Mobile Computing Lab
Modules 4 and 5: Graphics and Animation, Multimedia, and Location-Based Services

Assignment 2:

1. Write a program to draw different shapes on a canvas.

Aim: To create an Android application that draws different shapes (circle, rectangle) and text on a canvas

Objective:

- Implement a custom view that allows drawing on a canvas.
- Draw various shapes like circles and rectangles on the canvas.
- Display text on the canvas.

Theory:

This program demonstrates the use of the Canvas class in Android to create custom drawings. The Canvas class, combined with the Paint class, enables developers to draw shapes, lines, and text on a view. In this example, the ShapeView class extends the View class and overrides the onDraw method to render shapes like a circle and a rectangle, as well as display text on the screen. The Paint object is used to set the color and style for these drawings. The MainActivity sets this custom view as the content of the activity.

Custom View: The ShapeView class extends View to create a custom view that overrides the onDraw method. This method is where the drawing of shapes and text is performed.

Canvas: The Canvas class is used to draw shapes such as circles, rectangles, and text on the screen. It provides methods like drawCircle, drawRect, and drawText for rendering these elements.

Paint: The Paint class is used to define the style and color for drawing on the canvas. It allows setting the color and text size, and applying styles such as FILL for filling shapes.

Code:

> MainActivity.java

```
package com.example.module2;

import android.os.Bundle;
```

```
import android.widget.TextView;

import androidx.appcompat.app.AppCompatActivity;

public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(new ShapeView(MainActivity.this));
    }
}
```

> ShapeView.java

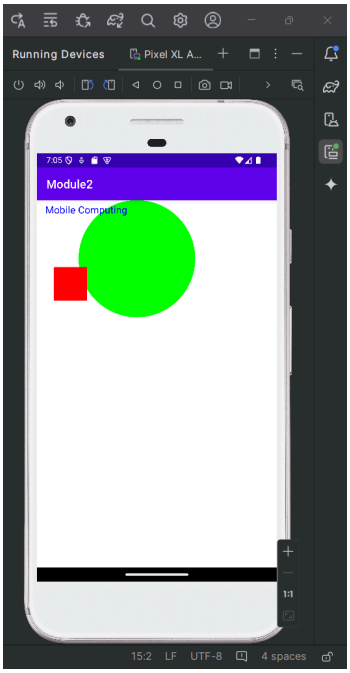
```
package com.example.module2;

import android.content.Context;
import android.graphics.Canvas;
import android.graphics.Color;
import android.graphics.Paint;
import android.view.View;

import androidx.annotation.NonNull;

public class ShapeView extends View {

    Paint paint = new Paint();

    public ShapeView(Context context) {
        super(context);
    }

    public void onDraw(@NonNull Canvas canvas) {
        super.onDraw(canvas);

        paint.setStyle(Paint.Style.FILL);
        paint.setColor(Color.GREEN);
        canvas.drawCircle(600, 350, 350, paint);

        paint.setColor(Color.RED);
        canvas.drawRect(100, 400, 300, 600, paint);

        paint.setColor(Color.BLUE);
        paint.setTextSize(60);
        canvas.drawText("Mobile Computing", 50, 80, paint);
    }
```

}

> activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    tools:context=".MainActivity">

</LinearLayout>
```

Output:



2. Write a program to play audio.

Aim: To create an Android application that plays audio using a service

Objective:

- Implement an Android service to handle background audio playback.
- Provide user controls to start and stop the audio playback.
- Display appropriate messages when the audio starts or stops.

Theory:

This program demonstrates how to play audio in an Android application using the MediaPlayer class within a service. The MusicService class extends the Service class to manage audio playback in the background. The service is started and stopped via intents from the MainActivity2 class, which provides user interface buttons for controlling the music.

The MediaPlayer object is used to load and play the audio file located in the res/raw directory. By using a service, the audio can continue playing even when the user navigates away from the activity, ensuring a seamless audio experience.

Service: A component that runs in the background to perform long-running operations, such as playing audio.

MediaPlayer: Provides methods to play, pause, and control audio playback. It is used here to manage and play the audio file.

IBinder: Used to provide communication between the service and other components, although it's not implemented in this case.

Code:

> MainActivity2.java

```
package com.example.module2;

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

import androidx.appcompat.app.AppCompatActivity;

public class MainActivity2 extends AppCompatActivity {

    Button Start, Stop;
```

```
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main2);

        Start = findViewById(R.id.start);
        Stop = findViewById(R.id.stop);
    }

    public void startMusic(View view) {
        Intent intent = new Intent(MainActivity2.this, MusicService.class);
        startService(intent);
    }

    public void stopMusic(View view) {
        Intent intent = new Intent(MainActivity2.this, MusicService.class);
        stopService(intent);
    }

}
```

> activity_main2.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:padding="10dp"
    tools:context=".MainActivity2">

    <TextView
        android:id="@+id/textView"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:textSize="24dp"
        android:gravity="center"
        android:text="14 Akash Choudhary" />

    <ImageView
        android:id="@+id/imageView"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
```

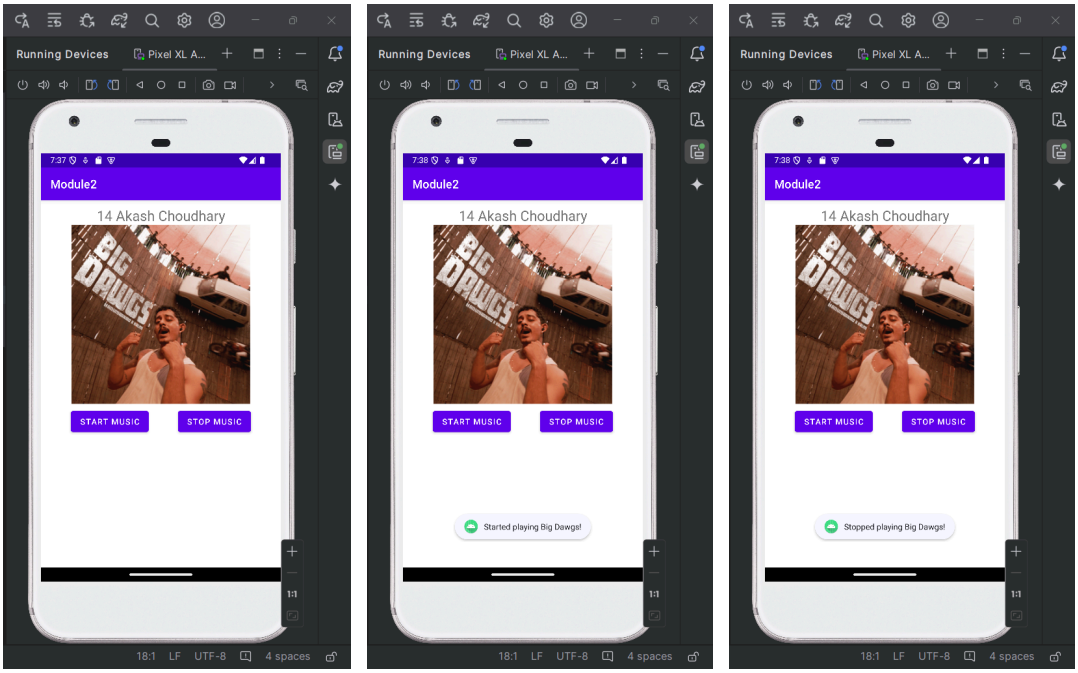
```
      app:srcCompat="@drawable/album" />

   <LinearLayout
      android:layout_width="match_parent"
      android:layout_height="60dp"
      android:gravity="center">

      <Button
         android:id="@+id/start"
         android:layout_width="wrap_content"
         android:layout_height="wrap_content"
         android:onClick="startMusic"
         android:text="Start Music" />

      <Button
         android:id="@+id/stop"
         android:layout_width="wrap_content"
         android:layout_height="wrap_content"
         android:onClick="stopMusic"
         android:layout_marginStart="50dp"
         android:text="Stop Music" />
   </LinearLayout>
</LinearLayout>
```

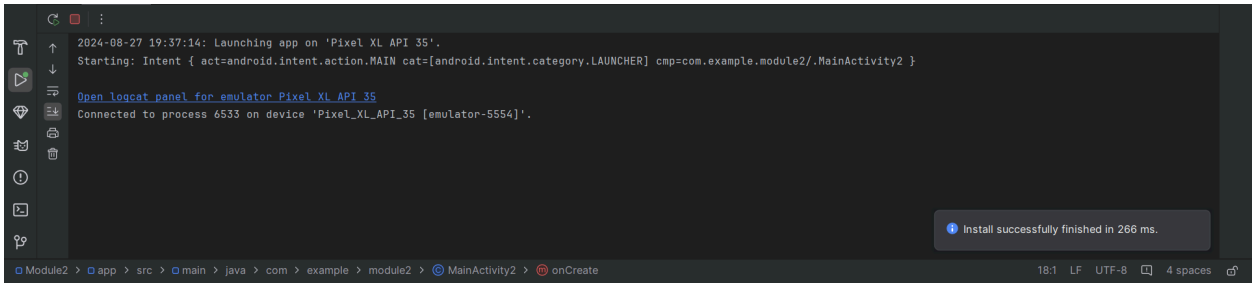
> MusicService.java

```
package com.example.module2;

import android.app.Service;
import android.content.Intent;
import android.media.MediaPlayer;
import android.os.IBinder;
import android.widget.Toast;

public class MusicService extends Service {

   MediaPlayer mediaPlayer;

   public int onStartCommand(Intent intent, int flags, int startId) {
      mediaPlayer = MediaPlayer.create(MusicService.this, R.raw.music);
      mediaPlayer.setLooping(true);
      mediaPlayer.start();
                        Toast.makeText(MusicService.this, "Started playing Big Dawgs!",
Toast.LENGTH_SHORT).show();
      return START_STICKY;
   }
```

```
    public void onDestroy() {
        super.onDestroy();
        mediaPlayer.stop();
                        Toast.makeText(MusicService.this, "Stopped playing Big Dawgs!",
Toast.LENGTH_SHORT).show();
    }

    public MusicService() {
    }

    @Override
    public IBinder onBind(Intent intent) {
        // TODO: Return the communication channel to the service.
        throw new UnsupportedOperationException("Not yet implemented");
    }
}
```

Output:





3. Write a program to play video.

Aim: To create an Android application that plays a video using a VideoView

Objective:

- Implement a VideoView to display video content.
- Use MediaController to provide video playback controls.
- Load and play a video resource from the app's resources.

Theory:

This program demonstrates the use of Android's VideoView class to play video files. The VideoView widget is used to display video content, while the MediaController provides playback controls such as play, pause, and seek.

The video is loaded from the app's raw resource directory using a URI, which is then set on the VideoView. This setup allows users to watch the video with basic playback controls embedded in the UI.

VideoView: A view that displays video content and provides basic playback functionality.

MediaController: Provides playback controls for the VideoView, such as play, pause, and seek.

Uri: Represents a Uniform Resource Identifier used to specify the location of the video resource.

Code:

> MainActivity3.java

```
package com.example.module2;

import android.net.Uri;
import android.os.Bundle;
import android.widget.MediaController;
import android.widget.VideoView;

import androidx.appcompat.app.AppCompatActivity;

public class MainActivity3 extends AppCompatActivity {

    VideoView videoView;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main3);
```

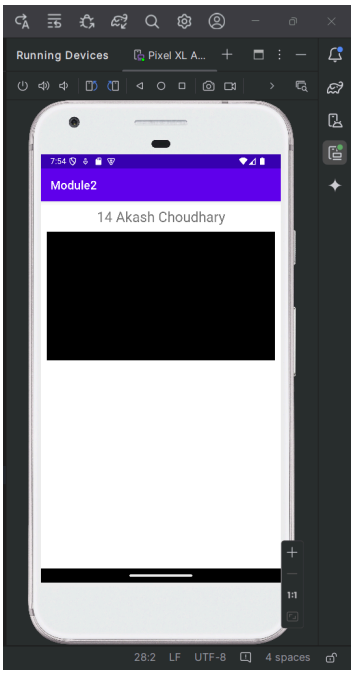
```
        videoView = findViewById(R.id.videoView);
        String path = "android.resource://" + getPackageName() + "/" + R.raw.video;
        Uri uri = Uri.parse(path);
        videoView.setVideoURI(uri);

        MediaController mediaController = new MediaController(MainActivity3.this);
        videoView.setMediaController(mediaController);
        mediaController.setAnchorView(videoView);
    }
}
```

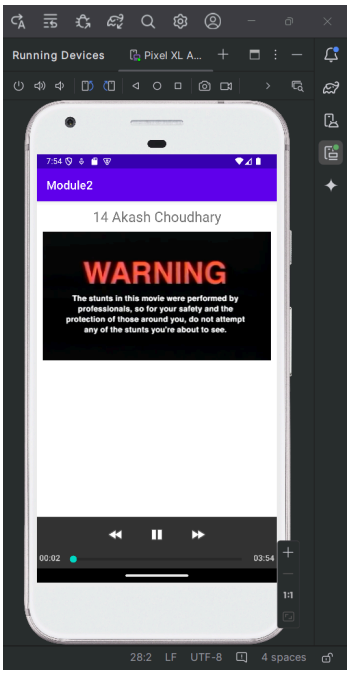
> activity_main3.xml

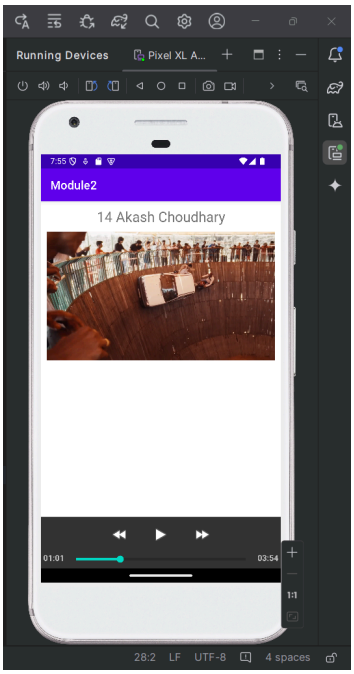
```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:padding="10dp"
    tools:context=".MainActivity3">

    <TextView
        android:id="@+id/textView2"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:textSize="24dp"
        android:gravity="center"
        android:text="14 Akash Choudhary" />

    <VideoView
        android:id="@+id/videoView"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginTop="10dp"
        android:layout_gravity="center" />
</LinearLayout>
```

Output:


Run
app
2024-08-27 19:53:40: Launching app on 'Pixel XL API 35'.
Starting: Intent { act=android.intent.action.MAIN cat=[android.intent.category.LAUNCHER] cmp=com.example.module2/.MainActivity3 }
Open logcat panel for emulator Pixel XL API 35
Connected to process 7612 on device 'Pixel_XL_API_35 [emulator-5554]'.
Install successfully finished in 2 s 98 ms.
Module2 > app > src > main > java > com > example > module2 > MainActivity3
28:2 LF UTF-8 4 spaces


Running Devices
Pixel XL A...
Module2
14 Akash Choudhary
28:2 LF UTF-8 4 spaces


Running Devices
Pixel XL A...
Module2
14 Akash Choudhary
WARNING
The stunts in this movie were performed by professionals, so for your safety and the protection of those around you, do not attempt any of the stunts you're about to see.
28:2 LF UTF-8 4 spaces


Running Devices
Pixel XL A...
Module2
14 Akash Choudhary
28:2 LF UTF-8 4 spaces

4. Write a program to perform frame-by-frame animation.

Aim: To create an Android application that performs frame-by-frame animation using AnimationDrawable

Objective:

- Implement a frame-by-frame animation using AnimationDrawable.
- Provide buttons to start and stop the animation.
- Use XML resources to define the animation frames and their durations.

Theory:

This program demonstrates the use of Android's AnimationDrawable class for frame-by-frame animation. The AnimationDrawable is used to define and control a sequence of drawable resources, which are displayed in succession to create the appearance of animation. In MainActivity4, the ImageView displays the animation defined in the running.xml file, which contains the sequence of drawable resources (frames) and their display durations.

The animation can be started and stopped using the corresponding methods in the MainActivity4 class. The activity_main4.xml layout file sets up the UI elements, including the ImageView for animation and buttons to control it.

ImageView: A UI component used to display images and animations. In this case, it displays the frame-by-frame animation.

AnimationDrawable: A class used to create frame-by-frame animations by displaying a sequence of drawable resources.

Code:

> MainActivity4.java

```
package com.example.module2;

import android.graphics.drawable.AnimationDrawable;
import android.os.Bundle;
import android.view.View;
import android.widget.ImageView;

import androidx.activity.EdgeToEdge;
import androidx.appcompat.app.AppCompatActivity;
import androidx.core.graphics.Insets;
import androidx.core.view.ViewCompat;
import androidx.core.view.WindowInsetsCompat;
```

```
public class MainActivity4 extends AppCompatActivity {

    ImageView imageView;
    AnimationDrawable animationDrawable;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main4);

        imageView = findViewById(R.id.imageView);
        imageView.setImageResource(R.drawable.running);

        animationDrawable = (AnimationDrawable) imageView.getDrawable();
    }

    public void startAnimation(View view) {
        animationDrawable.start();
    }

    public void stopAnimation(View view) {
        animationDrawable.stop();
    }
}
```

> activity_main4.xml

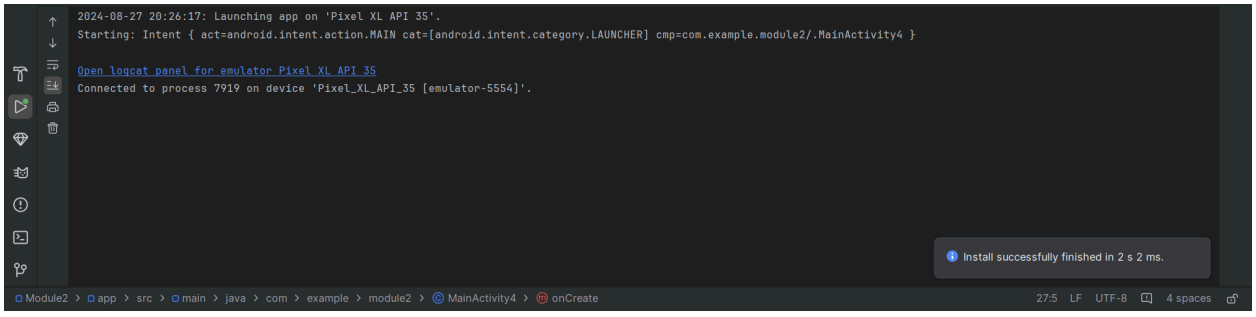
```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:padding="10dp"
    tools:context=".MainActivity4">

    <TextView
        android:id="@+id/textView"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:textSize="24dp"
        android:gravity="center"
        android:text="14 Akash Choudhary" />
```

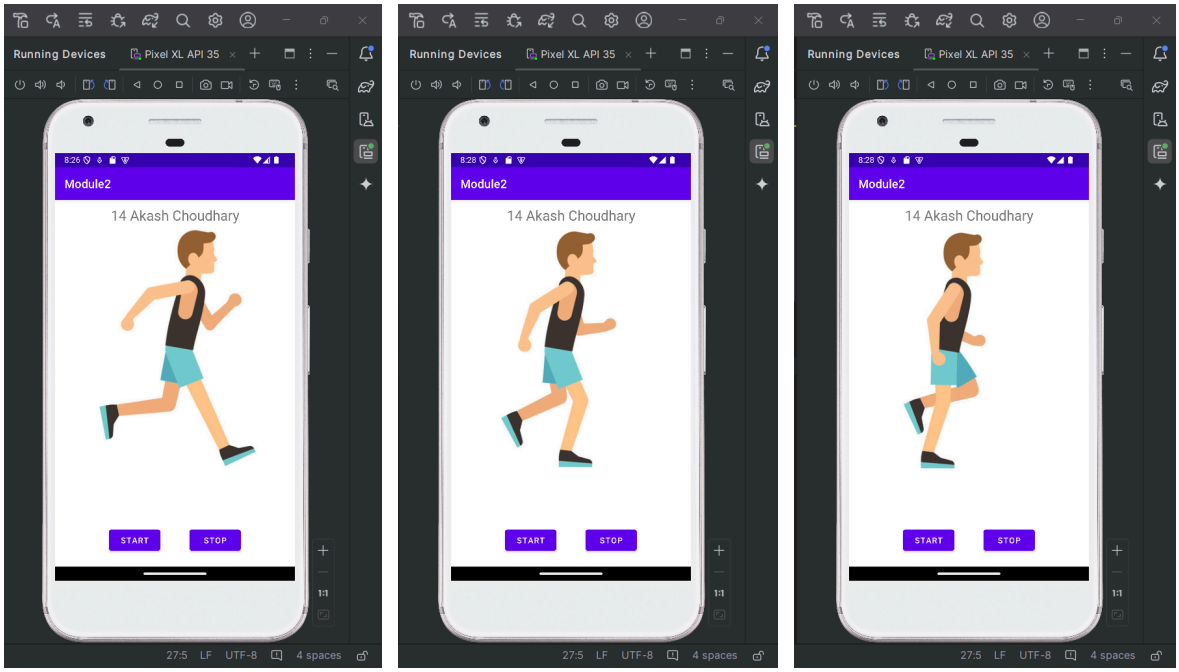
```
    <ImageView
        android:id="@+id/imageView"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        app:srcCompat="@drawable/one" />

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:gravity="center_horizontal">

        <Button
            android:id="@+id/start"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:onClick="startAnimation"
            android:text="Start" />

        <Button
            android:id="@+id/stop"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginStart="50dp"
            android:onClick="stopAnimation"
            android:text="Stop" />
    </LinearLayout>
</LinearLayout>
```

> running.xml

```
<?xml version="1.0" encoding="utf-8"?>
<animation-list xmlns:android="http://schemas.android.com/apk/res/android">
    <item android:drawable="@drawable/one" android:duration="100" />
    <item android:drawable="@drawable/two" android:duration="100" />
    <item android:drawable="@drawable/three" android:duration="100" />
    <item android:drawable="@drawable/four" android:duration="100" />
</animation-list>
```

Output:




Running Devices
Pixel XL API 35
Module2
14 Akash Choudhary
START
STOP
27:5 LF UTF-8 4 spaces

5. Write a program to perform different operations on an image (i.e., rotation, expansion, shrinking, movement, and setting).

Aim: To implement an Android application that performs image operations such as rotation, expansion, shrinking, movement, and setting using animations

Objective:

- Rotate an image using a rotation animation.
- Expand an image using a scaling animation.
- Shrink an image using a scaling animation.
- Move an image using a translation animation.
- Set an image's position using a translation animation.

Theory:

The program performs various operations on an image using animations in Android. In MainActivity5, an ImageView is used to display the image on which different animations are applied. The AnimationUtils class is employed to load animation resources from XML files, which define the specific animation effects.

The XML files contain different animation types: rotate.xml applies a rotation effect, expand.xml scales the image to make it larger, shrink.xml scales the image down, move.xml translates the image horizontally, and set.xml translates the image from a starting position to a set position. The Animation class is used to start and stop these animations, allowing the user to interact with the image through buttons that trigger the animations. This approach demonstrates the use of Android's animation framework to create dynamic and engaging visual effects on images.

- rotate.xml: Defines a rotation animation that rotates the image from 0 to 360 degrees.
- expand.xml: Defines a scaling animation that enlarges the image.
- shrink.xml: Defines a scaling animation that reduces the size of the image.
- move.xml: Defines a translation animation that moves the image horizontally.
- set.xml: Defines a translation animation that sets the image's position from the right to its original position.

Animation: The base class for animations in Android. Different types of animations are loaded using AnimationUtils.

AnimationUtils: A utility class for loading animation resources defined in XML.

Code:

> MainActivity5.java

```
package com.example.module2;

import android.os.Bundle;
import android.view.View;
import android.view.animation.Animation;
import android.view.animation.AnimationUtils;
import android.widget.ImageView;

import androidx.appcompat.app.AppCompatActivity;

public class MainActivity5 extends AppCompatActivity {

   ImageView imageView;

   @Override
   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_main5);

      imageView = findViewById(R.id.imageView);
   }

   public void rotateImage(View view) {
                  Animation rotate = AnimationUtils.loadAnimation(getApplicationContext(),
R.anim.rotate);
      imageView.startAnimation(rotate);
   }

   public void expandImage(View view) {
                 Animation expand = AnimationUtils.loadAnimation(getApplicationContext(),
R.anim.expand);
      imageView.startAnimation(expand);
   }

   public void shrinkImage(View view) {
                  Animation shrink = AnimationUtils.loadAnimation(getApplicationContext(),
R.anim.shrink);
      imageView.startAnimation(shrink);
   }

   public void moveImage(View view) {
      Animation move = AnimationUtils.loadAnimation(getApplicationContext(), R.anim.move);
```

```
        imageView.startAnimation(move);
    }

    public void setImage(View view) {
        Animation set = AnimationUtils.loadAnimation(getApplicationContext(), R.anim.set);
        imageView.startAnimation(set);
    }
}
```

> activity_main5.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:padding="10dp"
    tools:context=".MainActivity5">

    <TextView
        android:id="@+id/textView"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:textSize="24dp"
        android:gravity="center"
        android:text="14 Akash Choudhary" />

    <ImageView
        android:id="@+id/imageView"
        android:layout_width="wrap_content"
        android:layout_height="375dp"
        android:scaleType="fitStart"
        android:layout_marginTop="5dp"
        app:srcCompat="@drawable/image1" />

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:padding="5dp">

        <Button
            android:id="@+id/rotate"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
```

```
            android:layout_marginStart="7.5dp"
            android:onClick="rotateImage"
            android:text="Rotate" />

        <Button
            android:id="@+id/expand"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginStart="45dp"
            android:onClick="expandImage"
            android:text="Expand" />

        <Button
            android:id="@+id/shrink"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginStart="45dp"
            android:onClick="shrinkImage"
            android:text="Shrink" />
    </LinearLayout>

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:padding="5dp">

        <Button
            android:id="@+id/move"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginStart="75dp"
            android:onClick="moveImage"
            android:text="Move" />

        <Button
            android:id="@+id/set"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginStart="50dp"
            android:onClick="setImage"
            android:text="Set" />
    </LinearLayout>
</LinearLayout>
```

> rotate.xml

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

```
<set xmlns:android="http://schemas.android.com/apk/res/android">
   <rotate
      android:duration="1000"
      android:fromDegrees="0"
      android:toDegrees="360"
      android:pivotX="50%"
      android:pivotY="50%" />
</set>
```

> expand.xml

```
<?xml version="1.0" encoding="utf-8"?>
<set xmlns:android="http://schemas.android.com/apk/res/android">
   <scale
      android:duration="1000"
      android:fromXScale="1.0"
      android:toXScale="1.5"
      android:fromYScale="1.0"
      android:toYScale="1.5"
      android:pivotX="50%"
      android:pivotY="50%" />
</set>
```

> shrink.xml

```
<?xml version="1.0" encoding="utf-8"?>
<set xmlns:android="http://schemas.android.com/apk/res/android">
   <scale
      android:duration="1000"
      android:startOffset="1000"
      android:fromXScale="1.5"
      android:toXScale="1.0"
      android:fromYScale="1.5"
      android:toYScale="1.0"
      android:pivotX="50%"
      android:pivotY="50%" />
</set>
```

> move.xml

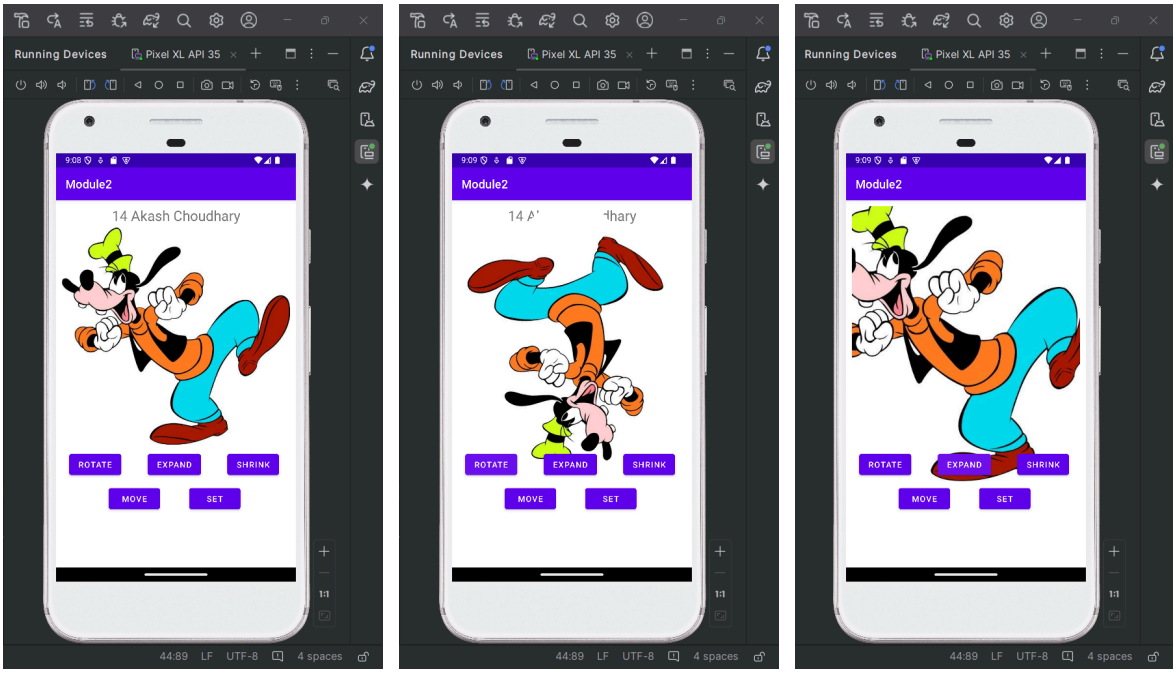
```
<?xml version="1.0" encoding="utf-8"?>
<set xmlns:android="http://schemas.android.com/apk/res/android">
   <translate
      android:duration="1000"
      android:startOffset="2000"
      android:fromXDelta="0%"
      android:toXDelta="100%"
      android:fromYDelta="0%"
      android:toYDelta="0%" />
```

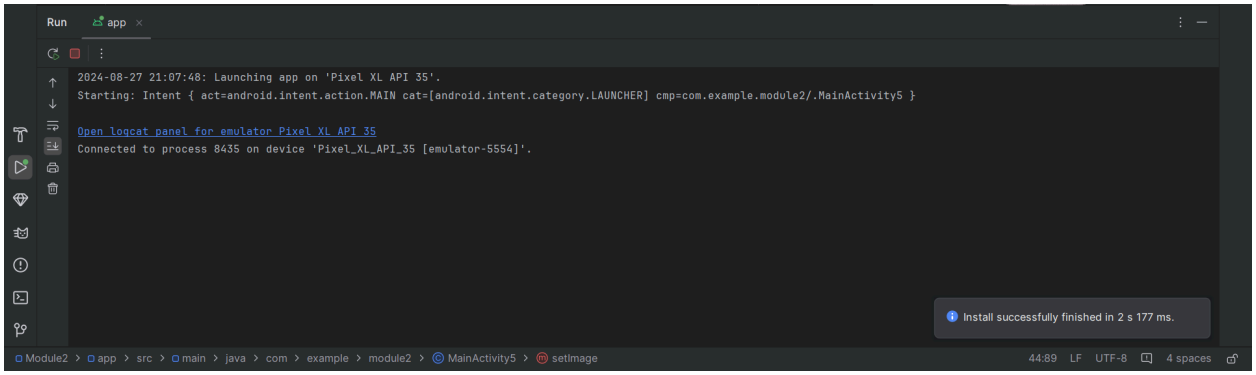
</set>

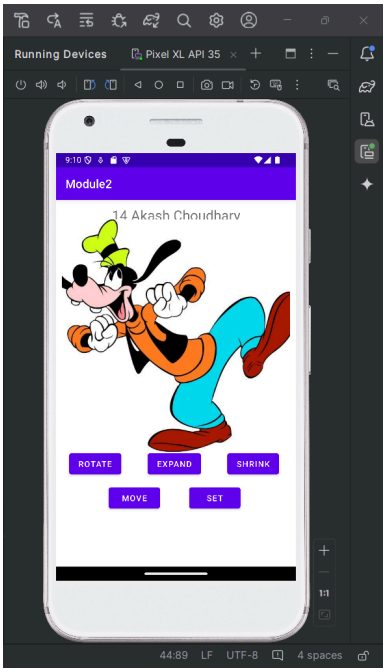
> set.xml

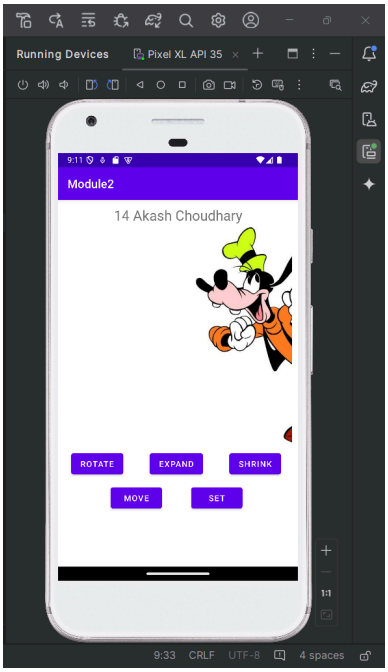
```
<?xml version="1.0" encoding="utf-8"?>
<set xmlns:android="http://schemas.android.com/apk/res/android">
   <translate
      android:duration="1000"
      android:startOffset="3000"
      android:fromXDelta="100%"
      android:toXDelta="0%"
      android:fromYDelta="0%"
      android:toYDelta="0%" />
</set>
```

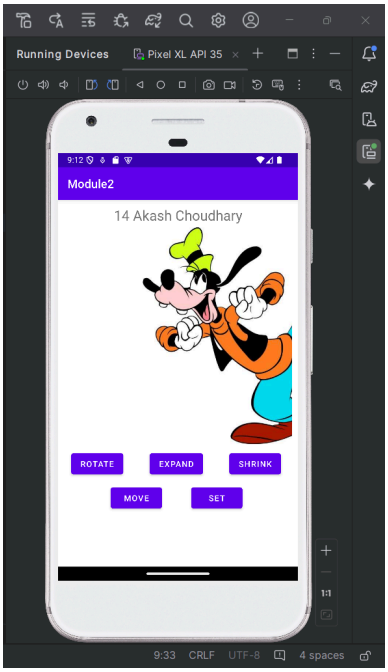
Output:






Running Devices
Pixel XL API 35
Module2
14 Akash Choudhary
ROTATE
EXPAND
SHRINK
MOVE
SET


Running Devices
Pixel XL API 35
Module2
14 Akash Choudhary
ROTATE
EXPAND
SHRINK
MOVE
SET


Running Devices
Pixel XL API 35
Module2
14 Akash Choudhary
ROTATE
EXPAND
SHRINK
MOVE
SET

6. Write a program to fetch the device's longitude and latitude.

Aim: To fetch and display the device's longitude and latitude coordinates

Objective:

- Request location permissions from the user.
- Use LocationManager to request location updates.
- Display the fetched coordinates in a TextView.

Theory:

The program utilizes the LocationManager and LocationListener classes to obtain the device's geographical coordinates. The LocationManager provides access to the system location services, while LocationListener is used to receive location updates. Permissions for accessing fine location data are requested at runtime using ActivityCompat and ContextCompat to ensure the app has the necessary access.

The onLocationChanged method of LocationListener is overridden to update the TextView with the device's latitude and longitude whenever the location changes. This approach demonstrates how to integrate location-based features into an Android application.

LocationManager: Manages access to the system location services. It provides methods to request location updates from various sources like GPS or network providers.

LocationListener: An interface for receiving location updates. It includes methods like onLocationChanged, which is called whenever the location changes.

Location: Represents a geographic location with latitude, longitude, altitude, and other attributes. It provides methods to retrieve these values.

ActivityCompat: A support class for managing permissions. It provides methods for requesting permissions and checking if the permissions are granted.

PackageManager: Provides information about the application packages available on the device, including their permissions. It is used to check if the app has the required permissions.

Code:

> MainActivity6.java

```
package com.example.module2;

import android.content.Context;
import android.content.pm.PackageManager;
import android.location.Location;
```

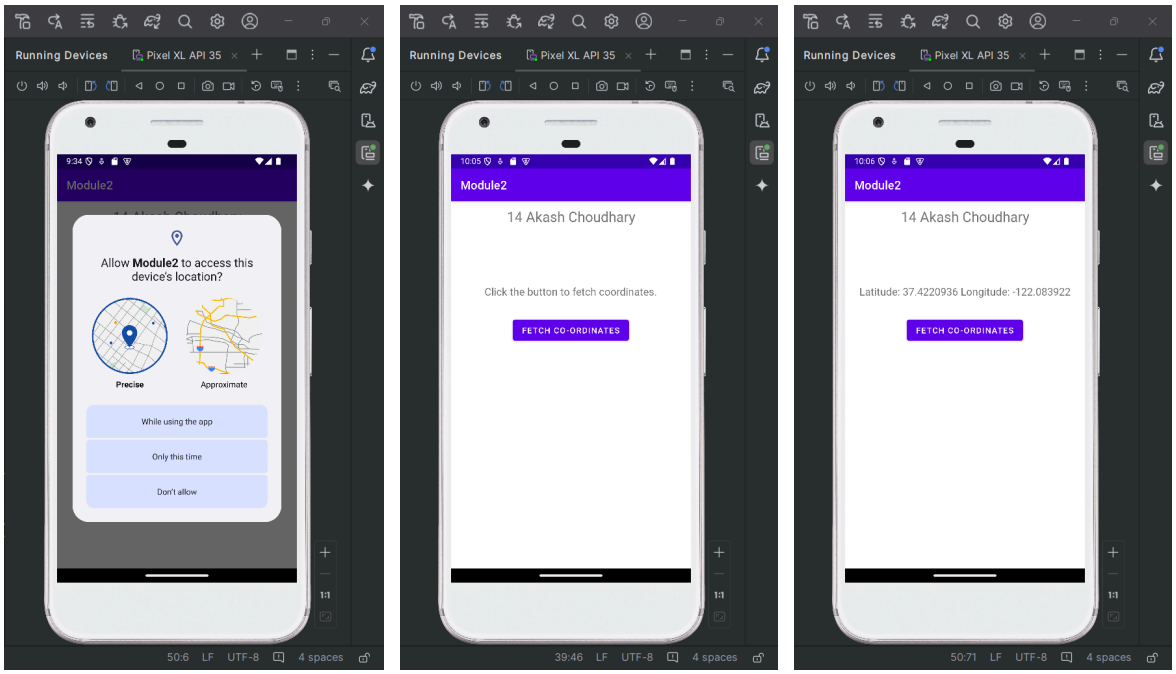
```
import android.location.LocationListener;
import android.location.LocationManager;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.TextView;

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

public class MainActivity6 extends AppCompatActivity implements LocationListener {

    TextView Coordinates;
    Button FetchCoordinates;
    LocationManager locationManager;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main6);

        Coordinates = findViewById(R.id.coordinates);
        FetchCoordinates = findViewById(R.id.fetchCoordinates);

                                        if(ContextCompat.checkSelfPermission(MainActivity6.this,
android.Manifest.permission.ACCESS_FINE_LOCATION)                                        !=
PackageManager.PERMISSION_GRANTED) {
            ActivityCompat.requestPermissions(MainActivity6.this, new String[]{
                                        android.Manifest.permission.ACCESS_FINE_LOCATION,
android.Manifest.permission.INTERNET
            }, 101);
        }
    }

    public void fetchCoordinates(View view) {
        try {
                                                    locationManager      =      (LocationManager)
getSystemService(Context.LOCATION_SERVICE);
            locationManager.requestLocationUpdates(LocationManager.NETWORK_PROVIDER,
500,5, MainActivity6.this);
        } catch (SecurityException se) {
            se.printStackTrace();
        }
    }
```

```
    @Override
    public void onLocationChanged(@NonNull Location location) {
                Coordinates.setText("Latitude: " + location.getLatitude() + " Longitude: " +
location.getLongitude());
    }
}
```

> activity_main6.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:padding="10dp"
    tools:context=".MainActivity6">

    <TextView
        android:id="@+id/textView"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:textSize="24dp"
        android:gravity="center"
        android:text="14 Akash Choudhary" />

    <TextView
        android:id="@+id/coordinates"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center_horizontal"
        android:textSize="18dp"
        android:layout_marginTop="100dp"
        android:text="Click the button to fetch coordinates." />

    <Button
        android:id="@+id/fetchCoordinates"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:onClick="fetchCoordinates"
        android:layout_marginTop="30dp"
        android:text="Fetch Co-ordinates" />
</LinearLayout>
```

Output:


Running Devices
Pixel XL API 35
Module2
Allow Module2 to access this device's location?
Precise
Approximate
While using the app
Only this time
Don't allow
14 Akash Choudhary
Click the button to fetch coordinates.
FETCH CO-ORDINATES
Latitude: 37.4220936 Longitude: -122.083922


Run
app
2024-08-27 22:04:05: Launching app on 'Pixel XL API 35'.
Starting: Intent { act=android.intent.action.MAIN cat=[android.intent.category.LAUNCHER] cmp=com.example.module2/.MainActivity6 }
Open logcat panel for emulator Pixel XL API 35
Connected to process 9771 on device 'Pixel_XL_API_35 [emulator-5554]'.
Install successfully finished in 2 s 527 ms.

7. Write a program to fetch the device's current location from longitude and latitude (reverse geocoding).

Aim: To fetch the device's current location and perform reverse geocoding to get the address based on latitude and longitude

Objective:

- Request location updates from the device's location service.
- Obtain the current latitude and longitude.
- Use reverse geocoding to convert latitude and longitude into a human-readable address.
- Display the latitude, longitude, and address on the user interface.

Theory:

In this program, the LocationManager class is utilized to access the system's location services and request periodic updates on the device's location using the network provider. The LocationListener interface is implemented in MainActivity7 to receive these location updates, allowing the application to respond when the device's location changes. The Geocoder class is used for reverse geocoding, which converts the latitude and longitude obtained from the location updates into a human-readable address.

The Address class represents this address and provides methods to retrieve detailed address components. The TextView widget displays the latitude, longitude, and address on the user interface, while the Button widget initiates the process of fetching and displaying this location information. ActivityCompat and ContextCompat are employed to handle permissions, ensuring that the application has the necessary access to location services.

LocationManager: Provides access to the system location services. Used here to request periodic updates on the device's location using the network provider.

LocationListener: Interface for receiving location updates. Implemented in MainActivity7 to handle changes in location and trigger actions accordingly.

Geocoder: A class used for converting latitude and longitude into a human-readable address. It communicates with the geocoding service to get address details from the provided coordinates.

Code:

> MainActivity7.java

```
package com.example.module2;
```

```
import android.content.Context;
import android.content.pm.PackageManager;
import android.location.Address;
import android.location.Geocoder;
import android.location.Location;
import android.location.LocationListener;
import android.location.LocationManager;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.widget.Button;
import android.widget.TextView;

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

import java.util.List;
import java.util.Locale;

public class MainActivity7 extends AppCompatActivity implements LocationListener {

    TextView Coordinates;
    Button FetchCoordinates;
    LocationManager locationManager;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main7);

        Coordinates = findViewById(R.id.coordinates);
        FetchCoordinates = findViewById(R.id.fetchCoordinates);

                                        if(ContextCompat.checkSelfPermission(MainActivity7.this,
android.Manifest.permission.ACCESS_FINE_LOCATION)                                       !=
PackageManager.PERMISSION_GRANTED) {
            ActivityCompat.requestPermissions(MainActivity7.this, new String[]{
                                            android.Manifest.permission.ACCESS_FINE_LOCATION,
android.Manifest.permission.INTERNET
            }, 101);
        }
    }
```

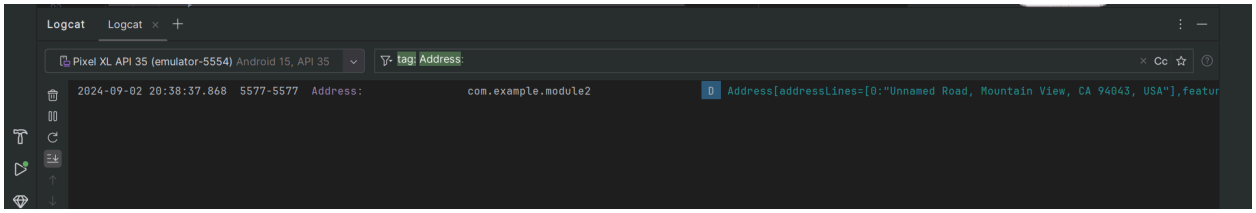
```
    public void fetchCoordinates(View view) {
        try {
                                                    locationManager = (LocationManager)
getSystemService(Context.LOCATION_SERVICE);
            locationManager.requestLocationUpdates(LocationManager.NETWORK_PROVIDER,
500, 5, MainActivity7.this);
        } catch (SecurityException se) {
            se.printStackTrace();
        }
    }

    private String getLocation(double latitude, double longitude) {
        String address = "";
        Geocoder geocoder = new Geocoder(MainActivity7.this, Locale.getDefault());
        try {
            List<Address> addresses = geocoder.getFromLocation(latitude, longitude, 1);
            if (addresses != null) {
                Address returnAddress = addresses.get(0);
                Log.d("Address:", String.valueOf(returnAddress));
                for (int i = 0; i <= returnAddress.getMaxAddressLineIndex(); i++) {
                    address += returnAddress.getAddressLine(i) + "\n";
                }
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
        return address;
    }

    @Override
    public void onLocationChanged(@NonNull Location location) {
        String address = getLocation(location.getLatitude(), location.getLongitude());
                Coordinates.setText("Latitude: " + location.getLatitude() + " Longitude: " +
location.getLongitude() + "\nAddress: " + address);
    }
}
```

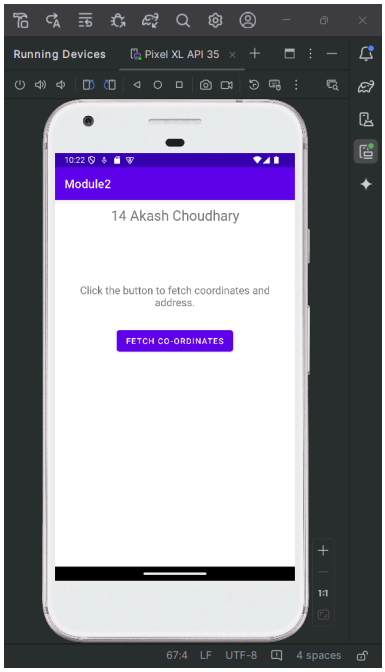
> activity_main7.xml

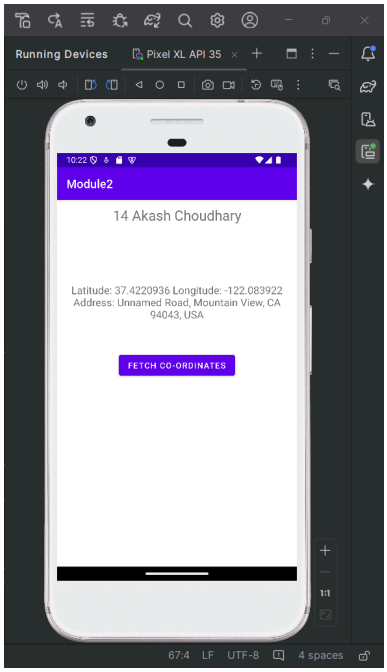
```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
```

```
    android:padding="10dp"
    tools:context=".MainActivity7">

    <TextView
        android:id="@+id/textView"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:textSize="24dp"
        android:gravity="center"
        android:text="14 Akash Choudhary" />

    <TextView
        android:id="@+id/coordinates"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:gravity="center"
        android:textSize="18dp"
        android:layout_marginTop="100dp"
        android:text="Click the button to fetch coordinates and address." />

    <Button
        android:id="@+id/fetchCoordinates"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:onClick="fetchCoordinates"
        android:layout_marginTop="30dp"
        android:text="Fetch Co-ordinates" />
</LinearLayout>
```

Output:




Running Devices
Pixel XL API 35
Module2
14 Akash Choudhary
Click the button to fetch coordinates and address.
FETCH CO-ORDINATES


Running Devices
Pixel XL API 35
Module2
14 Akash Choudhary
Latitude: 37.4220936 Longitude: -122.083922
Address: Unnamed Road, Mountain View, CA 94043, USA
FETCH CO-ORDINATES

8. Write a program to add a Google Map and mark your state/area on it.

Aim: To integrate Google Maps into an Android application and mark a specific location on the map

Objective:

- Add Google Maps to an Android application.
- Obtain and use a Google Maps API key.
- Display a map using SupportMapFragment.
- Add a marker to the map at a specified latitude and longitude.
- Move the camera to the marker's position.

Theory:

The program utilizes GoogleMap from the Google Maps API to display and interact with maps within the application. The SupportMapFragment is used to manage the map's lifecycle and provide the map interface. The OnMapReadyCallback interface ensures that the map is fully loaded and ready before performing any operations. The MarkerOptions class is employed to place a marker on the map at the specified geographic coordinates.

CameraUpdateFactory is used to move the camera to the marker's position, centering the view on it. The ActivityMapsBinding class is used for binding the XML layout to the Java code, providing access to the layout elements. The AndroidManifest.xml file includes necessary permissions and the Google Maps API key required for accessing the Google Maps services.

FragmentActivity: This class is a part of the AndroidX library and serves as a base class for activities that want to use fragments. It is used here as the base class for MapsActivity, enabling the use of fragments within the activity.

GoogleMap: This class is part of the Google Maps API and provides methods for manipulating and interacting with the map. It allows you to add markers, change camera positions, and perform various map-related operations.

OnMapReadyCallback: An interface provided by the Google Maps API that contains the callback method onMapReady(). This method is called when the map is fully initialized and ready to be used, allowing you to perform operations such as adding markers and moving the camera.

SupportMapFragment: A fragment provided by the Google Maps API that displays a map. It manages the map's lifecycle and integrates it into the activity's layout. The fragment is used to obtain a reference to the map and to receive map-related callbacks.

MarkerOptions: This class is used to create and configure markers on the map. It allows you to specify the position of the marker, its title, and other properties.

CameraUpdateFactory: A utility class provided by the Google Maps API that helps in creating CameraUpdate objects. These objects are used to change the map's camera position, such as moving to a specific location or zooming in.

ActivityMapsBinding: This is a generated class used for binding the XML layout file (activity_maps.xml) to the Java code in the activity. It provides access to the views defined in the layout file and ensures type safety when interacting with these views.

Code:

> AndroidMainfest.xml

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

    <meta-data
        android:name="com.google.android.geo.API_KEY"
        android:value="GOOGLE_MAP_API_KEY" />
```

> MapsActivity.java

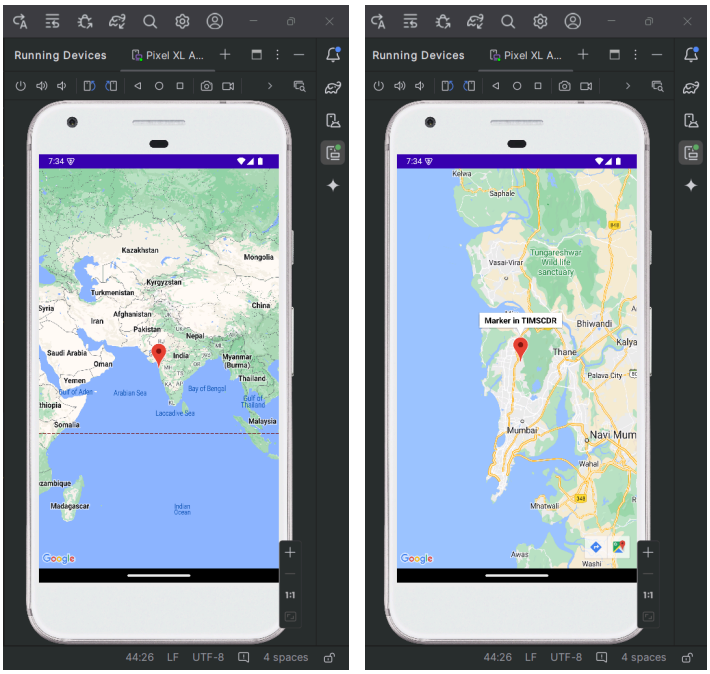
```
package com.example.module2;

import androidx.fragment.app.FragmentActivity;

import android.os.Bundle;

import com.google.android.gms.maps.CameraUpdateFactory;
import com.google.android.gms.maps.GoogleMap;
import com.google.android.gms.maps.OnMapReadyCallback;
import com.google.android.gms.maps.SupportMapFragment;
import com.google.android.gms.maps.model.LatLng;
import com.google.android.gms.maps.model.MarkerOptions;
import com.example.module2.databinding.ActivityMapsBinding;

public class MapsActivity extends FragmentActivity implements OnMapReadyCallback {

    private GoogleMap mMap;
    private ActivityMapsBinding binding;
```

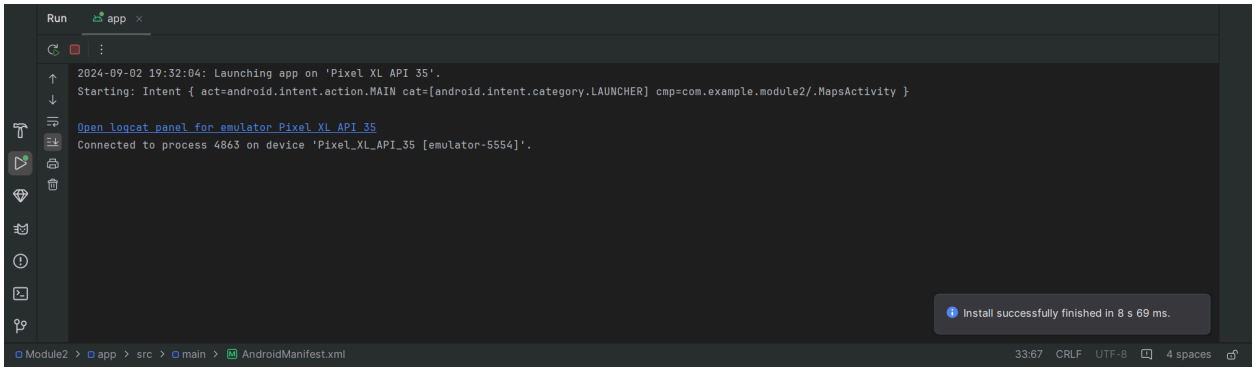
```
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        binding = ActivityMapsBinding.inflate(getLayoutInflater());
        setContentView(binding.getRoot());

        // Obtain the SupportMapFragment and get notified when the map is ready to be used.
                                SupportMapFragment mapFragment = (SupportMapFragment)
getSupportFragmentManager()
                .findFragmentById(R.id.map);
        mapFragment.getMapAsync(this);
    }

    /**
     * Manipulates the map once available.
     * This callback is triggered when the map is ready to be used.
     * This is where we can add markers or lines, add listeners or move the camera. In this case,
     * we just add a marker near Sydney, Australia.
     * If Google Play services is not installed on the device, the user will be prompted to install
     * it inside the SupportMapFragment. This method will only be triggered once the user has
     * installed Google Play services and returned to the app.
     */
    @Override
    public void onMapReady(GoogleMap googleMap) {
        mMap = googleMap;

        // Add a marker in Sydney and move the camera
        LatLng timscdr = new LatLng(19.206166115736004, 72.8738085241981);
        mMap.addMarker(new MarkerOptions().position(timscdr).title("Marker in TIMSCDR"));
        mMap.moveCamera(CameraUpdateFactory.newLatLng(timscdr));
    }
}
```

> activity_maps.xml

```
<?xml version="1.0" encoding="utf-8"?>
<fragment xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:map="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/map"
    android:name="com.google.android.gms.maps.SupportMapFragment"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MapsActivity" />
```

Output:


Running Devices
Pixel XL A...
Marker in TIMSCDR
44:26 LF UTF-8 4 spaces


Run
app
2024-09-02 19:32:04: Launching app on 'Pixel XL API 35'.
Starting: Intent { act=android.intent.action.MAIN cat=[android.intent.category.LAUNCHER] cmp=com.example.module2/.MapsActivity }
Open logcat panel for emulator Pixel XL API 35
Connected to process 4863 on device 'Pixel_XL_API_35 [emulator-5554]'.
Install successfully finished in 8 s 69 ms.
Module2 > app > src > main > AndroidManifest.xml
33:67 CRLF UTF-8 4 spaces