

# **SMART CITY MANAGEMENT SYSTEM**

## **Sathyabama Institute of Science and Technology(Deemed to be University)**

Submitted in partial fulfillment of the requirements for the  
award of Bachelor of Engineering Degree in Computer  
Science and Engineering

By

**DINESH V (Reg. No.38110130)**  
**GOKIL PRADEEP R (Reg. No.38110163)**



**DEPARTMENT OF COMPUTER SCIENCE AND  
ENGINEERING SCHOOL OF COMPUTING  
SATHYABAMA INSTITUTE OF SCIENCE AND  
TECHNOLOGY JEPPIAAR NAGAR, RAJIV GANDHI  
SALAI,  
CHENNAI – 600119, TAMILNADU**

**MARCH - 2022**



**SATHYABAMA**  
INSTITUTE OF SCIENCE AND TECHNOLOGY  
(DEEMED TO BE UNIVERSITY)  
**Accredited with Grade “A” by NAAC**  
(Established under Section 3 of UGC Act, 1956)  
**JEPPIAAR NAGAR, RAJIV GANDHI SALAI, CHENNAI– 600119**  
[www.sathyabamauniversity.ac.in](http://www.sathyabamauniversity.ac.in)



---

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

**BONAFIDE CERTIFICATE**

This is to certify that this Project Report is the bonafide work of **DINESH V (38110130)**, **GOKIL PRADEEP R (38110163)** who carried out the project entitled “**Smart City Management System**” under my supervision from November 2021 to May 2022.

**Internal Guide**

**Dr.S.Rajashree M.E.,Ph.D.,**

**Head of the Department**

**Dr.L.Lakshmanan M.E., Ph.D.**

---

Submitted for Viva voce Examination held on \_\_\_\_\_

**Internal Examiner**

**External Examiner**

## **DECLARATION**

We **DINESH V, GOKIL PRADEEP R** hereby declare that the Project Report entitled **SMART CITY MANAGEMENT SYSTEM** done by me under the guidance of **Dr.S.Rajashree M.E.,Ph.D.**, is submitted in partial fulfillment of the requirements for the award of Bachelor of Engineering degree in Computer Science and Engineering.

**DATE:**

**PLACE: Chennai**

**SIGNATURE OF THE CANDIDATE**

## **ACKNOWLEDGEMENT**

I am pleased to acknowledge my sincere thanks to **Board of Management** of **SATHYABAMA** for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

I convey my thanks to **Dr. T.Sasikala M.E., Ph.D., Dean**, School of Computing , **Dr.S.Vigneshwari M.E., Ph.D.,** and **Dr.L.Lakshmanan M.E., Ph.D.,** Heads of the Department of Computer Science and Engineering for providing me necessary support and details at the right time during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my Project Guide **Dr.S.Rajashree M.E.,Ph.D.,** for his valuable guidance, suggestions and constant encouragement paved way for the successful completion of my project work.

I wish to express my thanks to all Teaching and Non-teaching staff members of the **Department of Computer Science and Engineering** who were helpful in many ways for the completion of the project.

## **ABSTRACT**

Now a days mobile phone is a needful part of the people's life. There is continuously rising in a number of mobile computing applications, concentrate on the people's daily life. In such applications, location dependent systems have been detected as an significant application. Such application which presents the architecture and implementation of such a location is commonly known as Smart City Guide.

The main motive of the project is to explore how to realize a mobile city guide using the Android platform, including a prototype of the city guide. The project uses the research method design science. Through designing and implementing an artifact (that is prototype of city guide), the goal project is reached. Finally, the project is assess in four aspects including platform evaluation, general functional evaluation, scenario evaluation and non-functional evaluation.

The prototype implemented include basic functionalities of city guides such as showing the map, locating points of interest (POIs). Beside, the project has inspect how to combine present technologies like Google Map and the phone application into the prototype. The app comforts a new native in a city by showing information of all the nearby sites that can be used for public access Sites include Hospital Services, Police Station, Main Attraction Of City, Restaurants. As well, the project has investigated non-functional aspects including extendibility, tolerability, and usability. Overall, the project presents a comprehensive unrealized city guide on the new mobile Android platform.

INDEX  
**TABLE OF CONTENTS**

<b>CHAPTER NO</b>	<b>TITLE</b>	<b>PAGE NO</b>
	Abstract	I
	List Of Figures	IV
	List Of Abbreviations	V
1	<b>Introduction</b>	1
2	<b>Literature Review</b>	3
3	<b>Aim And Scope Of The Project</b>	5
	3.1 Aim Of The Project	5
	3.2 Scope Of The Project	5
	3.2.1 Application Programming Interface	6
	3.2.2 Advantages	6
	3.2.3 Disadvantages	7
4	<b>Working Theory Of The Project</b>	8
	4.1 Android Studio	8
	4.2 Android Os	8
	4.3 Java	9
	4.4 Xml	10
	4.5 Intergrated Development Environment	11
	4.6 Android Components	11
5	<b>Implementation And Methodology</b>	14
	5.1 Software Requirement	14
	5.2 Hardware Requirement	14
	5.3 Activity Class	15

	5.4 Activity	15
	5.4.1 Main Activity	15
	5.4.2 Category Activity	16
	5.4.3 Detail Activity	16
	5.4.4 View Activity	17
	5.4.5 Splash Activity	17
6	<b>Result And Discussion</b>	20
7	<b>Conclusion And Future Work</b>	22
	7.1 Conclusion	22
	7.2 Future Work	22
	<b>Reference</b>	23
	<b>Appendix</b>	26
	a. Source Code	26
	b. Snapshots	37
	c.Puplication With Plagarism Report	42

## LIST OF FIGURES

FIGURE NO	TITLE	PAGE NO
1	Gradle Scripts	13
2	Activity Modules	15
3	Main Activity	16
4	Flow Of Data Diagram	19
5	Flowchart Of Activity Modules	19
6	View Of a Category	21
7	Layout Of Category Module	21



## **LIST OF ABBREVIATIONS**

And. - Android

AS – Android Studio

Frag – Fragment

Cmd – Command

Res – Resource

XML – Extensible Markup Language

API – Application Programming Interface

GP – Google Places

WS – Web Server

App – Application

SDK – Software development kit

EM – Emulator

APK – Android Application Package

Dex – Dalvik executable

# CHAPTER – 1

## INTRODUCTION

### 1.1 Introduction

Android is a mobile operating system based on a modified version of the Linux kernel and other open source software, designed primarily for touchscreen mobile devices such as smartphones and tablets. Android is developed by a consortium of developers known as the Open Handset Alliance and commercially sponsored by Google. It was unveiled in November 2007, with the first commercial Android device, the HTC Dream, being launched in September 2008. Most versions of Android are proprietary. The core components are taken from the Android Open Source Project (AOSP), which is free and open-source software primarily licensed under the Apache License.

Java is a high-level, class-based, object-oriented programming language that is designed to have as few implementation dependencies as possible. It is a general-purpose programming language intended to let programmers write once, run anywhere (WORA), meaning that compiled Java code can run on all platforms that support Java without the need to recompile. Java applications are typically compiled to bytecode that can run on any Java virtual machine (JVM) regardless of the underlying computer architecture. The syntax of Java is similar to C and C++, but has fewer low-level facilities than either of them. The Java runtime provides dynamic capabilities (such as reflection and runtime code modification) that are typically not available in traditional compiled languages. As of 2019, Java was one of the most popular programming languages in use according to GitHub, particularly for client–server web applications, with a reported 9 million developers.

Android Studio is the official integrated development environment (IDE) for Google's Android operating system, built on JetBrains' IntelliJ IDEA software and

designed specifically for Android development. It is available for download on Windows, macOS and Linux based operating systems or as a subscription-based service in 2020. It is a replacement for the Eclipse Android Development Tools (E-ADT) as the primary IDE for native Android application development. Android Studio was announced on May 16, 2013, at the Google I/O conference. It was in early access preview stage starting from version 0.1 in May 2013, then entered beta stage starting from version 0.8 which was released in June 2014. The first stable build was released in December 2014, starting from version 1.0. On May 7, 2019, Kotlin replaced Java as Google's preferred language for Android app development. Java is still supported, as is C++. The Android SDK is a software development kit that includes a comprehensive set of development tools. These include a debugger, libraries, a handset emulator based on QEMU, documentation, sample code, and tutorials. Currently supported development platforms include computers running Linux (any modern desktop Linux distribution), Mac OS X 10.5.8 or later, and Windows 7 or later. As of March 2015, the SDK is not available on Android itself, but software development is possible by using specialized Android applications.

Enhancements to Android's SDK go hand-in-hand with the overall Android platform development. The SDK also supports older versions of the Android platform in case developers wish to target their applications at older devices. Development tools are downloadable components, so after one has downloaded the latest version and platform, older platforms and tools can also be downloaded for compatibility testing. Android applications are packaged in .apk format and stored under /data/app folder on the Android OS (the folder is accessible only to the root user for security reasons). APK package contains .dex files (compiled byte code files called Dalvik executables), resource files, etc.

## CHAPTER -2

### Literature Review

#### 2.1 Literary Review

**Golam Kibria Anik 2021** a review of a web-based system intended for the eco-tourism market in Bangladesh. This paper analyses and discusses the development cycle of the software and the tools used in the development cycle. The paper looks at the design impact of the application and attempts to identify the factors affected by the application. Using the developed web application, users can view and sign up for advertised tours and pay for the tour using a digital payment system. Each advertised tour contains information regarding the tour and users can view reviews written by previous travelers. Using the developed web application, users can view the location of a tourist spots and can get an idea about the distance between the user's current location and the tourist spots. There will be tour guides to guide the travelers to visit all the tourist spots under a tour package. This application will help the growth of the tourism industry by helping users with better accessibility to affordable options.

**Aphile Kondlo 2020** they investigate the Cape Flats Nature Reserve, situated in the Western Cape, South Africa provides refuge to over 200 plant species, many endemic to the Western Cape. As part of the reserve's recreational activities, scheduled guided tours are offered to the public. The tours focus on the ecological importance and educational aspect of the reserve. A complete tour usually takes more than an hour. Due to the lack of trained tour guides and its strenuous nature, tours are only offered once daily. This has been identified as a challenge by the management of the nature reserve. In this paper a solution to the challenge using an augmented reality mobile application is proposed. The application allows visitors to experience the nature reserve in their own time without a guide. Augmented reality markers are placed at points of interest around the reserve. These in conjunction with the mobile application provide information about plants thereby mimicking actual tour guides. Outlines for the design and development of this self-guided tour application and the results of user acceptance and unit tests are provided in this paper.

**Vienna Casteranita 2020**'s survey focus on a form of safety precautions, the government throughout nations has to implement health protocols, such as physical distancing and nationwide lockdown. All activities, including work and school, have to be performed from home to minimize the virus's spreading, which triggers the influx of internet use. So do the Satellite Technology Center of the National Institute of Aeronautics and Space of Indonesia (LAPAN) uses the internet and social media to keep their open house activities well functional and has since turned it into a virtual tour. This research addresses the design of this virtual tour through personas. The personas are constructed following the literature study approach's design thinking method and followed by an online questionnaire shared with particular respondents.

**Li Zhang 2021** proposed a new media in the intelligent scenic spot guidance system, computer technology is integrated into each part of the guidance system and put into the scenic spot. Thus, this paper performs the optimization design of virtual and real integration of the visual communication based scenic spot guidance system and intelligent APP. The smart tourism system is designed from the overall architecture of the system to the functional modules of the system, which adopts the C/S software architecture system, and it is combined with mobile Internet development, WebService, MVC and other technologies. They also design and implement the relevant functional modules of the scenic spot guidance of the smart tourism system, and provide the design of the database table. Through the combination of online and offline, the tourist guide system is designed to establish multi-dimensional interaction with tourists and optimize the experience mode of tourism information.

**Peilin Chen 2021** introduced the smart tourism guidance applications based on the 5G Internet of Things system environment. Through 5G Internet of Things and other information technologies, change and improve the methods of information collection, capture, transmission and processing in the traditional tourism system to realize information sharing. The system uses the collection technology, information transmission technology and computer software technology of various fire protection sensors to transmit the alarm signals collected by the sensors to the fire 5G IoT data.

## **CHAPTER 3**

### **AIM AND SCOPE OF THE PROJECT**

#### **3.1 Aim Of The Project**

- Android Studio and Java language are used for creating an application to provide information about a particular city.
- Using the GooglePlaces API for extracting the necessary data like images, reviews and ratings.
- Providing useful all-in-one place information about the city.

#### **3.2 Scope Of The Project**

- The API-enabled technologies are playing a bigger role in Data Delivery with accuracy. This Android application used GooglePlaces API to provide data about various places in a city including Images, Reviews and Ratings.
- This application gives an all-in-one place information within a few clicks for different categories. Unlike the existing system where the user needs to search in a inefficient way, this app provides a user friendly interface for interaction.
- The application includes categories like Police Station, Hospital, Medical Store, Bus Station, Fire Station, Cafe, Petrol Pump, Gym, Post Office, Shopping Mall, Movie Theater, Jewelry Shop, Super Market, Bakery, Book Store etc.

### **3.2.1 Application Programming Interface Used**

The Places API is a service that returns information about places using HTTP requests. Places are defined within this API as establishments, geographic locations, or prominent points of interest. Each of the services is accessed as an HTTP request, and returns either an JSON or XML response. All requests to a Places service must use the https:// protocol, and include an API key. The Places API lets you search for place information using a variety of categories, including establishments, prominent points of interest, and geographic locations. You can search for places either by proximity or a text string. A Place Search returns a list of places along with summary information about each place

The Place Photo service, part of the Places API, is a read- only API that allows you to add high quality photographic content to your application. The Place Photo service gives you access to the millions of photos stored in the Places database. When you get place information using a Place Details request, photo references will be returned for relevant photographic content. Find Place, Nearby Search, and Text Search requests also return a single photo reference per place, when relevant. Using the Photo service you can then access the referenced photos and resize the image to the optimal size for your application.

### **3.2.2 Advantages**

1. It has a User Friendly Interface.
2. Provides a detailed result for places in a particular city.
3. It provides an all-in-one place information center.
4. Provides images of the place of visit in the city.
5. It displays reviews by other for the user to get benefitted.
6. Supports in almost all android smartphones.
7. It provides module to module interface.

### **3.2.3 Disadvantages**

1. It requires Internet access for working.
2. It requires a device with Android Operating-System installed.
3. It is a prototype.
4. It requires a fairly new version of android for working.
5. There is a possibility of inaccurate location detection.



## **CHAPTER 4**

### **WORKING THEORY OF OUR PROJECT**

#### **4.1 Android Studio**

Android Studio is the official integrated development environment (IDE) for Google's Android operating system, built on JetBrains' IntelliJ IDEA software and designed specifically for Android development. It is available for download on Windows, macOS and Linux based operating systems or as a subscription-based service in 2020. It is a replacement for the Eclipse Android Development Tools (E-ADT) as the primary IDE for native Android application development. Android Studio was announced on May 16, 2013, at the Google I/O conference. It was in early access preview stage starting from version 0.1 in May 2013, then entered beta stage starting from version 0.8 which was released in June 2014. The first stable build was released in December 2014, starting from version 1.0. On May 7, 2019, Kotlin replaced Java as Google's preferred language for Android app development. Java is still supported, as is C++. The Android SDK is a software development kit that includes a comprehensive set of development tools.

#### **4.2 Android OS**

Android is a mobile operating system based on a modified version of the Linux kernel and other open source software, designed primarily for touchscreen mobile devices such as smartphones and tablets. Android is developed by a consortium of developers known as the Open Handset Alliance and commercially sponsored by Google. It was unveiled in November 2007, with the first commercial Android device, the HTC Dream, being launched in September 2008. Most versions of Android are proprietary. The core components are taken from the Android Open Source Project (AOSP), which is free and open-source software primarily licensed under the Apache License. When Android is actually installed on devices, ability to modify the otherwise FOSS software is usually restricted, either by not providing the corresponding source

code or preventing reinstallation through technical measures, rendering the installed version proprietary. Most Android devices ship with additional proprietary software pre-installed,[14] most notably Google Mobile Services (GMS)[15] which includes core apps such as Google Chrome, the digital distribution platform Google Play, and associated Google Play Services development platform.

### **4.3 Java**

Java is a high-level, class-based, object-oriented programming language that is designed to have as few implementation dependencies as possible. It is a general-purpose programming language intended to let programmers write once, run anywhere (WORA), meaning that compiled Java code can run on all platforms that support Java without the need to recompile. Java applications are typically compiled to bytecode that can run on any Java virtual machine (JVM) regardless of the underlying computer architecture. The syntax of Java is similar to C and C++, but has fewer low-level facilities than either of them. The Java runtime provides dynamic capabilities (such as reflection and runtime code modification) that are typically not available in traditional compiled languages. As of 2019, Java was one of the most popular programming languages in use according to GitHub, particularly for client–server web applications, with a reported 9 million developers.

Java was originally developed by James Gosling at Sun Microsystems and released in May 1995 as a core component of Sun Microsystems' Java platform. The original and reference implementation Java compilers, virtual machines, and class libraries were originally released by Sun under proprietary licenses. As of May 2007, in compliance with the specifications of the Java Community Process, Sun had relicensed most of its Java technologies under the GPL-2.0-only license. Oracle offers its own HotSpot Java Virtual Machine, however the official reference implementation is the OpenJDK JVM which is free open-source software and used by most developers and is the default JVM for almost all Linux distributions.

As of October 2021, Java 17 is the latest version. Java 8, 11 and 17 are the current long-term support (LTS) versions. Oracle released the last zero-cost public update for

the legacy version Java 8 LTS in January 2019 for commercial use, although it will otherwise still support Java 8 with public updates for personal use indefinitely. Other vendors have begun to offer zero-cost builds of OpenJDK 8 and 11 that are still receiving security and other upgrades.

Oracle (and others) highly recommend uninstalling outdated and unsupported versions of Java, due to unresolved security issues in older versions. Oracle advises its users to immediately transition to a supported version, such as one of the LTS versions (8, 11, 17).

#### **4.4 XML**

Android provides a straightforward XML vocabulary that corresponds to the View classes and subclasses, such as those for widgets and layouts. You can also use Android Studio's Layout Editor to build your XML layout using a drag-and-drop interface. Declaring your UI in XML allows you to separate the presentation of your app from the code that controls its behavior. Using XML files also makes it easy to provide different layouts for different screen sizes and orientations. The Android framework gives you the flexibility to use either or both of these methods to build your app's UI. For example, you can declare your app's default layouts in XML, and then modify the layout at runtime. Using Android's XML vocabulary, you can quickly design UI layouts and the screen elements they contain, in the same way you create web pages in HTML — with a series of nested elements. Each layout file must contain exactly one root element, which must be a View or ViewGroup object. Once you've defined the root element, you can add additional layout objects or widgets as child elements to gradually build a View hierarchy that defines your layout.

When you compile your app, each XML layout file is compiled into a View resource. You should load the layout resource from your app code, in your Activity.onCreate() callback implementation. Do so by calling setContentView(), passing it the reference to your layout resource in the form of: R.layout.layout\_file\_name. The onCreate() callback method in your Activity is called by the Android framework when your Activity is launched. Attributes - Every View and ViewGroup object supports their own variety of XML attributes. Some attributes are specific to a View object (for example, TextView

supports the `textSize` attribute), but these attributes are also inherited by any `View` objects that may extend this class. Some are common to all `View` objects, because they are inherited from the root `View` class (like the `id` attribute). And, other attributes are considered "layout parameters," which are attributes that describe certain layout orientations of the `View` object, as defined by that object's parent `ViewGroup` object.

#### 4.5 Intergrated Development Environment

The integrated development environment is a software application that provides comprehensive facilities to computer programmers for software development. An IDE normally consists of at least a source code editor, build automation tools and a debugger. Developers use numerous tools throughout software code creation, building and testing. Development tools often include text editors, code libraries, compilers and test platforms. Without an IDE, a developer must select, deploy, integrate and manage all of these tools separately, Android Studio is the official integrated development environment (IDE) for Google's Android operating system, built on JetBrains' IntelliJ IDEA software and designed specifically for Android development. It is available for download on Windows, macOS and Linux based operating systems. Android Studio is the authority incorporated improvement condition (IDE) for Google's Android working framework, based on JetBrains' IntelliJ IDEA programming and structured explicitly for Android advancement. It is accessible for download on Windows, macOS and Linux based working frameworks.

#### 4.6 Android Components









The **Android Manifest** is an XML file which contains important metadata about the Android app. This includes the package name, activity names, main activity (the entry point to the app), Android version support, hardware features support, permissions, and other configurations. `LauncherManifest.xml` - Located in the exported project, at `root/launcher/src/main/AndroidManifest.xml`. This file contains the app's: icons, name, starting activity and its Intents, install location, supported screen sizes, `isGame` setting. `LibraryManifest.xml` - Located in the exported project at `root/unityLibrary/src/main/AndroidManifest.xml`. You can override this manifest with a custom `_AndroidManifest.xml_` that you add in the `Plugins/Android` folder. This file

declares the: Unity activity, permissions, theme used by the Unity activity, VR modes, VR performance, making the activity non-resizable (for VR), setting max aspect ratio, reacting to configuration changes, orientations, launch modes, Android UI, hardware, acceleration, used features (like gamepad or graphics API), notch support, initial window size, ability to resize the window. The Android Runtime Permission System asks the user to grant permissions while the app is running, instead of when they first install the app. App users can usually grant or deny each permission when the app needs it while the app is running (for example, requesting camera permission before taking a picture). This allows an app to run with limited functionality without permissions. You can use the `Android.Permission` class in Unity to check whether the user granted or denied specific permissions. If a permission your app needs has been denied, you can inform the user why the app needs it and ask them to approve the permission. For more information, see documentation on Requesting Permissions. Your app normally prompts the user to allow what Android calls “dangerous” permissions on its startup.

**Gradle** is an Android build system that automates a number of build processes and prevents many common build errors. In Unity, Gradle reduces the method reference count in DEX (Dalvik Executable format) files, which means you are less likely to come across DEX limit problems. Unity uses Gradle for all Android builds. You can either build the output package (.apk, .aab) in Unity, or export a Gradle project from Unity, and then build it with an external tool such as Android Studio. Gradle templates describe and configure how to build your Android app with Gradle. Each Gradle template represents a single Gradle project. Gradle projects can include, and depend on other Gradle projects.

o Max Pooling – taking the largest element in the feature map.

You can use a custom build.gradle file for the unityLibrary module when you build the APK from Unity. This file contains specific build instructions specified in template variables. Unity then generates a default mainTemplate.gradle file in your Project’s Assets/Plugins/Android/ folder. The path to the new file also appears under the Custom Main Gradle Template option in the Player Settings. By default, Unity uses the settingsTemplate.gradle file from the Unity install directory to create the settings.gradle file for your build.

- ▼  Gradle Scripts
  -  build.gradle (Project: City\_Guide)
  -  build.gradle (Module: City\_Guide.app)
  -  gradle-wrapper.properties (Gradle Version)
  -  proguard-rules.pro (ProGuard Rules for City\_Guide.app)
  -  gradle.properties (Project Properties)
  -  settings.gradle (Project Settings)
  -  local.properties (SDK Location)

***Figure 4.1: Gradle Scripts***

## CHAPTER 5

### IMPLEMENTATION & METHODOLOGY

#### 5.1 Software Requirements:

Java - We have used Java which is a Object Oriented programming language like C/C++, instead of Kotlin due to the following reasons:

1. Java code is more compact, straight forward readable than Kotlin .
2. Java is more familiar than Kotlin .
3. It has static methods/variables and classes are not final by default.

Java is an object-oriented programming language developed by Sun Microsystems, which is now owned by Oracle. Being such an old language, Java does a whole lot more than just develop Android apps. So if you know your Java, you have a lot more job opportunities. You may not want to develop only Android apps all the time. Java lets you spread your wings wider.

- Easy to learn and understand.
- Works well for native as well as cross-platform apps.
- Since Android itself is built on Java, there are plenty of Java libraries to your aid.
- Java has a wide open-source ecosystem.
- Java apps are lighter and more compact, even when compared to Kotlin apps, resulting in a faster app experience.
- Thanks to the accelerated assembly with Gradle, assembling large projects becomes easier in Java.

#### 5.2 Hardware Requirements:

Processor: Intel® Core™ i3

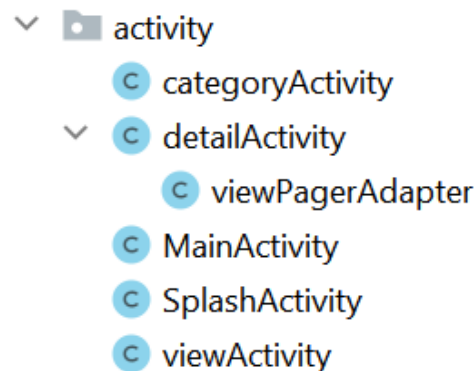
CPU @ 2.40GHz

Installed memory (RAM):4.00GB

System Type: 64-bit Operating System

### 5.3 Activity Class:

The Activity class is a crucial component of an Android app, and the way activities are launched and put together is a fundamental part of the platform's application model. Unlike programming paradigms in which apps are launched with a main() method, the Android system initiates code in an Activity instance by invoking specific callback methods that correspond to specific stages of its lifecycle. The Activity class is designed to facilitate this paradigm. When one app invokes another, the calling app invokes an activity in the other app, rather than the app as an atomic whole. In this way, the activity serves as the entry point for an app's interaction with the user. You implement an activity as a subclass of the Activity class.



**Figure 5.1: Activity Modules**

### 5.4 ACTIVITY

#### 5.4.1 Main Activity

The Activity Modules includes a 'Main Activity' that Extends 'AppCompatActivity'. It contains a method called 'onBackPressed' which navigates it to display the 'drawer' view. It has a 'getRef()' method that gets and sets the user information in the profile. The 'onNavigationItemSelectedListener' method sets the path to the selected layout. It has a 'setScreenTitle' that takes the item\_id and returns the title selected. 'displaySelectedFragment' method takes item\_id and returns the navigation view. It also contains the methods for location, longitude and latitude.



```

@Override
protected void onCreate(Bundle savedInstanceState) {...}

@Override
public void onBackPressed() {...}

void getRef() {...}

@Override
public boolean onNavigationItemSelected(@NonNull MenuItem item) {...}

public void setScreenTitle(int item_id) {...}

public void displaySelectedFragment(int item_id) {...}

```

**Figure 5.2: Main Activity**

### 5.4.2 Category Activity

The Category Activity extends the 'appCompatActivity'. This class contains the api id as a string. Which is implemented using the 'fetchData()' method where a JSON Object Request instance is created and fetches the name, place id, photos, rating etc for the gives city. This is done by the 'onResponse' method which takes a JSON object as input, loops through the JSON array and returns the result to the adapter. The URL used here is "https://maps.googleapis.com/maps/api/place/textsearch/json?". This activity has a 'onErrorResponse' method which displays a error message as a toast.

### 5.4.3 Detail Activity

The Detail Activity extends AppCompatActivity. This class has ImageView image; private ActivityDetailBinding binding; private detailActivity activity; private viewPagerAdapter adapter; properties. The 'initView()' method initiates the adapter with

'viewPagerAdapter(activity.getSupportFragmentManager(),activity.getLifecycle());'.

It has a viewPagerAdapter class that extends FragmentStateAdapter which contains two generic collection lists 'fragmentList' and 'fragmentTitleList' which is added with fragment and title respectively using the 'add()' property. It includes a Fragment called createFragment that takes a integer 'position' and returns the coressponding position in the fragment list using the 'getPosition()' property like 'return fragmentList.get(position);'. It has a method called 'getItemCount()' which returns the count of items present in the fragment list using the 'size()' property like 'return

`fragmentList.get(position);`’.

#### **5.4.4 View Activity**

The `viewActivity` class extends `AppCompatActivity` which contains an `ImageView` instance called `viewphoto`. This is invoked on `onCreate` which takes `Bundle savedInstanceState` as its argument and sets the `ContentView` as `setContentView(R.layout.activity_viewphoto);` The `viewphoto` variable of `ImageView` is assigned with a view using `findViewById` method. A `PostAdapter` is created using the new keyword and a string url is returned by the adapter using `getUrl()` method like `a.getUrl();`

#### **5.4.5 Splash Activity**

The `SplashActivity` class extends `AppCompatActivity`. This activity module imports `android.content.Intent`, An intent is an abstract description of an operation to be performed. It can be used with `startActivity` to launch an Activity, `broadcastIntent` to send it to any interested `BroadcastReceiver` components, and `Context.startService(Intent)` or `Context.bindService(Intent, ServiceConnection, int)` to communicate with a background Service. An Intent provides a facility for performing late runtime binding between the code in different applications. Its most significant use is in the launching of activities, where it can be thought of as the glue between activities. It is basically a passive data structure holding an abstract description of an action to be performed. It imports `android.os.Bundle`, which involves in A mapping from String keys to various `Parcelable` values and `Parcelables` are Interface for classes whose instances can be written to and restored from a `Parcel`.

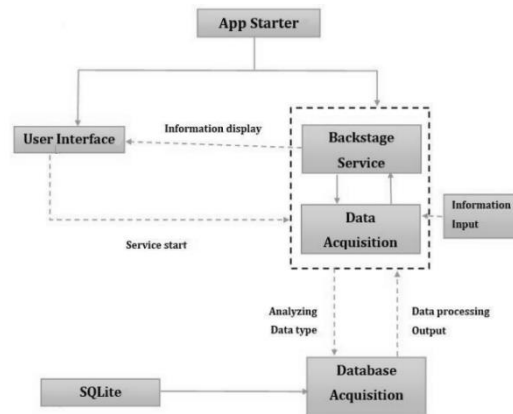
Classes implementing the `Parcelable` interface must also have a non-null static field called `CREATOR` of a type that implements the `Parcelable.Creator` interface, `Parcelable.ClassLoaderCreator<T>` Specialization of `Creator` that allows you to receive the `ClassLoader` the object is being created in and `Parcelable.Creator<T>` Interface that must be implemented and provided as a public `CREATOR` field that generates instances of your `Parcelable` class from a `Parcel`. This activity class imports `android.os.Handler`, a `Handler` allows you to send and process `Message` and `Runnable` objects associated with a thread's `MessageQueue`. Each `Handler` instance

is associated with a single thread and that thread's message queue. When you create a new Handler it is bound to a Looper. It will deliver messages and runnables to that Looper's message queue and execute them on that Looper's thread. There are two main uses for a Handler: (1) to schedule messages and runnables to be executed at some point in the future; and (2) to enqueue an action to be performed on a different thread than your own. Scheduling messages is accomplished with the `post(Runnable)`, `postAtTime(java.lang.Runnable, long)`, `postDelayed(Runnable, Object, long)`, `sendEmptyMessage(int)`, `sendMessage(Message)`, `sendMessageAtTime(Message, long)`, and `sendMessageDelayed(Message, long)` methods.

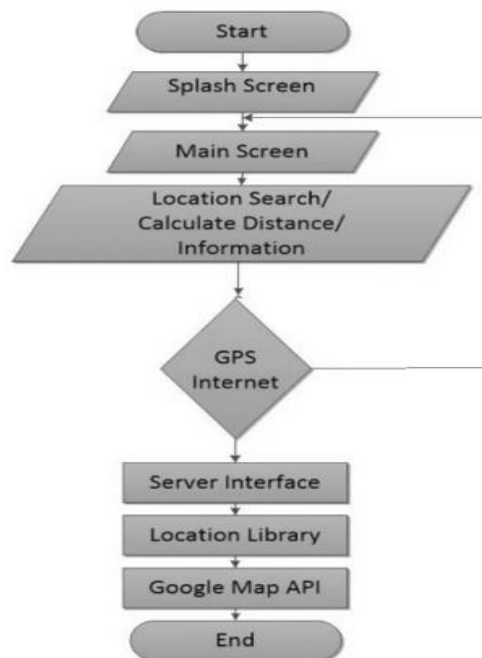
The `post` versions allow you to enqueue Runnable objects to be called by the message queue when they are received; the `sendMessage` versions allow you to enqueue a Message object containing a bundle of data that will be processed by the Handler's `handleMessage(Message)` method (requiring that you implement a subclass of Handler). When posting or sending to a Handler, you can either allow the item to be processed as soon as the message queue is ready to do so, or specify a delay before it gets processed or absolute time for it to be processed. The latter two allow you to implement timeouts, ticks, and other timing-based behavior. When a process is created for your application, its main thread is dedicated to running a message queue that takes care of managing the top-level application objects (activities, broadcast receivers, etc) and any windows they create. You can create your own threads, and communicate back with the main application thread through a Handler. This is done by calling the same `post` or `sendMessage` methods as before, but from your new thread.

The given Runnable or Message will then be scheduled in the Handler's message queue and processed when appropriate. The activity imports 'android.view.WindowManager', The interface that apps use to talk to the window manager. Each window manager instance is bound to a Display. The simplest way to show a window on a particular display is to create a Presentation, which automatically obtains a WindowManager and context for the display. In this class a `SPLASH_SCREEN_TIME_OUT` variable of type integer and private accessor is set to 2000 and methods like `getWindow()`, `setFlags()`, `postDelayed()`, `getCurrentUser()` are

used and a 'run()' method with arguments including the time variable is implemented.



**Figure 5.3: Flow of Data Diagram**

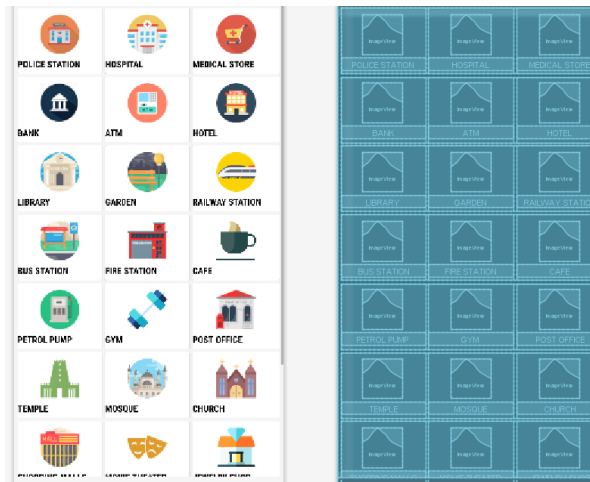


**Figure 5.4: Flowchart of Activity Modules**

## CHAPTER – 6

### RESULT DISCUSSION

The Proposed System provides an online information about the particular city going to visit. User Friendliness is provided in the application with various controls along with Rich User Interface. This system provides the benefits of all-at-one place application. Every necessary information is provided in the same application. The existing system is a manual system. Here the city information needs to save his information in the form of excel sheets or Disk Drives. The manual system gives us very less security for saving data; some data may be lost due to mismanagement.



**Figure 6.1: Layout of Category Module**

It shows the ease of access and user friendly interface of the category selection part of the application layout. Here each place is titled and corresponding icons are provided.



**Figure 6.2: View of a Category**

It shows the layout of available places of the selected category from the list. It shows the place name, distance from the user, rating of the place and address.

## **CHAPTER – 7**

### **CONCLUSION AND FUTURE WORK**

#### **7.1 CONCLUSION**

The main motive of the project is to explore how to realize a mobile city guide using the Android platform, including a prototype of the city guide. The prototype implemented include basic functionalities of city guides such as showing the map, locating places of interest and informations about the site. The app comforts a new native in a city by showing information of all the nearby sites of various categories. Caregories include Hospital Services, Police Station, Parks, Famous Restaurants, Gym, Bakery, Cafe, Shopping Malls, Hotels, Bus Stations, ATMs, Banks, Train Stations etc., with their reviews. This project uses Java for programming and uses Android studio IDE as the tool to implement the android application. This includes creating various XML layouts like Splash, Category etc., The information about a city is fetched from googleplaces API, which provides ratings, reviews, pictures and distance of the place.

#### **7.2 FUTURE WORK**

Building the City Guide application is other platforms like IOS, windows OS etc., as it would help the application to reach masses. The availability of the application in various Operating-Systems provides the user with the benefit of accessing the informations from the application more widely. Also implementing the app as a web-based service would make the process far more easy. As web application is available for every device irrespective of the platform, hardware and software. This helps the users in a more efficient way. Building a web site based on this android application is more easier because of the current web-technologies.

## REFERENCES

- [1] Benni Agung Nugroho; Abidatul Izzah; Ratna Widyastuti, Development of Android Application for City Tour Recommendation System Based on Dynamic Programming 2019.
- [2] D. Gavalas, C. Konstantopoulos, K. Mastakas, G. Pantziou, Mobile Recommender Systems in Tourism, Network and Computer Applications, vol. 39
- [3] N. Ingraham, "European Union proposal would significantly reduce mobile roaming costs for EU citizens"
- [4] Van Zoonen, L. Privacy concerns in smart cities. Gov. Inf. Q. 2016, 33, 472–480.
- [5] Lai, C.S.; Lai, L.L.; Lai, Q.H. Smart Grids and Big Data Analytics for Smart Cities, 1st ed.; Springer International Publishing: New York, NY, USA, 2020.
- [6] Camero, A.; Alba, E. Smart City and information technology: A review. Cities 2019, 93, 84–94.
- [7] Caird, S.P.; Hallett, S.H. Towards evaluation design for smart city development. J. Urban Des. 2019, 24, 88–209.
- [8] MongoDB, "Advantages of NoSQL," <https://www.mongodb.com/scale/>
- [9] R. Kramer, M. Modsching, and K. Hagen, "Development and evaluation of a context-driven, mobile tourist guide", Pervasive Computing and Communications, Vol. 3, Issue 4, 2005, pp. 378- 399.
- [10] K. Al-Rayes, A. Sevkli, H. Al-Moaiqel, H. Al-Ajlan, K. Al-Salem, N. Al-Fantoukh, "A Mobile Tourist Guide for Trip Planning", IEEE Multidisciplinary Engineering Education Magazine, vol. 6, no. 4, Dec 2011, pp. 1-6.
- [11] K. Luyten and K. Coninx, "Imogl: Take Control over a Context Aware Electronic Mobile Guide for Museums", HCI in Mobile Guides, University of Strathclyde,



Glasgow, September 2004.

- [12] Jing Zhou 2020, Design of Intelligent Scenic Area Guide System Based on Visual Communication.
- [13] BLi Zhang 2021, Intelligent Guide System of Scenic Spot Based on Visual Communication.
- [14] Benni Agung Nugroho; Abidatul Izzah; Ratna Widyastuti 2019, Development of Android Application for City Tour Recommendation System Based on Dynamic Programming.
- [15] Peilin Chen 2021, Smart Tour Guide Application Based on 5G IoT System Environment.
- [16] Xiao Zhou; Bin Sun; Sen Li; Shiyan Liu, Tour Route Planning Algorithm Based on Precise Interested Tourist Sight Data Mining.
- [17] Vladimir Mladenovic;Maja M. Lutovac;Miroslav D. Lutovac, Electronic tour guide for Android mobile platform with multimedia travel book.
- [18] Shan Li;Xueli Duan;Yanxia Bai;Caixia Yun, Development and Application of Intelligent Tour Guide System in Mobile Terminal
- [19] Li Liu; Yanfang Jing, Android city tour guide system based on Web service.
- [20] A. Smirnov, N. Shilov, A. Kashevnik, N. Teslya, M. Shchekotov, "Intelligent Tourist Guiding Service Based on Smart-M3 Plat- form", Proceedings of 13th Conference of Open Innovations As- sociation FRUCT.
- [21] Heba Kurdi;Nora Alnashwan, Design and implementation of mobile cloud tourism application 2017.
- [22] Jinn-Shing Cheng;Hung-Wei Hsiang;Wer-Chih Wu The design of intelligent mobile tourism service system 2010 International Computer Symposium (ICS2010) Year: 2010 | Conference Paper | Publisher: IEEE.
- [23] Junkai Zhong;Yanpeng Li, Design and implementation of intelligent guide system based on LBS 2019 IEEE International Conference on Consumer

Electronics - Taiwan (ICCE-TW) Year: 2019 | Conference Paper | Publisher: IEEE.

- [24] Riri Safitri;Deska Setiawan Yusra;Denny Hermawan;Endang Ripmiatin;Winangsari Pradani, Mobile tourism application using augmented reality 2017 5th International Conference on Cyber and IT Service Management (CITSM) Year: 2017 | Conference Paper | Publisher: IEEE.

## APPENDICES

### A.SOURCE CODE

```
//Category Activity

ackage com.example.cityguide.activity;

import android.content.Intent;
import android.net.Uri;
import android.os.Bundle;
import android.util.Log;
import android.widget.Toast;

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

import com.android.volley.Request;
import com.android.volley.Response;
import com.android.volley.VolleyError;
import com.android.volley.toolbox.JsonObjectRequest;
import com.example.cityguide.MySingleton;
import com.example.cityguide.R;
import com.example.cityguide.adapters.categoryAdapter;
import com.example.cityguide.modals.details_categories;

import org.json.JSONArray;
import org.json.JSONObject;

import java.util.ArrayList;

public class categoryActivity extends AppCompatActivity {

    RecyclerView recyclerView;
    String longitude, latitude;
    categoryAdapter adapter;
    ArrayList<details_categories> arrayList = new ArrayList<>();
    String query;

    String apiKey="AlzaSyAxrj_oW36wimhP-0Y2kvQSkK2CG6mTmKc";

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

```

setContentView(R.layout.category_recycle);
recyclerView = (RecyclerView) findViewById(R.id.recycle);
recyclerView.setLayoutManager(new LinearLayoutManager(this));
recyclerView.setHasFixedSize(true);
recyclerView.setItemAnimator(new DefaultItemAnimator());

Intent i=getIntent();
query=i.getStringExtra("query");
fetchData();
}

private void fetchData() {

MainActivity a = new MainActivity();
latitude = a.getLatitude();
longitude = a.getLongitude();
String url = "https://maps.googleapis.com/maps/api/place/textsearch/json?";
String lat=a.getLatitude()+"",lon=a.getLongitude();
Uri baseUrl= Uri.parse(url);
Uri.Builder uriBuilder = baseUrl.buildUpon();
uriBuilder.appendQueryParameter("ll",lat);
uriBuilder.appendQueryParameter("query",query);
uriBuilder.appendQueryParameter("key",apiKey);
Log.d("urlll",uriBuilder.toString());
JsonObjectRequest jsonObjectRequest = new
JsonObjectRequest(Request.Method.GET, uriBuilder.toString(), null, new
Response.Listener<JSONObject>() {
    @Override
    public void onResponse(JSONObject response) {
        try {
            JSONArray jsonArray=response.getJSONArray("results");
            for(int i=0;i<javascriptArray.length();i++){
                JSONObject jsonObject=jsonArray.getJSONObject(i);
                String address="null";
                if(jsonObject.has("formatted_address")) {
                    address = jsonObject.getString("formatted_address");
                }
                double lat=0,lon=0;
                if(jsonObject.has("geometry")) {
                    JSONObject jsonObject1 = jsonObject.getJSONObject("geometry");
                    JSONObject jsonObject2 = jsonObject1.getJSONObject("location");
                    lat = jsonObject2.getDouble("lat");
                    lon = jsonObject2.getDouble("lng");
                }
                String resname=null;

```

```

        if(jsonObject.has("name")) {
            resname = jsonObject.getString("name");
        }
        String placeid=null;
        if(jsonObject.has("place_id")) {
            placeid = jsonObject.getString("place_id");
        }
        double rating=0;
        if(jsonObject.has("rating")) {
            rating = jsonObject.getDouble("rating");
        }
        String photoref="null";
        if(jsonObject.has("photos")) {
            JSONArray jsonArray1 = jsonObject.getJSONArray("photos");

            JSONObject jsonObject3 = jsonArray1.getJSONObject(0);
            photoref = jsonObject3.getString("photo_reference");
        }

        Log.d("S",resname+"/"+address+"/"+lat+"/"+lon+"/"+photoref+"/"+placeid+"/"+rating);

        //    fetchPhoto(resname,address,lat,lon,photoref,placeid,rating);
        details_categories details=new
        details_categories(resname,address,lat,lon,photoref,placeid,rating);
        arrayList.add(details);
    }

    } catch (Exception e) {
        Toast.makeText(getApplicationContext(), e.getMessage()+"ss",
        Toast.LENGTH_SHORT).show();
    }
    adapter = new categoryAdapter(categoryActivity.this, arrayList);
    recyclerView.setAdapter(adapter);
}
}, new Response.ErrorListener() {
    @Override
    public void onErrorResponse(VolleyError error) {

        Toast.makeText(getApplicationContext(), "Failed:Check Internet",
        Toast.LENGTH_SHORT).show();
    }
});
MySingleton.getInstance(this).addToRequestQueue(jsonObjectRequest);
}
}

```

```

//Detail Activity
package com.example.cityguide.activity;

import android.os.Bundle;
import android.view.LayoutInflater;
import android.widget.ImageView;
import android.widget.TextView;

import androidx.annotation.NonNull;
import androidx.appcompat.app.AppCompatActivity;
import androidx.databinding.DataBindingUtil;
import androidx.fragment.app.Fragment;
import androidx.fragment.app.FragmentManager;
import androidx.lifecycle.Lifecycle;
import androidx.viewpager2.adapter.FragmentStateAdapter;

import com.bumptech.glide.Glide;
import com.example.cityguide.R;
import com.example.cityguide.adapters.categoryAdapter;
import com.example.cityguide.databinding.ActivityDetailBinding;
import com.example.cityguide.fragments.fragmentInfo;
import com.example.cityguide.fragments.fragmentPhoto;
import com.example.cityguide.fragments.fragmentReview;
import com.google.android.material.appbar.CollapsingToolbarLayout;
import com.google.android.material.tabs.TabLayoutMediator;

import java.util.ArrayList;
import java.util.List;

public class detailActivity extends AppCompatActivity {
    ImageView image;
    private ActivityDetailBinding binding;
    private detailActivity activity;
    private viewPagerAdapter adapter;
    CollapsingToolbarLayout text;

    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        binding=DataBindingUtil.setContentView(this,R.layout.activity_detail);
        activity=this;
        image=findViewById(R.id.image);
        text=findViewById(R.id.toolbar);

        categoryAdapter a=new categoryAdapter();

```

```

        Glide.with(getApplicationContext()).load(a.getphotoiurl()).into(image);

        text.setTitle(a.getpplacename());

    initView();
}

private void initView() {

    adapter=new
    viewPagerAdapter(activity.getSupportFragmentManager(),activity.getLifecycle());
    adapter.addFragement(new fragmentInfo(),"INFO" );
    adapter.addFragement(new fragmentPhoto(),"PHOTO");
    adapter.addFragement(new fragmentReview(),"REVIEWS" );
    binding.viewPager.setAdapter(adapter);
    binding.viewPager.setOffscreenPageLimit(1);
    new TabLayoutMediator(binding.tabLayout,binding.viewPager,
        (tab, position) -> {
    tab.setText(adapter.fragmentTitleList.get(position));
        }).attach();

    for(int i=0;i<binding.tabLayout.getTabCount();i++){

        TextView tv = (TextView) LayoutInflater.from(activity)
            .inflate(R.layout.custom_tab, null);

        binding.tabLayout.getTabAt(i).setCustomView(tv);
    }
}

class viewPagerAdapter extends FragmentStateAdapter {
    private final List<Fragment> fragmentList=new ArrayList<>();
    private final List<String> fragmentTitleList=new ArrayList<>();

    public viewPagerAdapter(@NonNull FragmentManager fragmentManager,
        @NonNull Lifecycle lifecycle) {
        super(fragmentManager, lifecycle);
    }
    public void addFragement(Fragment fragment,String title){
        fragmentList.add(fragment);
        fragmentTitleList.add(title);
    }

    @NonNull

```

```

        @Override
        public Fragment createFragment(int position) {
            return fragmentList.get(position);
        }

        @Override
        public int getItemCount() {
            return fragmentList.size();
        }
    }
}

//Main Activity
package com.example.cityguide.activity;

import android.Manifest;
import android.app.AlertDialog;
import android.content.DialogInterface;
import android.content.Intent;
import android.content.pm.PackageManager;
import android.location.Location;
import android.os.Bundle;

public class MainActivity extends AppCompatActivity implements
NavigationView.OnNavigationItemSelectedListener {
    ImageView imageview;
    TextView name;
    DatabaseReference rootref;
    FirebaseAuth mauth;
    FirebaseUser firebaseUser;
    public static Location currentLocation;
    FusedLocationProviderClient fusedLocationProviderClient;
    private static final int REQUEST_CODE = 101;
    NavigationView navigationView;
    androidx.appcompat.widget.Toolbar toolbar;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        getWindow().setFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN,
WindowManager.LayoutParams.FLAG_FULLSCREEN);
        setContentView(R.layout.activity_main);
        mauth = FirebaseAuth.getInstance();
        toolbar = (androidx.appcompat.widget.Toolbar) findViewById(R.id.toolbar);
        toolbar.setTitle("MY PLACE");
        navigationView = (NavigationView) findViewById(R.id.nav_view);
    }
}

```



```

        View hView = navigationView.getHeaderView(0);
        imageview = (ImageView) hView.findViewById(R.id.imageView);
        name = (TextView) hView.findViewById(R.id.menu_header_name);
        fusedLocationProviderClient =
LocationServices.getFusedLocationProviderClient(this);
        DrawerLayout drawer = (DrawerLayout) findViewById(R.id.drawer);
        ActionBarDrawerToggle toggle = new ActionBarDrawerToggle(
            this, drawer, toolbar, R.string.navigation_drawer_open,
R.string.navigation_drawer_close);
        drawer.setDrawerListener(toggle);
        toggle.syncState();

        navigationView.setNavigationItemSelectedListener(this);

        //showing default fragment
        displaySelectedFragment(R.id.nav_home);
        getRef();
        currentlocation();
    }

    @Override
    public void onBackPressed() {
        DrawerLayout drawer = (DrawerLayout) findViewById(R.id.drawer);
        if (drawer.isDrawerOpen(GravityCompat.START)) {
            drawer.closeDrawer(GravityCompat.START);
        } else {
            super.onBackPressed();
        }
    }

    void getRef() {
        firebaseUser = FirebaseAuth.getInstance().getCurrentUser();
        rootref = FirebaseDatabase.getInstance().getReference("Users");
        rootref.addValueEventListener(new ValueEventListener() {
            @Override
            public void onDataChange(@NonNull DataSnapshot snapshot) {
                for (DataSnapshot dataSnapshot : snapshot.getChildren()) {
                    Users user = dataSnapshot.getValue(Users.class);

                    Glide.with(getApplicationContext()).load(user.getImageUrl().toString()).transform(new
                    circle(getApplicationContext())).into(imageview);
                }
            }
        });
    }
}

```

```

@Override
public boolean onNavigationItemSelected(@NonNull MenuItem item) {

    // item id is being passed into the method here
    displaySelectedFragment(item.getItemId());

    DrawerLayout drawer = (DrawerLayout) findViewById(R.id.drawer);
    drawer.closeDrawer(GravityCompat.START);
    return true;
}

public void setScreenTitle(int item_id) {
    String title = "";
    switch (item_id) {
        case R.id.nav_home:
            title = "Home";
            break;
        case R.id.nav_category:
            title = "Category";
            break;

        case R.id.nav_Location:
            title = "Location";
            break;

        case R.id.nav_save:
            title = "Saved Location";
            break;
        case R.id.nav_profile:
            title = "Profile";
            break;

        case R.id.nav_logout:
            title = "Three";
            break;

        case R.id.nav_map:
            title = "MAP";
            break;

        case R.id.nav_about:
            title = "About";
            break;
    }
}

```

```

        toolbar.setTitle(title);
    }

    public void displaySelectedFragment(int item_id) {
        Fragment fragment = null;
        switch (item_id) {
            case R.id.nav_home:
                fragment = new fragmentHome();
                navigationView.getMenu().getItem(0).setChecked(true);
                break;
            case R.id.nav_category:
                fragment = new fragmentCategory();
                navigationView.getMenu().getItem(1).setChecked(true);
                break;
            case R.id.nav_Location:
                fragment = new fragmentLocation();
                navigationView.getMenu().getItem(2).setChecked(true);
                break;
            case R.id.nav_save:
                fragment = new fragmentSaved();
                navigationView.getMenu().getItem(3).setChecked(true);
                break;
            case R.id.nav_profile:
                fragment = new fragmentProfile();
                navigationView.getMenu().getItem(4).setChecked(true);
                break;
            case R.id.nav_logout:
                logOut();
                navigationView.getMenu().getItem(5).setChecked(true);
                break;
            case R.id.nav_map:
                fragment = new fragmentMap();
                break;
            case R.id.nav_about:
                fragment = new fragmentAbout();
                break;
        }
        if (fragment != null) {

            FragmentTransaction ft = getSupportFragmentManager().beginTransaction();
            //this is where the id of the FrameLayout is being mentioned. Hence the
            fragment would be loaded into the framelayout
            ft.replace(R.id.container, fragment);
            ft.commit();
        }
    }

```

```

    /** setting title to the screen */
    setTitle(item_id);
}

private void logOut() {
    AlertDialog.Builder builder = new AlertDialog.Builder(MainActivity.this);
    builder.setMessage("Do you want to Log Out ?");
    builder.setTitle("LOG OUT");

    builder.setCancelable(false);
    builder.setPositiveButton("Yes", new DialogInterface.OnClickListener() {

        @Override
        public void onClick(DialogInterface dialog, int which) {
            outofScreen();
        }
    });
    builder.setNegativeButton("No", new DialogInterface.OnClickListener() {

        @Override
        public void onClick(DialogInterface dialog,
            int which) {
            dialog.cancel();
        }
    });

    AlertDialog alertDialog = builder.create();
    alertDialog.show();
}

private void outofScreen() {
    FirebaseAuth.getInstance().signOut();
    Intent intent = new Intent(MainActivity.this, SignInActivity.class);
    startActivity(intent);
}

void currentlocation(){
    if (ActivityCompat.checkSelfPermission(this,
        Manifest.permission.ACCESS_FINE_LOCATION) !=
        PackageManager.PERMISSION_GRANTED &&
        ActivityCompat.checkSelfPermission(
            this, Manifest.permission.ACCESS_COARSE_LOCATION) !=
        PackageManager.PERMISSION_GRANTED) {
        ActivityCompat.requestPermissions(this, new
        String[]{Manifest.permission.ACCESS_FINE_LOCATION}, REQUEST_CODE);
        return;
    }
}

```

```

    }
    Task<Location> task = fusedLocationProviderClient.getLastLocation();
    task.addOnSuccessListener(new OnSuccessListener<Location>() {
        @Override
        public void onSuccess(Location location) {
            if (location != null) {
                currentLocation = location;
                Log.d("lan", String.valueOf(currentLocation.getLatitude()));

                Log.d("lot", String.valueOf(currentLocation.getLongitude()));
            }
        }
    });
}

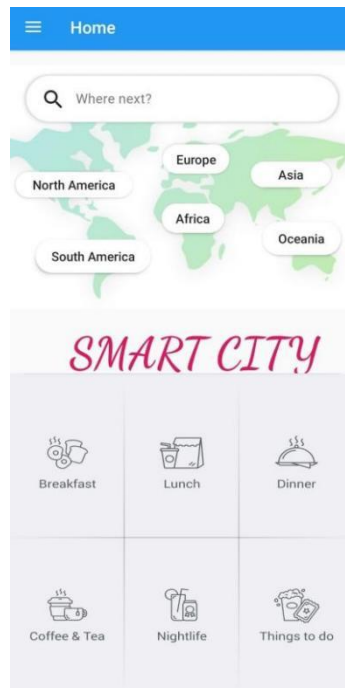
@Override
public void onRequestPermissionsResult ( int requestCode, @NonNull String[]
permissions,
                                     @NonNull int[] grantResults){
    super.onRequestPermissionsResult(requestCode, permissions, grantResults);
    switch (requestCode) {
        case REQUEST_CODE:
            if (grantResults.length > 0 && grantResults[0] ==
PackageManager.PERMISSION_GRANTED) {
                currentlocation();
            }
            break;
        }
    }
}

public String getLatitude(){
    return String.valueOf(currentLocation.getLatitude());
}
public String getLongitude(){
    return String.valueOf(currentLocation.getLongitude());
}
public Location getLoction(){
    return currentLocation;
}
}

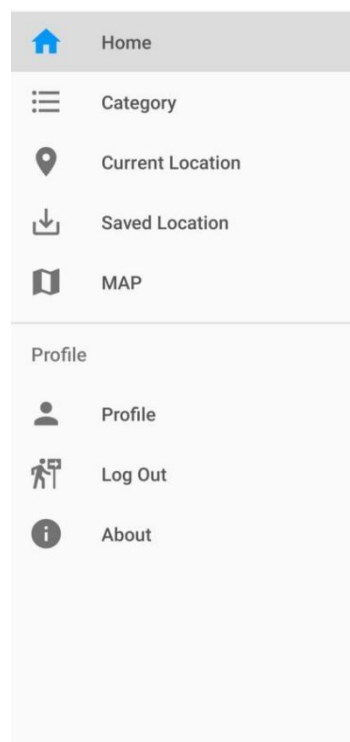
```

## B.OUTPUT SCREEN SHOTS

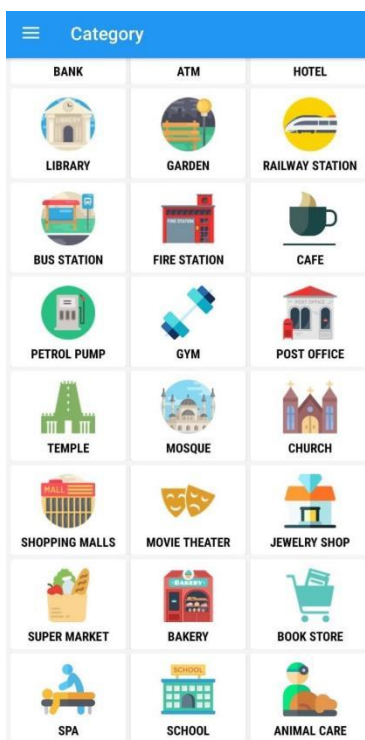
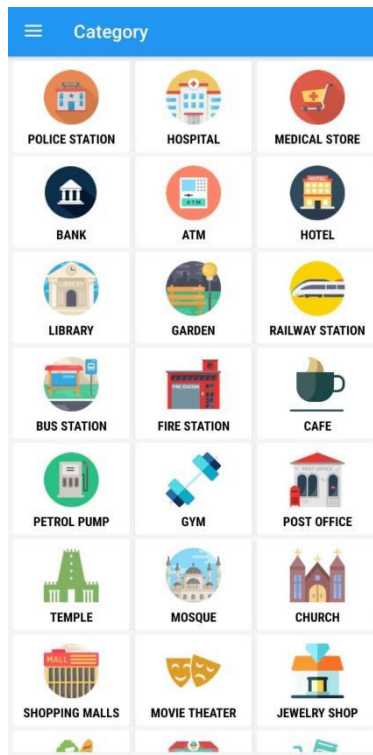
Home Screen :



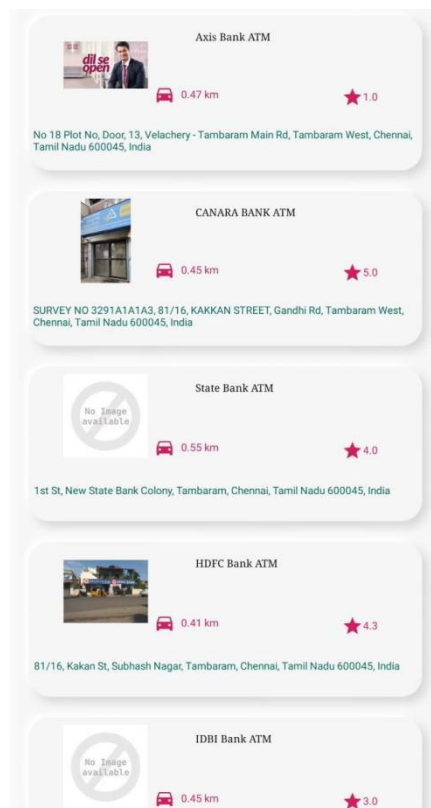
Menu :



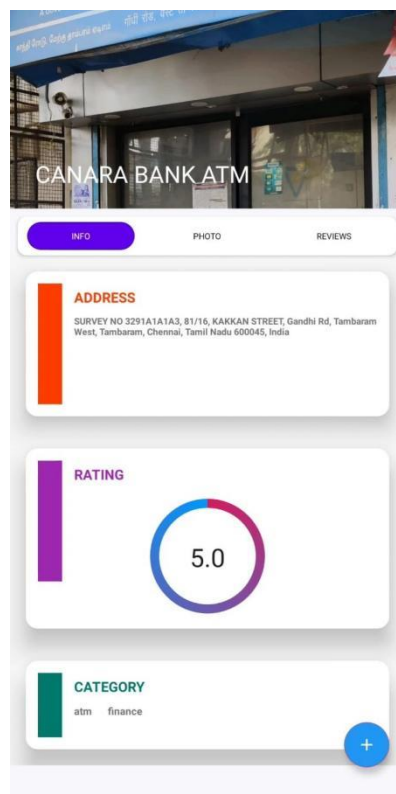
Category :



Selected :



Info :





Current Location :

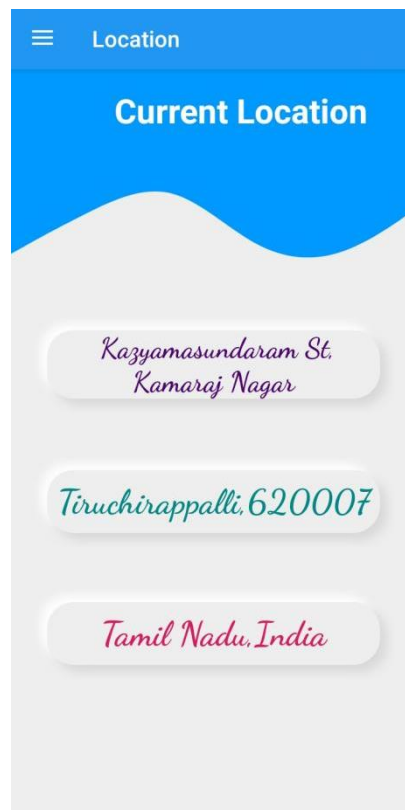
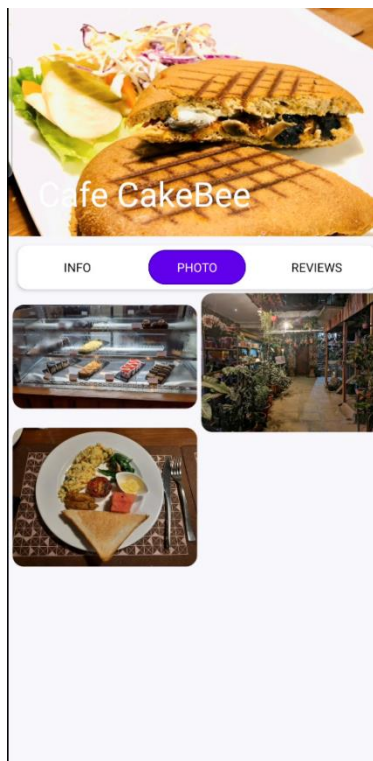
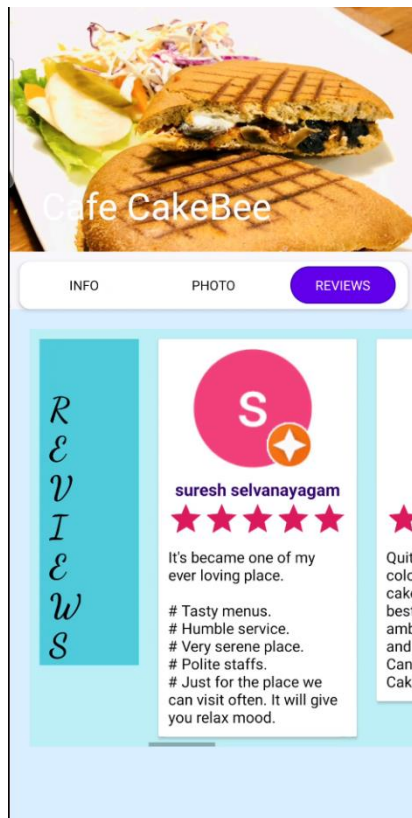


Photo :



Review :



## C.PUPLICATION WITH PLAGARISM REPORT

### PUPLICATION

#### Smart City Management.docx

##### ORIGINALITY REPORT

4%

SIMILARITY INDEX

4%

INTERNET SOURCES

4%

PUBLICATIONS

3%

STUDENT PAPERS

##### PRIMARY SOURCES

1

[www.ijrte.org](http://www.ijrte.org)

Internet Source

2%

2

[ijai.iaescore.com](http://ijai.iaescore.com)

Internet Source

1%

3

[ijarcsse.com](http://ijarcsse.com)

Internet Source

1%

4

"Smart Economy in Smart Cities", Springer  
Science and Business Media LLC, 2017

Publication

<1%