Thadomal Shahani Engineering College

Bandra (W.), Mumbai- 400 050.

CERTIFICATE

Certify that Mr./~~Miss~~ Rishab Mandal

of COMPUTERS Department, Semester VI with

Roll No. 2103110 has completed a course of the necessary

experiments in the subject Mobile Computing under my

supervision in the **Thadomal Shahani Engineering College**

Laboratory in the year 2023 - 2024

Teacher In-Charge

Head of the Department

Date 18/03/2024

Principal

TSEC

CONTENTS

THADOMAL SHAHANI TSEC ENGINEERING COLLEGE

EXPERIMENT NO. 1

Aim :- Write an Android application (WAA) to draw Basic graphical 2D primitives.

Theory :-

Android Studio is the official IDE for Android app development, offering a comprehensive toolkit for designing, coding and testing applications. Launched by Google in 2013, it has become an essential platform for developers of all skill levels.

Key Components and Features of Android Studio are Intuitive User Interface, Gradle Build System, Code Editor, XML Layout Editor, Emulator, Debugger and Profiler, and Version Control Integration.

Gradle Build System : Manages dependencies and builds in Android projects, flexibility, efficient incremental builds.
Code Editor : Core environment provided for coding and debugging, having features like code completion, syntax highlighting, debugging tools.
XML Layout Editor : Visual editor for designing Android layouts.
Emulator : It simulates android devices for testing and development purpose, test different screen sizes and simulate device features.

Android Studio projects are organized into layers, each serving a distinct purpose. The layers include the Presentation Layer, Domain Layer, and Data layer. The Presentation layer handles UI components and user interactions, the Domain Layer contains business logic, and the Data Layer manages data persistence.

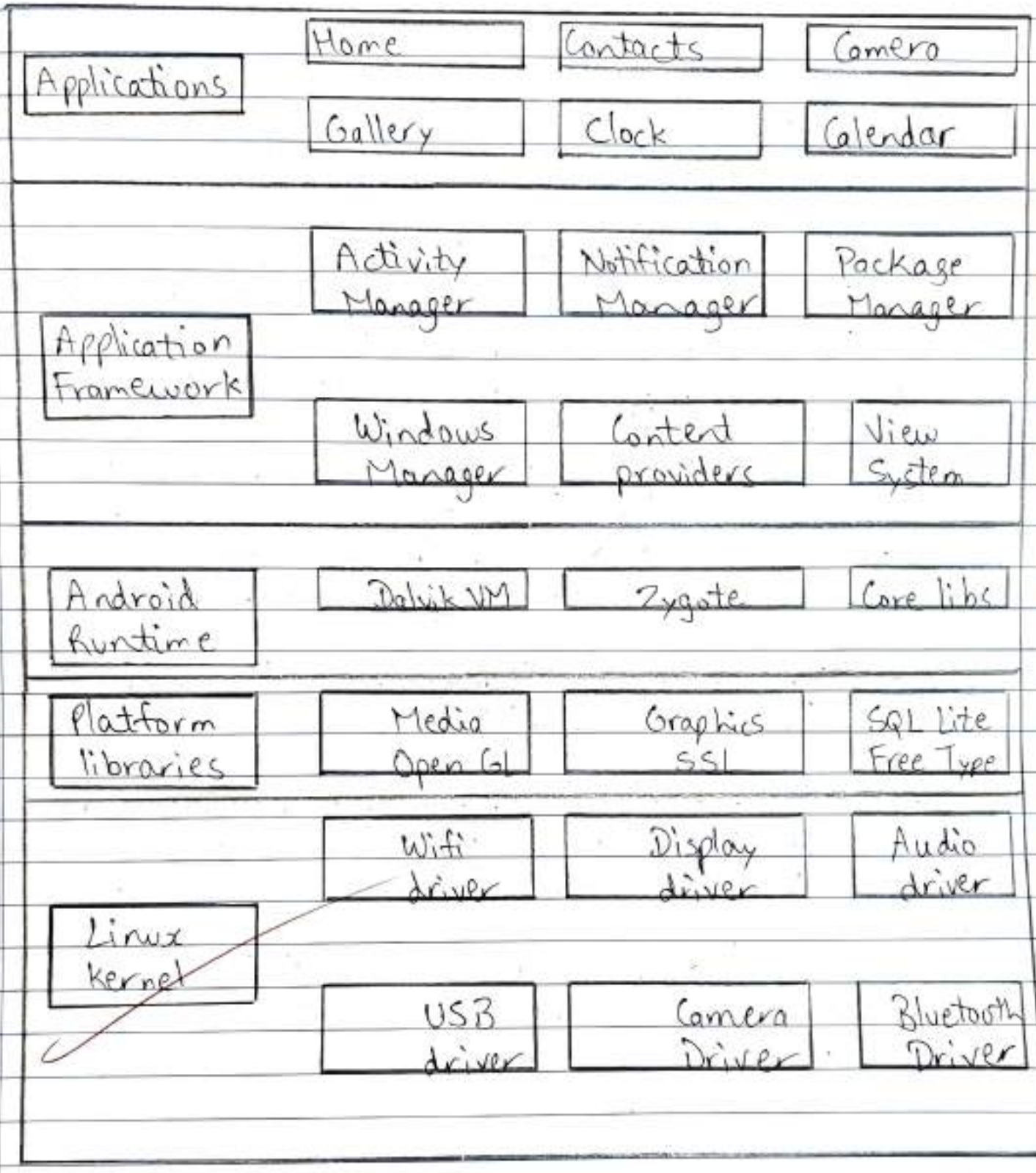
Steps

i) For initiating a new project, open Android Studio and select "Start a new Android Studio project".
ii) Choose the project template and set project's name and location. Select the form factors and target devices for your app.
iii) Choose the activity template that suits your project, such as "Empty Activity" for a clean slate. Customize additional project details and click "Finish" to create the project.

iv) For writing program, running the app and displaying the output, open the activity file (eg. MainActivity.java) to write program logic for creating a 2D figure.
v) Utilize the XML layout file (eg. activity_main.xml) to design the user interface with appropriate views. Run project by clicking the "Run" button. Observe the output in the emulator or on desired connected device.

THADOMAL SHAHANI TSEC ENGINEERING COLLEGE

Android Architecture :



The code explanation for 2D primitive is as follows :-

```
import android.app.Activity;
import android.graphics.Bitmap;
import android.graphics.Canvas;
import android.graphics.Color;
import android.graphics.Paint;
import android.graphics.drawable.BitmapDrawable;
import android.os.Bundle;
import android.widget.ImageView;
```

The above imports are utilized to provide methods for drawing on a bitmap or view, represent colors, styling and controlling appearances. OS.Bundle is used to pass data between activities.

→ `ImageView i = (ImageView) findViewById(R.id.imageView);`

→ `i.setBackgroundDrawable(new BitmapDrawable(bg));`
It finds the ImageView with id 'imageView' from the layout.

→ `Canvas canvas = new Canvas(bg);`
It creates a new canvas object associated with the bitmap, allowing drawing operations on it.

→ Paint paint = new Paint();
paint.setColor(Color.BLUE);
paint.setTextSize(50);
Paint Object creation. We have created a Paint object with blue color and a text size of 50.

→ Drawing Shape (Rectangle, Circle, Square, Line):
For circle, we use:
canvas.drawText("Circle", 120, 150, paint);
canvas.drawCircle(200, 350, 150, paint);
Similarly, we use drawLine, drawRect methods to draw similar 2D primitives.

Conclusion :-

The developed application uses built-in functions of Android Studio to display basic graphical 2D primitives.

A+

Exp No.1

Rishab Mandal

Batch: C23

Code:

activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.coordinatorlayout.widget.CoordinatorLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:fitsSystemWindows="true"
    tools:context=".MainActivity">

    <TextView
    android:id="@+id/textView"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="C23\nRishab Mandal\nRoll no: 2103110"
    android:textSize="20sp"
    android:textColor="@android:color/black"
    android:layout_centerHorizontal="true"
    android:layout_margin="50dp"/>

    <!-- Include your CustomView -->
    <com.example.exp1app.CustomView
        android:id="@+id/customView"
        android:layout_width="match_parent"
        android:layout_height="match_parent" />

</androidx.coordinatorlayout.widget.CoordinatorLayout>
```

MainActivity.java

```
package com.example.exp1app;

import android.os.Bundle;

import com.google.android.material.snackbar.Snackbar;

import androidx.appcompat.app.AppCompatActivity;

import android.view.View;

import androidx.core.view.WindowCompat;
```

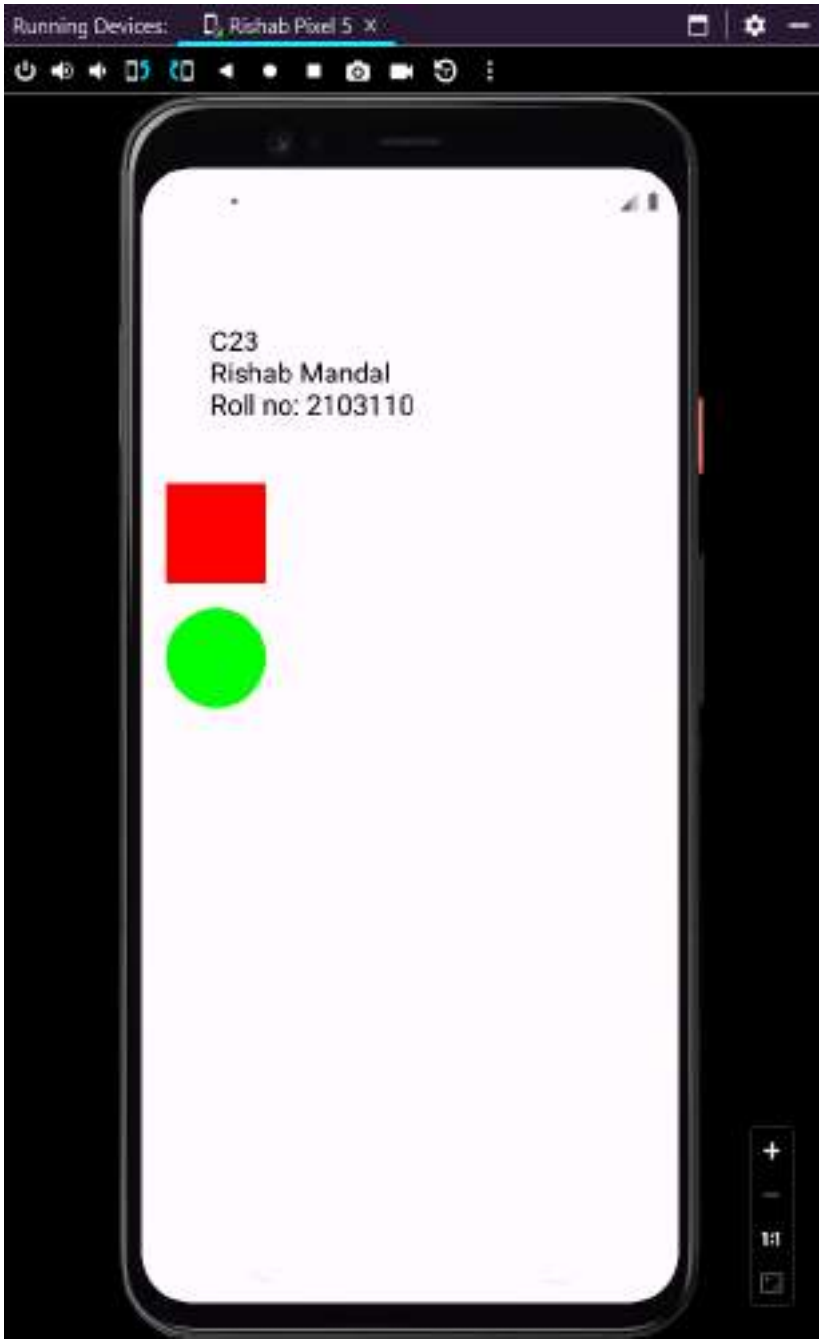
```
import androidx.navigation.NavController;
import androidx.navigation.Navigation;
import androidx.navigation.ui.AppBarConfiguration;
import androidx.navigation.ui.NavigationUI;

import com.example.exp1app.databinding.ActivityMainBinding;

import android.view.Menu;
import android.view.MenuItem;

public class MainActivity extends AppCompatActivity {

    private AppBarConfiguration appBarConfiguration;
    private ActivityMainBinding binding;

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

        CustomView customView = findViewById(R.id.customView);
    }
}
```

CustomView.java

```
package com.example.exp1app;

// CustomView.java
import android.content.Context;
import android.graphics.Canvas;
import android.graphics.Color;
import android.graphics.Paint;
import android.util.AttributeSet;
import android.view.View;

public class CustomView extends View {

    private Paint paint;

    public CustomView(Context context) {
        super(context);
        init();
    }

    public CustomView(Context context, AttributeSet attrs) {
        super(context, attrs);
        init();
    }

    private void init() {
        paint = new Paint();
        paint.setColor(Color.RED);
        paint.setStyle(Paint.Style.FILL);
```

```
    }

    @Override
    protected void onDraw(Canvas canvas) {
        super.onDraw(canvas);

        // Draw a rectangle
        int rectLeft = 50;
        int rectTop = 50;
        int rectRight = 250;
        int rectBottom = 250;
        canvas.drawRect(rectLeft, rectTop, rectRight, rectBottom, paint);

        // Draw a circle below the rectangle
        paint.setColor(Color.GREEN);
        int circleRadius = 100;
        int circleCenterX = (rectLeft + rectRight) / 2;
        int circleCenterY = rectBottom + circleRadius + 50;
        canvas.drawCircle(circleCenterX, circleCenterY, circleRadius,
paint);
    }
}
```

Output:


Running Devices:
Rishab Pixel 5
C23
Rishab Mandal
Roll no: 2103110

THADOMAL SHAHANI TSEC ENGINEERING COLLEGE

EXPERIMENT NO. 2

Aim :- Write an Android application to draw the basic graphical 3D primitives.

Theory :-

The android graphics API is a powerful set of tools and classes provided by the Android SDK for creating and manipulating the graphical elements in Android applications. It enables developers to render visual content, draw shapes, handle images, and perform various other graphical operations.

Program Description :

The program for the above experiment uses the Android SDK to create a graphical representation of various 3D figures using android Graphics API.

Structure : The program is structured as an Android activity, extending the AppCompactActivity class. In the onCreate method, the layout is set using the setContentView to associate the activity with the XML layout defined in the activity_mail.xml. A bitmap is created with a specified width and height, serving as a canvas for drawing 3D primitives.

An ImageView is then used to display the created bitmap as its background. The Canvas object is employed to draw shapes, lines and text, utilizing Paint objects to define styles and colors.

Built-in-Functions:

1) Bitmap.createBitmap:

Used to create a Bitmap object with a specified width, height and color configuration of RGB_565.

2) ImageView.setBackgroundDrawable:

Sets the background of the ImageView to the Bitmap created, making it the Canvas for subsequent drawings.

3) Canvas Constructor:

Initializes a canvas object, associating it with the previously created Bitmap for drawing operations.

4) Paint.setColor:

Sets the color of the paint used for the drawing.

In this case, two paint objects (paintBlack and paintBlue) are created, one for yellow text and another for green lines.

5) Paint.setTextSize :

Defines the text size for the paint. It sets the size of the text that will actually be drawn on the canvas.

6) Canvas.drawText :

Defines the text size for the paint it sets the size of the text that will be drawn, defined by Paint.setTextSize and is used to label the shapes like Cube, Cone or Prism.

7) Canvas.drawLine :

Draws a line on the canvas, defining its start and end points as parameters. Used extensively to outline the edges of geometric shapes.

8) Canvas.drawArc :

Draws an arc on the canvas. In this case, it is used to represent the base of the cone that has been drawn.

9) RectF constructor:

Initializes a RectF (rectangle) object, specifying co-ordinates for drawing arcs. Used in the conjunction with Canvas.drawArc to create the semi-circle for the cone base.

10) Canvas.drawLines:

Draws a series of lines on the canvas. Utilized to represent the edges of the prism.
The program follows the standard structure of an Android application with activities and layouts. It employs various built-in functions from the Android Graphics API to create and draw 3D shapes. The program showcases the versatility of Android's graphics capatibility of creating visually engaging applications.

Conclusion :-

Thus, we implemented the above experiment successfully by understanding the use of different built-in functions, present in Android Studio, meant for designing.

Exp No.2

Rishab Mandal

Batch: C23

Code:

activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:padding="16dp"
    tools:context=".MainActivity">

    <TextView
    android:id="@+id/textView"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_centerHorizontal="true"
    android:text="C23\nRishab Mandal\nRoll no: 2103110"
    android:textColor="@android:color/black"
    android:textSize="20sp"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent" />

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Name:"/>

    <EditText
        android:id="@+id/editTextName"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="Enter your name"/>

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Email:"/>

    <EditText
        android:id="@+id/editTextEmail"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:inputType="textEmailAddress"
        android:hint="Enter your email"/>

    <RadioGroup
        android:id="@+id/radioGroupGender"
        android:layout_width="match_parent"
```

```
        android:layout_height="wrap_content"
        android:orientation="horizontal">

        <RadioButton
            android:id="@+id/radioButtonMale"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Male"/>

        <RadioButton
            android:id="@+id/radioButtonFemale"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Female"/>

    </RadioGroup>

    <CheckBox
        android:id="@+id/checkBoxAgree"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="I agree to the terms and conditions"/>

    <Button
        android:id="@+id/buttonSubmit"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Submit"/>

</LinearLayout>
```

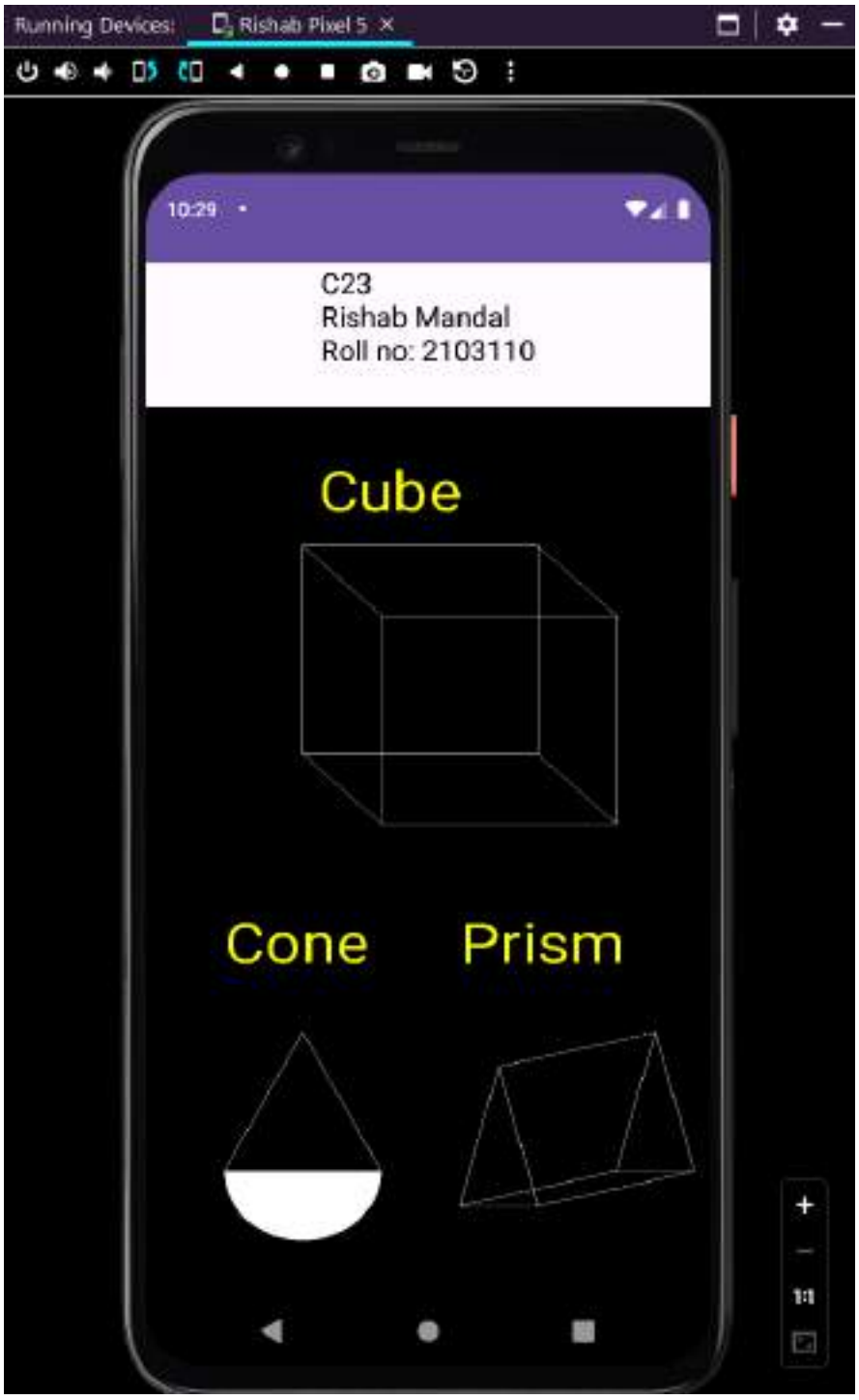
MainActivity.java

```
package com.example.a3dfigures;
//package com.example.exp2new38;

import androidx.appcompat.app.AppCompatActivity;

import android.app.Activity;
import android.graphics.Bitmap;
import android.graphics.Canvas;
import android.graphics.Color;
import android.graphics.Paint;
import android.graphics.RectF;
import android.graphics.drawable.BitmapDrawable;
import android.widget.ImageView;

import android.os.Bundle;

public class MainActivity extends AppCompatActivity {
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        //Creating a Bitmap
```

```
        Bitmap bg = Bitmap.createBitmap(720, 1280, Bitmap.Config.RGB_565);

        //Setting the Bitmap as background for the ImageView
        ImageView i = (ImageView) findViewById(R.id.imageView);
        i.setBackgroundDrawable(new BitmapDrawable(bg));

        //Creating the Canvas Object
        Canvas canvas = new Canvas(bg);

        //Creating the Paint Object and set its color & TextSize
        Paint paintBlack = new Paint();
        paintBlack.setColor(Color.YELLOW);
        paintBlack.setTextSize(80);

        Paint paintBlue = new Paint();
        paintBlue.setColor(Color.WHITE);
        paintBlue.setTextSize(100);



        //To draw a Cube
        canvas.drawText("Cube", 220, 150, paintBlack);
//       canvas.drawRect(400, 200, 500, 300, paint);
//       canvas.drawRect(500, 200, 600, 300, paint);
        canvas.drawLine(200, 200, 500, 200, paintBlue);
        canvas.drawLine(200, 500, 500, 500, paintBlue);
        canvas.drawLine(500, 200, 500, 500, paintBlue);
        canvas.drawLine(200, 200, 200, 500, paintBlue);

        canvas.drawLine(300, 300, 600, 300, paintBlue);
        canvas.drawLine(300, 600, 600, 600, paintBlue);
        canvas.drawLine(600, 300, 600, 600, paintBlue);
        canvas.drawLine(300, 300, 300, 600, paintBlue);

        canvas.drawLine(200, 200, 300, 300, paintBlue);
        canvas.drawLine(500, 200, 600, 300, paintBlue);
        canvas.drawLine(200, 500, 300, 600, paintBlue);
        canvas.drawLine(500, 500, 600, 600, paintBlue);


        canvas.drawText("Cone", 100, 800, paintBlack);
        canvas.drawLine(200, 900, 100, 1100, paintBlue);
        canvas.drawLine(200, 900, 300, 1100, paintBlue);

        //canvas.drawArc(180, 180, 180, 540, paintBlue);

        RectF rectF = new RectF(100, 1000, 300, 1200);
        canvas.drawArc(rectF, 0, 180, true, paintBlue);


        canvas.drawText("Prism", 400, 800, paintBlack);

        float[] points1 = {450, 950, 400, 1150, 450, 950, 500, 1150, 400,
1150, 500, 1150};
        canvas.drawLines(points1, paintBlue);

        float[] points2 = {650, 900, 600, 1100, 650, 900, 700, 1100, 600,
1100, 700, 1100};
        canvas.drawLines(points2, paintBlue);

        float[] points3 = {450, 950, 650, 900, 500, 1150, 700, 1100, 400,
```

```
1150, 600, 1100,};
        canvas.drawLines(points3, paintBlue);


    }
}
```

Output:



THADOMAL SHAHANI TSEC ENGINEERING COLLEGE

EXPERIMENT NO. 3

Aim :- Write an Android application to design a Form with GUI Components.

Theory :-

In Android app development, user interfaces are crucial for providing a seamless and intuitive experience. One common aspect is designing forms that allow users to input information. This theory discusses the development of a simple Android application using Android Studio, which creates a form with graphical user interface (GUI) components.

Program Description :

The Android application is developed as a user-friendly form, encapsulating a pivotal aspect of mobile app interaction. Focused on providing a seamless experience, the application is crafted using Android Studio, utilizing a combination of XML for layout design and Java for intricate programming logic.

XML Layout Design :

The Android application is a basic form that collects user information, including name, email, gender, and a checkbox for agreeing

to terms. The form includes 'res/layout/activity-main.xml' file which plays a crucial role in defining the visual structure of the form. Each GUI component is very thoughtfully arranged within a LinearLayout to ensure a responsive and visually pleasing layout. TextViews provide clear labels, EditText fields allow user input, and the RadioGroup with RadioButtons facilitate gender selection. Additionally, a CheckBox and a Button are incorporated for agreement confirmation and form submission, respectively.

Java Programming Logic :-

The 'src/main/java/com.example.FormApplication/MainActivity.java' file is the hub of program logic. The 'onCreate' method initializes views by connecting them to their XML counterparts. The 'setOnClickListener' function is implemented to respond to button clicks, validating user input and providing feedback through Toast messages.

Built-in Functions :

1) onCreate :

The 'onCreate' method, a vital lifecycle method,

initiates app's activity, ensuring seamless connectivity.

2) setOnClickListener():

The 'setOnClickListener' function is pivotal for user interaction. By defining a click listener for the submit button, it orchestrates the behaviour triggered upon button press, creating a responsive and interactive user experience.

3) findViewById:

The 'findViewById' function is employed to bridge the gap between the visual and logical aspects of the application. It locates and retrieves references to various GUI components, allowing the code to interact dynamically with the user interface.

4) getText() and toString():

The 'getText' function retrieves the input from EditText fields, fostering dynamic content handling. Coupled with 'toString', it converts the user's input into a manipulable String, enabling further processing and validation.

5) getCheckedRadioButtonId():

With the RadioGroup, the 'getCheckedRadioButtonId' function becomes instrumental.

It identifies the selected Radio Button, facilitating gender selection and enhancing the form's adaptability.

6) isChecked():

The 'isChecked' function, employed with the CheckBox, serves as a crucial input validation tool. It checks whether the user has agreed to the terms, ensuring a very comprehensive and error-free submission.

7) Toast.makeText():

The 'Toast.makeText' function is judiciously utilized to communicate with the user. It provides concise yet informative feedback, thus enhancing the overall user experience by offering real-time notifications and acknowledgement of successful submissions.

Conclusion :-

In this comprehensive exploration of Android form development, with a deeper understanding of built-in functions and processes, we understand how to design intricate and user-centric forms, setting stage for complex mobile applications.

Exp No.3

Rishab Mandal

Batch: C23

Code:

activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:padding="16dp"
    tools:context=".MainActivity">

    <TextView
    android:id="@+id/textView"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="C23\nRishab Mandal\nRoll no: 2103110"
    android:textSize="20sp"
    android:textColor="@android:color/black"
    android:layout_centerHorizontal="true"
    android:layout_margin="50dp"/>

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Name:"/>
    <EditText
        android:id="@+id/editTextName"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="Enter your name"/>
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Email:"/>
    <EditText
        android:id="@+id/editTextEmail"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:inputType="textEmailAddress"
        android:hint="Enter your email"/>
    <RadioGroup
        android:id="@+id/radioGroupGender"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="horizontal">
        <RadioButton
            android:id="@+id/radioButtonMale"
            android:layout_width="wrap_content"
```

```
            android:layout_height="wrap_content"
            android:text="Male"/>
        <RadioButton
            android:id="@+id/radioButtonFemale"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Female"/>
    </RadioGroup>
    <CheckBox
        android:id="@+id/checkBoxAgree"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="I agree to the terms and conditions"/>
    <Button
        android:id="@+id/buttonSubmit"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Submit"/>

</LinearLayout>
```

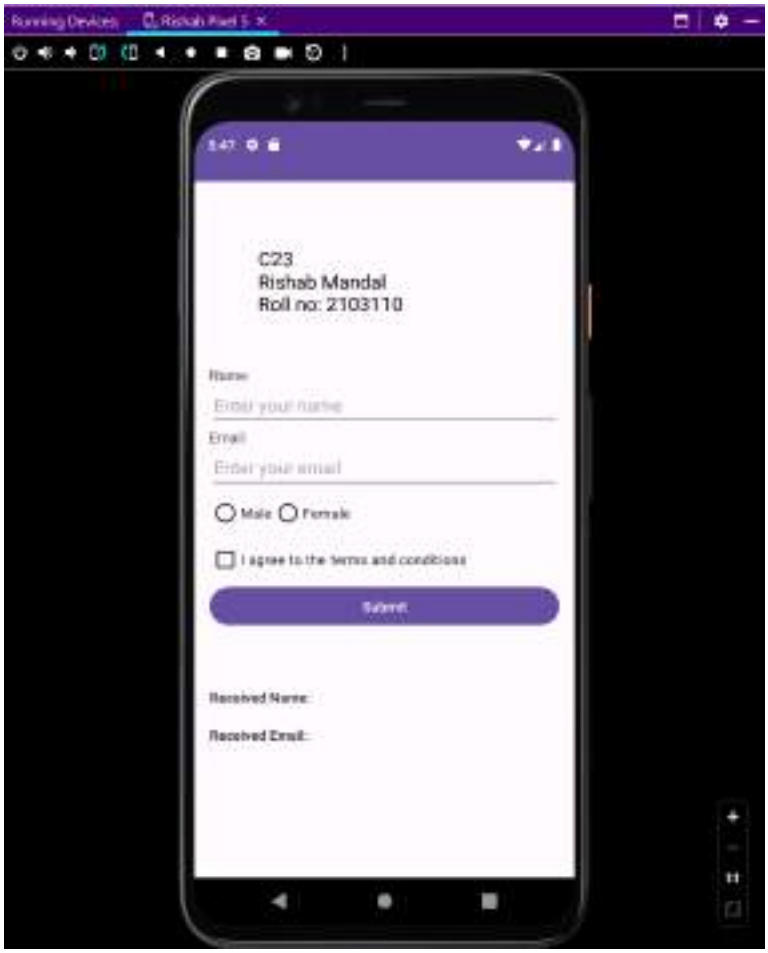
MainActivity.java

```
package com.example.formapplication;

import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.CheckBox;
import android.widget.EditText;
import android.widget.RadioButton;
import android.widget.RadioGroup;
import android.widget.Toast;

import androidx.appcompat.app.AppCompatActivity;

public class MainActivity extends AppCompatActivity {

    private EditText editTextName, editTextEmail;
    private RadioGroup radioGroupGender;
    private CheckBox checkBoxAgree;
    private Button buttonSubmit;

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

        // Initialize views
        editTextName = findViewById(R.id.editTextName);
        editTextEmail = findViewById(R.id.editTextEmail);
        radioGroupGender = findViewById(R.id.radioGroupGender);
        checkBoxAgree = findViewById(R.id.checkBoxAgree);
        buttonSubmit = findViewById(R.id.buttonSubmit);

        // Set click listener for the submit button
        buttonSubmit.setOnClickListener(new View.OnClickListener() {
            @Override
```

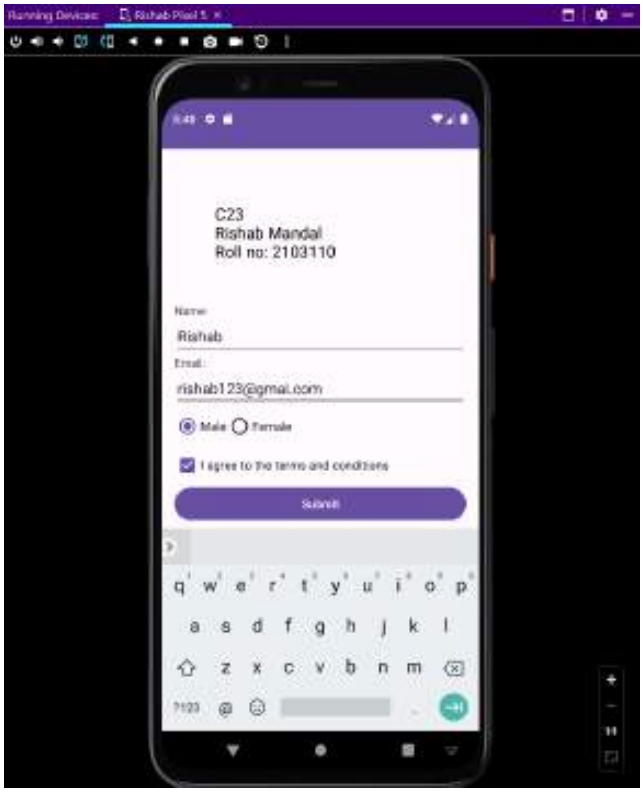
```
            public void onClick(View view) {
                // Get user input
                String name = editTextName.getText().toString().trim();
                String email = editTextEmail.getText().toString().trim();
                int genderId = radioGroupGender.getCheckedRadioButtonId();
                boolean agreeToTerms = checkBoxAgree.isChecked();

                // Validate input
                if (name.isEmpty() || email.isEmpty() || genderId == -1 ||
!agreeToTerms) {
                    Toast.makeText(MainActivity.this, "Please fill in all
fields and agree to terms", Toast.LENGTH_SHORT).show();
                } else {
                    // Display a toast with the form data
                    String gender = (genderId == R.id.radioButtonMale) ?
"Male" : "Female";
                    String message = "Name: " + name + "\nEmail: " + email
+ "\nGender: " + gender + "\nAgree to terms: " + agreeToTerms;
                    Toast.makeText(MainActivity.this, message,
Toast.LENGTH_LONG).show();
                }
            }
        });
    }
}
```

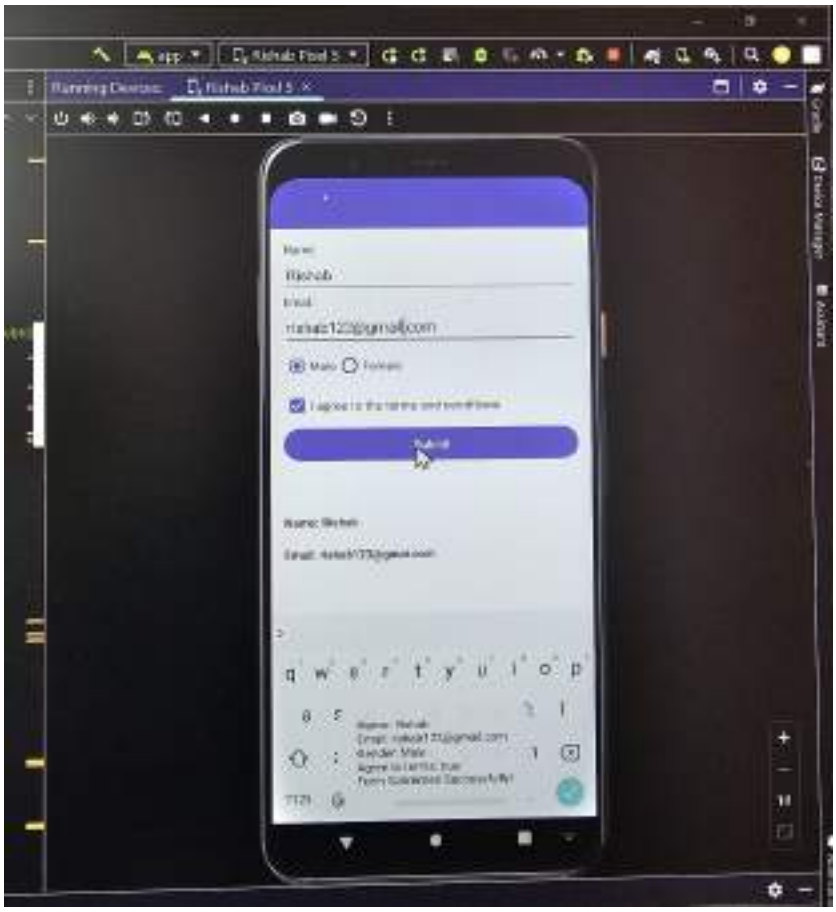
Output:

Opening and filing the form



Filled form, clicking on submit button


C23
Rishab Mandal
Roll no: 2103110
Name
Rishab
Email
rishab123@gmail.com
Male
Female
I agree to the terms and conditions
Submit



THADOMAL SHAHANI TSEC ENGINEERING COLLEGE

EXPERIMENT NO. 4

Aim :- Write an Android Application to design GUI components using Database.

Theory :-

To develop a form-based GUI application in Android Studio integrated with backend such as Express.js and MongoDB, you would start by designing the user interface in Android Studio, utilizing XML layouts to create form elements such as text fields, checkboxes, radio buttons, etc to capture user input. Then you'd implement the necessary logic in Java to handle user interactions, validate input, and prepare data for submission. Finally, appropriate error handling and feedback mechanisms should be implemented throughout the application to ensure a smooth UI experience.

Program structure :-

AndroidManifest.xml :

To ensure that your application has the necessary permissions for internet to allow to connect to internet, declare the following in this file,
<uses-permission android:name="android.permission.INTERNET" />

activity_main.xml :

The XML layout provided defines a UI for a form in an Android application. Below are the functions and attributes used :

1) LinearLayout :

The root layout used is a vertical LinearLayout. It arranges its child views in a single column.

2) TextView :

TextView elements are used to display static text labels.

3) EditText :

EditText elements allow user to input text.

4) RadioGroup :

RadioGroup is used to group RadioButtons together. In this layout, it's used to provide options for gender selection.

5) RadioButton :

RadioButton elements are used for selecting options. They provide choices for gender selection, with options for male and female.

6) CheckBox :

CheckBox is used to provide a single checkbox option. In this layout, it is used to obtain license.

7) Button :

Button elements are used to trigger actions when clicked, used to submit the form.

These UI elements and their corresponding attributes facilitate creation of a user-friendly interface.

Java Programming Logic :-

The Java code provided for app encompasses various functions and classes to facilitate functionality of a form submission and data retrieval process.
Functions utilized in this program :

1) onCreate() :

This method is part of Activity lifecycle and is called when activity is first created.

2) onClick() :

This method is used to handle the click event of the Submit button.

3) isChecked():

This method is used to determine whether a CheckBox is checked or not.

4) Toast.makeText():

This method is used to display a transient notification to user, commonly known as Toast.

5) onPostExecute(JSONArray receivedData):

This method is executed on main/UI thread after background task (GetFormDataTask) completes. It updates the UI with received from data by iterating through JSONArray, extracting each form data object, and setting the corresponding TextViews with received data.

6) getDataFromBackend():

This method initiates process of retrieving form data from the backend server. It invokes an AsyncTask (GetFormDataTask) to perform a GET request in background.

Basically, the application serves as a platform for users to input their name, email, gender

and agreement to terms and conditions through a form interface. Upon submission of form, data is validated, and if successful, a toast message displays the entered information. The application then constructs a JSON object containing the form data and sends it to a backend server via a POST request. Additionally, the application retrieves form data from the backend server through a GET request and updates to UI to display the received information. This process allows users to interact with the form, submit their data, and view previously submitted data fetched from the server, providing a comprehensive user experience.

Conclusion :-

Thus, we developed a form application which enables users to seamlessly submit and retrieve form data, facilitating efficient interaction and data management.

Exp No.4

Rishab Mandal

Batch: C23

Code:

activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:padding="16dp"
    tools:context=".MainActivity">

    <TextView
    android:id="@+id/textView"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="C23\nRishab Mandal\nRoll no: 2103110"
    android:textSize="20sp"
    android:textColor="@android:color/black"
    android:layout_centerHorizontal="true"
    android:layout_margin="50dp"/>

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Name:"/>

    <EditText
        android:id="@+id/editTextName"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="Enter your name"/>

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Email:"/>

    <EditText
        android:id="@+id/editTextEmail"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:inputType="textEmailAddress"
        android:hint="Enter your email"/>

    <RadioGroup
        android:id="@+id/radioGroupGender"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
```

```
        android:orientation="horizontal">

        <RadioButton
            android:id="@+id/radioButtonMale"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Male"/>

        <RadioButton
            android:id="@+id/radioButtonFemale"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Female"/>

    </RadioGroup>

    <CheckBox
        android:id="@+id/checkBoxAgree"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="I agree to the terms and conditions"/>

    <Button
        android:id="@+id/buttonSubmit"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Submit"/>

    <!-- Add TextViews to display received data -->
    <TextView
        android:id="@+id/textViewReceivedName"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Received Name:"
        android:layout_marginTop="60dp"
        android:textStyle="bold" />

    <TextView
        android:id="@+id/textViewReceivedEmail"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="20dp"
        android:text="Received Email:"
        android:textStyle="bold" />

</LinearLayout>
```

AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools">

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

    <application
        android:usesCleartextTraffic="true"
        android:allowBackup="true"
        android:dataExtractionRules="@xml/data_extraction_rules"
```

```
        android:fullBackupContent="@xml/backup_rules"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/Theme.FormApplication"
        tools:targetApi="31">
        <activity
            android:name=".MainActivity"
            android:exported="true">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER"
/>
            </intent-filter>
        </activity>
    </application>

</manifest>
```

MainActivity.java

```
package com.example.formapplication;

import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.widget.Button;
import android.widget.CheckBox;
import android.widget.EditText;
import android.widget.RadioButton;
import android.widget.RadioGroup;
import android.widget.TextView;
import android.widget.Toast;
import android.os.AsyncTask;

import org.json.JSONArray;
import org.json.JSONObject;
import java.io.OutputStream;
import java.net.HttpURLConnection;
import java.net.URL;
import java.io.*;

import androidx.appcompat.app.AppCompatActivity;

public class MainActivity extends AppCompatActivity {

    private EditText editTextName, editTextEmail;
    private RadioGroup radioGroupGender;
    private CheckBox checkBoxAgree;
    private Button buttonSubmit;
    private TextView textViewReceivedName, textViewReceivedEmail;

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

        // Initialize views
```

```
        editTextName = findViewById(R.id.editTextName);
        editTextEmail = findViewById(R.id.editTextEmail);
        radioGroupGender = findViewById(R.id.radioGroupGender);
        checkBoxAgree = findViewById(R.id.checkBoxAgree);
        buttonSubmit = findViewById(R.id.buttonSubmit);

        // Initialize TextViews
        textViewReceivedName = findViewById(R.id.textViewReceivedName);
        textViewReceivedEmail = findViewById(R.id.textViewReceivedEmail);

        // Set click listener for the submit button
        buttonSubmit.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                // Get user input
                String name = editTextName.getText().toString().trim();
                String email = editTextEmail.getText().toString().trim();
                int genderId = radioGroupGender.getCheckedRadioButtonId();
                boolean agreeToTerms = checkBoxAgree.isChecked();

                // Validate input
                if (name.isEmpty() || email.isEmpty() || genderId == -1 ||
!agreeToTerms) {
                    Toast.makeText(MainActivity.this, "Please fill in all
fields and agree to terms", Toast.LENGTH_SHORT).show();
                } else {
                    // Display a toast with the form data
                    String gender = (genderId == R.id.radioButtonMale) ?
"Male" : "Female";
                    String message = "Name: " + name + "\nEmail: " + email
+ "\nGender: " + gender + "\nAgree to terms: " + agreeToTerms+"\nForm
Submitted Successfully!";
                    Toast.makeText(MainActivity.this, message,
Toast.LENGTH_LONG).show();

                    // Create a JSON object with the form data
                    JSONObject postData = new JSONObject();
                    try {
                        postData.put("name", name);
                        postData.put("email", email);
                        postData.put("genderId", genderId);
                        postData.put("agreeToTerms", agreeToTerms);
                    } catch (Exception e) {
                        e.printStackTrace();
                    }

                    // Send the form data to the backend service
                    new SendFormDataTask().execute(postData);
                    getDataFromBackend();
                }
            }
        });
    }

    private class SendFormDataTask extends AsyncTask<JSONObject, Void,
Void> {
        @Override
        protected Void doInBackground(JSONObject... jsonObjects) {
            try {
                // Define the URL of your backend service
                URL url = new URL("https://mcc-exp-4-
```

```
server.vercel.app/saveFormData");

                // Open connection
                HttpURLConnection conn = (HttpURLConnection)
url.openConnection();
                conn.setRequestMethod("POST");
                conn.setRequestProperty("Content-Type",
"application/json");
                conn.setRequestProperty("Accept", "application/json");
                conn.setDoOutput(true);

                // Write JSON data to the output stream
                OutputStream os = conn.getOutputStream();
                os.write(jsonObjects[0].toString().getBytes());
                os.flush();
                os.close();

                // Get response code (optional)
                int responseCode = conn.getResponseCode();
                if (responseCode == HttpURLConnection.HTTP_OK) {
                    // Request was successful, you may handle the response
if needed
                    // For example, reading response body
                    BufferedReader in = new BufferedReader(new
InputStreamReader(conn.getInputStream()));
                    StringBuilder response = new StringBuilder();
                    String line;
                    while ((line = in.readLine()) != null) {
                        response.append(line);
                    }
                    in.close();
                    Log.d("Response", response.toString());
                } else {
                    // Request failed, read error response
                    BufferedReader in = new BufferedReader(new
InputStreamReader(conn.getErrorStream()));
                    StringBuilder errorMessage = new StringBuilder();
                    String line;
                    while ((line = in.readLine()) != null) {
                        errorMessage.append(line);
                    }
                    in.close();
                    Log.d("Response error: ",
String.valueOf(responseCode));
                    Log.d("Error saving form data: " ,
errorMessage.toString());
                }

                conn.disconnect();
            } catch (Exception e) {
                e.printStackTrace();
            }
            return null;
        }
    }

    private void getDataFromBackend() {
        // Perform GET request to get form data from backend
        new GetFormDataTask().execute();
    }
```

```
    private class GetFormDataTask extends AsyncTask<Void, Void, JSONArray>
{
        @Override
        protected JSONArray doInBackground(Void... voids) {
            JSONArray receivedData = null;
            try {
                // Define the URL of your backend service
                URL url = new URL("https://mcc-exp-4-
server.vercel.app/getFormData");

                // Open connection
                HttpURLConnection conn = (HttpURLConnection)
url.openConnection();
                conn.setRequestMethod("GET");

                // Get response code (optional)
                int responseCode = conn.getResponseCode();
                if (responseCode == HttpURLConnection.HTTP_OK) {
                    // Request was successful, read response
                    BufferedReader in = new BufferedReader(new
InputStreamReader(conn.getInputStream()));
                    StringBuilder response = new StringBuilder();
                    String line;
                    while ((line = in.readLine()) != null) {
                        response.append(line);
                    }
                    in.close();
                    receivedData = new JSONArray(response.toString());
                } else {
                    // Request failed
                    Log.d("Response error: ",
String.valueOf(responseCode));
                }

                conn.disconnect();
            } catch (Exception e) {
                e.printStackTrace();
            }
            return receivedData;
        }

        @Override
        protected void onPostExecute(JSONArray receivedData) {
            super.onPostExecute(receivedData);
            // Update UI with received data
            if (receivedData != null) {
                try {
                    // Iterate through the JSONArray and display each
object
                    for (int i = 0; i < receivedData.length(); i++) {
                        JSONObject formData =
receivedData.getJSONObject(i);
                        // Display each form data object as needed
                        String name = formData.getString("name");
                        String email = formData.getString("email");
                        // Handle other fields if needed
                        Log.d("Received Data:", "Name: " + name + ", Email:
" + email);
                        // Update UI accordingly
                        runOnUiThread(new Runnable() {
                            @Override
```

```
                            public void run() {
                                // Update UI elements with received data
                                // For example, you can set the received
data to TextViews
                                textViewReceivedName.setText("Name: " +
name);
                                textViewReceivedEmail.setText("Email: " +
email);
                                // You can add more UI elements for other
fields if needed
                            }
                        });
                    }
                } catch (Exception e) {
                    e.printStackTrace();
                }
            }
        }
    }


}
```

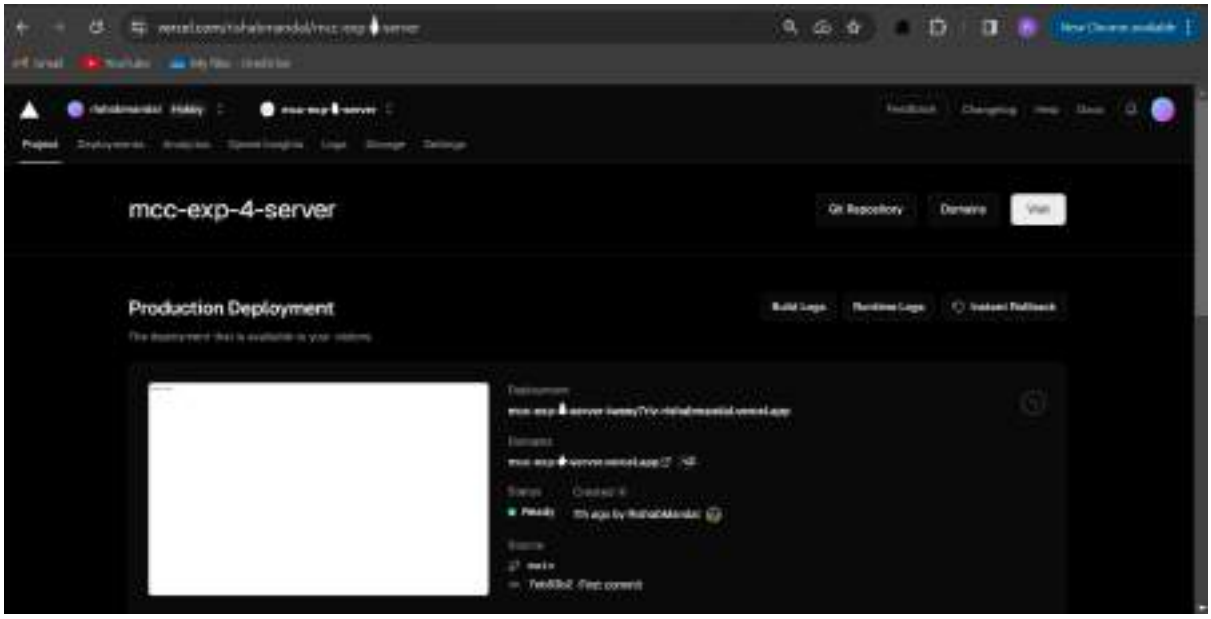
Backend server (server.js)

```
const express = require("express");
const bodyParser = require("body-parser");
const mongoose = require("mongoose");

const app = express();
const port = process.env.PORT || 3001; // or any other port you prefer

app.use(bodyParser.json());
app.use(express.json());
app.use(express.urlencoded({ extended: true }));

// Connect to MongoDB using Mongoose
// mongoose.connect('mongodb://localhost:27017/formdata', {

// Define endpoint to save form data
app.post("/saveFormData", async (req, res) => {
  try {
    await mongoose
      .connect(
        "mongodb+srv://***:***@expresstry.wqhmyb0.mongodb.net/formdata",
        {
          useNewUrlParser: true,
          useUnifiedTopology: true,
        }
      )
      .then(() => {
        console.log("Connected to MongoDB successfully");
      })
```

```
      .catch((err) => {
        console.error("Error connecting to MongoDB:", err);
      });

    let db = mongoose.connection;

    // Define schema for form data
    const formDataSchema = new mongoose.Schema({
      name: String,
      email: String,
      genderId: Number,
      agreeToTerms: Boolean,
    });

    // Define model for form data
    const FormData = mongoose.model("FormData", formDataSchema);
    const formData = req.body;
    // Create a new document using the FormData model
    const result = await db.collection("formData").insertOne(formData);
    console.log("Form data saved successfully:", result.insertedId);
    res.send("Form data saved successfully");
  } catch (err) {
    console.error("Error saving form data:", err);
    res.status(500).send("Error saving form data");
  }
});

app.get("/getFormData", async (req, res) => {
  try {
    await mongoose
      .connect(
        "mongodb+srv://Rishab***:***@expresstry.wqhmyb0.mongodb.net/formdata",
        {
          useNewUrlParser: true,
          useUnifiedTopology: true,
        }
      )
      .then(() => {
        console.log("Connected to MongoDB successfully");
      })
      .catch((err) => {
        console.error("Error connecting to MongoDB:", err);
      });

    let db = mongoose.connection;
    const allFormData = await db.collection("formData").find().toArray();
    res.json(allFormData); // Send the data as JSON response
  } catch (err) {
```

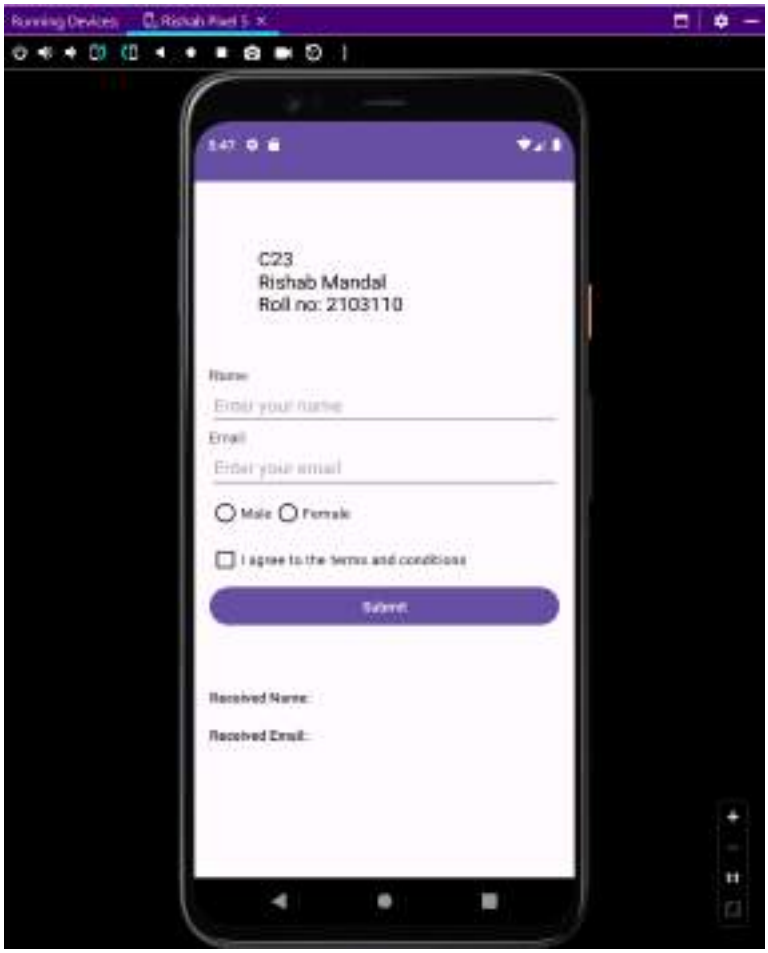
```
    console.error("Error fetching form data:", err);
    res.status(500).send("Error fetching form data: ", err);
  }
});

app.listen(port, () => {
  console.log(`Server is listening on port ${port}`);
});
```

Server Deployment on Vercel

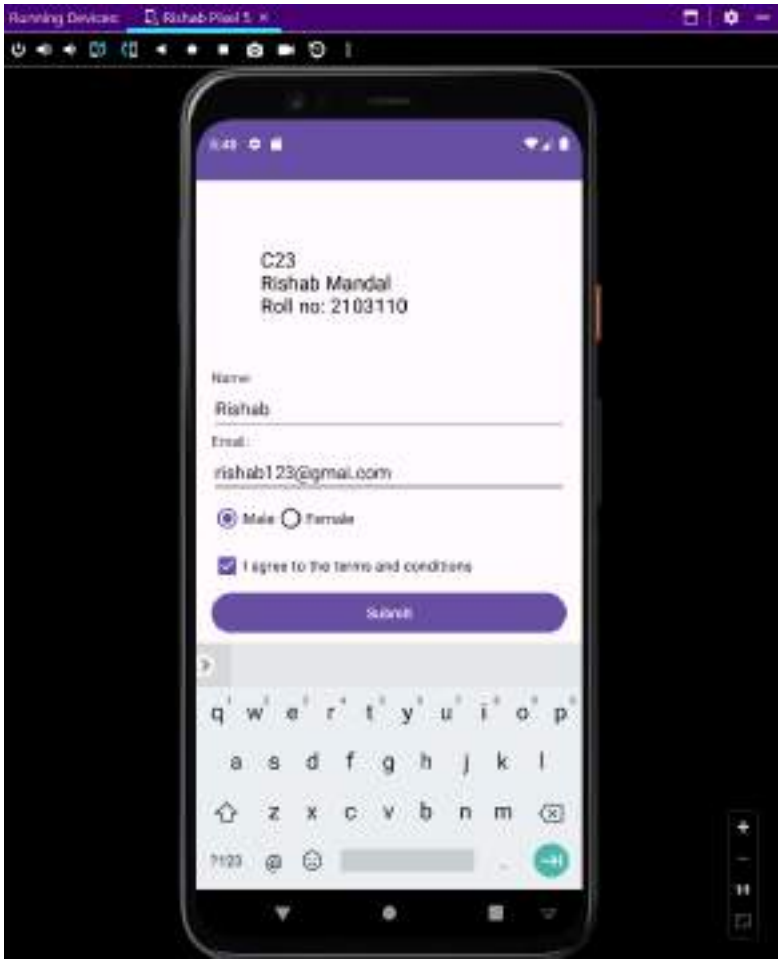


Output:

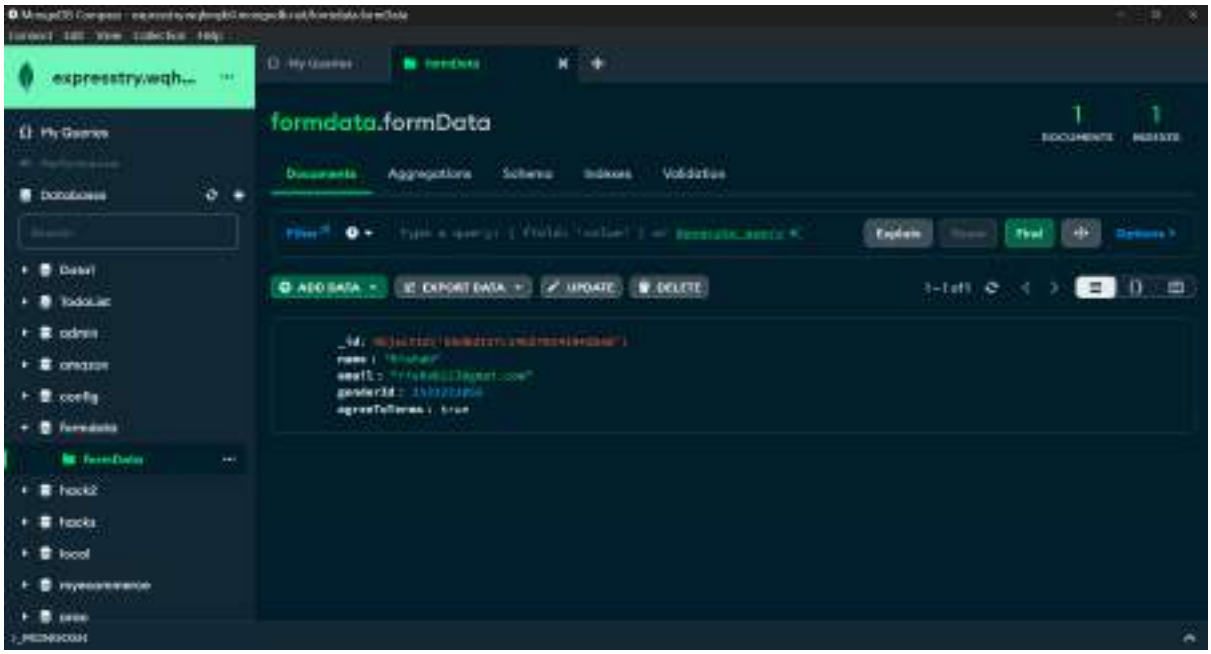
Opening and filing the form



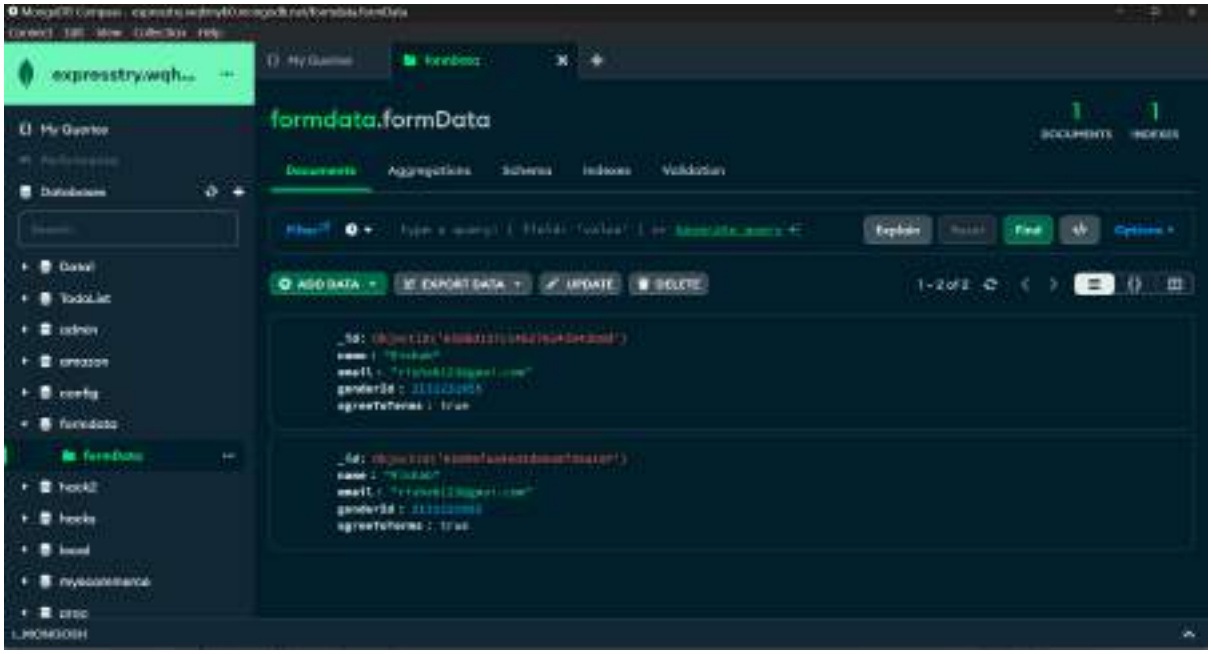
Filled form, clicking on submit button



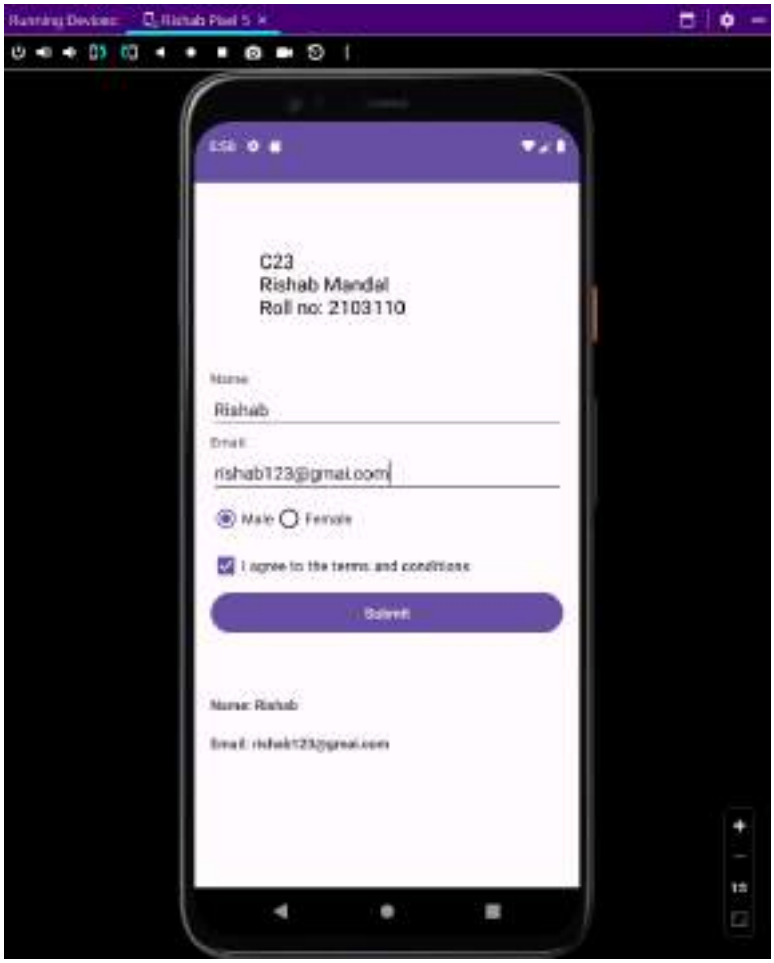
Database: Before submission



Database: After Submission of form



Received Name & Email from backend displayed below the submit button



THADOMAL SHAHANI TSEC ENGINEERING COLLEGE

EXPERIMENT NO. 5

Aim :- Write an Android application to develop an EMI Calculator application.

Theory :-

To develop an EMI calculator app using Android Studio, start by creating a new project and designing the user interface with input fields for loan amount, interest rate, loan tenure, and buttons for calculation. Utilize XML layout files for UI design. Implement the EMI calculation logic in the Java code, handling user input validation and computation of EMI based on formula. Finally display the calculated EMI in a text view. Ensure to test the application thoroughly to ensure accuracy and usability.

XML Layout Design :

The XML code represents the layout of app screen for calculating loan EMIs (Equated Monthly Installments). Here's its working are as follows :

1. Relative layout : This is root layout, allowing child views to be positioned relative to each other.

2. LinearLayout : Nested within the RelativeLayout, this layout is set to arrange its child views

vertically, and to center its content vertically.

3. EditTexts : Three EditText fields are provided for user input, Each EditText has attributes such as padding, text color, background drawable, text size, etc.

4. AppCompatButton : This button with the ID triggers the loan calculation. It has attributes for text, text color, text size, background drawable, padding and layout margins.

5. TextView : This TextView with the given ID is initially set to display "EMI". It will dynamically update with calculated EMI result.

Overall, this layout provides a user-friendly interface for inputting loan details and triggering a calculation, with results displayed dynamically on the screen.

Java Programming logic :-

The java code initializes variables for EditTexts, Button, and TextView by finding their respective views using their IDs. An onClickListener is set on button to listen for clicks. On clicking, following are retrieved : Values entered by the user from the EditText fields for principal

amount, interest rate, and loan tenure. These values are passed into double data types and passed to the calculate EMI method. Using the formulae, monthly interest rate and monthly installment amount is computed based on the principal amount, interest rate, and loan tenure.

Built-in functions :-

1) setContentView() :

This function sets the layout for the activity by inflating the XML Layout file (activity_main.xml) and displaying its UI components on the screen.

2) findViewById() :

This function is used to find and return the view object associated with the specified resource ID (R.id.*). It is used to initialize variables for EditTexts, Button, and TextView by locating them within the layout.

3) setOnClickListener() :

This function sets an OnClickListener on the button(calculate button, allowing app to listen for and handle clicks on this button.

4) getText(): The function getText().toString() retrieves the text entered by the user in the EditText fields and converts it into a String.

5) Double.parseDouble():

This function converts a String representation of a number into a double-precision floating-point value. It's used to parse the user-inputted values for principal amount, interest rate, and loan tenure from String to dOuble.

6) Math.pow():

This function calculates a base raised to the power of an exponent.

7) setText(): This function sets text to be displayed in textViewResult TextView.

Conclusion:-

Thus, we implemented EMI calculator by understanding the use of different built-in functions present in Android Studio.

Exp No.5

Rishab Mandal

Batch: C23

Code:

activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:padding="16dp"
    tools:context=".MainActivity">

    <TextView
    android:id="@+id/textView"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_centerHorizontal="true"
    android:text="C23\nRishab Mandal\nRoll no: 2103110"
    android:textColor="@android:color/black"
    android:textSize="20sp"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent" />

    <!-- Title -->
    <TextView
        android:id="@+id/textViewTitle"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="EMI Calculator"
        android:textSize="24sp"
        android:textStyle="bold"
        android:gravity="center"
        android:layout_marginBottom="180dp"/>

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_below="@id/textViewTitle"
        android:orientation="vertical">

        <!-- Input fields -->
        <EditText
            android:id="@+id/editTextPrincipal"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:hint="Principal Amount"
            android:inputType="numberDecimal"
            android:padding="12dp"
            android:background="@drawable/edit_text_bg"
            android:textColor="@android:color/black"
            android:textSize="16sp"
```

```
            android:layout_marginBottom="16dp"/>

        <EditText
            android:id="@+id/editTextInterestRate"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:hint="Interest Rate (%)"
            android:inputType="numberDecimal"
            android:padding="12dp"
            android:background="@drawable/edit_text_bg"
            android:textColor="@android:color/black"
            android:textSize="16sp"
            android:layout_marginBottom="16dp"/>

        <EditText
            android:id="@+id/editTextTenure"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:hint="Loan Tenure (in months)"
            android:inputType="number"
            android:padding="12dp"
            android:background="@drawable/edit_text_bg"
            android:textColor="@android:color/black"
            android:textSize="16sp"
            android:layout_marginBottom="24dp"/>

        <!-- Calculate button -->
        <androidx.appcompat.widget.AppCompatButton
            android:id="@+id/buttonCalculate"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:text="Calculate"
            android:textColor="@android:color/white"
            android:textSize="18sp"
            android:background="@drawable/button_bg"
            android:padding="16dp"
            android:layout_marginBottom="24dp"/>

        <!-- Result -->
        <TextView
            android:id="@+id/textViewResult"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:gravity="center"
            android:text="EMI: "
            android:textSize="20sp" />
    </LinearLayout>

</RelativeLayout>
```

edit_text_bg.xml

```
<!-- edit_text_bg.xml -->
<shape xmlns:android="http://schemas.android.com/apk/res/android"
    android:shape="rectangle">
    <solid android:color="#F0F0F0"/> <!-- Background color -->
    <corners android:radius="8dp"/> <!-- Rounded corners -->
    <stroke
        android:width="1dp"
```

```
        android:color="#CCCCCC"/> <!-- Border color and width -->
</shape>
```

button_bg.xml

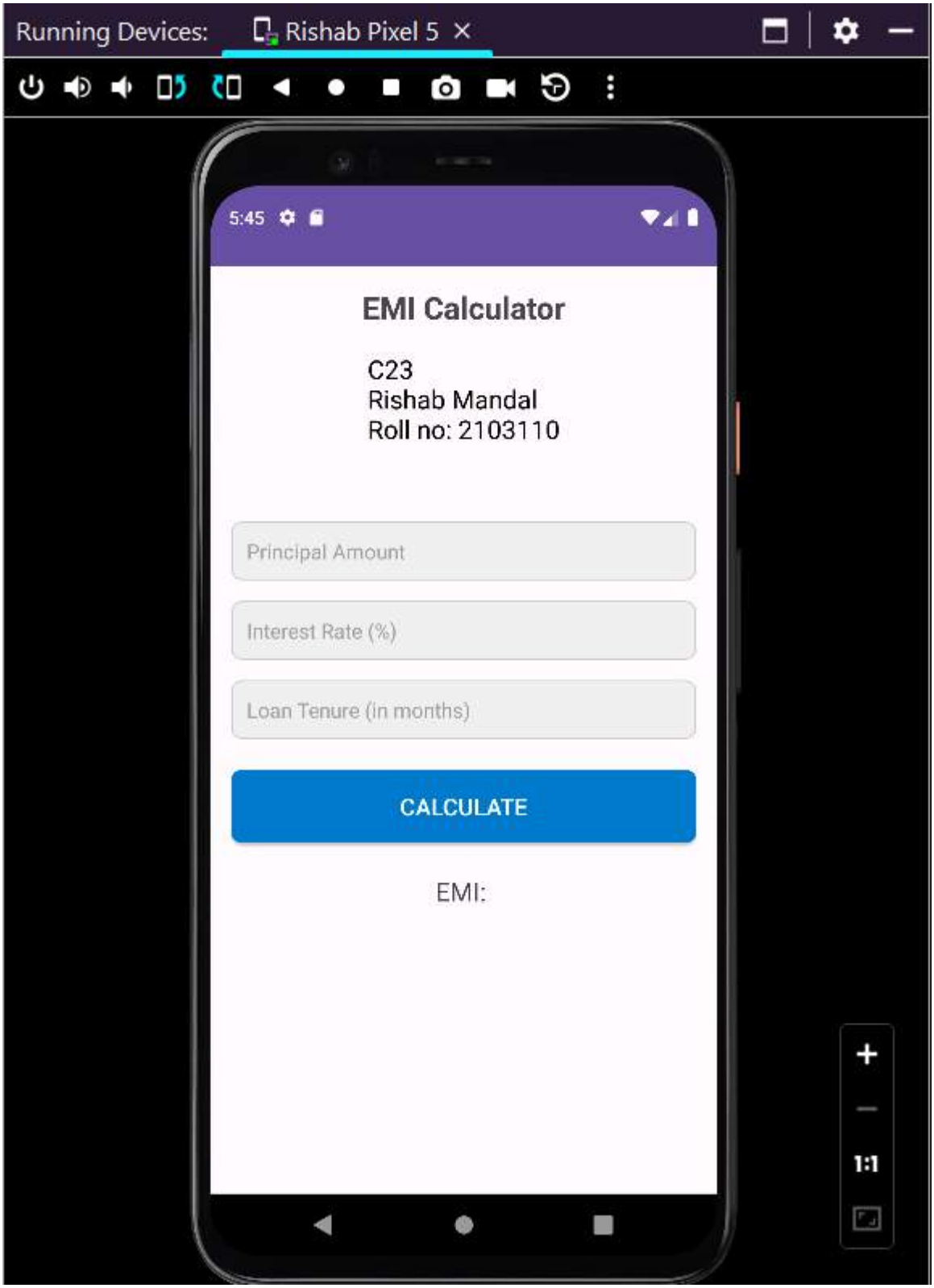
```
<!-- button_bg.xml -->
<shape xmlns:android="http://schemas.android.com/apk/res/android"
    android:shape="rectangle">
    <solid android:color="#007ACC"/> <!-- Button background color -->
    <corners android:radius="8dp"/> <!-- Rounded corners -->
</shape>
```

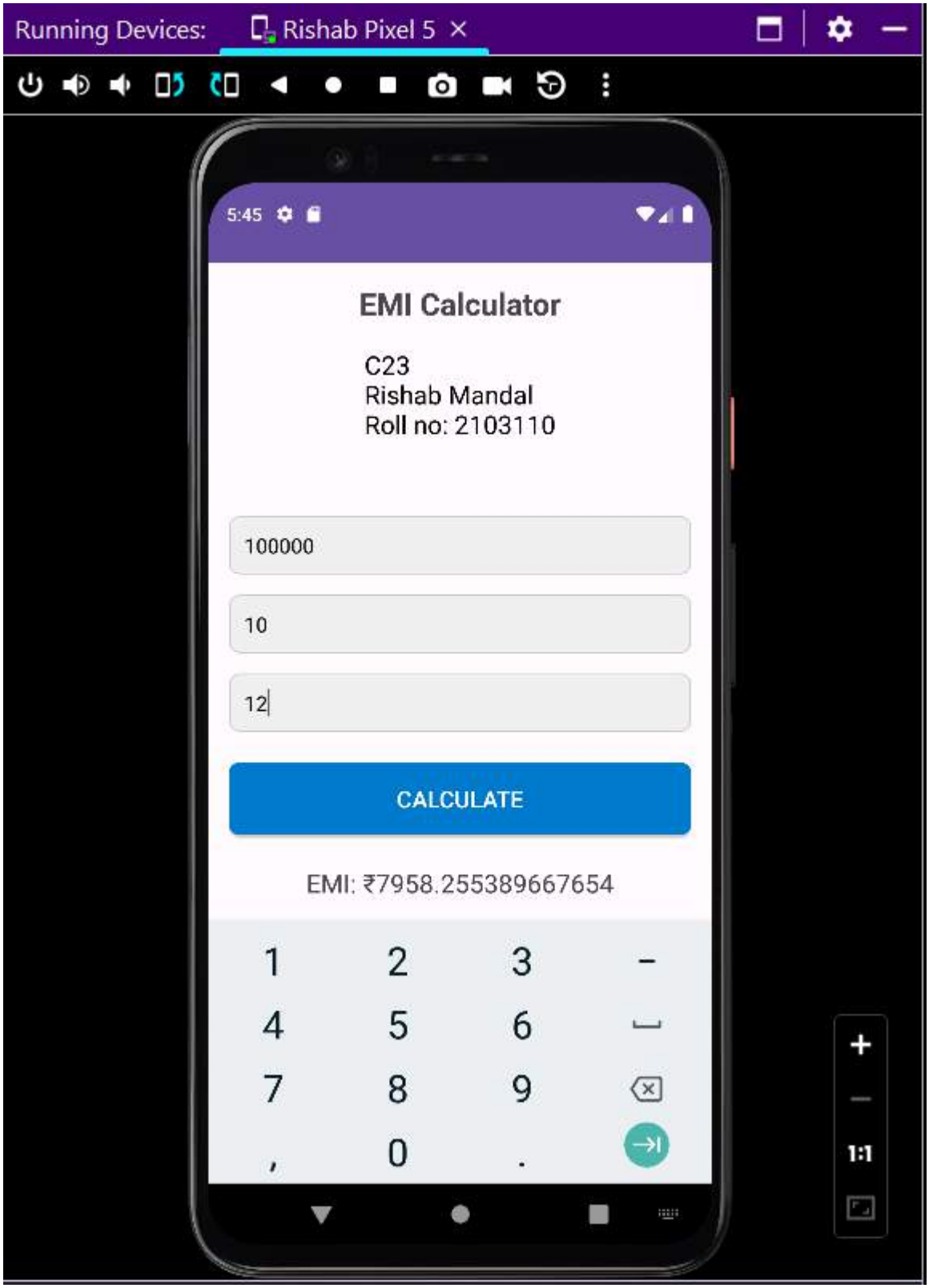
MainActivity.java

```
package com.example.emicaluclator;

import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;

import androidx.appcompat.app.AppCompatActivity;

public class MainActivity extends AppCompatActivity {

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

        final EditText editTextPrincipal =
findViewById(R.id.editTextPrincipal);
        final EditText editTextInterestRate =
findViewById(R.id.editTextInterestRate);
        final EditText editTextTenure = findViewById(R.id.editTextTenure);
        Button buttonCalculate = findViewById(R.id.buttonCalculate);
        final TextView textViewResult = findViewById(R.id.textViewResult);

        buttonCalculate.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                double principal =
Double.parseDouble(editTextPrincipal.getText().toString());
                double interestRate =
Double.parseDouble(editTextInterestRate.getText().toString()) / 100 / 12;
                double tenure =
Double.parseDouble(editTextTenure.getText().toString());

                double emi = calculateEMI(principal, interestRate, tenure);
                textViewResult.setText("EMI: " + emi);
            }
        });
    }

    private double calculateEMI(double principal, double interestRate,
double tenure) {
        double base = 1 + interestRate;
        double power = (-tenure);
        double emi = principal * interestRate * (Math.pow(base, power)) /
(1 - Math.pow(base, power));
```

```
        return emi;
    }
}
```

Output:


Running Devices:
Rishab Pixel 5
5:45
EMI Calculator
C23
Rishab Mandal
Roll no: 2103110
Principal Amount
Interest Rate (%)
Loan Tenure (in months)
CALCULATE
EMI:
1:1


Running Devices:
Rishab Pixel 5
5:45
EMI Calculator
C23
Rishab Mandal
Roll no: 2103110
100000
10
12
CALCULATE
EMI: ₹7958.255389667654
1 2 3 -
4 5 6
7 8 9
, 0 .
1:1

THADOMAL SHAHANI TSEC ENGINEERING COLLEGE

EXPERIMENT NO. 6

Aim :- Write an Android Application that creates an alert on receiving a message.

Theory :-

In Android development, creating an application that triggers an alert upon receiving a message involves utilizing the BroadcastReview component to listen for incoming messages and triggering an alert mechanism upon receipt. This process typically involves registering a dummy button which when clicked, will generate a message and as the message is generated, it triggers an alert, and displays it in an alert dialog.

Program structure :-

activity_main.xml :

The above file is the XML layout code provided which defines the layout for an Android activity's main UI screen. Following are the functions used in the code :

1) Relativelayout :

This is the root layout element used to arrange child views relative to each other or to parent.

THADOMAL SHAHANI TSEC ENGINEERING COLLEGE

2) xmlns : android :

This namespace declaration allows you to use Android-specific attributes in XML layout file.

3) xmlns : tools :

This namespace declaration is used for design-time attributes provided by Android Studio layout editor.

4) android : background :

This attribute sets the background color or drawable for Relative layout

5) android : padding :

This attribute specifies the padding (empty space) inside the Button. In this case, it's set to 15 dp, providing some space between the text and edges of the Button.

Overall, this layout defines a RelativeLayout with a single Button centered in the middle. The Button has rounded corners, a drop shadow effect, and padding, making it visually appealing and interactive.

Rounded_button_background.xml & colors.xml :-

The provided XML files, are used in app development to define custom drawable shapes and colors.

First file defines a drawable shape resource used as background for UI elements, such as buttons. It specifies a rectangle shape with rounded corners, utilizing '<shape>' element with other attributes.

The second file defines color resources that can be referenced throughout the app. It includes 'red' color (color values defined using hexadecimal notation #RRGGBB), allowing developers to maintain consistency in color usage across different UI elements and layouts in the app.

Java Programming Logic :-

This Java code represents an Android application that creates an alert dialog when a button is clicked Following are the functions used :-

1) Package Declaration :

The 'com.example.alertapplication' package declaration indicates name of the application.

2) Import Statements:

These bring in classes from Android framework that are used in the code, eg. dialogs, listeners, etc

3) MainActivity class and onCreate():

This method sets layout for activity using the 'setContentView()' method, passing layout resource 'R.layout.activity_main'.

4) showAlert():

This method creates and shows an alert dialog with a title ("Alert") and a message parameter. It uses an 'AlertDialog.Builder' to construct the dialog and sets a positive button "OK" that dismisses the dialog when clicked.

5) AlertDialog:

This class represents a dialog that can show title, message and buttons, shown using 'show()' method.

Conclusion:

This Android project demonstrates implementation of visually appealing UI featuring a button that triggers an alert dialog.

Exp No.6

Rishab Mandal

Batch: C23

Code:

activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@android:color/white"
    tools:context=".MainActivity">

    <TextView
    android:id="@+id/textView"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_centerHorizontal="true"
    android:text="C23\nRishab Mandal\nRoll no: 2103110"
    android:textColor="@android:color/black"
    android:textSize="20sp"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent" />

    <Button
        android:id="@+id/btnGenerateMessage"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Generate Message"
        android:textColor="@android:color/white"
        android:background="@drawable/rounded_button_background"
        android:elevation="4dp"
        android:padding="15dp"
        android:layout_centerInParent="true"/>

</RelativeLayout>
```

Rounded_button_background.xml

```
<?xml version="1.0" encoding="utf-8"?>
<shape xmlns:android="http://schemas.android.com/apk/res/android"
    android:shape="rectangle">
    <solid android:color="@color/red" />
    <corners android:radius="20dp" />
</shape>
```

colors.xml

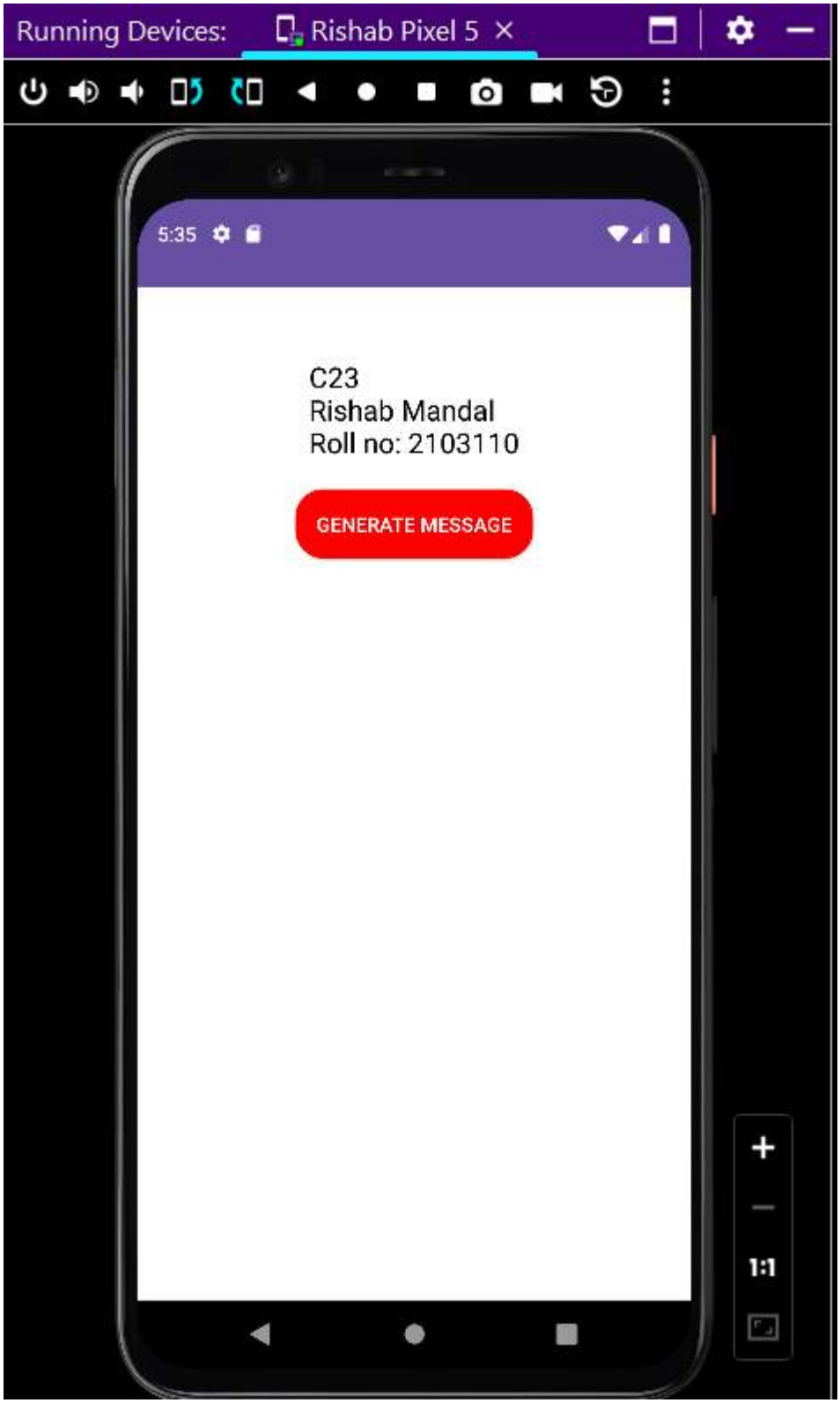
```
<?xml version="1.0" encoding="utf-8"?>
<resources>
```

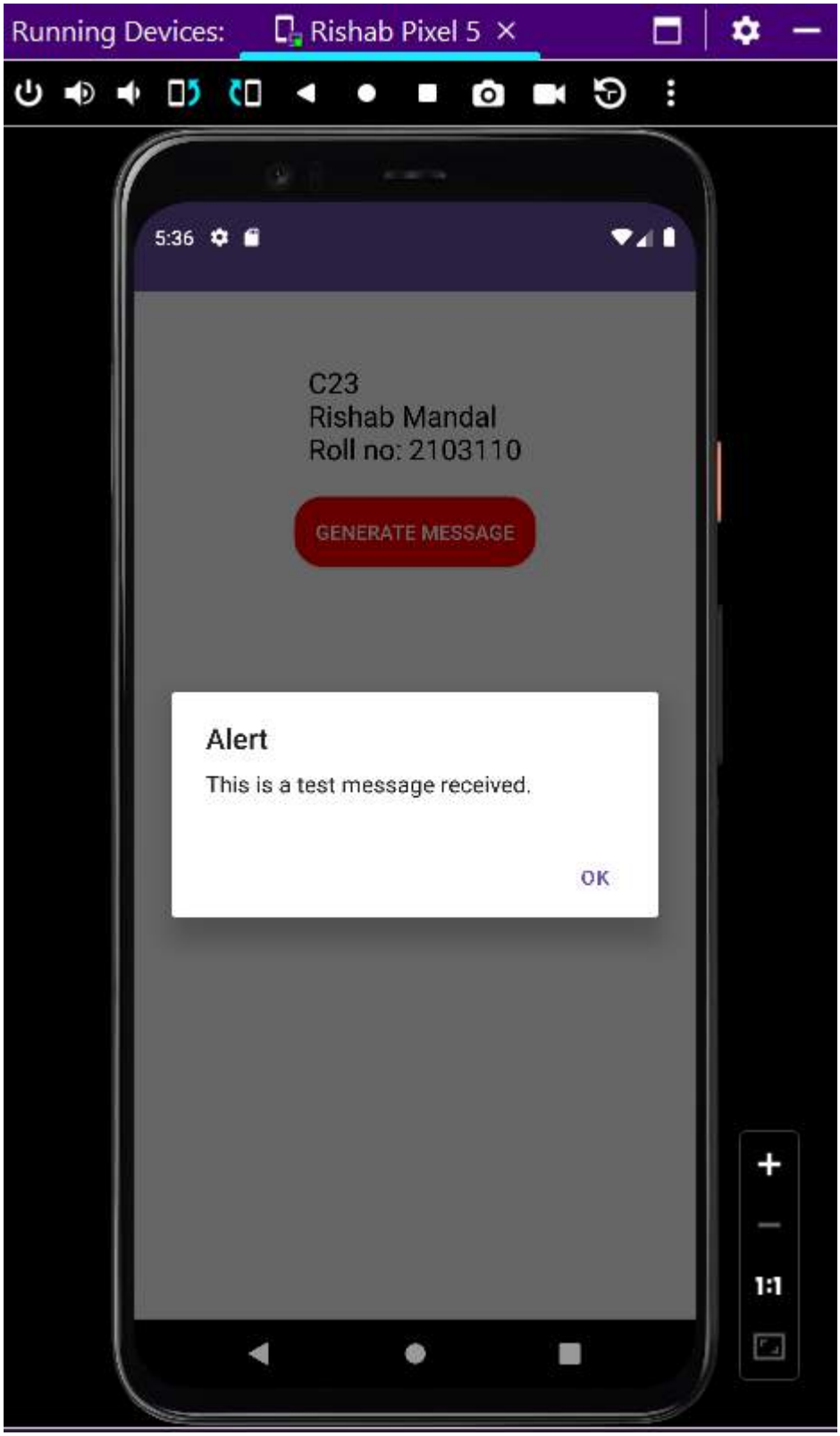
```
    <color name="black">#FF000000</color>
    <color name="white">#FFFFFFFF</color>
    <color name="red">#FF0000</color>
    <color name="colorPrimary">#2196F3</color>
</resources>
```

MainActivity.java

```
package com.example.alertapplication;

import android.app.Activity;
import android.app.AlertDialog;
import android.content.DialogInterface;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;

public class MainActivity extends Activity {

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

        Button btnGenerateMessage = findViewById(R.id.btnGenerateMessage);
        btnGenerateMessage.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                // Generate a test message
                String testMessage = "This is a test message received.";
                // Show alert dialog
                showAlert(testMessage);
            }
        });
    }

    private void showAlert(String message) {
        AlertDialog.Builder builder = new AlertDialog.Builder(this);
        builder.setTitle("Alert");
        builder.setMessage(message);
        builder.setPositiveButton("OK", new
DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialog, int which) {
                dialog.dismiss();
            }
        });
        AlertDialog alert = builder.create();
        alert.show();
    }
}
```

Output:


Running Devices:
Rishab Pixel 5
5:35
C23
Rishab Mandal
Roll no: 2103110
GENERATE MESSAGE
1:1


Running Devices:
Rishab Pixel 5
5:36
C23
Rishab Mandal
Roll no: 2103110
GENERATE MESSAGE
Alert
This is a test message received.
OK
1:1

THADOMAL SHAHANI TSEC ENGINEERING COLLEGE

EXPERIMENT NO. 7

Aim :- Write an Android application to implement Basic Calculator using Android.

Theory :-

The practical aims to create a user-friendly calculator application using Android Studio, leveraging Java for logic implementation and XML for interface design. The app will feature numeric buttons (0-9), operation buttons (+, -, *, /), a clear button (AC), and an equal button (=) to perform basic arithmetic operations. The primary objectives include designing an intuitive interface, accurate calculations with proper error handling.

XML Layout Design :

The app's XML layout represents the UI design for a basic calculator Android app using ConstraintLayout in Android Studio. The layout consists of a TextView at top to display results and several rows of buttons organized in LinearLayouts. Each row represents a set of numeric and operator buttons. The form includes 'res/layout/activity_main.xml' which essentially defines the visual structure of the app. It also uses two drawable files for giving styles to the buttons, 'button_bg.xml' for all buttons, and 'yellow_button_bg.xml' for especially a yellow button.

The root layout used in this XML file is ConstraintLayout. The TextView is used to display the result of calculations. Attributes such as 'android:textSize="48sp"' determines the font size, and 'app:layout_constraintTop_toTopOf="parent"' ensures it is constrained to top of parent layout. LinearLayouts are used to organize buttons into rows. AppCompatButtons elements represent buttons for numerics and operators. Buttons also have styles applied to them using 'style="@style/CalculatorButtonStyle"' to maintain a consistent appearance throughout the application.

Java Programming logic :-

Firstly, the most important file (hub of the program) is 'src/main/java/com.example.basiccalculator/MainActivity.java'. To connect with particular XML, the 'onCreate' methods are initialized.

Built-in Functions :

1) findViewById() :

This method is used to retrieve UI components from layout XML file by their respective IDs. It is a built-in method provided by Android framework.

2) setOnClickListener():

This method is used to set click event listeners on buttons, enabling them to respond to user interactions.

3) Integer.parseInt():

This built-in method converts a string representation of a number to its integer equivalent. It is used to extract numeric value from button text when a digit button is clicked.

4) String.valueOf():

This method converts various types of values, including integers, to their string representation.

5) append():

This method is part of TextView class and is used to append text to existing text in TextView. It is used to dynamically update the displayed expression as the user inputs digits and operators.

6) setText():

This method is used to set the text content

of a TextView. It is used to display error messages, clear the TextView, or display the final result.

7) Logical operators (&&):

These built-in operators are used for the logical AND operators. They are used in conditional statements to evaluate the multiple conditions together.

These built-in functions and methods which are provided by the Android SDK and Java language are essential for implementing the core functionality of the calculator application, including user interaction handling, data manipulation, and error checking.

Conclusion :-

Thus, we developed a basic calculator by understanding and implementing different built-in functions and XML Layout functions and attributes provided by the Android Studio for effective development.

Exp No.7

Rishab Mandal

Batch: C23

Code:

activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <TextView
    android:id="@+id/textView"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_centerHorizontal="true"
    android:text="C23\nRishab Mandal\nRoll no: 2103110"
    android:textColor="@android:color/black"
    android:textSize="20sp"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent" />

    <TextView
        android:id="@+id/textViewResult"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:text="0"
        android:textSize="60sp"
        android:textAlignment="textEnd"
        android:layout_marginBottom="32dp"
        android:padding="10dp"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintBottom_toTopOf="@+id/row5" />

    <!-- Row 5 -->
    <LinearLayout
        android:id="@+id/row5"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="horizontal"
        app:layout_constraintTop_toBottomOf="@id/textViewResult">

        <androidx.appcompat.widget.AppCompatButton
            android:id="@+id/buttonOpenBracket"
            style="@style/CalculatorButtonStyle"
            android:text="(" />
```

```
    <androidx.appcompat.widget.AppCompatButton
        android:id="@+id/buttonCloseBracket"
        style="@style/CalculatorButtonStyle"
        android:text=")" />

    <androidx.appcompat.widget.AppCompatButton
        android:id="@+id/buttonPercentage"
        style="@style/CalculatorButtonStyle"
        android:text="%" />

    <androidx.appcompat.widget.AppCompatButton
        android:id="@+id/buttonBackspace"
        style="@style/CalculatorButtonStyle"
        android:text=" " />

    <!-- Add more buttons as needed -->
</LinearLayout>

<!-- Row 4 -->
<LinearLayout
    android:id="@+id/row4"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="horizontal"
    app:layout_constraintTop_toBottomOf="@id/row5">

    <!-- Add more buttons as needed -->
</LinearLayout>

<!-- Row 3 -->
<LinearLayout
    android:id="@+id/row3"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="horizontal"
    app:layout_constraintTop_toBottomOf="@id/row4">

    <androidx.appcompat.widget.AppCompatButton
        android:id="@+id/button7"
        style="@style/CalculatorButtonStyle"
        android:text="7" />

    <androidx.appcompat.widget.AppCompatButton
        android:id="@+id/button8"
        style="@style/CalculatorButtonStyle"
        android:text="8" />

    <androidx.appcompat.widget.AppCompatButton
        android:id="@+id/button9"
        style="@style/CalculatorButtonStyle"
        android:text="9" />

    <androidx.appcompat.widget.AppCompatButton
        android:id="@+id/buttonAdd"
        style="@style/CalculatorButtonStyle"
        android:text="+" />

    <!-- Add more buttons as needed -->
</LinearLayout>

<!-- Operator Row -->
```

```
<LinearLayout
    android:id="@+id/row2"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="horizontal"
    app:layout_constraintTop_toBottomOf="@+id/row3">

    <androidx.appcompat.widget.AppCompatButton
        android:id="@+id/button4"
        style="@style/CalculatorButtonStyle"
        android:text="4" />

    <androidx.appcompat.widget.AppCompatButton
        android:id="@+id/button5"
        style="@style/CalculatorButtonStyle"
        android:text="5" />

    <androidx.appcompat.widget.AppCompatButton
        android:id="@+id/button6"
        style="@style/CalculatorButtonStyle"
        android:text="6" />

    <androidx.appcompat.widget.AppCompatButton
        android:id="@+id/buttonSubtract"
        style="@style/CalculatorButtonStyle"
        android:text="-" />

    <!-- Add more buttons as needed -->
</LinearLayout>

<LinearLayout
    android:id="@+id/row1"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="horizontal"
    app:layout_constraintTop_toBottomOf="@+id/row2">

    <androidx.appcompat.widget.AppCompatButton
        android:id="@+id/button1"
        style="@style/CalculatorButtonStyle"
        android:text="1" />

    <androidx.appcompat.widget.AppCompatButton
        android:id="@+id/button2"
        style="@style/CalculatorButtonStyle"
        android:text="2" />

    <androidx.appcompat.widget.AppCompatButton
        android:id="@+id/button3"
        style="@style/CalculatorButtonStyle"
        android:text="3" />

    <androidx.appcompat.widget.AppCompatButton
        android:id="@+id/buttonMultiply"
        style="@style/CalculatorButtonStyle"
        android:text="*" />

    <!-- Add more buttons as needed -->
</LinearLayout>
```

```
    <LinearLayout
        android:id="@+id/operatorRow"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="horizontal"
        app:layout_constraintTop_toBottomOf="@id/row1">

        <androidx.appcompat.widget.AppCompatButton
            android:id="@+id/buttonAC"
            style="@style/CalculatorButtonStyle"
            android:text="AC" />

        <androidx.appcompat.widget.AppCompatButton
            android:id="@+id/button0"
            style="@style/CalculatorButtonStyle"
            android:text="0" />

        <androidx.appcompat.widget.AppCompatButton
            android:id="@+id/buttonDivide"
            style="@style/CalculatorButtonStyle"
            android:text="/" />

        <!-- Equals Button -->
        <androidx.appcompat.widget.AppCompatButton
            android:id="@+id/buttonEquals"
            android:background="@drawable/yellow_button_bg"
            android:layout_width="75dp"
            android:layout_height="wrap_content"
            android:layout_margin="8dp"
            android:textSize="20sp"
            android:textAllCaps="false"
            android:padding="16dp"
            android:text="="
            app:layout_constraintEnd_toEndOf="parent"
            app:layout_constraintStart_toStartOf="parent"
            app:layout_constraintTop_toBottomOf="@id/operatorRow" />

        <!-- Add more operator buttons as needed -->
    </LinearLayout>



</androidx.constraintlayout.widget.ConstraintLayout>
```

styles.xml

```
<?xml version="1.0" encoding="utf-8"?>
<!-- res/values/styles.xml -->
<resources>
    <style name="CalculatorButtonStyle" parent="Widget.AppCompat.Button">
        <item name="android:layout_width">0dp</item>
        <item name="android:layout_weight">1</item>
        <item name="android:layout_height">wrap_content</item>
        <item name="android:layout_margin">8dp</item>
        <item name="android:textSize">24sp</item>
        <item name="android:textColor">@android:color/white</item>
        <item name="android:background">@drawable/button_bg</item>
        <item name="android:textAllCaps">false</item>
        <item name="android:padding">16dp</item>
```

```
    </style>
</resources>
```

button_bg

```
<!-- button_bg.xml -->
<shape xmlns:android="http://schemas.android.com/apk/res/android"
    android:shape="rectangle">
    <solid android:color="#007ACC"/> <!-- Button background color -->
    <corners android:radius="8dp"/> <!-- Rounded corners -->
</shape>
```

yellow_button_bg

```
<?xml version="1.0" encoding="utf-8"?>
<!-- res/drawable/yellow_button_bg.xml -->
<shape xmlns:android="http://schemas.android.com/apk/res/android"
    android:shape="rectangle">
    <solid android:color="#FFD700" /> <!-- Yellow color -->
    <corners android:radius="8dp" /> <!-- Rounded corners -->
</shape>
```

MainActivity.java

```
package com.example.basiccalculator;

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 implements
View.OnClickListener {

    private TextView textViewResult;
    private Button buttonAC, button0, button1, button2, button3, button4,
button5, button6, button7, button8, button9, buttonAdd, buttonEquals,
buttonSubtract, buttonMultiply, buttonDivide;

    private Button buttonBackspace, buttonOpenBracket, buttonCloseBracket,
buttonPercentage;
    private float num1 = 0, num2 = 0; private float result = 0;
    private boolean isAdditionClicked = false;
    private boolean isSubtractionClicked = false;
    private boolean isMultiplicationClicked = false;
    private boolean isDivisionClicked = false;

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

        textViewResult = findViewById(R.id.textViewResult);
        button0 = findViewById(R.id.button0);
        button1 = findViewById(R.id.button1);
        button2 = findViewById(R.id.button2);
        button3 = findViewById(R.id.button3);
```

```
        button4 = findViewById(R.id.button4);
        button5 = findViewById(R.id.button5);
        button6 = findViewById(R.id.button6);
        button7 = findViewById(R.id.button7);
        button8 = findViewById(R.id.button8);
        button9 = findViewById(R.id.button9);
        buttonAC = findViewById(R.id.buttonAC);
        buttonAdd = findViewById(R.id.buttonAdd);
        buttonEquals = findViewById(R.id.buttonEquals);
        buttonSubtract = findViewById(R.id.buttonSubtract);
        buttonMultiply = findViewById(R.id.buttonMultiply);
        buttonDivide = findViewById(R.id.buttonDivide);
        buttonBackspace = findViewById(R.id.buttonBackspace);
        buttonOpenBracket = findViewById(R.id.buttonOpenBracket);
        buttonCloseBracket = findViewById(R.id.buttonCloseBracket);
        buttonPercentage = findViewById(R.id.buttonPercentage);

        button0.setOnClickListener(this);
        button1.setOnClickListener(this);
        button2.setOnClickListener(this);
        button3.setOnClickListener(this);
        button4.setOnClickListener(this);
        button5.setOnClickListener(this);
        button6.setOnClickListener(this);
        button7.setOnClickListener(this);
        button8.setOnClickListener(this);
        button9.setOnClickListener(this);
        buttonAC.setOnClickListener(this);
        buttonAdd.setOnClickListener(this);
        buttonEquals.setOnClickListener(this);
        buttonSubtract.setOnClickListener(this);
        buttonMultiply.setOnClickListener(this);
        buttonDivide.setOnClickListener(this);
        buttonBackspace.setOnClickListener(this);
        buttonOpenBracket.setOnClickListener(this);
        buttonCloseBracket.setOnClickListener(this);
        buttonPercentage.setOnClickListener(this);
    }

    @Override
    public void onClick(View v) {
        int buttonId = v.getId();
        if (buttonId == R.id.button1 || buttonId == R.id.button2 ||
buttonId == R.id.button3 ||
                buttonId == R.id.button4 || buttonId == R.id.button5 ||
buttonId == R.id.button6 ||
                buttonId == R.id.button7 || buttonId == R.id.button8 ||
buttonId == R.id.button9 ||
                buttonId == R.id.button0) {
            int digit = Integer.parseInt(((Button)
v).getText().toString());
            if (!isAdditionClicked && !isSubtractionClicked &&
!isMultiplicationClicked && !isDivisionClicked) {
                num1 = num1 * 10 + digit;
                textViewResult.append(String.valueOf(digit));
            } else {
                num2 = num2 * 10 + digit;
                textViewResult.append(String.valueOf(digit));
            }
        } else if (buttonId == R.id.buttonAdd || buttonId ==
R.id.buttonSubtract || buttonId == R.id.buttonMultiply || buttonId ==
```

```
R.id.buttonDivide) {
            // If an operator button is clicked, set corresponding flag and
display the operator
            isAdditionClicked = buttonId == R.id.buttonAdd;
            isSubtractionClicked = buttonId == R.id.buttonSubtract;
            isMultiplicationClicked = buttonId == R.id.buttonMultiply;
            isDivisionClicked = buttonId == R.id.buttonDivide;
            char operator = ' ';
            if (isAdditionClicked) {
                operator = '+';
            } else if (isSubtractionClicked) {
                operator = '-';
            } else if (isMultiplicationClicked) {
                operator = '*';
            } else if (isDivisionClicked) {
                operator = '/';
            }
            textViewResult.append(" " + operator + " "); // Append the
operator to the TextView with spaces
        } else if (buttonId == R.id.buttonEquals) {
            // Perform the calculation based on the operator clicked
            if (isAdditionClicked) {
                result = num1 + num2;
            } else if (isSubtractionClicked) {
                result = num1 - num2;
            } else if (isMultiplicationClicked) {
                result = num1 * num2;
            } else if (isDivisionClicked) {
                if (num2 != 0) {
                    result = num1 / num2;
                } else {
                    textViewResult.setText("Error: Division by zero");
                    return; // Exit onClick() early if division by zero
                }
            }
            // Display the result
            textViewResult.append(" = " + String.valueOf(result)); //
Append the result to the TextView
            num1 = result;
            num2 = 0;
        } else if (buttonId == R.id.buttonAC) {
            // Clear all variables and text view
            num1 = 0;
            num2 = 0;
            result = 0;
            isAdditionClicked = false;
            isSubtractionClicked = false;
            isMultiplicationClicked = false;
            isDivisionClicked = false;
            textViewResult.setText("");
        } else if (buttonId == R.id.buttonBackspace) {
            // Remove the last character from the text view
            String currentText = textViewResult.getText().toString();
            if (!currentText.isEmpty()) {
                textViewResult.setText(currentText.substring(0,
currentText.length() - 1));
            }
        } else if (buttonId == R.id.buttonOpenBracket) {
            // Append an open bracket '(' to the text view
            textViewResult.append("(");
        } else if (buttonId == R.id.buttonCloseBracket) {
```

```
            // Append a close bracket ')' to the text view
            textViewResult.append(")");
        } else if (buttonId == R.id.buttonPercentage) {
            // Append a percentage sign '%' to the text view
            textViewResult.append("%");
        } else {
            // If any other button is clicked, display syntax error
            textViewResult.setText("Syntax Error");
        }
    }

}
```

Output:


Running Devices
Rishab Pixel 5
G23
Rishab Mandal
Roll no: 2103110
1 + 3 = 4.0
(
)
%
7
8
9
*
4
5
6
-
1
2
3
+
AC
0
/
=


Running Devices
Rishab Pixel 5
C23
Rishab Mandal
Roll no: 2103110
6 / 0
( ) % ⌫
7 8 9 +
4 5 6 -
1 2 3 *
AC 0 / =


Running Devices
Rishab Pixel 5
C23
Rishab Mandal
Roll no: 2103110
Error: Division by zero
( ) % ⌫
7 8 9 +
4 5 6 -
1 2 3 *
AC 0 / =

THADOMAL SHAHANI TSEC ENGINEERING COLLEGE

EXPERIMENT NO. 8

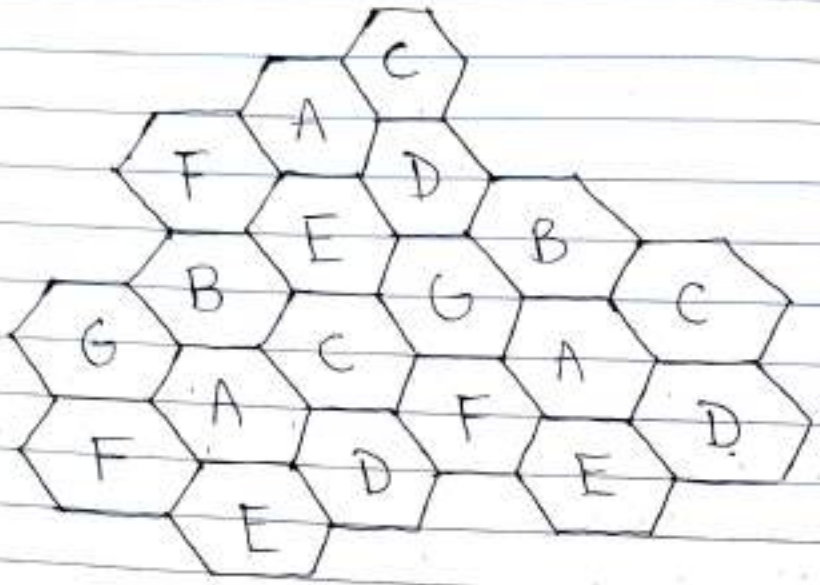
Aim :- Write a program to demonstrate Cellular frequency reuse.

Theory :-

Frequency Reuse is the scheme in which allocation and reuse of channels throughout a coverage region is done.
Each cellular base station is allocated a group of radio channels or Frequency sub-bands to be used within a small geographic area known as cell.
The shape of cell is Hexagonal.
The process of selecting and allocating the frequency sub-bands for all of the cellular base station within a system is called Frequency reuse or Frequency Planning.

Advantages of Frequency Reuse :

It improves Quality of Service (QoS).
In Frequency Reuse scheme, total bandwidth is divided into different sub-bands that are used by the cells.
Frequency reuse scheme allow operators to reuse the same frequencies at different cell sites.



Cell with same letter uses same set of channels group or frequencies sub-band.

To find the total number of channels allocated to a cell,

S = Total number of duplex channels available to use

k = Channels allocated to each cell ($k < S$)

N = Total number of cells or Cluster Size.

Thus, $S = kN$

Frequency Reuse Factor = $1/N$

The value of N is calculated by following formula:

$N = I^2 + I * J + J^2$

where I, J : Positive integers indicating position of cell

N : Total number of cells / size of cluster

If a Cluster is replicated or repeated M times, then Capacity, C, will be,

$$C = MkN = MS \qquad (\because S = kN)$$

Conclusion :-

Thus, implemented program to demonstrate frequency reuse (cellular) in mobile computing.

Exp No. 8

Rishab Mandal

Batch: C23

Code:

Exp8FrequencyReuse.py :

```
#!/usr/bin/python

from math import *

# import everything from Tkinter module
from tkinter import *

# Base class for Hexagon shape
class Hexagon(object):
        def __init__(self, parent, x, y, length, color, tags):
                self.parent = parent
                self.x = x
                self.y = y
                self.length = length
                self.color = color
                self.size = None
                self.tags = tags
```

```
        self.draw_hex()


# draw one hexagon
def draw_hex(self):
        start_x = self.x
        start_y = self.y
        angle = 60
        coords = []
        for i in range(6):
                end_x = start_x + self.length * cos(radians(angle * i))
                end_y = start_y + self.length * sin(radians(angle * i))
                coords.append([start_x, start_y])
                start_x = end_x
                start_y = end_y
        self.parent.create_polygon(coords[0][0],
                                                coords[0][1],
                                                coords[1][0],
                                                coords[1][1],
                                                coords[2][0],
                                                coords[2][1],
                                                coords[3][0],
                                                coords[3][1],
                                                coords[4][0],
                                                coords[4][1],
                                                coords[5][0],
```

```
                                                    coords[5][1],
                                                    fill=self.color,
                                                    outline="black",
                                                    tags=self.tags)

# class holds frequency reuse logic and related methods
class FrequencyReuse(Tk):
        CANVAS_WIDTH = 800
        CANVAS_HEIGHT = 650
        TOP_LEFT = (20, 20)
        BOTTOM_LEFT = (790, 560)
        TOP_RIGHT = (780, 20)
        BOTTOM_RIGHT = (780, 560)

        def __init__(self, cluster_size, columns=16, rows=10, edge_len=30):
                Tk.__init__(self)
                self.textbox = None
                self.curr_angle = 330
                self.first_click = True
                self.reset = False
                self.edge_len = edge_len
                self.cluster_size = cluster_size
                self.reuse_list = []
                self.all_selected = False
                self.curr_count = 0
```

```
        self.hexagons = []
        self.co_cell_endp = []
        self.reuse_xy = []
        self.canvas = Canvas(self,
                                        width=self.CANVAS_WIDTH,
                                        height=self.CANVAS_HEIGHT,
                                        bg="#4dd0e1")
        self.canvas.bind("<Button-1>", self.call_back)
        self.canvas.focus_set()
        self.canvas.bind('<Shift-R>', self.resets)
        self.canvas.pack()
        self.title("Frequency reuse and co-channel selection")
        self.create_grid(16, 10)
        self.create_textbox()
        self.cluster_reuse_calc()

# show lines joining all co-channel cells
def show_lines(self):
        # center(x,y) of first hexagon
        approx_center = self.co_cell_endp[0]
        self.line_ids = []
        for k in range(1, len(self.co_cell_endp)):

                end_xx = (self.co_cell_endp[k])[0]
                end_yy = (self.co_cell_endp[k])[1]
```

```
                    # move i^th steps
                    l_id = self.canvas.create_line(approx_center[0],
approx_center[1],
                                                   end_xx,
end_yy)
                    if j == 0:
                        self.line_ids.append(l_id)
                        dist = 0
                    elif i >= j and j != 0:
                        self.line_ids.append(l_id)
                        dist = j
                        # rotate counter-clockwise and move j^th step
                        l_id = self.canvas.create_line(
                            end_xx, end_yy, end_xx + self.center_dist *
dist *
                            cos(radians(self.curr_angle - 60)),
                            end_yy + self.center_dist * dist *
                            sin(radians(self.curr_angle - 60)))
                        self.line_ids.append(l_id)
                    self.curr_angle -= 60

    def create_textbox(self):
        txt = Text(self.canvas,
                    width=80,
                    height=1,
```

```
                    font=("Helvatica", 12),
                    padx=10,
                    pady=10)
        txt.tag_configure("center", justify="center")
        txt.insert("1.0", "Select a Hexagon")
        txt.tag_add("center", "1.0", "end")
        self.canvas.create_window((0, 600), anchor='w', window=txt)
        txt.config(state=DISABLED)
        self.textbox = txt

def resets(self, event):
        if event.char == 'R':
                self.reset_grid()

# clear hexagonal grid for new i/p
def reset_grid(self, button_reset=False):
        self.first_click = True
        self.curr_angle = 330
        self.curr_count = 0
        self.co_cell_endp = []
        self.reuse_list = []
        for i in self.hexagons:
                self.canvas.itemconfigure(i.tags, fill=i.color)

        try:
```

```
                self.line_ids
            except AttributeError:
                pass
            else:
                for i in self.line_ids:
                    self.canvas.after(0, self.canvas.delete, i)
                self.line_ids = []

            if button_reset:
                self.write_text("Select a Hexagon")

        # create a grid of Hexagons
        def create_grid(self, cols, rows):
            size = self.edge_len
            for c in range(cols):
                if c % 2 == 0:
                    offset = 0
                else:
                    offset = size * sqrt(3) / 2
                for r in range(rows):
                    x = c * (self.edge_len * 1.5) + 50
                    y = (r * (self.edge_len * sqrt(3))) + offset + 15
                    hx = Hexagon(self.canvas, x, y, self.edge_len,
"#fafafa",
                                 "{},{}".format(r, c))
```

```
                self.hexagons.append(hx)

        # calculate reuse distance, center distance and radius of the
hexagon
        def cluster_reuse_calc(self):
                self.hex_radius = sqrt(3) / 2 * self.edge_len
                self.center_dist = sqrt(3) * self.hex_radius
                self.reuse_dist = self.hex_radius * sqrt(3 * self.cluster_size)

        def write_text(self, text):
                self.textbox.config(state=NORMAL)
                self.textbox.delete('1.0', END)
                self.textbox.insert('1.0', text, "center")
                self.textbox.config(state=DISABLED)

        #check if the co-channels are within visible canvas
        def is_within_bound(self, coords):
                if self.TOP_LEFT[0] < coords[0] < self.BOTTOM_RIGHT[0] \
                and self.TOP_RIGHT[1] < coords[1] < self.BOTTOM_RIGHT[1]:
                        return True
                return False

        #gets called when user selects a hexagon
        #This function applies frequency reuse logic in order to
        #figure out the positions of the co-channels
```

```
        def call_back(self, evt):

                selected_hex_id = self.canvas.find_closest(evt.x, evt.y)[0]
                hexagon = self.hexagons[int(selected_hex_id - 1)]
                s_x, s_y = hexagon.x, hexagon.y
                approx_center = (s_x + 15, s_y + 25)

                if self.first_click:
                        self.first_click = False
                        self.write_text(
                                """Now, select another hexagon such
                                that it should be a co-cell of
                                the original hexagon."""
                        )
                        self.co_cell_endp.append(approx_center)
                        self.canvas.itemconfigure(hexagon.tags, fill="green")

                        for _ in range(6):

                                end_xx = approx_center[0] + self.center_dist * i *
cos(
                                        radians(self.curr_angle))
                                end_yy = approx_center[1] + self.center_dist * i *
sin(
                                        radians(self.curr_angle))
```

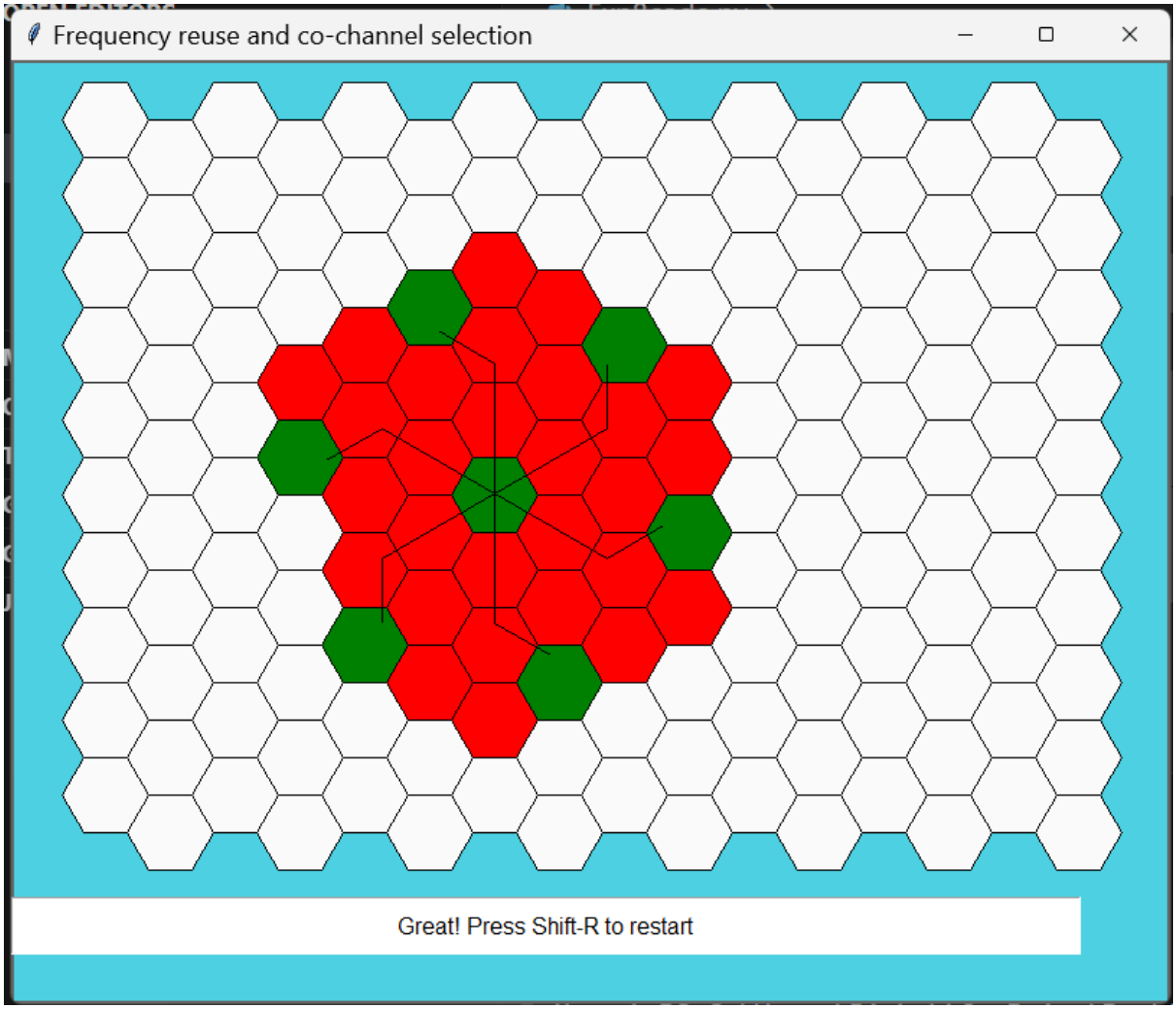
```
                    reuse_x = end_xx + (self.center_dist * j) * cos(
                        radians(self.curr_angle - 60))
                    reuse_y = end_yy + (self.center_dist * j) * sin(
                        radians(self.curr_angle - 60))

                    if not self.is_within_bound((reuse_x, reuse_y)):
                        self.write_text(
                            """co-cells are exceeding canvas
boundary.
                            Select cell in the center"""
                        )
                        self.reset_grid()
                        break

                    if j == 0:
                        self.reuse_list.append(
                            self.canvas.find_closest(end_xx,
end_yy)[0])
                    elif i >= j and j != 0:
                        self.reuse_list.append(
                            self.canvas.find_closest(reuse_x,
reuse_y)[0])

                    self.co_cell_endp.append((end_xx, end_yy))
                    self.curr_angle -= 60
```

```
            else:
                curr = self.canvas.find_closest(s_x, s_y)[0]
                if curr in self.reuse_list:
                    self.canvas.itemconfigure(hexagon.tags,
fill="green")
                    self.write_text("Correct! Cell {} is a co-
cell.".format(
                        hexagon.tags))
                    if self.curr_count == len(self.reuse_list) - 1:
                        self.write_text("Great! Press Shift-R to
restart")
                        self.show_lines()
                    self.curr_count += 1

                else:
                    self.write_text("Incorrect! Cell {} is not a co-
cell.".format(
                        hexagon.tags))
                    self.canvas.itemconfigure(hexagon.tags,
fill="red")

if __name__ == '__main__':
    print(
        """Enter i & j values. common (i,j) values are:
        (1,0), (1,1), (2,0), (2,1), (3,0), (2,2)"""
    )
```

```
i = int(input("Enter i: "))
j = int(input("Enter j: "))
if i == 0 and j == 0:
        raise ValueError("i & j both cannot be zero")
elif j > i:
        raise ValueError("value of j cannot be greater than i")
else:
        N = (i**2 + i * j + j**2)
        print("N is {}".format(N))
freqreuse = FrequencyReuse(cluster_size=N)
freqreuse.mainloop()
```

Output:

```
(base) PS C:\Users\Rishab\OneDrive\Desktop\MCC Exp Documents> python Exp8code.py
Enter i & j values. common (i,j) values are:
        (1,0), (1,1), (2,0), (2,1), (3,0), (2,2)
Enter i: 2
Enter j: 1
N is 7
```


Frequency reuse and co-channel selection
Great! Press Shift-R to restart

THADOMAL SHAHANI TSEC ENGINEERING COLLEGE

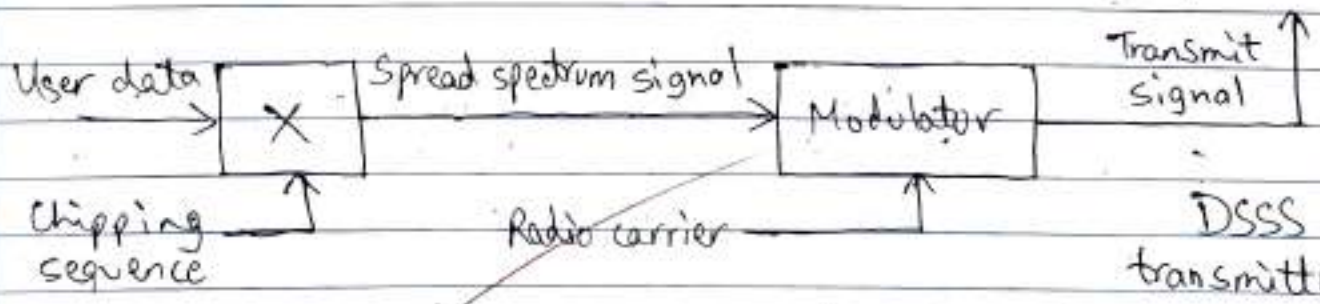
EXPERIMENT NO. 9

Aim :- Write a program to explain concept of DSSS.

Theory :-

Spread spectrum includes techniques involving spreading bandwidth needed to transmit data, reducing narrowband interference.
Direct sequence Spread Spectrum (DSSS) systems take a user bit stream and perform an (XOR) with a so-called chipping sequence.
Wireless systems use the sequence 10110111000, called as Barker's code. These Barker codes exhibit a good robustness against interference and insensitivity to multi-path propagation.

The first step in a DSSS transmitter, is the spreading of user data with chipping sequence (digital modulation).
The spread signal is then modulated with a radio carrier (radio modulation).

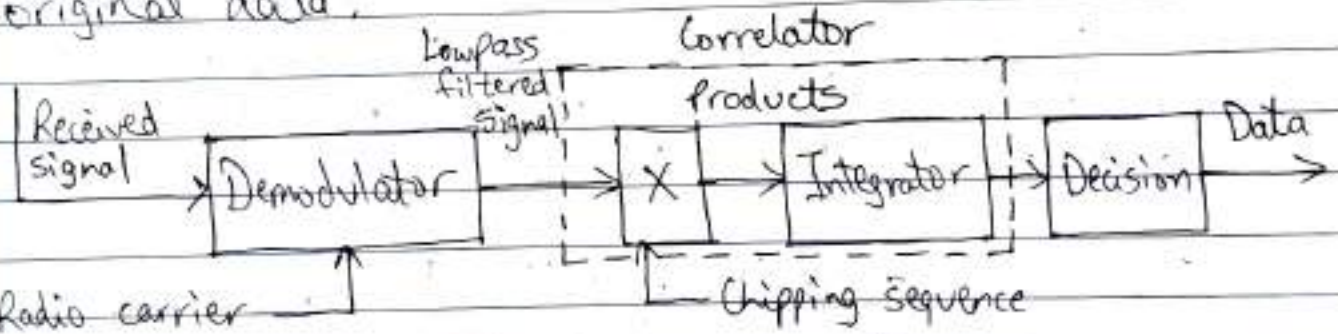


Assuming a user signal with a bandwidth of 1 Mhz, spreading it with 11-chip Barker code would result

in a signal with 11 MHz bandwidth. The radio carrier then shifts it to carrier frequency eg. 2.4GHz. This signal is then transmitted.

DSSS Receiver :

The receiver has to perform inverse functions of two transmitter modulation steps. However, noise and multi-path propagation require additional mechanisms to reconstruct the original data.



First step in the receiver involves demodulating the received signal. This results in a signal with approximately same bandwidth as original spread spectrum signal.

The receiver has to know original chipping sequence, sequences have to be precisely synchronized since receiver calculates product of chip with incoming signal.

This comprises XOR operation. An integrator adds all these products.

Calculating products of chips and signal, adding products in an integrator is also called correlation.

Finally, in each bit period a decision unit samples sums generated by integrator and decides if this sum represents a binary 1 or a 0.

Eg. Transmission of user data 01.

Soln

User Data	0	1
XOR with Barker code	10110111000	10110111000
Spread Spectrum	10110111000	01001000111

They are concatenated to 22 digits and sent.
At receiver, perform XOR operations on received signal with same Barker's code.

Received signal	10110111000	01001000111
XOR with Barker code	10110111000	10110111000
Result	00000000000	11111111111

Now, result is given to integrator which performs sum of products. Sum of products for first part is 0 and next in 11. The decision unit maps sums less than 4 to binary 0, and sums larger than 7 to binary 1.

This constitutes the original user data i.e 01.

Conclusion :-

Thus, understood the concept of Direct Sequence Spread Spectrum (DSSS) and its significance in Mobile Computing.

Exp No.9

Rishab Mandal

Batch: C23

Code:

Exp9DSSS.java :

```
import java.util.Arrays;

public class Exp9DSSS {

    public static void main(String[] args) {
        // Original data signal (binary representation)
        // System.out.print("Enter the length of data signal: ");
        int[] dataSignal = { 0, 1 };

        // Spreading code (PN sequence) Barker's code
        int[] spreadingCode = { 1, 0, 1, 1, 0, 1, 1, 1, 0, 0, 0 };

        // Spread the data signal using DSSS
        int[] spreadSignal = spreadDSSS(dataSignal, spreadingCode);

        // Display the results
```

```
        System.out.println("Original Data Signal: " +
Arrays.toString(dataSignal));
        System.out.println("Spreading Code (PN Sequence): " +
Arrays.toString(spreadingCode));
        System.out.println("Spread Signal: " + Arrays.toString(spreadSignal));

        // Recover the original signal by despread
        int[] recoveredSignal = despreadDSSS(spreadSignal, spreadingCode);

        // Display the recovered signal
        System.out.println("Recovered Signal: " +
Arrays.toString(recoveredSignal));
    }

    private static int[] spreadDSSS(int[] dataSignal, int[] spreadingCode) {
        int[] spreadSignal = new int[dataSignal.length *
spreadingCode.length];
        for (int i = 0; i < dataSignal.length; i++) {
            for (int j = 0; j < spreadingCode.length; j++) {
                spreadSignal[i * spreadingCode.length + j] = dataSignal[i] ^
spreadingCode[j];
            }
        }
        return spreadSignal;
    }
```

```
    private static int[] despreadDSSS(int[] spreadSignal, int[]
spreadingCode) {
        int length = spreadSignal.length / spreadingCode.length;
        int[] recoveredSignal = new int[length];


        for (int i = 0; i < length; i++) {
            int sum = 0;
            for (int j = 0; j < spreadingCode.length; j++) {
                sum += spreadSignal[i * spreadingCode.length + j] ^
spreadingCode[j];
            }
            System.out.print("Addition of " + " bit " + (i + 1) + " : " + sum);
            recoveredSignal[i] = (sum > 7) ? 1 : 0;
            if (sum > 7) {
                System.out.println(", Since sum is more than 7, it is converted to
1");
            } else
                System.out.println(", Since sum is less than 4, it is converted to
0");
        }


        return recoveredSignal;
    }
}
```

Output:

```
(base) PS C:\Users\Rishab\OneDrive\Desktop\MCC Exp Documents> cd "c:\Users\Rishab\OneDrive\Desktop\MCC Exp Documents\" ; if ($?) { javac Exp9DSSS.java } ; if ($?) { java Exp9DSSS }
Original Data Signal: [0, 1]
Spreading Code (PN Sequence): [1, 0, 1, 1, 0, 1, 1, 1, 0, 0, 0]
Spread Signal: [1, 0, 1, 1, 0, 1, 1, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1, 1, 1]
Addition of  bit 1 : 0, Since sum is less than 4, it is converted to 0
Addition of  bit 2 : 11, Since sum is more than 7, it is converted to 1
Recovered Signal: [0, 1]
(base) PS C:\Users\Rishab\OneDrive\Desktop\MCC Exp Documents>
```

THADOMAL SHAHANI TSEC ENGINEERING COLLEGE

EXPERIMENT NO. 10

Aim :- Write a program to implement the A3/A5/A8 GSM security algorithm.

Theory :-

Security Algorithm :

A security algorithm refers to a cryptographic method used to ensure the confidentiality and integrity of data transmitted over the GSM network.
The GSM network employs various security algorithms to protect communication between mobile devices and the n/w infrastructure.

GSM uses three different security algorithms called A3, A5 and A8. In practice, A3 and A8 are generally implemented together (known as A3/A8).

An A3/A8 algorithm is implemented in Subscriber Identity Module (SIM) cards and in GSM n/w Authentication Centres. It is used to authenticate the customer and generate a key for encrypting voice and data traffic. These protocols ensure that only authorized users can access the network and communicate securely.

- **A3 algorithm (Authentication):**

→ The A3 algorithm is responsible for the authentication of user's identity during the establishment of a connection with the GSM network.
It calculates the SRES (Signed Resource) based on the secret key K_i (stored on SIM card and in HLR and the Random challenge RAND sent by MSC).
SRES is then compared with expected SRES stored in the n/w's databases to verify the user's identity.
The algorithm is not standardized, meaning each GSM operator can choose its own implementation.

- **A8 algorithm (Key Generation):**

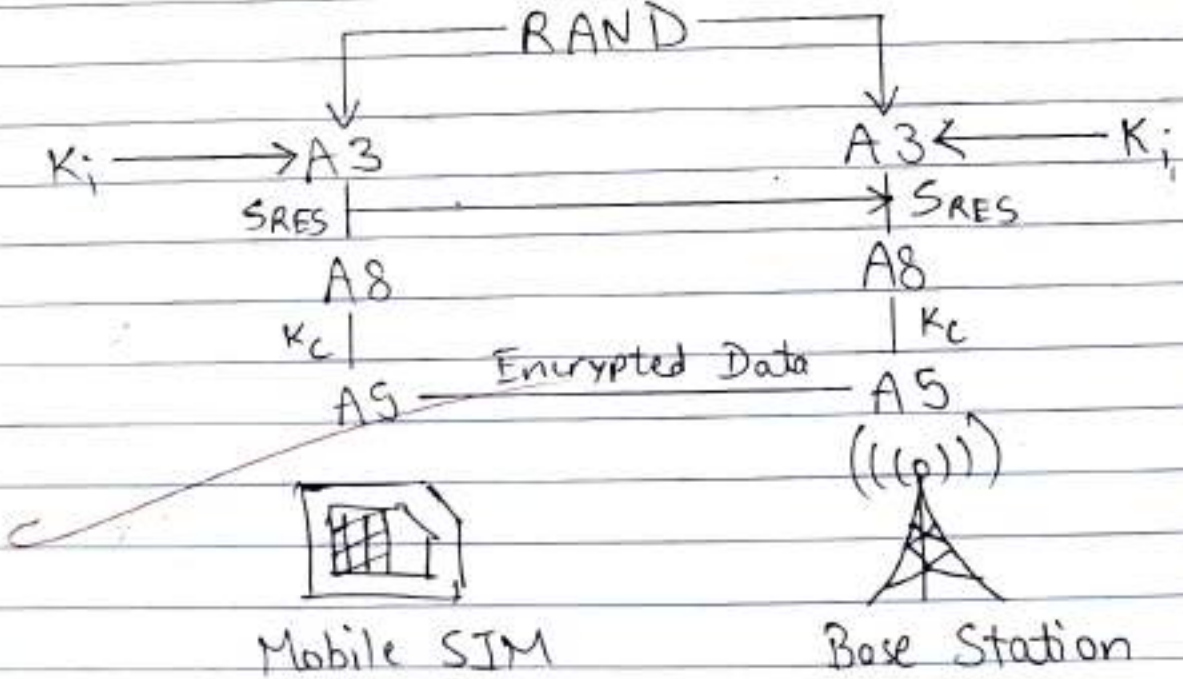
→ This algorithm is used to generate a session key K_c (cipher key) needed for encrypting the voice and data traffic between the mobile device and GSM network. It calculates the session key K_i and the random challenge RAND.
Session key K_c is unique for the each connection and ensures that communication b/w the mobile device and the n/w is fully secured.

A5 algorithm (Encryption)

→ A5 algorithm is used for encrypting the user's voice and data traffic over the air interface between the mobile handset and the base station subsystem (BSS) to ensure privacy and confidentiality.

- It operates as a stream cipher, where each bit of the data stream is encrypted very independently based on the session key K_c and the frame number.

A5 encryption is specified at international level to enable interoperability and roaming between different GSM networks.



- Flow Chart :

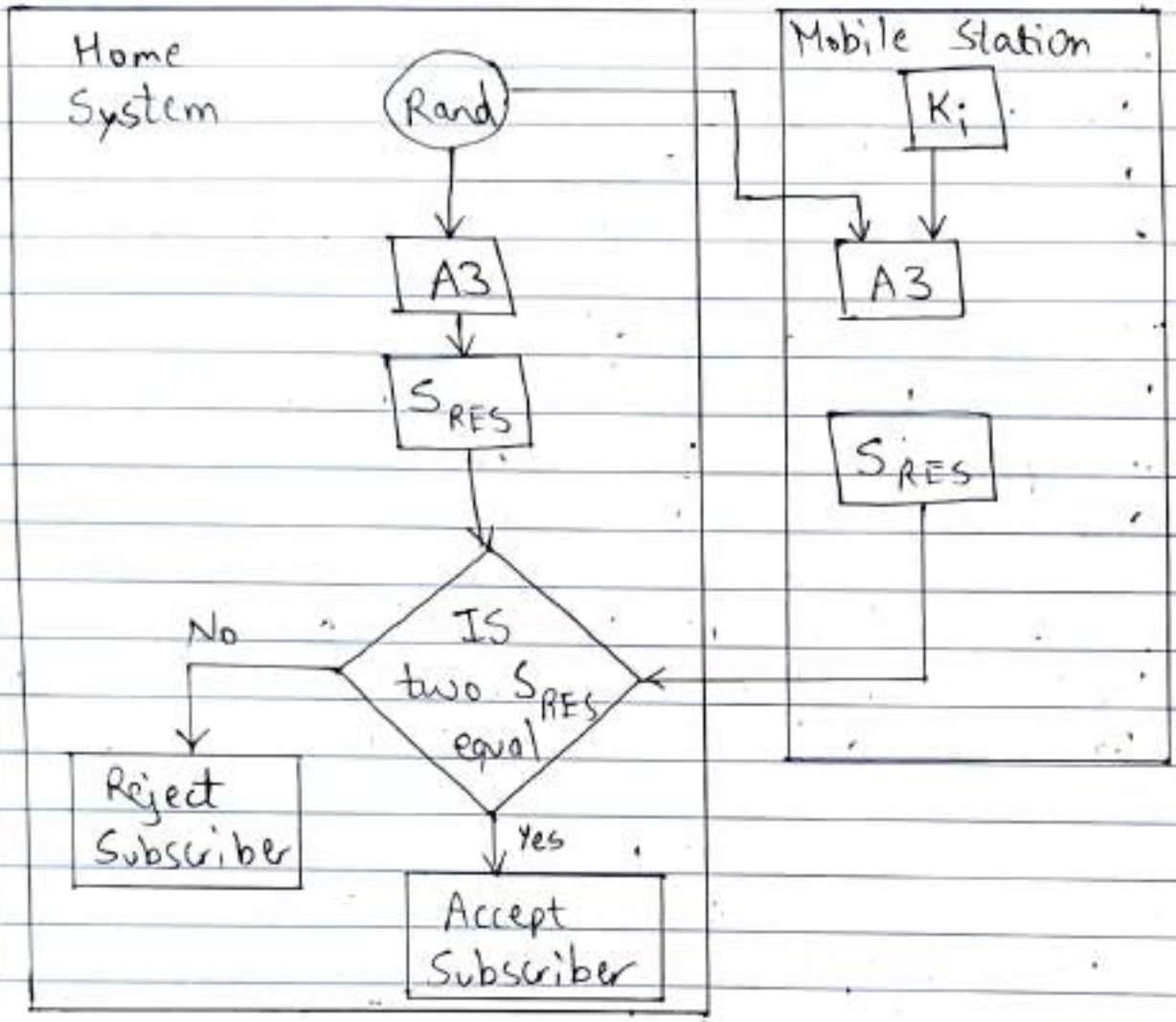


Fig. Authentication in GSM

Conclusion :-

Thus, implemented the GSM security algorithms A3 / A5 / A8.

Exp No. 10

Rishab Mandal

Batch: C23

Code:

Exp10A3Algo.java :

```
import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;
import java.util.Random;

public class Exp10A3Algo {

    public static void main(String[] args) {
        // Simulate generating a random Ki (secret key) and RAND (challenge)
        String ki = generateRandomHexString(32); // 128 bits (16 bytes) key
        String rand = generateRandomHexString(32); // 128 bits (16 bytes) challenge

        // Display the generated Ki and RAND
        System.out.println("Ki (Secret Key): " + ki);
        System.out.println("RAND (Challenge): " + rand);
```

```
    // Calculate the expected response (SRES) using the A3 algorithm
    String sres = calculateSRES(ki, rand);

    // Display the calculated SRES
    System.out.println("SRES (Expected Response): " + sres);
}

private static String generateRandomHexString(int length) {
    Random random = new Random();
    StringBuilder randomHex = new StringBuilder();

    for (int i = 0; i < length; i++) {
        int randomInt = random.nextInt(16); // 0-15
        randomHex.append(Integer.toHexString(randomInt));
    }

    return randomHex.toString();
}

private static String calculateSRES(String ki, String rand) {
    try {
        // Concatenate Ki and RAND
        String input = ki + rand;

        // Use SHA-1 hash function to calculate SRES
```

```
        MessageDigest sha1 = MessageDigest.getInstance("SHA-1");
        byte[] hashBytes = sha1.digest(hexStringToByteArray(input));

        // Convert the hash to a hexadecimal string
        StringBuilder sres = new StringBuilder();
        for (byte b : hashBytes) {
            sres.append(String.format("%02X", b));
        }

        return sres.toString();
    } catch (NoSuchAlgorithmException e) {
        e.printStackTrace();
        return null;
    }
}

private static byte[] hexStringToByteArray(String hexString) {
    int len = hexString.length();
    byte[] data = new byte[len / 2];

    for (int i = 0; i < len; i += 2) {
        data[i / 2] = (byte) ((Character.digit(hexString.charAt(i), 16) << 4)
                + Character.digit(hexString.charAt(i + 1), 16));
    }
```

```
        return data;
    }
}
```

Output:

```
(base) PS C:\Users\Rishab\OneDrive\Desktop\MCC Exp Documents> cd "c:\Users\Rishab\OneDrive\Desktop\MCC Exp Documents\" ; if ($?) { javac Exp10A3Algo.java } ; if ($?) { java Exp10A3Algo }
Ki (Secret Key): 75b3cd449c491cb7af27683de9dba3f8
RAND (Challenge): 5ed4d5250e87177fd4c4f4e9f9238cf4
SRES (Expected Response): 8A69046163903D54366D9AF1E410B40E56872AC6
```

Exp10A5Algo.java :

```
import java.lang.Math;

public class A5 {
    static int[] GenerateBits() {
        int[] a = new int[16];
        for (int i = 0; i < 16; i++) {
            double rand = Math.random();
            if (rand >= 0.5) {
                a[i] = 1;
            } else {
                a[i] = 0;
```

```
        }
    }
    return a;
}

static int[] XOR(int[] a, int[] b) {
    int[] temp = new int[16];
    for (int i = 0; i < 16; i++) {
        if (a[i] == 1 && b[i] == 1 || a[i] == 0 && b[i] == 0) {
            temp[i] = 0;
        } else {
            temp[i] = 1;
        }
        System.out.print(temp[i]);
    }
    return temp;
}

static int[] AND(int[] a, int[] b) {
    int[] temp = new int[16];
    for (int i = 0; i < 16; i++) {
        if (a[i] == 1 && b[i] == 1) {
            temp[i] = 1;
        } else {
            temp[i] = 0;
        }
        System.out.print(temp[i]);
    }
    return temp;
}
```

```
public static void main(String[] args) {
    int[] a;
    System.out.println("Generating the 1st key identification number");
    a = GenerateBits();
    for (int i = 0; i < 16; i++) {
        System.out.print(a[i]);
    }
    int[] b;
    System.out.println("\n\nGenerating the 2nd key identification number");
    b = GenerateBits();
    for (int i = 0; i < 16; i++) {
        System.out.print(b[i]);
    }
    int[] c;
    System.out.println("\n\nGenerating the random number");
    c = GenerateBits();
    for (int i = 0; i < 16; i++) {
        System.out.print(c[i]);
    }
    int[] d;
    System.out.println("\n\nGenerating the barker code");
    d = GenerateBits();
    for (int i = 0; i < 16; i++) {
        System.out.print(d[i]);
    }
    int[] z;
    System.out.println("\n\nAND of 1st key and 2nd key");
    z = AND(a, b);
    int[] p;
```

```
        System.out.println("\n\nXOR of random number and the AND of 1st key and 2nd key");
        p = XOR(z, c);
        System.out.println("\n\nXOR of the above number and barker code");
        p = XOR(p, d);
        int[] q;
        System.out.println("\n\nXOR of random number and the AND of 1st key and 2nd key");
        q = XOR(z, c);
        System.out.println("\n\nXOR of the above number and barker code");
        q = XOR(q, d);
        int flag = 0;
        for (int i = 0; i < 16; i++) {
            if (p[i] != q[i]) {
                flag = 1;
                break;
            }
        }
        if (flag == 1) {
            System.out.print("\n\nEncryption Failed");
        } else {
            System.out.print("\n\nEncryption Passed");
        }


    }
}
```

Output:

```
(base) PS C:\Users\Rishab\OneDrive\Desktop\MCC Exp Documents> cd "c:\Users\Rishab\OneDrive\Desktop\MCC Exp Documents\" ; if ($?) { javac Exp10A5Algo.java } ; if ($?) { java Exp10A5Algo }
```

Generating the 1st key identification number

0111001001011110

Generating the 2nd key identification number

1111110111000001

Generating the random number

1011000011011100

Generating the barker code

0001011100001111

AND of 1st key and 2nd key

0111000001000000

XOR of random number and the AND of 1st key and 2nd key

1100000010011100

XOR of the above number and barker code

1101011110010011

XOR of random number and the AND of 1st key and 2nd key

1100000010011100

XOR of the above number and barker code

1101011110010011

Encryption Passed

Keystream bit = 0 ^ 0 ^ 0 = 0

Keystream bit = 0 ^ 0 ^ 0 = 0

Keystream bit = 1 ^ 1 ^ 0 = 0

Keystream bit = 0 ^ 1 ^ 1 = 0

Keystream bit = 1 ^ 1 ^ 1 = 1

Keystream bit = 0 ^ 0 ^ 1 = 1

Exp10A8Algo.java :

```
public class A8Algorithm {
    // Secret key (Ki)
    private static final int[] KI = { 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1 };
```

```
    // A8 Algorithm
    public static int[] generateKeyStream(int[] rand) {
        int[] keyStream = new int[rand.length];

        // Generate key stream
        for (int i = 0; i < rand.length; i++) {
            keyStream[i] = rand[i] ^ KI[i % KI.length];
        }

        return keyStream;
    }

    // Example usage
    public static void main(String[] args) {
        // Example random number (RAND)
        int[] rand = { 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0 };

        // Generate key stream using A8 algorithm
        int[] keyStream = generateKeyStream(rand);

        // Print key stream
        System.out.println("Key Stream:");
        for (int keyBit : keyStream) {
            System.out.print(keyBit);
        }
    }
}
```

Output:

(base) PS C:\Users\Rishab\OneDrive\Desktop\MCC Exp Documents> cd "c:\Users\Rishab\OneDrive\Desktop\MCC Exp Documents\" ; if ($?) { javac Exp10A8Algo.java } ; if ($?) { java Exp10A8Algo }

Step 1: RAND = 1, KI = 0, Key Bit = RAND ^ KI = 1 ^ 0 = 1

Step 2: RAND = 0, KI = 1, Key Bit = RAND ^ KI = 0 ^ 1 = 1

Step 3: RAND = 1, KI = 0, Key Bit = RAND ^ KI = 1 ^ 0 = 1

Step 4: RAND = 0, KI = 1, Key Bit = RAND ^ KI = 0 ^ 1 = 1

Step 5: RAND = 1, KI = 0, Key Bit = RAND ^ KI = 1 ^ 0 = 1

Step 6: RAND = 0, KI = 1, Key Bit = RAND ^ KI = 0 ^ 1 = 1

Step 7: RAND = 1, KI = 0, Key Bit = RAND ^ KI = 1 ^ 0 = 1

Step 8: RAND = 0, KI = 1, Key Bit = RAND ^ KI = 0 ^ 1 = 1

Step 9: RAND = 1, KI = 0, Key Bit = RAND ^ KI = 1 ^ 0 = 1

Step 10: RAND = 0, KI = 1, Key Bit = RAND ^ KI = 0 ^ 1 = 1

Step 11: RAND = 1, KI = 0, Key Bit = RAND ^ KI = 1 ^ 0 = 1

Step 12: RAND = 0, KI = 1, Key Bit = RAND ^ KI = 0 ^ 1 = 1

Step 13: RAND = 1, KI = 0, Key Bit = RAND ^ KI = 1 ^ 0 = 1

Step 14: RAND = 0, KI = 1, Key Bit = RAND ^ KI = 0 ^ 1 = 1

Step 15: RAND = 1, KI = 0, Key Bit = RAND ^ KI = 1 ^ 0 = 1

Step 16: RAND = 0, KI = 1, Key Bit = RAND ^ KI = 0 ^ 1 = 1

Key Stream:

1111111111111111

THADOMAL SHAHANI TSEC ENGINEERING COLLEGE

ASSIGNMENT NO. 1

Q.1 Explain in detail with merits and demerits.

a) Snooping TCP

Snooping TCP refers to a type of TCP flow control mechanism that monitors the state of each connection in a network and adjusts the flow of data accordingly. This helps to improve network performance by avoiding congestions and ensuring that each connection gets the necessary bandwidth to function optimally. In snooping TCP, the network switches or routers are responsible for the monitoring and controlling the flow of data in real-time, making it a more efficient way of managing network traffic compared to a traditional TCP flow control mechanism.

Working of Snooping TCP:

Until it receives an acknowledgement from the mobile node, the foreign agent buffers the packet. A foreign agent snoops the packet flow and acknowledgement in both directions. If the foreign agent snoops acknowledged packet from the mobile node, or if it receives duplicate acknowledgements, it believes that the packet or acknowledgement

has been lost. The packet is immediately retransmitted by foreign agent from its buffer. In addition, the foreign agent maintains its own timer for retransmission of buffered packets in case it is lost on the wireless link. If the foreign agent detects a missing packet during data transfer from the mobile node to the correspondent node, it sends a NAck to the mobile node. The mobile node can then promptly retransmit the missing packet. Packet reordering is handled automatically by TCP. Even if the foreign agent crashes, the timeout correspondent node still triggers retransmission.

Merits :

- End to end TCP semantic is preserved.
- No modifications at fixed host, i.e, the fixed computer TCP does not need any type of changes.
- No packet loss during handovers.

Demerits :

- The behavior of the wireless link:

- Snooping TCP does not isolate the behaviour of the wireless link or I-TCP. Transmission errors can spread to the correspondent nodes (CH).

- A mobile node needs additional mechanisms: The use of NAck between foreign agent and the node requires the mobile node to have additional mechanisms.

- Encryption at end-to-end: If such method is used, then snooping and buffering data can be considered worthless.

b) Mobile TCP:

The M-TCP approach has the same goal as I-TCP and Snooping TCP; To prevent the sender window from shrinking if bit errors or disconnection but not congestion cause current problems. M-TCP wants to improve overall throughput, lower the delay, maintain end to end semantics of TCP, and to provide a more efficient handover. The M-TCP wants to assume a low bit error rate on the wireless link and refrains from caching/ retransmitting data via a supplementary

entity called SH (Switching Hub). If a packet is lost on the wireless link, the original sender must retransmit it to maintain TCP end-to-end semantics. The SH monitors packets sent to and Acks returned from the Mobile Host (MH). If no Ack is received within a certain time, the SH assumes MH disconnection, sets the sender window size to 0, forcing it into persistent mode.

Merits:

- Faster Packet Loss Recovery; thus enhancing overall reliability and performance.
- End-to-end semantics; preserves data integrity, by having the sender retransmit lost packets.
- Automatic Disconnection Handling; minimizes delays in response to network changes.
- No sender TCP modifications; simplifies modifications, implementation.

Demerits:

- Dependency on low bit error rate; thereby limiting effectiveness in environments with high error rates.

- No caching / retransmission via SH.

c) I-TCP (Indirect TCP):

It is a protocol designed for mobile computing environments to address the challenges of handovers and disconnections. Instead of modifying the end systems, I-TCP employs a Mobile host agent (MHA) to intercept and manage TCP connections on behalf of the mobile host.

Merits:

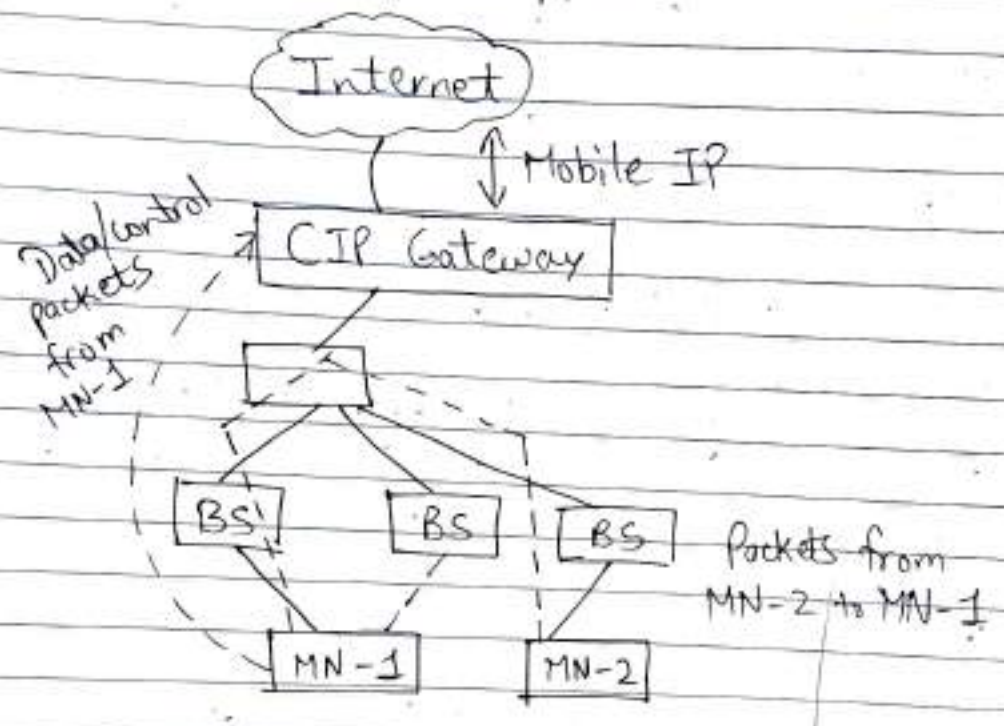
- Seamless Mobility; ensuring continuity without disruption.
- No modifications to end systems; minimizes the need for changes to existing implementations.
- Maintains connection integrity.

Demerits:

- Complexity of Mobile host agents.
- Potential latency; during handovers, some latency can occur.
- Compatibility issues could limit its adoption in heterogeneous network environments.

Q.2 Explain Cellular (Mobile) IP in detail.

Mobile IP is a communication protocol (created by extending Internet Protocol, IP that allows the users to move from on network to another with the same IP addr It ensures that the communication will continue without the user's sessions or connections being dropped.



Cellular IP address shortcomings in Mobile IP, particularly concerning handover duration and registeration scalability. Unlike Mobile IP, Cellular IP facilitates fast and consistent local handovers within limited geographical

areas. It deploys a cellular IP gateway (CIP) in each domain, acting as a foreign agent for the external network. Nodes within the domain collect routing information based on packet origins towards the CIPGW, enabling efficient handover control. Soft handovers, achieved through simultaneous packet flow forwarding along multiple paths, allow smooth transitions for mobile nodes moving between adjacent cells. The self-configuring and straight forward architecture of Cellular IP addresses these challenges, but it requires modifications to the basic Mobile IP protocol and is not transparent to existing systems.

Q.3 Explain MIPv6.

Mobile IPv6 enables a device to keep the same internet address worldwide, as it provides support for IPv6, allowing seamless connectivity while changing locations. It allows mobility across heterogenous and homogenous media. Each mobile device has two IP addresses; a permanent home address and a temporary care of address that changes with location. When moving to a new location, the device acquires a care-of address using the IPv6 neighbourhood discovery methods. The home

agent, located in the home network, plays a key role by associating home and care -of addresses. It intercepts and forwards packets, ensuring continuous connectivity. The mobile node informs correspondent nodes of its care-of address through binding updates. Key functions include maintaining bindings caches, supporting Dynamic Home agent Address dislovery, and aiding movement detection in visited networks. Basically, a fundamental function of MIPv6 is to provide a mechanism for mobile devices to move across different domains/ networks while maintaining a consistent and unchanging home address.

Q.4 Write short notes on HAWAII.

HAWAII stands for Handoff-Aware Wireless Access Internet Infrastructure. It is an extension protocol of Mobile IP designed to enhance the support of micro-mobility and paging functionality in wireless network.

It addresses the challenges associated with micro-mobility, which involves movement of a mobile node within a small geographical area. It provides mechanisms to efficiently

handle handovers in a small scale environment. It also enhances paging functionality, allowing the network to effectively locate a mobile node when needed, minimizing delays and optimizing resource usage.

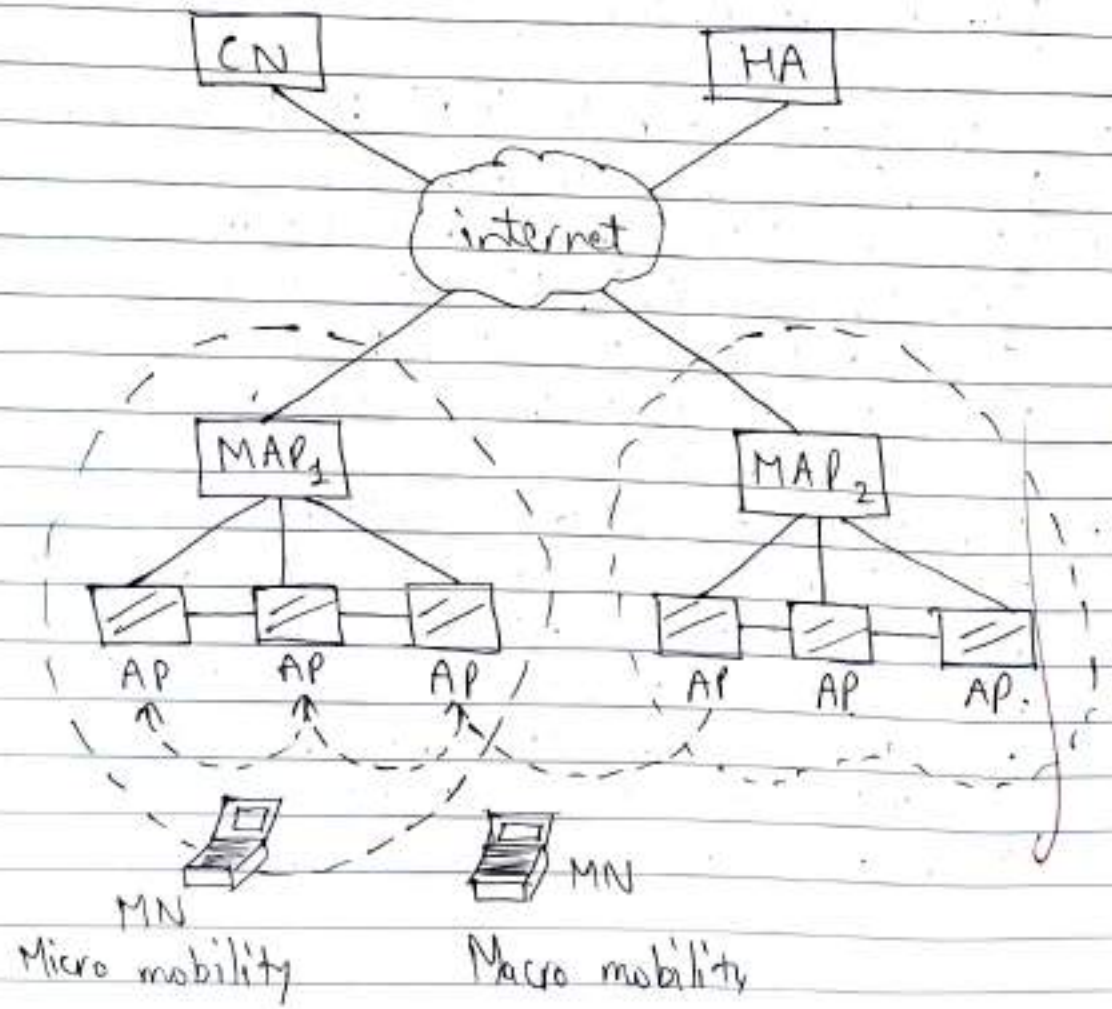
It also seamlessly integrates with existing Mobile IP infrastructure, thereby reducing the latency associated with locating and establishing connections with mobile nodes.

Working of HAWAII:

- When a mobile node (MN) is in a home domain, HAWAII is not involved.
- When an MN moves to a HAWAII domain (foreign domain), it obtains a co-located COA, and registers with HA.
- This COA remains unchanged as long as MN is in foreign domain.
- MN sends registeration request to new BS.
- The BS intercepts and sends out hand-off update message, which reconfigures all routers, on the path from the old and new BS to the so-called 'crossover router.
- The BS then sends a registeration reply to MN as if it were the FA.

Q.5 Write short notes on HMIPv6.

Hierarchical Mobile IPv6 (HMIPv6) is an extension of MIPV6 designed to enhance seamless mobility in wireless networks. It introduces a hierarchical structure with multiple domains, each having a home agent (HA) and a Mobility Anchor Point (MAP). This structure reduces signaling overhead, improves scalability, and minimizes the handover latency.



HMIPv6 includes a Mobile Node (MN), a correspondent Node (CN), Home agent (HA), MN moves across domains, maintaining communication, while HA and MAP play key roles in managing home and current location information.

When MN moves to a new domain, handover process involves detecting the move, exchanging router solicitation, updating binding tables, and ensuring the seamless communication with the correspondent Node using the care-of Adress (CoA).

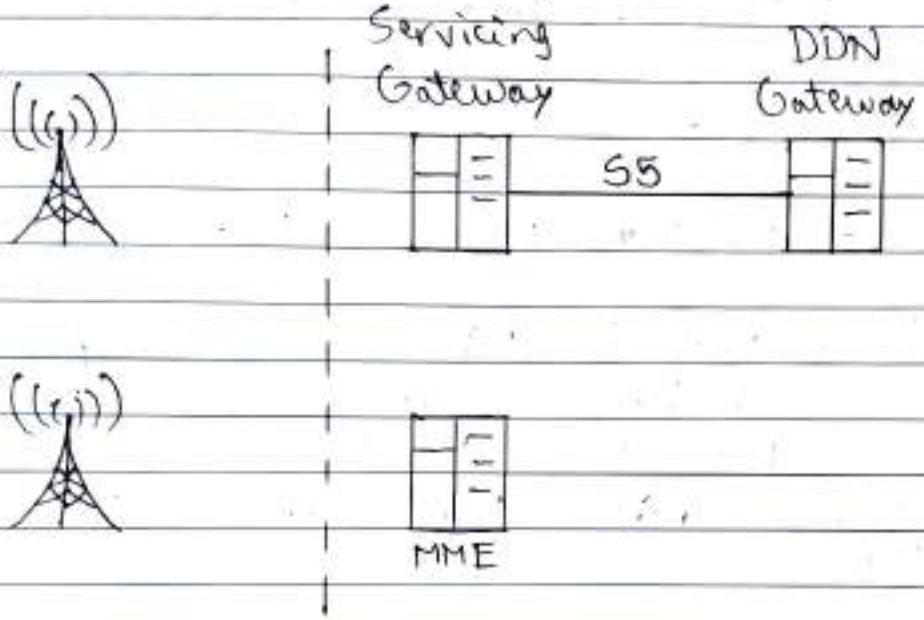
A+

THADOMAL SHAHANI TSEC ENGINEERING COLLEGE

ASSIGNMENT NO. 2

Q.1 Draw and explain SAE Architecture.

System Architecture Evolution



System Architecture Evolution (SAE) is a new network architecture designed to simplify LTE networks and establish a flat architecture similar to the other IP, based communications networks. SAE uses an eNB and Access Gateway (aGW) and removes the RNC and SGSN from the equivalent 3G network architecture to create a simpler mobile network.

This allows the network to be built with an "All-IP" based n/w architecture. SAE also includes entities to allow full inter-working with other related wireless (WCDMA, WiMAX, WLAN, etc)

Key components:

- MME (Mobility Management Entity)

It controls idle mode UE tracking, paging procedures, and bearer activation. It also manages intra-LTE handovers and user authentication with the Home Subscriber Server (HSS).

- SGW (Serving Gateway)

Routes and forwards user data packets, facilitates inter-eNB handovers, and supports mobility b/w LTE and other N/ws. It also terminates DL data from idle state UEs, triggering UE paging.

- PGW (PDN Gateway)

It connects UEs to external n/ws, serving as entry point for data traffic, thereby enabling UEs to access multiple PDNs.

Q.2 Describe evolution from UMTS to LTE.

LTE evolved from an earlier 3GPP system known as the Universal Mobile Telecomm. System (UMTS), which in turn evolved from the Global system for the Mobile Communications.

There was no global standard for wireless broadband until the advent of LTE. The goal with LTE was to merge a fragmented market and offer a more efficient n/w for n/w operators.

The evolution from UMTS to LTE was driven by various factors. There was a need to ensure the ongoing competitiveness of 3G systems, responding to the user demands for the higher data rates and also improved quality of service. LTE addressed these demands by offering upto 100 Mbps of downstream and 30 Mbps upstream, surpassing the capabilities of 3G. Also, there was a continuous demand for cost reduction (CAPEX and OPEX), and low complexity. Also, LTE provides reduced latency, scalable bandwidth capacity and back-ward compatibility with GSM and UMTS.

LTE successfully avoided the unnecessary fragmentation in technologies for the both paired and unpaired band operations.

Q.3 Compare Mobile Generations (1G, 2G, 3G, 4G, 5G)

	1G	2G	3G	4G	5G
Require-ments	No official req. Analog technology	No official req. Digital technology	ITU's IMT 2000 req. 144 kbps mobile, 384 kbps pedes-trian, 2 Mbps indoor.	ITU's IMT Advanced req. operates upto 40 Mhz radio channels.	Atleast 1 GB/s or more data rates to support high definition videos.
Data bandwidth	1.9 kbps	14.4 to 384 kbps	2 Mbps	2 Mbps to 1 Gbps	1 Gbps and higher
Core N/W	PSTN	PSTN packet network	Packet network	All IP network	Flatter IP N/W and 5G NW-I (interface)
Frequency	800-900 MHz	850 & 1900 MHz	1.6 - 2.5 GHz	2 - 8 GHz	3 - 90 GHz

3.4 What are self organizing networks?

Self Organizing Networks (SONs) are radio access networks (RANs) that automatically plan, configure, manage, optimize, and heal themselves. SONs can offer automated functions such as the self-configurations, self-optimization, self-healing, and self-protection.

SONs strive to make complicated n/w administration a thing of the past by enabling the creation of a plug-and-play environment for both simple and complex network tasks. This is in stark constrast to the traditional implementation of cellular wireless networks we see in enterprises today, most of which require teams of technicians for maintenance, management, and optimization.

SON can offer a variety of the different functions, including self-configuration, etc. These networks utilize real-time information, such as traffic load, interference and user mobility, to dynamically adjust settings like power levels, handovers, and frequency allocations.

It enhances N/w performance, reduces operational costs, and improves user experience by continuously adapting to changing conditions.

Q.5 Explain VoLTE in detail.

VoLTE stands for Voice over Long-Term Evolution or Voice over LTE.
VoLTE offers the possibility to voice call via the LTE / 4G+ mobile network.

Benefits of VoLTE are as follows:

- Fast call set-up
- High voice quality and reduced background noise.
- The phone remains in the 4G / LTE network during voice calls.
- You can use 4G / LTE data services, such as web surfing and tethering, while simultaneously making and receiving calls.
- When calling over LTE, you don't use extra data but call minutes. You pay only for the voice call service, not the data you use.