"Major Project Report"

On

Anuppur Utilities

Submitted in partial fulfillment for the award Of diploma of engineering in Computer Science Engineering(2021-2022)

Rajiv Gandhi Proudyogiki Vishwavidyalaya Bhopal (M.P.)

Government Polytechnic College Anuppur

(M.P.)

Guided By:-Mahendra Gupta

Submitted By:- Sneha Gupta

(19111C04039)

RAJIV GANDHI PROUDYOGIKI VISHWAVIDYALAYA BHOPAL

[M.P.]

GOVERNMENT POLYTECHNIC COLLEGE ANUPPUR

CERTIFICATE

This certify that the project Report entitled as **"Anuppur utilities"**which has been completed by **Sneha Gupta** in partial fulfillment of the requirements for the award Of the **diploma engineering in computer science** for the session 2021-22 is benifited work by them and has been completed under my guidance and supervision.It has not been submitted else were.

For any other degree.

Mr.Raju Singh Paraste.	**Mr.Mahendra Gupta.**	**Mr.Utkarsh Agrawal**
(Principal)	**(Submitted To)**	**(H.O.D)**

Rajiv Gandhi Proudyogiki Vishwavidyalaya Bhopal (M.P.)

CERTIFICATE OF APPROVAL

This certify that the project Report entitled as **"Anuppur utilities** which has been completed by **Sneha Gupta** in partial fulfillment ofthe requirements for the award Of the **diploma engineering in computer science** for the session 2021-22

(INTERNAL EXAMINER) (EXTERNAL EXAMIER)

Date: ..... Date: .....

GOVERNMENT POLYTECHNIC COLLEGE ANUPPUR Department of Computer Science and Engineering

DECLARATION

Sneha Gupta , a student of Diploma in Computer Science & Engineering, in Govt. Polytechnic Collage, Anuppur (M.P.), hereby declare that the work presented in this project entitled "**Anuppur utilities**" is the outcome of our

own work, is Bonafede and correct to the best of our knowledge and this work has been carried out taking care of Engineering Ethics. The work presented does not infringe any patented work and has not been submitted to any other university or anywhere else for the award of any degree or any professional diploma.

Date: ..... **Sneha Gupta**

(19111C04045)

Project Associates :-

Sneha Gupta

AKNOWLEDGEMENT

A project this like one involves many people and in would complete without the mention of all those people whose guidance and encouragement helped in the successful completion of this project.
Our heartily thanks our faculty member of Department of Computer Science Govt. Polytechnic College Anuppur for their efforts towords our project.We would like to thanks our project in charge Mr. Mahendra Gupta who has been great source of inspiration for us and without whose humble guidance of project was never to shape. We are also indebted to our guide Mr. Mahendera Gupta for the Encouragement Guidance and Support. We are also thankful to all many people whose timely help out paucity of space is restricting us from their name. And finally, we also thanks to all my college who were constant support during the whole project.

PROJECT MEMBER:

SNEHA GUPTA

ABSTRACT

Content Table

- Flowchart
- Making Process
- App Description……

  6.1) Aim and objective of the application……

  6.2) Function…….

  6.3) Working Process…….

  6.4) Features……

  6.5) App Registration

    6.5.1) One time login- Step wise description……..

    6.5.2) Home Page…….
- Requisite Images

1.Introduction

The project name opted for this Major Project is "**Annupur Utilities**". It is an application which application is developed by using 'Android Studio and 'Fire Base'. The application is developed by **"Sneha Gupta".** This application would benefit the users during the time of emergency such as: road accidents, street short circuits and other local issues related to municipalities. The app would immediately indicate and call the Police Station, Fire helpline, Ambulance, Electricity Department and Municipality at the time of emergency or when required. The app would ease the situations and tackle the tactical situation simultaneously by taking immediate action by calling and indicating required department.

In this report further you will get to know more detailing about the development process, working process, purpose/aim and objective of the above mention application.

1.1 Purpose

The "Anuppur Utilities" is proposed to help people in case of emergency. This would assist people to reach out emergency contact in case of any problem.

1.2 Scope

The Anuppur Utilities will let people find the nearest department and contact through their location. This will help in solving the problem faster without any delay, also this will help to save human life and reduce hazard.

1.3 Aim

Ease the emergency problems, and tackle the tactical situation by taking appropriate calling action to the required department simultaneously.

1.4 Objective

The Anuppur Utilities will assist the user to contact appropriate department in following ways:

- To contact electricity department if they find any street light not functioning properly or not working.
- The “Anuppur utilities” will help to reach emergency contact hospital in case any accident or health emergency.
- The “Anuppur utilities” will help to reach emergency contact municipality in case of any damaged road, broke pipeline, cramped seawage.

Overview

This app will provide you the option of registration that is one time login which is totally secure. The app will send you the one time password (OTP) during the time of registration. The app will also provide you the option of location in the form of address so that you can get access of contact of nearest Utilities department. After the registration process you could access the home page and use the app when needed. One can get contact and address of the Utility department near them easily by using this app and contact the respective department to help the victim or casual at the time of emergency or accident.

Software Requirements

Android studio

Android Studio is a powerful and sophisticated development environment, designed with the specific purpose of developing, testing, and packaging Android applications. It is a collection of tools and components. Android Studio is not the only way to develop Android apps. It is even possible to develop a complete app using nothing more than Notepad and the command line.

Android Studio also provides an amazing time-saving device in the form of Instant Run. This feature cleverly only builds the part of a project that has been edited, meaning that developers can test small changes to code without having to wait for a complete build to be performed for each test. This feature can bring waiting time down from minutes to almost zero.

Android Studio provides a unified environment where you can build apps for Android phones, tablets, Android Wear, Android TV, and Android Auto. Structured code modules allow one to divide their project into units of functionality that you can independently build, test, and debug.

Android SDK

The Android SDK (software development kit) is a set of development tools used to develop applications for the Android platform that has become Apple's biggest rival in the smartphone space. The Android SDK includes the following:

- Required libraries.
- Debugger.
- An emulator.
- Relevant documentation for the Android application program interfaces (APIs).
- Sample source code.
- Tutorials for the Android OS.

XML

XML is a software- and hardware-independent tool for storing and transporting data. XML stands for eXtensible Markup Language. It is a markup language much like HTML.

- XML was designed to store and transport data
- XML was designed to be self-descriptive
- XML is a W3C Recommendation
- It has sender information
- It has receiver information
- It has a heading
- It has a message body
- XML was designed to carry data - with focus on what data is
- XML tags are not predefined like HTML tags are

XML Simplifies Things

- XML simplifies data sharing
- XML simplifies data transport
- XML simplifies platform changes
- XML simplifies data availability

XML stores data in plain text format. This provides a software- and hardware-independent way of storing, transporting, and sharing data. XML also makes it easier to expand or upgrade to new operating systems, new applications, or new browsers, without losing data. With XML, data can be available to all kinds of "reading machines" like people, computers, voice machines, news feeds, etc.

JDK

The Java Development Kit (JDK) is one of three core technology packages used in Java programming, along with the JVM (Java Virtual Machine) and the JRE (Java Runtime Environment). It's important to differentiate between these three technologies, as well as understanding how they're connected:

The JVM is the Java platform component that executes programs.

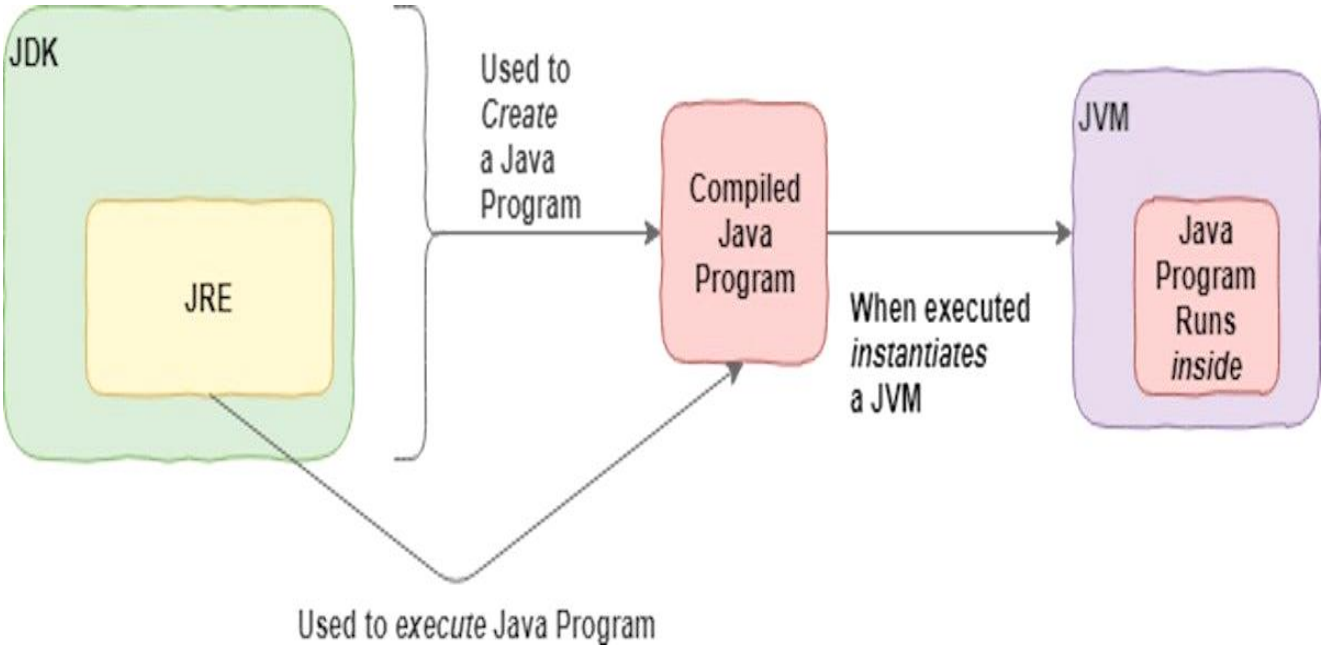
The JRE is the on-disk part of Java that creates the JVM.

The JDK allows developers to create Java programs that can be executed and run by the JVM and JRE.

Developers new to Java often confuse the Java Development Kit and the Java Runtime Environment. The distinction is that the JDK is a package of tools for *developing* Java-based software, whereas the JRE is a package of tools for *running* Java code.

The JRE can be used as a standalone component to simply run Java programs, but it's also part of the JDK. The JDK requires a JRE because running Java programs is part of developing them.

Figure 1 shows how the JDK fits into the Java application development lifecycle.



Matthew Tyson

Figure 1. High-level view of the JDK

Just as we did with my recent introduction to the Java Virtual Machine, let's consider the technical and everyday definitions of the JDK:

Technical definition: The JDK is an implementation of the Java platform specification, including compiler and class libraries.

Everyday definition: The JDK is a software package you download in order to create Java-based applications.

In addition to the JRE, which is the environment used to run Java applications, every JDK contains a Java compiler. T

Firebase

Firebase is a Backend-as-a-Service (Baas). It provides developers with a variety of tools and services to help them develop quality apps, grow their user base, and earn profit. It is built on Google's infrastructure. Firebase is categorized as a NoSQL database program, which stores data in JSON-like documents.

In Firebase, a document is a set of key-value pairs defined by a schema. A group of documents makes up a collection.

Key Features

1. Authentication

It supports authentication using passwords, phone numbers, Google, Facebook, Twitter, and more. The Firebase Authentication (SDK) can be used to manually integrate one or more sign-in methods into an app.

2. Realtime database

Data is synced across all clients in realtime and remains available even when an app goes offline.

3. Hosting

Firebase Hosting provides fast hosting for a web app; content is cached into content delivery networks worldwide.

4. Test lab

The application is tested on virtual and physical devices located in Google's data centers.

5. Notifications

Notifications c be sent with firebase with no additional coding.

Users can get started with firebase for free; more details can be found on the official website.

Connect to Firebase

Firebase is a mobile platform that helps you quickly develop high-quality apps, grow your user base, and earn more money. Firebase is made up of complementary features that you can mix-and-match to fit your needs, with Google Analytics for Firebase at the core. You can explore and integrate Firebase services in your app directly from Android Studio using the Assistant window.

Firebase Hosting works out-of-the-box with Firebase services, including Cloud Functions, Authentication, Realtime Database, Cloud Firestore, and Cloud Messaging. One can build powerful microservices and web apps using these complementary Firebase services.

The Benefits of Having Firebase for Mobile App Development

The Firebase is a Backend-as-a-Service (BaaS) that offers the developers a wide spectrum of tools and services to develop high-quality apps at a much faster pace. To define the BaaS, it is a cloud computing service model using which the web app and mobile app developers can connect their applications with

backend cloud storage and APIs rendered by the backend applications. It allows the developers to develop feature for apps on different platforms like Remote Configs, Notifications and Real-time Databases.

In addition, it also helps you to develop storage for the app including authentication functionality. Moreover, Firebase can also be used effectively for app marketing, enhancing the user experience and user engagement.

Advantages of Using Firebase For Mobile App Development

The various benefits that Firebase offers to the app developers to make their app development journey simpler:

1. Real-time Database Helps to Store and Synchronize Data

The cloud-hosted NoSQL database is offered by Firebase real-time database that helps you store and synchronize data between the clients. This indeed makes it easier for the developers to access the data using any of the devices and helps developing collaborative feature.

The Benefits of Having Firebase for Mobile App Development

For all those developers who have been searching for a robust platform for building mobile and web application, you can bank

For all those developers who have been searching for a robust platform for building mobile and web application, you can bank upon Firebase.

The Firebase is a Backend-as-a-Service (BaaS) that offers the developers a wide spectrum of tools and services to develop high-quality apps at a much faster pace.

Now, if we were to define the BaaS, it is a cloud computing service model using which the web app and mobile app developers can connect their applications with backend cloud storage and APIs rendered by the backend applications.

Initially, James Tamplin and Andrew Lee had founded the Company in 2011 as Envolve, but after Google acquired it in 2014, Firebase had an emphatic growth to become a unified platform for developers.

In fact, it allows the developers to develop feature for apps on different platforms like Remote Configs, Notifications and Real-time Databases.

In addition, it also helps you to develop storage for the app including authentication functionality.

Moreover, Firebase can also be used effectively for app marketing, enhancing the user experience and user engagement.

Advantages of Using Firebase For Mobile App Development

Analyze the various benefits that Firebase offers to the app developers to make their app development journey simpler:

1. Real-time Database Helps to Store and Synchronize Data

The cloud-hosted NoSQL database is offered by Firebase real-time database that helps you store and synchronize data between the clients. This indeed makes it easier for the developers to access the data using any of the devices and helps developing collaborative feature.

Another advantage of a real-time database for the developers is that they do not need the support of backend to build apps as it comes with SDKs for various platforms, including Android, iOS and Web.

It assists in the execution of backend code responding to events activated by databases. Furthermore, it is optimized for offline use too.

2. Firebase has Become Smarter with Google Analytics

Easily track down your user's journey on a number of devices. It means you would know whether he is using a smartphone, tablet or laptop.

One can also export your mobile app data to BigQuery with the help of Google Analytics. It can further support in engaging more users once you match the UX based on user id.

3. Firebase Offers Facility of Crash Reporting to Fix Bugs Quickly

It has often been seen that a lot of apps suffer due to bug issues, which tends to slow down navigation speed and users opt out of it. The result is that rating of the app also declines.

However, you have to credit Firebase as now it is offering the facility of crash reporting to fix the bugs at the quicker pace and with ease. The app developers and QA testers can identify the problems in the stages, whether it is the app version, the device or the OS.

The Firebase SDK provides log crashes that ultimately save much time as you don't have to find the cause of the problem. The coding database is available for both native and cross-platform apps.

4. Fast and Secured Web Hosting

The benefit of Firebase Hosting allows you to set-up a single-page, a mobile landing page, web page or progressive web page with ease. It also helps to deliver the content rapidly anywhere.

The developers can deploy the web apps as well as static content at CDN (Content Delivery Network).

The Firebase hosting also automatically configures the free SSN certificate for custom domains. You can now deploy a local directory to the web with just a single command.

5. Firebase Authentication

Nowadays, most of the apps have the login facility and the developer aims to simplify and secure it better. Therefore, the support of Firebase Authentication is there to do that task with an easy sign-in process.

It also provides identity solution for the emails, passwords and other important apps such as Facebook, Twitter or Instagram. The Firebase UI is also flexible, customized and drop-in dealing with the UI flow of the users. There is no compromise from the security point of view.

6. Firebase Allows the Content Storage with Ease

It has become much easier to store the user-dedicated content that includes texts, images and videos. In fact, you can also develop the final phase of your app from prototype effortlessly using advanced technology.

The Firebase team has also provided SDK for cloud storage to link-up the mobile for users that aren't online. As such they can continue to automatically transfer as soon as connectivity is established.

7. Developers have the Accessibility of Machine Learning

As we move on to discuss the advantages of Firebase, we also get familiar that it also provides developers the facility of Machine Learning. This benefit is available for both Android and iOS developers as well as experienced or newcomers.

The ML kit has ready-to-use APIs for various mobile functionalities such as detecting the face, identifying the text, barcode scanning and labeling images etc. You have a choice between on-device and Cloud APIs that can be selected according to your needs.

8. Send Notifications and Messages to Targeted Audiences

The Firebase Cloud Messaging offers you an opportunity to send notifications and messages to your targeted audiences for free across all devices and platforms with the help of battery-efficient connection.

So, if you are interested in sending the push notifications to a specific group of people based on demography and their behavior.

In addition, you can choose your own particular time for sending the message, which is more convenient. The developers will be pleased to note that they do not require coding to send the notifications.

You can try the A/B testing for sending notifications to choose the best version that fits your need.

9. Send the App Recommendation through Dynamic Links

Dynamic Links available with Firebase that enables the app owner to recommend his app to people familiar such as family and friends after the completion and launch of the app.

You can make your app viral by sharing a small URL. However, if they are confused about the Operating System, the app is related to the Dynamic Links will come to their rescue as it is has a browser-based property.

The dynamic links help you in converting the web users to native app users.

10. Generate More Traffic to your App with App Indexing

The app indexing in Firebase is used to re-engage the users from Google Search by surfacing the links to your app on Google Search.

When you index the application, it also helps to improve your app ranking on the Search Once, the rank gets better the new users will discover and install your app. In fact, the Android Instant App helps to access the content of the app.

11. The Facility of In-App Advertising

The Firebase is also equipped with AdMob, which is an in-app advertising facility that helps the app owner to underline the monetizing policies for his business. You can exhibit various advertisements from millions of advertisers and increase your revenue.

The AdMob also provides you the option of enhancing the user experience and lets you chose the appropriate template from plethora of option

Hardware Requirment

13 11th GEN PROCESSER

64-BIT CPU

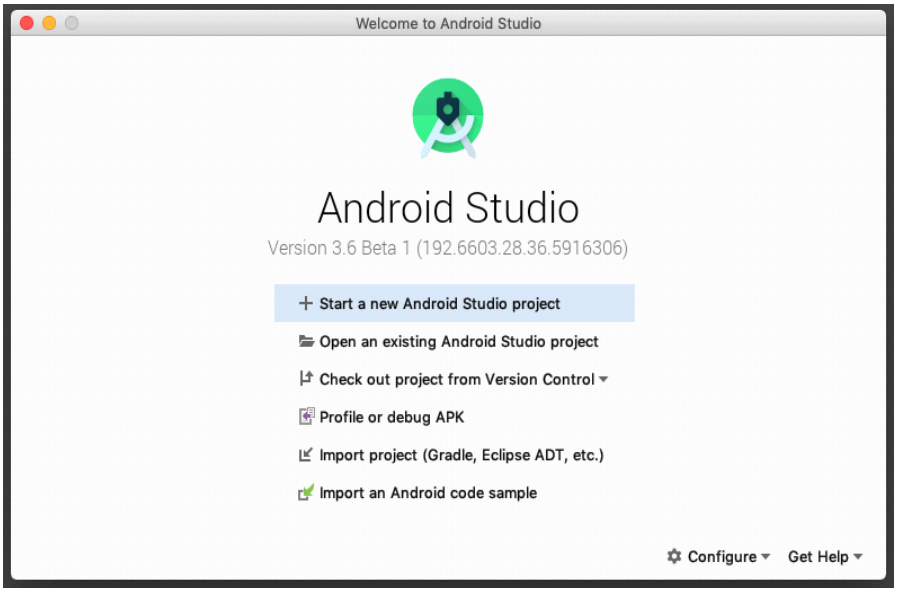
15GB+ FREE DISK SPACE

OPRATING SYSTEM WINDOW S 10

Making process

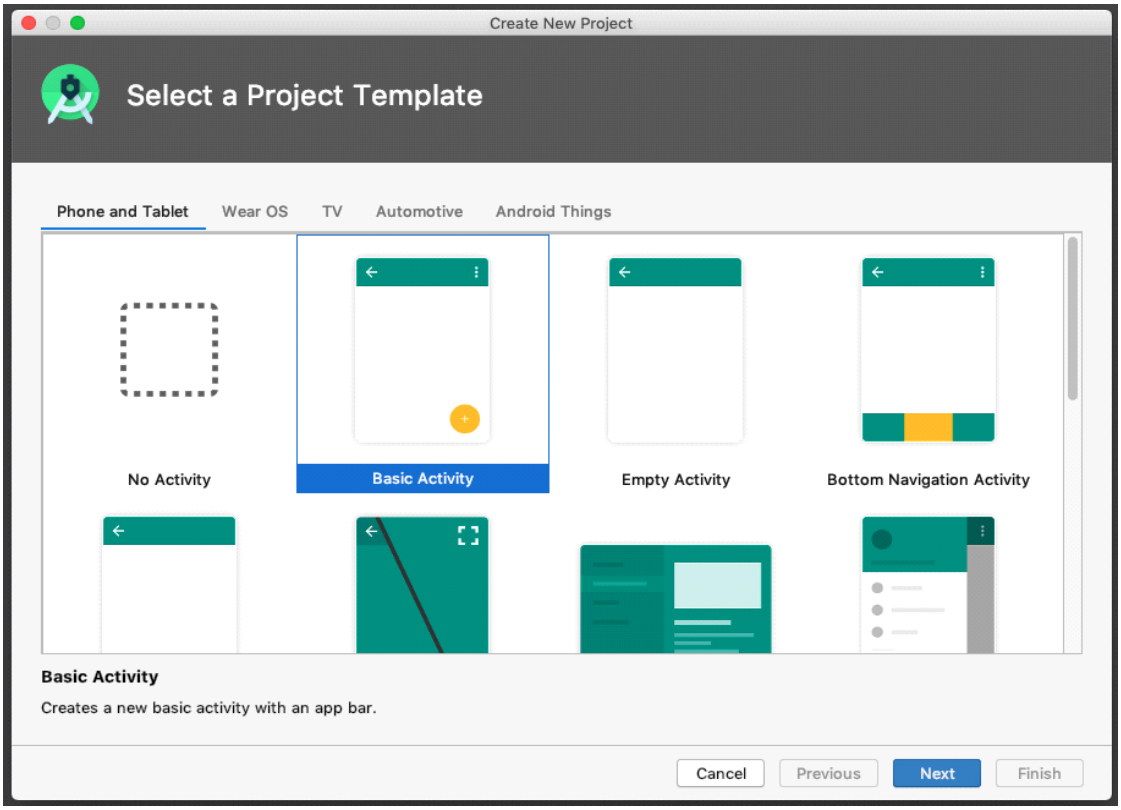
Step 1: Create a new project

- Open Android Studio.

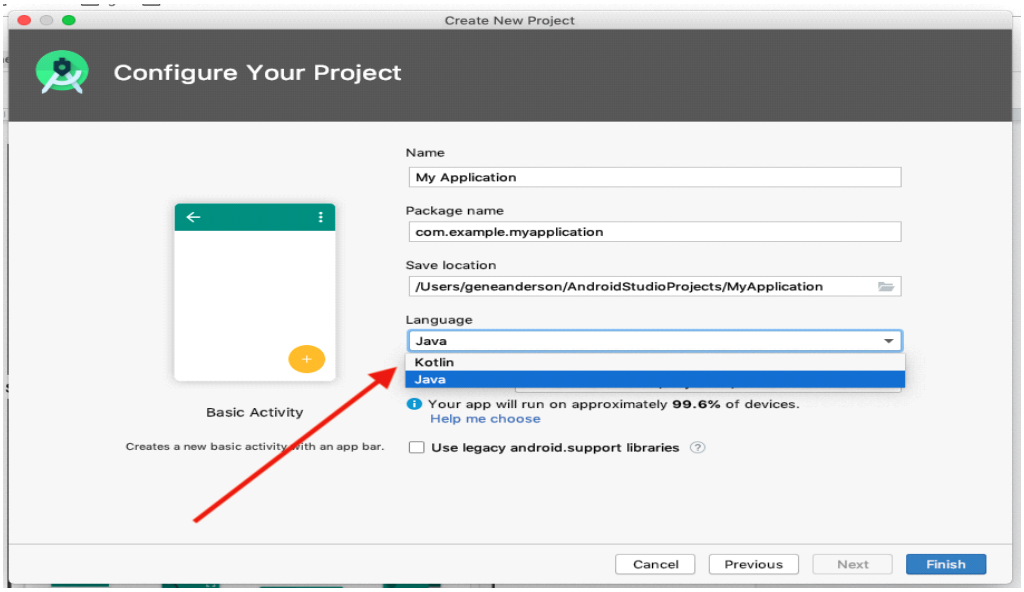


- In the Welcome to Android Studio dialog, click Start a new Android Studio project

- Select Basic Activity (not the default).
  Click Next.



- Give your application a name such as My First App.



- Make sure the Language is set to Java.
- Leave the defaults for the other fields.
- Click Finish**.**

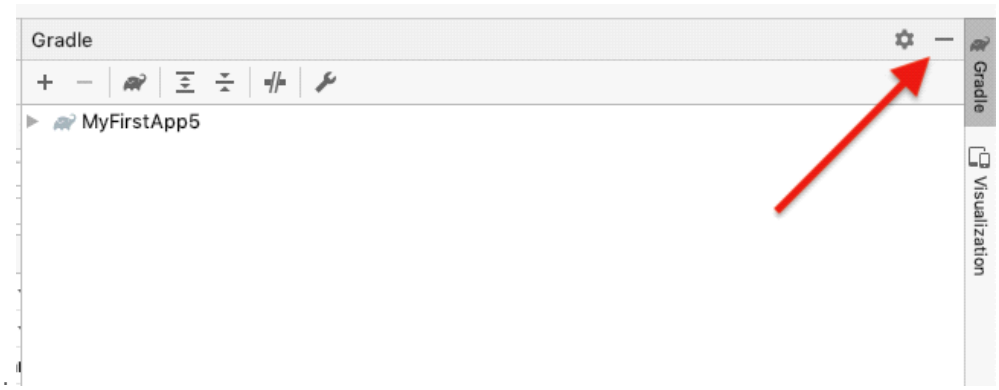
After these steps, Android Studio:

- Creates a folder for your Android Studio project called MyFirstApp. This is usually in a folder called “Android Studio Projects” below your home directory.
- Builds your project (this may take a few moments). Android Studio uses Gradle as its build system. You can follow the build progress at the bottom of the Android Studio window.
- Opens the code editor showing your project

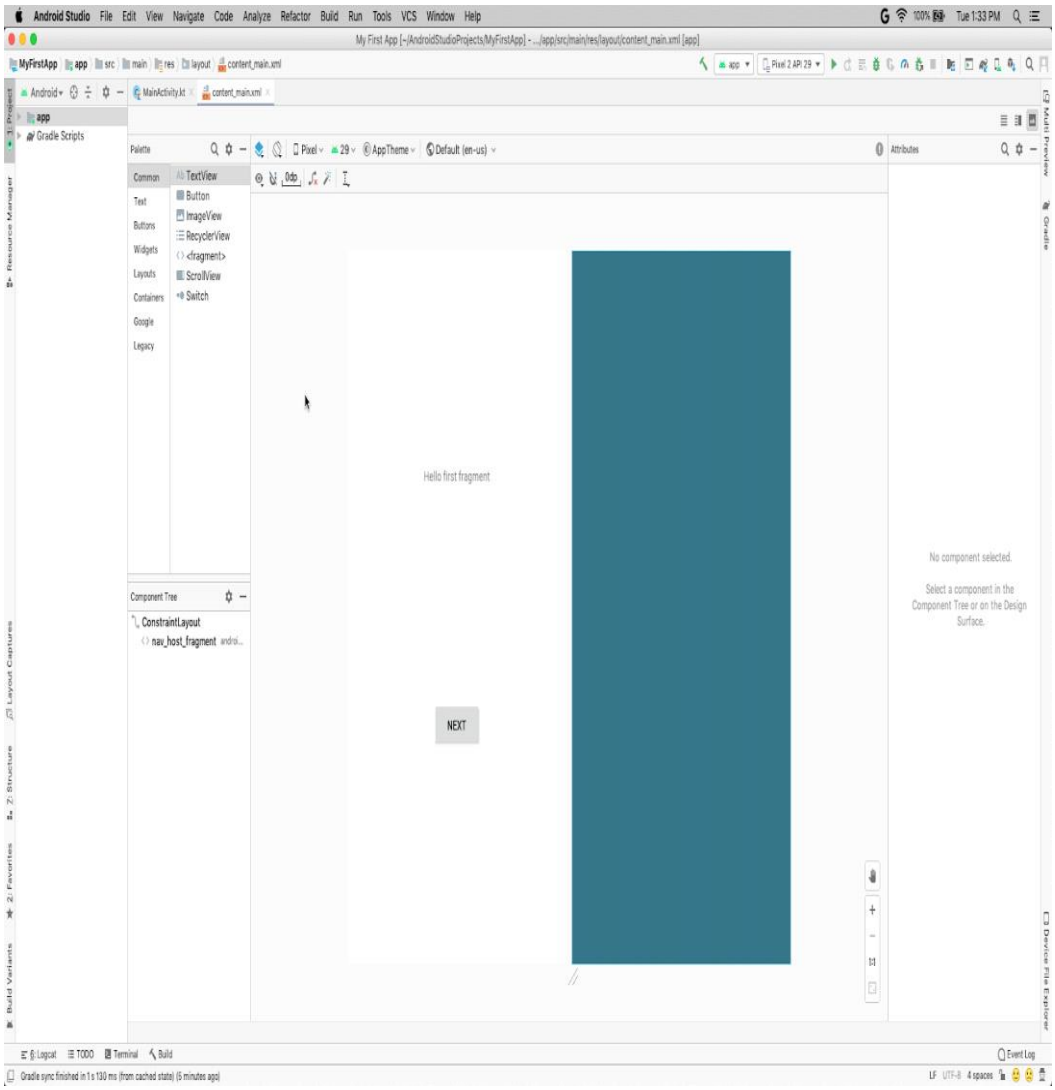
Step 2: Get your screen set up

When your project first opens in Android Studio, there may be a lot of windows and panes open. To make it easier to get to know Android Studio, here are some suggestions on how to customize the layout.

- If there's a Gradle window open on the right side, click on the minimize button (—) in the upper right corner to hideit.

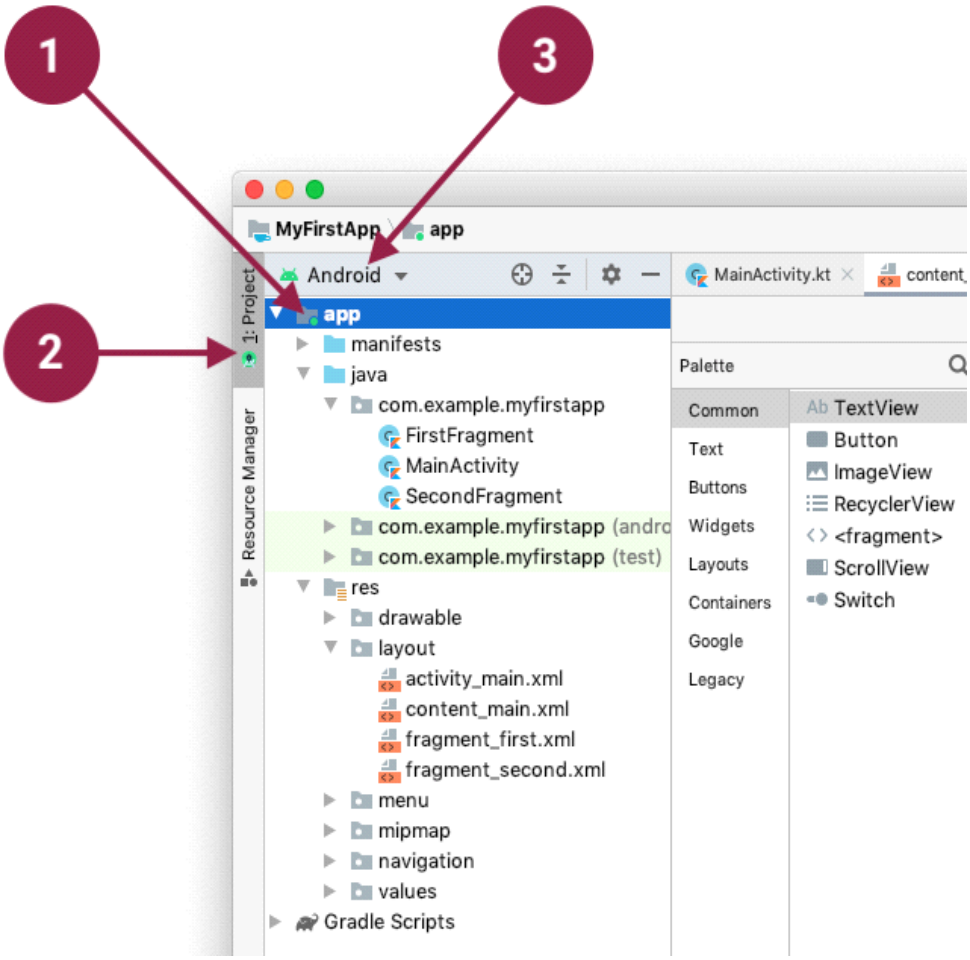


- Depending on the size of your screen, consider resizing the pane on the left showing the project folders to take up less space.
- At this point, your screen should look a bit less cluttered, similar to the screenshot shown below.



Step 3: Explore the project structure and layout

The upper left of the Android Studio window should look similar to the following diagram:



Based on you selecting the **Basic Activity** template for your project, Android Studio has set up a number of files for you. You can look at the hierarchy of the files for your app in multiple ways, one is in Project view. Project view shows your files and folders structured in a way that is convenient for working with an Android project. (This does not always match the file hierarchy! To see the file hierarchy, choose the Project files view by clicking (3).)

- Double-click the app (1) folder to expand the hierarchy of app files. (See **(1)** in the screenshot.)
- If you click Project (2), you can hide or show the **Project** view. You might need to select **View** > Tool Windows to see this option.
- The current Project view selection (3) is Project > Android.

In the Project > Android view you see three or four top-level folders below
your app folder: manifests, java, java (generated) and res. You may not see **java (generated)** right away.

- Expand the **manifests** folder.

This folder contains `AndroidManifest.xml`. This file describes all the components of your Android app and is read by the Android runtime system when your app is executed.

2. Expand the **java** folder. All your Java language files are organized here. The **java** folder contains three subfolders:

com.example.myfirstapp: This folder contains the Java source code files for your app.

com.example.myfirstapp (androidTest): This folder is where you would put your instrumented tests, which are tests that run on an Android device. It starts out with a skeleton test file.

com.example.myfirstapp (test): This folder is where you would put your unit tests. Unit tests don't need an Android device to run. It starts out with a skeleton unit test file. 3. Expand the **res** folder. This folder contains all the resources for your app, including images, layout files, strings, icons, and styling. It includes these subfolders:

drawable: All your app's images will be stored in this folder.

layout: This folder contains the UI layout files for your activities. Currently, your app has one activity that has a layout file called `activity_main.xml`. It also contains `content_main.xml`, `fragment_first.xml`, and `fragment_second.xml`.

menu: This folder contains XML files describing any menus in your app.

mipmap: This folder contains the launcher icons for your app.

navigation: This folder contains the navigation graph, which tells Android Studio how to navigate between different parts of your application.

values: This folder contains resources, such as strings and colors, used in your app.

Step 4: Create a virtual device (emulator)

In this task, you will use the Android Virtual Device (AVD) manager to create a virtual device (or emulator) that simulates the configuration for a particular type of Android device.

The first step is to create a configuration that describes the virtual device.

- In Android Studio, select **Tools** > **AVD Manager**, or click the AVD Manager icon in the toolbar. 

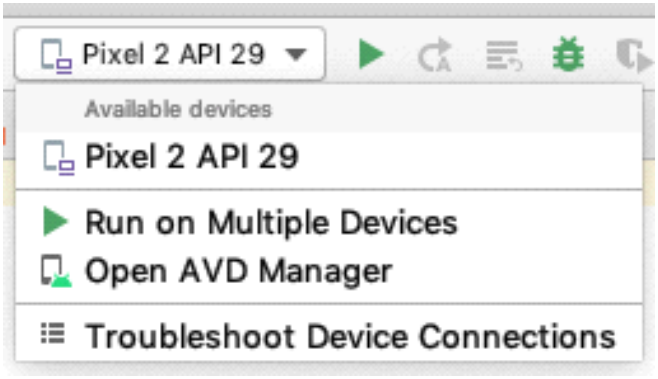
- Click **+Create Virtual Device**. (If you have created a virtual device before, the window shows all of your existing devices and the **+Create Virtual Device** button is at the bottom.) The **Select Hardware** window shows a list of pre-configured hardware device definitions.
    - Choose a device definition, such as **Pixel 2**, and click **Next.** (For this codelab, it really doesn't matter which device definition you pick).
    - In the **System Image** dialog, from the **Recommended** tab, choose the latest release. (This does matter.)
    - If a **Download** link is visible next to a latest release, it is not installed yet, and you need to download it first. If necessary, click the link to start the download, and click **Next** when it's done. This may take a while depending on your connection speed.
    - In the next dialog box, accept the defaults, and click Finish.
    - The AVD Manager now shows the virtual device you added.
    - If the Your Virtual Devices AVD Manager window is still open, go ahead and close it.

Step 5: Run your app on your new emulator

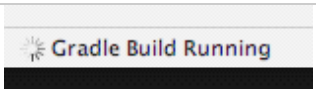
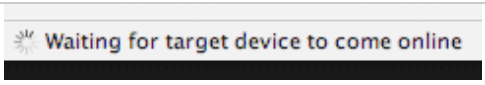
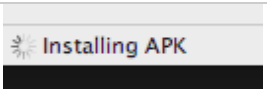
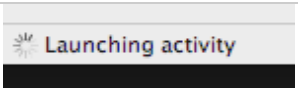
- In Android Studio, select **Run > Run 'app'** or click the **Run** icon in the toolbar.  The icon will change when your app is already running. 

If you get a dialog box stating "Instant Run requires that the platform corresponding to your target device (Android N...) is installed" go ahead and click **Install and continue.**

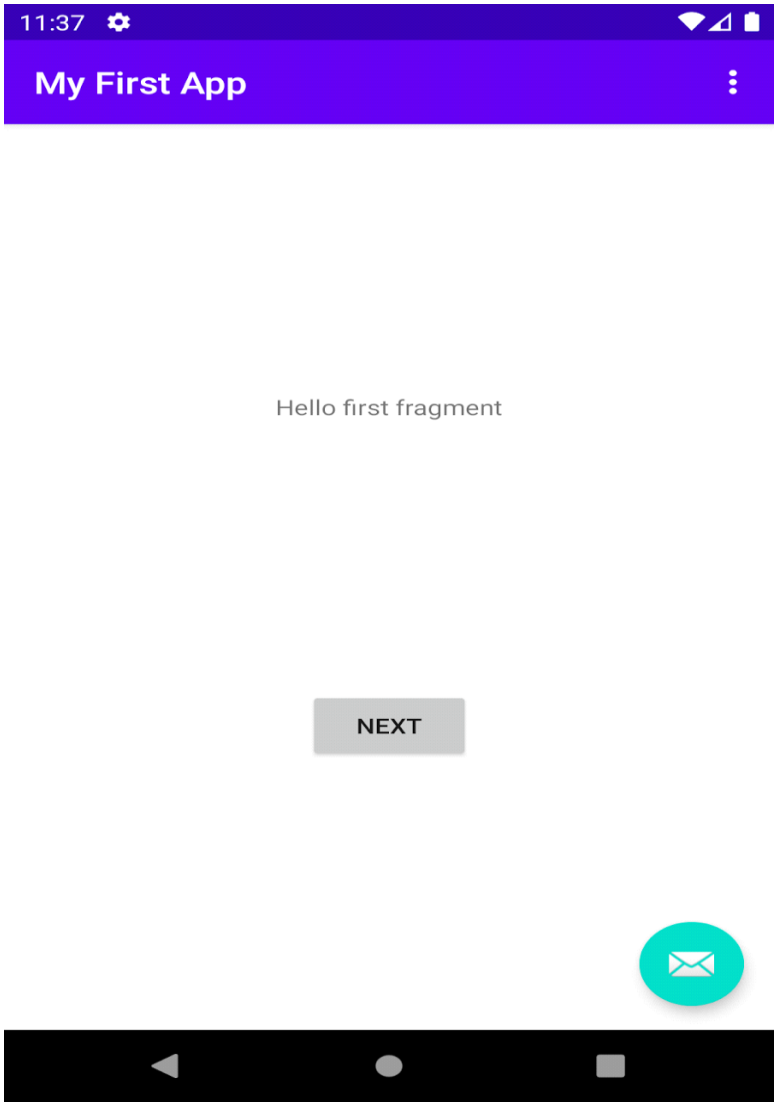
- In **Run > Select Device**, under **Available devices**, select the virtual device that you just configured. This menu also appears in the toolbar.



The emulator starts and boots just like a physical device. Depending on the speed of your computer, this may take a while. You can look in the small horizontal status bar at the very bottom of Android Studio for messages to see the progress

Messages that might appear briefly in the status bar		
Gradle build running		
Waiting for target device to come on line		
Installing APK		
Launching activity		

Once your app builds and the emulator is ready, Android Studio uploads the app to the emulator and runs it. You should see your app as shown in the following screenshot.


11:37
My First App
Hello first fragment
NEXT

Step 6: Run your app on a device (if you have one)

What you need:

An Android device such as a phone or tablet

A data cable to connect your Android device to your computer via the USB port.

If you are using a Linux or Windows OS, you may need to perform additional steps to run your app on a hardware device. Check the Run Apps on a Hardware Device documentation. On Windows, you may need to install the appropriate USB driver for your device. See OEM USB Drivers.

Run your app on a device

To let Android Studio communicate with your device, you must turn on USB Debugging on your Android device.

On Android 4.2 and higher, the Developer options screen is hidden by default. To show Developer options and enable USB Debugging:

On your device, open **Settings > About phone** and tap **Build number** seven times.

Return to the previous screen (**Settings**). **Developer options** appears at the bottom of the list.
Tap **Developer options**.

Enable **USB Debugging**.

Now you can connect your device and run the app from Android Studio.

Connect your device to your development machine with a USB cable. On the device, you might need to agree to allow USB debugging from your development device.

In Android Studio, click **Run**  in the toolbar at the top of the window. (You might need to select **View > Toolbar** to see this option.) The **Select Deployment Target** dialog opens with the list of available emulators and connected devices.

Select your device, and click **OK**. Android Studio installs the app on your device and runs it.

Troubleshooting

If you're stuck, quit Android Studio and restart it.

If Android Studio does not recognize your device, try the following:

Disconnect your device from your development machine and reconnect it.

Restart Android Studio.

If your computer still does not find the device or declares it "unauthorized":

Disconnect the device.

On the device, open **Settings->Developer Options**.

Tap **Revoke USB Debugging authorizations**.

Reconnect the device to your computer.

When prompted, grant authorizations.

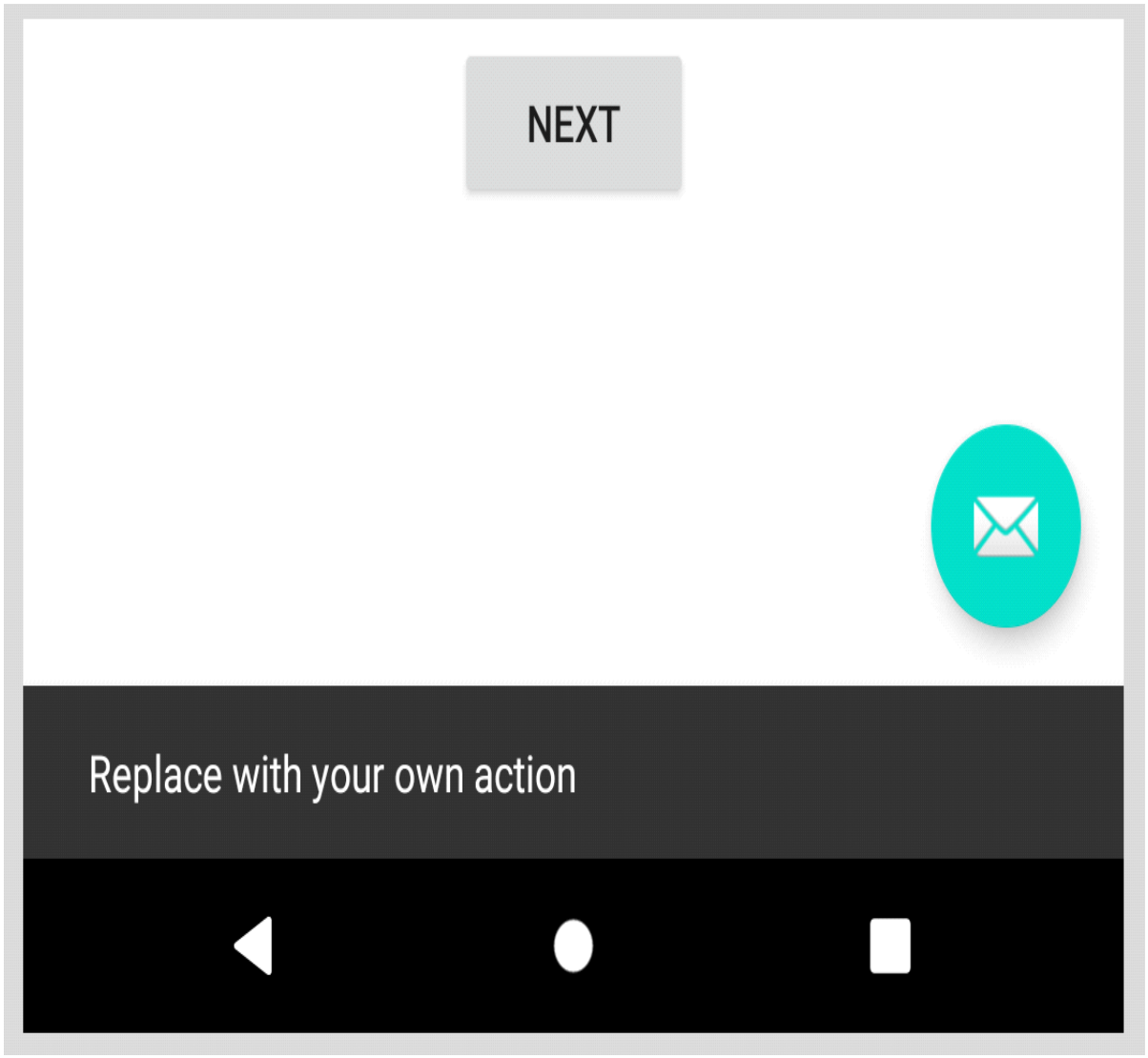
If you are still having trouble, check that you installed the appropriate USB driver for your device. See the Using Hardware Devices documentation.

Check the troubleshooting section in the Android Studio documentation.

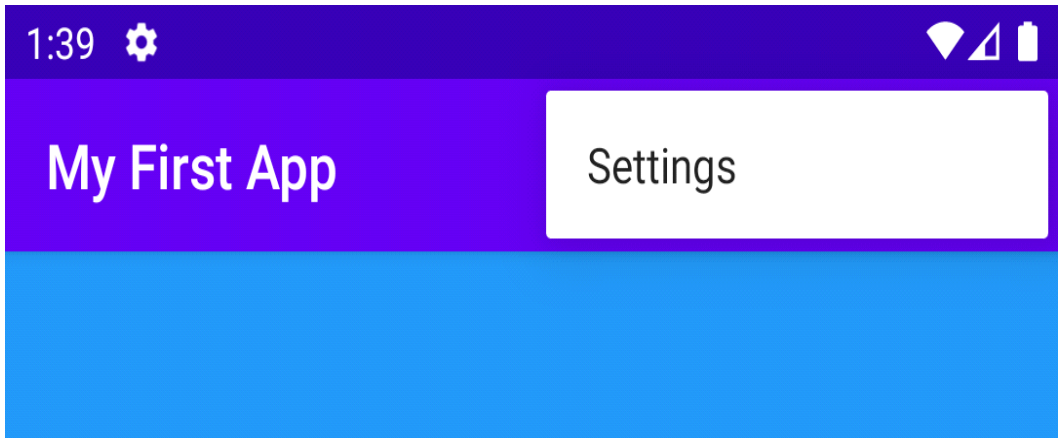
Step 7: Explore the app template

When you created the project and selected **Basic Activity**, Android Studio set up a number of files, folders, and also user interface elements for you, so you can start out with a working app and major components in place. This makes it easier to build your application.

Looking at your app on the emulator or your device, in addition to the **Next** button, notice the floating action button with an email icon.  If you tap that button, you'll see it has been set up to briefly show a message at the bottom of the screen. This message space is called a snackbar, and it's one of several ways to notify users of your app with brief information.



At the top right of the screen, there's a menu with 3 vertical dots.  If you tap on that, you'll see that Android Studio has also created an options menu with a **Settings** item. Choosing **Settings** doesn't do anything yet, but having it set up for you makes it easier to add user-configurable settings to your app.



Later in this codelab, you'll look at the **Next** button and modify the way it looks and what it does.

App Description

The name of this developed app is “Anuppur Utility”. The name of this app is decided on the basis of its functionality; as the Utility is for a service provided for the public such as water, street assets, fire fighting facility and others. The app will bring the contact of some emergency department of Annupur at the time of need or any casualty. So, the accurate name for the app according to its functioning system is Anuppur Utility.

Aim & Objective of the application

The aim of this application is to help victim and add a positive effort to resist any hazard to happen. The objective is to depict and provide the contact of the nearest emergency department to help victim and aid any type of casualty.

The Anuppur Utilities will let people find the nearest department and contact through their location. This will help in solving the problem faster without any delay, also this will help to save human life and reduce hazard.

Function

This app will provide you the option of registration that is one time login which is totally secure. The app will send you the one time password (OTP) during the time of registration. The app will also provide you the option of location in the form of address so that you can get access of contact of nearest Utilities department. After the registration process you could access the home page and use the app when needed. One can get contact and address of the Utility department near them easily by using this app and contact the respective department to help the victim or casual at the time of emergency or accident.

Working process

The app is as simple and easy to use, just follow the following steps to monitor this app:

- Click the app
- The app opens and the Home page appears
- The user could see five options in the Home page. Each option is linked with the contact number, email, and address of the respective department the option is named at.
- As the user click the department name he/she will reach to next page.
- The next page brings the contact and other required details of the ward. There are contacts of more than one wards of each department. For example: for hospital there is more than one option of the hospital as contact number of many hospitals is shown.
- By clicking the contact button you can directly access the calling the department.

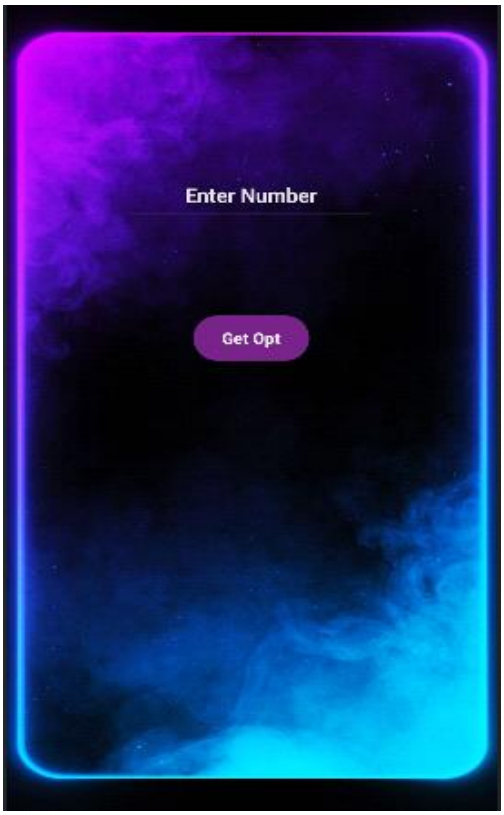
Features of Anuppur Utility

The characteristic features of the application Anuppur Utility are as follows:

- Ease the emergency
- Easy to use
- One time login
- Add value to the phone library
- Multi optional

Requirement Images

Activity_main.Xml



Working Code-MainActivity

```
package com.example.anuppurutilites;

import androidx.appcompat.app.AppCompatActivity;

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

import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.auth.FirebaseUser;


public class MainActivity extends AppCompatActivity {
   EditText t;
   Button b1;
   String t1;


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

      t=(EditText)findViewById(R.id.editTextNumber);
      b1= findViewById(R.id.button);
//       t1="+916269040396";
//       t1= String.valueOf(t.getText());



      b1.setOnClickListener(new View.OnClickListener() {
         @Override
         public void onClick(View view) {
            Intent  intent = new Intent(MainActivity.this,otp_register.class);
            if(t.getText().toString().isEmpty())
            {
               Toast.makeText(getApplicationContext(), "Please Enter Number",
Toast.LENGTH_LONG).show();
            }
            else if(t.getText().toString().length()<10)
            {
               Toast.makeText(getApplicationContext(), "Number is less than 10",
Toast.LENGTH_LONG).show();
            }
            else if(t.getText().toString().length()>10)
```

```
            {
                Toast.makeText(getApplicationContext(), "Number is greater than 10",
Toast.LENGTH_LONG).show();
            }
            else {
                intent.putExtra("mobile", "+91" + t.getText().toString());
                startActivity(intent);
                System.out.println(t.getText());
            }

        }
    });


}

@Override
protected void onStart() {
    super.onStart();
   FirebaseUser user= FirebaseAuth.getInstance().getCurrentUser();
   if (user!=null){
      startActivity(new Intent(MainActivity.this,homepage.class));
      finish();
   }
}
```

Activity_otp_register.Xml

Working Code-otp_register

```
package com.example.anuppurutilites;

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

import android.content.Intent;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;
```

```
import com.google.android.gms.tasks.OnCompleteListener;
import com.google.android.gms.tasks.Task;
import com.google.firebase.FirebaseException;
import com.google.firebase.auth.AuthResult;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.auth.FirebaseAuthInvalidCredentialsException;
import com.google.firebase.auth.FirebaseUser;
import com.google.firebase.auth.PhoneAuthCredential;
import com.google.firebase.auth.PhoneAuthOptions;
import com.google.firebase.auth.PhoneAuthProvider;

import java.util.concurrent.TimeUnit;

public class otp_register extends AppCompatActivity {
   EditText t2;
   Button b2;
   String phone;
   FirebaseAuth mAuth;
   String otpid;
   @Override
   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_otp_register);

      t2 = findViewById(R.id.editTextNumber2);
      b2 = findViewById(R.id.button2);
      phone = getIntent().getStringExtra("mobile").toString();
      System.out.println(phone);
      mAuth = FirebaseAuth.getInstance();
      initiateotp();
      b2.setOnClickListener(new View.OnClickListener() {
         @Override
         public void onClick(View view) {
            if (t2.getText().toString().isEmpty()) {
               Toast.makeText(getApplicationContext(), "blank field can not be process",
Toast.LENGTH_LONG).show();
            } else if (t2.getText().toString().length() != 6) {
               Toast.makeText(getApplicationContext(), "Invalid otp", Toast.LENGTH_LONG).show();
            } else {
               PhoneAuthCredential credential = PhoneAuthProvider.getCredential(otpid,
t2.getText().toString());
               signInWithPhoneAuthCredential(credential);
            }
         }
      });
   }
```

```
   private void initiateotp() {
      PhoneAuthOptions options =
            PhoneAuthOptions.newBuilder(mAuth)
                  .setPhoneNumber(phone)      // Phone number to verify
                  .setTimeout(60L, TimeUnit.SECONDS) // Timeout and unit
                  .setActivity(this)              // Activity (for callback binding)
                  .setCallbacks(new PhoneAuthProvider.OnVerificationStateChangedCallbacks() {
                     @Override
                     public void onCodeSent(@NonNull String s, @NonNull
PhoneAuthProvider.ForceResendingToken forceResendingToken) {
                        otpid=s;
                     }

                     @Override
                     public void onVerificationCompleted(@NonNull PhoneAuthCredential
phoneAuthCredential) {
                        signInWithPhoneAuthCredential(phoneAuthCredential);
                     }

                     @Override
                     public void onVerificationFailed(@NonNull FirebaseException e) {
                        Toast.makeText(getApplicationContext(),"something went
worng",Toast.LENGTH_LONG).show();
                     }
                  })       // OnVerificationStateChangedCallbacks
                  .build();
      PhoneAuthProvider.verifyPhoneNumber(options);
   }

   private void signInWithPhoneAuthCredential(PhoneAuthCredential credential) {
      mAuth.signInWithCredential(credential)
            .addOnCompleteListener(this, new OnCompleteListener<AuthResult>() {
               @Override
               public void onComplete(@NonNull Task<AuthResult> task) {

                  if (task.isSuccessful()) {
//
                     startActivity(new Intent(otp_register.this,homepage.class));
                     finish();
                     System.out.println("welcome");
                  }
                  else {
                     Toast.makeText(getApplicationContext(),"something went
worng",Toast.LENGTH_LONG).show();

                  }
               }
```

```
        });
    }

}
```

Activity_Homepage.Xml

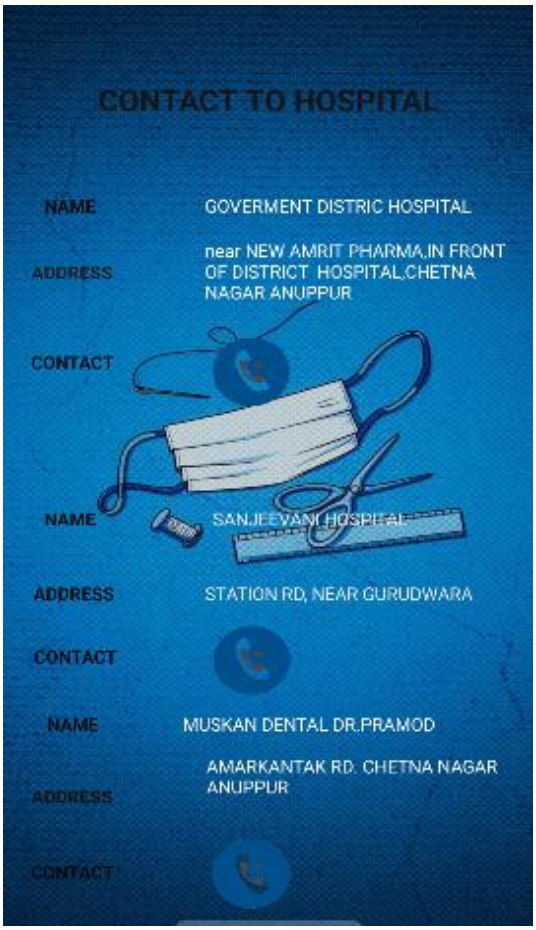
Working of code-homepage.xml

```
package com.example.anuppurutilites;

import androidx.appcompat.app.AppCompatActivity;
```

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

public class homepage extends AppCompatActivity {
   Button b1;
   Button b2;
   Button b3;
   Button b4;
   Button b5;
   @Override
   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_homepage);
      b1 = findViewById(R.id.button6);
      b1.setOnClickListener(new View.OnClickListener() {
         @Override
         public void onClick(View view) {

            startActivity(new Intent(homepage.this,Hospital.class));

         }
      });
      b3=findViewById(R.id.button3);
      b3.setOnClickListener(new View.OnClickListener() {
         @Override
         public void onClick(View view) {
            startActivity(new Intent(homepage.this,ele.class));
         }
      });

     b4=findViewById(R.id.button7);
     b4.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
           startActivity(new Intent(homepage.this,Municipality.class));
        }
     });
     b5=findViewById(R.id.button4);
     b5.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
           startActivity(new Intent(homepage.this, Emergency.class));
        }
     });
     b2=findViewById(R.id.button5);
```

```
    b2.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            startActivity(new Intent(homepage.this, Policestation.class));
        }
    });
  }
}
```

Activity_hospital.xml

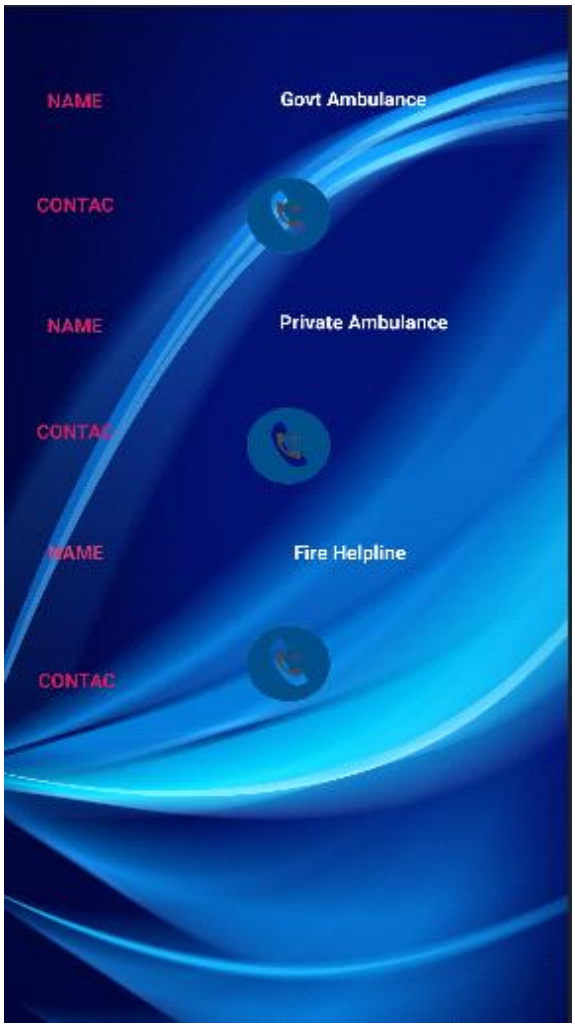


Working of code-hospital.xml

```
package com.example.anuppurutilites;
```

```
import androidx.appcompat.app.AppCompatActivity;

import android.content.Intent;
import android.net.Uri;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.ImageButton;
import android.widget.Toast;

public class Hospital extends AppCompatActivity {
 ImageButton b1 ;
 ImageButton b2;
 ImageButton b3;
   @Override
   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_hospital);
     b1=findViewById(R.id.callBtn);
     b1.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
           Intent callingIntent=new Intent(Intent.ACTION_DIAL);
           callingIntent.setData(Uri.parse("tel:+91 8770786131"));
           startActivity(callingIntent);
        }
     });
     b2=findViewById(R.id.call1);
     b2.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
           Intent callingIntent=new Intent(Intent.ACTION_DIAL);
           callingIntent.setData(Uri.parse("tel:+91 8770786131"));
           startActivity(callingIntent);
        }
     });
     b3=findViewById(R.id.call2);
     b3.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
           Intent callingIntent=new Intent(Intent.ACTION_DIAL);
           callingIntent.setData(Uri.parse("tel:+91 8770786131"));
           startActivity(callingIntent);
        }
     });
   }
```

Activity_emergency.xml



Working Code_emergency.xml

```
public class Emergency extends AppCompatActivity {
   ImageButton B1;
   ImageButton B2;
   ImageButton B3;
   @Override
   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_emergrncy);
      B1=findViewById(R.id.CALL1);
      B1.setOnClickListener(new View.OnClickListener() {
```

```
        @Override
        public void onClick(View view) {
            Intent callingIntent=new Intent(Intent.ACTION_DIAL);
            callingIntent.setData(Uri.parse("tel:+918770786131"));
            startActivity(callingIntent);
        }
    });
    B2=findViewById(R.id.CALL2);
    B2.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            Intent callingIntent =new Intent(Intent.ACTION_DIAL);
            callingIntent.setData(Uri.parse("tel:+918770786131"));
            startActivity(callingIntent);
        }
    });
    B3=findViewById(R.id.CALL3);
    B3.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            Intent callingIntent =new Intent(Intent.ACTION_DIAL);
            callingIntent.setData(Uri.parse("tel:+918770786131"));
            startActivity(callingIntent);
        }
    });
}
```

Policestation.xml

Workingcode policestation.xml

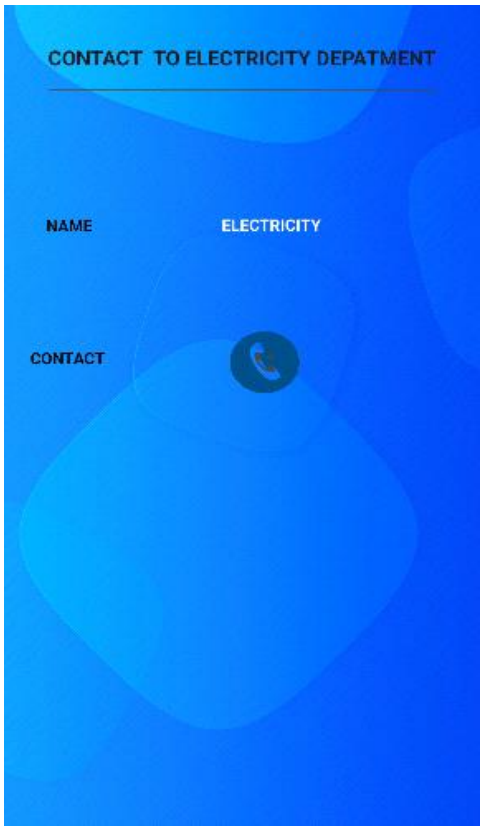
```
package com.example.anuppurutilites;

import androidx.appcompat.app.AppCompatActivity;

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

public class Policestation extends AppCompatActivity {
   Button b1,b2,b3,b4,b5,b6,b7,b8,b9,b10,b11;
   @Override
   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.policestation);
```

```
    b1=findViewById(R.id.button10);
    b1.setOnClickListener(new View.OnClickListener() {
      @Override
      public void onClick(View view) {
        Intent callingIntent=new Intent(Intent.ACTION_DIAL);
        callingIntent.setData(Uri.parse("tel:+91 8770786131"));
        startActivity(callingIntent);
      }
    });
    b2=findViewById(R.id.button8);
    b2.setOnClickListener(new View.OnClickListener() {
      @Override
      public void onClick(View view) {
        Intent callingIntent=new Intent(Intent.ACTION_DIAL);
        callingIntent.setData(Uri.parse("tel:+91 8770786131"));
        startActivity(callingIntent);
      }
    });
    b3=findViewById(R.id.button9);
    b3.setOnClickListener(new View.OnClickListener() {
      @Override
      public void onClick(View view) {
        Intent callingIntent=new Intent(Intent.ACTION_DIAL);
        callingIntent.setData(Uri.parse("tel:+91 8770786131"));
        startActivity(callingIntent);
      }
    });
    b4=findViewById(R.id.button12);
    b4.setOnClickListener(new View.OnClickListener() {
      @Override
      public void onClick(View view) {
        Intent callingIntent=new Intent(Intent.ACTION_DIAL);
        callingIntent.setData(Uri.parse("tel:+91 8770786131"));
        startActivity(callingIntent);
      }
    });
    b5=findViewById(R.id.button6);
    b5.setOnClickListener(new View.OnClickListener() {
      @Override
      public void onClick(View view) {
        Intent callingIntent=new Intent(Intent.ACTION_DIAL);
        callingIntent.setData(Uri.parse("tel:+91 8770786131"));
        startActivity(callingIntent);
      }
    });
    b6=findViewById(R.id.button7);
    b6.setOnClickListener(new View.OnClickListener() {
      @Override
```

```
        public void onClick(View view) {
            Intent callingIntent=new Intent(Intent.ACTION_DIAL);
            callingIntent.setData(Uri.parse("tel:+91 8770786131"));
            startActivity(callingIntent);
        }
    });
    b7=findViewById(R.id.button5);
    b7.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            Intent callingIntent=new Intent(Intent.ACTION_DIAL);
            callingIntent.setData(Uri.parse("tel:+91 8770786131"));
            startActivity(callingIntent);
        }
    });
    b8=findViewById(R.id.button1);
    b8.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            Intent callingIntent=new Intent(Intent.ACTION_DIAL);
            callingIntent.setData(Uri.parse("tel:+91 8770786131"));
            startActivity(callingIntent);
        }
    });
    b9=findViewById(R.id.button17);
    b9.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            Intent callingIntent=new Intent(Intent.ACTION_DIAL);
            callingIntent.setData(Uri.parse("tel:+91 8770786131"));
            startActivity(callingIntent);
        }
    });
    b10=findViewById(R.id.button3);
    b10.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            Intent callingIntent=new Intent(Intent.ACTION_DIAL);
            callingIntent.setData(Uri.parse("tel:+91 8770786131"));
            startActivity(callingIntent);
        }
    });
    b11=findViewById(R.id.button4);
    b11.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            Intent callingIntent=new Intent(Intent.ACTION_DIAL);
            callingIntent.setData(Uri.parse("tel:+91 8770786131"));
```

```
            startActivity(callingIntent);
        }
    });


}
```

Activity_electricity.xml



Working code-electricity.xml

```
package com.example.anuppurutilites;

import androidx.appcompat.app.AppCompatActivity;

import android.content.Intent;
import android.net.Uri;
import android.os.Bundle;
import android.view.View;
import android.widget.ImageButton;

public class ele extends AppCompatActivity {
    ImageButton b1;
    @Override
```

```
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_ele);
        b1=findViewById(R.id.CALL);
        b1.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                Intent callingIntent=new Intent(Intent.ACTION_DIAL);
                callingIntent.setData(Uri.parse("tel:+91 8770786131"));
                startActivity(callingIntent);
            }
        });
    }
}
```

Activity_Municipality.xml

Working code-municipality.xml

```
package com.example.anuppurutilites;

import androidx.appcompat.app.AppCompatActivity;

import android.content.Intent;
import android.net.Uri;
```

```
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.ImageButton;

public class Municipality extends AppCompatActivity {
   ImageButton B1;
   ImageButton B2;
   ImageButton B3;
   @Override
   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_municipality);
      B1=findViewById(R.id.call4);
      B1.setOnClickListener(new View.OnClickListener() {
         @Override
         public void onClick(View view) {
            Intent callingIntent=new Intent(Intent.ACTION_DIAL);
            callingIntent.setData(Uri.parse("tel:+91 8770786131"));
            startActivity(callingIntent);
         }
      });
      B2=findViewById(R.id.call);
      B2.setOnClickListener(new View.OnClickListener() {
         @Override
         public void onClick(View view) {
            Intent callingIntent=new Intent(Intent.ACTION_DIAL);
            callingIntent.setData(Uri.parse("tel:+91 8770786131"));
            startActivity(callingIntent);
         }
      });
      B3=findViewById(R.id.call2);
      B3.setOnClickListener(new View.OnClickListener() {
         @Override
         public void onClick(View view) {
            Intent callingIntent=new Intent(Intent.ACTION_DIAL);
            callingIntent.setData(Uri.parse("tel:+91 8770786131"));
            startActivity(callingIntent);
         }
      });


   }
}
```