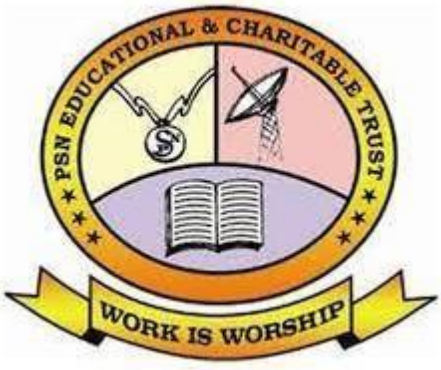
PSN ENGINEERING COLLEGE

DEPARTMENT OF INFORMATION TECHNOLOGY



LAB MANUAL

Mr.K.PUNGARAJAN,M.E.,

Assistant Professor/CSE

IT3681– MOBILE APPLICATIONS DEVELOPMENT LABORATORY

Year/ Semester : III/ 06

2023-2024

LIST OF EXPERIMENTS:

- Study and installation of Flutter/Kotlin multi-platform environment
- Develop an application that uses Widgets, GUI components, Fonts, and Colors.
- Develop a native calculator application.
- Develop a gaming application that uses 2-D animations and gestures.
- Develop a movie rating application (similar to IMDB)
- Develop an application to connect to a web service and to retrieve data with HTTP.
- Develop a simple shopping application.
- Design a web server supporting push notifications.
- Develop an application by integrating Google maps
- Mini Projects involving Flutter/Kotlin multi-platform

EX.NO: 1	STUDY AND INSTALLATION OF FLUTTER/KOTLIN MULTI-PLATFORM ENVIRONMENT

AIM:

To Study and installation of Flutter/Kotlin multi-platform environment.

PROCEDURES:

Get the Flutter SDK

Step 1: Download Flutter SDK:

Download the following installation bundle to get the latest stable release of the Flutter SDK

Step 2: Extract the File: Extract the downloaded zip file and move it to the desired location where you want to install Flutter SDK.

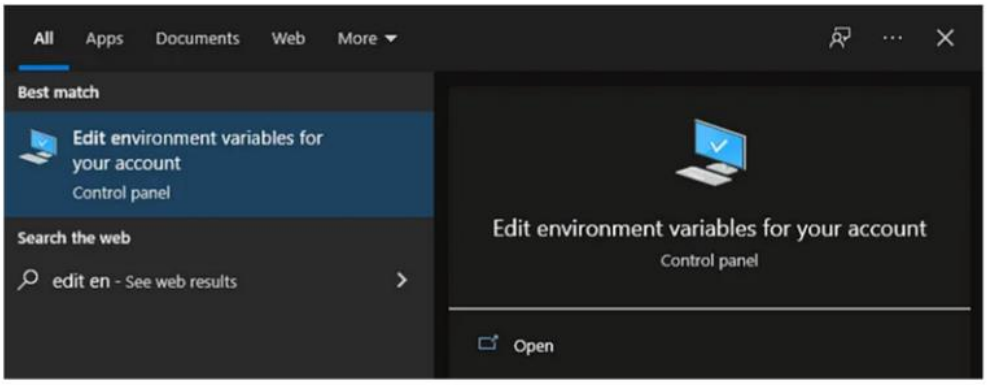
Do not install it in a folder or directory that requires elevated privileges, (such as *C:\Program Files*\) to ensure the program runs properly. For this tutorial, it will be stored in *C:\development\flutter*.

You are now ready to run Flutter commands in the Flutter Console.

Step 3: Update Path Variable for Windows PowerShell

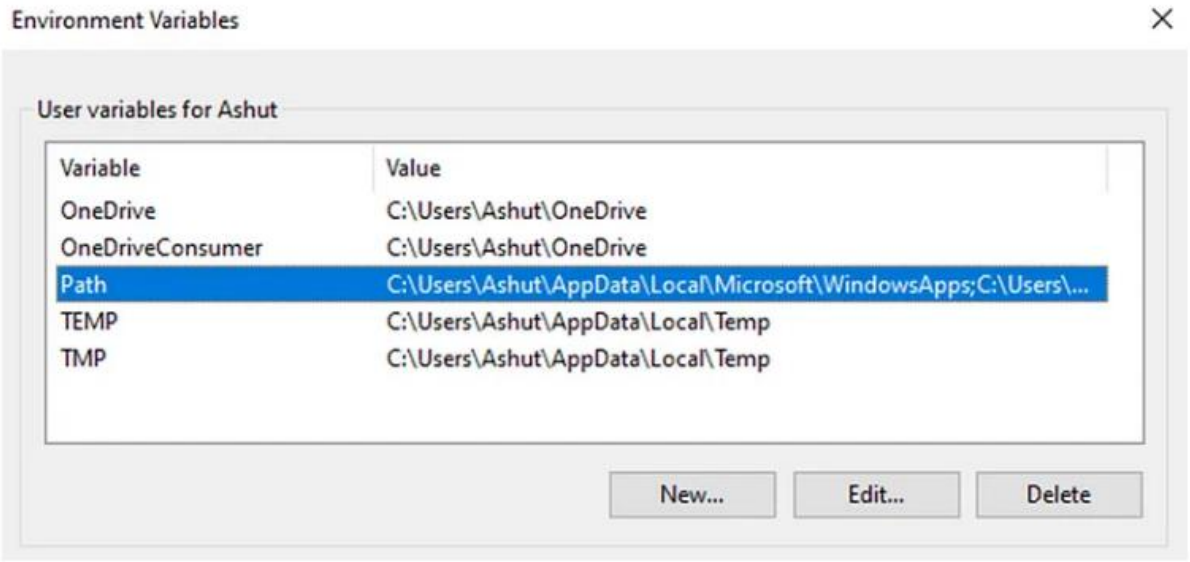
If you wish to run Flutter commands in the regular Windows console, take these steps to add Flutter to the PATH environment variable:

- From the Start search bar, enter 'env' and select **Edit environment variables for your account**.

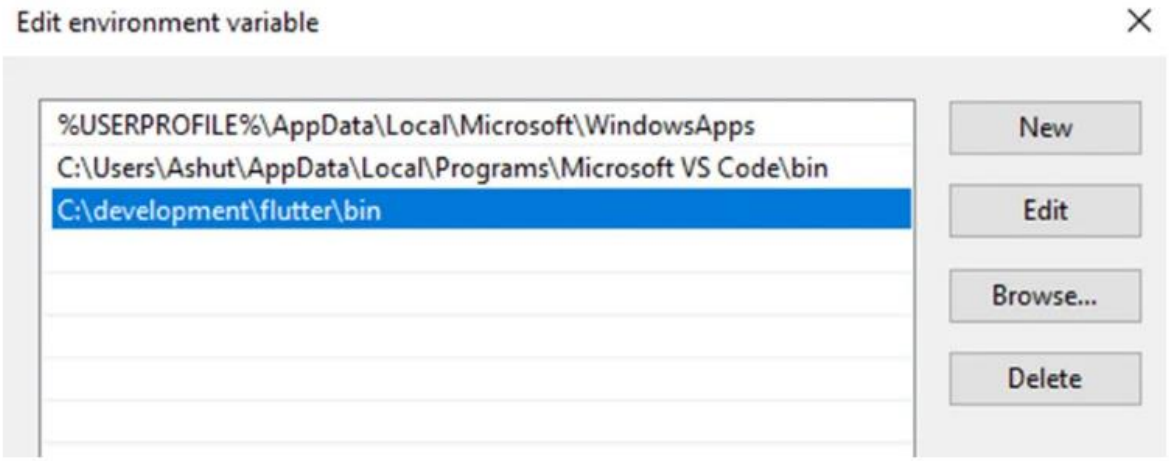


- Under **User variables** check if there is an entry called **Path**:

- If the entry exists, append the full path to flutter\bin using; as a separator from existing values.



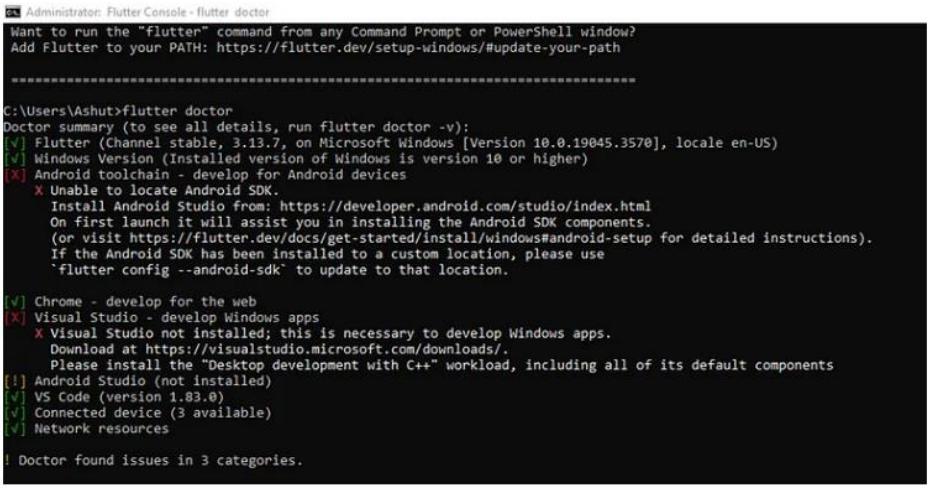
  - On the next screen, click **New** and add the full path to your *flutter\bin* directory. For this guide, it is shown below.
  - Click OK on both windows to enable running Flutter commands in Windows consoles.



- If the entry doesn't exist, create a new user variable named Path with the full path to flutter\bin as its value.

Step 4: Confirm Installed Tools for Running Flutter

In CMD, run the *flutter doctor* command to confirm the installed tools along with brief descriptions.



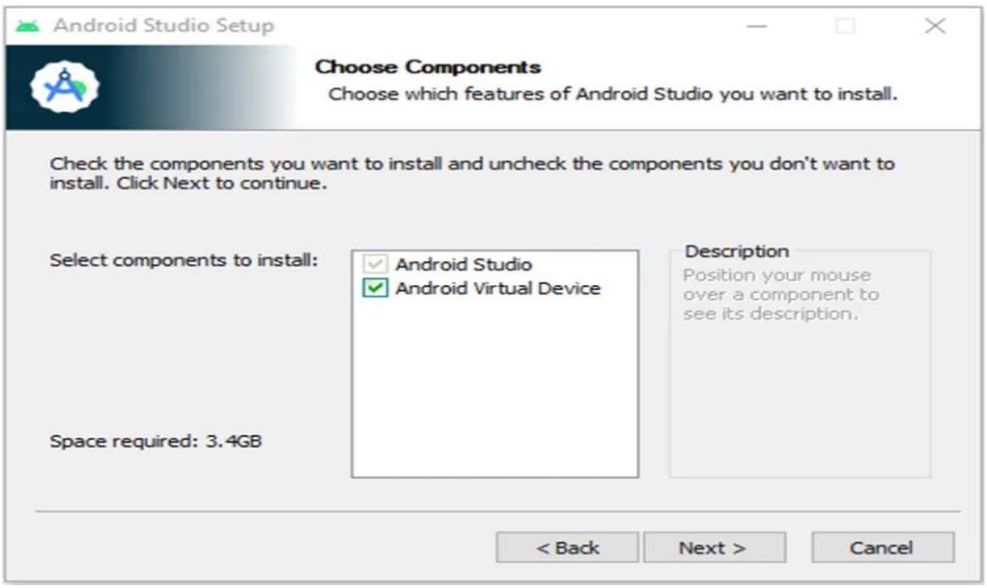
As visible, several components still need to be installed to complete the installation.

Step 5: Download and Install Android Studio

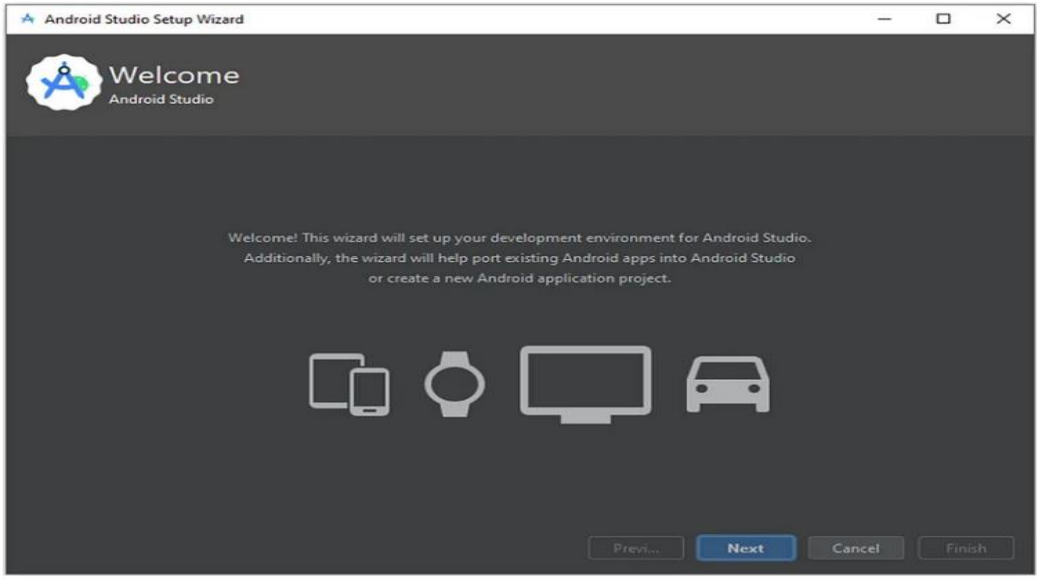
Download Android Studio:

- Visit the official Android Studio download page at https://developer.android.com/studio.
- Click on the “Download Android Studio” button.

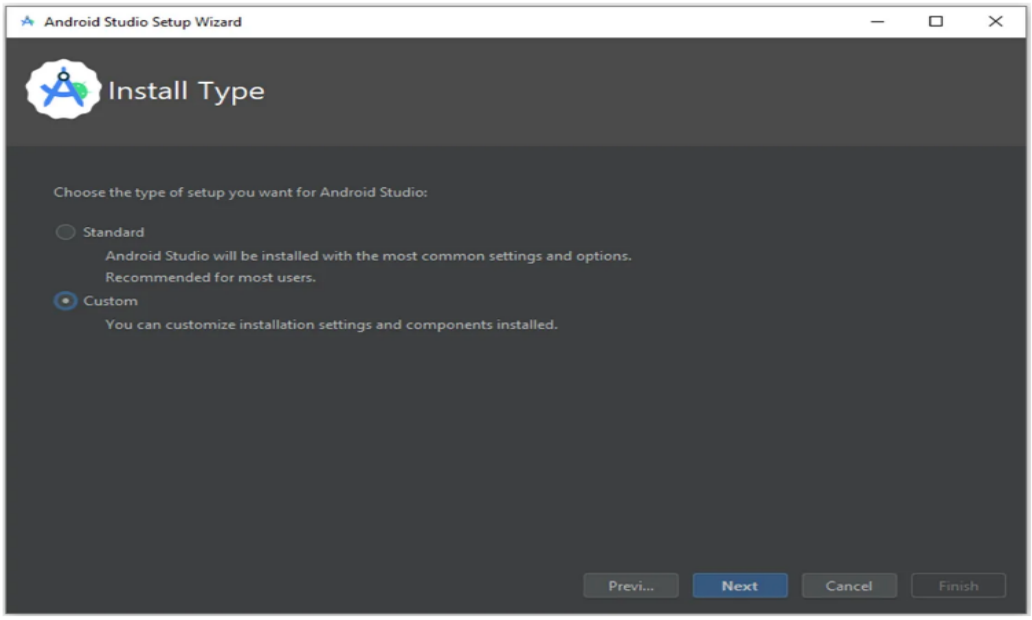
Next, proceed by downloading Android Studio. During the setup, unless you have unique requirements, simply click “Next” on all screens to keep the default settings. On the “Choose Components” screen, be sure to select the “Android Virtual Device” option to enable an Android emulator for your app development needs.



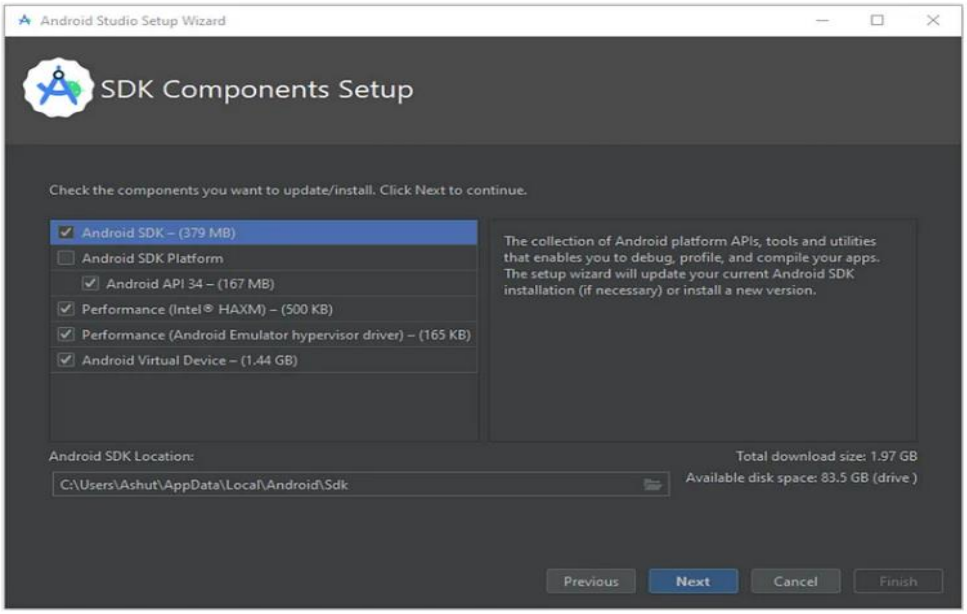
Afterward, The Android Studio Setup Wizard will start and you can proceed by clicking **Next**.



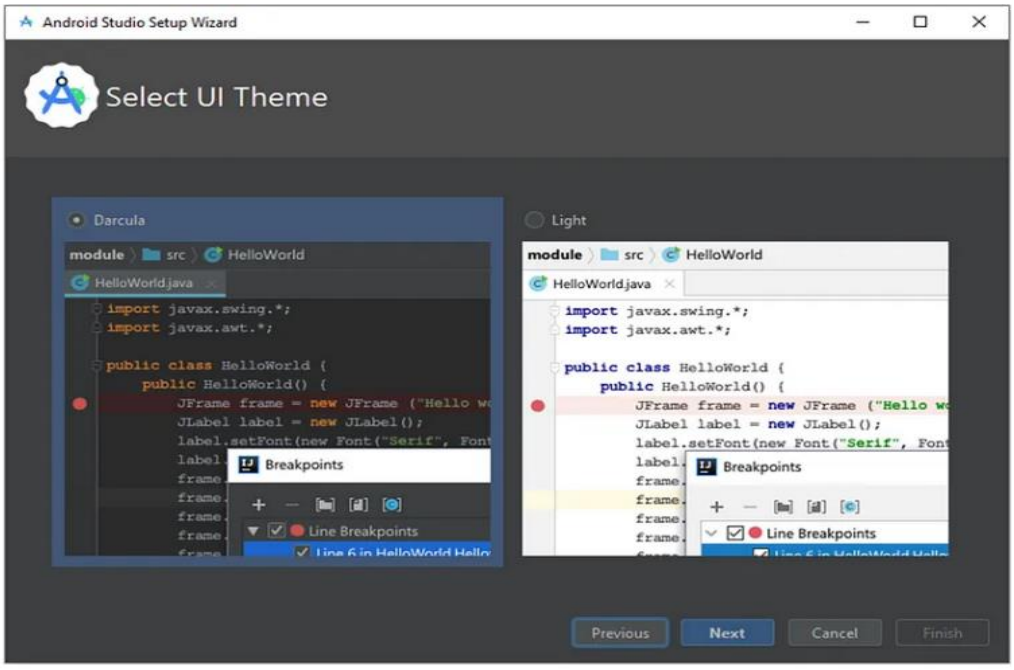
On the Install Type screen, select Custom and click **Next.**

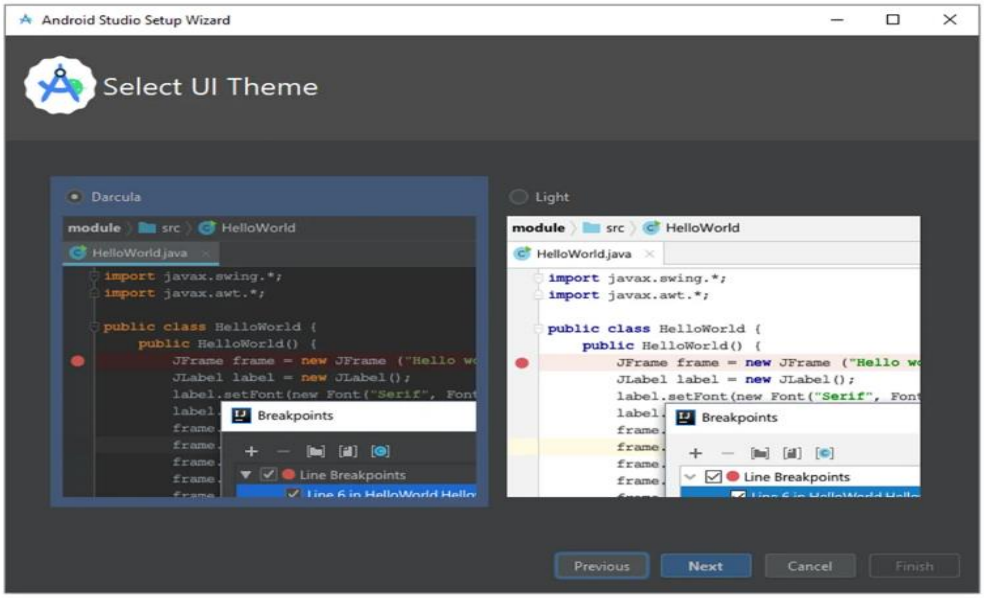


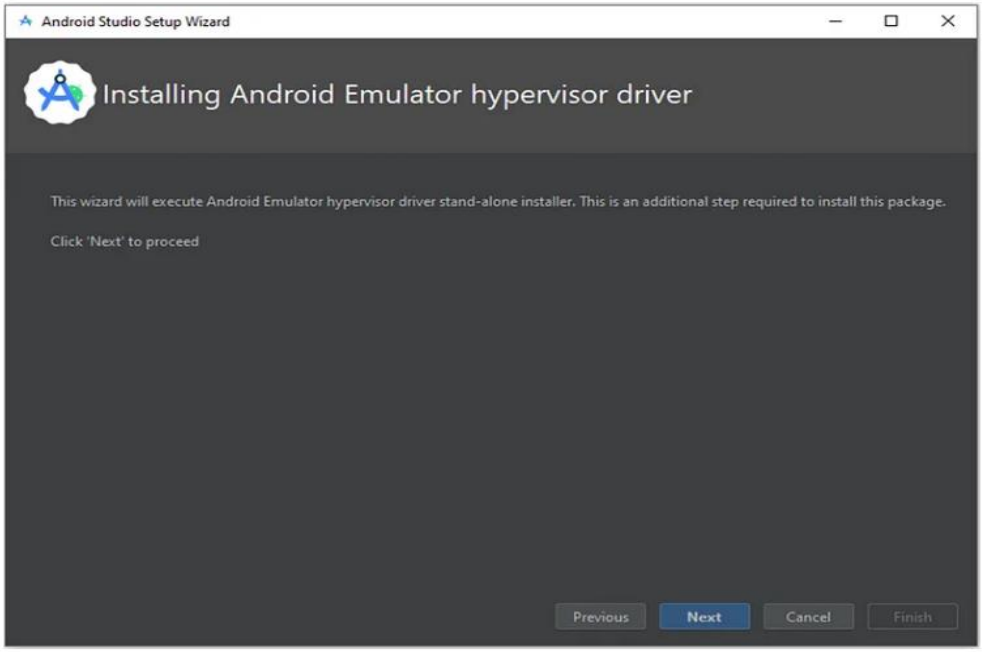
Select the installation location or leave the default path and click **Next**.

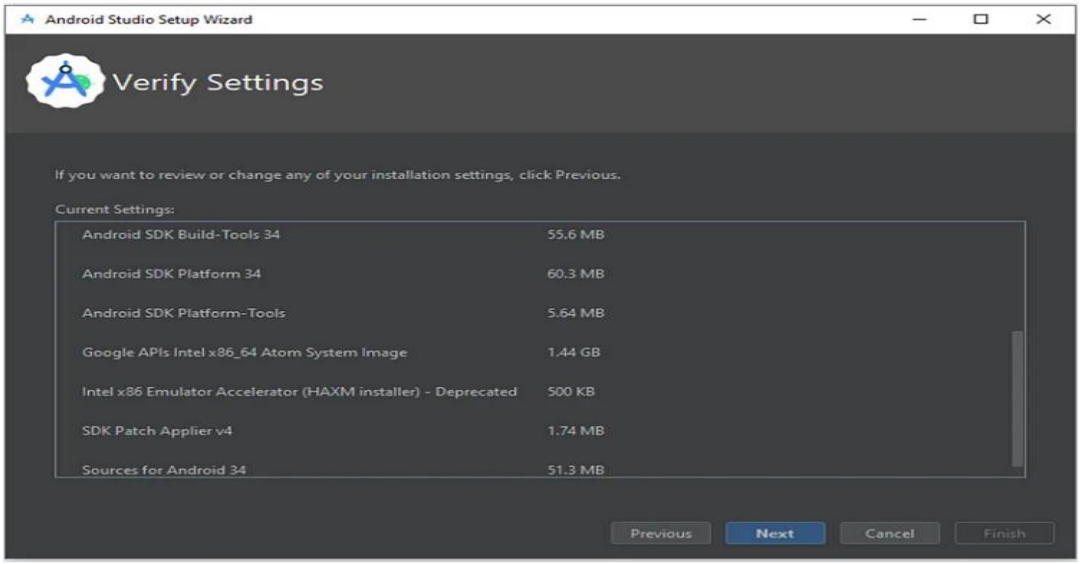


Select your UI theme and click **Next**.

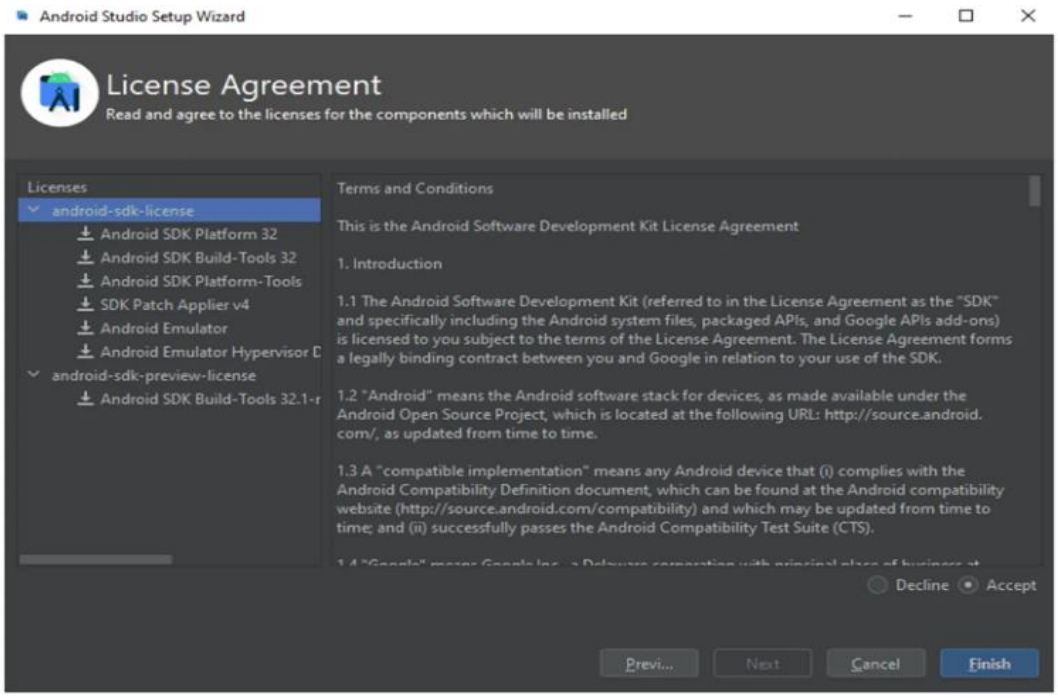



Android Studio Setup Wizard
Select UI Theme
Darcula
Light
module
src
HelloWorld
HelloWorld.java
Breakpoints
Line Breakpoints
Previous
Next
Cancel
Finish

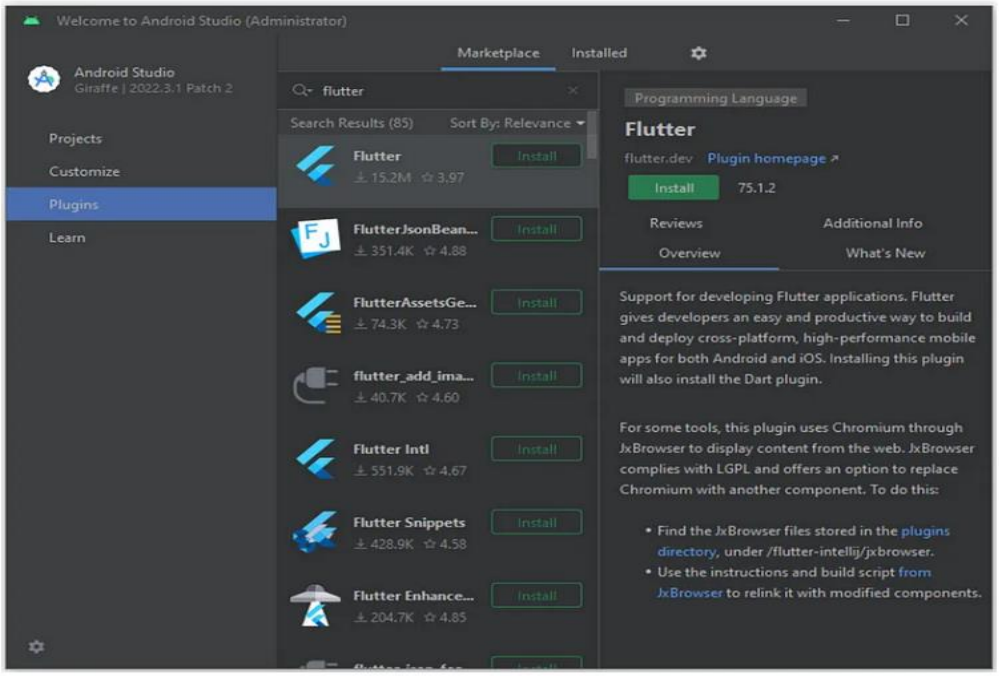

Android Studio Setup Wizard
Installing Android Emulator hypervisor driver
This wizard will execute Android Emulator hypervisor driver stand-alone installer. This is an additional step required to install this package.
Click 'Next' to proceed
Previous
Next
Cancel
Finish



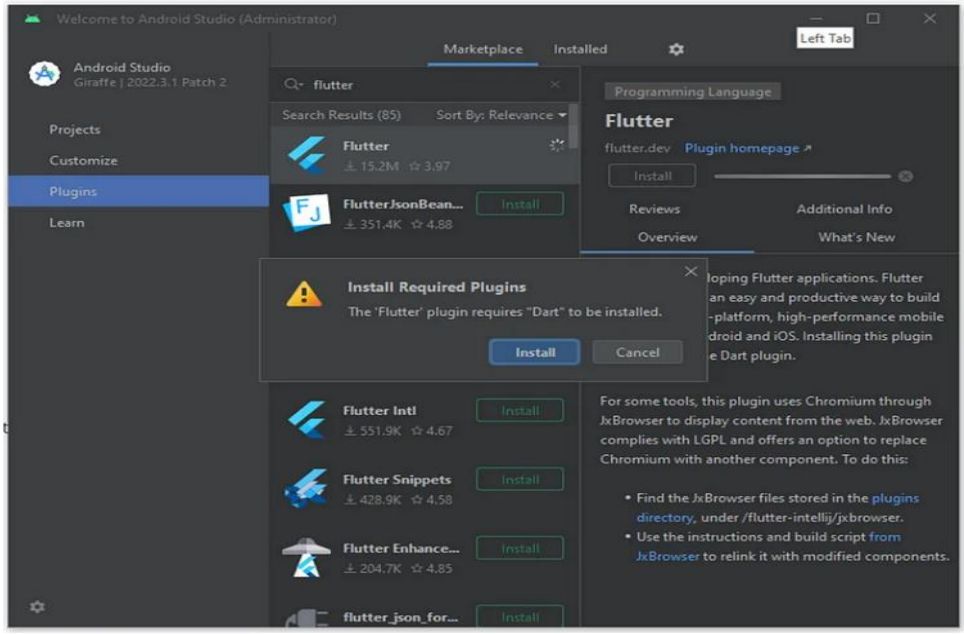
On the next screen, accept the License Agreement and click **Finish**.



- The download of the components will start and Android Studio install. Once completed, click **Finish**.
- After the installation, start Android Studio. On the left side, click **Plugins**. Search for Flutter and click **Install** to install the Flutter plugin.



It will also prompt you to install Dart, a programming language used to create Flutter apps. Click **Install** at the prompt.



Finally, click **Restart IDE** so that the plugin changes are applied. Click **Restart** at the prompt to confirm this action.

Afterward, run the *flutter doctor* command in CMD to confirm the Android Studio installation.

```
C:\Users\blup>flutter doctor

Doctor summary (to see all details, run flutter doctor -v):

[√] Flutter (Channel stable, 2.10.4, on Microsoft Windows [version 10.0.19041.746), locale en-US)

[!] Android toolchain - develop for Android devices (Android SDK version 32.1.0-rc1)

 ! Some Android licenses not accepted. To resolve this, run: flutter doctor --android-licenses

[√] Chrome - develop for the web

[X] Visual Studio - develop for Windows

  X Visual Studio not installed; this is necessary for Windows development

 Download at https://visualstudio.microsoft.com/downloads/

  Please install the "Desktop development with C++- workload, including all of its default components

[√] Android Studio (version 2021.1)

[√] Connected device (2 available)

[√] HTTP Host Availability
! Doctor found issues in 2 categories
```

Android Studio was successfully installed; however, it found an issue with Android licenses. This issue is fairly common and is mitigated by running the following command in CMD.

```
flutter doctor --android-licenses
```

When asked, input **y** to all prompts, to accept licenses.

```
C:\Users\blup>flutter doctor --android-licenses
5 of 7 SDK package licenses not accepted. 100% Computing updates...
Review licenses that have not been accepted (y/N)? y
```

Running the *Flutter Doctor* command again shows the issue resolved.

```
C:\Users\blup>flutter doctor
Doctor summary (to see all details, run flutter doctor -v):
[√] Flutter (Channel stable, 2.10.4, on Microsoft Windows [Version 10.0.19041.746], locale en-US)
[√] Android toolchain - develop for Android devices (Android SDK version 32.1.0-rc1)
[√] Chrome - develop for the web
[X] Visual Studio - develop for Windows
    X Visual Studio not installed; this is necessary for Windows development.
      Download at https://visualstudio.microsoft.com/downloads/.
      Please install the "Desktop development with C++" workload, including all of its default
components
[√] Android Studio (version 2021.1)
[√] Connected device (2 available)
[√] HTTP Host Availability

! Doctor found issues in 1 category
```

Step 6: Install Visual Studio (Optional)

- ✓ The above output also shows that Visual Studio is not installed. Visual Studio is not needed unless you want to use Flutter for Windows desktop development.
- ✓ If you need to use it, you can download Microsoft's Visual Studio 2022 with C++. Once the *VisualStudioSetup.exe* file is downloaded, open it and proceed with the installation by agreeing to all default installation options. This installation requires at least 20 GB of free disk space. After the installation completes, run the *flutter doctor* command in CMD to confirm the Visual Studio installation.

```
C:\Users\blup>flutter doctor
Doctor summary (to see all details, run flutter doctor -v):
[√] Flutter (Channel stable, 2.10.4, on Microsoft Windows [Version 10.0.19041.746], locale en-US)
[√] Android toolchain - develop for Android devices (Android SDK version 32.1.0-rc1)
[√] Chrome - develop for the web
[√] Visual Studio - develop for Windows (Visual Studio Community 2022 17.1.3)
[√] Android Studio (version 2021.1)
[√] Connected device (2 available)
[√] HTTP Host Availability
```

• No issues found!

- ✓ At this point, all the tools for flutter projects are ready to be used for the development of flutter apps. Depending on your needs, you can start your projects in android studio or visual studio.

RESULT:

To Study and installation of Flutter/Kotlin multi-platform environment installed successfully.

EX.NO : 2	**TO DEVELOP A SIMPLE ANDROID APPLICATION THAT USES GUI COMPONENTS, FONT AND COLORS.**

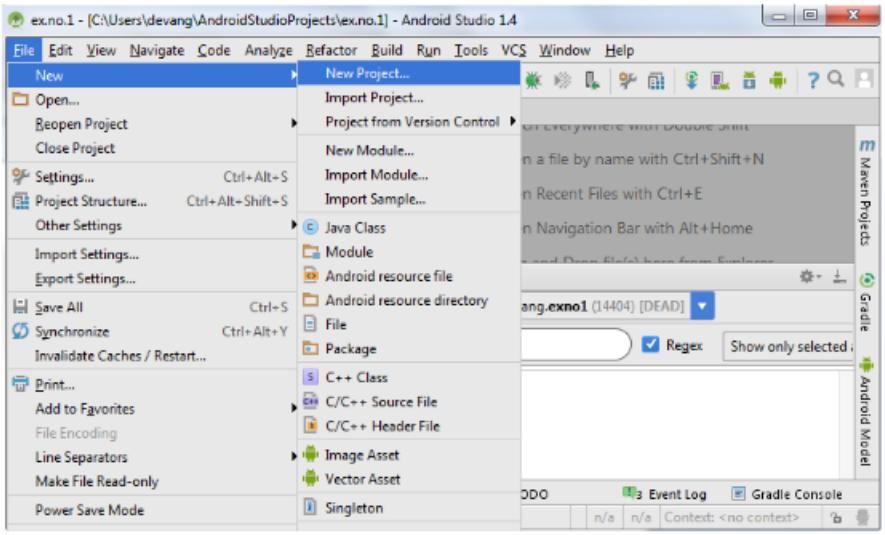
AIM:

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

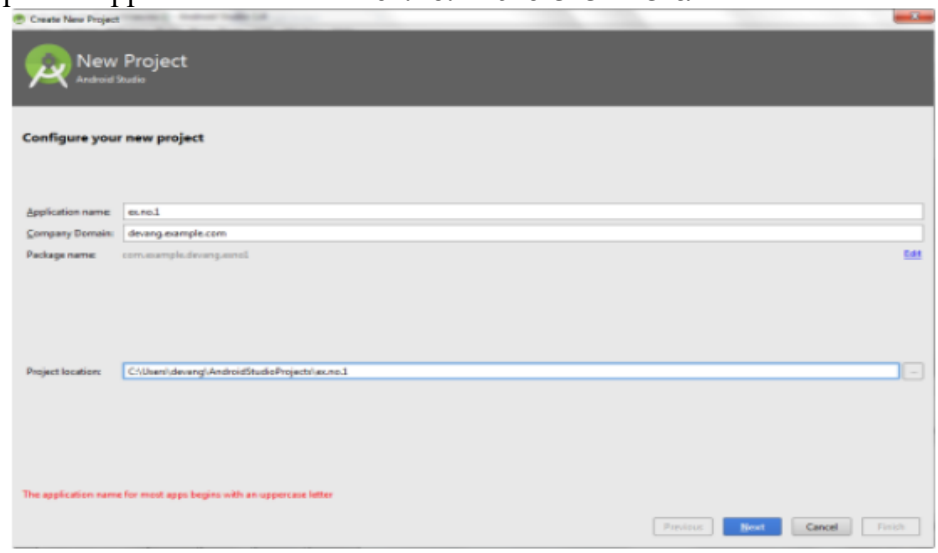
PROCEDURE:

CREATING A NEW PROJECT:

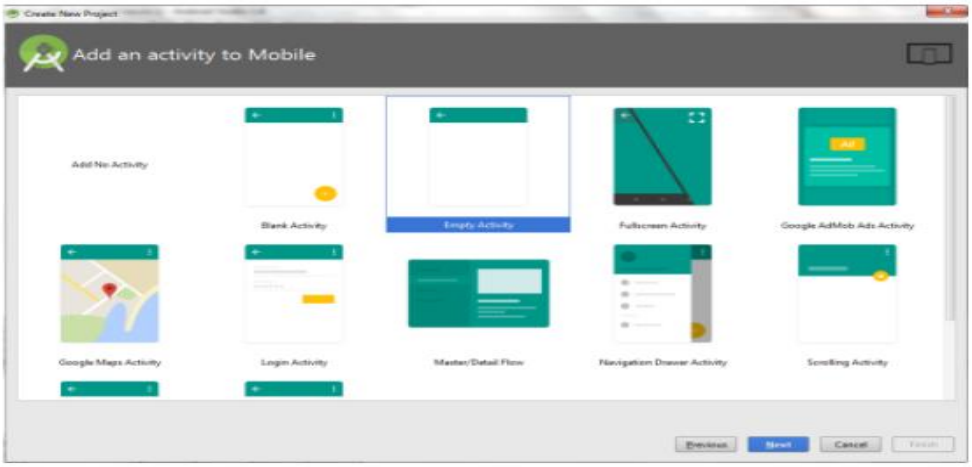
- Open Android Stdio and then click on **File -> New -> New project.**



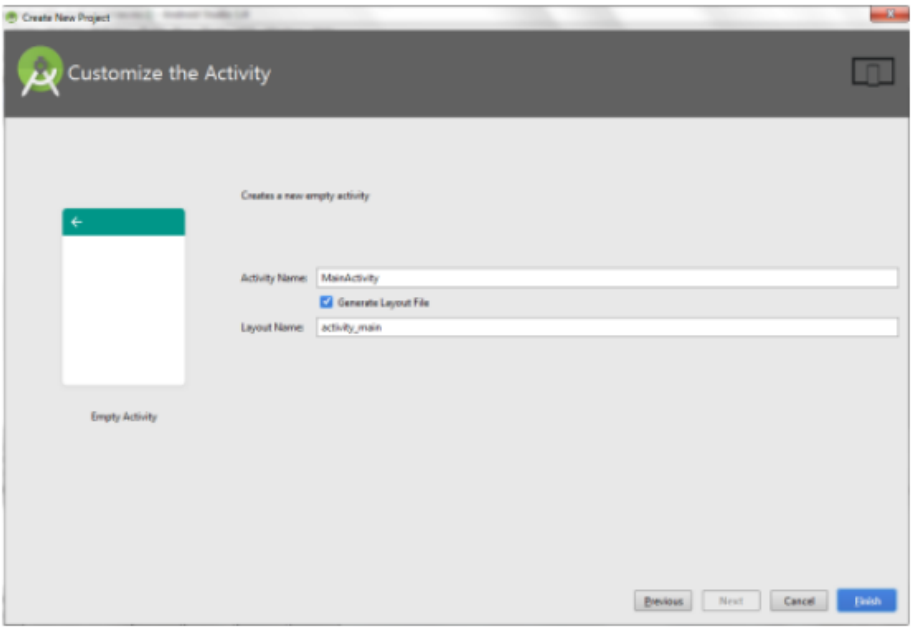
- Then type the Application name as "**ex.no.1"** and click **Next.**



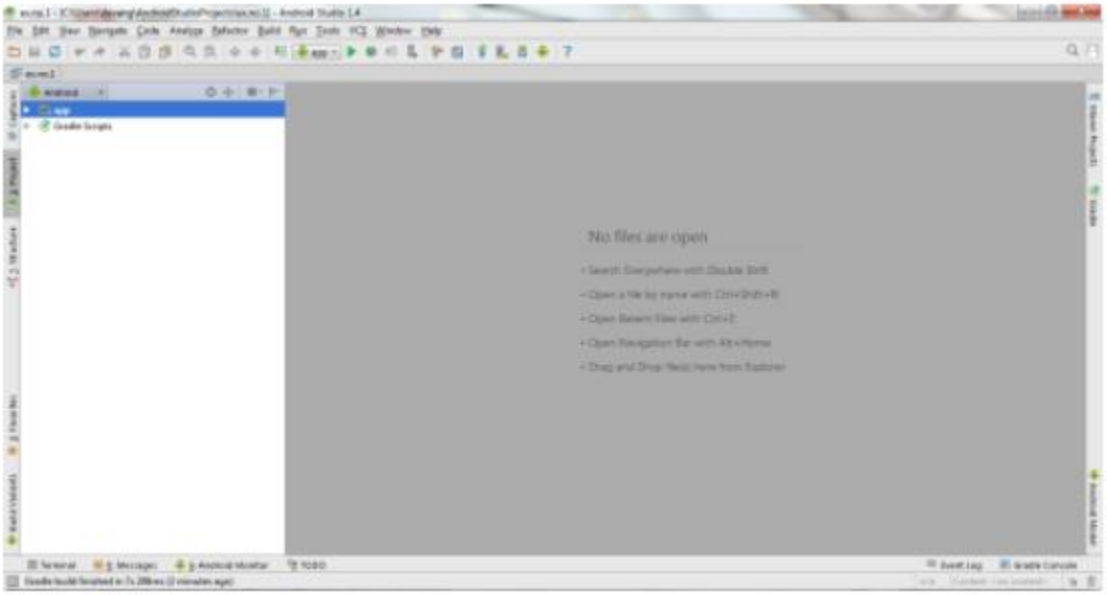
- Then select the **Minimum SDK** as shown below and click **Next**.



- Finally click F**inish**.

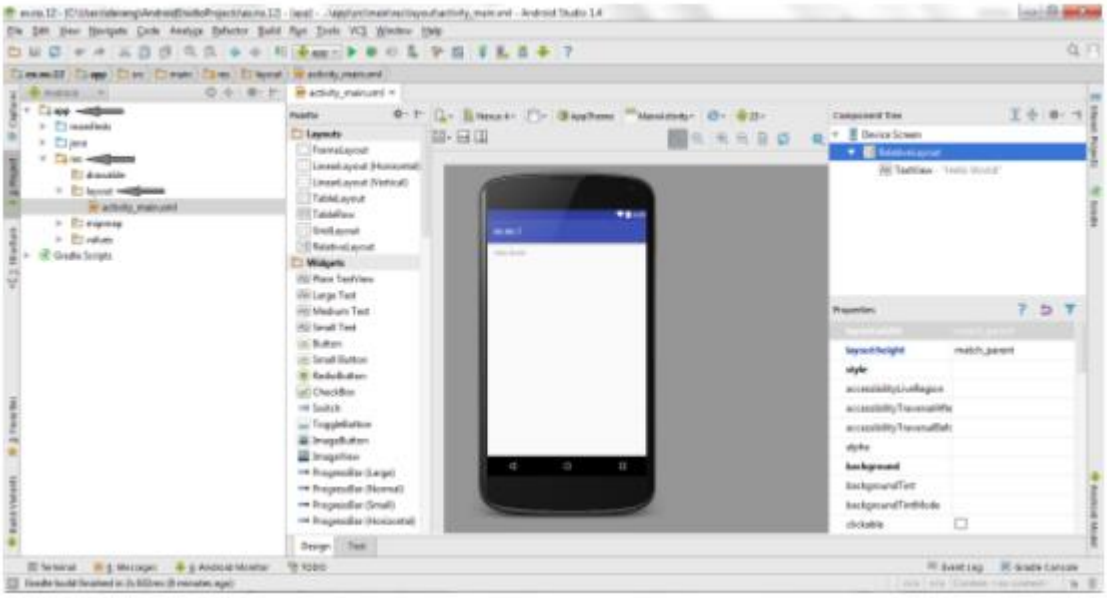


- It will take some time to build and load the project.
- After completion it will look as given below.

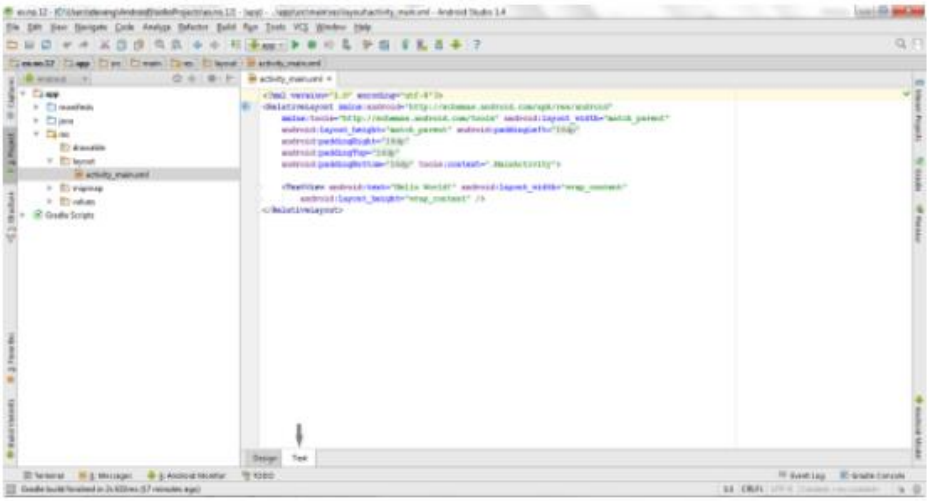


Designing layout for the Android Application:

- 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.

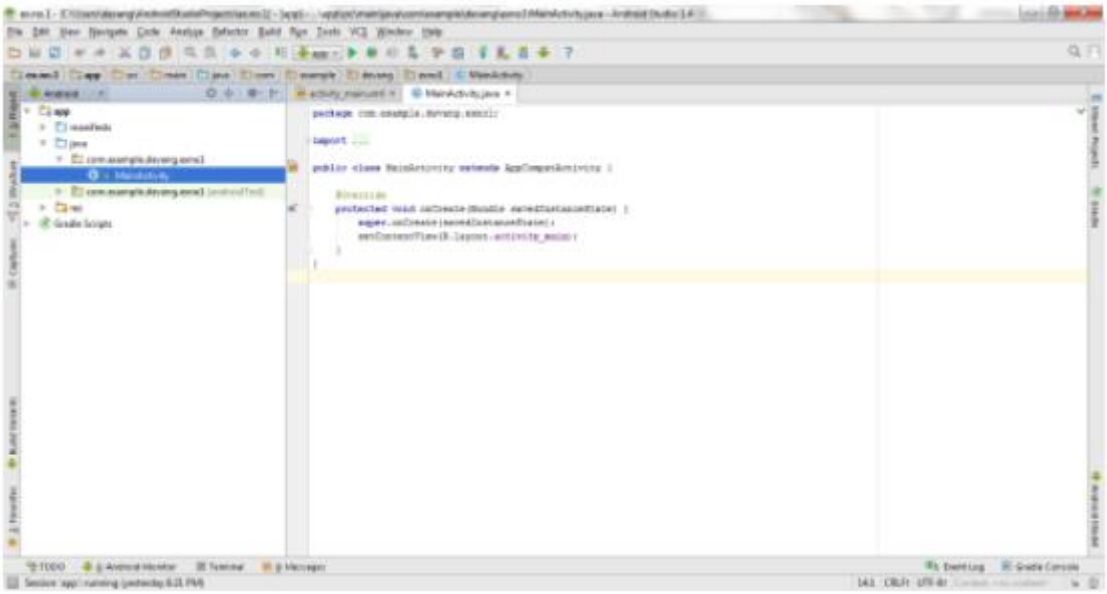
Code for Activity_main.xml:

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

- Now click on Design and your application will look as given below.

- Click on **app -> java -> com.example.exno1 -> MainActivity.**



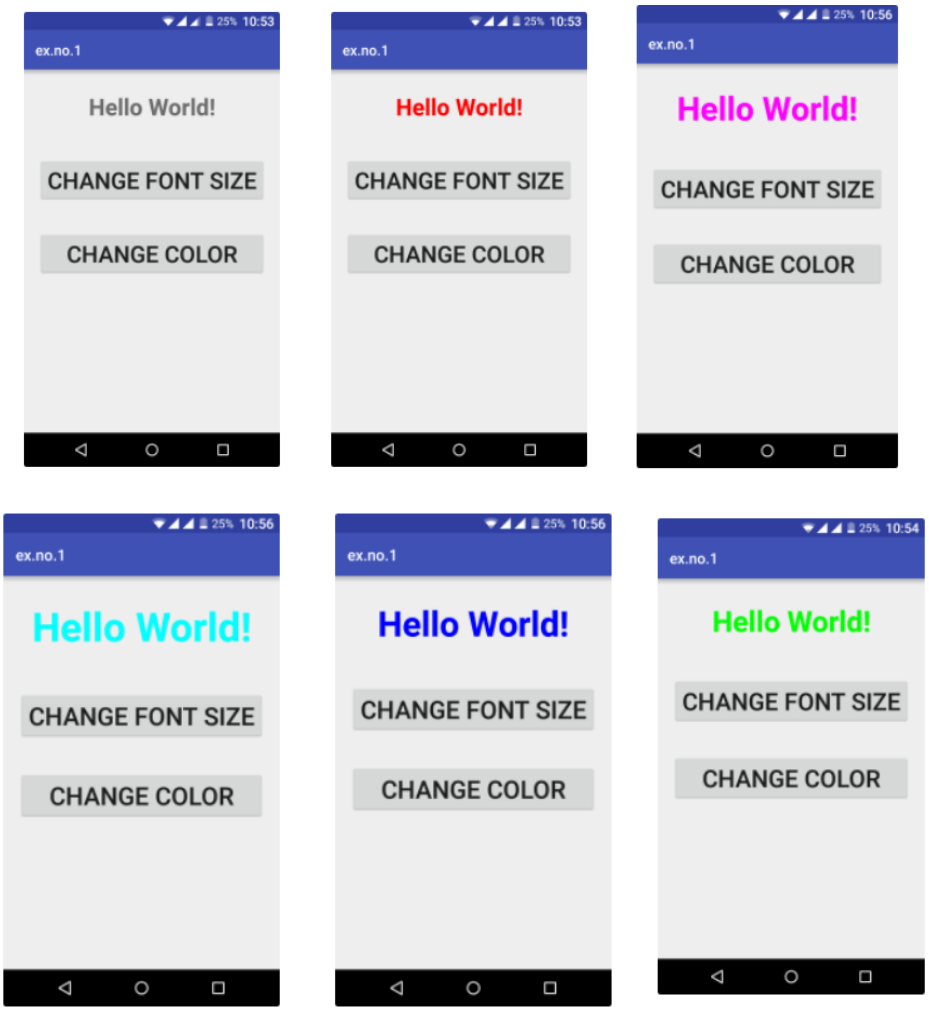
- Then delete the code which is there and type the code as given below.

```
package com.example.exno1;
 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;
        }
    });
  }
}
```

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

OUTPUT:



RESULT:

Thus a Simple Android Application that uses GUI components, Font and Colors is developed and executed successfully

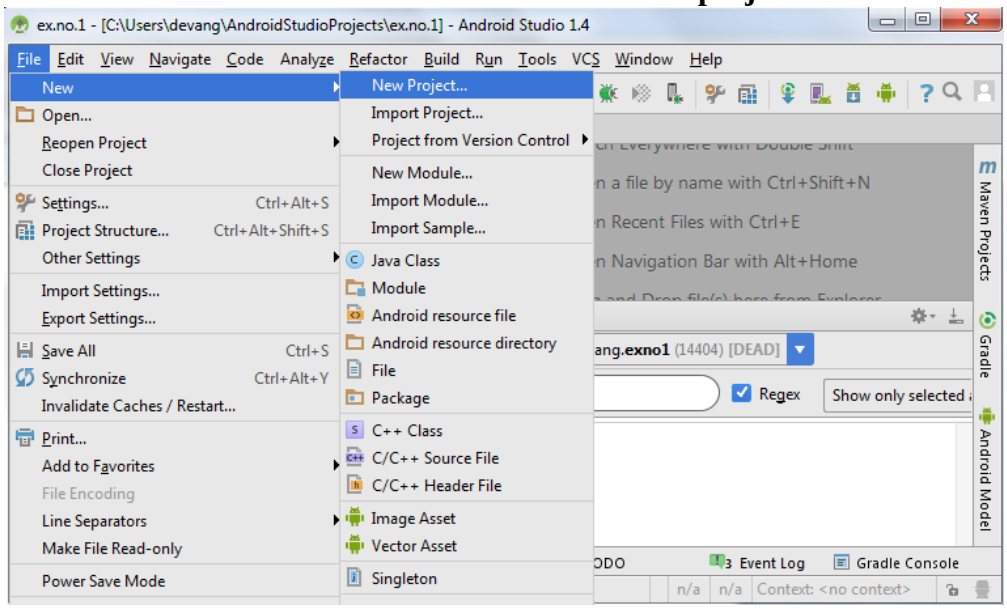
EX.NO:3	
	DEVELOP A NATIVE CALCULATOR APPLICATION.

Aim:

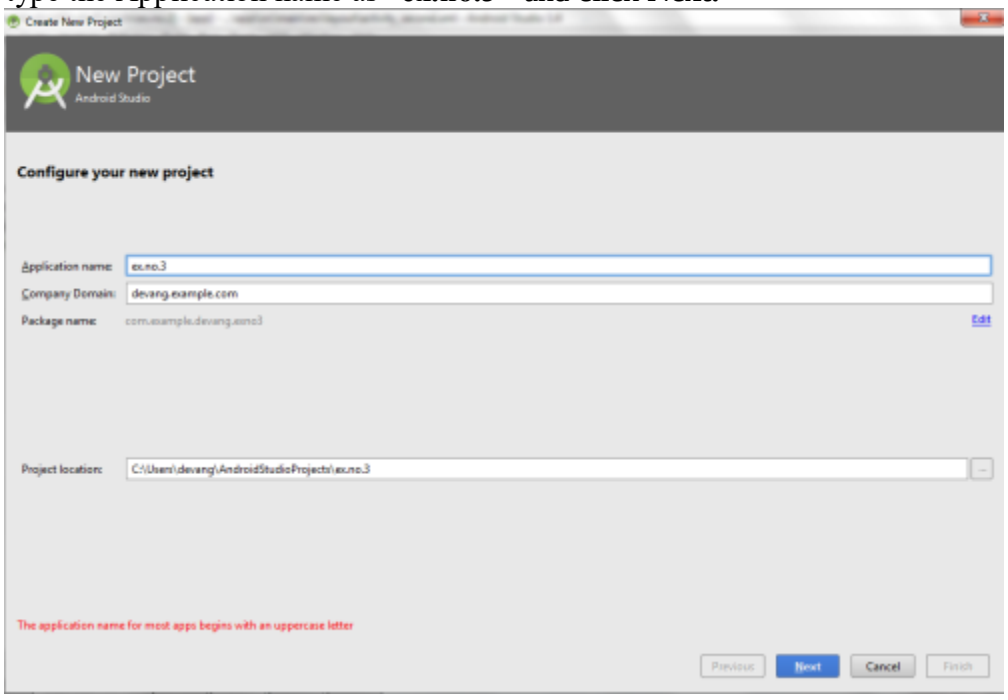
To develop a Simple Android Application for Native Calculator.

Creating a New project:

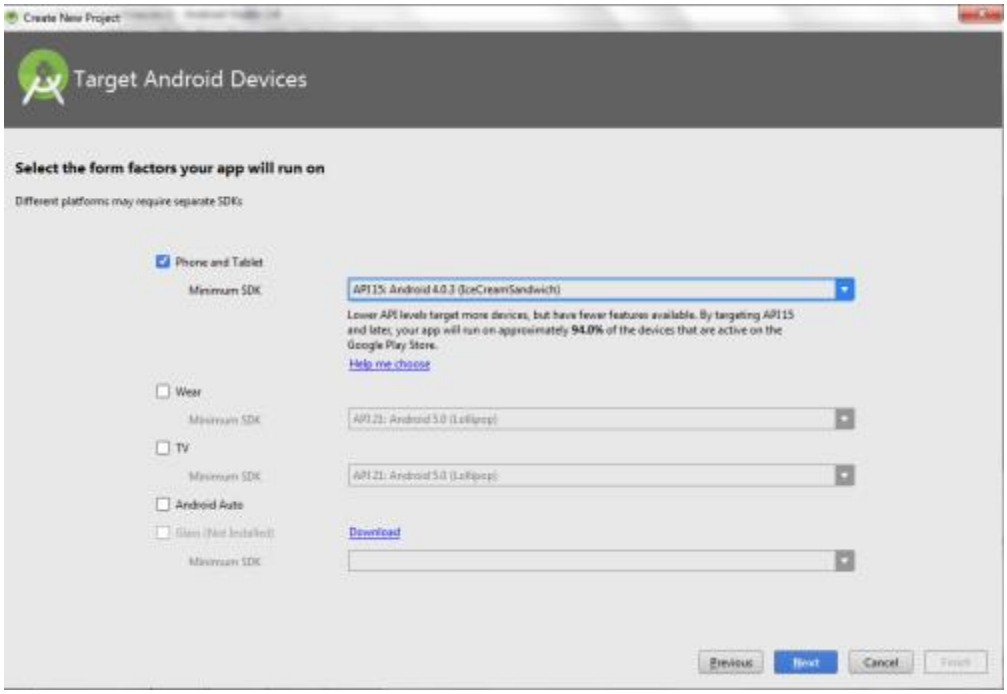
- Open Android Stdio and then click on **File -> New -> New project.**



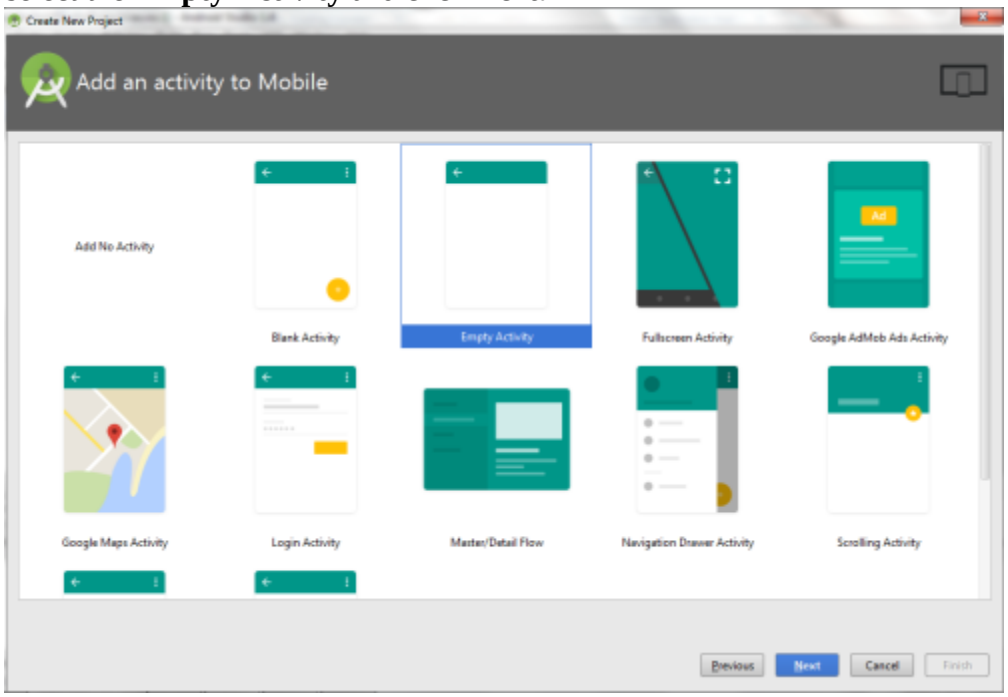
- Then type the Application name as "**ex.no.3"** and click **Next.**



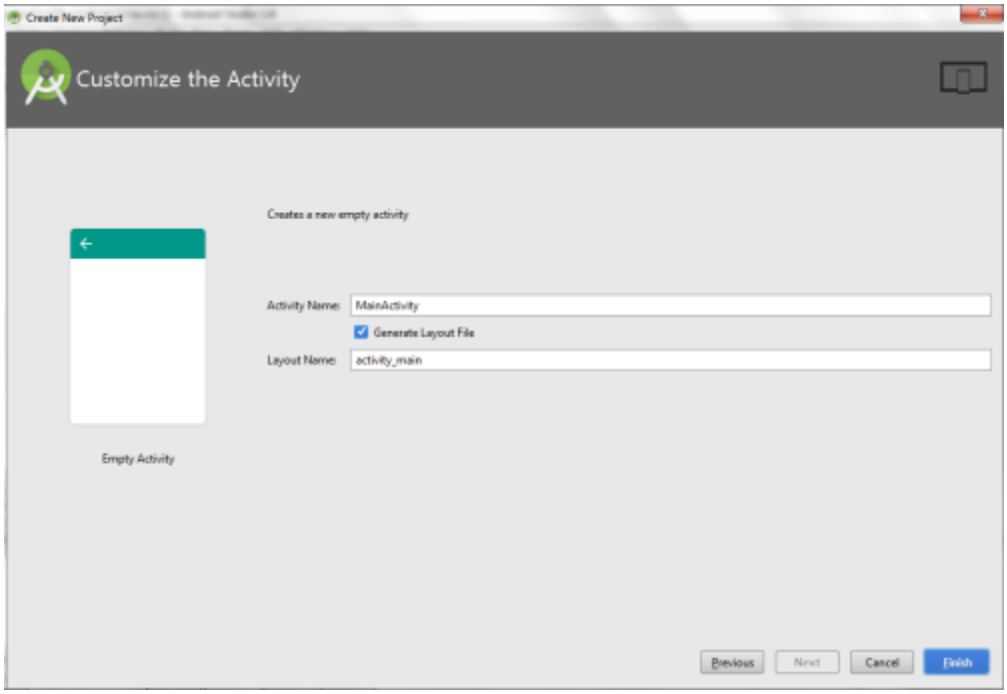
- Then select the **Minimum SDK** as shown below and click **Next**.



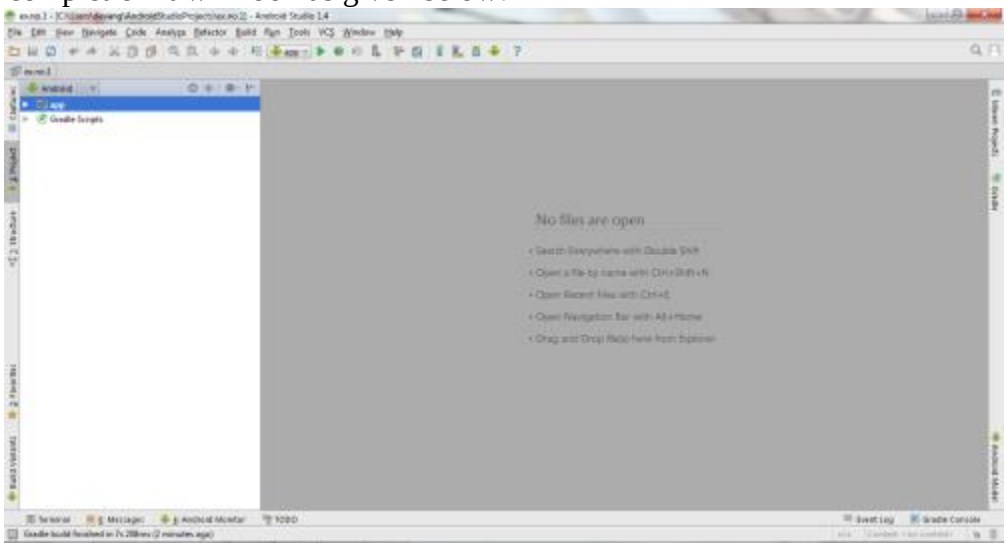
- Then select the **Empty Activity** and click **Next**.



- Finally click F**inish**.

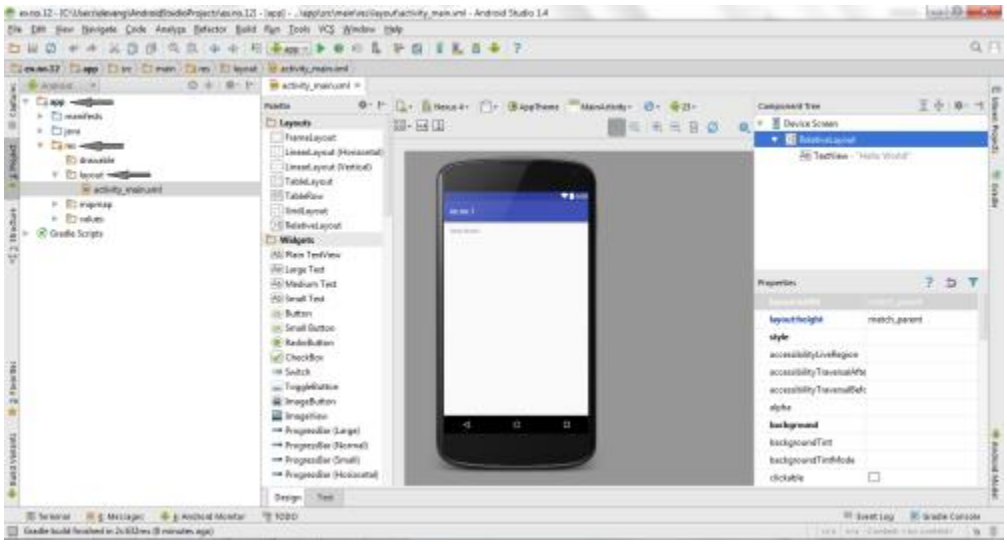


- It will take some time to build and load the project.
- After completion it will look as given below.

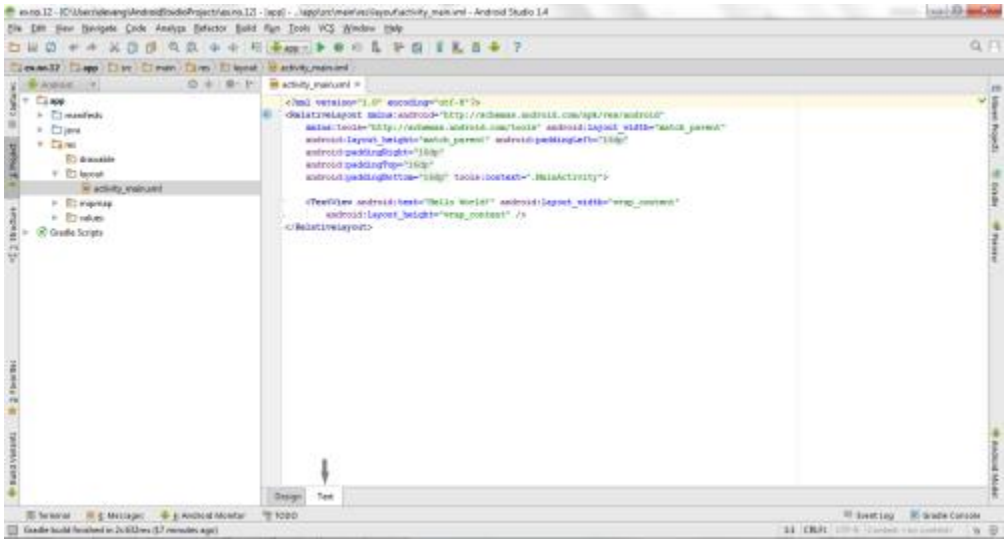


Designing layout for the Android Application:

- Click on **app -> res -> layout -> activity_main.xml.**



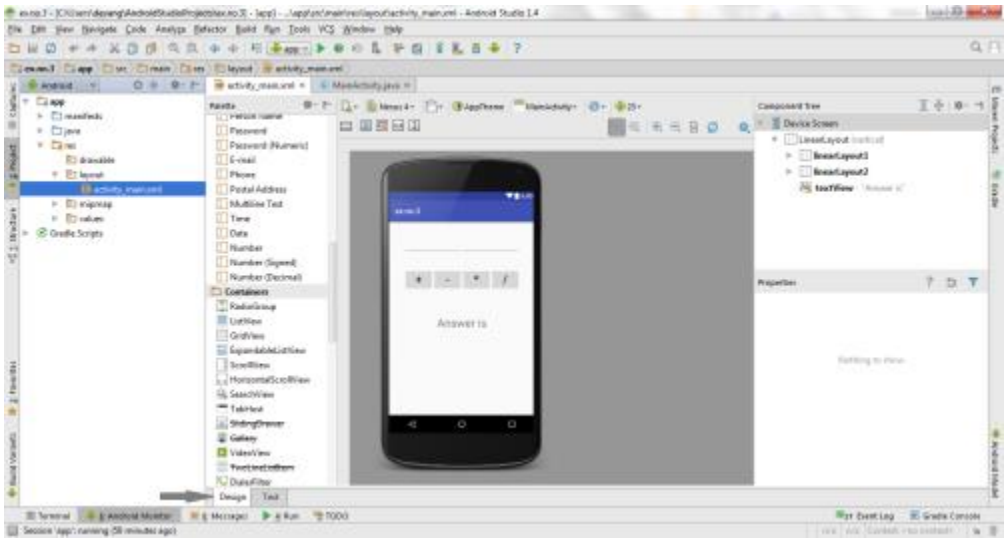
- Now click on **Text** as shown below.



- Then delete the code which is there and type the code as given below.

Code for Activity_main.xml:

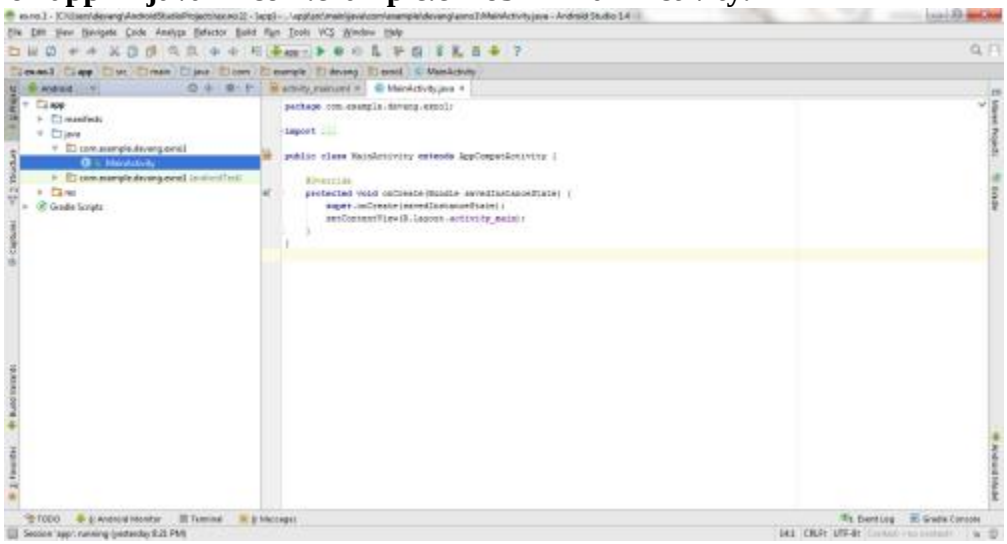
- Now click on Design and your application will look as given below.



- So now the designing part is completed.

Java Coding for the Android Application:

- Click on **app -> java -> com.example.exno3 -> MainActivity.**



- Then delete the code which is there and type the code as given below.

Code for MainActivity.java:

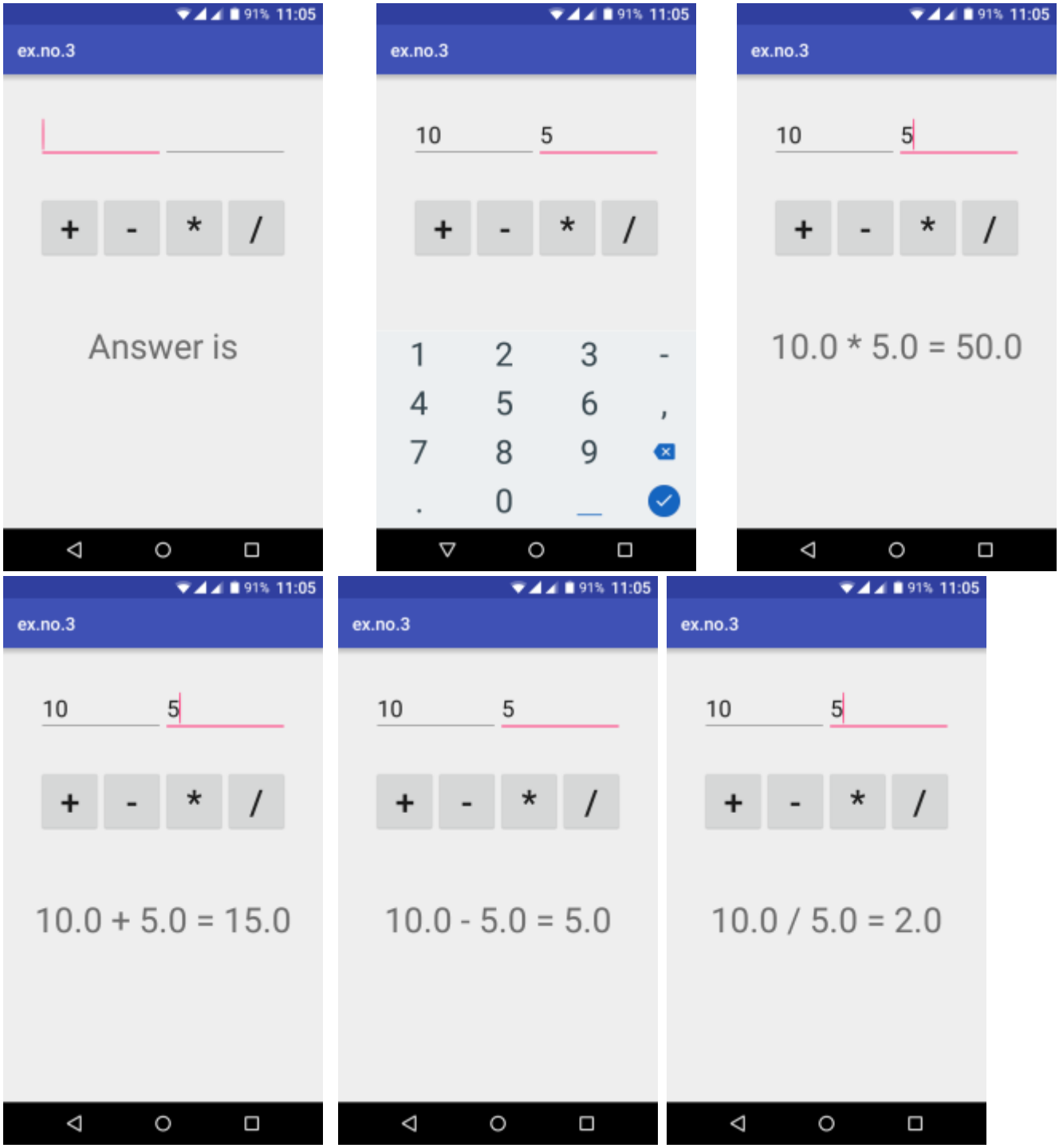
```
package com.example.devang.exno3;
 import android.os.Bundle;
import android.support.v7.app.AppCompatActivity;
import android.text.TextUtils;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;
```

```
public class MainActivity extends AppCompatActivity implements OnClickListener
{
   //Defining the Views
   EditText Num1;
   EditText Num2;
   Button Add;
   Button Sub;
   Button Mul;
   Button Div;
   TextView Result;

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

      //Referring the Views
      Num1 = (EditText) findViewById(R.id.editText1);
      Num2 = (EditText) findViewById(R.id.editText2);
      Add = (Button) findViewById(R.id.Add);
      Sub = (Button) findViewById(R.id.Sub);
      Mul = (Button) findViewById(R.id.Mul);
      Div = (Button) findViewById(R.id.Div);
      Result = (TextView) findViewById(R.id.textView);

      // set a listener
      Add.setOnClickListener(this);
      Sub.setOnClickListener(this);
      Mul.setOnClickListener(this);
      Div.setOnClickListener(this);
   }

   @Override
   public void onClick (View v)
   {

      float num1 = 0;
      float num2 = 0;
      float result = 0;
```

```
        String oper = "";

        // check if the fields are empty
        if (TextUtils.isEmpty(Num1.getText().toString()) ||
 TextUtils.isEmpty(Num2.getText().toString()))
                return;

        // read EditText and fill variables with numbers
        num1 = Float.parseFloat(Num1.getText().toString());
        num2 = Float.parseFloat(Num2.getText().toString());

        // defines the button that has been clicked and performs the corresponding operation
        // write operation into oper, we will use it later for output
        switch (v.getId())
        {
            case R.id.Add:
                oper = "+";
                result = num1 + num2;
                break;
            case R.id.Sub:
                oper = "-";
                result = num1 - num2;
                break;
            case R.id.Mul:
                oper = "*";
                result = num1 * num2;
                break;
            case R.id.Div:
                oper = "/";
                result = num1 / num2;
                break;
            default:
                break;
        }
        // form the output line
        Result.setText(num1 + " " + oper + " " + num2 + " = " + result);
    }
}
```

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

OUTPUT:



RESULT:

Thus a Simple Android Application for Native Calculator is developed and executed successfully.

EX.NO:4	**DEVELOP A GAMING APPLICATION THAT USES 2-D ANIMATIONS AND GESTURES**

AIM:

To Develop a gaming application that uses 2-D animations and gestures.

Procedure:

Flutter Flame setup:

To get started with Flame, you need to install the package. In your `pubspec.yaml` file, add the dependency as shown below:

```
dependencies:
  flame: ^1.1.1
```

To render a game, you use the `GameWidget`. Adding the code snippet below in the `main.dart` file renders a Flame game, which is currently a black screen.

```
void main() {
  final game = FlameGame();
  runApp(
    GameWidget(
      game: game,
    ),
  );
}
```

You are now ready to add some graphics to your game.

Loading sprites

- ✓ To render static images, you'll need to make use of the `SpriteComponent` class. Add your game graphics in the `assets/images` folder, and update your `pubspec.yaml` file to load the assets.
- ✓ This tutorial contains player and background images that will be loaded.

- ✓ You'll create and update the three files below in the `lib` folder:
- ✓ `dino_player.dart`, which will load and position our player:

```dart
import 'package:flame/components.dart';

class DinoPlayer extends SpriteComponent with HasGameRef {
  DinoPlayer() : super(size: Vector2.all(100.0));

  @override
  Future<void> onLoad() async {
    super.onLoad();
    sprite = await gameRef.loadSprite('idle.png');
    position = gameRef.size / 2;
  }
}
```

- dino_world.dart, which will load our game background:

```dart
import 'package:flame/components.dart';

class DinoWorld extends SpriteComponent with HasGameRef {
  @override
  Future<void> onLoad() async {
    super.onLoad();
    sprite = await gameRef.loadSprite('background.png');
    size = sprite!.originalSize;
  }
}
```

- dino_game.dart, which will manage all our game components. It adds our game player and background and positions them:

```
import 'dart:ui';
import 'package:flame/game.dart';
import 'dino_player.dart';
import 'dino_world.dart';
   class DinoGame extends FlameGame
{
  DinoPlayer _dinoPlayer = DinoPlayer();
 DinoWorld _dinoWorld = DinoWorld();
  @override
  Future<void> onLoad() async
 {
    super.onLoad();
    await add(_dinoWorld);
    await add(_dinoPlayer);
    _dinoPlayer.position = _dinoWorld.size / 1.5;
    camera.followComponent(_dinoPlayer,
      worldBounds: Rect.fromLTRB(0, 0, _dinoWorld.size.x, _dinoWorld.size.y));
  }
 }
```

  - ✓ The `camera.followComponent` function sets the game viewport to follow the player.
  - ✓ This function is necessary, as we'll be adding motion to our player.
  - ✓ Update your `main.dart` file to load the `DinoGame` as shown below:

```dart
import 'package:flame/game.dart';
import 'package:flutter/material.dart';
import 'dino_game.dart';

void main() {
  final game = DinoGame();
  runApp(
    GameWidget(game: game),
  );
}
```

Running your application should display your player and a background.

OUTPUT:

SPRITE MOVEMENT:

First, create a helpers folder with the files below, and update them as shown:

- directions.dart contains the directions enum:

```
enum Direction { up, down, left, right, none }
```

- navigation_keys.dart contains the UI and logic of the navigation keys:

```
import 'package:flutter/gestures.dart';
import 'package:flutter/material.dart';
import 'directions.dart';

class NavigationKeys extends StatefulWidget {
  final ValueChanged<Direction>? onDirectionChanged;

  const NavigationKeys({Key? key, required this.onDirectionChanged})
      : super(key: key);

  @override
  State<NavigationKeys> createState() => _NavigationKeysState();
}

class _NavigationKeysState extends State<NavigationKeys> {
  Direction direction = Direction.none;

  @override
  Widget build(BuildContext context) {
    return SizedBox(
      height: 200,
      width: 120,
      child: Column(
        children: [
          ArrowKey(
            icons: Icons.keyboard_arrow_up,
            onTapDown: (det) {
              updateDirection(Direction.up);
            },
            onTapUp: (dets) {
              updateDirection(Direction.none);
            },
            onLongPressDown: () {
              updateDirection(Direction.up);
            },
            onLongPressEnd: (dets) {
              updateDirection(Direction.none);
            },
          ),
          Row(
            mainAxisAlignment: MainAxisAlignment.center,
```

```
          children: [
            ArrowKey(
              icons: Icons.keyboard_arrow_left,
              onTapDown: (det) {
                updateDirection(Direction.left);
              },
              onTapUp: (dets) {
                updateDirection(Direction.none);
              },
              onLongPressDown: () {
                updateDirection(Direction.left);
              },
              onLongPressEnd: (dets) {
                updateDirection(Direction.none);
              },
            ),
            ArrowKey(
              icons: Icons.keyboard_arrow_right,
              onTapDown: (det) {
                updateDirection(Direction.right);
              },
              onTapUp: (dets) {
                updateDirection(Direction.none);
              },
              onLongPressDown: () {
                updateDirection(Direction.right);
              },
              onLongPressEnd: (dets) {
                updateDirection(Direction.none);
              },
            ),
          ],
        ),
        ArrowKey(
          icons: Icons.keyboard_arrow_down,
          onTapDown: (det) {
            updateDirection(Direction.down);
          },
          onTapUp: (dets) {
            updateDirection(Direction.none);
          },
          onLongPressDown: () {
            updateDirection(Direction.down);
          },
          onLongPressEnd: (dets) {
            updateDirection(Direction.none);
          },
```

```
            ),
          ],
        ),
      );
    }

    void updateDirection(Direction newDirection) {
      direction = newDirection;
      widget.onDirectionChanged!(direction);
    }
  }

  class ArrowKey extends StatelessWidget {
    const ArrowKey({
      Key? key,
      required this.icons,
      required this.onTapDown,
      required this.onTapUp,
      required this.onLongPressDown,
      required this.onLongPressEnd,
    }) : super(key: key);
    final IconData icons;
    final Function(TapDownDetails) onTapDown;
    final Function(TapUpDetails) onTapUp;
    final Function() onLongPressDown;
    final Function(LongPressEndDetails) onLongPressEnd;

    @override
    Widget build(BuildContext context) {
      return GestureDetector(
        onTapDown: onTapDown,
        onTapUp: onTapUp,
        onLongPress: onLongPressDown,
        onLongPressEnd: onLongPressEnd,
        child: Container(
          margin: const EdgeInsets.all(8),
          decoration: BoxDecoration(
            color: const Color(0x88ffffff),
            borderRadius: BorderRadius.circular(60),
          ),
          child: Icon(
            icons,
            size: 42,
          ),
        ),
      );
    }
```

```
}
```

✓ Then, update the `main.dart` file to display your game and keys as shown below:

```
void main()
{
final game = DinoGame();
runApp(
MaterialApp(
debugShowCheckedModeBanner: false,
home: Scaffold(
body: Stack(
children: [
GameWidget(
game: game,
),
Align(
alignment: Alignment.bottomRight,
child: NavigationKeys(onDirectionChanged: game.onArrowKeyChanged,),),],),),),);
```

✓ Add the function below into the `dino_game.dart` file to execute the player's movement:

```
onArrowKeyChanged(Direction direction){
  _dinoPlayer.direction = direction;
}
```

✓ Finally, update the `dino_player.dart` file to update the player's position by including the code snippet below:

```
Direction direction = Direction.none;
 @override
void update(double dt) {
  super.update(dt);
  updatePosition(dt);
}

updatePosition(double dt) {
  switch (direction) {
    case Direction.up:
      position.y --;
      break;
    case Direction.down:
      position.y ++;
      break;
    case Direction.left:
```

```
                position.x --;
                break;
              case Direction.right:
                position.x ++;
                break;
              case Direction.none:
                break;
            }
  }
```

Running your application and pressing any of the arrow keys should update your player's position.

OUTPUT:

RESULT:

To develop a gaming application that uses 2-D animations and gestures executed successfully.

EX.NO:5	DEVELOP A MOVIE RATING APPLICATION (SIMILAR TO IMDB)

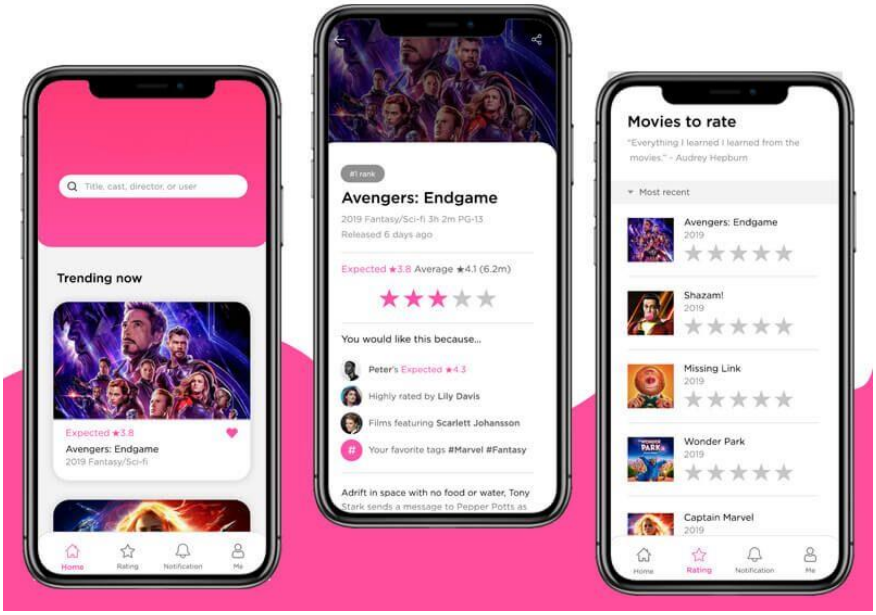
AIM:

To develop a movie rating application (SIMILAR TO IMDB)

PROCEDURE:

Step 1:
Understand Your Niche and Target Audience
The first thing to understand and work on, when you have an application idea is studying the niche in detail. Here we are talking about ***movie review app development*** and a few things that you need to be clear on are:

Step 2:
Monetization Strategy for Your IMDb Application
As the **movie rating app development** team enters the market with an app idea the purpose is to resolve the problems that our potential users may face. Along with that another point to consider here is the revenue. Entering the market with a highly rewarding application is an important point to focus on.



Platform to Launch Your Application

- **Native Application:** You can reach our team for native mobile app development where your application would be available for limited platforms. It could be that you enter the market with just an Android or iOS mobile application instead of a universal application.
- **Hybrid Application:** Media and entertainment industry with advanced technology has seen a great revolution. You can hire a **hybrid mobile app development** team for your product that would be compatible with all the devices and platforms.
- **Web Application:** Another sustainable alternative is to go for a web application. In this case, you would not have to worry about the platforms and other details. This would simply be the application that opens on your web browser. This is much similar to the amazon web app model.

2. Features to Include in Your IMDB like Movie Review and Rating Application

Another important factor that you need to focus on when working on the idea of your application is the feature set. There are a few common key mobile app features that our experts would integrate with your product, but along with them, there are a few more that would make things easier**.**

Step 3: Determining Movie Review App Development Cost

The cost of development for any mobile application is determined by various factors. We understand investing in your idea is a great opportunity and doing it the right way is vital.

We share here the most dominant factors to **create a movie rating app** that would define the cost of development of your application.

- Platforms
- Features and Functions
- Technology
- Region of Development Team

Hiring a development team for your project is a difficult decision. Simply check on the factors and see the response that would help you make a wise decision.

1. Platform to Launch Your Application

- ✓ If you have a universal user base you want your application to be available over all the possible platforms. If you want to go for a limited audience or be available for simply one of the platforms then the cost of development would vary.
- ✓ **Native Application:** You can reach our team for native mobile app development where your application would be available for limited platforms. It could be that you enter the market with just an Android or iOS mobile application instead of a universal application.

- ✓ Since you have a vast user base, we wouldn't suggest this for your **movie review app development.**

- ✓ **Hybrid Application:** Media and entertainment industry with advanced technology has seen a great revolution. You can hire a **hybrid mobile app development** team for your product that would be compatible with all the devices and platforms.

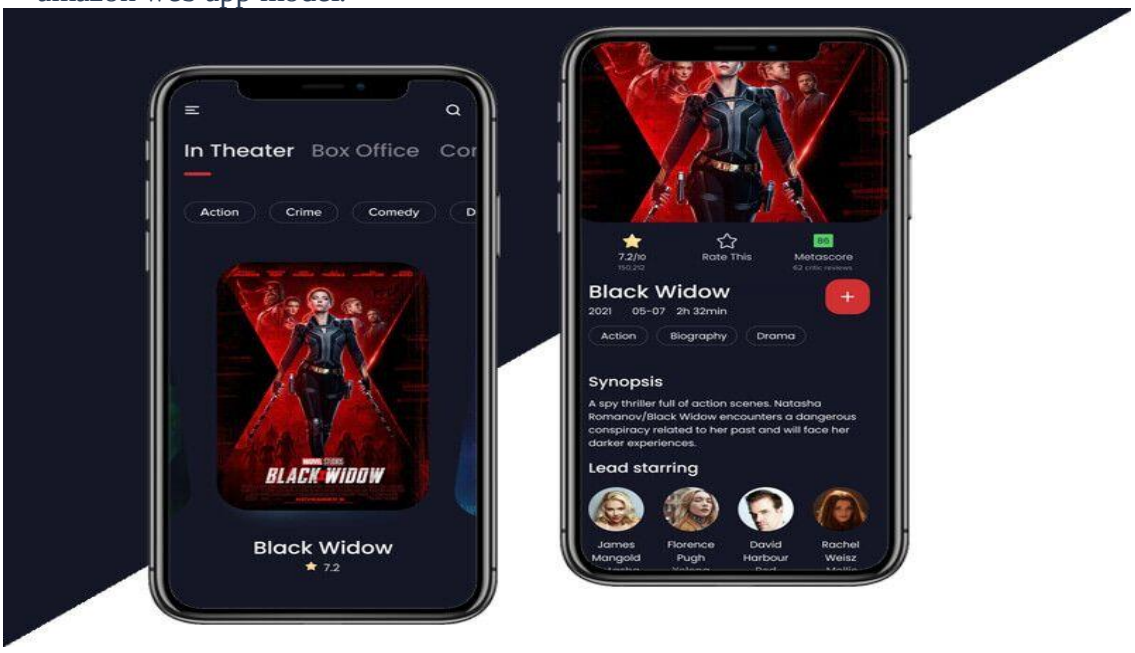
Step 4: Mobile App Development Process

- ✓ The **mobile app development process** is quite complicated. First, the wireframe is prepared where the details of the application as it is supposed to be is prepared. Here the look, feel, and working of the application are determined.

- ✓ Next, the UI/UX team builds a solution that functions according to the application. The development team then enters the picture and further develops the functions and features that the UI/UX team integrated with the application.

- ✓ As the development of the application is ongoing there are regular sprints that are conducted where you would be updated on the latest development of your application. If there is something that you need to work on or want to modify we would take the suggestions here.

- ✓ **Web Application:** Another sustainable alternative is to go for a web application. In this case, you would not have to worry about the platforms and other details. This would simply be the application that opens on your web browser. This is much similar to the amazon web app model.



PROGRAM:

First, make sure you have Flask installed

```bash
pip install Flask
```

Then, you'll also need SQLite3 which usually comes pre-installed with Python.

```
from flask import Flask, render_template, request, redirect, url_for
import sqlite3

app = Flask(__name__)

# Database Initialization
conn = sqlite3.connect('movies.db')
c = conn.cursor()
c.execute('''CREATE TABLE IF NOT EXISTS movies
             (id INTEGER PRIMARY KEY, title TEXT, rating REAL)''')
conn.commit()
conn.close()

@app.route('/')
def index():
    conn = sqlite3.connect('movies.db')
    c = conn.cursor()
    c.execute("SELECT * FROM movies")
    movies = c.fetchall()
    conn.close()
    return render_template('index.html', movies=movies)

@app.route('/rate/<int:id>', methods=['POST'])
def rate(id):
    rating = request.form['rating']
    conn = sqlite3.connect('movies.db')
    c = conn.cursor()
    c.execute("UPDATE movies SET rating = ? WHERE id = ?", (rating, id))
    conn.commit()
    conn.close()
    return redirect(url_for('index'))

if __name__ == '__main__':
    app.run(debug=True)
```

- We create a simple Flask application.
- We initialize a SQLite database to store movie data (id, title, and rating).
- We define two routes: / for displaying movies and **/rate/<id>** for updating movie ratings.
- The / route fetches all movies from the database and renders them on a template (**index.html**).

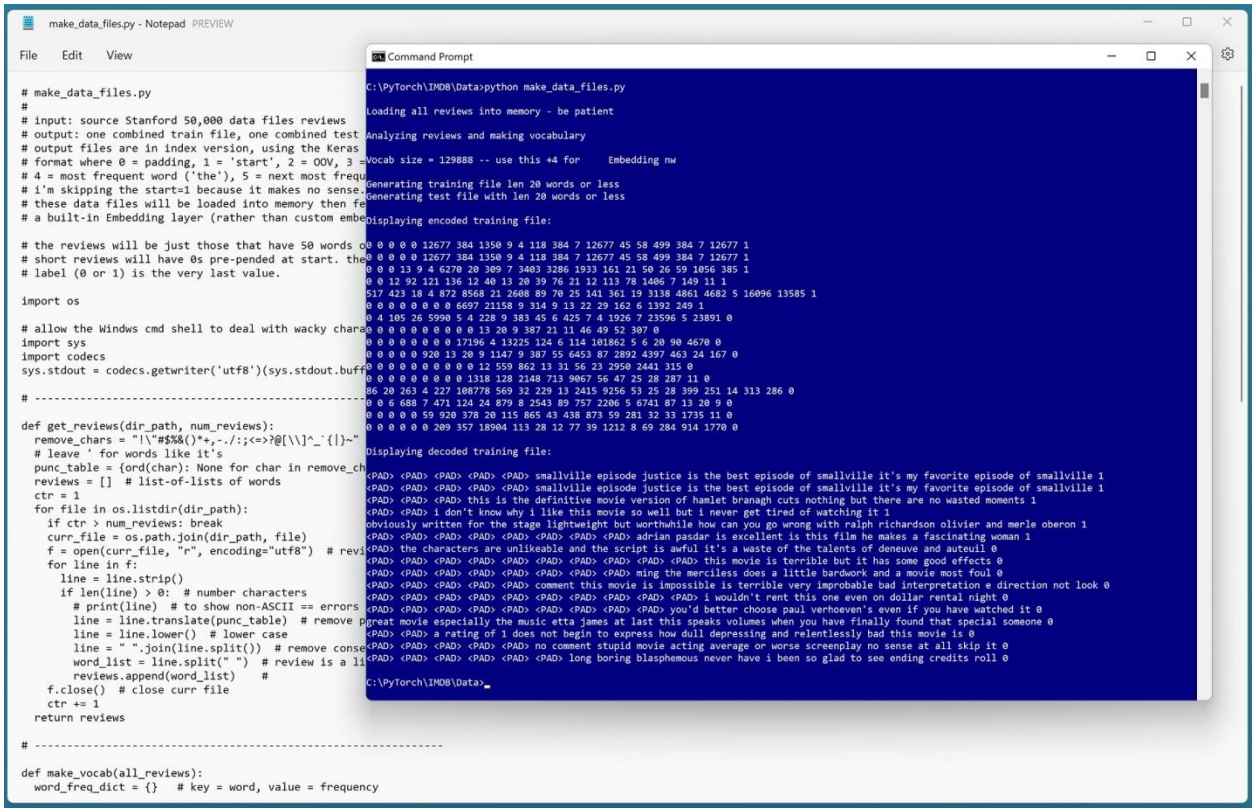
- The **/rate/<id>** route updates the rating for a specific movie and then redirects back to the homepage.
- To create a template file named **index.html** in a folder named **templates**. This file will display the movies and allow users to rate them

HTML CODING:

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <title>Movie Rating App</title>
</head>
<body>
   <h1>Movie Ratings</h1>
   <ul>
      {% for movie in movies %}
         <li>{{ movie[1] }} - Rating: {{ movie[2] or 'Not Rated Yet' }}
            <form action="{{ url_for('rate', id=movie[0]) }}" method="post">
               <select name="rating">
                  <option value="1">1</option>
                  <option value="2">2</option>
                  <option value="3">3</option>
                  <option value="4">4</option>
                  <option value="5">5</option>
               </select>
               <input type="submit" value="Rate">
            </form>
         </li>
      {% endfor %}
   </ul>
</body>
</html>
```

Now you can run your Flask application by executing the Python script. Open a browser and go to **http://localhost:5000** to see the movie list and rate them.

OUTPUT:



RESULT:

To develop a movie rating application executed successfully.

EX.NO: 6	DEVELOP AN APPLICATION TO CONNECT TO A WEB SERVICE AND TO RETRIEVE DATA WITH HTTP

AIM:

To develop an application to connect to a web service and to retrieve data with HTTP

PROCEDURE:

- We import the necessary classes from **java.io** and **java.net** packages.
- We specify the URL of the web service we want to connect to (**url**).
- We create a **URL** object with the specified URL.
- We open a connection to the URL using **openConnection()** method, which returns an instance of **HttpURLConnection**.
- We set the request method to GET using **setRequestMethod("GET")**.
- We get the response code using **getResponseCode()** method.
- If the response code is **200** (OK), we read the response body using a **BufferedReader** and append it to a **StringBuffer**.
- Finally, we print the response body.

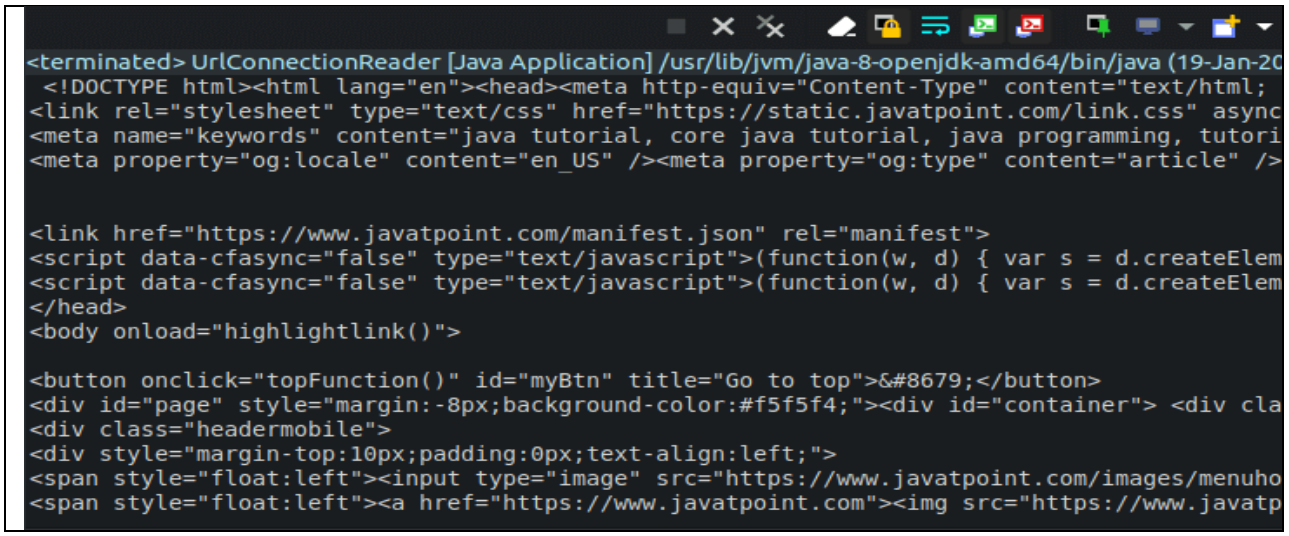
PROGRAM:

```
import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.net.HttpURLConnection;
import java.net.URL;

public class WebServiceClient {

    public static void main(String[] args) {
        String url = "https://jsonplaceholder.typicode.com/posts";

        try {
            URL obj = new URL(url);
            HttpURLConnection con = (HttpURLConnection) obj.openConnection();

            // Set request method
            con.setRequestMethod("GET");

            // Get response code
            int responseCode = con.getResponseCode();
            System.out.println("Response Code: " + responseCode);

            // Read response body
            BufferedReader in = new BufferedReader(new
```

```
InputStreamReader(con.getInputStream()));
            String inputLine;
            StringBuffer response = new StringBuffer();

            while ((inputLine = in.readLine()) != null) {
                response.append(inputLine);
            }
            in.close();

            // Print response body
            System.out.println("Response Body:");
            System.out.println(response.toString());
        } catch (Exception e) {
            System.out.println("Error: " + e.getMessage());
        }
    }
}
```

OUTPUT:

```
Response Code: 200
Response Body:
[
  {
    "userId": 1,
    "id": 1,
    "title": "sunt aut facere repellat provident occaecati excepturi optio reprehenderit",
    "body": "quia et suscipit\nsuscipit recusandae consequuntur expedita et cum\n reprehenderit molestiae ut ut quas totam\nnostrum rerum est autem sunt rem eveniet architecto"
  },
  {
    "userId": 1,
    "id": 2,
    "title": "qui est esse",
    "body": "est rerum tempore vitae\nsequi sint nihil reprehenderit dolor beatae ea dolores neque\nfugiat blanditiis voluptate porro vel nihil molestiae ut reiciendis\nqui aperiam non debitis possimus qui neque nisi nulla"
  },
  // More JSON data...
]
```



RESULT:

To develop an application to connect to a web service and to retrieve data with HTTP executed successfully.

EX.NO:7	**DEVELOP A SIMPLE SHOPPING APPLICATION**.

AIM:

To Develop a simple shopping application.

PROCEDURE:

1. Market Research for Your Online Store
2. Platform and Tech Stack Selection
3. Choose Agency for Shopping App Development
4. Identify Features for Minimum Viable Product
5. List Additional Features for Your Store to Stand Out in the Future
6. Create a Mockup of Your Online Store App
7. Begin Mobile Shopping App Development Process
8. Test Your Shopping App
9. Launch Your Shopping App

PROGRAM: JAVA CODE

```
import java.util.ArrayList;
import java.util.List;
class Product {
    private String name;
    private double price;

    public Product(String name, double price) {
        this.name = name;
        this.price = price;
    }
    public String getName() {
        return name;
    }

    public double getPrice() {
        return price;
    }
}
class ShoppingCart {
    private List<Product> items;

    public ShoppingCart() {
        items = new ArrayList<>();
```

```
    }
    public void addItem(Product product) {
        items.add(product);
    }
    public void removeItem(Product product) {
        items.remove(product);
    }
    public double getTotalPrice() {
        double totalPrice = 0;
        for (Product item : items) {
            totalPrice += item.getPrice();
        }
        return totalPrice;
    }
    public void displayCart() {
        System.out.println("Shopping Cart:");
        for (Product item : items) {
            System.out.println(item.getName() + " - $" + item.getPrice());
        }
        System.out.println("Total Price: $" + getTotalPrice());
    }
}
public class ShoppingApplication {
    public static void main(String[] args) {
        // Creating some products
        Product apple = new Product("Apple", 1.00);
        Product banana = new Product("Banana", 0.50);
        Product orange = new Product("Orange", 1.20);
       // Creating a shopping cart
        ShoppingCart cart = new ShoppingCart();
        // Adding items to the cart
        cart.addItem(apple);
        cart.addItem(banana);
        cart.addItem(orange);
        // Displaying the cart
        cart.displayCart();
    }
}
```

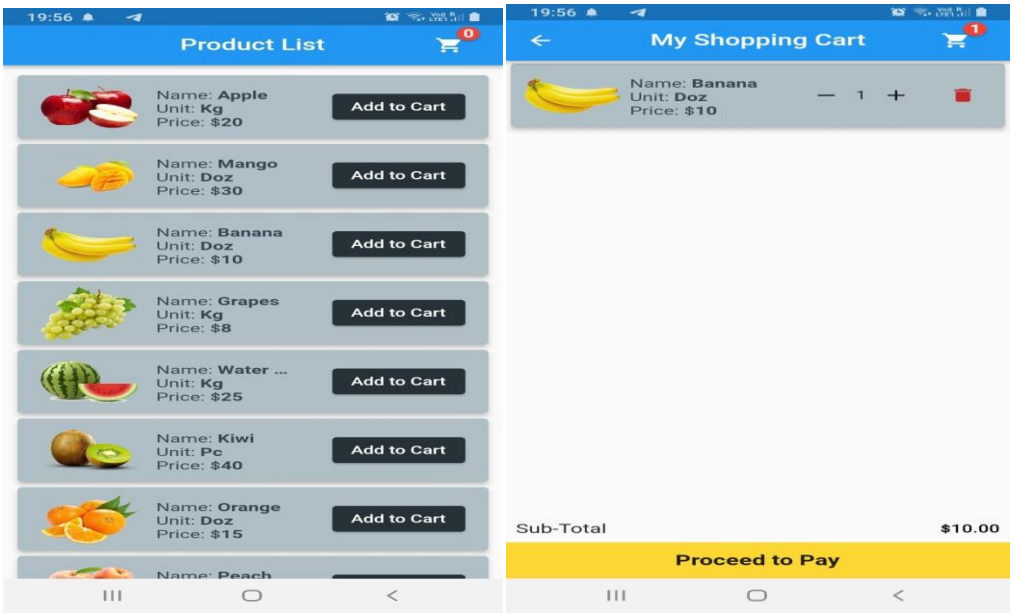
PROGRAM : HTML CODE

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Fruit Shopping</title>
```

```
    <style>
        body {
            font-family: Arial, sans-serif;
        }
        .container {
            max-width: 600px;
            margin: 0 auto;
            padding: 20px;
        }
        h1 {
            text-align: center;
        }
        ul {
            list-style-type: none;
            padding: 0;
        }
        li {
            margin-bottom: 10px;
        }
        button {
            background-color: #4CAF50;
            color: white;
            padding: 10px 20px;
            border: none;
            cursor: pointer;
            border-radius: 5px;
        }
        button:hover {
            background-color: #45a049;
        }
        .cart {
            background-color: #f2f2f2;
            padding: 10px;
            border-radius: 5px;
        }
    </style>
</head>
<body>
    <div class="container">
        <h1>Fruit Shopping</h1>
        <ul id="fruit-list">
            <li><span>Apple - $1.00</span> <button onclick="addToCart('Apple', 1.00)">Add
to Cart</button></li>
            <li><span>Banana - $0.50</span> <button onclick="addToCart('Banana',
0.50)">Add to Cart</button></li>
            <li><span>Orange - $1.20</span> <button onclick="addToCart('Orange',
1.20)">Add to Cart</button></li>
```

```
    </ul>
    <div class="cart">
      <h2>Shopping Cart</h2>
      <ul id="cart-items"></ul>
      <p>Total Price: $<span id="total-price">0.00</span></p>
    </div>
  </div>

  <script>
    let cart = [];

    function addToCart(itemName, itemPrice) {
      cart.push({ name: itemName, price: itemPrice });
      displayCart();
    }

    function displayCart() {
      let cartItems = document.getElementById('cart-items');
      cartItems.innerHTML = '';
      let totalPrice = 0;

      cart.forEach(item => {
        let li = document.createElement('li');
        li.textContent = item.name + ' - $' + item.price;
        cartItems.appendChild(li);
        totalPrice += item.price;
      });

      let totalPriceElement = document.getElementById('total-price');
      totalPriceElement.textContent = totalPrice.toFixed(2);
    }
  </script>
</body>
</html>
```

OUTPUT:



RESULT:

To develop a simple shopping application executed successfully.

EX.NO: 8	**DESIGN A WEB SERVER SUPPORTING PUSH NOTIFICATIONS**.

AIM:

To design a web server supporting push notifications.

PROCEDURE: JAVA

- We define a **PushNotificationApplication** class annotated with **@SpringBootApplication** and **@EnableWebSocket** to enable WebSocket support.
- We declare a **ServerEndpointExporter** bean to register WebSocket endpoints.
- We define a **NotificationController** class annotated with **@Controller** to handle WebSocket messages. When a message is received on the **/notification** destination, it's forwarded to the **/topic/notification** topic.
- We define a **NotificationSender** class annotated with **@RestController** to handle HTTP requests. When a GET request is made to **/send_notification** with a **message** parameter, it sends a notification message to all subscribed WebSocket clients.

PROCEDURE: HTML

- We use JavaScript to create a WebSocket object and establish a connection to the server endpoint **ws://localhost:8080/notification**.
- We define event handlers for **onopen**, **onmessage**, **onclose**, and **onerror** events to handle WebSocket connection events.
- When a message is received (**onmessage** event), an alert is displayed with the notification message.
- Note that you need to replace **'ws://localhost:8080/notification'** with the actual WebSocket server endpoint URL where your backend server is running.

PROGRAM : JAVA

```
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.context.annotation.Bean;
import org.springframework.messaging.handler.annotation.MessageMapping;
import org.springframework.messaging.handler.annotation.SendTo;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.bind.annotation.RestController;
import org.springframework.web.socket.config.annotation.EnableWebSocket;
import org.springframework.web.socket.server.standard.ServerEndpointExporter;

@SpringBootApplication
@EnableWebSocket
public class PushNotificationApplication {
```

```
    public static void main(String[] args) {
        SpringApplication.run(PushNotificationApplication.class, args);
    }

    @Bean
    public ServerEndpointExporter serverEndpointExporter() {
        return new ServerEndpointExporter();
    }

}

@Controller
class NotificationController {

    @MessageMapping("/notification")
    @SendTo("/topic/notification")
    public String sendNotification(String message) {
        return message;
    }

}

@RestController
class NotificationSender {

    private final NotificationController notificationController;

    NotificationSender(NotificationController notificationController) {
        this.notificationController = notificationController;
    }

    @GetMapping("/send_notification")
    public String sendNotification(@RequestParam String message) {
        notificationController.sendNotification(message);
        return "Notification sent";
    }
}
```

PROGRAM: HTML

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
```

```
    <title>Push Notifications</title>
    <script>
        // Establish WebSocket connection
        const socket = new WebSocket('ws://localhost:8080/notification');

        // Event handler for open connection
        socket.onopen = function(event) {
            console.log('WebSocket connection established');
        };

        // Event handler for receiving messages
        socket.onmessage = function(event) {
            const message = event.data;
            alert('Received notification: ' + message);
        };

        // Event handler for connection closed
        socket.onclose = function(event) {
            console.log('WebSocket connection closed');
        };

        // Event handler for connection errors
        socket.onerror = function(error) {
            console.error('WebSocket error: ' + error);
        };
    </script>
</head>
<body>
    <h1>Push Notifications</h1>
</body>
</html>
```

OUTPUT:

RESULT:

To design a web server supporting push notifications executed successfully.

EX.NO : 9	DEVELOP AN APPLICATION BY INTEGRATING GOOGLE MAPS

AIM:

To Develop an application by integrating Google maps.

PROCEDURE:

1. **reate a Spring Boot Project**: First, create a Spring Boot project using Spring Initializr or your preferred IDE.
2. **Add Google Maps JavaScript API Key**: Get an API key from the Google Cloud Console for the Google Maps JavaScript API and add it to your HTML page.
3. **Create HTML Page**: Create an HTML page with the necessary JavaScript code to display the Google Map.
4. **Serve HTML Page from Spring Boot**: Set up a Spring Boot controller to serve the HTML page.
5. **Run the Application**: Run your Spring Boot application and access the HTML page to view the integrated Google Map.

PROGRAM: HTML

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <title>Google Maps Integration</title>
   <style>
      #map {
         height: 400px;
         width: 100%;
      }
   </style>
</head>
<body>
   <h1>Google Maps Integration</h1>
   <div id="map"></div>

   <script
src="https://maps.googleapis.com/maps/api/js?key=YOUR_API_KEY"></script>
   <script>
```

```
      function initMap() {
         const map = new google.maps.Map(document.getElementById("map"), {
            center: { lat: -34.397, lng: 150.644 },
            zoom: 8,
         });
      }
   </script>
   <script async defer
src="https://maps.googleapis.com/maps/api/js?key=YOUR_API_KEY&callback=initMap"
></script>
</body>
</html>
```

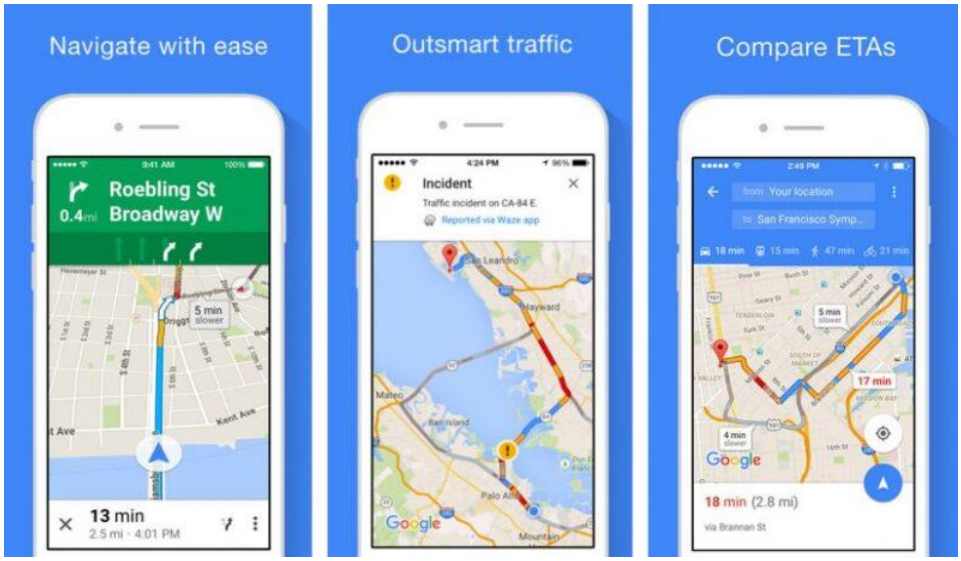
Spring Boot Controller

```
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.GetMapping;

@Controller
public class MapController {

   @GetMapping("/")
   public String index() {
      return "index";
   }
}
```

Spring Boot Application

```
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class Application {

   public static void main(String[] args) {
      SpringApplication.run(Application.class, args);
   }
}
```

OUTPUT:



RESULT:

To develop an application by integrating Google maps executed successfully.

EX.NO : 10	**MINI PROJECTS INVOLVING FLUTTER/KOTLIN MULTI-PLATFORM (FLUTTER Smart Health Prediction)**

ABSTRACT:

- It might have happened so many times that you or someone yours need doctors help immediately, but they are not available due to some reason.
- The Health Prediction application is an end user support and online consultation project.
- Here we propose an android application that allows users to get instant guidance on their health issues through an intelligent health care application online.
- The application is fed with various symptoms and the disease/illness associated with those systems.
- The application allows user to share their symptoms and issues.
- It then processes user's symptoms to check for various illness that could be associated with it.
- Here we use some intelligent data mining techniques to guess the most accurate illness that could be associated with patient's symptoms.
- If the application is not able to provide suitable results, it urges users to go for blood test, x-ray, CITI scan or whichever report it feels user's symptoms are associated with, so next time user may be able to login and upload an image of those reports.
- The application also has a doctor login, these uploaded images are now sent to respective doctor along with patient contact details.
- The doctors may now contact the patient for further process.

Modules and their Description

This application comprises of 3 major modules with their sub-modules:

User:

- **Patient Login:**
  - Patient Login to the application using his ID and Password.
- **Patient Registration:**
  - If Patient is a new user, he will enter his personal details and he will user Id and password through which he can login to the application.
- **My Details:**
  - Patient can view his personal details.
- **Disease Prediction:**
  - Patient will specify the symptoms caused due to his illness. Application will ask certain question regarding his illness and application predict the disease based on the symptoms specified by the patient and application will also suggest doctors based on the disease.
- **Search Doctor:**
  - Patient can search for doctor by specifying name, address or type.
- **Feedback:**
  - Patient will give feedback this will be reported to the admin.

Doctor:

- **Doctor Login:**
  - Doctor will access the application using his User ID and Password.
- **Patient Details:**
  - Doctor can view patient's personal details.
- **Patient's Previous Details:**
  - Doctor will get all information about patient's previous case history. That will help him to serve him better.

Admin:

- **Admin Login:**
  - Admin can login to the application using his ID and Password.
- **Add Doctor:**
  - Admin can add new doctor details into the database.
- **Add Disease:**
  - Admin can add disease details along with symptoms and type.
- **View Doctor:**
  - Admin can view various Doctors along with their personal details.
- **View Disease:**
  - Admin can view various diseases details stored in database.
- **View Patient:**
  - Admin can view various patient details who had accessed the application.
- **View Feedback:**
  - Admin can view feedback provided by various users.

❖ **Hardware Requirement: -**
- i3 Processor Based Computer
- 1GB-Ram
- 5 GB Hard Disk
- Android Device
- Internet Connection

❖ **Software Requirement:**
- Windows 7 or higher
- Android Development Toolkit(ADT)
- Visual Studio 2010
- SQL Server 2008
- Android 4.0 or higher

Advantages
- User can search for doctor's help at any point of time.
- User can talk about their illness and get instant diagnosis.
- Doctors get more clients online.

Disadvantages:

The system is not fully automated, it needs doctors for full diagnosis.

Application:

This application can be used by all patients or their family members who need help in emergency.

Sample code: python

```
from flask import Flask, request, jsonify
import numpy as np
import pickle

app = Flask(__name__)

# Load the trained machine learning model
with open('model.pkl', 'rb') as f:
    model = pickle.load(f)

@app.route('/predict', methods=['POST'])
def predict():
    # Receive health-related data from the Android application
    data = request.json

    # Preprocess the data
    # Example: Convert data to a numpy array
    input_data = np.array([data['feature1'], data['feature2'], ...])

    # Perform prediction using the trained model
    prediction = model.predict(input_data.reshape(1, -1))[0]

    # Return prediction results to the Android application
    return jsonify({'prediction': prediction})

if __name__ == '__main__':
    app.run(debug=True)
```

Sample Code : java

```
import android.os.AsyncTask;
import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.net.HttpURLConnection;
import java.net.URL;

public class HealthPredictionTask extends AsyncTask<String, Void, String> {
    private HealthPredictionListener listener;

    public HealthPredictionTask(HealthPredictionListener listener) {
        this.listener = listener;
    }
```

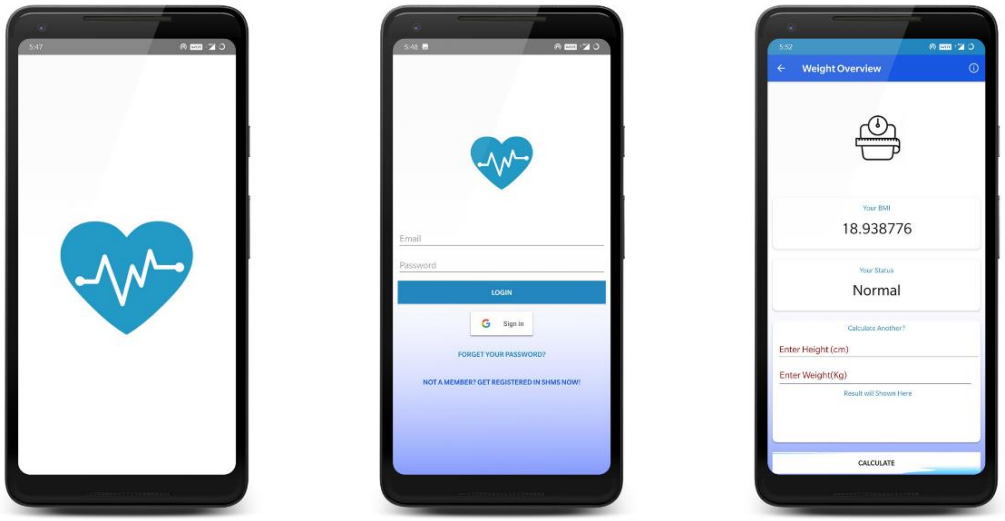
```
    @Override
    protected String doInBackground(String... params) {
        String healthData = params[0];
        String predictionResult = "";

        try {
            // Replace "SERVER_URL" with the URL of your prediction server
            URL url = new URL("SERVER_URL/predict?data=" + healthData);
            HttpURLConnection conn = (HttpURLConnection) url.openConnection();
            conn.setRequestMethod("GET");

            BufferedReader in = new BufferedReader(new
InputStreamReader(conn.getInputStream()));
            String inputLine;
            StringBuilder response = new StringBuilder();

            while ((inputLine = in.readLine()) != null) {
                response.append(inputLine);
            }
            in.close();

            predictionResult = response.toString();
        } catch (Exception e) {
            e.printStackTrace();
        }

        return predictionResult;
    }

    @Override
    protected void onPostExecute(String predictionResult) {
        listener.onPredictionCompleted(predictionResult);
    }

    public interface HealthPredictionListener {
        void onPredictionCompleted(String predictionResult);
    }
}
```

Sample code : HTML

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
```

```
    <title>Health Prediction</title>
    <script>
        function predictHealth() {
            // Get user input
            var age = document.getElementById('age').value;
            var weight = document.getElementById('weight').value;
            var height = document.getElementById('height').value;

            // Perform prediction (dummy example)
            // Replace this with actual prediction logic
            var predictionResult = "Healthy";

            // Display prediction result
            document.getElementById('predictionResult').innerText = "Prediction: " +
predictionResult;
        }
    </script>
</head>
<body>
    <h1>Health Prediction</h1>
    <label for="age">Age:</label>
    <input type="number" id="age"><br><br>
    <label for="weight">Weight (kg):</label>
    <input type="number" id="weight"><br><br>
    <label for="height">Height (cm):</label>
    <input type="number" id="height"><br><br>
    <button onclick="predictHealth()">Predict</button><br><br>
    <div id="predictionResult"></div>
</body>
</html>
```

OUTPUT:



RESULT:

Thus, the flutter smart health prediction application created was successfully.