SRI MUTHUKUMARAN INSTITUTE OF TECHNOLOGY

Approved by AICTE, New Delhi &Affiliated to ANNA UNIVERSITY ,Chennai.

CHIKKARAYAPURAM, (NEAR MANGADU) CHENNAI - 600069

NAME : _____

REG. NO : _____

BRANCH : _____

YEAR : _____

SEMESTER : _____

SRI MUTHUKUMARAN INSTITUTE OF TECHNOLOGY

Approved by AICTE, New Delhi & Affiliated to Anna University, Chennai.

CHIKKARAYAPURAM, (NEAR MANGADU) CHENNAI - 600 069

Register No. | | | | | | | | | | | |

Bonafide Certificate

This is to Certify that this is the bonafide record of the work done by

Selvan/Selvi _____ of

_____ year MCA (_____)

in the _____ Laboratory

during the Academic Year _____

Staff - In - Charge **Head of the Department**

Submitted for the University Practical Examination held on _____ .

Internal Examiner **External Examiner**

INDEX

EX NO: 1 **Implement mobile applications using UI toolkits and frameworks**

DATE:

AIM:

To Implement mobile applications using UI toolkits and frameworks

PROCEDURE:

Step 1: Set up the Development Environment
- Install Android Studio, the official IDE for Android app development.
- Configure the necessary Android SDK and emulators.

Step 2: Create a New Project
- Open Android Studio and select "Start a new Android Studio project" or go to File -> New -> New Project.
- Configure your project settings, including the application name, package name, and project location.

Step 3: Design the User Interface (UI)
- Open the layout file associated with the main activity. By default, it is called "activity_main.xml."
- Design your UI using XML markup, leveraging the available UI components, layouts, and styles.
- Here's an example using LinearLayout and TextView:

Step 4: Implement the UI Toolkits in Java Code
- Open the Java file associated with the main activity. By default, it is called "MainActivity.java."
- Inside the `onCreate` method, set the layout using `setContentView(R.layout.activity_main)`.
- Access the UI elements defined in the layout using `findViewById`.
- You can add additional code logic as needed.

Step 5: Build and Run the Application
- Connect a physical Android device or use an emulator to test your application.
- Click the "Run" button in Android Studio, and the application will be installed and launched on the connected device/emulator.

PROGRAMS:-

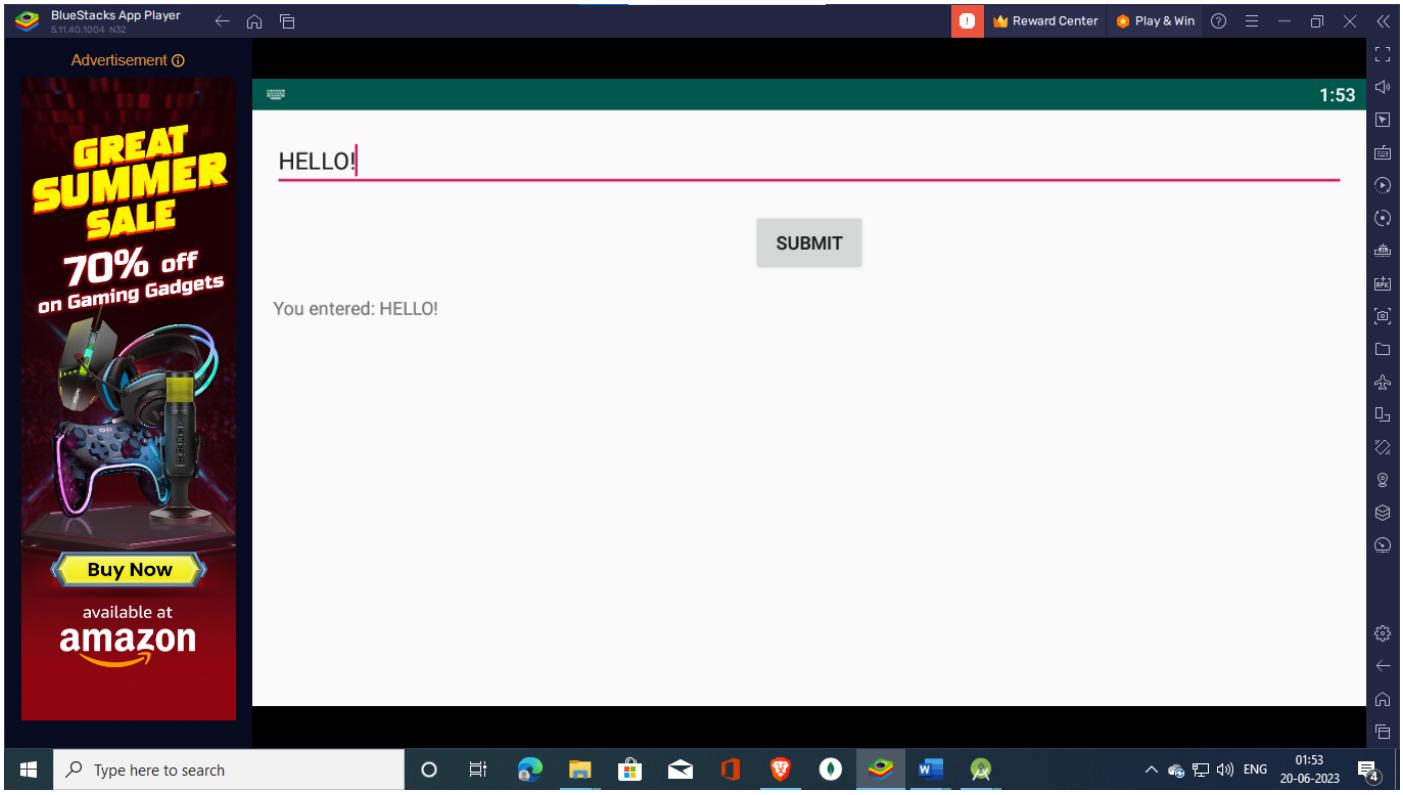
Code for activity_main.xml:-

```
<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:paddingLeft="16dp"
    android:paddingTop="16dp"
    android:paddingRight="16dp"
    android:paddingBottom="16dp"
    tools:context=".MainActivity">

    <EditText
        android:id="@+id/inputEditText"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="Enter text" />

    <Button
        android:id="@+id/submitButton"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@+id/inputEditText"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="16dp"
        android:text="Submit" />
    <TextView
        android:id="@+id/displayTextView"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_below="@+id/submitButton"
        android:layout_marginTop="16dp"
        android:text="Text will be displayed here" />
</RelativeLayout>
```

Code for main activity_java:-

```
package com.example.myapplication;

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

import com.example.myapplication.R;

public class MainActivity extends AppCompatActivity {

    private EditText inputEditText;
    private Button submitButton;
    private TextView displayTextView;

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

        // Initialize UI elements
        inputEditText = findViewById(R.id.inputEditText);
        submitButton = findViewById(R.id.submitButton);
        displayTextView = findViewById(R.id.displayTextView);

        // Set click listener for the submit button
        submitButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                String inputText = inputEditText.getText().toString();
                displayTextView.setText("You entered: " + inputText);
            }
        });
    }
}
```

OUTPUT: -


BlueStacks App Player
Reward Center
Play & Win
Advertisement
GREAT SUMMER SALE
70% off
on Gaming Gadgets
Buy Now
available at
amazon
1:53
HELLO!
SUBMIT
You entered: HELLO!
Type here to search
ENG
01:53
20-06-2023

Result:-

Thus the application that makes use of database has been Developed and the output was verified.

EX NO: 2 **Design an application that uses Layout Managers and event listeners.**

DATE:

AIM:

To Design an application that uses Layout Managers and event listeners.

PROCEDURE:

Step 1: Set up the Development Environment
- Install Android Studio, the official IDE for Android app development.
- Configure the necessary Android SDK and emulators.

Step 2: Create a New Project
- Open Android Studio and select "Start a new Android Studio project" or go to File -> New -> New Project.
- Configure your project settings, including the application name, package name, and project location.

Step 3: Design the User Interface (UI)
- Open the layout file associated with the main activity. By default, it is called "activity_main.xml."
- Design your UI using XML markup, leveraging the available UI components, layouts, and styles.
- Here's an example using LinearLayout and TextView:

Step 4: Implement the UI Toolkits in Java Code
- Open the Java file associated with the main activity. By default, it is called "MainActivity.java."
- Inside the `onCreate` method, set the layout using `setContentView(R.layout.activity_main)`.
- Access the UI elements defined in the layout using `findViewById`.
- You can add additional code logic as needed.

Step 5: Build and Run the Application
- Connect a physical Android device or use an emulator to test your application.
- Click the "Run" button in Android Studio, and the application will be installed and launched on the connected device/emulator.

PROGRAMS:-

Code for 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"
    tools:context=".MainActivity">
    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="100dp">
        <TextView
            android:id="@+id/textView"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_margin="30dp"
            android:text="Details Form"
            android:textSize="25sp"
            android:gravity="center"/>
    </LinearLayout>
    <GridLayout
        android:id="@+id/gridLayout"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_marginTop="100dp"
        android:layout_marginBottom="200dp"
        android:columnCount="2"
        android:rowCount="3">

        <TextView
            android:id="@+id/textView1"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_row="0"
            android:layout_column="0"
            android:layout_margin="10dp"
            android:gravity="center"
            android:text="Name"
            android:textSize="20sp" />
        <EditText
```

```
        android:id="@+id/editText"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_margin="10dp"
        android:layout_row="0"
        android:layout_column="1"
        android:ems="10"/>
    <TextView
        android:id="@+id/textView2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_margin="10dp"
        android:layout_row="1"
        android:layout_column="0"
        android:text="Reg.No"
        android:textSize="20sp"
        android:gravity="center"/>
    <EditText
        android:id="@+id/editText2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_margin="10dp"
        android:layout_row="1"
        android:layout_column="1"
        android:inputType="number"
        android:ems="10"/>
    <TextView
        android:id="@+id/textView3"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_margin="10dp"
        android:layout_row="2"
        android:layout_column="0"
        android:text="Dept"
        android:textSize="20sp"
        android:gravity="center"/>
    <Spinner
        android:id="@+id/spinner"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_margin="10dp"
        android:layout_row="2"
        android:layout_column="1"
        android:spinnerMode="dropdown"/>
</GridLayout>
```

```
        <Button
            android:id="@+id/button"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_alignParentBottom="true"
            android:layout_centerInParent="true"
            android:layout_marginBottom="150dp"
            android:text="Submit"/>
</RelativeLayout>
```

Code for Second_activity.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"
            tools:context="com.example.myapplication.SecondActivity"
            android:orientation="vertical"
            android:gravity="center">
            <TextView
                android:id="@+id/textView1"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_margin="20dp"
                android:text="New Text"
                android:textSize="30sp"/>
            <TextView
                android:id="@+id/textView2"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_margin="20dp"
                android:text="New Text"
                android:textSize="30sp"/>
            <TextView
                android:id="@+id/textView3"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_margin="20dp"
                android:text="New Text"
                android:textSize="30sp"/>
</LinearLayout>
```

Code for main activity_java:-

```
package com.example.myapplication;
import android.content.Intent;
//import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.ArrayAdapter;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Spinner;

import androidx.appcompat.app.AppCompatActivity;

public class MainActivity extends AppCompatActivity {
    //Defining the Views
    EditText e1,e2;
    Button bt;
    Spinner s;
    //Data for populating in Spinner
    String [] dept_array={"CSE","ECE","IT","Mech","Civil"};
    String name,reg,dept;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        //Referring the Views
        e1= (EditText) findViewById(R.id.editText);
        e2= (EditText) findViewById(R.id.editText2);
        bt= (Button) findViewById(R.id.button);
        s= (Spinner) findViewById(R.id.spinner);
        //Creating Adapter for Spinner for adapting the data from array to Spinner
        ArrayAdapter adapter= new
                ArrayAdapter(MainActivity.this,android.R.layout.simple_spinner_item,dept_array);
        s.setAdapter(adapter);
        //Creating Listener for Button
        bt.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                //Getting the Values from Views(Edittext & Spinner)
                name=e1.getText().toString();
                reg=e2.getText().toString();
                dept=s.getSelectedItem().toString();
```

```
                //Intent For Navigating to Second Activity
                Intent i = new Intent(MainActivity.this,SecondActivity.class);
                //For Passing the Values to Second Activity
                i.putExtra("name_key", name);
                i.putExtra("reg_key",reg);
                i.putExtra("dept_key", dept);
                startActivity(i);
            }
        });
    }
}
```

Code for Second_activity_java:-

```
package com.example.myapplication;
import android.content.Intent;
//import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.widget.TextView;
import androidx.appcompat.app.AppCompatActivity;

public class SecondActivity extends AppCompatActivity {
    TextView t1,t2,t3;
    String name,reg,dept;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_second);
        t1= (TextView) findViewById(R.id.textView1);
        t2= (TextView) findViewById(R.id.textView2);
        t3= (TextView) findViewById(R.id.textView3);
        //Getting the Intent
        Intent i = getIntent();
        //Getting the Values from First Activity using the Intent received
        name=i.getStringExtra("name_key");
        reg=i.getStringExtra("reg_key");
        dept=i.getStringExtra("dept_key");
        //Setting the Values to Intent
        t1.setText(name);
        t2.setText(reg);
        t3.setText(dept);}}
```

OUTPUT: -


2:00
My Application
Details Form
Name
Reg.No
Dept
CSE
SUBMIT


2:01
My Application
Details Form
Name
Victor
Reg.No
123456
Dept
CSE
SUBMIT


2:01
My Application
Victor
123456
CSE

Result:-

Thus the application that makes use of database has been Developed and the output was verified.

EX NO: 3 **Design a mobile application that is aware of the resource constraints of mobile devices**

DATE:

AIM:

To Implement mobile applications using UI toolkits and frameworks

PROCEDURE:

Step 1: Set up the Development Environment
- Install Android Studio, the official IDE for Android app development.
- Configure the necessary Android SDK and emulators.

Step 2: Create a New Project
- Open Android Studio and select "Start a new Android Studio project" or go to File -> New -> New Project.
- Configure your project settings, including the application name, package name, and project location.

Step 3: Design the User Interface (UI)
- Open the layout file associated with the main activity. By default, it is called "activity_main.xml."
- Design your UI using XML markup, leveraging the available UI components, layouts, and styles.
- Here's an example using LinearLayout and TextView:

Step 4: Implement the UI Toolkits in Java Code
- Open the Java file associated with the main activity. By default, it is called "MainActivity.java."
- Inside the `onCreate` method, set the layout using `setContentView(R.layout.activity_main)`.
- Access the UI elements defined in the layout using `findViewById`.
- You can add additional code logic as needed.

Step 5: Build and Run the Application
- Connect a physical Android device or use an emulator to test your application.
- Click the "Run" button in Android Studio, and the application will be installed and launched on the connected device/emulator.

PROGRAMS:-

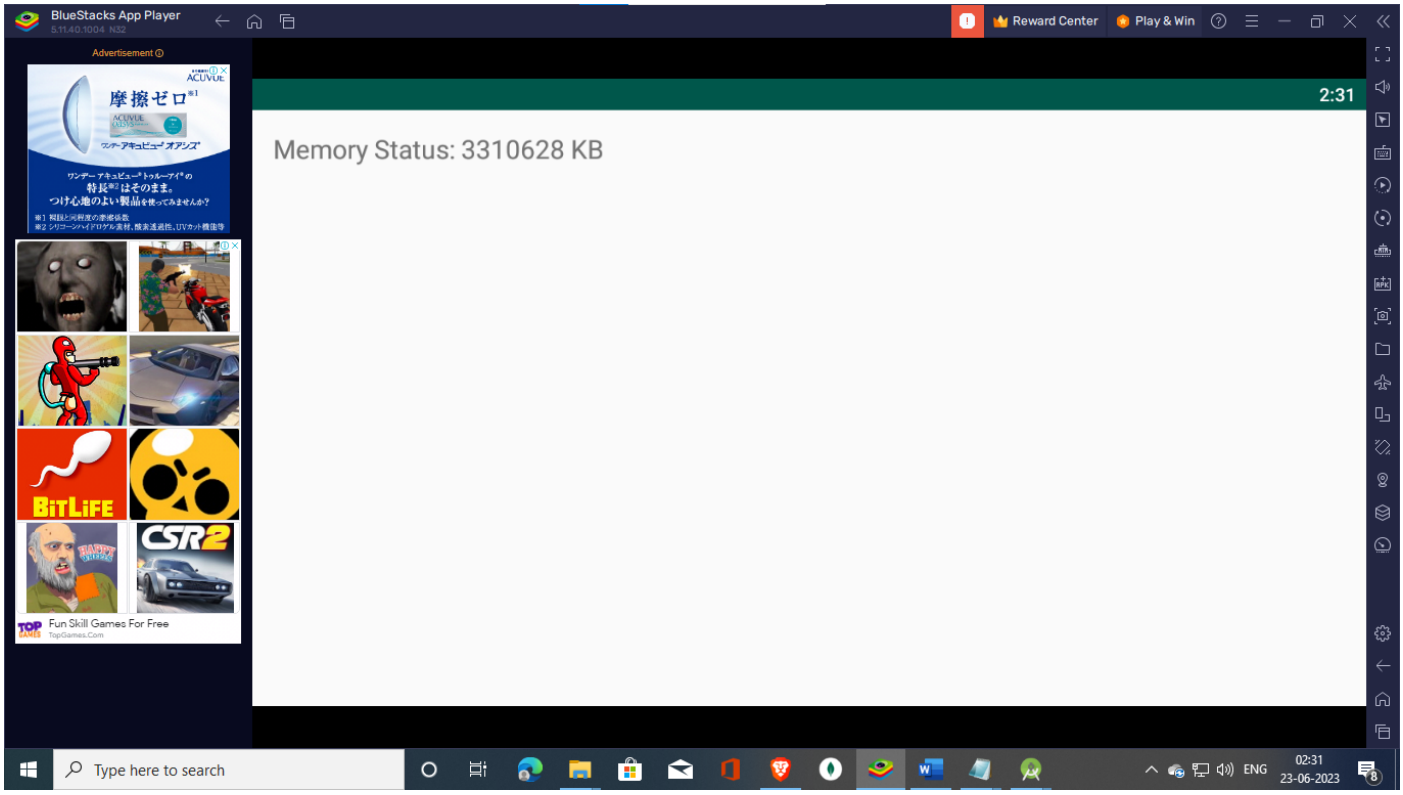
Code for activity_main.xml:-

```
<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/memoryStatusTextView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Memory Status: "
        android:textSize="20sp" />

</RelativeLayout>
```

Code for main activity_java:-

```
package com.example.myapplication;

import android.app.ActivityManager;
import android.content.Context;
import android.os.Bundle;
import android.widget.TextView;

import androidx.appcompat.app.AppCompatActivity;

import com.example.myapplication.R;

public class MainActivity extends AppCompatActivity {

    private TextView memoryStatusTextView;

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

        memoryStatusTextView = findViewById(R.id.memoryStatusTextView);

        ActivityManager activityManager = (ActivityManager)
getSystemService(Context.ACTIVITY_SERVICE);
        ActivityManager.MemoryInfo memoryInfo = new ActivityManager.MemoryInfo();
        activityManager.getMemoryInfo(memoryInfo);
        long availableMemory = memoryInfo.availMem / 1024;

        memoryStatusTextView.setText("Memory Status: " + availableMemory + " KB");

        if (availableMemory < 1024) {

        } else {

        }
    }
}
```

OUTPUT: -


BlueStacks App Player
Reward Center
Play & Win
Advertisement
2:31
Memory Status: 3310628 KB
Fun Skill Games For Free
Type here to search
ENG
02:31
23-06-2023

Result:-

Thus the application that makes use of database has been Developed and the output was verified

.

EX NO: 4 **Develop an Android Application that uses GPS location information.**

DATE:

Aim:

To develop an Android Application that uses GPS location information.

Procedure:

Creating a New project:

- Open Android Studio and then click on **File -> New -> New project**.
- Then type the Application name as **"exno7"**and click Next.
- Then **select the Minimum SDK** as shown below and click Next.
- Then **select the Empty Activity** and click Next.
- Finally click **Finish**.
- It will take some time to build and load the project.
- After completion it will look as given below.

Designing layout for the Android Application:

- Click on **app -> res -> layout -> activity_main.xml**.
- Now click on Text as shown below.
- Then delete the code which is there and type the code as given below.

Code for Activity_main.xml:

```
<?xml version = "1.0" encoding = "utf-8"?>
<LinearLayout xmlns:android = "http://schemas.android.com/apk/res/android"
  android:layout_width = "fill_parent"
  android:layout_height = "fill_parent"
  android:orientation = "vertical" >

<Button
    android:id = "@+id/button"
    android:layout_width = "fill_parent"
    android:layout_height = "wrap_content"
    android:text = "getlocation"/>

</LinearLayout>
```

- Now click on Design and your application will look as given below.
- So now the designing part is completed.

Following will be the content of res/values/strings.xml to define two new constants −

```
<?xml version = "1.0" encoding = "utf-8"?>
<resources>
<string name = "app_name">Tutorialspoint</string>
</resources>
```

Adding permissions in Manifest for the Android Application:

- Click on **app -> manifests -> AndroidManifest.xml.**

Code for AndroidManifest.xml:

```
<?xml version = "1.0" encoding = "utf-8"?>
<manifest xmlns:android = "http://schemas.android.com/apk/res/android"
  package = "com.example.tutorialspoint7.myapplication">
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
<uses-permission android:name = "android.permission.INTERNET" />
<application
    android:allowBackup =
    "true"
    android:icon =
    "@mipmap/ic_launcher"android:label
    = "@string/app_name"
    android:supportsRtl = "true"
    android:theme =
    "@style/AppTheme">

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

</manifest>
```

Java Coding for the Android Application:

- Click on **app -> java -> com.example.exno7 -> MainActivity**.
- Then delete the code which is there and type the code as given below.

Code for MainActivity.java:

```
packagecom.example.exno7;

import android.Manifest;
import android.app.Activity;
import android.os.Bundle;
import android.support.v4.app.ActivityCompat;
import android.test.mock.MockPackageManager;
import android.view.View;
import android.widget.Button;
import android.widget.Toast;

public class MainActivity extends Activity

  {Button btnShowLocation;
  private static final int REQUEST_CODE_PERMISSION = 2;
  String mPermission = Manifest.permission.ACCESS_FINE_LOCATION;

  // GPSTracker
  classGPSTracker
  gps;

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

    try {
      if (ActivityCompat.checkSelfPermission(this, mPermission)
        != MockPackageManager.PERMISSION_GRANTED) {

        ActivityCompat.requestPermissions(this, new String[]{mPermission},
          REQUEST_CODE_PERMISSION);

        // If any permission above not allowed by user, this condition will
          execute every time, else your else part will work
```

```
      }
    } catch (Exception e)
      {
      e.printStackTrace();
    }

    btnShowLocation = (Button) findViewById(R.id.button);
    // show location button click event btnShowLocation.setOnClickListener(new
    View.OnClickListener() {

      @Override
      public void onClick(View arg0) {
        // create class object
        gps = new GPSTracker(MainActivity.this);

        // check if GPS enabled
        if(gps.canGetLocation()){

          double latitude = gps.getLatitude();
          double longitude = gps.getLongitude();

          // \n is for new line
          Toast.makeText(getApplicationContext(), "Your Location is - \nLat: "
            + latitude + "\nLong: " + longitude, Toast.LENGTH_LONG).show();
        }else{
          // can't get location
          // GPS or Network is not enabled
          // Ask user to enable GPS/network in settings
          gps.showSettingsAlert();
        }

      }
    });
  }
}
```

- Following is the content of the modified main activity file **GPSTracker.java**.

Code for GPDTracker.Java

```
packagecom.example.exno7;
import android.app.AlertDialog;
import android.app.Service;
import android.content.Context;
import android.content.DialogInterface;import
android.content.Intent;
import android.location.Location;
import android.location.LocationListener;
```

```
import android.location.LocationManager;
import android.os.Bundle;
import   android.os.IBinder;
import android.provider.Settings;
import android.util.Log;
public class GPSTracker extends Service implements LocationListener {

  private final Context mContext;

  // flag for GPS status
  boolean isGPSEnabled = false;

  // flag for network status
  boolean isNetworkEnabled = false;

  // flag for GPS status
  boolean canGetLocation = false;

  Location location; // location
  double latitude; // latitude
  double longitude; // longitude

  // The minimum distance to change Updates in meters
  private static final long MIN_DISTANCE_CHANGE_FOR_UPDATES = 10; // 10 meters

  // The minimum time between updates in milliseconds
  private static final long MIN_TIME_BW_UPDATES = 1000 * 60 * 1; // 1 minute

  // Declaring a Location Manager
  protected LocationManager locationManager;

  public GPSTracker(Context context) {
    this.mContext = context;
    getLocation();
  }

  public Location getLocation()
    {try {
      locationManager = (LocationManager) mContext.getSystemService(LOCATION_SERVICE);

      // getting GPS status
      isGPSEnabled = locationManager.isProviderEnabled(LocationManager.GPS_PROVIDER);

      // getting network status
      isNetworkEnabled = locationManager
        .isProviderEnabled(LocationManager.NETWORK_PROVIDER);
```

```
if (!isGPSEnabled && !isNetworkEnabled) {
  // no network provider is enabled
} else {
  this.canGetLocation = true;
  // First get location from Network Provider
  if (isNetworkEnabled) {
    locationManager.requestLocationUpdates(
      LocationManager.NETWORK_PROVIDER,
      MIN_TIME_BW_UPDATES,
      MIN_DISTANCE_CHANGE_FOR_UPDATES, this);

    Log.d("Network", "Network");
    if (locationManager != null) {
      location = locationManager
        .getLastKnownLocation(LocationManager.NETWORK_PROVIDER);

      if (location != null) {
        latitude = location.getLatitude();
        longitude = location.getLongitude();
      }
    }
  }

  // if GPS Enabled get lat/long using GPS Services
  if (isGPSEnabled) {
    if (location == null) {
      locationManager.requestLocationUpdates(
        LocationManager.GPS_PROVIDER,
        MIN_TIME_BW_UPDATES,
        MIN_DISTANCE_CHANGE_FOR_UPDATES, this);

      Log.d("GPS Enabled", "GPS Enabled");
      if (locationManager != null) {
        location = locationManager
          .getLastKnownLocation(LocationManager.GPS_PROVIDER);

        if (location != null) {
          latitude = location.getLatitude();
          longitude = location.getLongitude();
        }
      }
```

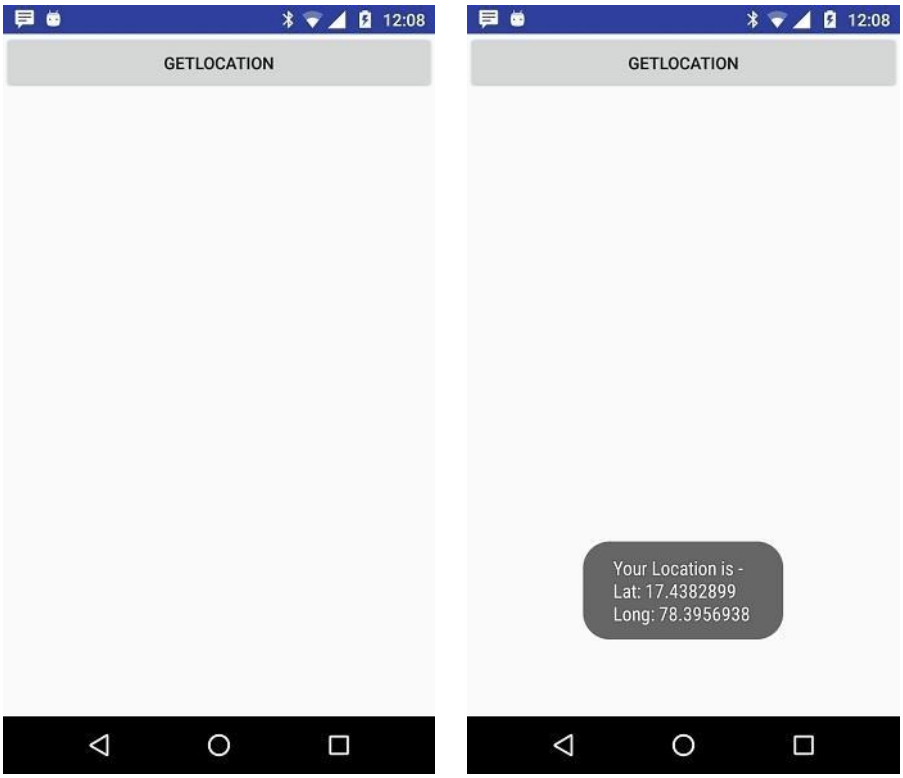
```
        }
      }
    }

  } catch (Exception e)
    {
    e.printStackTrace();
  }

  return location;
}

/**
  * Stop using GPS listener
  * Calling this function will stop using GPS in your app
* */

public void stopUsingGPS(){
  if(locationManager !=
  null){
    locationManager.removeUpdates(GPSTracker.this);
  }
}

/**
  * Function to get latitude
* */

public double getLatitude(){
  if(location != null){
    latitude = location.getLatitude();
  }

  // return latitude
  return latitude;
}

/**
  * Function to get longitude
* */

public double getLongitude(){
  if(location != null){
    longitude = location.getLongitude();
```

```
    }

    // return longitude
    return longitude;
  }

  /**
    * Function to check GPS/wifi enabled
    * @return boolean
  * */

  public boolean canGetLocation() {
    return this.canGetLocation;
  }

  /**
    * Function to show settings alert dialog
    * On pressing Settings button will lauch Settings Options
  * */

  public void showSettingsAlert(){
    AlertDialog.Builder alertDialog = new AlertDialog.Builder(mContext);

    // Setting Dialog Title
    alertDialog.setTitle("GPS is settings");

    // Setting Dialog Message
    alertDialog.setMessage("GPS is not enabled. Do you want to go to settings menu?");

    // On pressing Settings button
    alertDialog.setPositiveButton("Settings", new DialogInterface.OnClickListener() {public
      void onClick(DialogInterface dialog,int which) {
        Intent intent = new Intent(Settings.ACTION_LOCATION_SOURCE_SETTINGS);
        mContext.startActivity(intent);
      }
    });

    // on pressing cancel button
    alertDialog.setNegativeButton("Cancel", new DialogInterface.OnClickListener() {
      public void onClick(DialogInterface dialog, int which) {
        dialog.cancel();
      }
```

```
    });

    // Showing Alert Message
    alertDialog.show();
  }

  @Override
  public void onLocationChanged(Location location) {
  }

  @Override
  public void onProviderDisabled(String provider) {
  }

  @Override
  public void onProviderEnabled(String provider) {
  }

  @Override
  public void onStatusChanged(String provider, int status, Bundle extras) {
  }

  @Override
  public IBinder onBind(Intent arg0)
    {return null;
  }
}
```

- So now the Coding part is also completed.
- Now run the application to see the output.

Output:



Result:

Thus Android Application that implements GPS Location Information is developed and executedsuccessfully.

EX NO: 5 **Develop an application that makes use of mobile database**

DATE:

AIM:

To Develop an application that makes use of mobile database

PROCEDURE:

Step 1: Set up the Development Environment
- Install Android Studio, the official IDE for Android app development.
- Configure the necessary Android SDK and emulators.

Step 2: Create a New Project
- Open Android Studio and select "Start a new Android Studio project" or go to File -> New -> New Project.
- Configure your project settings, including the application name, package name, and project location.

Step 3: Design the User Interface (UI)
- Open the layout file associated with the main activity. By default, it is called "activity_main.xml."
- Design your UI using XML markup, leveraging the available UI components, layouts, and styles.
- Here's an example using LinearLayout and TextView:

Step 4: Implement the UI Toolkits in Java Code
- Open the Java file associated with the main activity. By default, it is called "MainActivity.java."
- Inside the `onCreate` method, set the layout using `setContentView(R.layout.activity_main)`.
- Access the UI elements defined in the layout using `findViewById`.
- You can add additional code logic as needed.

Step 5: Build and Run the Application
- Connect a physical Android device or use an emulator to test your application.
- Click the "Run" button in Android Studio, and the application will be installed and launched on the connected device/emulator.

PROGRAMS:-

Code for activity_main.xml:-

```
<?xml version="1.0" encoding="utf-8"?>
<AbsoluteLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent">
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_x="50dp"
        android:layout_y="20dp"
        android:text="Student Details"
        android:textSize="30sp" />
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_x="20dp"
        android:layout_y="110dp"
        android:text="Enter Rollno:"
        android:textSize="20sp" />
    <EditText
        android:id="@+id/Rollno"
        android:layout_width="150dp"
        android:layout_height="wrap_content"
        android:layout_x="175dp"
        android:layout_y="100dp"
        android:inputType="number"
        android:textSize="20sp" />
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_x="20dp"
        android:layout_y="160dp"
        android:text="Enter Name:"
        android:textSize="20sp" />
    <EditText
        android:id="@+id/Name"
        android:layout_width="150dp"
        android:layout_height="wrap_content"
        android:layout_x="175dp"
        android:layout_y="150dp"
        android:inputType="text"
        android:textSize="20sp" />
    <TextView
        android:layout_width="wrap_content"
```

```
        android:layout_height="wrap_content"
        android:layout_x="20dp"
        android:layout_y="210dp"
        android:text="Enter Marks:"
        android:textSize="20sp" />
    <EditText
        android:id="@+id/Marks"
        android:layout_width="150dp"
        android:layout_height="wrap_content"
        android:layout_x="175dp"
        android:layout_y="200dp"
        android:inputType="number"
        android:textSize="20sp" />
    <Button
        android:id="@+id/Insert"
        android:layout_width="150dp"
        android:layout_height="wrap_content"
        android:layout_x="25dp"
        android:layout_y="300dp"
        android:text="Insert"
        android:textSize="30dp" />
    <Button
        android:id="@+id/Delete"
        android:layout_width="150dp"
        android:layout_height="wrap_content"
        android:layout_x="200dp"
        android:layout_y="300dp"
        android:text="Delete"
        android:textSize="30dp" />
    <Button
        android:id="@+id/Update"
        android:layout_width="150dp"
        android:layout_height="wrap_content"
        android:layout_x="25dp"
        android:layout_y="400dp"
        android:text="Update"
        android:textSize="30dp" />
    <Button
        android:id="@+id/View"
        android:layout_width="150dp"
        android:layout_height="wrap_content"
        android:layout_x="200dp"
        android:layout_y="400dp"
        android:text="View"
        android:textSize="30dp" />
    <Button
        android:id="@+id/ViewAll"
        android:layout_width="200dp"
        android:layout_height="wrap_content"
```

```
            android:layout_x="100dp"
            android:layout_y="500dp"
            android:text="View All"
            android:textSize="30dp" />
</AbsoluteLayout>
```

Code for main activity_java:-

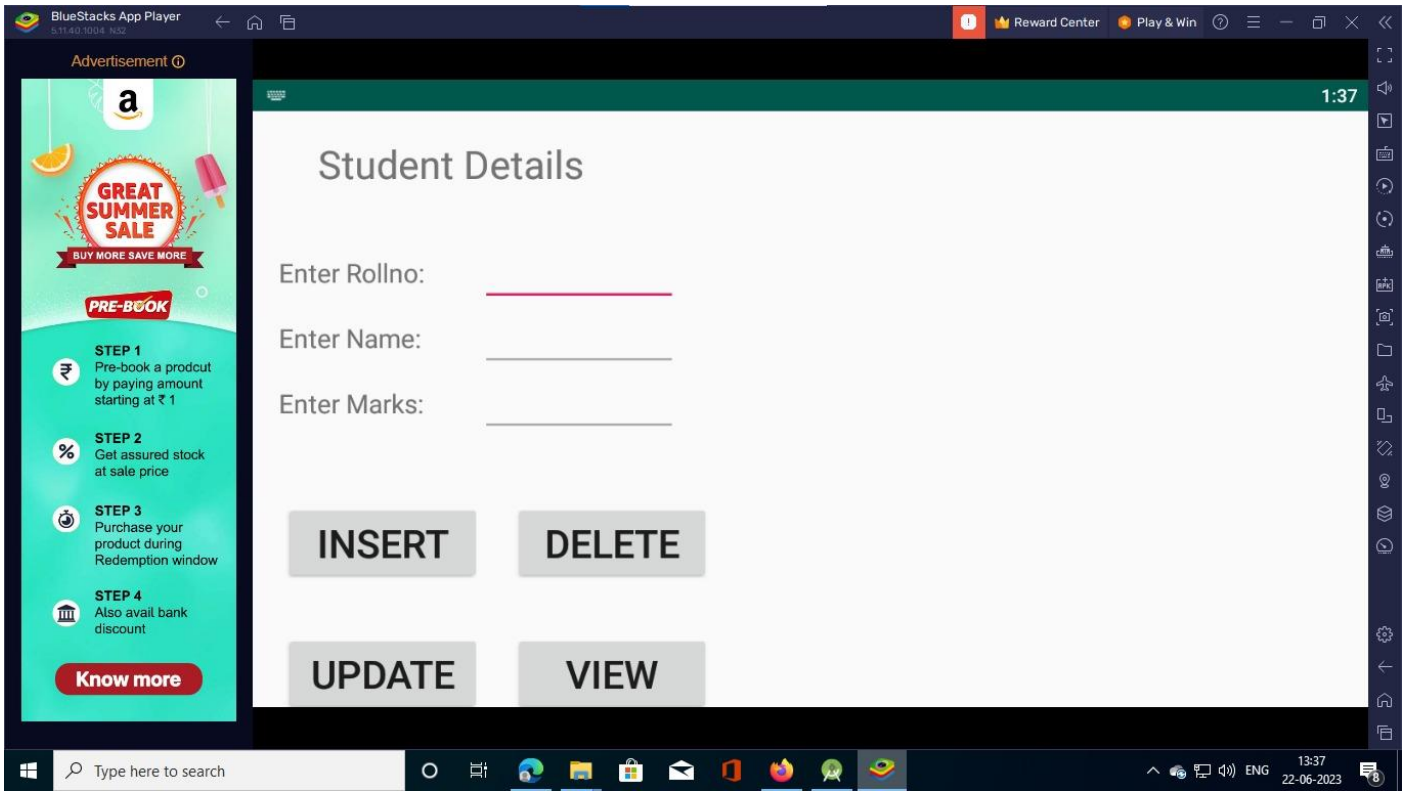
```
package com.example.myapplication;
import android.app.Activity;
import android.app.AlertDialog.Builder;
import android.content.Context;
import android.database.Cursor;
import android.database.sqlite.SQLiteDatabase;
import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.EditText;
public class MainActivity extends Activity implements OnClickListener
{
    EditText Rollno,Name,Marks;
    Button Insert,Delete,Update,View,ViewAll;
    SQLiteDatabase db;
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        Rollno=(EditText)findViewById(R.id.Rollno);
        Name=(EditText)findViewById(R.id.Name);
        Marks=(EditText)findViewById(R.id.Marks);
        Insert=(Button)findViewById(R.id.Insert);
        Delete=(Button)findViewById(R.id.Delete);
        Update=(Button)findViewById(R.id.Update);
        View=(Button)findViewById(R.id.View);
        ViewAll=(Button)findViewById(R.id.ViewAll);
        Insert.setOnClickListener(this);
        Delete.setOnClickListener(this);
        Update.setOnClickListener(this);
        View.setOnClickListener(this);
        ViewAll.setOnClickListener(this);
        // Creating database and table
        db=openOrCreateDatabase("StudentDB", Context.MODE_PRIVATE, null);
        db.execSQL("CREATE TABLE IF NOT EXISTS student(rollno VARCHAR,name
```

```
VARCHAR,marks VARCHAR);");
    }
    public void onClick(View view)
    {
        // Inserting a record to the Student table
        if(view==Insert)
        {
            // Checking for empty fields
            if(Rollno.getText().toString().trim().length()==0||
                    Name.getText().toString().trim().length()==0||
                    Marks.getText().toString().trim().length()==0)
            {
                showMessage("Error", "Please enter all values");
                return;
            }
            db.execSQL("INSERT INTO student
VALUES('"+Rollno.getText()+"','"+Name.getText()+
                    "','"+Marks.getText()+"');");
            showMessage("Success", "Record added");
            clearText();
        }
        // Deleting a record from the Student table
        if(view==Delete)
        {
            // Checking for empty roll number
            if(Rollno.getText().toString().trim().length()==0)
            {
                showMessage("Error", "Please enter Rollno");
                return;
            }
            Cursor c=db.rawQuery("SELECT * FROM student WHERE
rollno='"+Rollno.getText()+"'", null);
            if(c.moveToFirst())
            {
                db.execSQL("DELETE FROM student WHERE rollno='"+Rollno.getText()+"'");
                showMessage("Success", "Record Deleted");
            }
            else
            {
                showMessage("Error", "Invalid Rollno");
            }
            clearText();
        }
        // Updating a record in the Student table
        if(view==Update)
        {
            // Checking for empty roll number
            if(Rollno.getText().toString().trim().length()==0)
            {
```

```
            showMessage("Error", "Please enter Rollno");
            return;
        }
        Cursor c=db.rawQuery("SELECT * FROM student WHERE
rollno='"+Rollno.getText()+"'", null);
        if(c.moveToFirst()) {
            db.execSQL("UPDATE student SET name='" + Name.getText() + "',marks='" +
                Marks.getText() +
                "' WHERE rollno='"+Rollno.getText()+"'");
            showMessage("Success", "Record Modified");
        }
        else {
            showMessage("Error", "Invalid Rollno");
        }
        clearText();
    }
    // Display a record from the Student table
    if(view==View)
    {
        // Checking for empty roll number
        if(Rollno.getText().toString().trim().length()==0)
        {
            showMessage("Error", "Please enter Rollno");
            return;
        }
        Cursor c=db.rawQuery("SELECT * FROM student WHERE
rollno='"+Rollno.getText()+"'", null);
        if(c.moveToFirst())
        {
            Name.setText(c.getString(1));
            Marks.setText(c.getString(2));
        }
        else
        {
            showMessage("Error", "Invalid Rollno");
            clearText();
        }
    }
    // Displaying all the records
    if(view==ViewAll)
    {
        Cursor c=db.rawQuery("SELECT * FROM student", null);
        if(c.getCount()==0)
        {
            showMessage("Error", "No records found");
            return;
        }
        StringBuffer buffer=new StringBuffer();
        while(c.moveToNext())
```

```
            {
                buffer.append("Rollno: "+c.getString(0)+"\n");
                buffer.append("Name: "+c.getString(1)+"\n");
                buffer.append("Marks: "+c.getString(2)+"\n\n");
            }
            showMessage("Student Details", buffer.toString());
        }
    }
    public void showMessage(String title,String message)
    {
        Builder builder=new Builder(this);
        builder.setCancelable(true);
        builder.setTitle(title);
        builder.setMessage(message);
        builder.show();
    }
    public void clearText()
    {
        Rollno.setText("");
        Name.setText("");
        Marks.setText("");
        Rollno.requestFocus();
    }
}
```

OUTPUT: -


BlueStacks App Player
Reward Center
Play & Win
Advertisement
GREAT SUMMER SALE
BUY MORE SAVE MORE
PRE-BOOK
STEP 1
Pre-book a prodcut by paying amount starting at ₹ 1
STEP 2
Get assured stock at sale price
STEP 3
Purchase your product during Redemption window
STEP 4
Also avail bank discount
Know more
1:37
Student Details
Enter Rollno:
Enter Name:
Enter Marks:
INSERT
DELETE
UPDATE
VIEW
Type here to search
ENG
13:37
22-06-2023

Result:-

Thus the application that makes use of database has been Developed and the output was verified.

EX NO: 6 **Implement an android application that writes data into the SD card**

DATE:

AIM:

To Implement an android application that writes data into the SD card

PROCEDURE:

Step 1: Set up the Development Environment
- Install Android Studio, the official IDE for Android app development.
- Configure the necessary Android SDK and emulators.

Step 2: Create a New Project
- Open Android Studio and select "Start a new Android Studio project" or go to File -> New -> New Project.
- Configure your project settings, including the application name, package name, and project location.

Step 3: Design the User Interface (UI)
- Open the layout file associated with the main activity. By default, it is called "activity_main.xml."
- Design your UI using XML markup, leveraging the available UI components, layouts, and styles.
- Here's an example using LinearLayout and TextView:

Step 4: Implement the UI Toolkits in Java Code
- Open the Java file associated with the main activity. By default, it is called "MainActivity.java."
- Inside the `onCreate` method, set the layout using `setContentView(R.layout.activity_main)`.
- Access the UI elements defined in the layout using `findViewById`.
- You can add additional code logic as needed.

Step 5: Build and Run the Application
- Connect a physical Android device or use an emulator to test your application.
- Click the "Run" button in Android Studio, and the application will be installed and launched on the connected device/emulator.

PROGRAMS:-

Code for activity_main.xml:-

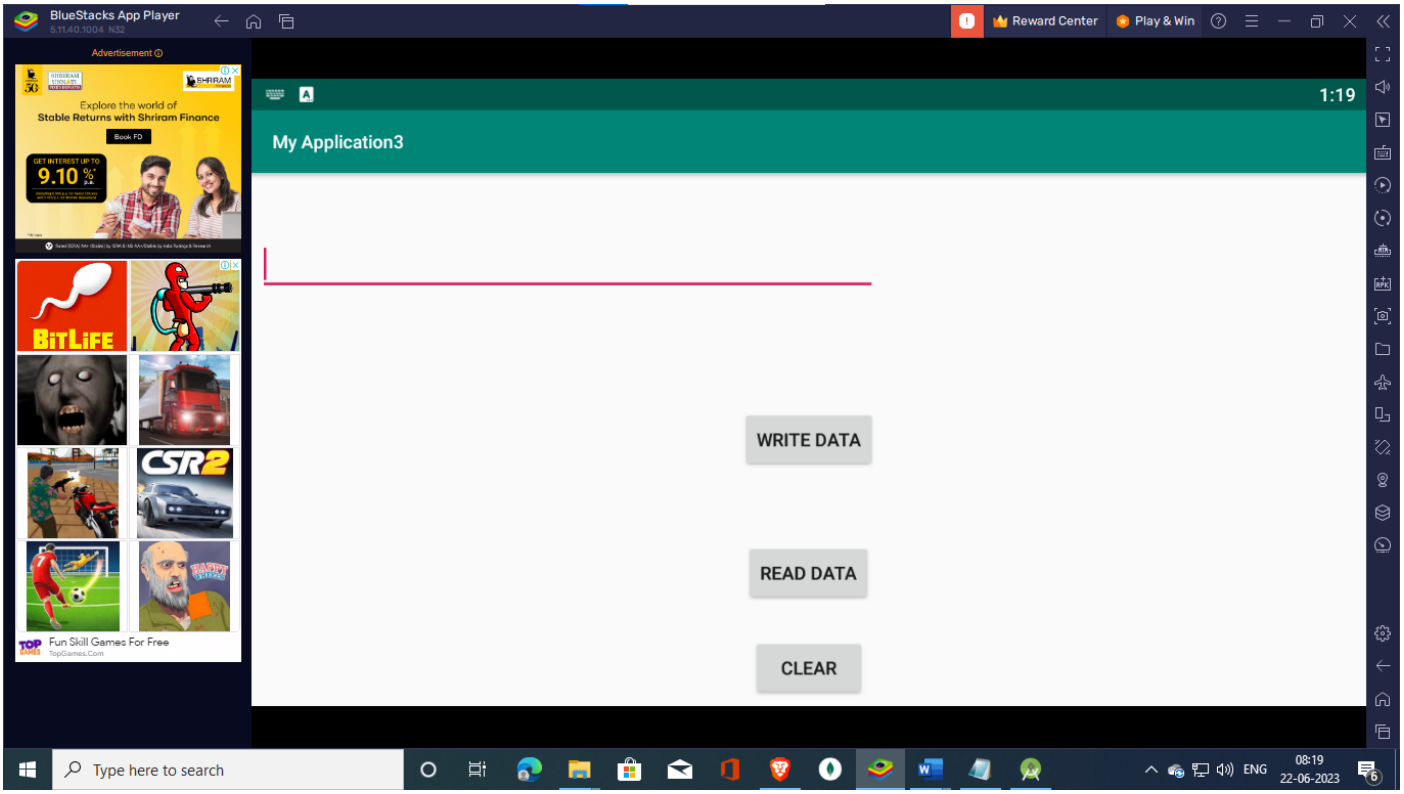
```
<?xml version="1.0" encoding="utf-8"?>
<androidx.coordinatorlayout.widget.CoordinatorLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"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

    <com.google.android.material.appbar.AppBarLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:theme="@style/AppTheme.AppBarOverlay">

        <androidx.appcompat.widget.Toolbar
            android:id="@+id/toolbar"
            android:layout_width="match_parent"
            android:layout_height="?attr/actionBarSize"
            android:background="?attr/colorPrimary"
            app:popupTheme="@style/AppTheme.PopupOverlay" />

    </com.google.android.material.appbar.AppBarLayout>

    <include layout="@layout/content_main" />

    <com.google.android.material.floatingactionbutton.FloatingActionButton
        android:id="@+id/fab"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="bottom|end"
        android:layout_margin="@dimen/fab_margin"
        app:srcCompat="@android:drawable/ic_dialog_email" />

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

Code for main activity_java:-

```
package com.example.myapplication;

import android.os.Bundle;

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

import androidx.appcompat.app.AppCompatActivity;
import androidx.appcompat.widget.Toolbar;

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

public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        Toolbar toolbar = findViewById(R.id.toolbar);
        setSupportActionBar(toolbar);

        FloatingActionButton fab = findViewById(R.id.fab);
        fab.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                Snackbar.make(view, "Replace with your own action", Snackbar.LENGTH_LONG)
                        .setAction("Action", null).show();
            }
        });
    }

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

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

```
        //noinspection SimplifiableIfStatement
        if (id == R.id.action_settings) {
            return true;
        }

        return super.onOptionsItemSelected(item);
    }
}
```

OUTPUT: -


BlueStacks App Player
Reward Center
Play & Win
Advertisement
Explore the world of
Stable Returns with Shriram Finance
Book FD
BitLiFE
CSR2
Fun Skill Games For Free
1:19
My Application3
WRITE DATA
READ DATA
CLEAR
Type here to search
ENG
08:19
22-06-2023

Result:-

Thus the application that makes use of database has been Developed and the output was verified.

EX NO: 7 DEVELOP A MOBILE APPLICATION THAT USES GUICOMPONENTS, FONT AND COLORS.

DATE:

Aim:-

To develop a Simple Android Application that uses GUI components, Font and Colors.

Procedure:-

Creating a New project:-

- Open Android Studio and then click on **File -> New -> New project.**
- Then type the Application name as "**ex.no.1a"** and click **Next.**
- Then select the **Minimum SDK** as shown below and click **Next**.
- Then select the **Empty Activity** and click **Next.**
- Finally click F**inish**.
- It will take some time to build and load the project.

Designing layout for the Android Application:-

- **Click on** app -> res -> layout -> activity_main.xml.
- Now click on Design
- Drag and drop the following components:
  - One TextView with text Hello World
  - Three Buttons with labeled as Change Font Size, Change Font Color and Change Font Style
- Now click on Text and do the necessary modification/insert properties in activity_main.xml file.

Java Coding and Build for the Android Application:-

- Click on **app -> java -> com.example.exno1a -> MainActivity.**java and enter the java coding for android application.
- Select and click Build Project(Android) from Build Menu.
- Select and click Build APK from Build Menu.
- Open BlueStack software. Drag and Drop apk file into Blue stack and double click on android application to view the output or Run android application with emulator.

PROGRAMS:-

activity_main.xml:-

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="match_parent" android:layout_height="match_parent">
 <TextView
        android:id="@+id/textView"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_margin="30dp"
        android:gravity="center" android:text="Hello
        World!" android:textSize="25sp"
        android:textStyle="bold" />
```

```
    <Button
        android:id="@+id/button1"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_margin="20dp"
        android:gravity="center"
        android:text="Change font size"
        android:textSize="25sp" />
    <Button
        android:id="@+id/button2"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_margin="20dp"
        android:gravity="center"
        android:text="Change color"
        android:textSize="25sp" />
</LinearLayout>
```

Code for MainActivity.java:-

```
package com.example.exno1a;
import android.graphics.Color;
import android.support.v7.app.AppCompatActivity;import
android.os.Bundle;
import android.view.View; import
android.widget.Button; import
android.widget.TextView;
public class MainActivity extends AppCompatActivity
{
    int  ch=1; float
    font=30;
    @Override
    protected void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState); setContentView(R.layout.activity_main);
        final TextView t= (TextView) findViewById(R.id.textView);Button
        b1= (Button) findViewById(R.id.button1);
        b1.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                t.setTextSize(font);
                 font = font + 5;if
                (font == 50) font
                        = 30;
            }
         });
        Button b2= (Button) findViewById(R.id.button2);
        b2.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                switch (ch) {
                    case 1:
                        t.setTextColor(Color.RED);
                        break;
                    case 2:
```

```
                t.setTextColor(Color.GREEN);
                break;
            case 3:
                t.setTextColor(Color.BLUE);break;
            case 4:
                t.setTextColor(Color.CYAN);
                break;
            case 5:
                t.setTextColor(Color.YELLOW);break;
            case 6:
                t.setTextColor(Color.MAGENTA);break;
        }
        ch++;
        if (ch == 7)
            ch = 1;
    }
});
}
```

OUTPUT:-


BlueStacks
Home
Exno1A
Account
Exno1A
MAD LAB
CHANGE TEXT SIZE
CHANGE FONT COLOR
CHANGE FONT STYLE
exno9ms - [E:\Sat...
MAD LAB OUTPU...
BlueStacks
12:48 PM
3/18/2019

Result:-

Thus the application that uses GUI Components, Fonts and Colors has been developed andthe output was verified.

EX NO: 8 **Develop an android application using telephony to send**

DATE: **SMS**

AIM:

To Develop an android application using telephony to send SMS.

PROCEDURE:

Step 1: Set up the Development Environment
- Install Android Studio, the official IDE for Android app development.
- Configure the necessary Android SDK and emulators.

Step 2: Create a New Project
- Open Android Studio and select "Start a new Android Studio project" or go to File -> New -> New Project.
- Configure your project settings, including the application name, package name, and project location.

Step 3: Design the User Interface (UI)
- Open the layout file associated with the main activity. By default, it is called "activity_main.xml."
- Design your UI using XML markup, leveraging the available UI components, layouts, and styles.
- Here's an example using LinearLayout and TextView:

Step 4: Implement the UI Toolkits in Java Code
- Open the Java file associated with the main activity. By default, it is called "MainActivity.java."
- Inside the `onCreate` method, set the layout using `setContentView(R.layout.activity_main)`.
- Access the UI elements defined in the layout using `findViewById`.
- You can add additional code logic as needed.

Step 5: Build and Run the Application
- Connect a physical Android device or use an emulator to test your application.
- Click the "Run" button in Android Studio, and the application will be installed and launched on the connected device/emulator.

PROGRAMS:-

Code for activity_main.xml:-

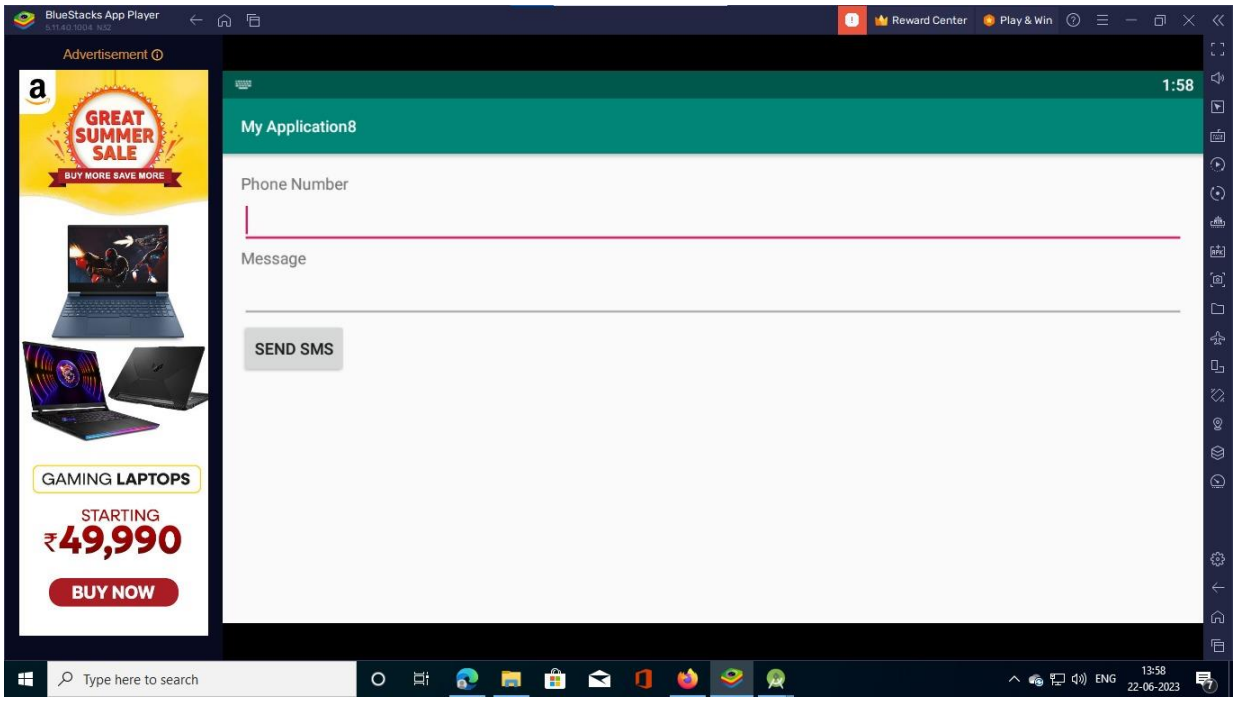
```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:padding="16dp"
    tools:context=".MainActivity">

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

    <EditText
        android:id="@+id/phoneEditText"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:inputType="phone" />

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

    <EditText
        android:id="@+id/messageEditText"
        android:layout_width="match_parent"
        android:layout_height="wrap_content" />

    <Button
        android:id="@+id/sendButton"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Send SMS" />

</LinearLayout>
```

Code for main activity_java:-

```
package com.example.myapplication;
import android.Manifest;
import android.content.pm.PackageManager;
import android.os.Bundle;
import android.telephony.SmsManager;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;

import androidx.appcompat.app.AppCompatActivity;
import androidx.core.app.ActivityCompat;
import androidx.core.content.ContextCompat;

public class MainActivity extends AppCompatActivity {

    private static final int PERMISSION_REQUEST_CODE = 1;

    private EditText phoneEditText;
    private EditText messageEditText;
    private Button sendButton;

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

        phoneEditText = findViewById(R.id.phoneEditText);
        messageEditText = findViewById(R.id.messageEditText);
        sendButton = findViewById(R.id.sendButton);

        sendButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                if (ContextCompat.checkSelfPermission(MainActivity.this,
                        Manifest.permission.SEND_SMS) !=
```

```
PackageManager.PERMISSION_GRANTED) {
                ActivityCompat.requestPermissions(MainActivity.this,
                        new String[]{Manifest.permission.SEND_SMS},
PERMISSION_REQUEST_CODE);
            } else {
                sendSMS();
            }
        }
    });
}

private void sendSMS() {
    String phoneNumber = phoneEditText.getText().toString();
    String message = messageEditText.getText().toString();

    try {
        SmsManager smsManager = SmsManager.getDefault();
        smsManager.sendTextMessage(phoneNumber, null, message, null, null);
        Toast.makeText(getApplicationContext(), "SMS sent successfully.",
Toast.LENGTH_SHORT).show();
    } catch (Exception e) {
        Toast.makeText(getApplicationContext(), "Failed to send SMS.",
Toast.LENGTH_SHORT).show();
    }}}
```

OUTPUT: -


BlueStacks App Player
Reward Center
Play & Win
Advertisement
GREAT SUMMER SALE
BUY MORE SAVE MORE
GAMING LAPTOPS
STARTING
₹49,990
BUY NOW
1:58
My Application8
Phone Number
Message
SEND SMS
Type here to search
ENG
13:58
22-06-2023

Result:-

Thus the application that makes use of database has been Developed and the output was verified.