

VISVESVARAYA TECHNOLOGICAL UNIVERSITY

JNANA SANGAMA, BELAGAVI – 590 018



A Mini Project Report on

“Tourism Management System”

Submitted in partial fulfilment of the requirements as a part of the

Mobile Application Development Laboratory with

Mini Project (18CSMP68)

For the award of degree of

**Bachelor of Engineering
in**

Information Science and Engineering

Submitted by

SUHITH

1RN20IS165

TEJAS V KANGOD

1RN20IS172

VARUN

1RN20IS180

Mini Project Coordinator

Ms. Purnima S M

Assistant Professor

Dept. of ISE, RNSIT



Department of Information Science and Engineering

RNS Institute of Technology

**Channasandra, Dr. Vishnuvardhan Road, RR Nagar Post,
Bengaluru – 560 098**

2022 -2023

RNS Institute of Technology

Channasandra, Dr. Vishnuvardhan Road, RR Nagar Post,

Bengaluru – 560 098

DEPARTMENT OF INFORMATION SCIENCE & ENGINEERING



CERTIFICATE

This is to certify that the mini project report entitled **TIME TO TRAVEL** has been successfully completed by **TEJAS V KANGOD** bearing USN **1RN20IS172** and **VARUN** bearing USN **1RN20IS180** and **SUHITH** bearing USN **1RN20IS165** presently VI semester student of **RNS Institute of Technology** in partial fulfilment of the requirements as a part of the **Mobile Application Development Laboratory (18CSMP68)** for the award of the degree of **Bachelor of Engineering in Information Science and Engineering** under **Vishveshvaraya Technological University, Belagavi** during academic year **2023 – 2024**

It is certified that all corrections/suggestions indicated for Internal Assessment have been incorporated in the report and deposited in the departmental library. The mini project report has been approved as it satisfies the academic requirements as a part of Mobile Application Development Laboratory.

Ms. Purnima S M

Coordinator
Assistant Professor

Dr. Suresh L

Professor and HOD

Dr Ramesh Babu H S

Principal

External Viva

Name of the Examiners

Signature with date

1. _

2. _

DECLARATION

We, **SUHITH PR [USN: 1RN20IS165]** , **VARUN D [USN: 1RN20IS180]** and **TEJAS V KANGOD [USN: 1RN20IS172]** students of VI Semester BE, in Information Science and Engineering, RNS Institute of Technology hereby declare that the Project entitled “**TOURISM MANAGEMENT SYSTEM**” has been carried out by us and submitted in partial fulfillment of the requirements for the VI Semester of degree of Bachelor of Engineering in Information Science and Engineering of Visvesvaraya Technological University, Belagavi during academic year 2022-2023.

Place: Bengaluru

Date:

SUHITH P R
1RN20IS165

TEJAS V KANGOD
1RN20IS172

VARUN D
1RN20IS180

ABSTRACT

“Time to travel” is a travel management app. The app is created using android studio where the user can first sign up and make an account post which the user logs into respective account and can book a cab, hotel, guide or a combo of services according to the dates. The main objective of the app is to facilitate easy and fast booking of services to users so that travel can be made convenient and effortless. Computer applications have become a powerful medium for rapid and economical distribution of services. There is virtually no area where we cannot use computer applications to our advantage. This travel app provide an effortless way of booking and using travel services.

ACKNOWLEDGMENT

The fulfilment and rapture that go with the fruitful finishing of any assignment would be inadequate without the specifying the people who made it conceivable, whose steady direction and support delegated the endeavors with success.

I would like to profoundly thank **Management** of **RNS Institute of Technology** for providing such a healthy environment to carry out this Mobile Application Development Laboratory with Mini Project Work.

I would like to express my thanks to our Principal **Dr. Ramesh Babu H S** for his support and inspired us towards the attainment of knowledge.

I wish to place on record my words of gratitude to **Dr. Suresh L**, Professor and Head of the Department, Information Science and Engineering, for being the enzyme and master mind behind our Mobile Application Development Laboratory with Mini Project Work.

I would like to express our profound and cordial gratitude to my Mini Project Coordinator, **Ms. Purnima S M** Assistant Professor, Department of Information Science and Engineering for her valuable guidance, constructive comments, continuous encouragement throughout the Mini Project Work and guidance in preparing report.

I would like to thank all other teaching and non-teaching staff of Information Science & Engineering who have directly or indirectly helped us to carry out the Mini Project Work.

Also, I would like to acknowledge and thank our parents who are source of inspiration and instrumental in carrying out this Mini Project Work.

SUHITH P R

1RN20IS165

TEJAS V KANGOD

1RN20IS172

VARUN D

1RN20IS180

TABLE OF CONTENTS

CERTIFICATE	
DECLARATION	iii
ABSTRACT	iv
ACKNOWLEDGMENT	v
TABLE OF CONTENTS	vi
LIST OF FIGURES	vii
ABBREVIATIONS	viii
1. INTRODUCTION TO ANDROID	1
1.1 History	1
1.2 Android Version	1
1.3 Architecture Of Android	3
1.4 Android Studio Installation	5
2. INTRODUCTION TO PROJECT	7
2.1 Overview Of The Project	8
2.2 Aim Of The Project	8
3. About Software	8
3.1 Feature of Android Studio	8
3.2 Android studio Project Structure	9
3.3 Android studio Main Window	9
3.4 Gradle Build System	10
3.5 Install and Setup Android Studio and java JDK	10
4. System Requirement	12
5. Design and Implementation	14
6. Snapshots	26
7. Conclusion, Future Enhancements and Reference	31

LIST OF FIGURES

Figure	Description	Page no
Figure 1.1	Android Version	2
Figure 1.2	Android architecture Diagram	3
Figure 3.1	Android Project Structure	9
Figure 3.2	Android Main Window	9
Figure 5.1	Flowchart of Application	14
Figure 5.2	Recycle View Design	15
Figure 6.1	Sign up page	26
Figure 6.2	Sign in page	26
Figure 6.3	Home page	27
Figure 6.4	Timing and destination page	27
Figure 6.5	Explore page	28
Figure 6.6	Hotel page	28
Figure 6.7	Guide page	29
Figure 6.8	Combo page	29
Figure 6.9	Details page	30
Figure 6.10	Booking page	30

ABBREVIATIONS

iOS - iPhone Operating System

Inc. - Incorporated

OS - Operating System

SDK - Software Development Kit

DVM - Dalvik Virtual Machine

JVM - Java Virtual Machine

IDE - Integrated Development Environment

RAM - Random Access Memory

XML - Extensible Markup Language

UI - User Interface

DDMS - Dalvik Debug Monitor Server

Chapter 1

INTRODUCTION TO ANDRIOD

1.1 History

In past mobile phones were used only to make calls but with the introduction of smartphone the mobile phone has evolved to a low powered hand-held processing system. This evolution was caused by the operating system for the mobile phones making them smart that have processing and storage of their own. Now the mobile provides numerous functionalities from calling to texting, multimedia sharing, emails, socializing applications, word processor, excel sheets to various multiplayer games and much more.

The operating system for these hand-held devices is iOS by Apple Inc., Windows by Windows Inc. and Android by Google. Among the competitors in smartphone operating system industry Android holds the largest market share in terms of units shipped worldwide and number of users.

Android is an open-source operating system based on Linux kernel on which applications run on an application framework that controls the activities supported by the libraries and Dalvik virtual machine which compiles and converts all java class files into a single file. There can be number of virtual machines running simultaneously on a single device handling different applications or instances of an application.

Android operating system provides memory management, process management to the applications and services running. Each release of android improved user experience and brought enhanced features. In 2012 Android became the most popular operating system for mobile devices, surpassing Apple's iOS, and, as of 2020, about 75 percent of mobile devices run Android.

1.2 Android Versions

The development of the Android operating system was started in 2003 by Android, Inc. Later on, it was purchased by Google in 2005. The beta version of Android OS was released on

November 5, 2007, while the software development kit (SDK) was released on November 12, 2007.

The first Android mobile was publicly released with Android 1.0 of the T-Mobile G1 (aka HTC Dream) in October 2008. The first Android version which was released under the numerical order format was Android 10.

Code name	Version numbers	API level	Release date
No codename	1.0	1	September 23, 2008
No codename	1.1	2	February 9, 2009
Cupcake	1.5	3	April 27, 2009
Donut	1.6	4	September 15, 2009
Eclair	2.0 - 2.1	5 - 7	October 26, 2009
Froyo	2.2 - 2.2.3	8	May 20, 2010
Gingerbread	2.3 - 2.3.7	9 - 10	December 6, 2010
Honeycomb	3.0 - 3.2.6	11 - 13	February 22, 2011
Ice Cream Sandwich	4.0 - 4.0.4	14 - 15	October 18, 2011
Jelly Bean	4.1 - 4.3.1	16 - 18	July 9, 2012
KitKat	4.4 - 4.4.4	19 - 20	October 31, 2013
Lollipop	5.0 - 5.1.1	21- 22	November 12, 2014
Marshmallow	6.0 - 6.0.1	23	October 5, 2015
Nougat	7.0	24	August 22, 2016
Nougat	7.1.0 - 7.1.2	25	October 4, 2016
Oreo	8.0	26	August 21, 2017
Oreo	8.1	27	December 5, 2017
Pie	9.0	28	August 6, 2018

Android 10	10.0	29	September 3, 2019
Android 11	11	30	September 8, 2020

Fig 1.1 Android Versions

1.3 Android Architecture

Android operating system is a stack of software components which is roughly divided into five sections and four main layers as shown below in the architecture diagram.

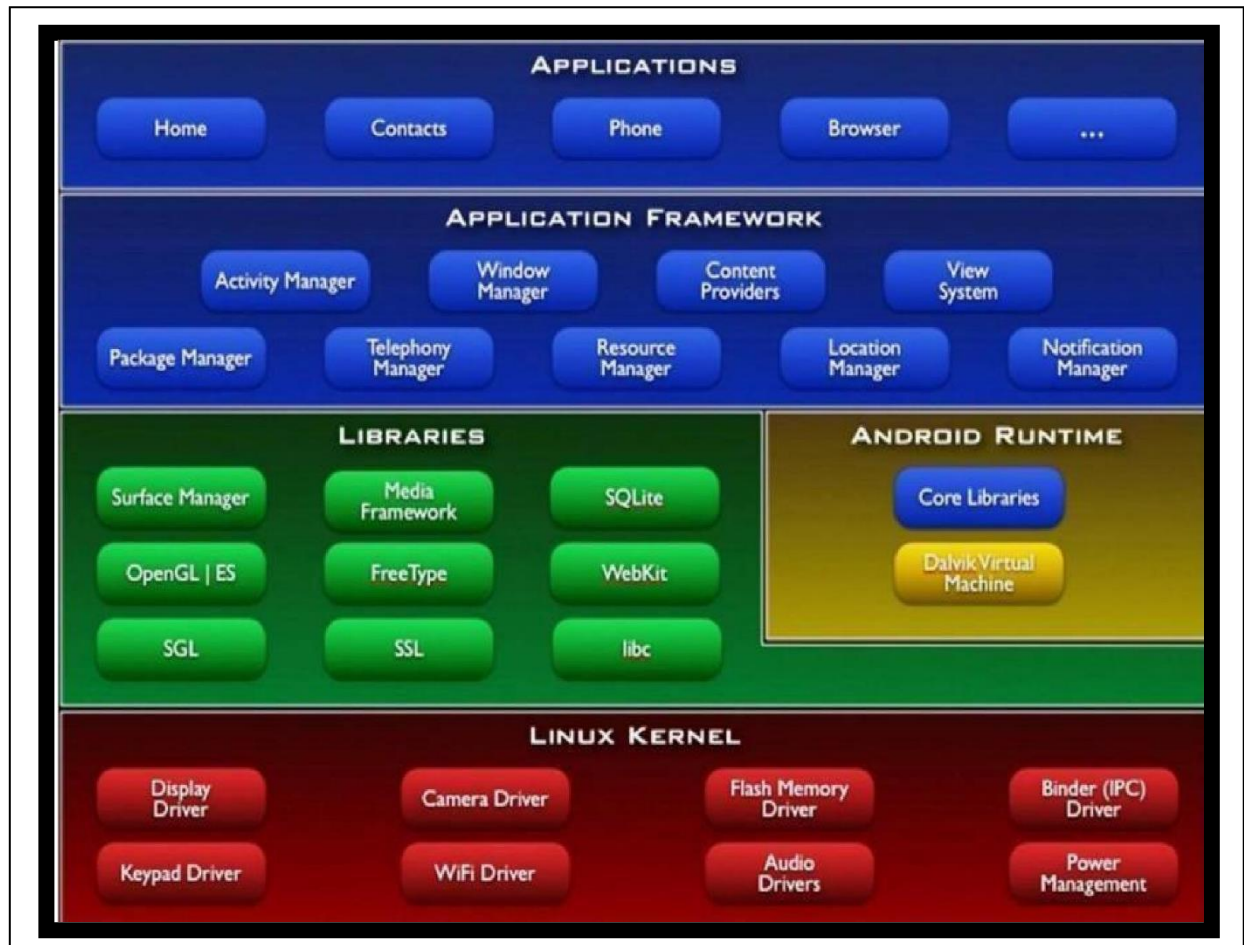


Figure 1.2 Android Architecture diagram

- ❖ **Linux Kernel:** This is the layer at the very bottom of the Android architecture. All other layers run on top of the Linux kernel and rely on this kernel to interact with the hardware. This layer contains all the essential hardware drivers which help to control and communicate with the hardware. It provides the basic functionality like Process Management, Memory Management and Device Management like Camera, Display, Flash etc.

- ❖ **Libraries:** This is a set of common functions of the application framework that enables the device to handle different types of data. Some of the most important set of libraries that are included are – Web kit which is the browser engine to display HTML, OpenGL used to render 2- D or 3-D graphics on to the screen, SQLite which is a useful repository for storing and sharing of application data.

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.
-
- ❖ **Android Runtime:** The Android runtime mainly consist of the Dalvik Virtual Machine (DVM). DVM is very much like the standard Java Virtual Machine (JVM) except that it is optimized for mobile devices that have low processing power and low memory. DVM generates a.dex file from the .class file at compile time and provides higher efficiency in low resources devices. Each application has its own process and an instance of DVM.

Android runtime also provides core libraries that enable the Android developers to create applications using the Java language.

- ❖ **Application Framework:** The Android runtime mainly consist of the Dalvik Virtual Machine (DVM). DVM is very much like the standard Java Virtual Machine (JVM) except that it is optimized for mobile devices that have low processing power and low memory. DVM generates a.dex file from the .class file at compile time and provides higher efficiency in low resources devices. Each application has its own process and an instance of DVM. Android runtime also provides core libraries that enable the Android developers to create applications using the Java language.
- ❖ **Applications:** This is the topmost layer in the architecture and the layer where the application that we develop fits in. This layer provides several pre-installed applications that are default for certain things like Contacts Books, Browser etc..

1.4 Android Studio Installation

Android Studio is the official integrated development environment (IDE) for Android application development. It is based on the IntelliJ IDEA, a Java integrated development environment for software, and incorporates its code editing and developer tools.

To support application development within the Android operating system, Android Studio uses a Gradle-based build system, emulator, code templates, and GitHub integration. Every project in Android Studio has one or more modalities with source code and resource files. These modalities include Android app modules, Library modules, and Google App Engine modules.

PROCEDURE TO BE FOLLOWED TO DOWNLOAD AND INSTALL ANDROID STUDIO:

STEP 1: Android Studio and the Software Development Kit can be downloaded directly from any web browser using the below link.

<https://developer.android.com/studio>

STEP 2: Android Studio is available for Mac, Windows, and Linux desktop platforms.

Windows

To install Android Studio on Windows, proceed as follows:

- i. If you downloaded an .exe file (recommended), double-click to launch it. If you downloaded a .zip file, unpack the ZIP, copy the android-studio folder into your Program Files folder, and then open the android-studio > bin folder and launch studio64.exe (for 64-bit machines) or studio.exe (for 32-bit machines).
- ii. Follow the setup wizard in Android Studio and install any SDK packages that it recommends.

Mac

To install Android Studio on your Mac, proceed as follows:

- i. Launch the Android Studio DMG file.
- ii. Drag and drop Android Studio into the Applications folder, then launch it.
- iii. Select if you want to import previous Android Studio settings, then press OK
- iv. The Android Studio Setup Wizard guides you through the rest of the setup, which includes downloading Android SDK components that are required for development.

Linux

To install Android Studio on Linux, proceed as follows:

- i. Unpack the .zip file you downloaded to an appropriate location for your applications, such as within /usr/local/ for your user profile, or /opt/ for shared users. If you're using a 64-bit version of Linux, make sure you first install the required libraries for 64-bit machines.
- ii. To launch Android Studio, open a terminal, navigate to the android-studio/bin/ directory and execute studio.sh.
- iii. Select whether you want to import previous Android Studio settings or not, then click OK.
- iv. The Android Studio Setup Wizard guides you through the rest of the setup, which includes downloading Android SDK components that are required for development.

Chapter 2

INTRODUCTION TO PROJECT

2.1 Overview of the Project:

The purpose of travel app gets users a direct answer to everything they need to know about planning a vacation – the way to reach there, flight/train ticket details, hotel booking, places of interest in the destination, etc. Travel applications tend to become one-stop solution for all the travel needs. Users do not have to head on to multiple browsers to get answers to their variety of queries, they can get them all in one place. One of the best parts of travel applications are the ease to make hassle-free payments. By giving the users a secure platform to pay online, travel mobile applications save them a lot of time. The travel application benefits that you read above are the doing of well-strategized features that ease the whole process while becoming the key reason behind the industry's growth.

2.2 Aim of the Project:

Travel mobile apps allow travelers to avoid extensive long-term planning which allows them to be as spontaneous—something that many travelers enjoy. When check-in is about three to four months away, about 6% of booking occurs via a smartphone. The numbers steadily grow as the check-in time approaches. This allows for informed decision-making on sustainable mobility by the traveler, by directly providing incentives .App are not only a source of inspiration but also. The app helps travelers find the cheapest hotels and rental cars.

Chapter 3

ABOUT SOFTWARE

Android Studio is the official Integrated Development Environment (IDE) for android application development. Android Studio provides more features that enhance our productivity while building Android apps.

Android Studio was announced on 16th May 2013 at the Google I/O conference as an official IDE for Android app development. It started its early access preview from version 0.1 in May 2013. The first stable built version was released in December 2014, starts from version 1.0. Since 7th May 2019, Kotlin is Google's preferred language for Android application development. Besides this, other programming languages such as Java are supported by Android Studio.

3.1 Features of Android Studio

- ▶ It has a flexible Gradle-based build system.
- ▶ It has a fast and feature-rich emulator for app testing.
- ▶ Android Studio has a consolidated environment where we can develop for all Android devices.
- ▶ Apply changes to the resource code of our running app without restarting the app.
- ▶ Android Studio provides extensive testing tools and frameworks.
- ▶ It supports Java, C++ and NDK.
- ▶ It provides build-in supports for Google Cloud Platform. It makes it easy to integrate Google Cloud Messaging and App Engine.

3.2 Android Studio Project Structure

The Android Studio project contains one or more modules with resource files and source code files. These include different types of modules-

- ▶ Android app modules
- ▶ Library modules
- ▶ Google App Engine modules

By default, Android Studio displays our project files in the Android project view, as shown in the below Figure 2.1. This view is formed by modules to provide quick access to our project's key source files

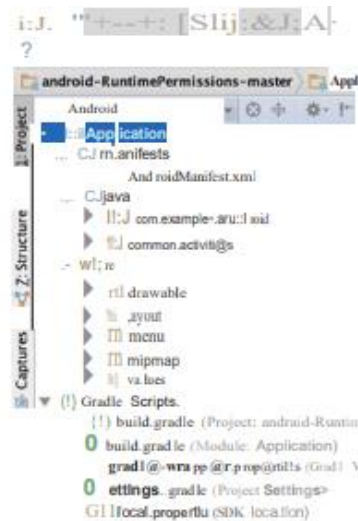


Figure 2.1 Android Project Structure

These build files are visible to the top-level under Gradle Scripts. And the app module contains the following folders:

- manifests: It contains the AndroidManifest.xml file.
- java: It contains the source code of Java files, including the junit test code.
- res: It contains all non-code resources, UI strings, XML layouts, and bitmap images.

We will see the actual file structure of the project by selecting the Project from the Project dropdown.

3.3 Android Studio User Interface

The Android Studio main window contains the several logical areas which are shown below:

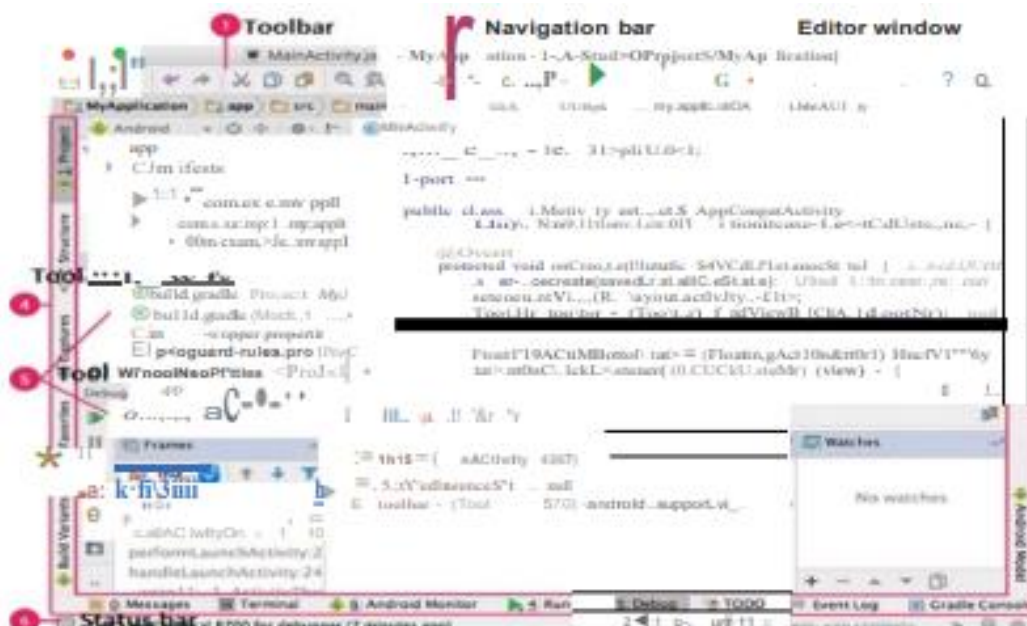


Figure 3.2 Android Studio Main Window

1. The toolbar provides us a wide range of actions, which includes running apps and launching Android tools.

11. The navigation bar helps in navigating our project and open files for editing. It gives a compact view

of structure visible in the Project window.

111. The editor window is a space where we can create and modify our code. On the basis of the current file

type, the editor can change. While viewing a layout file, the editor displays the Layout Editor.

1v. The tool window bar runs around the outside the IDE window and contains buttons that allow us to expand and collapse individual tool windows.

v. The tool windows provide us access specific tasks like search, project management, version control, and more. We can able expand and collapse them.

v1. The status bar displays the status of our project and IDE itself, as well as any messages or warnings.

3.4 Gradle Build System

Gradle build used as the foundation of the build system in Android Studio. It uses more Android-specific

capabilities provided by the Android plugin for Gradle. This build system runs independently from the command

line and integrated tool from the Android Studio menu. We can use build features for the following purpose:

- ▶ Configure, customize, and extend the build process.
- ▶ We can create multiple APKs from our app, with different features using the same project and modules.
- ▶ Reuse resource and code across source sets.

3.5 Install and Setup Android Studio and Java JDK

i. JDK installation

- Step 1: Download the JDK by using below link according to your Operating system.

<https://www.oracle.com/java/technologies/javase-jdk16-downloads.html>

- Step 2: Install the downloaded JDK in PC.
- Step 3: After Installing, Right Click on This PC or My Computer on Desktop
- Step 4: Click on Properties
- Step 5: Click on Advance System Settings
- Step 6: Click on Environment Variables in the bottom
- Step 7: Double Click on path in the second dialog box

- Step 8: Click on New to add path
- Step 9: Go to JDK path in C Drive
- Step 10: Copy and paste the JDK path in system settings
- Step 11: Go to JRE path in C drive
- Step 12: Copy and paste the JRE path in system settings
- Step 13: For verification, Go to Command Prompt and press javac command. If it is getting executed, then you have successfully installed the Java JDK.

ii. Android Studio Setup

- Step 1: Open Android Studio
- Step 2: Click on create new project
- Step 3: Select Empty Activity
- Step 4: You can name your application of your own choice in Name field and you can select your language either Java/Kotlin, and you can also select the Minimum SDK- The lower the version you select as a developer, your app will run approx. in all the existing devices. Then press finish to start programming.
- Step 5: It takes time to build gradle which depends on your Internet connectivity and system configuration. (Approx. 1-3 minutes)
- Step 6: After gradle building, you will get the Android Studio Main Window.
- Step 7: Run the program by clicking on Run command
- Step 8: Output will be showed in Virtual Emulator as your mobile phone

Chapter 4

SYSTEM DESIGN

4.1 System Requirements

4.1.1 Hardware Requirements:

- 64-bit processor.
- x86_64 CPU architecture; 2nd generation Intel Core or newer, or AMD CPU with support for a Windows Hypervisor.
- 8 GB RAM or more.
- 8 GB of available disk space minimum (IDE + Android SDK + Android Emulator)
- 1280 x 800 minimum screen resolution.

4.1.2 Software Requirements:

- Operating System: Windows 7 or above
- IDE: Android Studio version
- API Level: 19 or above

4.1.2.1 Technology Used

- Front End - XML
- Backend – JAVA

4.1.3 Platform

The platform organizations need to develop, deploy and manage mobile applications is made from many components, and tools allow a developer to write, test and deploy

Front-end development tools

Front-end development tools are focused on the user interface and user experience (UI/UX) and provide the following capabilities:

- UI design tools
- SDKs to access device features
- Cross-platform accommodations/support

Back-end servers

Back-end tools pick up where the front-end tools leave off, and provide a set of reusable

services that are centrally managed and controlled and provide the following capabilities:

- Integration with back-end systems
- User authentication/authorization
- Data services
- Reusable business logic

Security add-on layers

The security is more important within more enterprises, IT departments often need stopgap, tactical solutions that layer on top of existing apps, phones, and platform component.

- App wrapping for security.
- Data encryption
- Client actions
- Reporting and statistics

Chapter 5

DESIGN AND IMPLEMENTATION

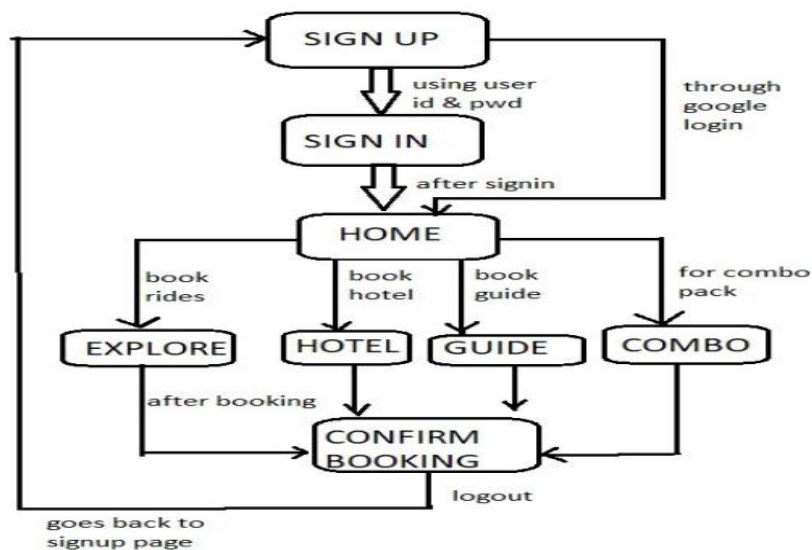
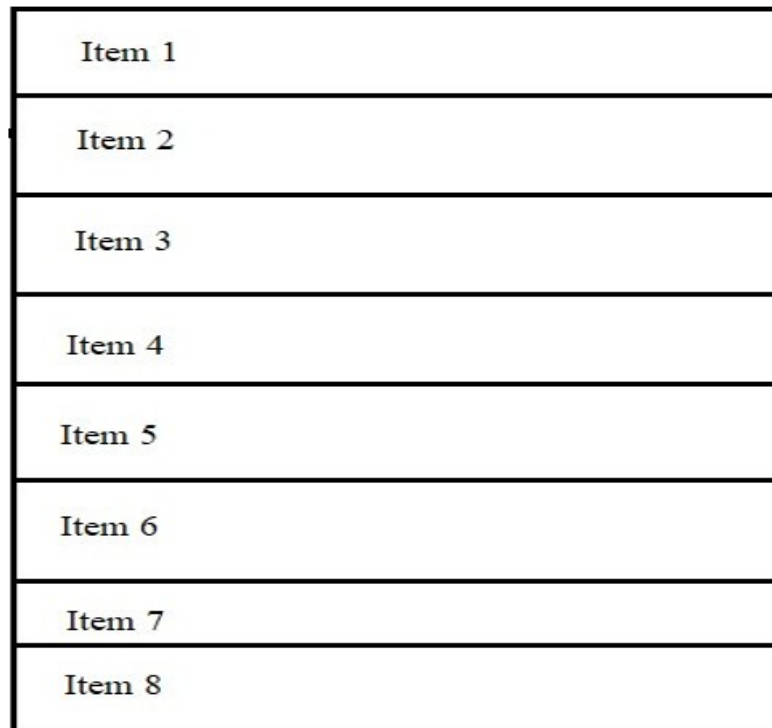


Fig 5.1:Flowchart of application

Once the app is installed in the android mobile phone, the user will be able to open the app and the sign in page appears on the screen as the app starts. A user can sign up through username and password and it bundle all the details take to login page and from login page user and their same details and login to their account. .User can also login through real google API .As soon as user reach home page of an app they can start **planning for their trip**.They can book rides for their trip,or they book hotels and guide separately,if they want super saving offer they book though various combo pack and get exclusive offers.Once thi is done they can confirm their booking and history is saved in the history page user can logout or stay sign in.

5.1 Design using XML

5.1.1 Recycler view design



Item 1
Item 2
Item 3
Item 4
Item 5
Item 6
Item 7
Item 8

Fig 5.2 Recycler view design

RecyclerView is used to create vertical scrolling grid. In RecyclerView item can be added using custom adapter. The main function of the adapter in RecyclerView is to fetch data from a database or array and insert each piece of data in an appropriate item that will be displayed in RecyclerView. Using RecyclerView one can save the memory in the mobile.

Activity_Explore.XML:

```
<?xml version="1.0" encoding="utf-8"?>

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"

    android:layout_width="match_parent"

    android:layout_height="fill_parent"

    android:orientation="vertical"
```

```
android:weightSum="2.0">
```

```
<android.recyclerview.widget.RecyclerView  
android:layout_width="match_parent"  
android:layout_height="fill_parent"  
android:orientation="vertical"  
android:id="@id/recyclerview"  
app:layout_constraintBottom_toBottomof="parent"  
app:layout_constraintEnd_toEndof="parent"/
```

5.1.2 Strings

A string resource provides text strings for the application with optional list styling and formatting. String array is a XML resource that provides an array of strings.

Strings.xml:

```
<resources>  
  
<string name="app_name">TRAVEL</string>  
  
<string array name="names">  
  
<Item>AGRA</Item>  
  
<Item>GOA</Item>  
  
<Item>PONDICHERRY</Item>  
  
<Item>KOLKATA</Item>  
  
<Item>SIKKIM</Item>  
  
<Item>TAMIL NADU</Item>  
  
</string array>  
  
</resources>
```


5.1.3 Manifest

Android Manifest is an XML file that is the root of the project source set.

AndroidManifest.xml:

```
<?xml version="1.0" encoding="utf-8"?>

<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:dist="http://schemas.android.com/apk/distribution"
    package="com.example.feed">

    <dist:module dist:instant="true" />

    <uses-permission android:name="android.permission.INTERNET" />

    <uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION"
    />

    <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />

    <uses-permission android:name="com.sec.android.provider.badge.permission.WRITE" />

    <uses-permission android:name="com.sec.android.provider.badge.permission.WRITE" />

    <application android:allowBackup="true"
        android:icon="@drawable/logo"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportRtl="true"
        android:theme="@style/AppTheme">

        <activity android:name=".DisplayActivity"></activity>

        <activity android:name=".RecieveActivity" />

        <activity android:name=".DonateActivity" />

        <activity
            android:name=".RegisterActivity"
```

```
android:label="@string/title_activity_register"

android:theme="@style/AppTheme.NoActionBar" />

<activity android:name=".SecondActivity" />

<activity android:name=".FirstActivity" />

<activity android:name=".MainActivity">

<intent-filter>

<action android:name="android.intent.action.MAIN" />

<category android:name="android.intent.category.LAUNCHER" />

</intent-filter>

</activity>

</application>

</manifest>
```

5.2 Main Activity

- An **intent** is to perform an action on the screen. It is mostly used to start activity, send broadcast receiver, start services and send message between two activities. There are two intents available in android as Implicit Intents and Explicit Intents. Here is a sample example to start new activity with old activity.
- Android **Bundles** are generally used for passing data from one activity to another. Basically here concept of key-value pair is used where the data that one wants to pass is the value of the map, which can be later retrieved by using the key. Bundles are used with intent and values are sent and retrieved in the same fashion, as it is done in the case of Intent. It depends on the user what type of values the user wants to pass, but bundles can hold all types of values (int, String, boolean, char) and pass them to the new activity.
- **GoogleSignInOptions** configure the sign in request the user id email and other details. Id and basic profile are included in DEFAULT_SIGN_IN.
- **GoogleSignInClient** check the existing google sign in account if already sign in the google sign in account will be non null.
-

CODE SNIPPET

```
package com.example.travel;

import androidx.appcompat.app.AppCompatActivity;import
android.content.Intent;
import android.net.Uri;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;
import com.google.android.gms.auth.api.signin.GoogleSignIn;

import com.google.android.gms.auth.api.signin.GoogleSignInAccount;import
com.google.android.gms.auth.api.signin.GoogleSignInClient; import
com.google.android.gms.auth.api.signin.GoogleSignInOptions;

import com.google.android.gms.common.SignInButton; import
com.google.android.gms.common.api.ApiException;import
com.google.android.gms.tasks.Task;
import java.util.regex.Matcher;import
java.util.regex.Pattern;

public class MainActivity extends AppCompatActivity implements View.OnClickListener { private
static int RC_SIGN_IN=100;
GoogleSignInClient mGoogleSignInClient;
EditText usersignup,pwdsignup;
Button btnsignup;
String Rexp="^(?=.*[A-Z])(?=.*[a-z])(?=.*\\d)(?=.*[@#$!])[A-Za-z\\d@#$!]{8,}";
@Override
protected void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.activity_main);
usersignup = (EditText) findViewById(R.id.signupuser);
pwdsignup = (EditText) findViewById(R.id.signuppwd);
btnsignup = (Button) findViewById(R.id.loginbtn);
btnsignup.setOnClickListener(this);
```

```
// Configure sign-in to request the user's ID, email address, and basic
```

```
// profile. ID and basic profile are included in DEFAULT_SIGN_IN.
```

```
GoogleSignInOptions gso = new
```

```
GoogleSignInOptions.Builder(GoogleSignInOptions.DEFAULT_SIGN_IN)
```

```
.requestEmail()
```

```
.build();
```

```
// Build a GoogleSignInClient with the options specified by gso.
```

```
mGoogleSignInClient = GoogleSignIn.getClient(this, gso);
```

```
// Check for existing Google Sign In account, if the user is already signed in
```

```
// the GoogleSignInAccount will be non-null.
```

```
GoogleSignInAccount account = GoogleSignIn.getLastSignedInAccount(this);
```

```
// Set the dimensions of the sign-in button.
```

```
SignInButton = findViewById(R.id.sign_in_button);
```

```
signInButton.setSize(SignInButton.SIZE_STANDARD);
```

```
findViewById(R.id.sign_in_button).setOnClickListener(new View.OnClickListener() {
```

```
@Override
```

```
public void onClick(View view) {signIn();
```

```
}
```

```
});
```

```
}
```

```
private void signIn() {
```

```
Intent signInIntent = mGoogleSignInClient.getSignInIntent();
```

```
startActivityForResult(signInIntent, RC_SIGN_IN);
```

```
}
```

```
@Override
```

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

```
super.onActivityResult(requestCode, resultCode, data);
```

```
// Result returned from launching the Intent from GoogleSignInClient.getSignInIntent(...);if
```

```
(requestCode == RC_SIGN_IN) {
```

```
// The Task returned from this call is always completed, no need to attach
```

```
// a listener.
```

```
Task<GoogleSignInAccount> task = GoogleSignIn.getSignedInAccountFromIntent(data);
```

```
handleSignInResult(task);
```

```
}
```

```
}  
private void handleSignInResult(Task<GoogleSignInAccount> completedTask) { try {  
    GoogleSignInAccount account = completedTask.getResult(ApiException.class);  
    GoogleSignInAccount acct = GoogleSignIn.getLastSignedInAccount(this);  
    if (acct != null) {  
        String personName = acct.getDisplayName(); String  
        personGivenName = acct.getGivenName(); String  
        personFamilyName = acct.getFamilyName();String  
        personEmail = acct.getEmail();  
        String personId = acct.getId();  
        Uri personPhoto = acct.getPhotoUrl();  
        Intent i=new Intent(MainActivity.this,home.class);  
        i.putExtra("acctname",personName); startActivity(i);  
    }  
    // Signed in successfully, show authenticated UI.  
    // Intent i=new Intent(MainActivity.this,home.class);  
    // i.putExtra("acctname",pm);  
    //startActivity(i);  
    } catch (ApiException e) {  
        // The ApiException status code indicates the detailed failure reason.  
        // Please refer to the GoogleSignInStatusCodes class reference for more information.  
        Log.d("message",e.toString());  
    }  
    }  
    @Override  
    public void onClick(View view) {  
        String Username=usersignup.getText().toString(); String  
        Password=pwdsignup.getText().toString();  
        if(validatepassword>Password)){  
            Toast.makeText(getBaseContext(),"VALID PASSWORD",Toast.LENGTH_LONG).show();Bundle  
            bundle=new Bundle();  
            bundle.putString("user",Username);  
            bundle.putString("pass",Password); Intent =new  
            Intent(this,login.class);  
            intent.putExtra(("data"),bundle);
```

```
startActivity(intent);
}else
{
Toast.makeText(getBaseContext(),"INVALID
PASSWORD",Toast.LENGTH_LONG).show();
}
}
private boolean validatepassword(String password) { Pattern
=Pattern.compile(Rexp);
Matcher matcher=pattern.matcher(password);return
matcher.matches();
}}
```

5.3 Explore

[RecyclerView](#) is the [ViewGroup](#) that contains the views corresponding to your data. It's a view itself, so you add RecyclerView into your layout the way you would add any other UI element.

- Each individual element in the list is defined by a *view holder* object. When the view holder is created, it doesn't have any data associated with it. After the view holder is created, the RecyclerView *binds* it to its data. You define the view holder by extending [RecyclerView.ViewHolder](#).
- The RecyclerView requests those views, and binds the views to their data, by calling methods in the *adapter*. You define the adapter by extending [RecyclerView.Adapter](#).
- The *layout manager* arranges the individual elements in your list. You can use one of the layout managers provided by the RecyclerView library, or you can define your own. Layout managers are all based on the library's [LayoutManager](#) abstract class

The type of custom adapter you create depends on the format of the incoming data and the technology you use in the adapter code to convert the incoming data to Oracle Stream Explorer events. Custom adapters typically do the following:

- Use APIs from a data vendor, such as Reuters, Wombat, or Bloomberg.
 - Use messaging systems, such as TIBCO Rendezvous.
 - Establish a socket connection to the customer's own data protocol.
 - Authenticate themselves as needed with the input or output external component.
-

- Receive raw event data from an external component, convert the data into events, and send the events to the next application node.
- Receive events from within the event processing network and convert the event data to a form that is recognized by the target external component.

CODE SNIPPET:

```
package com.example.travel;

import androidx.appcompat.app.AppCompatActivity;

import androidx.recyclerview.widget.DividerItemDecoration; import
androidx.recyclerview.widget.LinearLayoutManager; import
androidx.recyclerview.widget.RecyclerView;

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

import android.view.View;

import android.widget.ImageView;

public class Explore extends AppCompatActivity {

    RecyclerView;

    ImageView HISTORY, LOGOUT, HOME;

    Details o1=new Details(R.drawable.swift, "Suzuki, Swift", "Mini|4Seater|AC", "Spacious eco-
friendly car", "12Km included", "Rs 15.0/Km after 12Km", "Electric Car", "Free
Cancellation:Till 1hr untill departure", "RS:299/day");

    Details o2=new Details(R.drawable.brezza, "Maruthi, Brezza", "Sedan|4Seater|AC", "Spacious eco-
friendly car", "12Km included", "Rs 15.0/Km after 12Km", "Disel Car", "Free Cancellation:Till
1hr untill departure", "RS:299/day");

    Details o3=new Details(R.drawable.inova, "Toyota, Innova", "SUV|7Seater|AC", "Spacious eco-
friendly car", "12Km included", "Rs 17.0/Km after 12Km", "Disel Car", "Free Cancellation:Till 1hr
untill departure", "RS:350/day");
```

Details o4=new

Details(R.drawable.fortuner,"Mahindra,Fortuner","SUV|7Seater|AC","Spacious eco-friendly car","12Km included","Rs 17.0/Km after 12Km","Disel Car","Free Cancellation:Till 1hr untill departure","RS:350/day");

Details o5=new

Details(R.drawable.scorpio,"Mahindra,Scorpio","SUV|8Seater|AC","Spacious eco-friendly car","12Km included","Rs 18.0/Km after 12Km","Disel Car","Free Cancellation:Till 1hr untill departure","RS:399/day");

Details o6=new Details(R.drawable.xuv,"Mahindra,XUV500","SUV|6Seater|AC","Spacious eco-friendlycar","12Km included","Rs 16.0/Km after 12Km","Disel Car","Free Cancellation:Till 1hruntill departure","RS:350/day");

Details o7=new Details(R.drawable.bolero,"Mahindra,Bolero","SUV|8Seater|AC","Spacious eco-friendlycar","12Km included","Rs 18.0/Km after 12Km","Disel Car","Free Cancellation:Till 1hruntill departure","RS:499/day");

Details o8=new Details(R.drawable.traveller,"Traveller","BUS|17Seater|AC","Spaciouseco-friendly car","12Km included","Rs 20.0/Km after 12Km","Petrol Car","Free Cancellation:Till 1hr untill departure","RS:799/day");

Details[] details={o1,o2,o3,o4,o5,o6,o7,o8};

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState); setContentView(R.layout.activity_explore);

recyclerView=findViewById(R.id.recyclerView);

recyclerView.setLayoutManager(new LinearLayoutManager(this));

CustomAdapter c=new CustomAdapter(details,getContext());

recyclerView.setAdapter(c);

recyclerView.addItemDecoration(new
DividerItemDecoration(this,DividerItemDecoration.VERTICAL));

LOGOUT = (ImageView) findViewById(R.id.man1);

LOGOUT.setOnClickListener(new View.OnClickListener() {

@Override


```
public void onClick(View view) {
```

```
    Intent intent5 = new Intent(Explore.this, MainActivity.class);
```

```
    startActivity(intent5);
```

```
}
```

```
});
```

```
HISTORY = (ImageView) findViewById(R.id.hist1);
```

```
HISTORY.setOnClickListener(new View.OnClickListener() {
```

```
    @Override
```

```
    public void onClick(View view) {
```

```
        Intent intent5 = new Intent(Explore.this, History.class);
```

```
        startActivity(intent5);
```

```
    }
```

```
});
```

```
HOME= (ImageView) findViewById(R.id.hm1);
```

```
HOME.setOnClickListener(new View.OnClickListener() {
```

```
    @Override
```

```
    public void onClick(View view) {
```

```
        Intent intent5 = new Intent(Explore.this, home.class);
```

```
        startActivity(intent5);} }); }
```

Chapter 6

RESULTS

All the Activities provided in the application and its operations have been presented in as snapshots. A detailed view of all the snapshots of the application is given in this section.

6.1 SIGN UP

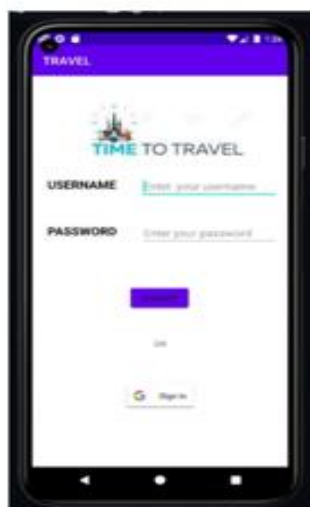


FIG 6.1:SIGN UP PAGE

Fig 6.1 shows the signup page for user using login id and password or through google sign in.

6.2 SIGN IN



FIG 6.2:SIGN IN PAGE

Fig 6.2 shows the signin page for user using same login id and password through which they signup .

6.3 HOME



FIG 6.3:HOME PAGE

Fig 6.3 shows the home page consisting of most visited place and option for book rental hotels etc.

6.4 TIMING AND DESTINATION



FIG 6.4:TIMING AND DESTINATION PAGE

Fig 6.4 shows the detination and timing page which ask the user to enter the destination and date and show their result as per availability.

6.5 EXPLORE



FIG 6.5:EXPLORE PAGE

Fig 6.5 shows the explore page so tha user can book their required retanls..

6.6 HOTEL



FIG 6.6:HOTEL PAGE

Fig 6.6 shows the hotel page so tha user can book their required hotels..

6.7 GUIDE



FIG 6.7:GUIDE PAGE

Fig 6.7 shows the guide page so tha user can book their required guide.

6.8 COMBO



FIG 6.8:COMBO PAGE

Fig 6.8 shows the combo page so tha user can book their required combo offer..

6.9 DETAILS



FIG 6.9:DETAILS PAGE

Fig 6.9 shows the details page of the rentals or hotels which they are booking.

6.10 BOOKING CONFIRM



FIG 6.10:BOOKING CONFIRM PAGE

Fig 6.10 shows the bookingconfirm page once the user confirm their booking.

Chapter 7

CONCLUSION AND FUTURE ENHANCEMENT

7.1 Conclusion

The proposed application provides overall best experience throughout the journey. It provides user all services in one app like booking rentals ,hotels and even travel guide. Timeto travel app also available for 24 hrs support to each and every user. App provide super savings combos for the users to make their journey more pleasant and safe.

7.2 Future Enhancement

- App tells about weather information like temperature, humidity etc.
- The app will be able to scan the words spoken or the text written and translate it.
- Will add payment through app i.e payment card with offers.

REFERENCES

- 1) www.google.com
- 2) www.youtube.com
- 3) www.github.com
- 4) www.w3schools.com
- 5) www.tutorialspoint.com
- 6) <https://developer.android.com>
- 7) <https://ijcsmc.com/docs/papers/October2019/V8I10201907.pdf>