**DIRECTORY APP USING ANDROID NATIVE**

**By**

**S. AFRIN**

**Register No.: 910618621001**

# A PROJECT REPORT

*Submitted to the*

## FACULTY OF INFORMATION & COMMUNICATION ENGINEERING

*In partial fulfillment of the requirements for the award of the*

*degree of*

## MASTER OF COMPUTER APPLICATIONS (MCA)

**IN**

**K.L.N. COLLEGE OF ENGINEERING**

**(An Autonomous Institution)**

**POTTAPALAYAM, SIVAGANGAI – 630 612.**

**ANNA UNIVERSITY, CHENNAI – 600 025.**

**June 2021**

# K.L.N. COLLEGE OF ENGINEERING

**(An Autonomous Institution)**

## DEPARTMENT OF MASTER OF COMPUTER APPLICATIONS (MCA)



# BONAFIDE CERTIFICATE

Certified that this project report titled **“DIRECTORY APP USING ANDROID NATIVE”** is the bonafide work of **Ms. S. AFRIN (Reg. No.: 910618621001)** who carried out the research under my supervision. Certified further, that to the best of my knowledge the work reported herein does not form part of any other project report or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

## Dr.MR. ILANGO, M.C.A., M.Phil., Ph.D. Dr. MR. ILANGO, M.C.A., M.Phil., Ph.D.,

**DIRECTOR / MCA, Internal Guide,**

**K.L.N. College of Engineering. K.L.N. College of Engineering.**

Certified that the candidate was examined by us in the viva-voce examination held at K.L.N. College of Engineering, Sivagangai on…………………… .

**INTERNAL EXAMINER EXTERNAL EXAMINER**

# PROJECT COMPLETION CERTIFICATE

# 

## 

# ACKNOWLEDGEMENT

I would like to express my sincere thanks to our honourable Management and our Respected PRINCIPAL, **Dr. A.V. RAM PRASAD M.E., Ph.D.,** for his support and wonderful opportunity for doing the project in our college.

I would like to express my sincere thanks to our Respected DIRECTOR **Dr. M.R. ILANGO MCA., M.Phil., Ph.D.,** Director of MCA Department, for his permission, guidance and support in preceding the project.

I would like to express my sincere inertness and gratitude to

**Dr. M.R. ILANGO MCA., M.Phil., Ph.D.,** Internal guide of MCA Department, for his immense guidance and valuable suggestion.

I would like to express my sincere thanks to my internship guide, **Mr.SRIDHAR** for his immense guidance and valuable suggestion in completing my project successfully.

I thank my parents, faculty members of MCA Department and my friends for their moral support to complete the project in a successful manner.

A project is never the outcome of a single person’s efforts. It is a confluence of a varied thought process harmoniously integrated into a resourceful product. It is, but natural that I feel indebted to several people for having made this project possible.

## ABSTRACT

The project is to be implemented on Madurai city in which it helps the user to find location based on various category. The goal of the project is to explore how to realize a mobile city guide using the Android platform. The project uses the research method Design Science. The project is evaluated in four aspects including platform evaluation, general functional evaluation, scenario evaluation, and nonfunctional evaluation. The prototype implemented includes basic functionalities of city guide such as showing a map, locating points of interest (POIs) on a map, locating location of a user, retrieving information of POIs, add reviews about POIs, plan a tour, support communication (e.g. phone, short message), show route direction to POIs, add reminder, and choose different kinds of POIs to show on map. Moreover, the project has explored how to integrate current technologies like Google Calendar, Google Map, Browser, Contact application and Phone application into the prototype.

Directories contain a large number of listings under different categories. It provide tools like searching, filtering, pictures, location information with maps, places information, contact details, and more. It is community driven and built on usergenerated content.

# TABLE OF CONTENTS

|  |  |  |
| --- | --- | --- |
| **CHAPTER** | **TITLE** | **PAGE**  **NO** |
|  | **ABSTRACT**  **LIST OF TABLES**  **LIST OF FIGURES** |  |
| **1** | **INTRODUCTION**  Organization Profile    Project Description | **9**    **10** |
| **2** | **SYSTEM ANALYSIS**    Existing System    Proposed System    Feasibility Study | **11**    **11**    **12** |
| **3** | **SYSTEM DESIGN**    System Design Goals    Normalization    Table Design    Data Flow Diagram    E-R Diagram | **14**      **14**    **17**    **19**    **22** |

|  |  |  |
| --- | --- | --- |
| **4** | **MODULE DESCRIPTION** | **24** |
| **5** | **SYSTEM REQUIREMENTS**  **AND ANALYSIS**    Software and Hardware Requirements    Software Explanation | **26**      **26** |
| **6** | **TESTING AND**  **IMPLEMENTATION**    Introduction    Strategic Approach to Software  Testing    System Testing    Implementation | **29**    **30**      **31**    **33** |
| **7** | **CONCLUSION** | **34** |
| **8** | **FUTURE ENHANCEMENT** | **35** |
| **9** | **APPENDIX**    Source Code    Screenshots    Bibliography | **36**    **36**    **65**      **68** |

# LIST OF TABLES

|  |  |  |
| --- | --- | --- |
| **TABLE NO.** | **TABLE NAME** | **PAGE NO.** |
| **3.4.1** | User Registration Table | **17** |
| **3.4.2** | Admin Table | **18** |
| **3.4.3** | City catagories | **18** |

**LIST OF FIGURES**

|  |  |  |
| --- | --- | --- |
| **FIGURE NO.** | **FIGURE NAME** | **PAGE NO.** |
| **3.5.1** | Level 0 DFD | **21** |
| **3.5.2** | Level 1 DFD | **22** |
| **3.6.1** | E-R Diagram | **23** |
| **5.2.1.1** | Android Framework | **27** |
| **6.2.1** | Testing Designs | **31** |

# CHAPTER 1 INTRODUCTION

## ORGANIZATION PROFILE



TECHELF is a software devolpement company, create bonding relationship

Among customers in order to nurture various organizations with our services.

We don’t just deliver results; we deliver it with values and resolute solutions to each of our clients. Our leaders are accountable to institutional performances and to the community as a whole. As Peter Drucker says we embody "The Sprit of Performance" by exhibiting high levels of integrity in our moral code of conduct; focusing on results; building on strengths and leading beyond borders to meet the requirements of stake holders ultimately serving the common good.

# Our Vision

 To lead, to serve, to innovate and inspire in contributing to our society.

# Our Mission

* Continuously improve and innovate to strengthen our clients.
* Delivering outstanding solutions with high quality.

# PROJECT DESCRIPTION

This project acts as an important role in helping people to locate to their places inside a city and promote their business. The project directory app using android native is developed so that users can view the detailed information about the particular city. The users are register by providing details such as Name, MailId. The project also has a login page where the registered user can login, then the user can explore popular places, tourist places, restaurant & residency, jewellery, banking and finance, automobile, shops and they can also promote their shops using ads. Thus this application helps to select the location they need to explore in a efficient manner .The main aim of developing this application is to reduce the time and to give a accurate location in a city. Thus this application provides the required information for quicker decision making.

.

**CHAPTER 2 SYSTEM ANALYSIS**

# EXISTING SYSTEM

In the Existing system the directory app shows the route with a voice recongisation but it is hard to explore places based on the catagories such as tourist places, restaurant & residency, jewellery, banking and finance, automobile, shops ,hospital public utility .

# DISADVANTAGES

* There is a Less catagoriesed list.
* Hard for the user to provide the appropriate keywords.
* Sometimes the route is unapproximate.

# PROPOSED SYSTEM

The motive of the proposed system is to provide the most accurate route and reliable path to the user and to overcome the drawbacks of the existing system. A suggestion-based system determines the similarity to users or items and produces a prediction for the user by taking the weighted average of all the ratings.

Therefore, we use some indexes which are appropriate to evaluate our approaches, such direction, Contact information ,reviews, updates,promotions. Then, we design several evaluation methods to analyze performance.

To predict a rating for an place,we calculate the overall review such as like comments. This enables users to find not only content items they are currently interested in, but also those in which they might become interested.

# ADVANTAGES

* The current data can be added easily and incrementally with an updated set of information.
* It’s used for collecting and analyzing a large amount of information on a user's behaviour, preference and predicting what users will like based on similarity to other users.

# PROJECT FEASIBILITY STUDY

The preliminary investigation examines project feasibility; the likelihood the system will be useful to the organization. The main objective of the feasibility study is to test the Technical, Operational and Economical feasibility for adding new modules and debugging oldest running system. All system is feasible if they are unlimited resources and infinite time. There are aspects in the feasibility study portion of the preliminary investigation:

* Technical Feasibility

* Operational Feasibility

* Economic Feasibility

# TECHNICAL FEASIBILITY

The technical issue usually raised during the feasibility stage of the investigation includes the following:

* Does the necessary technology exist to do what is suggested?

* Do the proposed equipment’s have the technical capacity to hold the data required to use the new system?
* Will the proposed system provide an adequate response to inquiries, regardless of the number or location of users?
* Can the system be upgraded if developed?

Earlier no system existed to cater to the needs of ‘Secure Infrastructure

Implementation System’. The current system developed is technically feasible. It is a browser-based user interface for audit workflow. Thus, it provides easy access to users.

The purpose is to create, establish and maintain a workflow among various entities in order to facilitate all concerned users in their various capacities or roles. Permission to the users would be granted based on the rules specified. Therefore, it provides the technical guarantee of accuracy, reliability and security. The software and hard requirements for the development of this project are not many and are already available or are available as free as open source.

The work for the project is done with the current equipment and existing software technology. Necessary bandwidth exists for providing fast feedback to the users irrespective of the number of users using the system.

# OPERATIONAL FEASIBILITY

The analyst considers the extent the proposed system will fulfil his departments. That is, whether the proposed system covers all aspects of the working system and whether it has considerable improvements. We have found that the proposed “Secure transaction” will certainly have considerable improvements over the existing system.  **ECONOMIC FEASIBILITY**

The proposed system is economically feasible because the cost involved in purchasing the hardware and the software is within approachable. Working in this system need not require a highly qualified professional. The operating-environment costs are marginal. The less time involved also helped in its economic feasibility.

**CHAPTER 3 SYSTEM DESIGN**

# SYSTEM DESIGN GOALS

Software design sits in the technical kernel of the software engineering process and is applied regardless of the development paradigm and area of application. Design is the first step in the development phase for any engineered product or system. The designer’s goal is to produce a model or representation of an entity that will later be built. Beginning, once system requirement has been specified and analyzed, system design is the first of the three technical activities -design, code and test that is required to build and verify software.

The importance can be stated with a single word “Quality”. Design is the place where quality is fostered in software development. The design provides us with representations of software that can assess for quality. Design is the only way that we can accurately translate a customer’s view into a finished software product or system. Software design serves as a foundation for all the software engineering steps that follow. Without a strong design, we risk building an unstable system – one that will be difficult to test, one whose quality cannot be assessed until the last stage.

During design, progressive refinement of data structure, program structure, and procedural details are developed reviewed and documented. System design can be viewed from either a technical or project management perspective. From the technical point of view, the design is comprised of four activities – architectural design, data structure design, interface design and procedural design.

# NORMALIZATION

It is a process of converting a relation to a standard form. The process is used to handle the problems that can arise due to data redundancy, i.e. repetition of data in the database, maintain data integrity as well as handling problems that can arise due to insert, update, delete anomalies.

Decomposing is the process of splitting relations into multiple relations to eliminate anomalies and maintain anomalies and maintain data integrity. To do this we use normal forms or rules for structuring relation.

* **Insertion anomaly**: Inability to add data to the database due to the absence of other data.
* **Deletion anomaly**: Unintended loss of data due to deletion of other data.
* **Update anomaly**: Data inconsistency resulting from data redundancy and partial update
* **Normal Forms**: These are the rules for structuring relations that eliminate anomalies.

# FIRST NORMAL FORM

A relation is said to be in first normal form if the values in the relation are atomic for every attribute in the relation. By this, we mean simply that no attribute value can be a set of values or, as it is sometimes expressed, a repeating group.

# SECOND NORMAL FORM

A relation is said to be in the second normal form is it is in first normal form and it should satisfy any one of the following rules.

* The primary key is not a composite primary key

* No, non- key attributes are present

* Every non- key attribute is fully functionally dependent on a full set of a primary key.

# THIRD NORMAL FORM

A relation is said to be in third normal form if their exits no transitive dependencies.

**Transitive Dependency**: If two non key attributes depend on each other as well as on the primary key, then they are said to be transitively dependent. The above normalization principles were applied to decompose the data in multiple tables, thereby making the data to be maintained in a consistent state.

# 

# 3.4.1 USER REGISTER TABLE

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **S.No.** | **Logical**  **Name** | **Data Type** | **Is**  **Null** | **Comments** |
| 1 | Name | VARCHAR(20) | N | Describes the name of the user |
| 2 | Email | VARCHAR(25) | N | Used to confirm the identity of a user |
| 3 | Password | VARCHAR(20) | N | Used to confirm the identity of a user |
| 4 | Profile-Pic | IMAGE | N | To get the user profile |

3**.4.2 ADMIN TABLE**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **S.No.** | **Logical**  **Name** | **Data Type** | **Is Null** | **Comments** |
| 1 | NAME | VARCHAR(25) | N | Name of the admin |
| 2 | Email | VARCHAR(25) | N | Used to confirm the identity of admin |
| 3 | Password | VARCHAR(20) | N | Used to confirm the identity of a admin |
| 4 | Phone-number | VARCHAR(MAX) | N | To get identity of the admin |
| 6 | Logo | IMAGE | N | To get the identity |
| 7 | Status | VARCHAR(25) | N | To get the status of approve or decline |

## 3.4.3 CITY CATAGORIES

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **S.No.** | **Logical**  **Name** | **Data Type** | **Is**  **Null** | **Comments** |
| 1 | Id | INT | N | Used to confirm the identity of a city. |
| 2 | NAME | VARCHAR(25) | N | Describes the original name of the city. |
| 3 | Picture | IMAGE | N | Used to identify the category of the city |
| 4 | Description | VARCHAR(MAX) | N | Used to describe the city |
| 5 | Added date | Not Null | N | To get the launched date of city |
| 6 | Update-Date | DATE | N | To get the update date of city |
| 7 | Added user Id | VARCHAR(50) | N | To get the information about the user. |

**DATAFLOW DESIGN**

# Definition

A Data Flow Diagram (DFD) is a graphical tool used to describe and analyze the movement of data through the system. It is a graphical representation of the “flow” of data through a computer system or a data or it looks at how data flows through a system. These are a central tool and basic from which the other components are developed. The transformation of data from input to output, through processed, may be described logically and independently of physical components associated with the system. The development of DFD is done at several levels. The flow diagram describes the boxes that describe computations, decisions, interactions & loops. It is important to keep in mind that the flow diagrams are not flowcharts and should not include control elements.

# Characteristics

* Information and/or data flow is represented by a labeled arrow

* Processes (transformations) are represented by labeled circles (bubbles)

* Information sources and sinks are represented by boxes

* Files and depositories are represented by a rounded rectangle or a double line.

# Types

* Logical data flow diagram

* Physical data flow diagram

# Features

* The DFD shows data, not the control loops and decisions are controlled considerations do not appear on a DFD
* The DFD does not indicate the time factor involved in any process, whether the data flow takes place easily daily, weekly, monthly or yearly
* The sequence of events is to bring out on DFD

**DFD Symbols**

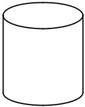
# Process

A process transforms the incoming data flow into outsourcing data flow.



# Data Store

The data source is repositories of data in the system.



# Data Flow

Data flows are pipelines through which packets of information flow. Label the arrows with the name of the data that moves through it.

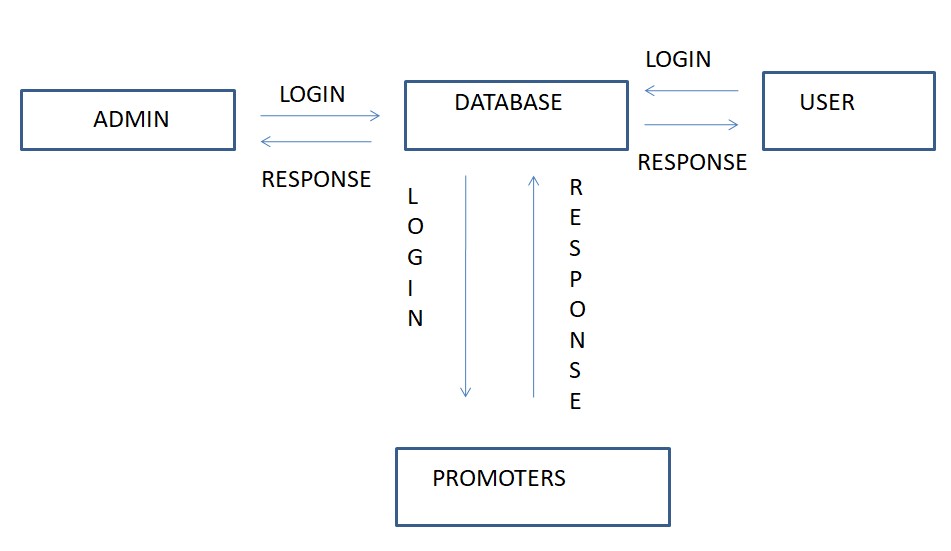


# External Entity

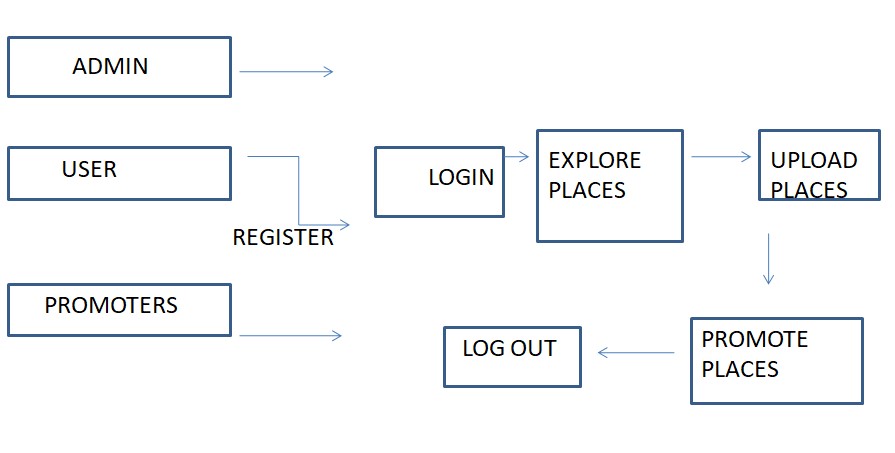
External entities are objects outside the system, with which the system communicates.



# Level 0 DFD



## Level 1 DFD



# E-R DIAGRAM

* The relation upon the system is structured through a conceptual ER-Diagram, which not only specifics the existing entities, but also the standard relations through which the system exists and the cardinalities that are necessary for the system state to continue.
* The Entity-Relationship Diagram (ERD) depicts the relationship between the data objects. The ERD is the notation that is used to conduct, the date modelling activity. The attributes of each data object noted in the ERD can be described to resign a data object description.
* The set of primary components that are identified by the ERD are

* + Data object

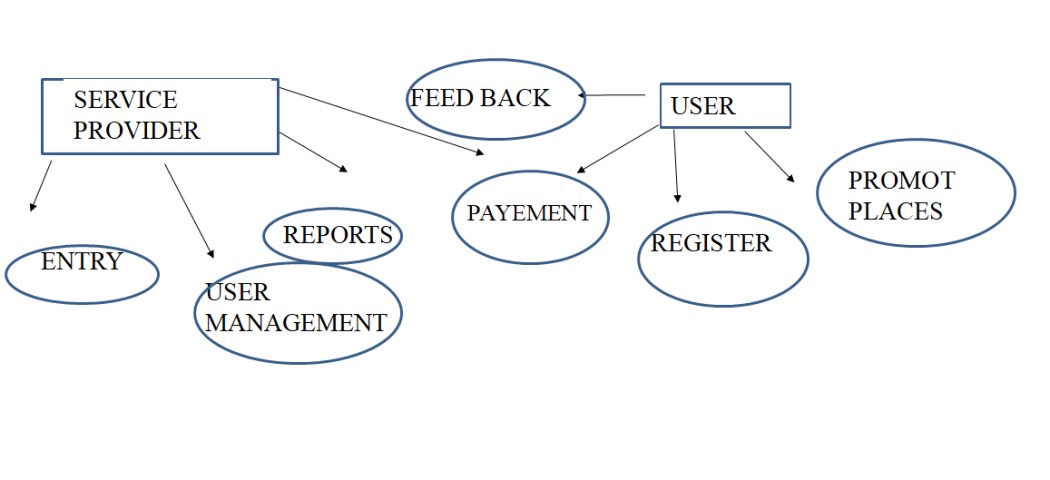
* + Relationships

* + Attributes

* + Various types of indicators.

➢The primary purpose of the ERD is to represent data objects and their

relationships.



**FIGURE 3.6.1 E-R Diagram**

**CHAPTER 4 MODULE DESCRIPTION**

The Modules in this application are:

* REGISTER / LOGIN ACCOUNT
* PROFILE SETTING
* INTERST
* UPLOADED ITEMS
* USER MANAGEMENT

# REGISTER / LOGIN ACCOUNT

Register/ login account is about verification of user detail such as Username and password to verify your personality. Usually, authentication is done with a username and password, although there are various ways to be authenticated. However, the authorization process of this is to give access in the form of approval to the user. So that after the approval the one can able to access by verifying your rights.

# UPLOADED ITEMS

Uploaded item is the process that will upload the new places to this application. The agent uploads the places. The uploaded articles can view all users. User can search based on the catagories , then it will show all the uploaded places to user .

# USER MANAGEMENT

The user management is the process of approval. user uploads places are approved by the admin . The user request to admin, through mail to the admin. Once it is verified , then the places is getting approved to be uploaded. If the places is not matched then it will be rejected.

# INTEREST

Interest is the process of filtering based on searching for content by users. In this filtering consists of trending based on catagories will display while searching. The model used to predict items that the user may have an interest in. Content- based filtering approaches appropriate a series of distinct, identification characteristics of an place in order to justify increased review with similar properties.

# PROFILE SETTING

Profile setting is used to update and modify the changes of the user /admin profile .

**CHAPTER 5 SYSTEM REQUIREMENTS AND ANALYSIS**

**SOFTWARE AND HARDWARE REQUIREMENTS**

# Hardware Requirements

* RAM 8GB
* I3 Processor and Above
* 1 TB Hard Disk Space and Above

# Software Requirements

* WINDOWS OS (10)
* Android Studio
* Java
* Firebase , Mysql

**SOFTWARE EXPLANATION**

# The ANDROID Architecture

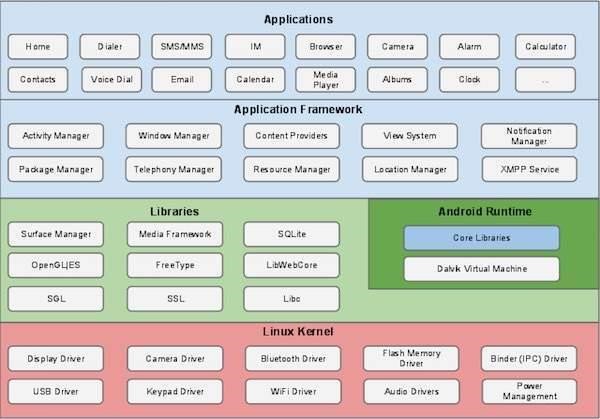
Android is an open source and Linux-based Operating System for mobile devices such as smartphones and tablet computers. Android was developed by the Open Handset Alliance, led by Google, and other companies.

Android offers a unified approach to application development for mobile devices which means developers need only develop for Android, and their applications should be able to run on different devices powered by Android.

The first beta version of the Android Software Development Kit (SDK) was released by Google in 2007 where as the first commercial version, Android 1.0, was released in September 2008.

On June 27, 2012, at the Google I/O conference, Google announced the next Android version, 4.1 Jelly Bean. Jelly Bean is an incremental update, with the primary aim of improving the user interface, both in terms of functionality and performance.

The source code for Android is available under free and open source software licenses. Google publishes most of the code under the Apache License version 2.0 and the rest, Linux kernel changes, under the GNU General Public License version.



**FIGURE 5.2.1.1 .Android Framework**

# Linux kernel

At the bottom of the layers is Linux - Linux 3.6 with approximately 115 patches. This provides a level of abstraction between the device hardware and it contains all the essential hardware drivers like camera, keypad, display etc. Also, the kernel handles all the things that Linux is really good at such as networking and a vast array of device drivers, which take the pain out of interfacing to peripheral hardware.

# Libraries

On top of Linux kernel there is a set of libraries including open-source Web browser engine WebKit, well known library libc, SQLite database which is a useful repository for storage and sharing of application data, libraries to play and record audio and video, SSL libraries responsible for Internet security etc.

# Android Libraries

This category encompasses those Java-based libraries that are specific to Android development. Examples of libraries in this category include the application framework libraries in addition to those that facilitate user interface building, graphics drawing and database access. A summary of some key core Android libraries available to the Android developer is as follows −

* **android.app** − Provides access to the application model and is the cornerstone of all Android applications.
* **android.content** − Facilitates content access, publishing and messaging between applications and application components.
* **android.database** − Used to access data published by content providers and includes SQLite database management classes.
* **android.opengl** − A Java interface to the OpenGL ES 3D graphics rendering API.
* **android.os** − Provides applications with access to standard operating system services including messages, system services and inter-process communication.
* **android.text** − Used to render and manipulate text on a device display.
* **android.view** − The fundamental building blocks of application user interfaces.
* **android.widget** − A rich collection of pre-built user interface components such as buttons, labels, list views, layout managers, radio buttons etc.
* **android.webkit** − A set of classes intended to allow web-browsing capabilities to be built into applications.

Having covered the Java-based core libraries in the Android runtime, it is now time to turn our attention to the C/C++ based libraries contained in this layer of the Android software stack.

Android Runtime

This is the third section of the architecture and available on the second layer from the bottom. This section provides a key component called **Dalvik Virtual Machine** which is a kind of Java Virtual Machine specially designed and optimized for Android.

The Dalvik VM makes use of Linux core features like memory management and multi-threading, which is intrinsic in the Java language. The Dalvik VM enables every Android application to run in its own process, with its own instance of the Dalvik virtual machine.

The Android runtime also provides a set of core libraries which enable Android application developers to write Android applications using standard Java programming language.

Application Framework

The Application Framework layer provides many higher-level services to applications in the form of Java classes. Application developers are allowed to make use of these services in their applications.

The Android framework includes the following key services −

* **Activity Manager** − Controls all aspects of the application lifecycle and activity stack.
* **Content Providers** − Allows applications to publish and share data with other applications.
* **Resource Manager** − Provides access to non-code embedded resources such as strings, color settings and user interface layouts.
* **Notifications Manager** − Allows applications to display alerts and notifications to the user.
* **View System** − An extensible set of views used to create application user interfaces.

## CHAPTER 6 TESTING AND IMPLEMENTATION

# INTRODUCTION

Software testing is a critical element of software quality assurance and represents the ultimate review of specification, design and coding. In fact, testing is the one step in the software engineering process that could be viewed as destructive rather than constructive.

A strategy for software testing integrates software test case design methods into a well- planned series of steps that result in the successful construction of software. Testing is the set of activities that can be planned in advance and conducted systematically. The underlying motivation of program testing is to affirm software quality with methods that can economically and effectively apply both strategic to both large and small-scale systems.

# STRATEGIC APPROACH TO SOFTWARE TESTING

The software engineering process can be viewed as a spiral. Initially, system engineering defines the role of software and leads to software requirement analysis where the information domain, functions, behaviour, performance, constraints and validation criteria for software are established. Moving inward along the spiral, we come to design and finally to coding. To develop computer software we spiral in along streamlines that decrease the level of abstraction at each turn.

A strategy for software testing may also be viewed in the context of the spiral. Unit testing begins at the vertex of the spiral and concentrates on each unit of the software as implemented in source code. Testing progress is done by moving outward along the spiral to integration testing, where the focus is on the design and the construction of the software architecture. Talking another turn on outward on the spiral we encounter validation testing where requirements established as part of software requirements analysis are validated against the software that has been constructed. Finally, we arrive at system testing, where the software and other system elements are tested as a whole. To follow the concept of white-box testing we have tested each form .we have created independently to verify that Data flow is correct, All conditions are exercised to check their validity, All loops are executed on their boundaries.

# BASIC PATH TESTING

The established technique of flow graph with Cyclomatic complexity was used to derive test cases for all the functions. The main steps in deriving test cases were:

Use the design of the code and draw correspondent flow graphs.

Determine the Cyclomatic complexity of the resultant flow graph, using the formula:

## V (G) =E-N+2 Equation 6.3.2.1

Where,

V(G) is Cyclomatic complexity,

E is the number of edges,

N is the number of flow graph nodes, P is the number of predicate nodes.

Determine the basis of the set of linearly independent paths.

**CONDITIONAL TESTING**

In this part of the testing, each of the conditions was tested to both true and false aspects. And all the resulting paths were tested. So that each path that may be generated on particular condition is traced to uncover any possible errors.

# IMPLEMENTATION

Once the system is implemented successfully.

The major tasks of implementation are,

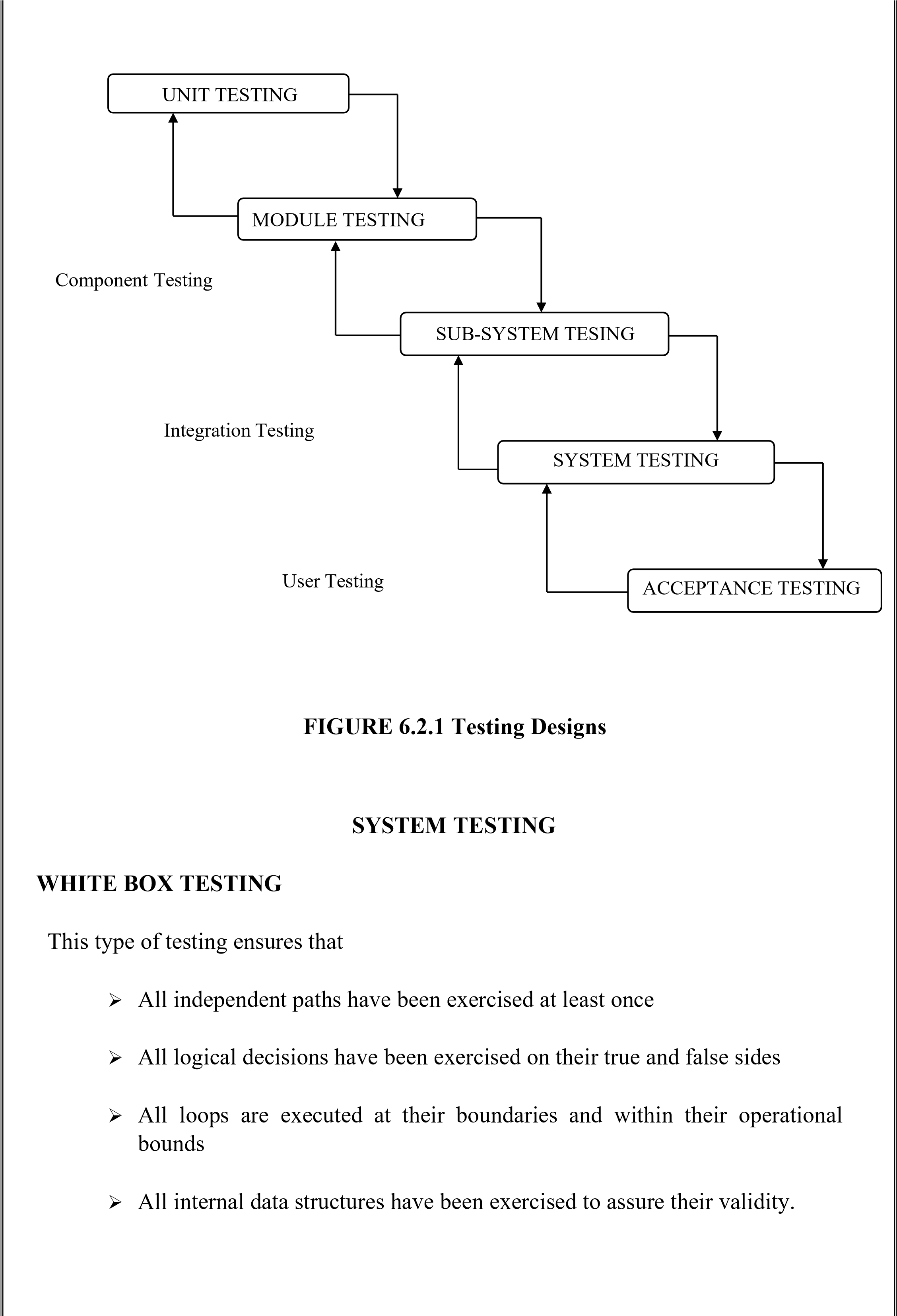
* to read the articles by the reader.
* for the agents to upload the latest trending articles for gaining of more knowledge.
* the news is viewed based on recommendations.

**CHAPTER 7 CONCLUSION**

I stepped away from what I am capable of doing which mainly developing web applications and moved to an interesting field of development using Android. I got to enrich my capacities of self-learning when it comes to learning about new platforms such as Android in duration of less than 4 months and working with it in a real life application. Working on this project would widen the areas in which I can work in future internships or real jobs.

**CHAPTER 8 FUTURE ENHANCEMENT**

Our future work is based on making more convenient to the public who can access this process more easily and also get their details. In this study, a filtering strategy is proposed to achieve personalized route recommendations.The next level of the project to make a app more userfriendly by devolping news and mazagines i nto the app, which help the user read the dialy new inside a city.



# CHAPTER 9 APPENDIX

package com.nmmadurai.city.api;

import androidx.annotation.NonNull;

import androidx.annotation.Nullable;

import androidx.collection.ArrayMap;

import com.nmmadurai.city.utils.Utils;

import org.json.JSONException;

import org.json.JSONObject;

import java.io.IOException;

import java.util.Collections;

import java.util.Map;

import java.util.regex.Matcher;

import java.util.regex.Pattern;

import okhttp3.ResponseBody;

import retrofit2.Response;

/\*\*

\* Common class used by API responses.

\* @param <T>

\*/

public class ApiResponse<T> {

private static final Pattern LINK\_PATTERN = Pattern

.compile("<([^>]\*)>[\\s]\*;[\\s]\*rel=\"([a-zA-Z0-9]+)\"");

private static final Pattern PAGE\_PATTERN = Pattern.compile("\\bpage=(\\d+)");

private static final String NEXT\_LINK = "next";

public int code;

@Nullable

public final T body;

@Nullable

public final String errorMessage;

@NonNull

private final Map<String, String> links;

public ApiResponse(Throwable error) {

code = 500;

body = null;

errorMessage = error.getMessage();

links = Collections.emptyMap();

Utils.psLog("API Response Error : " + errorMessage);

}

public ApiResponse(Response<T> response) {

Utils.psLog("URL : " + response.raw().request().url());

if(response.isSuccessful()) {

Utils.psLog("ApiResponse Successful.");

code = response.code();

body = response.body();

errorMessage = null;

} else {

Utils.psLog("ApiResponse Something wrong.");

String message = null;

try {

ResponseBody responseBody = response.errorBody();

if(responseBody != null) {

message = responseBody.string();

try {

JSONObject jObjError = new JSONObject(message);

Utils.psLog("API Error Response : " + jObjError.getString("message"));

message = jObjError.getString("message");

if(message.contains("##")) {

String [] messageDataList = message.split("##");

if(messageDataList.length > 1) {

code = Integer.parseInt(messageDataList[0]);

message = messageDataList[1];

}

}

}

catch (JSONException e) {

Utils.psErrorLog("JSON Parsing error.", e);

}

}

} catch (NullPointerException ne) {

Utils.psErrorLog("Null Pointer Exception.", ne);

}catch (IOException ignored) {

Utils.psErrorLog("error while parsing response", ignored);

}

if (message == null || message.trim().length() == 0) {

message = response.message();

}

if(code == 0) {

code = response.code();

}

errorMessage = message;

body = null;

}

String linkHeader = response.headers().get("link");

if (linkHeader == null) {

links = Collections.emptyMap();

} else {

links = new ArrayMap<>();

Matcher matcher = LINK\_PATTERN.matcher(linkHeader);

while (matcher.find()) {

int count = matcher.groupCount();

if (count == 2) {

links.put(matcher.group(2), matcher.group(1));

}

}

}

}

public boolean isSuccessful() {

return code >= 200 && code < 300;

}

public Integer getNextPage() {

String next = links.get(NEXT\_LINK);

if (next == null) {

return null;

}

Matcher matcher = PAGE\_PATTERN.matcher(next);

if (!matcher.find() || matcher.groupCount() != 1) {

return null;

}

try {

return Integer.parseInt(matcher.group(1));

} catch (NumberFormatException ex) {

Utils.psLog("cannot parse next page from %s");

return null;

}

}

}

package com.nmmadurai.city.api;

import androidx.lifecycle.LiveData;

import com.nmmadurai.city.viewobject.AboutUs;

import com.nmmadurai.city.viewobject.ApiStatus;

import com.nmmadurai.city.viewobject.Blog;

import com.nmmadurai.city.viewobject.City;

import com.nmmadurai.city.viewobject.Comment;

import com.nmmadurai.city.viewobject.CommentDetail;

import com.nmmadurai.city.viewobject.Image;

import com.nmmadurai.city.viewobject.Item;

import com.nmmadurai.city.viewobject.ItemCategory;

import com.nmmadurai.city.viewobject.ItemCollectionHeader;

import com.nmmadurai.city.viewobject.ItemPaidHistory;

import com.nmmadurai.city.viewobject.ItemSpecs;

import com.nmmadurai.city.viewobject.ItemStatus;

import com.nmmadurai.city.viewobject.ItemSubCategory;

import com.nmmadurai.city.viewobject.Noti;

import com.nmmadurai.city.viewobject.PSAppInfo;

import com.nmmadurai.city.viewobject.Rating;

import com.nmmadurai.city.viewobject.User;

import java.util.List;

import okhttp3.MultipartBody;

import okhttp3.RequestBody;

import retrofit2.Call;

import retrofit2.http.Field;

import retrofit2.http.FormUrlEncoded;

import retrofit2.http.GET;

import retrofit2.http.Multipart;

import retrofit2.http.POST;

import retrofit2.http.Part;

import retrofit2.http.Path;

/\*\*

\* REST API access points

\*/

public interface PSApiService {

//region Get favourite product list

@GET("rest/items/get\_favourite/api\_key/{API\_KEY}/login\_user\_id/{login\_user\_id}/limit/{limit}/offset/{offset}")

LiveData<ApiResponse<List<Item>>> getFavouriteList(@Path("API\_KEY") String apiKey, @Path("login\_user\_id") String login\_user\_id, @Path("limit") String limit, @Path("offset") String offset);

//endregion

//region Post Favourite Product

@FormUrlEncoded

@POST("rest/favourites/press/api\_key/{API\_KEY}")

Call<Item> setPostFavourite(

@Path("API\_KEY") String api\_key,

@Field("item\_id") String itemId,

@Field("user\_id") String userId);

//endregion

//region Get Product Detail

@GET("rest/items/get/api\_key/{API\_KEY}/id/{id}/login\_user\_id/{login\_user\_id}")

LiveData<ApiResponse<Item>> getItemDetail(@Path("API\_KEY") String apiKey, @Path("id") String Id, @Path("login\_user\_id") String login\_user\_id);

//endregion

//region Get Image List

@GET("rest/images/get/api\_key/{API\_KEY}/img\_parent\_id/{img\_parent\_id}/img\_type/{img\_type}")

LiveData<ApiResponse<List<Image>>> getImageList(@Path("API\_KEY") String apiKey, @Path("img\_parent\_id") String img\_parent\_id, @Path("img\_type") String imageType);

//endregion

//region Comments

//region Get commentlist

@GET("rest/commentheaders/get/api\_key/{API\_KEY}/item\_id/{item\_id}/limit/{limit}/offset/{offset}")

LiveData<ApiResponse<List<com.nmmadurai.city.viewobject.Comment>>> getCommentList(@Path("API\_KEY") String apiKey, @Path("item\_id") String itemId, @Path("limit") String limit, @Path("offset") String offset);

//endregion

//region Get comment detail list

@GET("rest/commentdetails/get/api\_key/{API\_KEY}/header\_id/{header\_id}/limit/{limit}/offset/{offset}")

LiveData<ApiResponse<List<com.nmmadurai.city.viewobject.CommentDetail>>> getCommentDetailList(@Path("API\_KEY") String apiKey, @Path("header\_id") String headerId, @Path("limit") String limit, @Path("offset") String offset);

//endregion

//region Get comment detail count

@GET("rest/commentheaders/get/api\_key/{API\_KEY}/id/{id}")

Call<Comment> getRawCommentDetailCount(@Path("API\_KEY") String apiKey, @Path("id") String id);

//endregion

//region Post comment header

@FormUrlEncoded

@POST("rest/commentheaders/press/api\_key/{API\_KEY}")

Call<List<Comment>> rawCommentHeaderPost(

@Path("API\_KEY") String apiKey,

@Field("item\_id") String itemId,

@Field("user\_id") String userId,

@Field("header\_comment") String headerComment);

//endregion

//region Post comment detail

@FormUrlEncoded

@POST("rest/commentdetails/press/api\_key/{API\_KEY}")

Call<List<CommentDetail>> rawCommentDetailPost(

@Path("API\_KEY") String apiKey,

@Field("header\_id") String headerId,

@Field("user\_id") String userId,

@Field("detail\_comment") String detailComment);

//endregion

//endregion

//region Notification

//region Submit Notification Token

@FormUrlEncoded

@POST("rest/notis/register/api\_key/{API\_KEY}")

Call<ApiStatus> rawRegisterNotiToken(@Path("API\_KEY") String apiKey, @Field("platform\_name") String platform,

@Field("device\_id") String deviceId,@Field("user\_id") String userId);

@FormUrlEncoded

@POST("rest/notis/unregister/api\_key/{API\_KEY}")

Call<ApiStatus> rawUnregisterNotiToken(@Path("API\_KEY") String apiKey, @Field("platform\_name")

String platform, @Field("device\_id") String deviceId,@Field("user\_id") String userId);

//endregion

//region Get Notification List

@FormUrlEncoded

@POST("rest/notis/all\_notis/api\_key/{API\_KEY}/limit/{limit}/offset/{offset}")

LiveData<ApiResponse<List<Noti>>> getNotificationList(@Path("API\_KEY") String apiKey,

@Path("limit") String limit,

@Path("offset") String offset,

@Field("user\_id") String userId,

@Field("device\_token") String deviceToken);

//endregion

//region Get Notification detail

@GET("rest/notis/get/api\_key/{API\_KEY}/id/{id}")

LiveData<ApiResponse<Noti>> getNotificationDetail(@Path("API\_KEY") String apiKey, @Path("id") String id);

//endregion

//region Is Read Notificaiton

@FormUrlEncoded

@POST("rest/notis/is\_read/api\_key/{API\_KEY}")

Call<Noti> isReadNoti(

@Path("API\_KEY") String apiKey,

@Field("noti\_id") String noti\_id,

@Field("user\_id") String userId,

@Field("device\_token") String device\_token);

//endregion

@FormUrlEncoded

@POST("rest/users/google\_register/api\_key/{API\_KEY}")

Call<User> postGoogleLogin(

@Path("API\_KEY") String API\_KEY,

@Field("google\_id") String googleId,

@Field("user\_name") String userName,

@Field("user\_email") String userEmail,

@Field("profile\_photo\_url") String profilePhotoUrl,

@Field("device\_token") String deviceToken

);

@FormUrlEncoded

@POST("rest/users/verify/api\_key/{API\_KEY}")

Call<User> verifyEmail(

@Path("API\_KEY") String API\_KEY,

@Field("user\_id") String userId,

@Field("code") String code);

@FormUrlEncoded

@POST("rest/users/request\_code/api\_key/{API\_KEY}")

Call<ApiStatus> resentCodeAgain(

@Path("API\_KEY") String API\_KEY,

@Field("user\_email") String user\_email

);

//region Get category list

@FormUrlEncoded

@POST("rest/categories/search/api\_key/{API\_KEY}/limit/{limit}/offset/{offset}")

LiveData<ApiResponse<List<ItemCategory>>> getSearchCategory(@Path("API\_KEY") String apiKey, @Path("limit") String limit,

@Path("offset") String offset, @Field("order\_by") String order\_by, @Field("shop\_id") String shopId);

//endregion

//region GET User

@GET("rest/users/get/api\_key/{API\_KEY}/user\_id/{user\_id}")

LiveData<ApiResponse<List<User>>> getUser(@Path("API\_KEY") String apiKey, @Path("user\_id") String userId);

//endregion

//region POST Upload Image

@Multipart

@POST("rest/images/upload/api\_key/{API\_KEY}")

LiveData<ApiResponse<User>> doUploadImage(@Path("API\_KEY") String apiKey, @Part("user\_id") RequestBody userId, @Part("file") RequestBody name, @Part MultipartBody.Part file, @Part("platform\_name") RequestBody platformName);

//endregion

//region POST User for Login

@FormUrlEncoded

@POST("rest/users/login/api\_key/{API\_KEY}")

LiveData<ApiResponse<User>> postUserLogin(@Path("API\_KEY") String apiKey, @Field("user\_email") String userEmail, @Field("user\_password") String userPassword,@Field("device\_token") String deviceToken);

//endregion

//region POST User for Register

@FormUrlEncoded

@POST("rest/users/facebook\_register/api\_key/{API\_KEY}")

Call<User> postFBUser(@Path("API\_KEY") String apiKey, @Field("facebook\_id") String facebookId, @Field("user\_name") String userName, @Field("user\_email") String userEmail, @Field("profile\_img\_id") String profilePhotoUrl,@Field("device\_token") String deviceToken);

@FormUrlEncoded

@POST("rest/users/add/api\_key/{API\_KEY}")

Call<User> postUser(@Path("API\_KEY") String apiKey, @Field("user\_id") String userId, @Field("user\_name") String userName, @Field("user\_email") String userEmail, @Field("user\_password") String userPassword, @Field("user\_phone") String userPhone, @Field("device\_token") String deviceToken);

//endregion

//region POST Forgot Password

@FormUrlEncoded

@POST("rest/users/reset/api\_key/{API\_KEY}")

LiveData<ApiResponse<ApiStatus>> postForgotPassword(@Path("API\_KEY") String apiKey, @Field("user\_email") String userEmail);

//endregion

//region PUT User for User Update

@FormUrlEncoded

@POST("rest/users/profile\_update/api\_key/{API\_KEY}")

LiveData<ApiResponse<ApiStatus>> putUser(@Path("API\_KEY") String apiKey,

@Field("user\_id") String loginUserId,

@Field("user\_name") String userName,

@Field("user\_email") String userEmail,

@Field("user\_phone") String userPhone,

@Field("user\_about\_me") String userAboutMe);

//endregion

//region PUT for Password Update

@FormUrlEncoded

@POST("rest/users/password\_update/api\_key/{API\_KEY}")

LiveData<ApiResponse<ApiStatus>> postPasswordUpdate(@Path("API\_KEY") String apiKey, @Field("user\_id") String loginUserId, @Field("user\_password") String password);

//endregion

//endregion

//region About Us

@GET("rest/abouts/get/api\_key/{API\_KEY}")

LiveData<ApiResponse<List<AboutUs>>> getAboutUs(@Path("API\_KEY") String apiKey);

//endregion

//region Contact Us

@FormUrlEncoded

@POST("rest/contacts/add/api\_key/{API\_KEY}")

Call<ApiStatus> rawPostContact(@Path("API\_KEY") String apiKey, @Field("contact\_name") String contactName, @Field("contact\_email") String contactEmail, @Field("contact\_message") String contactMessage, @Field("contact\_phone") String contactPhone);

//endregion

//region GET SubCategory List

@GET("rest/subcategories/get/api\_key/{API\_KEY}")

LiveData<ApiResponse<List<ItemSubCategory>>> getAllSubCategoryList(@Path("API\_KEY") String apiKey);

@GET("rest/subcategories/get/api\_key/{API\_KEY}/cat\_id/{cat\_id}/limit/{limit}/offset/{offset}")

LiveData<ApiResponse<List<ItemSubCategory>>> getSubCategoryList(@Path("API\_KEY") String apiKey, @Path("cat\_id") String catId, @Path("limit") String limit, @Path("offset") String offset);

@GET("rest/subcategories/get/api\_key/{API\_KEY}/limit/{limit}/offset/{offset}/cat\_id/{cat\_id}")

LiveData<ApiResponse<List<ItemSubCategory>>> getSubCategoryListWithCatId(@Path("API\_KEY") String apiKey, @Path("cat\_id") String catId, @Path("limit") String limit, @Path("offset") String offset);

//endregion

//region Delete Shop list by date

@FormUrlEncoded

@POST("rest/appinfo/get\_delete\_history/api\_key/{API\_KEY}")

Call<PSAppInfo> getDeletedHistory(

@Path("API\_KEY") String apiKey,

@Field("start\_date") String start\_date,

@Field("end\_date") String end\_date,

@Field("user\_id") String user\_id);

//endregion

//region Get All Rating List

@GET("rest/rates/get/api\_key/{API\_KEY}/item\_id/{item\_id}/limit/{limit}/offset/{offset}")

LiveData<ApiResponse<List<Rating>>> getAllRatingList(@Path("API\_KEY") String apiKey, @Path("item\_id") String itemId, @Path("limit") String limit, @Path("offset") String offset);

//endregion

//region Post Rating

@FormUrlEncoded

@POST("rest/rates/add\_rating/api\_key/{API\_KEY}")

Call<Rating> postRating(

@Path("API\_KEY") String api\_key,

@Field("title") String title,

@Field("description") String description,

@Field("rating") String rating,

@Field("user\_id") String userId,

@Field("item\_id") String item\_id);

//endregion

//endregion

//region Touch Count

@FormUrlEncoded

@POST("rest/touches/add\_touch/api\_key/{API\_KEY}")

Call<ApiStatus> setrawPostTouchCount(

@Path("API\_KEY") String apiKey,

@Field("type\_id") String typeId,

@Field("type\_name") String typeName,

@Field("user\_id") String userId);

//endregion

//region News|Blog

@GET("rest/feeds/get/api\_key/{API\_KEY}/limit/{limit}/offset/{offset}")

LiveData<ApiResponse<List<Blog>>> getAllNewsFeed(@Path("API\_KEY") String api\_key, @Path("limit") String limit, @Path("offset") String offset);

@GET("rest/feeds/get/api\_key/{API\_KEY}/id/{id}")

LiveData<ApiResponse<Blog>> getNewsById(@Path("API\_KEY") String api\_key, @Path("id") String id);

//endregion

///////////////////////////////////////////////////////////////////////////////////////////////////MultiCity/////////////////////////////////////////////////////////////////

//region City Info

@GET("rest/cities/get\_city\_info/api\_key/{API\_KEY}")

LiveData<ApiResponse<City>> getCityInfo(@Path("API\_KEY") String api\_key);

// @FormUrlEncoded

// @POST("rest/cities/search/api\_key/{API\_KEY}/limit/{limit}/offset/{offset}")

// LiveData<ApiResponse<List<City>>> searchCity(

// @Path("API\_KEY") String API\_KEY,

// @Path("limit") String limit,

// @Path("offset") String offset,

// @Field("id") String id,

// @Field("keyword") String keyword,

// @Field("is\_featured") String is\_featured,

// @Field("order\_by") String order\_by,

// @Field("order\_type") String order\_type);

//endregion

//region SearchItem

@FormUrlEncoded

@POST("rest/items/search/api\_key/{API\_KEY}/limit/{limit}/offset/{offset}/login\_user\_id/{login\_user\_id}")

LiveData<ApiResponse<List<Item>>> searchItem(

@Path("API\_KEY") String API\_KEY,

@Path("limit") String limit,

@Path("offset") String offset,

@Path("login\_user\_id") String login\_user\_id,

@Field("keyword") String keyword,

@Field("cat\_id") String cat\_id,

@Field("sub\_cat\_id") String sub\_cat\_id,

@Field("order\_by") String order\_by,

@Field("order\_type") String order\_type,

@Field("rating\_value") String rating\_value,

@Field("is\_featured") String is\_featured,

@Field("is\_promotion") String is\_promotion,

@Field("lat") String lat,

@Field("lng") String lng,

@Field("miles") String miles,

@Field("added\_user\_id") String addedUserId,

@Field("is\_paid") String isPaid,

@Field("item\_status\_id") String status);

//endregion

//region ItemCategory

@GET("rest/categories/get/api\_key/{API\_KEY}/limit/{limit}/offset/{offset}")

LiveData<ApiResponse<List<ItemCategory>>> getCityCategory(

@Path("API\_KEY") String API\_KEY,

@Path("limit") String limit,

@Path("offset") String offset

);

//region ItemCategory

//region Collection

@GET("rest/collections/get/api\_key/{API\_KEY}/limit/{limit}/offset/{offset}")

LiveData<ApiResponse<List<ItemCollectionHeader>>> getCollectionHeaderByCityId(

@Path("API\_KEY") String API\_KEY,

@Path("limit") String limit,

@Path("offset") String offset

);

@GET("rest/items/all\_collection\_items/api\_key/{API\_KEY}/limit/{limit}/offset/{offset}/id/{id}")

LiveData<ApiResponse<List<Item>>> getCollectionItems(@Path("API\_KEY") String apiKey, @Path("limit") String limit, @Path("offset") String offset, @Path("id") String id);

//regionFor ItemUpload

@FormUrlEncoded

@POST("rest/items/submit\_items/api\_key/{API\_KEY}")

LiveData<ApiResponse<Item>> saveItem(

@Path("API\_KEY") String API\_KEY,

@Field("user\_id") String userId,

@Field("city\_id") String cityId,

@Field("cat\_id") String categoryId,

@Field("sub\_cat\_id") String subCategoryId,

@Field("status") String status,

@Field("name") String name,

@Field("description") String description,

@Field("search\_tag") String searchTag,

@Field("highlight\_information") String highlightInformation,

@Field("is\_featured") String isFeatured,

@Field("lat") String latitude,

@Field("lng") String longitude,

@Field("opening\_hour") String openingHour,

@Field("closing\_hour") String closingHour,

@Field("is\_promotion") String isPromotion,

@Field("phone1") String phoneOne,

@Field("phone2") String phoneTwo,

@Field("phone3") String phoneThree,

@Field("email") String email,

@Field("address") String address,

@Field("facebook") String facebook,

@Field("google\_plus") String googlePlus,

@Field("twitter") String twitter,

@Field("youtube") String youtube,

@Field("instagram") String instagram,

@Field("pinterest") String pinterest,

@Field("website") String website,

@Field("whatsapp") String whatsapp,

@Field("messenger") String messenger,

@Field("time\_remark") String timeRemark,

@Field("terms") String terms,

@Field("cancelation\_policy") String cancelationPolicy,

@Field("additional\_info") String additionalInfo,

@Field("id") String id);

//endregion

//regionFor item image

@Multipart

@POST("rest/images/upload\_item/api\_key/{API\_KEY}")

LiveData<ApiResponse<Image>> uploadItemImage(

@Path("API\_KEY") String API\_KEY,

@Part("item\_id") RequestBody itemId,

@Part("img\_desc") RequestBody imgDesc,

@Part MultipartBody.Part file,

@Part("img\_id") RequestBody id

);

//endregion

//regionFor specification list

@GET("rest/items/item\_specification/api\_key/{API\_KEY}/item\_id/{item\_id}/limit/{limit}/offset/{offset}")

LiveData<ApiResponse<List<ItemSpecs>>> getAllSpecificationByItemId(@Path("API\_KEY") String API\_KEY, @Path("item\_id") String itemId, @Path("limit") String limit, @Path("offset") String offset);

//endregion

//regionFor add specification

@FormUrlEncoded

@POST("rest/items/add\_spec/api\_key/{API\_KEY}")

LiveData<ApiResponse<ItemSpecs>> addSpecification (@Path("API\_KEY") String API\_KEY, @Field("item\_id") String itemId, @Field("name") String name, @Field("description") String description, @Field("id") String id);

//endregion

//regionFor delete specification

@FormUrlEncoded

@POST("rest/items/delete\_spec/api\_key/{API\_KEY}")

Call<ApiStatus> deleteSpecification(@Path("API\_KEY") String API\_KEY, @Field("item\_id") String itemId, @Field("id") String id);

//endregion

//regionFor image in item upload after image upload

@GET("rest/images/item\_image/api\_key/{API\_KEY}/item\_id/{item\_id}/limit/{limit}/offset/{offset}")

LiveData<ApiResponse<List<Image>>> getImagesInItemUploadAfterImageUpload(@Path("API\_KEY") String API\_KEY, @Path("item\_id") String itemId, @Path("limit") String limit, @Path("offset") String offset);

//endregion

//regionFor delete image

@FormUrlEncoded

@POST("rest/images/delete\_image/api\_key/{API\_KEY}")

Call<ApiStatus> deleteImage(

@Path("API\_KEY") String API\_KEY,

@Field("item\_id") String shopId,

@Field("img\_id") String imgId

);

//endregion

//regionFor delete item

@FormUrlEncoded

@POST("rest/items/item\_delete/api\_key/{API\_KEY}")

Call<ApiStatus> deleteItem(

@Path("API\_KEY") String API\_KEY,

@Field("item\_id") String itemId,

@Field("user\_id") String userId

);

//endregion

//regionFor item status

@GET("rest/items/item\_status/api\_key/{API\_KEY}/limit/{limit}/offset/{offset}")

LiveData<ApiResponse<List<ItemStatus>>> getItemStatus(

@Path("API\_KEY") String API\_KEY,

@Path("limit") String limit,

@Path("offset") String offset

);

@FormUrlEncoded

@POST("rest/users/phone\_register/api\_key/{API\_KEY}")

Call<User> postPhoneLogin(

@Path("API\_KEY") String API\_KEY,

@Field("phone\_id") String phoneId,

@Field("user\_name") String userName,

@Field("user\_phone") String userPhone,

@Field("device\_token") String deviceToken

);

//endregion

//region upload item paid history

@FormUrlEncoded

@POST("rest/paid\_items/add/api\_key/{API\_KEY}")

Call<ItemPaidHistory> uploadItemPaidHistory(

@Path("API\_KEY") String API\_KEY,

@Field("item\_id") String itemId,

@Field("amount") String amount,

@Field("start\_date") String startDate,

@Field("how\_many\_day") String howManyDay,

@Field("payment\_method") String paymentMethod,

@Field("payment\_method\_nonce") String paymentMethodNonce,

@Field("start\_timestamp") String startTimeStamp,

@Field("razor\_id") String razorId);

// endregion

//region get paid ad

@GET("rest/paid\_items/get/api\_key/{API\_KEY}/login\_user\_id/{login\_user\_id}/limit/{limit}/offset/{offset}")

LiveData<ApiResponse<List<ItemPaidHistory>>> getPaidAd(

@Path("API\_KEY") String apiKey,

@Path("login\_user\_id") String login\_user\_id,

@Path("limit") String limit,

@Path("offset") String offset

);

//endregion

//region Paypal

@GET("rest/paypal/get\_token/api\_key/{API\_KEY}")

Call<ApiStatus> getPaypalToken(@Path("API\_KEY") String apiKey);

//endregion

}

package com.nmmadurai.city.ui.city.detail;

import android.content.Context;

import android.content.SharedPreferences;

import android.os.Bundle;

import android.preference.PreferenceManager;

import androidx.databinding.DataBindingUtil;

import com.nmmadurai.city.Config;

import com.nmmadurai.city.R;

import com.nmmadurai.city.databinding.ActivityCityBinding;

import com.nmmadurai.city.ui.common.PSAppCompactActivity;

import com.nmmadurai.city.utils.Constants;

import com.nmmadurai.city.utils.MyContextWrapper;

public class CityActivity extends PSAppCompactActivity {

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

ActivityCityBinding activityFilteringBinding = DataBindingUtil.setContentView(this, R.layout.activity\_city);

initUI(activityFilteringBinding);

}

@Override

protected void attachBaseContext(Context newBase) {

SharedPreferences preferences = PreferenceManager.getDefaultSharedPreferences(newBase);

String LANG\_CURRENT = preferences.getString(Constants.LANGUAGE\_CODE, Config.DEFAULT\_LANGUAGE);

String CURRENT\_LANG\_COUNTRY\_CODE = preferences.getString(Constants.LANGUAGE\_COUNTRY\_CODE, Config.DEFAULT\_LANGUAGE\_COUNTRY\_CODE);

super.attachBaseContext(MyContextWrapper.wrap(newBase, LANG\_CURRENT, CURRENT\_LANG\_COUNTRY\_CODE, true));

}

private void initUI(ActivityCityBinding binding) {

initToolbar(binding.toolbar, getIntent().getStringExtra(Constants.CITY\_NAME));

setupFragment(new CityFragment());

}

}package com.nmmadurai.city.ui.city.detail;

import android.os.Bundle;

import android.view.LayoutInflater;

import android.view.View;

import android.view.ViewGroup;

import androidx.annotation.NonNull;

import androidx.annotation.VisibleForTesting;

import androidx.databinding.DataBindingUtil;

import androidx.lifecycle.ViewModelProvider;

import com.google.android.gms.ads.AdRequest;

import com.google.android.gms.maps.CameraUpdateFactory;

import com.google.android.gms.maps.GoogleMap;

import com.google.android.gms.maps.MapsInitializer;

import com.google.android.gms.maps.OnMapReadyCallback;

import com.google.android.gms.maps.model.LatLng;

import com.google.android.gms.maps.model.MarkerOptions;

import com.nmmadurai.city.Config;

import com.nmmadurai.city.R;

import com.nmmadurai.city.binding.FragmentDataBindingComponent;

import com.nmmadurai.city.databinding.FragmentCityBinding;

import com.nmmadurai.city.ui.common.PSFragment;

import com.nmmadurai.city.utils.AutoClearedValue;

import com.nmmadurai.city.utils.Utils;

import com.nmmadurai.city.viewmodel.city.CityViewModel;

import com.nmmadurai.city.viewobject.City;

public class CityFragment extends PSFragment implements OnMapReadyCallback {

//region Variables

private final androidx.databinding.DataBindingComponent dataBindingComponent = new FragmentDataBindingComponent(this);

private GoogleMap map;

private CityViewModel cityViewModel;

private Bundle bundle;

@VisibleForTesting

private AutoClearedValue<FragmentCityBinding> binding;

//endregion

//region Override Methods

@Override

public View onCreateView(@NonNull LayoutInflater inflater, ViewGroup container,

Bundle savedInstanceState) {

// Inflate the layout for this fragment

FragmentCityBinding dataBinding = DataBindingUtil.inflate(inflater, R.layout.fragment\_city, container, false, dataBindingComponent);

binding = new AutoClearedValue<>(this, dataBinding);

setHasOptionsMenu(true);

bundle = savedInstanceState;

binding.get().mapView.onCreate(bundle);

return binding.get().getRoot();

}

private void initializeMap(Bundle savedInstanceState) {

try {

if (this.getActivity() != null) {

MapsInitializer.initialize(this.getActivity());

}

} catch (Exception e) {

e.printStackTrace();

}

binding.get().mapView.onCreate(savedInstanceState);

binding.get().mapView.onResume();

binding.get().mapView.getMapAsync(googleMap -> {

map = googleMap;

map.addMarker(new MarkerOptions()

.position(new LatLng(Double.valueOf(cityViewModel.lat), Double.valueOf(cityViewModel.lng)))

.title("City Name"));

//zoom

if (!cityViewModel.lat.isEmpty() && !cityViewModel.lng.isEmpty()) {

int zoomlevel = 8;

// Animating to the touched position

map.animateCamera(CameraUpdateFactory.newLatLngZoom(new LatLng(Double.parseDouble(cityViewModel.lat), Double.parseDouble(cityViewModel.lng)), zoomlevel));

}

});

}

@Override

public void onDestroyView() {

super.onDestroyView();

}

@Override

public void onDestroy() {

if (binding != null) {

if (binding.get() != null) {

if (binding.get().mapView != null) {

binding.get().mapView.onDestroy();

if (map != null) {

map.clear();

}

}

}

}

super.onDestroy();

}

@Override

public void onLowMemory() {

binding.get().mapView.onLowMemory();

super.onLowMemory();

}

@Override

public void onPause() {

binding.get().mapView.onPause();

super.onPause();

}

@Override

public void onResume() {

binding.get().mapView.onResume();

super.onResume();

}

@Override

protected void initUIAndActions() {

if (Config.SHOW\_ADMOB && connectivity.isConnected()) {

AdRequest adRequest = new AdRequest.Builder()

.build();

binding.get().adView.loadAd(adRequest);

} else {

binding.get().adView.setVisibility(View.GONE);

}

}

@Override

protected void initViewModels() {

cityViewModel = new ViewModelProvider(this, viewModelFactory).get(CityViewModel.class);

}

@Override

protected void initAdapters() {

}

// private void replaceAboutUsData()

@Override

protected void initData() {

cityViewModel.setCityInfoObj();

cityViewModel.getCityInfoData().observe(this, resource -> {

if (resource != null) {

Utils.psLog("Got Data" + resource.message + resource.toString());

switch (resource.status) {

case LOADING:

// Loading State

// Data are from Local DB

if (resource.data != null) {

fadeIn(binding.get().getRoot());

binding.get().setCity(resource.data);

setAboutUsData(resource.data);

addLatLong(resource.data);

}

break;

case SUCCESS:

// Success State

// Data are from Server

if (resource.data != null) {

binding.get().setCity(resource.data);

setAboutUsData(resource.data);

addLatLong(resource.data);

bindingMap();

}

break;

case ERROR:

// Error State

break;

default:

// Default

break;

}

} else {

// Init Object or Empty Data

Utils.psLog("Empty Data");

}

// we don't need any null checks here for the adapter since LiveData guarantees that

// it won't call us if fragment is stopped or not started.

if (resource != null) {

Utils.psLog("Got Data Of About Us.");

} else {

//noinspection Constant Conditions

Utils.psLog("No Data of About Us.");

}

});

}

private void setAboutUsData(City city) {

binding.get().setCity(city);

cityViewModel.selectedCityId = city.id;

// For Contact

// For SourceAddress

}

@Override

public void onMapReady(GoogleMap googleMap) {

}

private void addLatLong(City city) {

cityViewModel.lat = city.lat;

cityViewModel.lng = city.lng;

}

private void bindingMap() {

if (!cityViewModel.lat.isEmpty() && !cityViewModel.lng.isEmpty()) {

initializeMap(bundle);

}

}

//endregion

}

package com.nmmadurai.city.ui.imageupload;

import android.content.Context;

import android.content.Intent;

import android.content.SharedPreferences;

import android.os.Bundle;

import android.preference.PreferenceManager;

import androidx.databinding.DataBindingUtil;

import androidx.fragment.app.Fragment;

import com.nmmadurai.city.Config;

import com.nmmadurai.city.R;

import com.nmmadurai.city.databinding.ActivityImageUploadBinding;

import com.nmmadurai.city.ui.common.PSAppCompactActivity;

import com.nmmadurai.city.utils.Constants;

import com.nmmadurai.city.utils.MyContextWrapper;

import com.nmmadurai.city.utils.Utils;

import java.util.Objects;

public class ImageUploadActivity extends PSAppCompactActivity {

public int flag;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

ActivityImageUploadBinding dataBinding = DataBindingUtil.setContentView(this, R.layout.activity\_image\_upload);

flag = Objects.requireNonNull(getIntent().getIntExtra(Constants.FLAG, 0));

initUI(dataBinding);

}

@Override

protected void attachBaseContext(Context newBase) {

SharedPreferences preferences = PreferenceManager.getDefaultSharedPreferences(newBase);

String LANG\_CURRENT = preferences.getString(Constants.LANGUAGE\_CODE, Config.DEFAULT\_LANGUAGE);

String CURRENT\_LANG\_COUNTRY\_CODE = preferences.getString(Constants.LANGUAGE\_COUNTRY\_CODE, Config.DEFAULT\_LANGUAGE\_COUNTRY\_CODE);

super.attachBaseContext(MyContextWrapper.wrap(newBase, LANG\_CURRENT, CURRENT\_LANG\_COUNTRY\_CODE, true));

}

@Override

public void onActivityResult(int requestCode, int resultCode, Intent data) {

super.onActivityResult(requestCode,resultCode,data);

Utils.psLog("Inside Result MainActivity");

Fragment fragment = getSupportFragmentManager().findFragmentById(R.id.content\_frame);

assert fragment != null;

fragment.onActivityResult(requestCode, resultCode, data);

}

public void initUI(ActivityImageUploadBinding binding) {

if (flag == Constants.IMAGE\_UPLOAD\_ITEM) {

initToolbar(binding.categoryToolBar, getResources().getString(R.string.image\_upload\_\_title\_item));

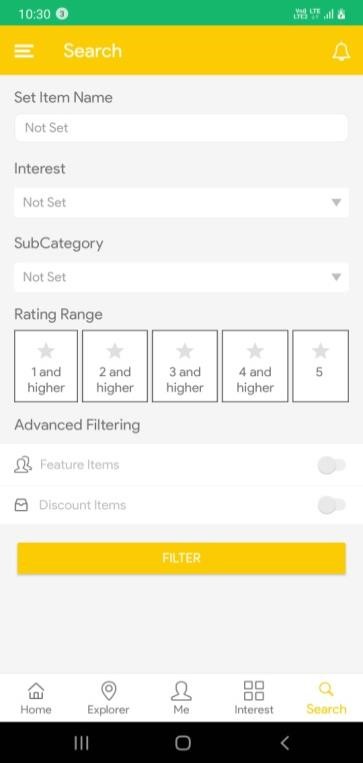
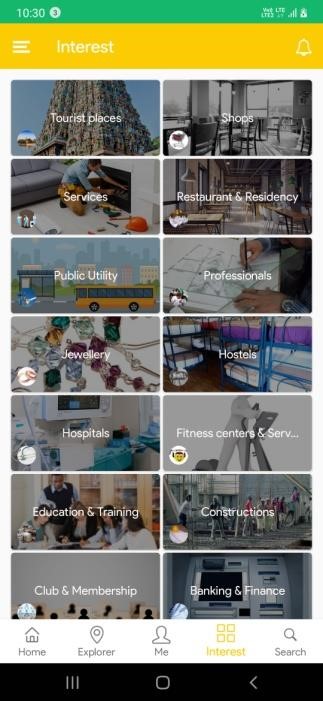
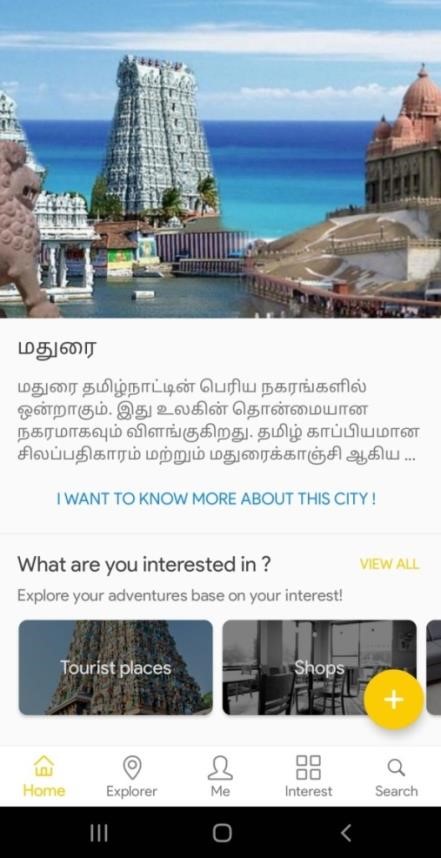
setupFragment(new ItemEntryImageUploadFragment());

}

}

}

**SCREENSHOTS**



## BIBLOGRAPHY

1. [**Headfirst Android Development: A Brain-Friendly Guide 1st Edition**](https://geni.us/KqzA)  Headfirst Android Development is a book written by Davi.
2. [**Android Programming for Beginners**](https://geni.us/HE9hvlN)

Android Programming for Beginners is written by John Horton.