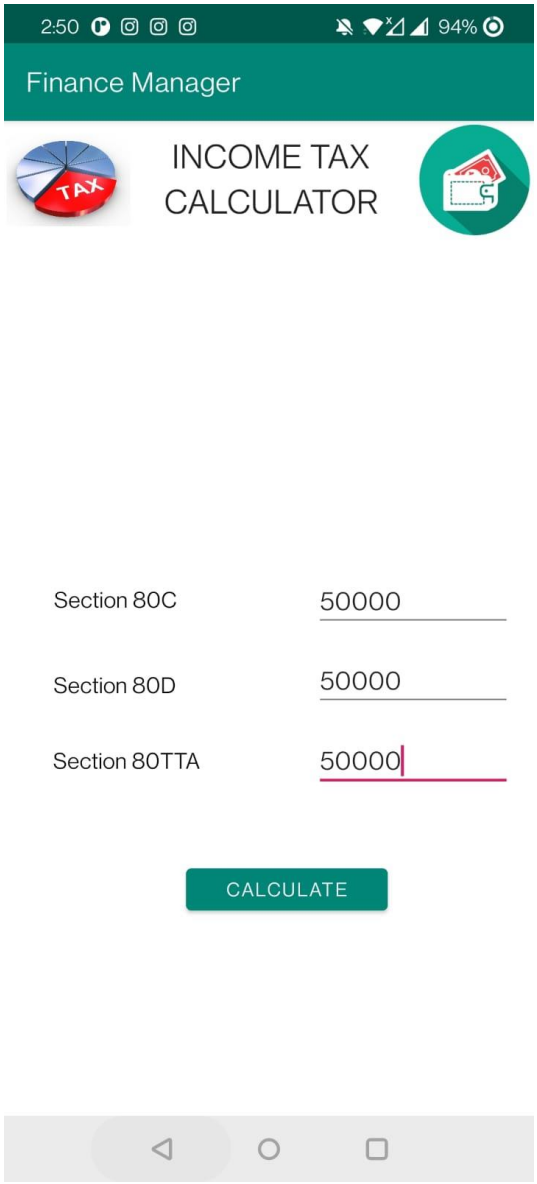
```
package com.cmt.taxcalculator;

import android.content.Intent;
import androidx.appcompat.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.TextView;

public class MainActivity extends AppCompatActivity {
    TextView t;
    Button b1;
```

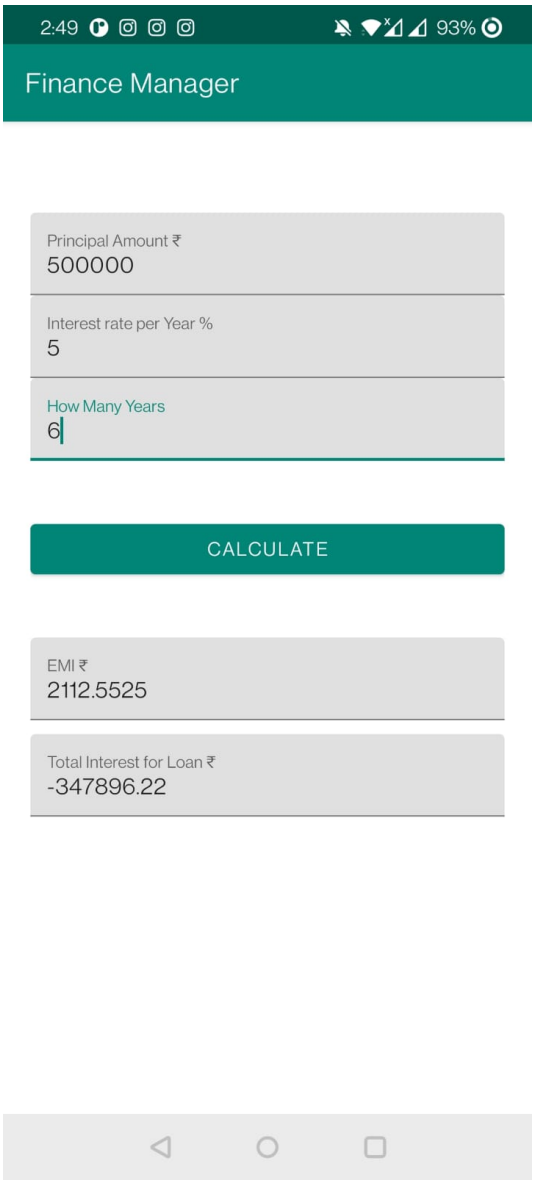
```
    Button b2;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);// To Display The Layout from activy_main
resource layout file.
        setContentView(R.layout.activity_main);
        b1 = findViewById(R.id.button1);
        b2 = findViewById(R.id.button2);

        b1.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                Intent intent= new Intent(MainActivity.this,
com.cmt.taxcalculator.IncomeActivity.class);
                startActivity(intent); // On clicking on the button, Income Tax Calculator activity is
called
            }
        });

        b2.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                Intent intent=new Intent(MainActivity.this,
com.cmt.taxcalculator.EMIActivity.class);
                startActivity(intent);// On clicking on the button, EMI activity is called
            }
        });
    }

    public void exitApp() {
        finish();
    }

}
```

Output:


2:42
91%
Finance Manager
FINANCE MANAGER
INCOME TAX CALCULATOR
EMI CALCULATOR
Home
Info
Developer


2:49
93%
INCOME TAX CALCULATOR
Basic Salary
123465
HRA
49
Special Allowance
494
LTA
494
Stand Deduction
50000
CALCULATE
CONTINUE





Conclusion: We have successfully implemented EMI and Tax calculator in Android.

Experiment No 10

Aim: Illustration of Hidden Terminal Problem (NS-2)

Theory:

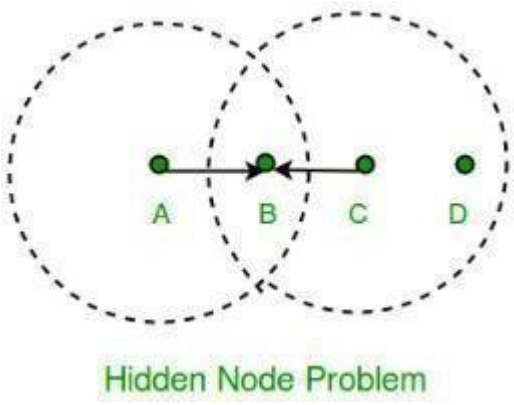
A wireless network with lack of centralized control entity, sharing of wireless bandwidth among network access nodes i.e. medium access control (MAC) nodes must be organized in decentralized manner. The hidden terminal problem occurs when a terminal is visible from a wireless access point (APs), but not from other nodes communicating with that AP. This situation leads the difficulties in medium access control sublayer over wireless networking.

In a formal way hidden terminal are nodes in a wireless network that are out of range of other node or a collection of nodes. Consider a wireless networking, each node at the far edge of the access point's range, which is known as A, can see the access point, but it is unlikely that the same node can see a node on the opposite end of the access point's range, C. These nodes are known as hidden. The problem is when nodes A and C start to send packets simultaneously to the access point B. Because the nodes A and C are out of range of each other and so cannot detect a collision while transmitting, Carrier sense multiple access with collision detection (CSMA/CD) does not work, and collisions occur, which then corrupt the data received by the access point. To overcome the hidden node problem, RTS/CTS handshaking (IEEE 802.11 RTS/CTS) is implemented in conjunction with the Carrier sense multiple accesses with collision avoidance (CSMA/CA) scheme. The same problem exists in a MANET.

The transmission range of access point A reaches at B, but not at access point C, similarly transmission range of access point C reaches B, but not at A. These nodes are known as hidden terminals. The problem occurs when nodes A and C start to send data packets simultaneously to the access point B. Because the access points A and C are out of range of each other and resultant they cannot detect a collision while transmitting, Carrier sense multiple access with collision detection (CSMA/CD) does not work, and collisions occur, which then corrupt the data received by the access point B due to the hidden terminal problem.

The hidden terminal analogy is described as follows:

- Terminal A sends data to B, terminal C cannot hear A
- Terminal C wants to send data to B, terminal C senses a "free" medium (CS fails) and starts transmitting
- Collision at B occurs, A cannot detect this collision (CD fails) and continues with its transmission to B
- Terminal A is "hidden" from C and vice versa.



The solution of hidden terminal problem is as follows.

When A wants to send a packet to B, A first sends a Request-to-send (RTS) to B. On receiving RTS, B responds by sending Clear-to-Send (CTS). When C overhears a CTS, it keeps quiet for the duration of the transfer. Transfer duration is included in both RTS and CTS. RTS and CTS are short frames, reduces collision chance.

Code:

```
BEGIN{
   sim
   _en
   d =
   200;
   i=0;
   while (i<=sim_end) {sec[i]=0; i+=1;};
}


{
if ($1=="r" && $7=="cbr"&&
   $3=="_0_") { sec[int($2)]+=$8;
   };
}


END{
   i=0;
   while (i<=sim_end) {print i " " sec[i]; i+=1;};
}# Define options

set val(chan)   Channel/WirelessChannel ;# channel type
set val(prop)               Propagation/FreeSpace ;# radio-
propagation model set val(netif) Phy/WirelessPhy
        ;# network interface type set val(mac)
        Mac/802_11        ;# MAC type
set val(ifq)                Queue/DropTail/PriQueue ;# interface
queue type set val(ll)    LL        ;# link layer type
set val(ant)     Antenna/OmniAntenna     ;# antenna
model set val(ifqlen)               10000               ;#
```

```
max packet in ifq set val(nn)      5          ;# number
of mobilenodes
set val(rp)       DSR      ;# routing protocol
set val(x)                              600
                  ;# X dimension of topography set
val(y)            600                  ;# Y dimension
of topography set val(stop)        100       ;# time
of simulation end
set val(R)        300
set opt(tr)       out.tr
set ns [new
Simulator] set
tracefd [open
$opt(tr) w]
set windowVsTime2 [open
win.tr w] set namtrace
        [open simwrls.nam
w] Mac/802_11 set
dataRate_         1.2e6
Mac/802_11 set
RTSThreshold_ 100
$ns trace-
all
$tracefd
#$ns use-
newtrace
$ns namtrace-all-wireless $namtrace $val(x) $val(y)

# set up topography object
set topo          [new Topography]

$topo load_flatgrid

$val(x) $val(y) create-

god $val(nn)

#
# Create nn mobilenodes [$val(nn)] and attach them to
the channel. #

# configure the nodes

    $ns node-config -adhocRouting $val(rp) \
          -llType $val(ll) \
          -macType $val(mac) \
          -ifqType $val(ifq) \
          -ifqLen $val(ifqlen) \
          -antType $val(ant) \
          -propType $val(prop) \
          -phyType $val(netif) \
          -channelType $val(chan) \
          -topoInstance $topo \
          -agentTrace ON \
          -routerTrace ON \
          -macTrace ON \
          -movementTrace ON
```

Phy/Wireles Phy

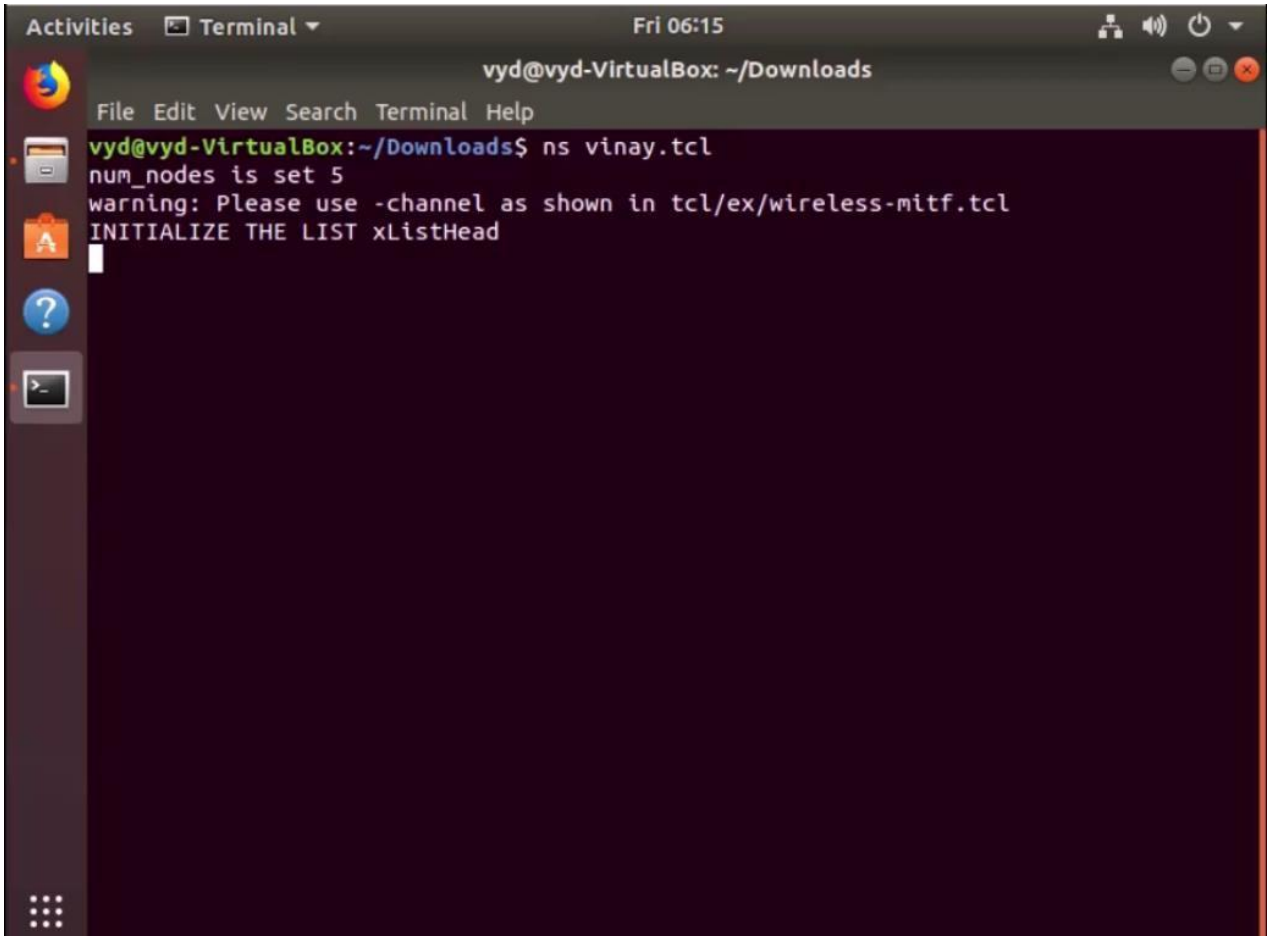
setCSThresh 30.5e-10

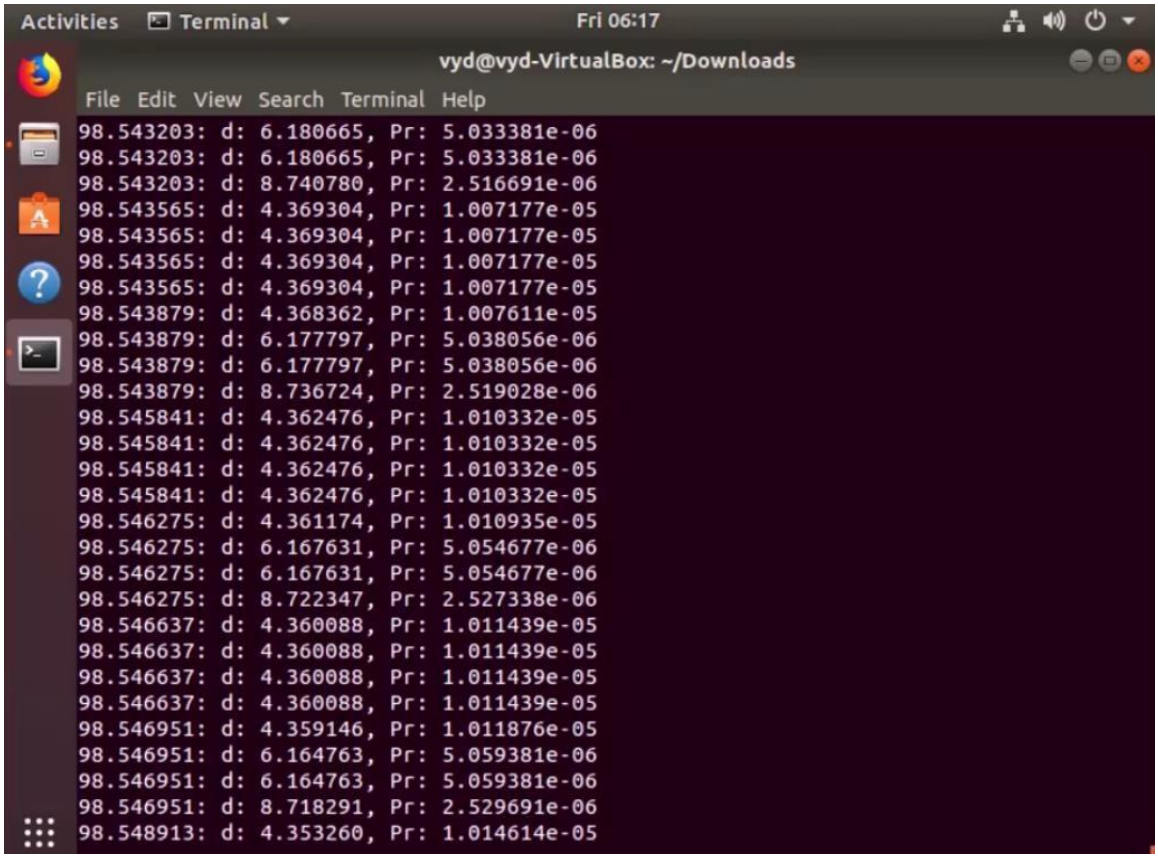
```
for {set i 0} {$i < $val(nn) } {
   incr i } { set node_($i) [$ns
   node]
   }
$node_(0) set X_ $val(R)
$node_(0) set Y_ $val(R)
$node_(0) set Z_ 0
$node_(1) set X_ $val(R)
$node_(1) set Y_ 0
$node_(1) set Z_ 0
$node_(2) set X_ 0
$node_(2) set Y_ $val(R)
$node_(2) set Z_ 0
$node_(3) set X_ [expr $val(R) *2]
$node_(3) set Y_ $val(R)
$node_(3) set Z_ 0
$node_(4) set X_ $val(R)
$node_(4) set Y_ [expr $val(R) *2]
$node_(4) set Z_ 0
 for {set i 0} {$i<$val(nn)} {incr i} {
  $ns initial_node_pos $node_($i) 30
 }
# Generation of movements
 $ns at 0 "$node_(1) setdest $val(R) $val(R) 3.0"
$ns at 0 "$node_(2) setdest $val(R) $val(R) 3.0"
$ns at 0 "$node_(3) setdest $val(R) $val(R) 3.0"
$ns at 0 "$node_(4) setdest $val(R) $val(R) 3.0"
# Set a TCP connection between node_(0)
and node_(1) set tcp [new
Agent/TCP/Newreno]
#$tcp set class_ 2
set tcp [new Agent/UDP]
$tcp set class_ 2
set sink [new Agent/Null]
$ns attach-agent $node_(1) $tcp
$ns attach-agent $node_(0) $sink
$ns connect $tcp $sink
set ftp [new Application/Traffic/CBR]
$ftp attach-agent $tcp
$ns at 0.0 "$ftp start"
# ################################################
# For coloring but doesnot work
# ################################################
$tcp set fid_ 1
$ns color 1 blue
#/////////////////////////////////
//////////////// set tcp [new
Agent/UDP]
$tcp set class_ 2
set sink [new Agent/Null]
$ns attach-agent $node_(2) $tcp
$ns attach-agent $node_(0) $sink
$ns connect $tcp $sink
```

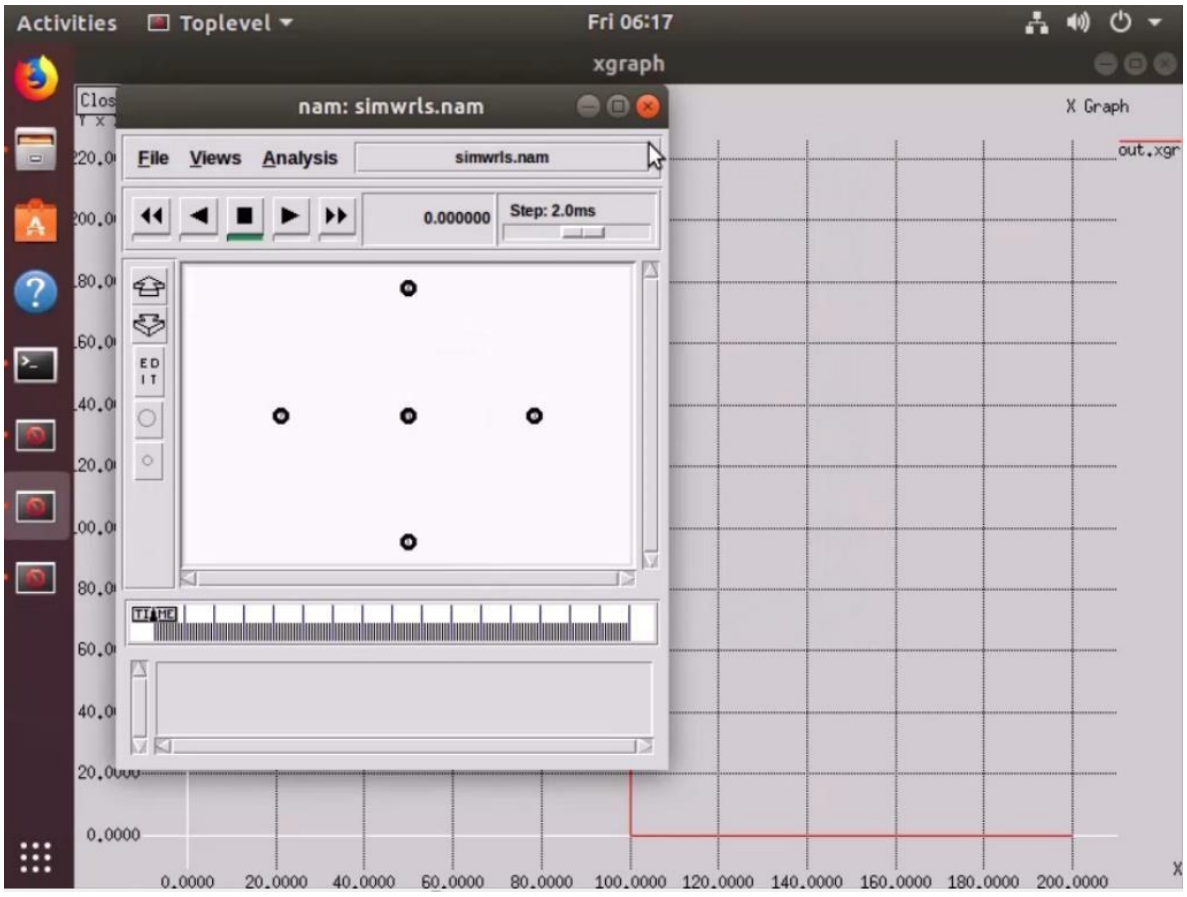
```
set ftp [new Application/Traffic/CBR]
$ftp attach-agent $tcp
$ns at 0.0 "$ftp start"
set tcp [new Agent/UDP]

$tcp set class_ 2
set sink [new Agent/Null]
$ns attach-agent $node_(3) $tcp
$ns attach-agent $node_(0) $sink
$ns connect $tcp $sink
set ftp [new Application/Traffic/CBR]
$ftp attach-agent $tcp
$ns at 0.0 "$ftp start"
set tcp [new Agent/UDP]
$tcp set class_ 2
set sink [new Agent/Null]
$ns attach-agent $node_(4) $tcp
$ns attach-agent $node_(0) $sink
$ns connect $tcp $sink
set ftp [new Application/Traffic/CBR]
$ftp attach-agent $tcp
$ns at 0.0 "$ftp start"
# Telling nodes when the
simulation ends #for {set i 0}
{$i < $val(nn) } { incr i } {
#   $ns at $val(stop)
"$node_($i) reset"; #}

# ending nam and the simulation
$ns at $val(stop) "$ns nam-end-wireless $val(stop)"
$ns at $val(stop) "stop"
$ns at $val(stop) "puts \"end
simulation\" ; $ns halt" proc stop {} {
 exec awk -f fil.awk
out.tr > out.xgr exec
xgraph out.xgr &


    global ns tracefd namtrace
    $ns
   flush
   -
   trace
   clos
   e
   $trac
   efd
   clos
   e
   $na
   mtra
   ce
    exec nam simwrls.nam &
}

$ns run
```

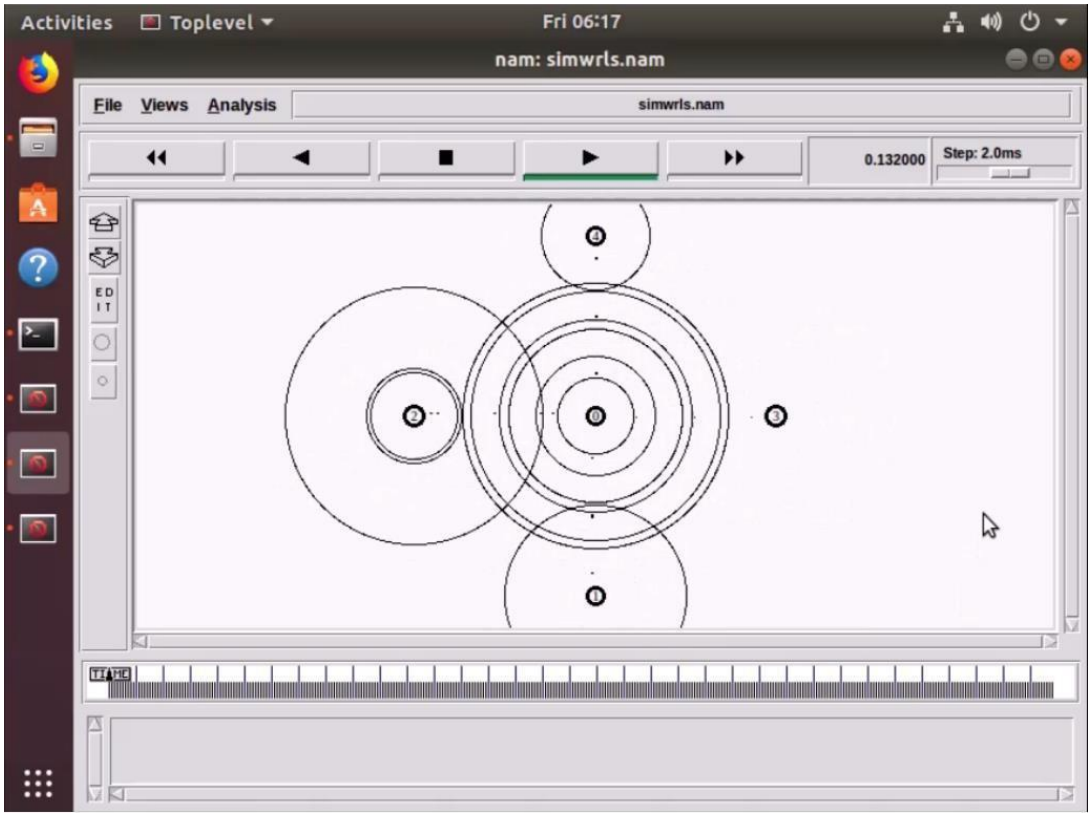
Output:

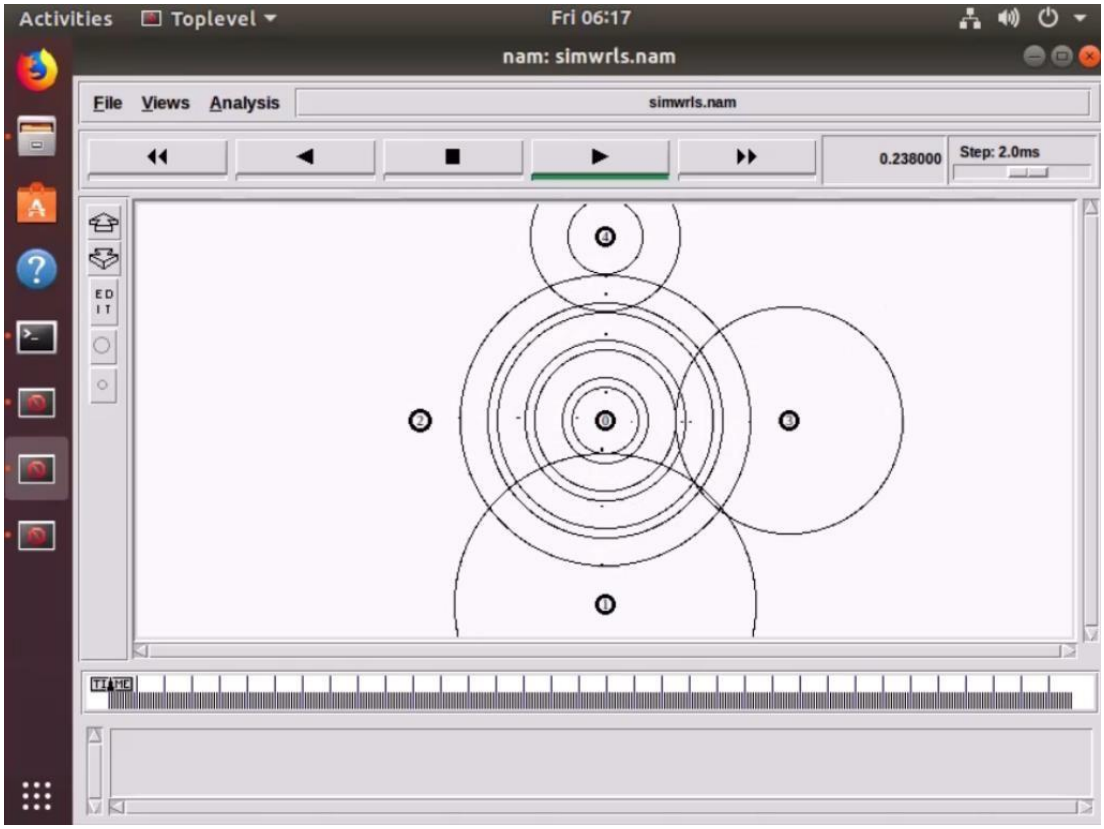
1. The node 0 and 2 want to send data to node 1 the range of node 0 and 2 is limited to 1 they do not know that other node is also sending data to 1 and therefore collision occurs.

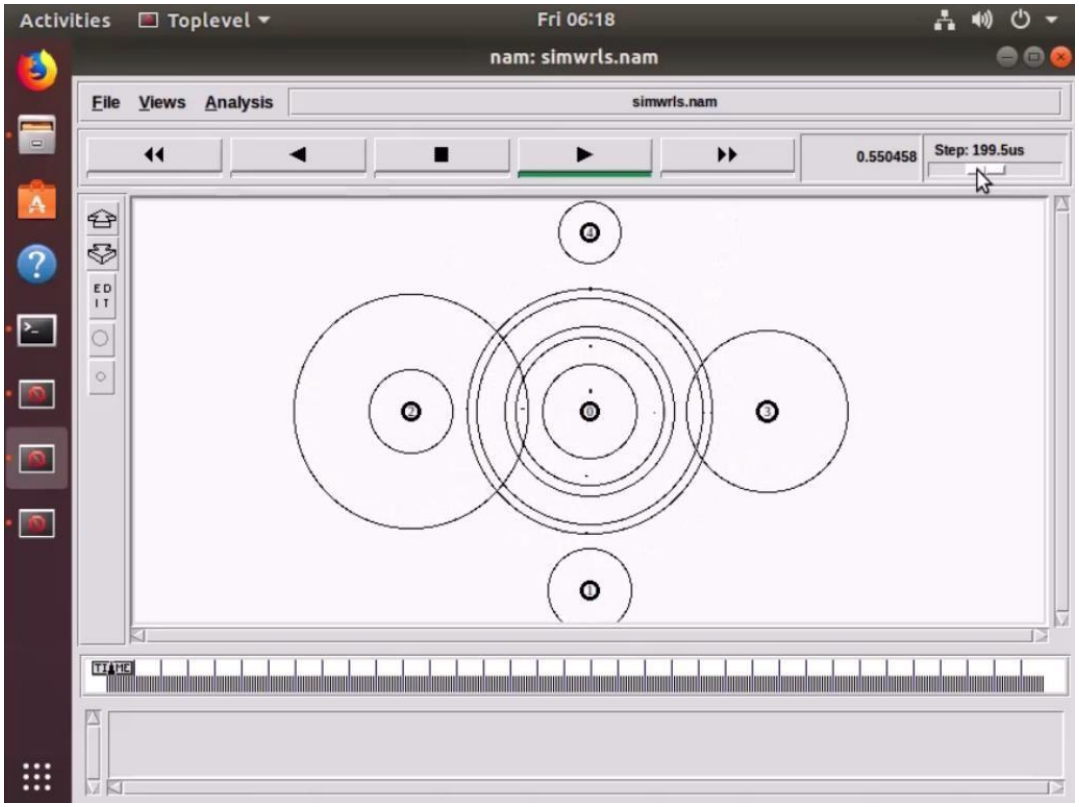


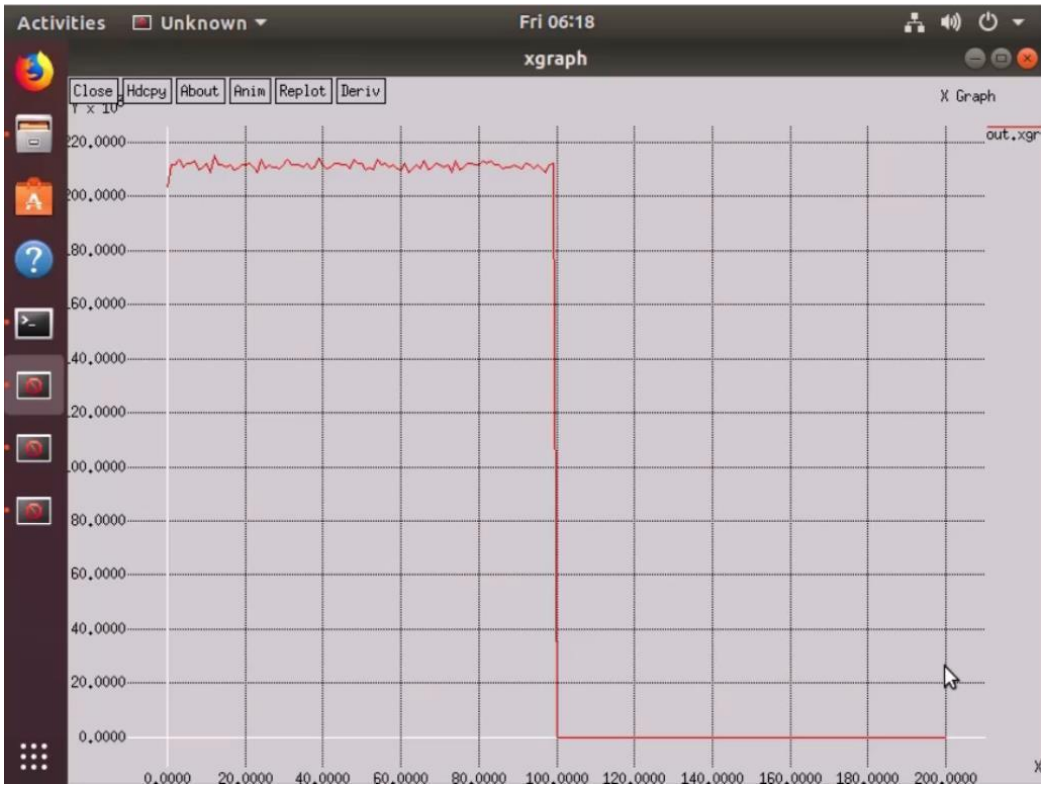

Activities
Terminal
Fri 06:17
vyd@vyd-VirtualBox: ~/Downloads
File Edit View Search Terminal Help
98.543203: d: 6.180665, Pr: 5.033381e-06
98.543203: d: 6.180665, Pr: 5.033381e-06
98.543203: d: 8.740780, Pr: 2.516691e-06
98.543565: d: 4.369304, Pr: 1.007177e-05
98.543565: d: 4.369304, Pr: 1.007177e-05
98.543565: d: 4.369304, Pr: 1.007177e-05
98.543565: d: 4.369304, Pr: 1.007177e-05
98.543879: d: 4.368362, Pr: 1.007611e-05
98.543879: d: 6.177797, Pr: 5.038056e-06
98.543879: d: 6.177797, Pr: 5.038056e-06
98.543879: d: 8.736724, Pr: 2.519028e-06
98.545841: d: 4.362476, Pr: 1.010332e-05
98.545841: d: 4.362476, Pr: 1.010332e-05
98.545841: d: 4.362476, Pr: 1.010332e-05
98.545841: d: 4.362476, Pr: 1.010332e-05
98.546275: d: 4.361174, Pr: 1.010935e-05
98.546275: d: 6.167631, Pr: 5.054677e-06
98.546275: d: 6.167631, Pr: 5.054677e-06
98.546275: d: 8.722347, Pr: 2.527338e-06
98.546637: d: 4.360088, Pr: 1.011439e-05
98.546637: d: 4.360088, Pr: 1.011439e-05
98.546637: d: 4.360088, Pr: 1.011439e-05
98.546637: d: 4.360088, Pr: 1.011439e-05
98.546951: d: 4.359146, Pr: 1.011876e-05
98.546951: d: 6.164763, Pr: 5.059381e-06
98.546951: d: 6.164763, Pr: 5.059381e-06
98.546951: d: 8.718291, Pr: 2.529691e-06
98.548913: d: 4.353260, Pr: 1.014614e-05


Activities
Toplevel
Fri 06:17
xgraph
nam: simwrls.nam
File Views Analysis
simwrls.nam
0.000000
Step: 2.0ms
X Graph
out.xgr
0.0000
0.0000
20.0000
40.0000
60.0000
80.0000
100.0000
120.0000
140.0000
160.0000
180.0000
200.0000


Activities
Toplevel
Fri 06:17
nam: simwrls.nam
File
Views
Analysis
simwrls.nam
0.132000
Step: 2.0ms
EDIT


Activities
Toplevel
Fri 06:17
nam: simwrls.nam
File
Views
Analysis
simwrls.nam
0.238000
Step: 2.0ms
EDIT





Conclusion: Thus, we have performed the experiment of and illustrated the hidden terminal problem using NS2 and properly explained the same which helps to understand better

Assignment no. 1

Q.1) Case study on security tools (Kismet and NetStumbler)

Kismet

i) Kismet is a monitoring tool for wireless - originally only supporting 802.11 Wi-Fi, with the right hardware Kismet can now capture Bluetooth advertisements, BTLE, nRF-based wireless mice and keyboards, weather stations, wireless thermometers, switches, smoke detectors, 802.15.4 / Zigbee, ADSB airplane transponders, AMR wireless power, water, and gas meters, and more.

ii) Kismet works with Wi-Fi interfaces, Bluetooth interfaces, some SDR (software defined radio) hardware like the RTLSDR, and other specialized capture hardware.

iii) Kismet works on Linux, OSX, and, to a degree, Windows 10 under the WSL framework. On Linux it works with most Wi-Fi cards, Bluetooth interfaces, and other hardware devices. On OSX it works with the built-in Wi-Fi interfaces, and on Windows 10 it will work with remote captures.

iv) **Passive monitoring:** Kismet is largely focused on collecting, collating, and sorting wireless data. The logs generated by Kismet can be fed into other tools (the pcap, handshakes, and other data) like hashcat, aircrack, and more.

v) **Device-oriented display:** a) Kismet is different from Wireshark; Kismet primarily focuses on representing devices - access points, clients, bridged wired devices, sensors, Bluetooth entities, and so on, while Wireshark focuses on displaying a deep dive of specific packets and all the content.

b) Kismet and Wireshark work best when used *together* - Kismet collects packets and logs them to standard formats (pcap and pcapng) or the kismetdb format which can be converted directly to pcap and pcapng, and collects location, changes over time, etc, while Wireshark can open the pcap logs and give extensive detailed information about specific packets. Each tool is designed for a different job, but operate together.

vi) **Features:** a) Kismet differs from other wireless network detectors in working passively. Namely, without sending any loggable packets, it is able to detect the presence of both wireless access points and wireless clients, and to associate them with each other. It is also the most widely used and up to date open source wireless monitoring tool.

b) Kismet also includes basic wireless IDS features such as detecting active wireless sniffing programs including NetStumbler, as well as a number of wireless network attacks.

c) Kismet features the ability to log all sniffed packets and save them in a tcpdump/Wireshark or Airsnort compatible file format. Kismet can also capture "Per-Packet Information" headers.

d) Kismet also features the ability to detect default or "not configured" networks, probe requests, and determine what level of wireless encryption is used on a given access point. In order to find as many networks as possible, Kismet supports channel hopping. This means that it constantly changes from channel to channel non-sequentially, in a user-defined sequence with a default value that leaves big holes between channels (for example, 1-6-11-2-7-12-3-8-13-4-9-14-5-10). The advantage with this method is that it will capture more packets because adjacent channels overlap.

e) Kismet also supports logging of the geographical coordinates of the network if the input from a GPS receiver is additionally available.

vii) **Disadvantages of Kismet:** a) It takes long time to search networks.

b) It can only identify the wireless network (Wi-Fi) in a small area, if the range is more it cannot work properly.

NetStumbler

i) NetStumbler detects wireless local area networks (WLANs) that are based on the 802.11b and 802.11g data formats in the Industrial Scientific and Medical (ISM) radio band and Unlicensed National Information Infrastructure (U-NII), a band using 802.11a data formats.

ii) It provides radio frequency (RF) signal information and other data related to the peculiarities of combining computers and radios. MiniStumbler is the PocketPC version of NetStumbler. Both NetStumbler and MiniStumbler are active wireless network detection applications.

iii) **Uses:** a) Verify that your network is set up the way you intended.

b) Find locations with poor coverage in your WLAN.

c) Detect other networks that may be causing interference on your network.

d) Detect unauthorized "rogue" access points in your workplace.

e) Help aim directional antennas for long-haul WLAN links.

f) Use it recreationally for WarDriving.

iv) NetStumbler is an “active” wireless network detection application. This means the program takes a specific action to accomplish the WLAN detection. The action is to send out a specific data probe called a Probe Request. The Probe Request frame and the associated Probe Response frame are part of the 802.11 standard.

v) **Wireless Ethernet Cards that Work with NetStumbler and MiniStumbler:** To use NetStumbler or MiniStumbler, you need a wireless Ethernet card. There are a wide variety of makes and models available, and every day new models are released, so the question becomes:

Which ones work with NetStumbler? Generally, the best cards are those that use the Hermes chipset. Primarily, this refers to the ORiNOCO Gold or Silver "Classic" cards or "re-badged" versions of those cards. The big disadvantage to these cards, however, is that they only work with 802.11b data. "Re-badges", are made by manufacturers such as ORiNOCO, but sold under another brand name, such as Dell. The marking decals or "badge" is changed to reflect the new brand, hence the term "re-badge."

vi) **Damage & Defense… Disabling the Beacon:** a) NetStumbler transmits a "Broadcast Request" probe to discover the WLAN. Most access points will respond to a Broadcast Request by default. When it responds, the AP (access point) transmits its SSID, MAC number, and other information. However, many brands and models of AP allow this feature to be disabled. Once an AP ceases to respond to the request, NetStumbler can no longer detect it. If you don't want your wireless LAN to show up on the screen of another NetStumbler user, disable the SSID broadcast on your access point. Check your AP manual for "Disable SSID Broadcast", "Closed SSID," or similar features.

b) The one caveat to this is if the SSID that the WarDriver enters for NetStumbler happens to have the same SSID as your network, then your AP will still respond to the probe. This is another good reason to change the default SSID.

Q2. Explain and implement Hidden Terminal/Exposed terminal problem.

Hidden and exposed terminals

Consider the scenario with three mobile phones as shown in Figure 1. The transmission range of A reaches B, but not C (the detection range does not reach C either). The transmission range of C reaches B, but not A. Finally, the transmission range of B reaches A and C, i.e., A cannot detect C and vice versa. A starts sending to B, C does not receive this transmission. C also wants to send something to B and senses the medium. The medium appears to be free, the carrier sense fails. C also starts sending causing a collision at B. But A cannot detect this collision at B and continues with its transmission. A is hidden for C and vice versa.

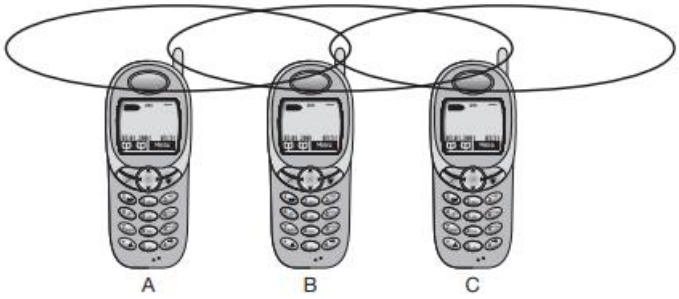


Figure 1

While hidden terminals may cause collisions, the next effect only causes unnecessary delay. Now consider the situation that B sends something to A and C wants to transmit data to some other mobile phone outside the interference ranges of A and B. C senses the carrier and detects that the carrier is busy (B's signal). C postpones its transmission until it detects the medium as being idle again. But as A is outside the interference range of C, waiting is not necessary. Causing a 'collision' at B does not matter because the collision is too weak to propagate to A. In this situation, C is exposed to B.

Multiple access with collision avoidance

Solution to hidden terminal

Multiple access with collision avoidance (MACA) presents a simple scheme that solves the hidden terminal problem, does not need a base station, and is still a random access Aloha scheme – but with dynamic reservation. Figure 2 shows the same scenario as Figure 1 with the hidden terminals. Remember, A and C both want to send to B. A has already started the transmission, but is hidden for C, C also starts with its transmission, thereby causing a collision at B.

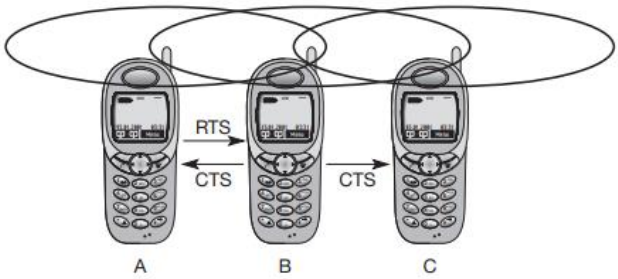


Figure 2

With MACA, A does not start its transmission at once, but sends a request to send (RTS) first. B receives the RTS that contains the name of sender and receiver, as well as the length of the future transmission. This RTS is not heard by C, but triggers an acknowledgement from B, called clear to send (CTS). The CTS again contains the names of sender (A) and receiver (B) of the user data, and the length of the future transmission. This CTS is now heard by C and the medium for future use by A is now reserved for the duration of the transmission. After receiving a CTS, C is not allowed to send anything for the duration indicated in the CTS toward B. A collision cannot occur at B during data transmission, and the hidden terminal problem is solved – provided that the transmission conditions remain the same. (Another station could move into the transmission range of B after the transmission of CTS.) Still, collisions can occur during

the sending of an RTS. Both A and C could send an RTS that collides at B. RTS is very small compared to the data transmission, so the probability of a collision is much lower. B resolves this contention and acknowledges only one station in the CTS (if it was able to recover the RTS at all). No transmission is allowed without an appropriate CTS. This is one of the medium access schemes that is optionally used in the standard IEEE 802.11.

Solution to exposed terminal

Can MACA also help to solve the 'exposed terminal' problem? Remember, B wants to send data to A, C to someone else. But C is polite enough to sense the medium before transmitting, sensing a busy medium caused by the transmission from B. C defers, although C could never cause a collision at A. With MACA, B has to transmit an RTS first (as shown in Figure 3) containing the name of the receiver (A) and the sender (B). C does not react to this message as it is not the receiver, but A acknowledges using a CTS which identifies B as the sender and A as the receiver of the following data transmission. C does not receive this CTS and concludes that A is outside the detection range. C can start its transmission assuming it will not cause a collision at A. The problem with exposed terminals is solved without fixed access patterns or a base station. One problem of MACA is clearly the overheads associated with the RTS and CTS transmissions – for short and time-critical data packets, this is not negligible. MACA also assumes symmetrical transmission and reception conditions. Otherwise, a strong sender, directed antennas etc. could counteract the above scheme

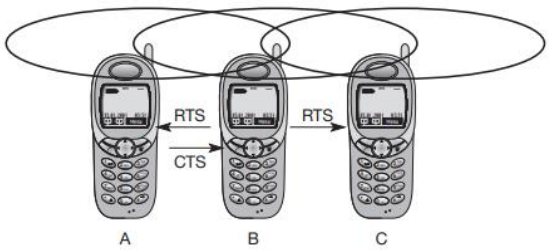


Figure 3

Figure 4 shows simplified state machines for a sender and receiver. The sender is idle until a user requests the transmission of a data packet. The sender then issues an RTS and waits for the right to send. If the receiver gets an RTS and is in an idle state, it sends back a CTS and waits for data. The sender receives the CTS and sends the data. Otherwise, the sender would send an RTS again after a time-out (e.g., the RTS could be lost or collided). After transmission of the data, the sender waits for a positive acknowledgement to return into an idle state. The receiver sends back a positive acknowledgement if the received data was correct. If not, or if the waiting time for data is too long, the receiver returns into idle state. If the sender does not receive any acknowledgement or a negative acknowledgement, it sends an RTS and again waits for the right to send. Alternatively, a receiver could indicate that it is currently busy via a separate RxBusy. Real implementations have to add more states and transitions, e.g., to limit the number of retries.

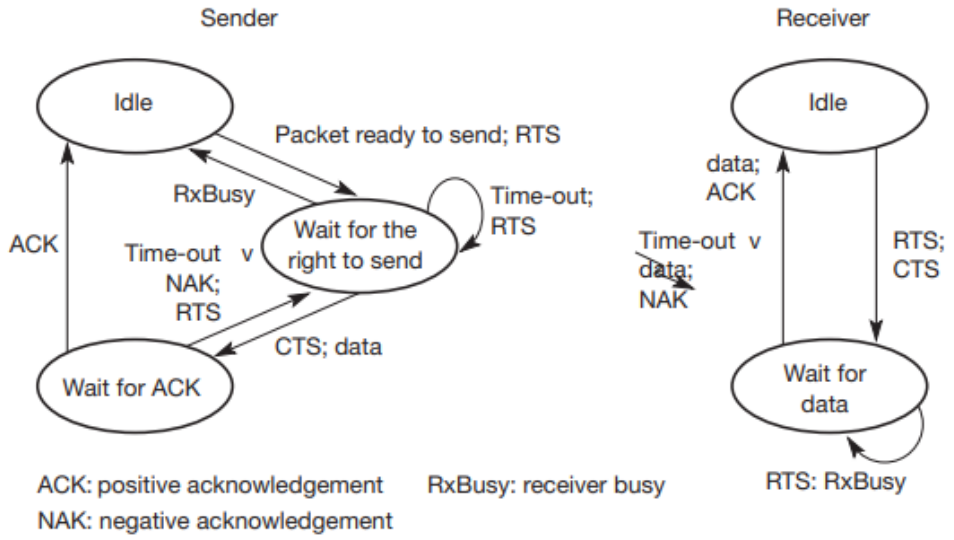


Figure 4