CHAPTER 1

INTRODUCTION

1.1 PROBLEM DEFINITION

The Toll Mate application aims to address the challenges faced by highway commuters by providing a user-friendly solution that enhances their experience at toll booths. The primary issues include the lack of real-time information on toll booth status, the need for immediate assistance during emergencies, uncertainty about nearby locations, confusion regarding toll pricing, and a general lack of awareness about toll gate operations. The application seeks to solve these problems by offering features such as a real-time status check for toll booths, helpline assistance with emergency contact numbers, integration with Google Maps for nearby locations, a transparent pricing chart for tolls, and educational videos to increase user understanding of toll gate procedures. By doing so, Toll Mate aspires to improve overall user satisfaction and safety during highway travels, providing a comprehensive and efficient tool for both regular and long-distance commuters.

1.2 OBJECTIVES

The objectives of this application would include

1. Enhance User Convenience:
   Streamline the toll booth experience for users by providing a one-stop solution for real-time status checks, nearby locations, and toll pricing information.

2. Ensure User Safety:
   Improve user safety by offering quick access to helpline numbers for emergency assistance,ensuring users feel secure and supported during their highway journeys.

3. Increase Transparency:
   Promote transparency in toll booth operations by presenting a clear and detailed pricing chart, enabling users to make informed decisions about toll payments.

4. Facilitate Navigation:
   Integrate Google Maps functionality to help users easily identify and navigate to nearby locations such as gas stations, rest areas, and essential services along their route.

5. Educate Users:
   Enhance user understanding of toll gate procedures and regulations through informative and engaging educational videos, contributing to a more informed and confident user base.

1.3 METHODOLOGY TO BE FOLLOWED

GUI has been created using XML, Java and Manifest files.

The flow of the application goes as follows . The user gets started with the Toll Mate application . It provides four different options.

1. About Page - It shows the user the fasttag status of its vehicle. It also provides us with helpline numbers to dial directly.

2. Locations Page - It shows us nearby toll gates, restaurants and hospitals which are connected to google maps.

3. Pricings Page - It shows a chart of prices to be paid at the tollgate.

4. Videos Page - It provides us with educational videos about fasttag to improve our knowledge.

1.4 HARDWARE AND SOFTWARE REQUIREMENTS

➢ Hardware specifications:

  A personal computer/laptop

  Processor :Intel(r)core i7 v pro

  RAM : 8GB

  Operating system : Windows 10 and above

➢ Software specifications:

  Android Studio 2021.3.1

  Java 8 and above

CHAPTER 2

FUNDAMENTALS OF APPLICATION DEVELOPMENT

2.1 ANDROID PROGRAMMING LANGUAGES

Android application development relies on a combination of programming languages to handle both the user interface (front-end) and the underlying functionality (back-end). The primary languages used in Android development are JAVA, C++, XML (Extension Markup Language), and, more recently, Kotlin. Each language serves a specific purpose in the development process, contributing to the overall functionality and design of the application.

- JAVA:

  Java is one of the primary and most widely used programming languages for Android development. It serves as the main back-end language, handling the application's logic, data manipulation, and overall functionality.

  Java is an object-oriented programming (OOP) language, allowing developers to structure their code in a modular and organized manner. They are known for their platform independence, meaning they can run on any device with the Java Virtual Machine (JVM).

  Java comes with a rich set of libraries that simplify common programming tasks, reducing the amount of code developers need to write from scratch.

- XML (Extension Markup Language):

  XML is used in Android primarily for defining the layout, structure, and presentation of the user interface. It serves as the front-end language, focusing on design elements rather than functionality.

  XML is a markup language that uses tags to define elements and their attributes, making it easy to structure and describe data. It is human-readable and easy to understand, facilitating collaboration between designers and developers.

- AndroidManifest.xml:

  The AndroidManifest.xml file is a crucial configuration file for an Android app. It contains essential information about the app, including its package name, version, permissions, activities, services, and broadcast receivers. The manifest file is where you declare the permissions your app requires to function, such as accessing the internet, using the camera, or reading device storage.

2.2 ANDROID COMPONENTS

Android components are fundamental building blocks that define different aspects of an Android application's behavior and functionality. Each component plays a unique role and has a specific lifecycle. The four major app components are:

- Activities:
  Activities represent the user interface and handle user interactions with the screen. Each screen in an Android app is typically associated with an activity. Activities form the backbone of the application's UI, managing the presentation of information and user interactions.

- Lifecycle: Activities have a well-defined lifecycle, including methods such as onCreate, onStart, onResume, onPause, onStop, onRestart, and onDestroy. These methods allow developers to manage the state of an activity throughout its existence.

- Services:
  Services perform background operations independently of the user interface. They handle long-running tasks, such as playing music in the background or downloading data from the internet. Services run in the background, ensuring that the app's functionality continues even when the user is not actively interacting with it.

- Broadcast Receivers:
  Broadcast Receivers respond to system-wide broadcast announcements or messages sent by other applications. For example, a broadcast receiver can react to low battery notifications and trigger actions in response. They enable communication between different parts of an application or even between different applications.

- Content Providers:
  Content Providers manage and transfer data between applications. They act as a bridge, allowing one application to request and access data from another application's database or content repository. Content providers are essential for sharing data securely between different apps.

Understanding and effectively utilizing these four major components—Activities, Services, Broadcast Receivers, and Content Providers—are crucial for developing robust and interactive Android applications. Developers must implement these components in harmony to create a seamless and well-functioning user experience.

2.3 ACTIVITY LIFECYCLE

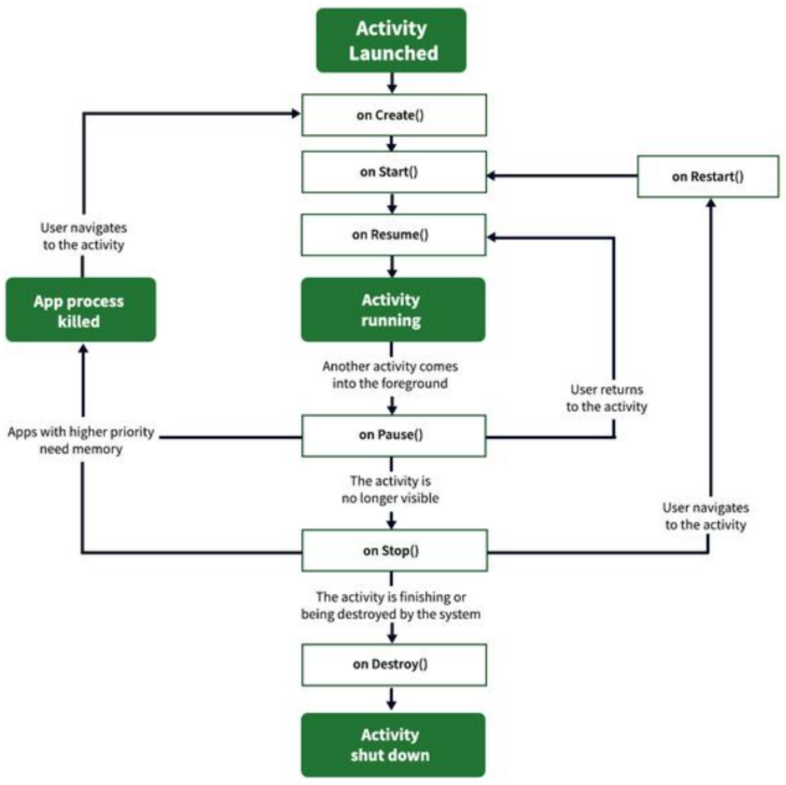

Activity Launched
on Create()
on Start()
on Restart()
User navigates to the activity
on Resume()
App process killed
Activity running
Another activity comes into the foreground
User returns to the activity
Apps with higher priority need memory
on Pause()
The activity is no longer visible
User navigates to the activity
on Stop()
The activity is finishing or being destroyed by the system
on Destroy()
Activity shut down

Fig 2.1 Activity Lifecycle

The lifecycle of an activity in an Android app defines the sequence of states an activity goes through, from its creation to its destruction. This lifecycle is crucial for managing the behavior and resources associated with an activity. The diagram illustrating the activity lifecycle demonstrates the transitions between these states.

- OnCreate: The OnCreate state is triggered when the activity is first created. In this phase, developers initialize essential components, set up the user interface, and perform any necessary setup operations.

- OnStart: The OnStart state is called when the activity becomes visible to the user. At this point, the activity is about to become interactive but may not yet be in the foreground.

- OnResume: The OnResume state is invoked when the activity starts to interact with the user. This is the point at which the activity is in the foreground and actively handling user input.

- OnPause: OnPause is called when the activity is no longer visible to the user but has not been stopped or destroyed. This state is often used to save unsaved changes or pause ongoing processes.

- OnStop: The OnStop state is reached when the activity is no longer visible. This could happen when the user navigates to another activity or when the activity is sent to the background. Resources that are not needed while the activity is not visible can be released in this state.

- OnRestart: OnRestart is called when an activity that was previously stopped is being restarted. This occurs after OnStop but before OnStart, indicating that the activity is resuming from a stopped state.

- OnDestroy: The OnDestroy state marks the end of an activity's lifecycle. It is called when the activity is about to be closed or destroyed. Developers use this state to release resources, unregister listeners, or perform cleanup operations.

Understanding the activity lifecycle is essential for managing the flow of an Android app and ensuring proper resource allocation and deallocation at different stages of an activity's existence. Developers leverage these lifecycle methods to create responsive and efficient applications that provide a seamless user experience

CHAPTER 3

ANALYSIS AND DESIGN GOALS

3.1 FLOWCHART

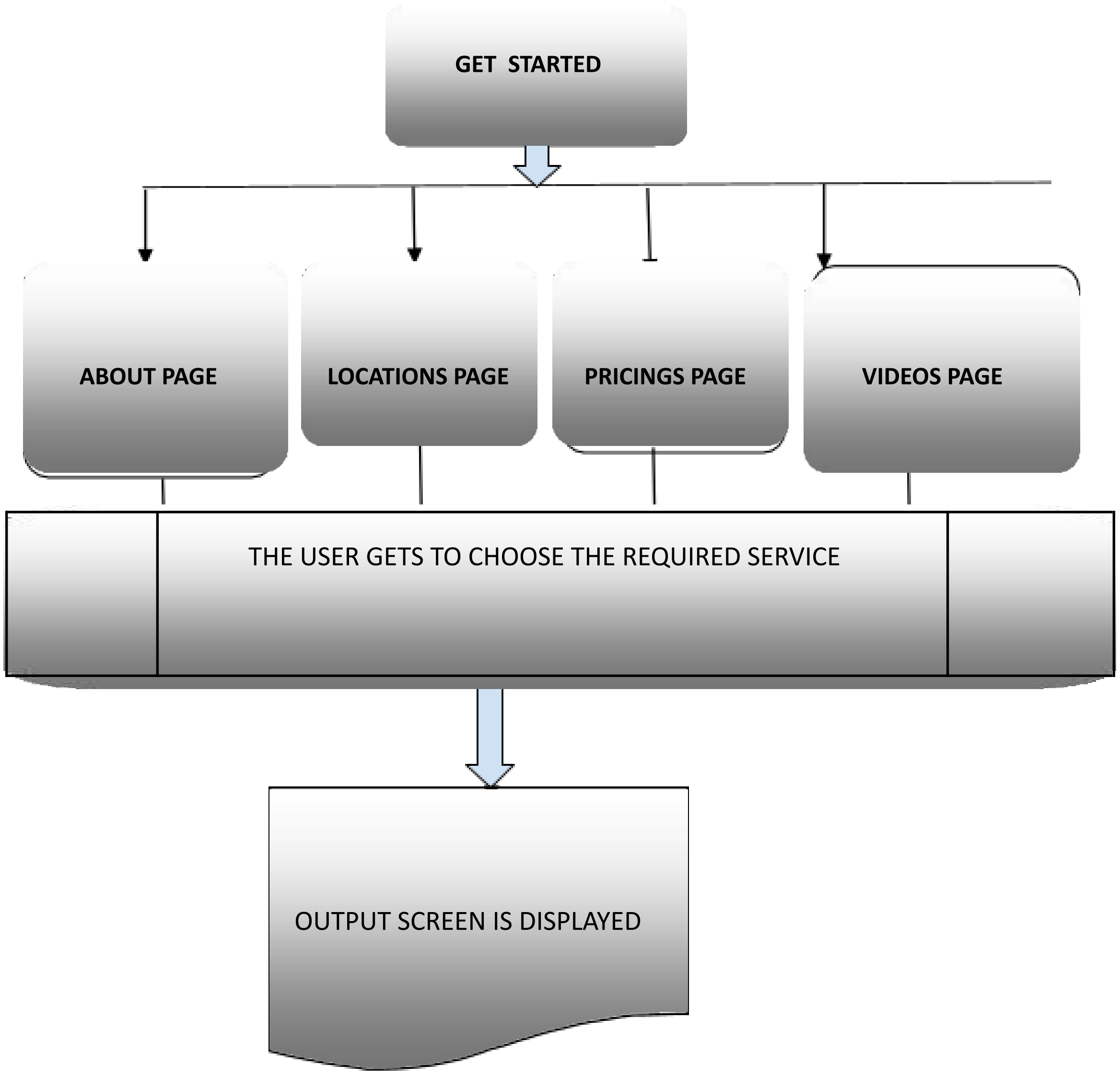

GET STARTED
ABOUT PAGE
LOCATIONS PAGE
PRICINGS PAGE
VIDEOS PAGE
THE USER GETS TO CHOOSE THE REQUIRED SERVICE
OUTPUT SCREEN IS DISPLAYED

Fig 3.1 :- Workflow of the Application

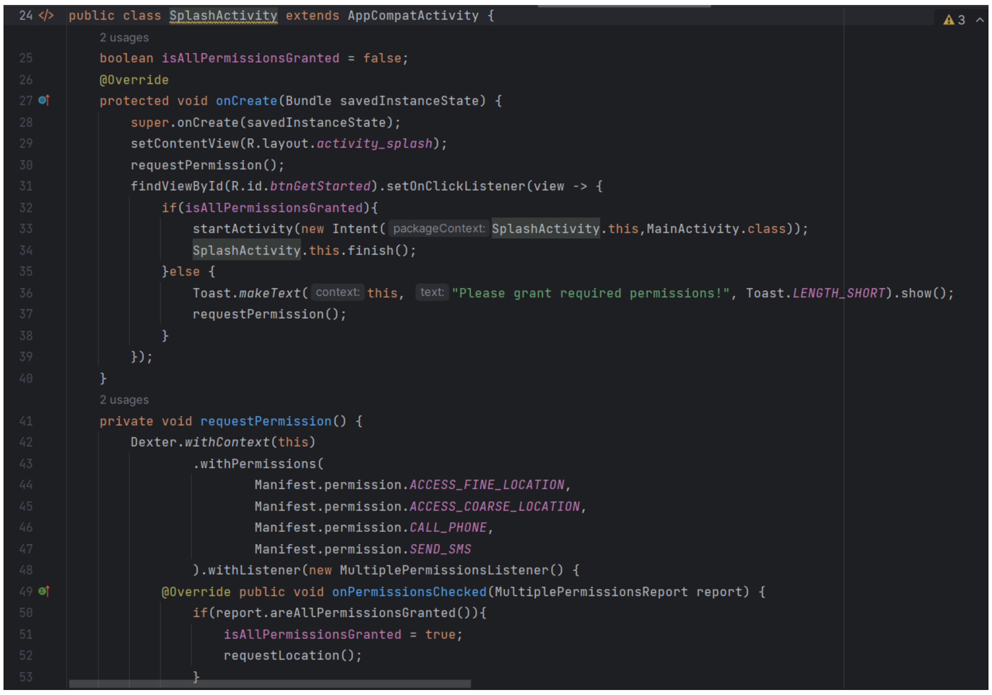
CHAPTER 4

IMPLEMENTATION

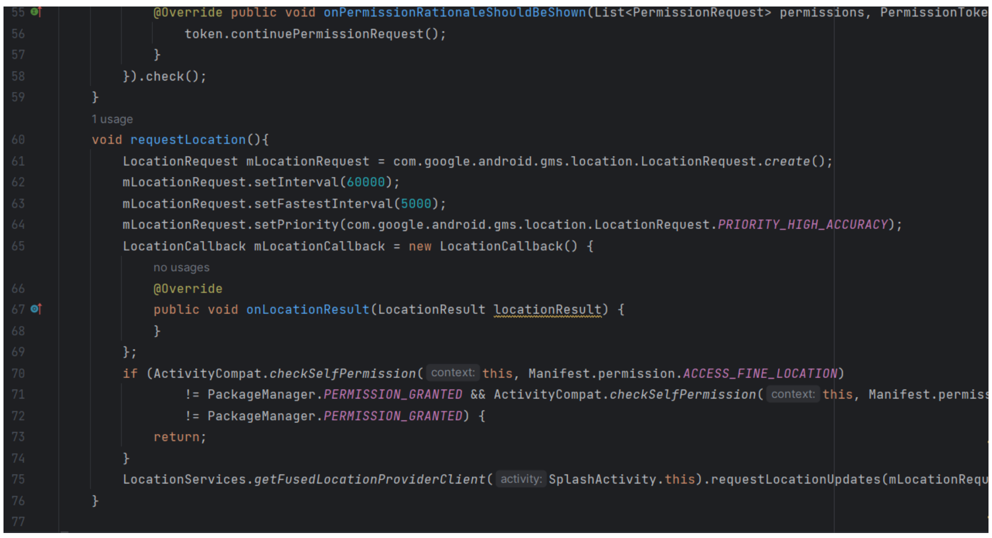
4.1 FUNCTIONALITY OF THE PROJECT

4.1.1 WELCOME PAGE

This page allows the user to get started by providing access to the application to use features like location, sms and dialer.



```
public class SplashActivity extends AppCompatActivity {
    2 usages
    boolean isAllPermissionsGranted = false;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_splash);
        requestPermission();
        findViewById(R.id.btnGetStarted).setOnClickListener(view -> {
            if(isAllPermissionsGranted){
                startActivity(new Intent( packageContext: SplashActivity.this,MainActivity.class));
                SplashActivity.this.finish();
            }else {
                Toast.makeText( context: this,  text: "Please grant required permissions!", Toast.LENGTH_SHORT).show();
                requestPermission();
            }
        });
    }
    2 usages
    private void requestPermission() {
        Dexter.withContext(this)
                .withPermissions(
                        Manifest.permission.ACCESS_FINE_LOCATION,
                        Manifest.permission.ACCESS_COARSE_LOCATION,
                        Manifest.permission.CALL_PHONE,
                        Manifest.permission.SEND_SMS
                ).withListener(new MultiplePermissionsListener() {
            @Override public void onPermissionsChecked(MultiplePermissionsReport report) {
                if(report.areAllPermissionsGranted()){
                    isAllPermissionsGranted = true;
                    requestLocation();
                }
```

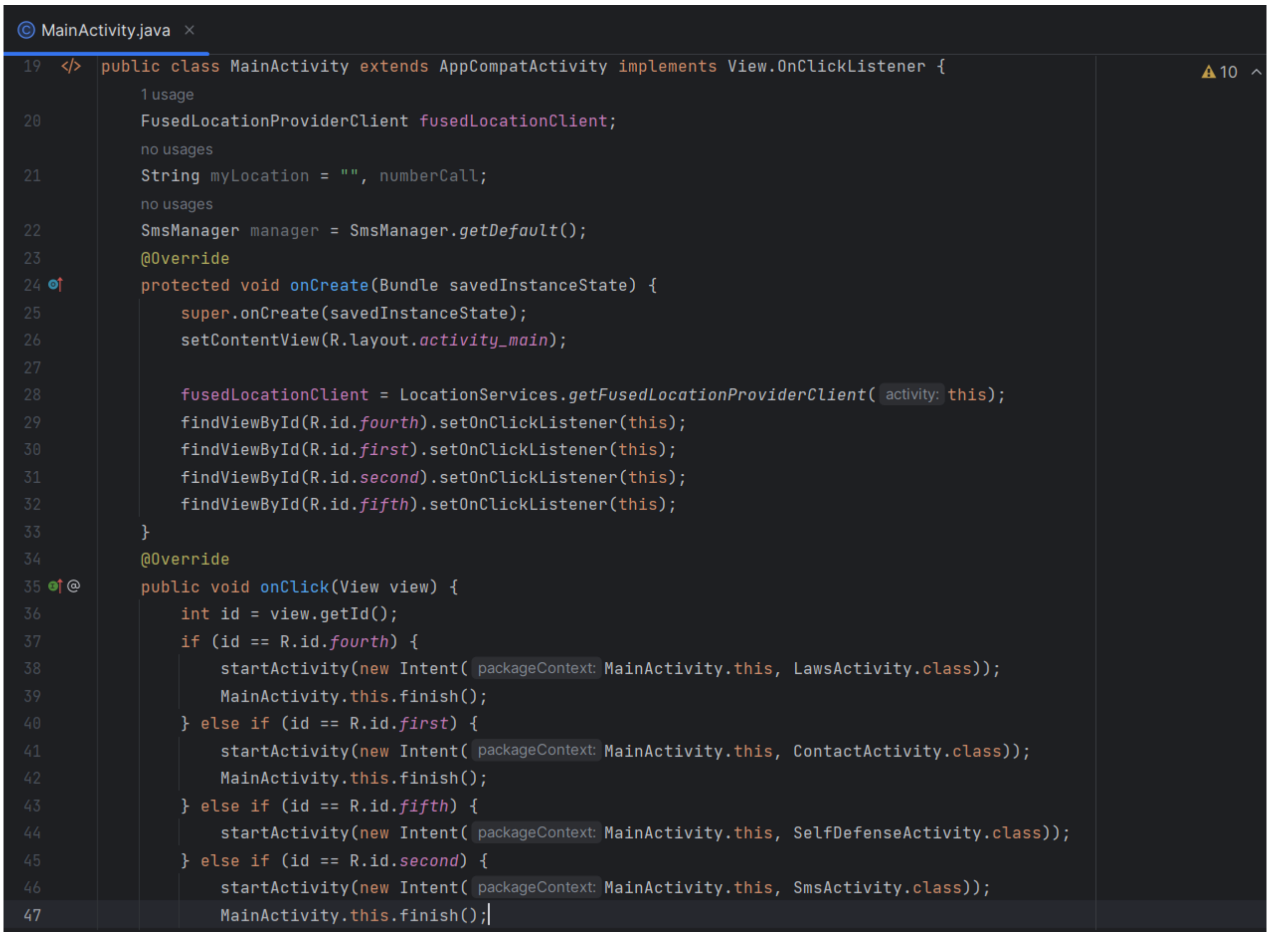


```
            @Override public void onPermissionRationaleShouldBeShown(List<PermissionRequest> permissions, PermissionToke
                token.continuePermissionRequest();
            }
        }).check();
    }
    1 usage
    void requestLocation(){
        LocationRequest mLocationRequest = com.google.android.gms.location.LocationRequest.create();
        mLocationRequest.setInterval(60000);
        mLocationRequest.setFastestInterval(5000);
        mLocationRequest.setPriority(com.google.android.gms.location.LocationRequest.PRIORITY_HIGH_ACCURACY);
        LocationCallback mLocationCallback = new LocationCallback() {
            no usages
            @Override
            public void onLocationResult(LocationResult locationResult) {
            }
        };
        if (ActivityCompat.checkSelfPermission( context: this, Manifest.permission.ACCESS_FINE_LOCATION)
                != PackageManager.PERMISSION_GRANTED && ActivityCompat.checkSelfPermission( context: this, Manifest.permiss
                != PackageManager.PERMISSION_GRANTED) {
            return;
        }
        LocationServices.getFusedLocationProviderClient( activity: SplashActivity.this).requestLocationUpdates(mLocationRequ
    }
```

Fig 4.1 Signup page

4.1.2 Home Page

This page is the home page of the application where the other features like locations, helplines, pricing chart and videos can be accessed through.



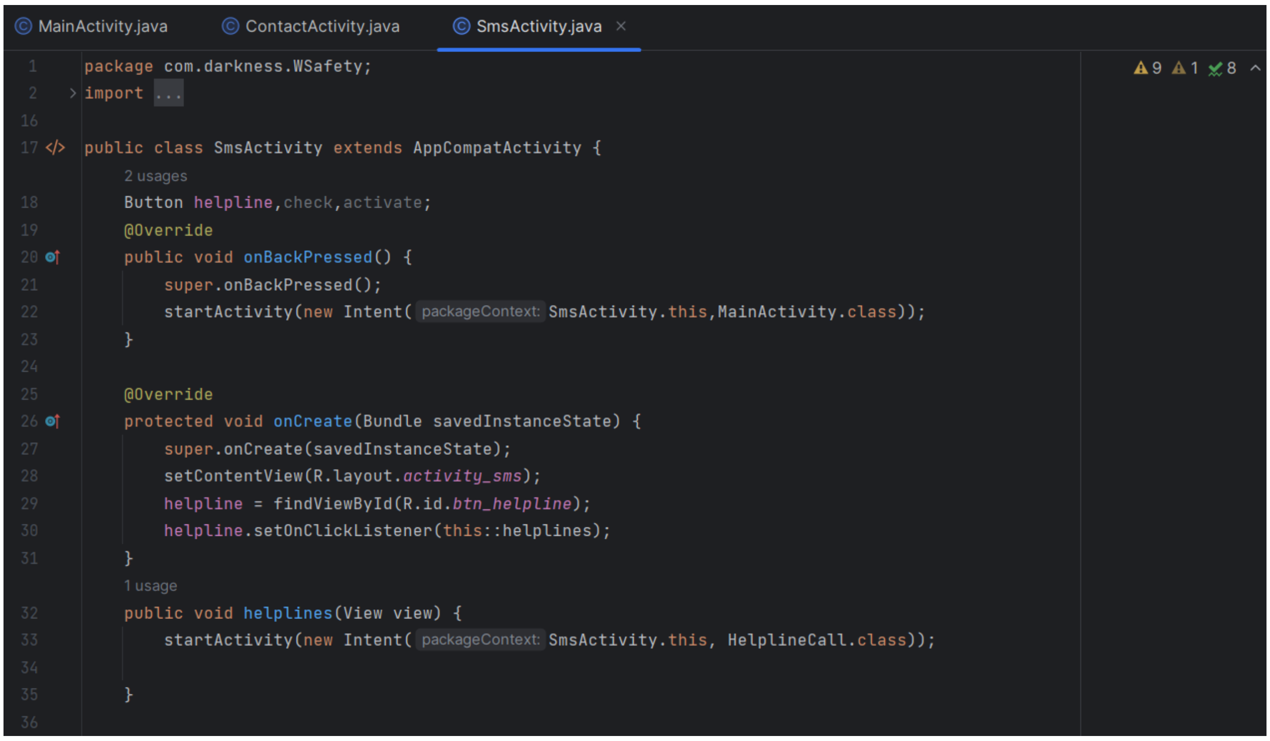
MainActivity.java

```java
public class MainActivity extends AppCompatActivity implements View.OnClickListener {
    1 usage
    FusedLocationProviderClient fusedLocationClient;
    no usages
    String myLocation = "", numberCall;
    no usages
    SmsManager manager = SmsManager.getDefault();
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        fusedLocationClient = LocationServices.getFusedLocationProviderClient( activity: this);
        findViewById(R.id.fourth).setOnClickListener(this);
        findViewById(R.id.first).setOnClickListener(this);
        findViewById(R.id.second).setOnClickListener(this);
        findViewById(R.id.fifth).setOnClickListener(this);
    }
    @Override
    public void onClick(View view) {
        int id = view.getId();
        if (id == R.id.fourth) {
            startActivity(new Intent( packageContext: MainActivity.this, LawsActivity.class));
            MainActivity.this.finish();
        } else if (id == R.id.first) {
            startActivity(new Intent( packageContext: MainActivity.this, ContactActivity.class));
            MainActivity.this.finish();
        } else if (id == R.id.fifth) {
            startActivity(new Intent( packageContext: MainActivity.this, SelfDefenseActivity.class));
        } else if (id == R.id.second) {
            startActivity(new Intent( packageContext: MainActivity.this, SmsActivity.class));
            MainActivity.this.finish();
```

Fig 4.2 Home page

4.1.3 About Page

This page helps the user check the fast tag status of their vehicle once they activate their account with the help of their username and vehicle registration number .

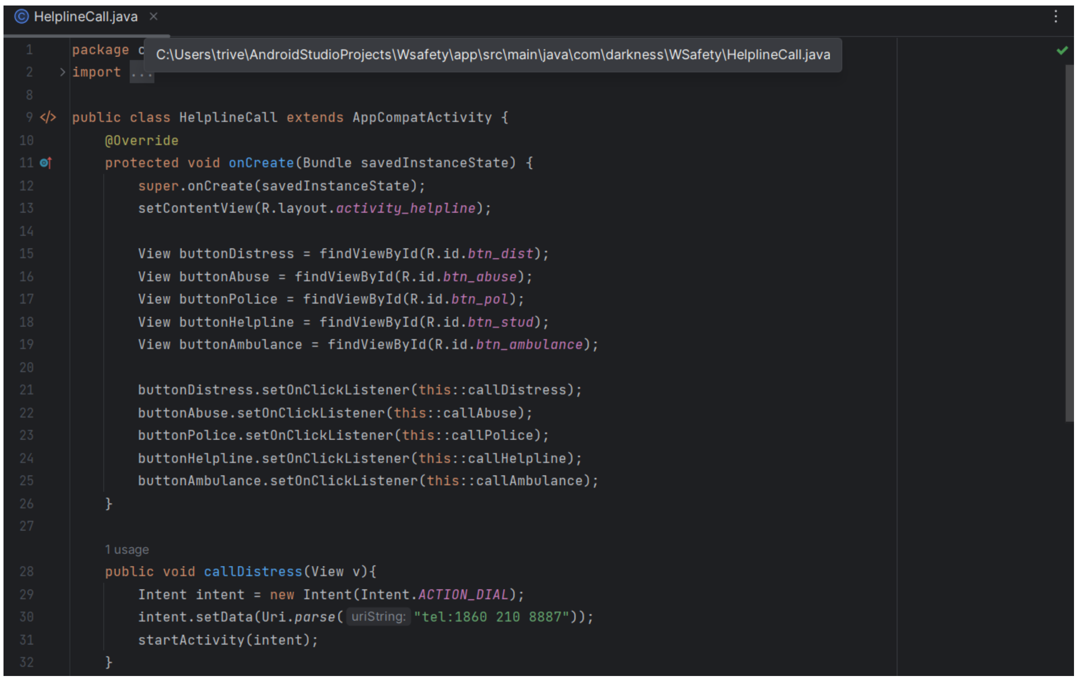

MainActivity.java ContactActivity.java SmsActivity.java

```java
package com.darkness.WSafety;
import ...

public class SmsActivity extends AppCompatActivity {
    2 usages
    Button helpline,check,activate;
    @Override
    public void onBackPressed() {
        super.onBackPressed();
        startActivity(new Intent( packageContext: SmsActivity.this,MainActivity.class));
    }

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_sms);
        helpline = findViewById(R.id.btn_helpline);
        helpline.setOnClickListener(this::helplines);
    }
    1 usage
    public void helplines(View view) {
        startActivity(new Intent( packageContext: SmsActivity.this, HelplineCall.class));

    }
```

Fig 4.3 About page

4.1.4 Helpline Page

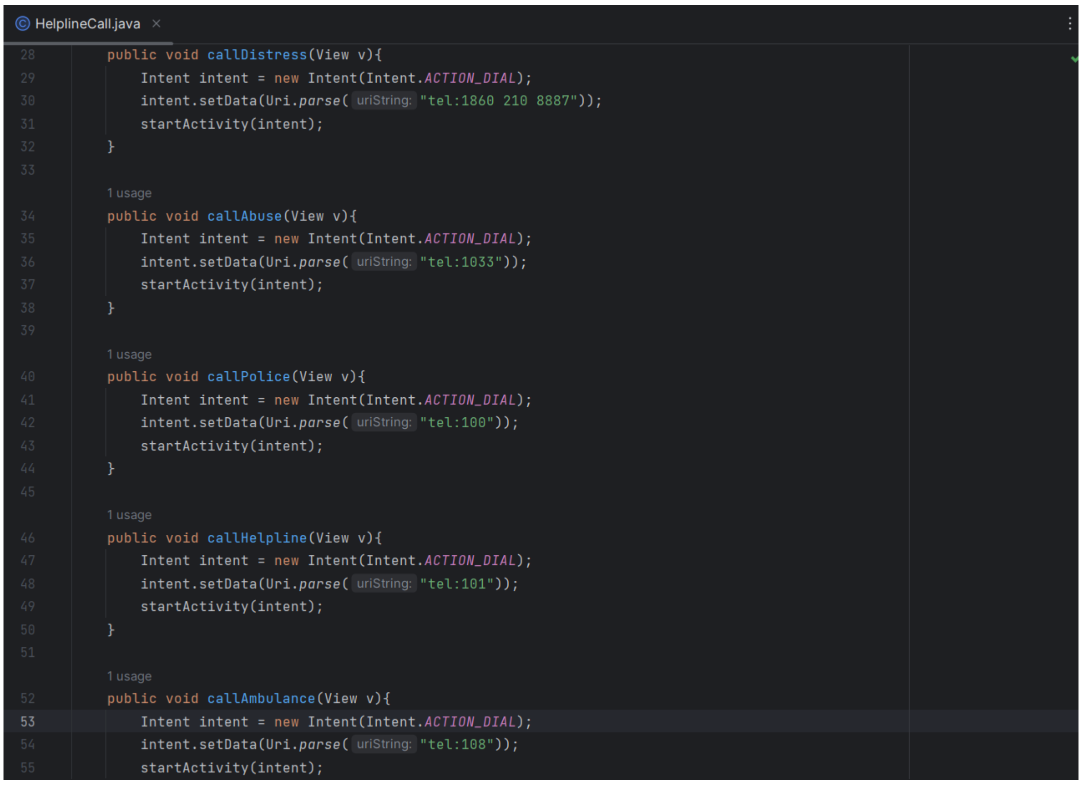
This page helps the user place calls directly to important dialer numbers like the ambulance, fire emergency, fast tag assistance, police, etc.



HelplineCall.java

C:\Users\trive\AndroidStudioProjects\Wsafety\app\src\main\java\com\darkness\WSafety\HelplineCall.java

```java
package c
import ...

public class HelplineCall extends AppCompatActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_helpline);

        View buttonDistress = findViewById(R.id.btn_dist);
        View buttonAbuse = findViewById(R.id.btn_abuse);
        View buttonPolice = findViewById(R.id.btn_pol);
        View buttonHelpline = findViewById(R.id.btn_stud);
        View buttonAmbulance = findViewById(R.id.btn_ambulance);

        buttonDistress.setOnClickListener(this::callDistress);
        buttonAbuse.setOnClickListener(this::callAbuse);
        buttonPolice.setOnClickListener(this::callPolice);
        buttonHelpline.setOnClickListener(this::callHelpline);
        buttonAmbulance.setOnClickListener(this::callAmbulance);
    }

    1 usage
    public void callDistress(View v){
        Intent intent = new Intent(Intent.ACTION_DIAL);
        intent.setData(Uri.parse( uriString: "tel:1860 210 8887"));
        startActivity(intent);
    }
```



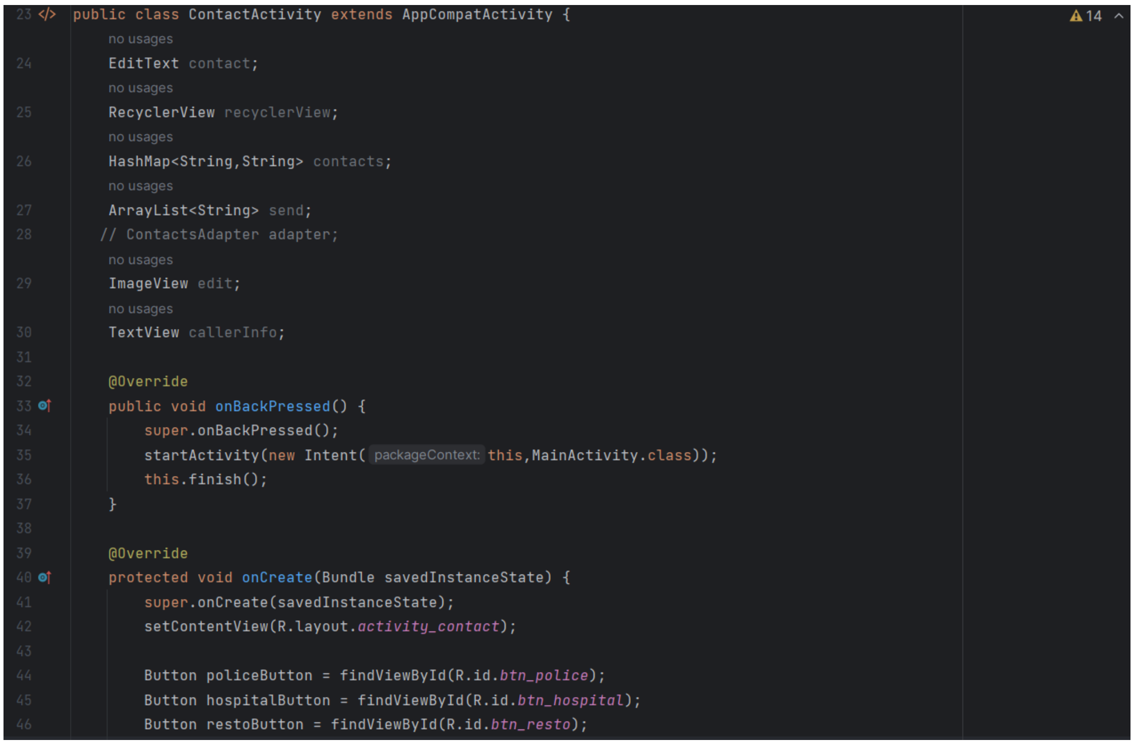
HelplineCall.java

```java
    public void callDistress(View v){
        Intent intent = new Intent(Intent.ACTION_DIAL);
        intent.setData(Uri.parse( uriString: "tel:1860 210 8887"));
        startActivity(intent);
    }

    1 usage
    public void callAbuse(View v){
        Intent intent = new Intent(Intent.ACTION_DIAL);
        intent.setData(Uri.parse( uriString: "tel:1033"));
        startActivity(intent);
    }

    1 usage
    public void callPolice(View v){
        Intent intent = new Intent(Intent.ACTION_DIAL);
        intent.setData(Uri.parse( uriString: "tel:100"));
        startActivity(intent);
    }

    1 usage
    public void callHelpline(View v){
        Intent intent = new Intent(Intent.ACTION_DIAL);
        intent.setData(Uri.parse( uriString: "tel:101"));
        startActivity(intent);
    }

    1 usage
    public void callAmbulance(View v){
        Intent intent = new Intent(Intent.ACTION_DIAL);
        intent.setData(Uri.parse( uriString: "tel:108"));
        startActivity(intent);
```

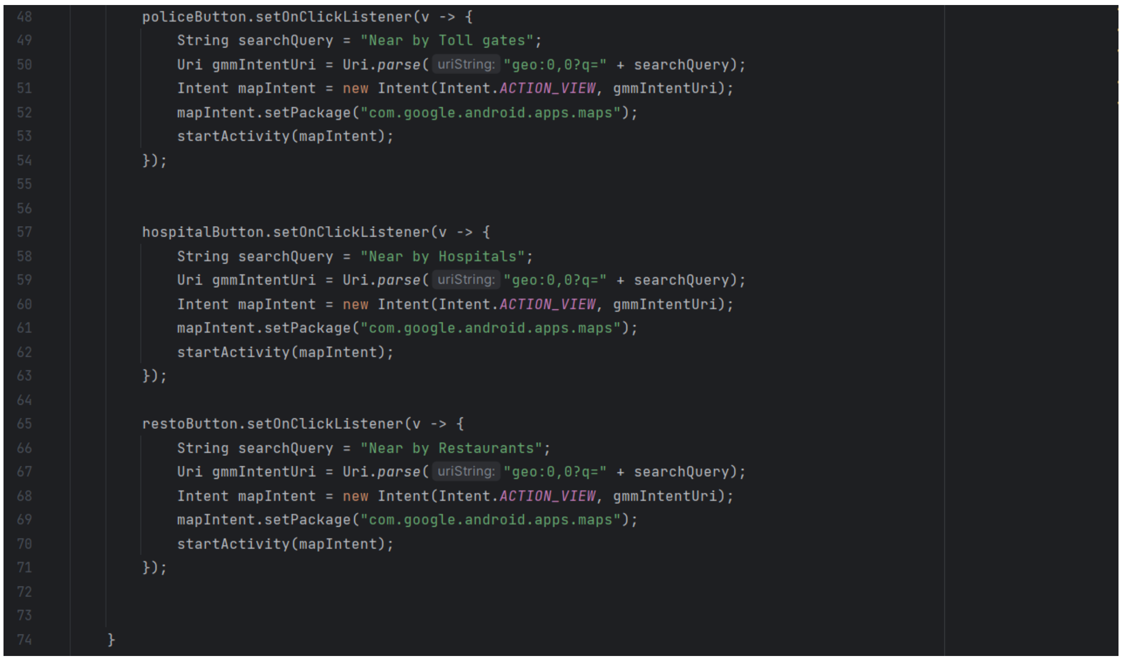
Fig 4.4 Helpline Page

4.1.5 Locations page

This page helps the user with nearby toll gates, nearby hospitals and nearby restaurants according to your present location as it is connected to android’s google maps.



```
public class ContactActivity extends AppCompatActivity {
    no usages
    EditText contact;
    no usages
    RecyclerView recyclerView;
    no usages
    HashMap<String,String> contacts;
    no usages
    ArrayList<String> send;
   // ContactsAdapter adapter;
    no usages
    ImageView edit;
    no usages
    TextView callerInfo;

    @Override
    public void onBackPressed() {
        super.onBackPressed();
        startActivity(new Intent( packageContext: this,MainActivity.class));
        this.finish();
    }

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

        Button policeButton = findViewById(R.id.btn_police);
        Button hospitalButton = findViewById(R.id.btn_hospital);
        Button restoButton = findViewById(R.id.btn_resto);
```

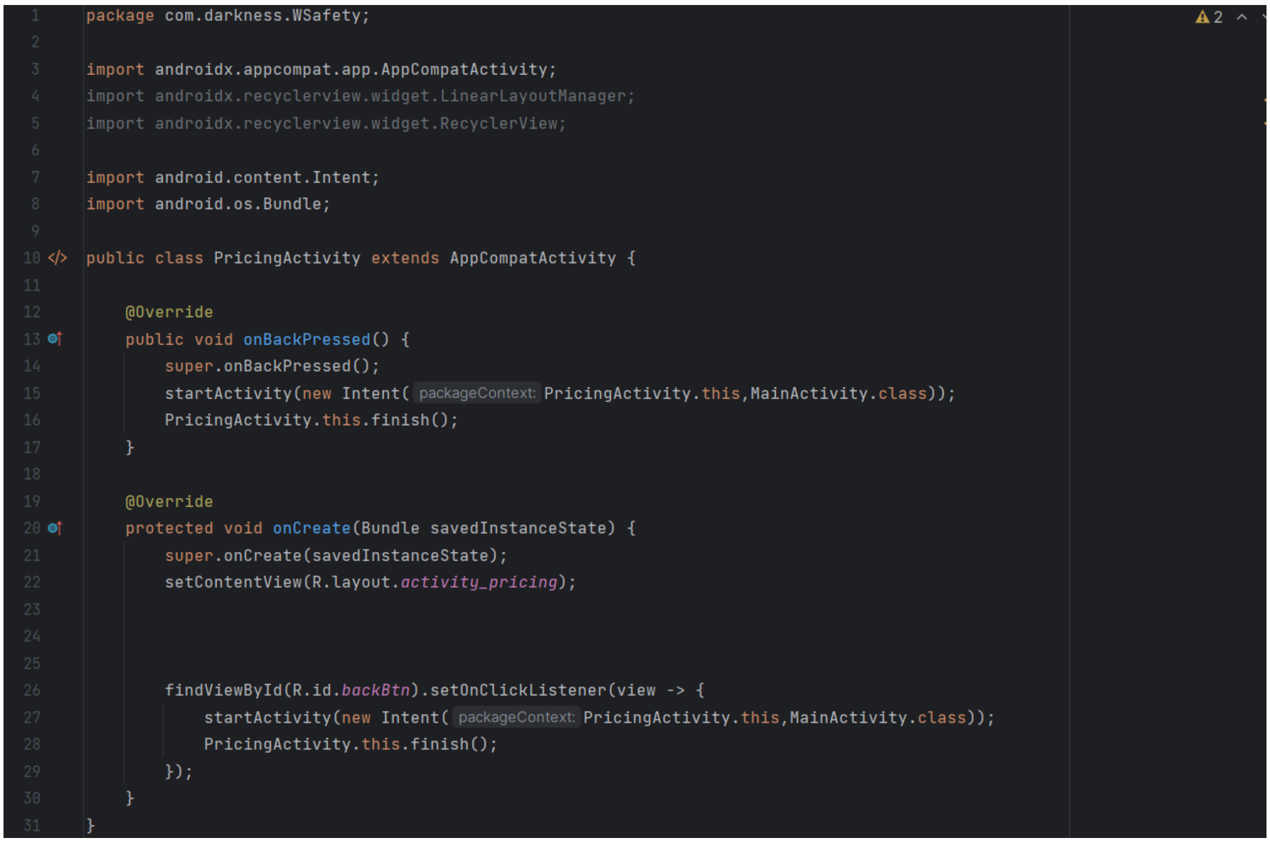


```
        policeButton.setOnClickListener(v -> {
            String searchQuery = "Near by Toll gates";
            Uri gmmIntentUri = Uri.parse( uriString: "geo:0,0?q=" + searchQuery);
            Intent mapIntent = new Intent(Intent.ACTION_VIEW, gmmIntentUri);
            mapIntent.setPackage("com.google.android.apps.maps");
            startActivity(mapIntent);
        });


        hospitalButton.setOnClickListener(v -> {
            String searchQuery = "Near by Hospitals";
            Uri gmmIntentUri = Uri.parse( uriString: "geo:0,0?q=" + searchQuery);
            Intent mapIntent = new Intent(Intent.ACTION_VIEW, gmmIntentUri);
            mapIntent.setPackage("com.google.android.apps.maps");
            startActivity(mapIntent);
        });

        restoButton.setOnClickListener(v -> {
            String searchQuery = "Near by Restaurants";
            Uri gmmIntentUri = Uri.parse( uriString: "geo:0,0?q=" + searchQuery);
            Intent mapIntent = new Intent(Intent.ACTION_VIEW, gmmIntentUri);
            mapIntent.setPackage("com.google.android.apps.maps");
            startActivity(mapIntent);
        });


    }
```

Fig 4.5 Locations Page

4.1.6 Pricing Page

This pricing page displays the prices to pay for a toll gate according to the vehicle manufacturing.

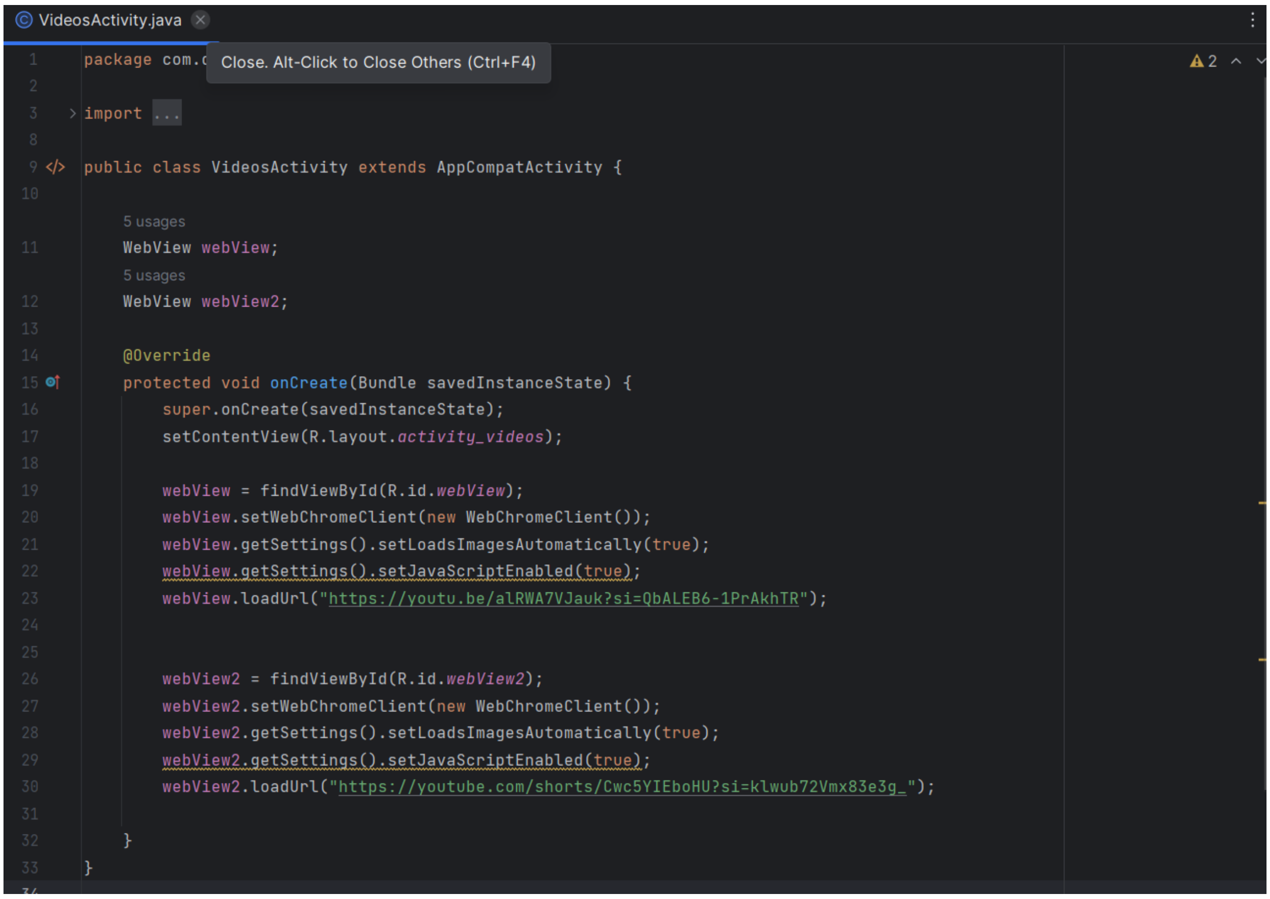


```java
package com.darkness.WSafety;

import androidx.appcompat.app.AppCompatActivity;
import androidx.recyclerview.widget.LinearLayoutManager;
import androidx.recyclerview.widget.RecyclerView;

import android.content.Intent;
import android.os.Bundle;

public class PricingActivity extends AppCompatActivity {

    @Override
    public void onBackPressed() {
        super.onBackPressed();
        startActivity(new Intent( packageContext: PricingActivity.this,MainActivity.class));
        PricingActivity.this.finish();
    }

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



        findViewById(R.id.backBtn).setOnClickListener(view -> {
            startActivity(new Intent( packageContext: PricingActivity.this,MainActivity.class));
            PricingActivity.this.finish();
        });
    }
}
```

Fig 4.6 Pricing Page

4.1.7 Videos Page

This page allows the user to gain knowledge regarding fast tag affixation process as it is connected to a youtube video that plays it.



VideosActivity.java

Close. Alt-Click to Close Others (Ctrl+F4)

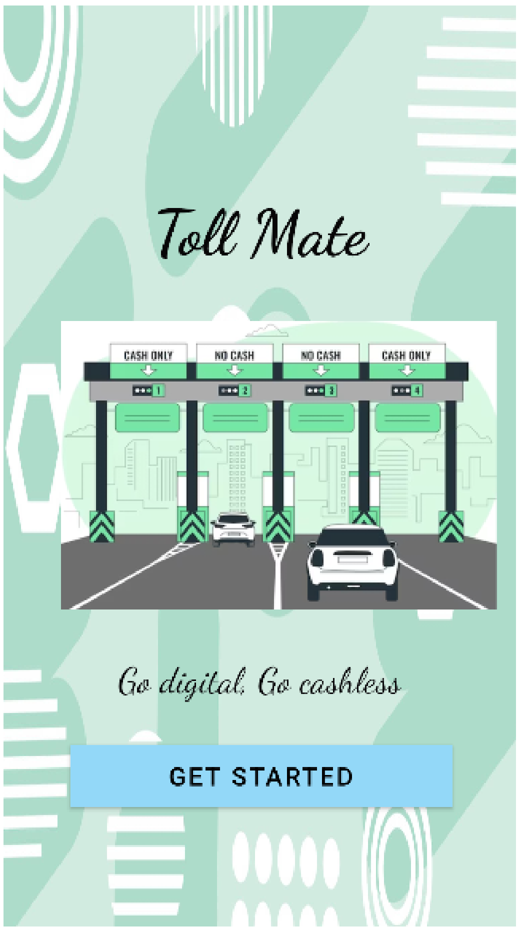
```
package com.c

import ...

public class VideosActivity extends AppCompatActivity {

    5 usages
    WebView webView;
    5 usages
    WebView webView2;

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

        webView = findViewById(R.id.webView);
        webView.setWebChromeClient(new WebChromeClient());
        webView.getSettings().setLoadsImagesAutomatically(true);
        webView.getSettings().setJavaScriptEnabled(true);
        webView.loadUrl("https://youtu.be/alRWA7VJauk?si=QbALEB6-1PrAkhTR");


        webView2 = findViewById(R.id.webView2);
        webView2.setWebChromeClient(new WebChromeClient());
        webView2.getSettings().setLoadsImagesAutomatically(true);
        webView2.getSettings().setJavaScriptEnabled(true);
        webView2.loadUrl("https://youtube.com/shorts/Cwc5YIEboHU?si=klwub72Vmx83e3g_");

    }
}
```

Fig 4.7 Videos Page

CHAPTER 5

SAMPLE OUTPUT

5.1 WELCOME AND HOME PAGE



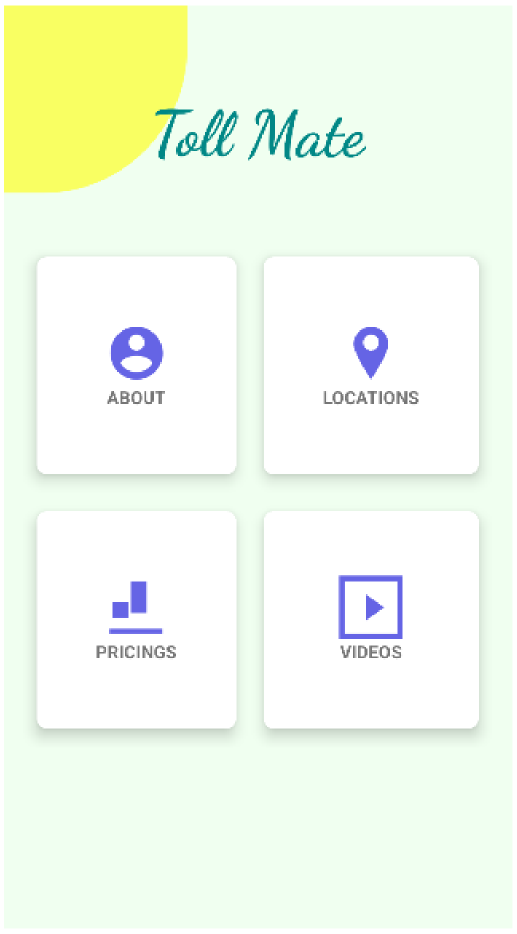


Fig 5.1.1 Welcome page **Fig 5.1.2 Homepage**

5.2 ABOUT PAGE

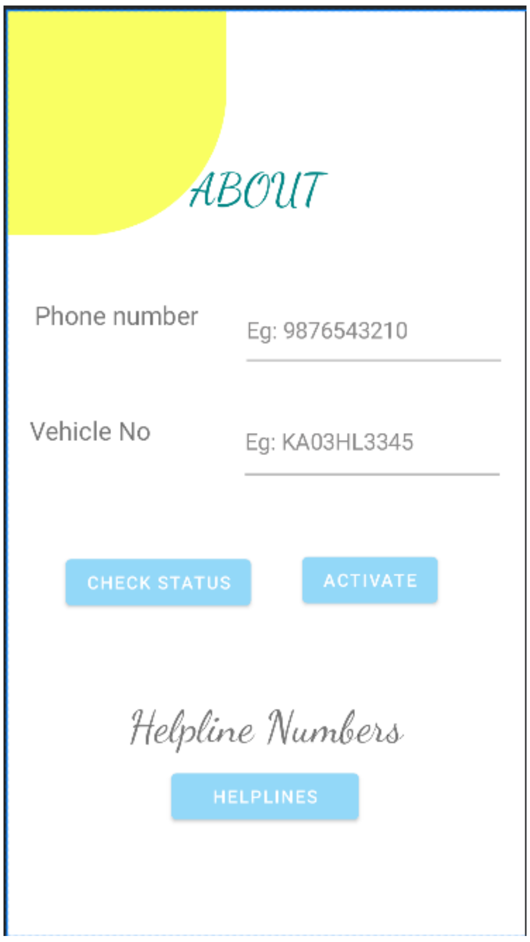
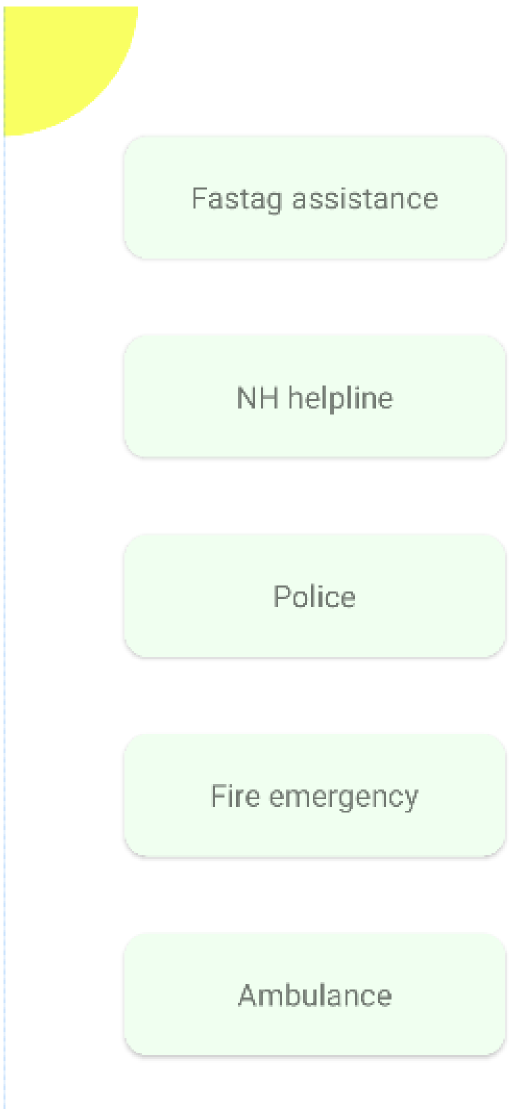


Fig 5.2 About page

5.3 HELPLINE PAGE



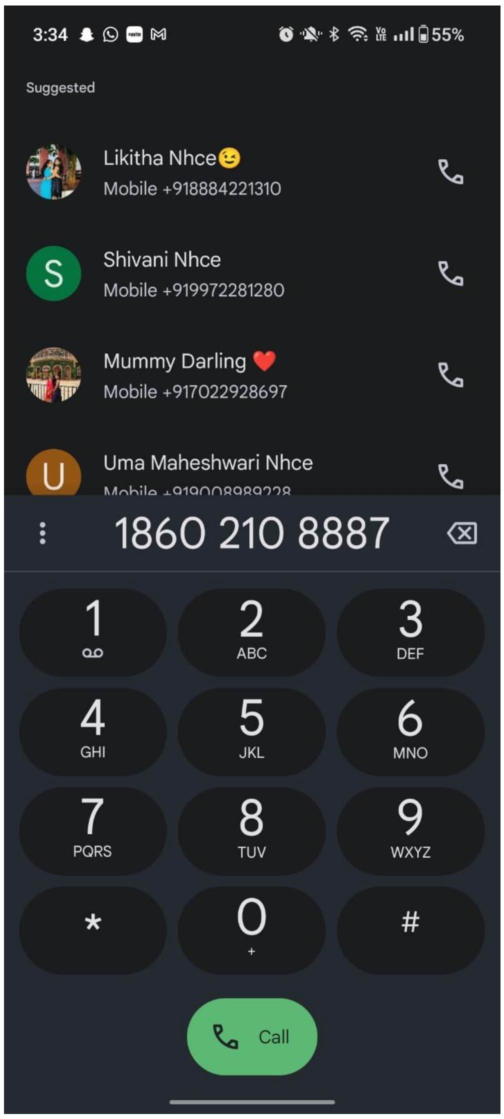


Fig 5.3.1 Helpline page

Fig 5.3.2 Helpline Dialer

5.4 LOCATIONS PAGE

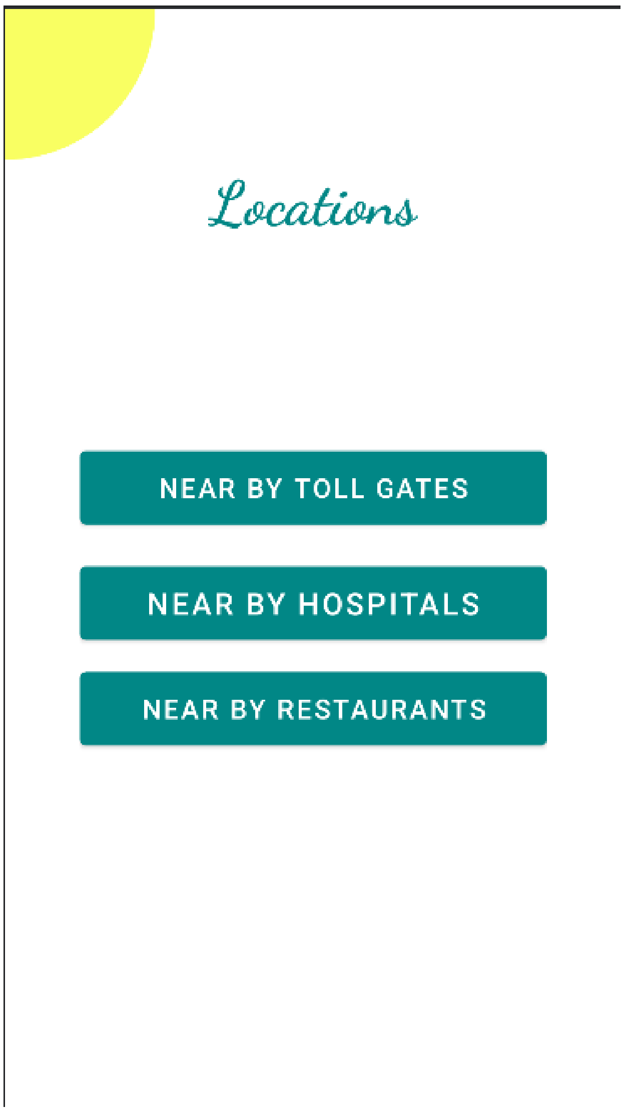
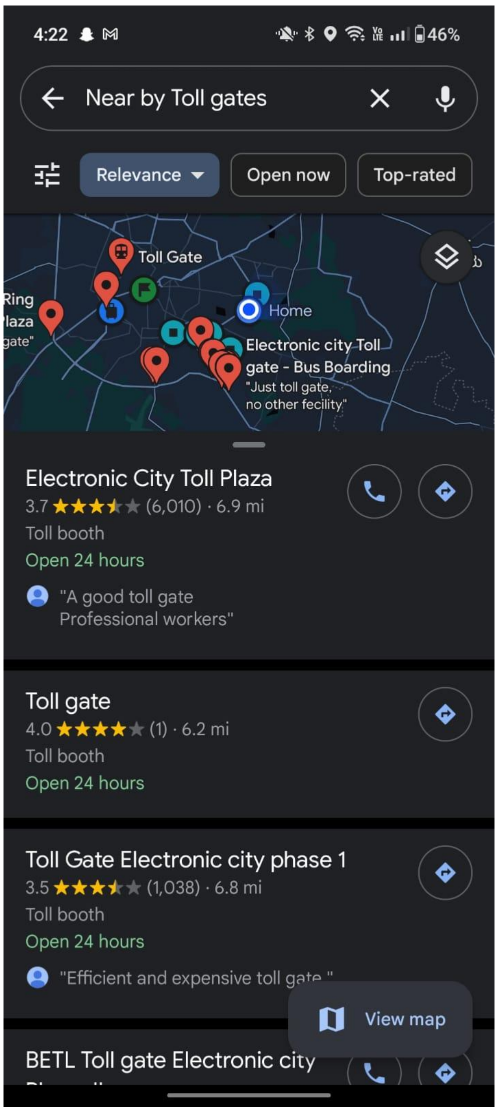


Fig 5.4.1 Location Page


4:22
46%
Near by Toll gates
Relevance
Open now
Top-rated
Toll Gate
Home
Electronic city Toll gate - Bus Boarding
"Just toll gate, no other fecility"
View map

Electronic City Toll Plaza
3.7 ★★★★☆ (6,010) · 6.9 mi
Toll booth
Open 24 hours
"A good toll gate Professional workers"

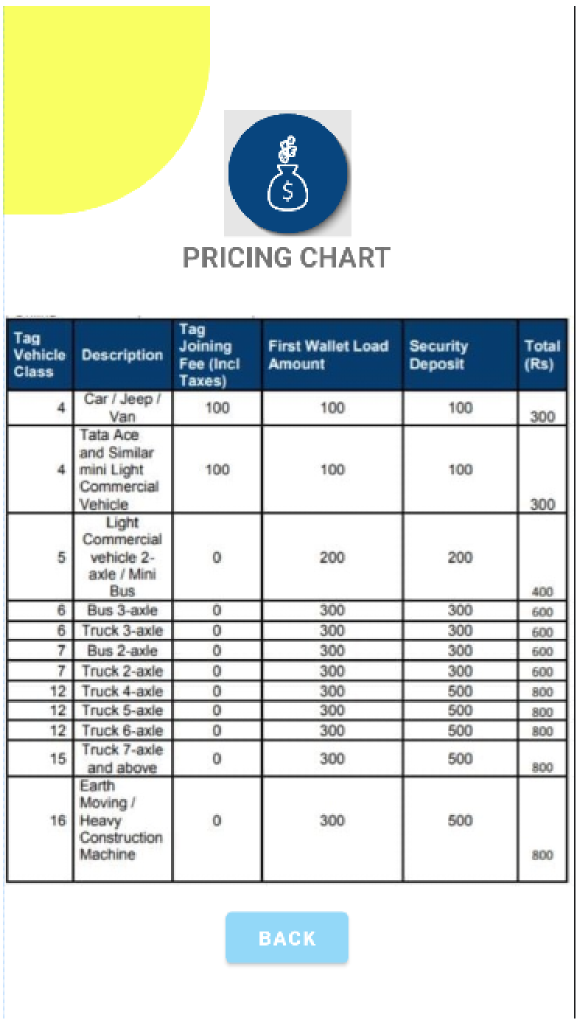
Toll gate
4.0 ★★★★☆ (1) · 6.2 mi
Toll booth
Open 24 hours

Toll Gate Electronic city phase 1
3.5 ★★★★☆ (1,038) · 6.8 mi
Toll booth
Open 24 hours
"Efficient and expensive toll gate "

BETL Toll gate Electronic city

Fig 5.4.2 Maps with nearby tollgates

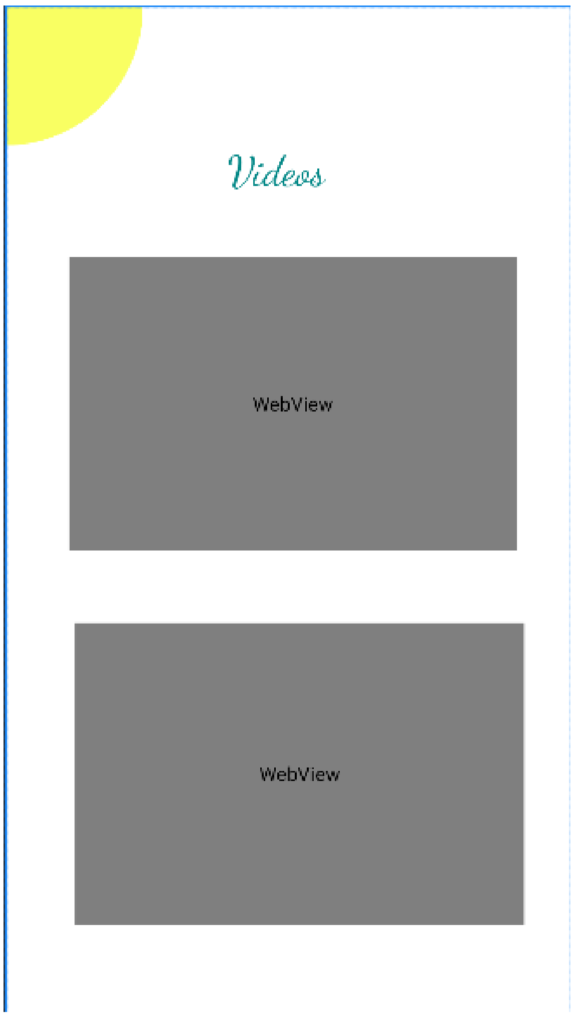
5.5 PRICING PAGE



Tag Vehicle Class	Description	Tag Joining Fee (Incl Taxes)	First Wallet Load Amount	Security Deposit	Total (Rs)
4	Car / Jeep / Van	100	100	100	300
4	Tata Ace and Similar mini Light Commercial Vehicle	100	100	100	300
5	Light Commercial vehicle 2-axle / Mini Bus	0	200	200	400
6	Bus 3-axle	0	300	300	600
6	Truck 3-axle	0	300	300	600
7	Bus 2-axle	0	300	300	600
7	Truck 2-axle	0	300	300	600
12	Truck 4-axle	0	300	500	800
12	Truck 5-axle	0	300	500	800
12	Truck 6-axle	0	300	500	800
15	Truck 7-axle and above	0	300	500	800
16	Earth Moving / Heavy Construction Machine	0	300	500	800

Fig 5.5 Pricing chart

5.6 VIDEOS PAGE



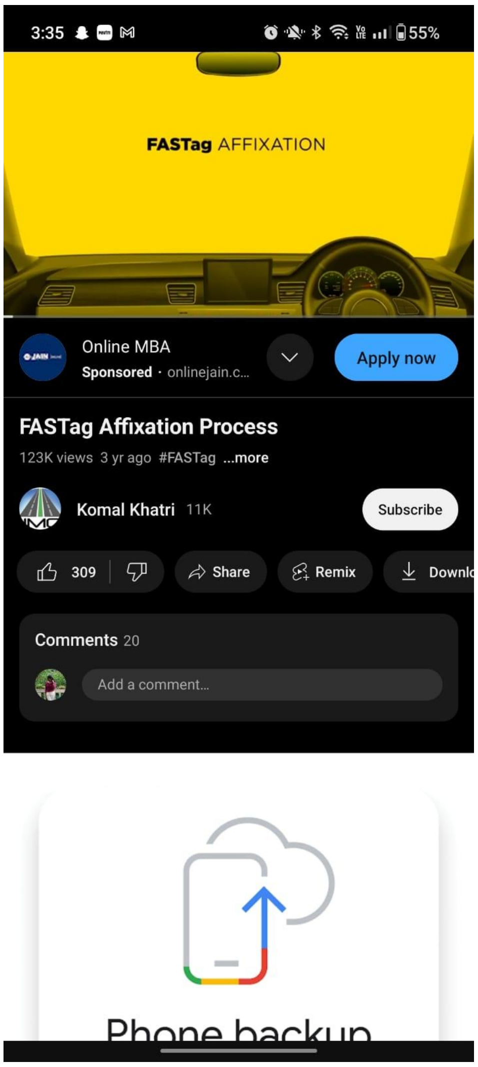


Fig 5.6.1 Videos Page

Fig 5.6.1 Youtube fasttag video

CHAPTER 6

CONCLUSION

In conclusion, the Toll Mate application serves as a comprehensive and user-friendly tool for individuals navigating through toll gates. With its ability to seamlessly handle data and check the status of tolls, users can enjoy a hassle-free travel experience. The incorporation of helpline numbers ensures that assistance is readily available in case of emergencies or unexpected situations on the highway, adding an extra layer of safety and convenience.

The integration with Google Maps for providing nearby locations enhances the user experience by offering real-time information on essential services such as gas stations, rest areas, and dining options. This feature not only aids in trip planning but also contributes to a smoother and more enjoyable journey.

The pricing chart for tolls provides users with transparent and upfront information about the costs associated with their route, helping them make informed decisions and plan their budgets accordingly. This feature promotes financial transparency and minimizes surprises during travel.

In essence, Toll Mate goes beyond being a mere toll status checker and emerges as a holistic travel companion, ensuring users have the necessary tools and information at their fingertips to make their journeys safer, more convenient, and enjoyable. The application's multifaceted approach to addressing the needs of travelers makes it an indispensable tool for those navigating through toll gates and highways.

REFERENCES

[1] www.javatpoint.com

[2] https://www.digitalauthority.me/resources/mobile-app-development-guide/

[3] https://youtu.be/3CiiYOSxAPM?si=ZX39WG4-6d6JzACb

[4] https://firebase.google.com/docs/firestore/?utm_source=studio

[5] https://www.geeksforgeeks.org/android-app-development-fundamentals-for-beginners/