**MAJOR PROJECT REPORT ON**

Tourism Recommendation Application

*Submitted in partial fulfilment of the requirements for the award of the degree of*

**BACHELOR OF COMPUTER APPLICATIONS**

****

**Batch: 2021 - 24**

***Under the Guidance of Submitted By***

Dr. Ruchi Sawhney Ankit Alex Minz

(Assistant Professor) 04328402021

**Bosco Technical Training Society**

**Don Bosco Technical School, Okhla Road, New Delhi-110025**

**9643868820, 8527787221, 011-41033889**

**Affiliated to**

****

**Guru Gobind Singh Indraprastha University**

### Sector - 16C, Dwarka, New Delhi - 110078 (India)

### Phone: +91-11-25302170, Fax: +91-11- 25302111

### E-Mail Id: ggsipu.pr@rediffmail.com, pro@ipu.ac.in

## **Acknowledgement**

The note starts with thanks to Almighty who actually created this piece of work and helped us when things were not easy for us.

I am very grateful and indebted to my Faculty/Guide Dr. Ruchi Sawhney (Assistant Professor) who immensely helped and rendered her valuable advice, precious time, knowledge and relevant information regarding the collection of material. She has been a major source of inspiration throughout the project as she not only guided me throughout this Project Report “Tourism Recommendation Application” but also encouraged me to solve problems that arose during this report.

Her guidance and suggestions about this Project report have really enlightened me. It has been a great help to support to have him around.

And finally, I would like to mention appreciation to our parents and friends who have been instrumental throughout this period by providing unrelenting encouragement.

Signature

Ankit Alex Minz

04328402021

**SELF-CERTIFICATE**

This is to certify that the dissertation/project report entitled “Tourism Recommendation Application” done by me is an authentic work carried out for the partial fulfilment of the requirements for the award of the degree of Bachelor of Computer Applications under the guidance of Dr. Ruchi Sawhney (Assistant Professor). The matter embodied in this project work has not been submitted earlier for award of any degree or diploma to the best of my knowledge and belief.

Signature of the student

Ankit Alex Minz

04328402021

**CERTIFICATE**

This is to certify that this project entitled **“Tourism Recommendation Application”** submitted in partial fulfilment of the degree of Bachelor of Computer Applications to **Dr. Ruchi Sawhney** (Assistant Professor) through Guru Gobind Singh Indraprastha University, Delhi done by **Mr. Ankit Alex Minz, Roll No. 04328402021** is an authentic work carried out by him at Bosco Technical Training Society, Don Bosco Technical School under my guidance. The matter embodied in this project work has not been submitted earlier for award of any degree to the best of my knowledge and belief.

Signature of the student Signature of the Guide

Synopsis

# **1. Problem Statement:**

## **1.1 Problem:**

Traditional tourism apps will just sell you a plan regarding the place you are visiting, your time is not your own during such a vacation as you will have set time you can visit certain places in a bid to visit other places that you will be guided to by the tourism agency guide. The challenge is to develop an app that will help you travel on your own terms, by assisting you in related things like what landmarks to visit, telling the general pricing of various amenities in the area etc.

## **1.2 Background:**

A lot of times for their forth coming vacations some people might rather prefer traveling on their own leisure to the places they want to tour and as such might need to conduct detailed research regarding the area in order to guarantee a safe and satisfying journey with little amount conventional roadblocks. However, such information is spread out all over the internet not mention or requires communication with locals. Adding the problem of live information or regularly fluctuating information on top of that, makes it a need that such a place exists to provide people with such information.

## **1.3 Relevance:**

The project displays its relevance in the fact that it will keep users up to date regarding various factors of their self-traveling journey with no assistance from tourism agencies for example., pricing of various amenities like hotel, travel etc., traffic in the places they are planning to visit in their travels etc.

# **2.Objective and Scope:**

## **2.1. Objective**

The objective of this project is to create a Tourism Recommendation App that will take various information from the user regarding the vacation and provide a customized plan for them which referring to which they may plan the journey for their travel. The recommended plan will include various information like pricing of the various main parts of the plan such hotel pricing, travel pricing etc., travel routes through google maps, etc. There will also be live reporting about some things like current traffic of routes, pricing etc.

## **2.2 Scope**

The scope of the project encompasses the development and implementation of recommendation algorithms that will tailor the plans and services provided by the app according to the details and interests provided by the users. Such tasks will include pricing, travel, etc related plan recommendations as well as details regarding hotels, eateries and other necessities. The Application will also include services to assist users during their execution of their selected traveling plan such live location of necessities nearby user.

# **3. Methodology**

* **Data Collection & Creation:** Procure data regarding tourism plans, pricing, places, cities etc., from the internet as well as creating it.
* **Machine Learning Model:** Develop a machine learning model which provides the best suiting recommendation to the user, incorporating various inputs developed during data collection and creation.
* **Recommendation Generation:** Utilize the model’s filtering algorithm to generate individualized plan recommendation for each user, incorporating the various details they provide such as their budget, place of travel, number of days, possible route etc.
* **User Interface**: Make a user-friendly and interactive interface that is intuitive to use and attractive to look at, it should be able to incorporate all planned functions in an organized and smooth manner.
* **Functionality:** The user interface will be implemented with required functionality after being planned and created.

# **4. Hardware and Software**

## **4.1 Hardware requirements:**

Computer: windows 10, 8GB Ram, 4GB space

Android device: 8GB Ram, Version: 13

## **4.2 Software Requirement:**

Android Studio

Java

Android Virtual Device or Emulator

XML for UI

Database (SQLite or Firebase)

Google Maps API

Version Control System (Git)

# **5. Resources and Limitations**

## **5.1 Resources:**

* Programming languages, libraries etc., suitable for making the project.
* Access to diverse data that would either be existing data extracted from the internet or created data for dummy purposes.
* Availability of hardware and software infrastructure.
* Knowledge regarding different subjects in tourism.

## **5.2 Limitations:**

* Recommendations are limited by accuracy & availability of data.
* Recommendations might become out of date if political and economic changes happen between time of booking and time of departure.
* May necessitate regular updates to evolve various components to meets changing standards.

# **6. Testing Methodology**

The testing methodology employed for this project is Manual Testing, utilizing White-Box Testing techniques.

**Testing Process:** The testing process encompasses the following steps:

* **Requirements Analysis:** Review and analysis of project documentation to identify testing objectives and requirements.
* **Test Case Developmen**t: Creation of test cases tailored to the identified objectives and requirements.
* **Manual Testing:** Execution of test cases through manual processes.
* **Bug Identification:** The process of identifying and documenting defects or issues encountered during manual testing.

**Completion Criteria:** The testing process is considered complete when all identified test cases have been executed, and no further defects or bugs are found.

# **7. Conclusion**

The “Tourism Recommendation Application” project endeavors to address the pressing issue of solitary unguided travel that may result from not choosing a tourism company to guide people on their travels. The Application provide individualized recommendations and active/live guidance by leveraging various factors and stored information from the database. The project holds significance towards the tourism industry in helping people by providing a gathered and centralized access for data and guidance.

Main Report

**Contents**

[**1. Objective and Scope 15**](#_Toc167151440)

[**1.1. Objective 15**](#_Toc167151441)

[**1.2. Scope 15**](#_Toc167151442)

[**1.3. Technical Components 16**](#_Toc167151443)

[**2. Definition of Problem 25**](#_Toc167151444)

[**2.1. Problem: 25**](#_Toc167151445)

[**2.2. Background: 25**](#_Toc167151446)

[**2.3. Relevance: 25**](#_Toc167151447)

[**3. System Requirements 26**](#_Toc167151448)

[**3.1. Hardware Requirements: 26**](#_Toc167151449)

[**3.2. Software Requirements: 26**](#_Toc167151450)

[**4. System Analysis & Design 27**](#_Toc167151451)

[**4.1. Iterative model 27**](#_Toc167151452)

[**4.2. Use Case Diagram: 29**](#_Toc167151453)

[**5. System Planning (PERT CHART) 35**](#_Toc167151454)

[**6. Methodology 38**](#_Toc167151455)

[**7. Detail life cycle of the project 39**](#_Toc167151456)

[**7.1. ERD (Entity Relationship Diagram) 39**](#_Toc167151457)

[**7.2. DFD (Data Flow Diagram) 43**](#_Toc167151458)

[**7.3. Input and output screen design 46**](#_Toc167151459)

[**7.4. Process Involved 49**](#_Toc167151460)

[**7.5. Methodology used in testing: 53**](#_Toc167151461)

[**7.6. Test Report: 54**](#_Toc167151462)

[**8. Code and Screenshots 61**](#_Toc167151463)

[**9. Conclusion and Future Scope 139**](#_Toc167151464)

[**10. References 140**](#_Toc167151465)

**List of Figures**

[**Figure 1:Iterative model 27**](#_Toc167173889)

[**Figure 2:Use case Actor 29**](#_Toc167173890)

[**Figure 3: Use case representation 30**](#_Toc167173891)

[**Figure 4: System boundary representation 30**](#_Toc167173892)

[**Figure 5: Association Relationship 31**](#_Toc167173893)

[**Figure 6: Include relationship 32**](#_Toc167173894)

[**Figure 7:Extend Relationship 33**](#_Toc167173895)

[**Figure 8: Generalization 33**](#_Toc167173896)

[**Figure 9 Use Case Diagram 34**](#_Toc167173897)

[**Figure 10 Pert Chart 37**](#_Toc167173898)

[**Figure 11 ER Components chart 39**](#_Toc167173899)

[**Figure 12 ER Diagram 43**](#_Toc167173900)

[**Figure 13: Level 0 DFD 46**](#_Toc167173901)

[**Figure 14:Level 1 DFD 46**](#_Toc167173902)

[**Figure 15: XML layout in Android Studio 49**](#_Toc167173903)

[**Figure 16: Activity Diagram Notations 50**](#_Toc167173904)

[**Figure 17: Activity Diagram 53**](#_Toc167173905)

[**Figure 18: Test Report 1 55**](#_Toc167173906)

[**Figure 19: Test Report 2 56**](#_Toc167173907)

[**Figure 20: Test Report 3 56**](#_Toc167173908)

[**Figure 21: Test Report 4 57**](#_Toc167173909)

[**Figure 22: Test Report 5 57**](#_Toc167173910)

[**Figure 23: Test Report 6 58**](#_Toc167173911)

[**Figure 24: Test Report 7 58**](#_Toc167173912)

[**Figure 25: Test Report 8 59**](#_Toc167173913)

[**Figure 26: Test Report 9 59**](#_Toc167173914)

[**Figure 27 Test Report 10 60**](#_Toc167173915)

[**Figure 28: Test Report 11 61**](#_Toc167173916)

[**Figure 29: Test Report 12 62**](#_Toc167173917)

[**Figure 30: Output 1 65**](#_Toc167173918)

[**Figure 31: Output 2 70**](#_Toc167173919)

[**Figure 32: Output 3 76**](#_Toc167173920)

[**Figure 33: Output 4 87**](#_Toc167173921)

[**Figure 34: Output 5 99**](#_Toc167173922)

[**Figure 35: Output 6 114**](#_Toc167173923)

[**Figure 36: Output 7 125**](#_Toc167173924)

[**Figure 37: Output 8 134**](#_Toc167173925)

[**Figure 38: Output 9 142**](#_Toc167173926)

# **Chapter 1: Objective and Scope**

The Tourism Recommendation System is an android application project created using the Java programming language. This project was created in light of the understanding that traditional methods or tourism through companies leave people with a lack of time to personally enjoy the places they tour and may force them to move in a strict timetable. Thus, for people wanting to tour a place at their own expense an application that will provide them with the necessary information for their travels was necessary.

## **1.1 Objective**

The objective of this project is to create a Tourism Recommendation App that will take various information from the user regarding the vacation and provide a customized plan for them which referring to which they may plan the journey for their travel. The recommended plan will include various information like pricing of the various main parts of the plan such hotel pricing, travel pricing etc., travel routes through google maps, etc. There will also be live reporting about some things like current traffic of routes, pricing etc.

## **1.2 Scope**

The scope of the project encompasses the development and implementation of recommendation algorithms that will tailor the plans and services provided by the app according to the details and interests provided by the users. Such tasks will include pricing, travel, etc related plan recommendations as well as details regarding hotels, eateries and other necessities. The Application will also include services to assist users during their execution of their selected traveling plan such live location of necessities nearby user.

## **1.3 Technical Components**

**Java**

Java is a class-based, object-oriented programming language that is designed to have as few implementation dependencies as possible. It is intended to let application developers write once, and run anywhere (WORA), meaning that compiled Java code can run on all platforms that support Java without the need for recompilation. Java was first released in 1995 and is widely used for developing applications for desktop, web, and mobile devices. Java is known for its simplicity, robustness, and security features, making it a popular choice for enterprise-level applications.

JAVA was developed by James Gosling at Sun Microsystems Inc in the year 1995 and later acquired by Oracle Corporation.

**Features:**

**1. Platform Independent:** Compiler converts source code to bytecode and then the JVM executes the bytecode generated by the compiler. This bytecode can run on any platform be it Windows, Linux, or macOS which means if we compile a program on Windows, then we can run it on Linux and vice versa. Each operating system has a different JVM, but the output produced by all the OS is the same after the execution of the bytecode. That is why we call java a platform-independent language.

**2. Object-Oriented Programming Language:** Organizing the program in the terms of a collection of objects is a way of object-oriented programming, each of which represents an instance of the class.

The four main concepts of Object-Oriented programming are:

* Abstraction
* Encapsulation
* Inheritance
* Polymorphism

**3. Simple:** Java is one of the simple languages as it does not have complex features like pointers, operator overloading, multiple inheritances, and Explicit memory allocation.

**4. Robust:** Java language is robust which means reliable. It is developed in such a way that it puts a lot of effort into checking errors as early as possible, that is why the java compiler is able to detect even those errors that are not easy to detect by another programming language. The main features of java that make it robust are garbage collection, Exception Handling, and memory allocation.

**5. Secure:** In java, we don’t have pointers, so we cannot access out-of-bound arrays i.e it shows ArrayIndexOutOfBound Exception if we try to do so. That’s why several security flaws like stack corruption or buffer overflow are impossible to exploit in Java. Also, java programs run in an environment that is independent of the os(operating system) environment which makes java programs more secure.

**XML**

Extensible Markup Language (XML) is a markup language that defines a set of rules for encoding documents in a format that is both human-readable and machine-readable. The design goals of XML focus on simplicity, generality, and usability across the Internet. It is a textual data format with strong support via Unicode for different human languages. Although the design of XML focuses on documents, the language is widely used for the representation of arbitrary data structures such as those used in web services.

**Use of XML in Android Studio:**

1. **Layout Definition:** XML is primarily used in Android Studio to define the layout of user interfaces. Developers use XML files to specify the arrangement and appearance of UI components such as buttons, text fields, images, and more. This separation of layout from code promotes cleaner and more manageable codebases.
2. **Resource Management:** XML is extensively utilized for managing app resources such as strings, colors, dimensions, styles, and more. By defining these resources in XML files, developers can easily reference and reuse them throughout the app, ensuring consistency and facilitating localization.
3. **UI Customization**: Android XML files support the use of styles and themes to customize the appearance of UI elements. Developers can create reusable styles in XML files to apply consistent visual attributes across multiple components, thereby maintaining a cohesive design language throughout the app.
4. **Data Exchange:** XML is often employed for data exchange between different components of an Android application.
5. **Configuration Settings:** Android applications often utilize XML files to store configuration settings, such as permissions, metadata, and manifest information. The AndroidManifest.xml file, for example, contains essential metadata about the application, including its package name, permissions, activities, services, and more.

**Android Studio**

Android Studio provides many excellent features that enhance productivity when building Android apps, such as a blended environment where one can develop for all Android devices, apply Changes to push code and resource changes to the running app without restarting the app, a flexible Gradle-based build system, a fast and feature-rich emulator, GitHub and Code template integration to assist you to develop common app features and import sample code, extensive testing tools and frameworks, C++ and NDK support, and many more.

**Features of Android Studio:**

* **Compose Design**: Android Studio supports Jetpack Compose, a modern UI toolkit for building native Android apps. With Compose, developers can create dynamic and interactive user interfaces declaratively using Kotlin, simplifying the UI development process and providing powerful tools for UI composition.
* **Code Completion:** Android Studio offers robust code completion features that help developers write code faster and with fewer errors. It provides intelligent suggestions and auto-completion for classes, methods, variables, and other code elements, enhancing productivity and reducing manual typing.
* **Live Edit:** Android Studio includes a live editing feature that allows developers to preview changes to their app's UI in real-time as they edit the code. This instant feedback loop enables rapid iteration and refinement of the user interface, making it easier to visualize and implement design changes.
* **Gradle-based Flexible Build System:** Android Studio is built on top of the Gradle build system, which provides a flexible and customizable way to build, test, and package Android apps. Gradle simplifies complex build configurations, supports dependency management, and allows for incremental builds, improving build performance and scalability.
* **Android Device Emulator:** Android Studio includes a powerful emulator that simulates Android devices for testing and debugging apps. The emulator provides a virtual environment for running Android apps on different device configurations, screen sizes, and API levels, enabling developers to test their apps across a wide range of scenarios without needing physical devices.

**Machine Learning**

Machine Learning is a branch of artificial intelligence that develops algorithms by learning the hidden patterns of the datasets used it to make predictions on new similar type data, without being explicitly programmed for each task.

Machine learning is used in many different applications, from image and speech recognition to natural language processing, recommendation systems, fraud detection, portfolio optimization, automated task, and so on. Machine learning models are also used to power autonomous vehicles, drones, and robots, making them more intelligent and adaptable to changing environments.

A typical machine learning tasks are to provide a recommendation. Recommender systems are a common application of machine learning, and they use historical data to provide personalized recommendations to users.

**Types of Machine Learning:**

1. Supervised Machine Learning
2. Unsupervised Machine Learning
3. Reinforcement Machine Learning

**Unsupervised Machine Learning**

Unsupervised Learning Unsupervised learning is a type of machine learning technique in which an algorithm discovers patterns and relationships using unlabelled data. The primary goal of Unsupervised learning is often to discover hidden patterns, similarities, or clusters within the data, which can then be used for various purposes, such as data exploration, visualization, dimensionality reduction, and more.

**Reinforcement Machine learning**

Reinforcement Learning (RL) is a type of machine learning paradigm where an agent learns to make decisions by interacting with an environment. In this technique, the model keeps on increasing its performance using Reward Feedback to learn the behaviour or pattern. These algorithms are specific to a particular problem e.g., Google Self Driving car, AlphaGo where a bot competes with humans and even itself to get better and better performers in Go Game. Each time we feed in data, they learn and add the data to their knowledge which is training data. So, the more it learns the better it gets trained and hence experienced.

**Supervised Learning:**

In this project supervised learning has been chosen for usage, supervised learning is a type of machine learning in which the algorithm is trained on the labelled dataset i.e., the data may have correct values as well. It learns to map input features to targets based on labelled training data. In supervised learning, the algorithm is provided with input features and corresponding output labels, and it learns to generalize from this data to make predictions on new, unseen data.

There are two main types of supervised learning:

1. Regression
2. Classification

**Content based filtering**

Content-based filtering is a machine learning approach used in recommendation systems to suggest items to users based on the characteristics of the items and the preferences of the users. content-based filtering considers the features or attributes of items and recommends new items that are similar to those a user has liked or interacted with in the past.

In machine learning applications, content-based filtering finds its use in various recommendation systems, such as suggesting movies, music, articles, or products to users based on their historical preferences or explicit feedback.

Using content-based filtering in machine learning models can recommend new items that align with users' tastes, making the recommendation process more personalized and potentially increasing user satisfaction and engagement.

**TensorFlow**

TensorFlow is an open-source machine learning library developed by Google. TensorFlow is used to build and train deep learning models as it facilitates the creation of computational graphs and efficient execution on various hardware platforms.

TensorFlow offers multiple data tools to help you consolidate, clean and pre-process data at scale:

* Standard datasets for initial training and validation
* Highly scalable data pipelines for loading data
* Pre-processing layers for common input transformations
* Tools to validate and transform large datasets

Additionally, responsible AI tools help you uncover and eliminate bias in your data to produce fair, ethical outcomes from your models.

**Keras**Keras is the high-level API of the TensorFlow platform. It provides an approachable, highly-productive interface for solving machine learning (ML) problems, with a focus on modern deep learning. Keras covers every step of the machine learning workflow, from data processing to hyperparameter tuning to deployment. It was developed with a focus on enabling fast experimentation.

With Keras, you have full access to the scalability and cross-platform capabilities of TensorFlow. You can run Keras on a TPU Pod or large clusters of GPUs, and you can export Keras models to run in the browser or on mobile devices. You can also serve Keras models via a web API.

Keras is designed to reduce cognitive load by achieving the following goals:

* Offer simple, consistent interfaces.
* Minimize the number of actions required for common use cases.
* Provide clear, actionable error messages.
* Follow the principle of progressive disclosure of complexity: It's easy to get started, and you can complete advanced workflows by learning as you go.
* Help you write concise, readable code.

**Keras API components**

The core data structures of Keras are layers and models. A layer is a simple input/output transformation, and a model is a directed acyclic graph (DAG) of layers.

**Layers**

The tf.keras.layers.Layer class is the fundamental abstraction in Keras. A Layer encapsulates a state (weights) and some computation (defined in the tf.keras.layers.Layer.call method).

Weights created by layers can be trainable or non-trainable. Layers are recursively composable: If you assign a layer instance as an attribute of another layer, the outer layer will start tracking the weights created by the inner layer.

You can also use layers to handle data preprocessing tasks like normalization and text vectorization. Preprocessing layers can be included directly into a model, either during or after training, which makes the model portable.

**Models**

A model is an object that groups layers together and that can be trained on data.

The simplest type of model is the Sequential model, which is a linear stack of layers. For more complex architectures, you can either use the Keras functional API, which lets you build arbitrary graphs of layers, or use subclassing to write models from scratch.

The tf.keras.Model class features built-in training and evaluation methods:

* tf.keras.Model.fit: Trains the model for a fixed number of epochs.
* tf.keras.Model.predict: Generates output predictions for the input samples.
* tf.keras.Model.evaluate: Returns the loss and metrics values for the model; configured via the tf.keras.Model.compile method.

These methods give you access to the following built-in training features:

* Callbacks: You can leverage built-in callbacks for early stopping, model checkpointing, and TensorBoard monitoring. You can also implement custom callbacks.
* Distributed training: You can easily scale up your training to multiple GPUs, TPUs, or devices.
* Step fusing: With the steps\_per\_execution argument in tf.keras.Model.compile, you can process multiple batches in a single tf.function call, which greatly improves device utilization on TPUs.

# **Chapter 2: Definition of Problem**

## **2.1 Problem:**

Traditional tourism apps will just sell you a plan regarding the place you are visiting, your time is not your own during such a vacation as you will have set time you can visit certain places in a bid to visit other places that you will be guided to by the tourism agency guide. The challenge is to develop an app that will help you travel on your own terms, by assisting you in related things like what landmarks to visit, telling the general pricing of various amenities in the area etc.

## **2.2 Background:**

A lot of times for their forth coming vacations some people might rather prefer traveling on their own leisure to the places they want to tour and as such might need to conduct detailed research regarding the area in order to guarantee a safe and satisfying journey with little amount conventional roadblocks. However, such information is spread out all over the internet not mention or requires communication with locals. Adding the problem of live information or regularly fluctuating information on top of that, makes it a need that such a place exists to provide people with such information.

## **2.3 Relevance:**

The project displays its relevance in the fact that it will keep users up to date regarding various factors of their self-traveling journey with no assistance from tourism agencies for example., pricing of various amenities like hotel, travel etc., traffic in the places they are planning to visit in their travels etc.

# **Chapter 3: System Requirements**

## **3.1 Hardware Requirements:**

Windows 10 or above

RAM: 16 or above

Processor: Intel core i5 5th Gen or above

Android device: 4GB Ram, Version: 11+

## **3.2 Software Requirements:**

Android Studio

Java

Python

Jupyter Notebook

Android Virtual Device or Emulator

XML for UI

Database using Firebase

Version Control System (Git)

Machine learning with Tensorflow

# **Chapter 4: System Analysis & Design**

## **4.1 Iterative model**

The Iterative Waterfall Model is a software development approach that combines the sequential steps of the traditional Waterfall Model with the flexibility of iterative design. It allows for improvements and changes to be made at each stage of the development process, instead of waiting until the end of the project. The iterative waterfall model provides feedback paths from every phase to its preceding phases, which is the main difference from the classical waterfall model.

When errors are detected at some later phase, these feedback paths allow for correcting errors committed by programmers during some phase. The feedback paths allow the phase to be reworked in which errors are committed and these changes are reflected in the later phases.

But, there is no feedback path to the stage – feasibility study, because once a project has been taken, does not give up the project easily.

It is good to detect errors in the same phase in which they are committed. It reduces the effort and time required to correct the errors.

A real-life example could be building a new website for a small business.

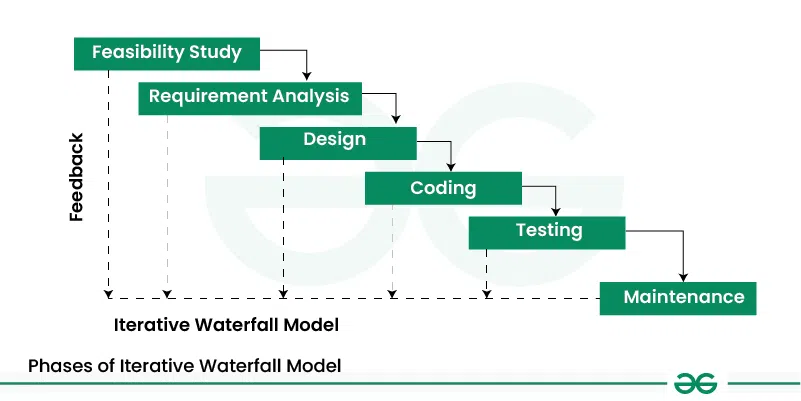


Figure 1:Iterative model

**1. Requirements Gathering**

This is the first stage where the business owners and developers meet to discuss the goals and requirements of the website.

**2. Design**

In this stage, the developers create a preliminary design of the website based on the requirements gathered in stage 1.

**3. Implementation**

In this stage, the developers begin to build the website based on the design created in stage 2.

**4. Testing**

Once the website has been built, it is tested to ensure that it meets the requirements and functions properly.

**5. Deployment**

The website is then deployed and made live to the public.

**6. Review and Improvement**

After the website has been live for a while, the business owners and developers review its performance and make any necessary improvements.

This process is repeated until the website meets the needs and goals of the business. Each iteration builds upon the previous one, allowing for continuous improvement and iteration until the final product is complete.

## **4.2 Use Case Diagram:**

A Use Case Diagram is a type of Unified Modeling Language (UML) diagram that represents the interaction between actors (users or external systems) and a system under consideration to accomplish specific goals. It provides a high-level view of the system’s functionality by illustrating the various ways users can interact with it.

**1. Use Case Diagram Notations**

UML notations provide a visual language that enables software developers, designers, and other stakeholders to communicate and document system designs, architectures, and behaviors in a consistent and understandable manner.

**1.1 Actors**

Actors are external entities that interact with the system. These can include users, other systems, or hardware devices. In the context of a Use Case Diagram, actors initiate use cases and receive the outcomes. Proper identification and understanding of actors are crucial for accurately modeling system behavior.

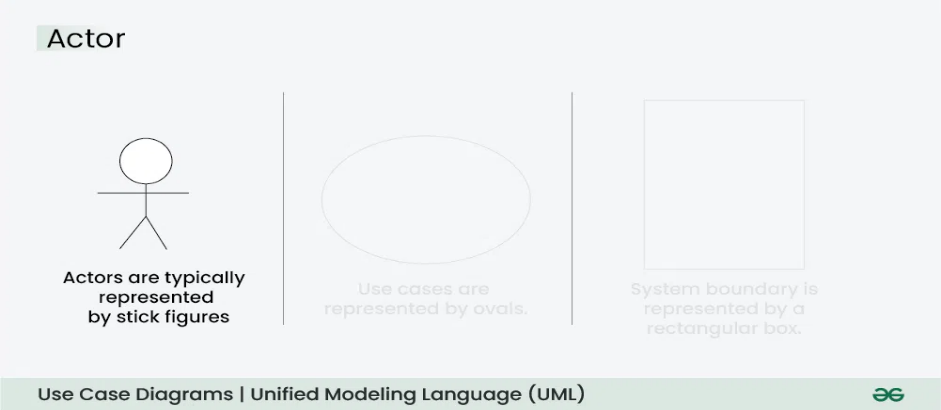


Figure 2:Use case Actor

**1.2 Use Cases**

Use cases are like scenes in the play. They represent specific things your system can do. In the online shopping system, examples of use cases could be “Place Order,” “Track Delivery,” or “Update Product Information”. Use cases are represented by ovals.

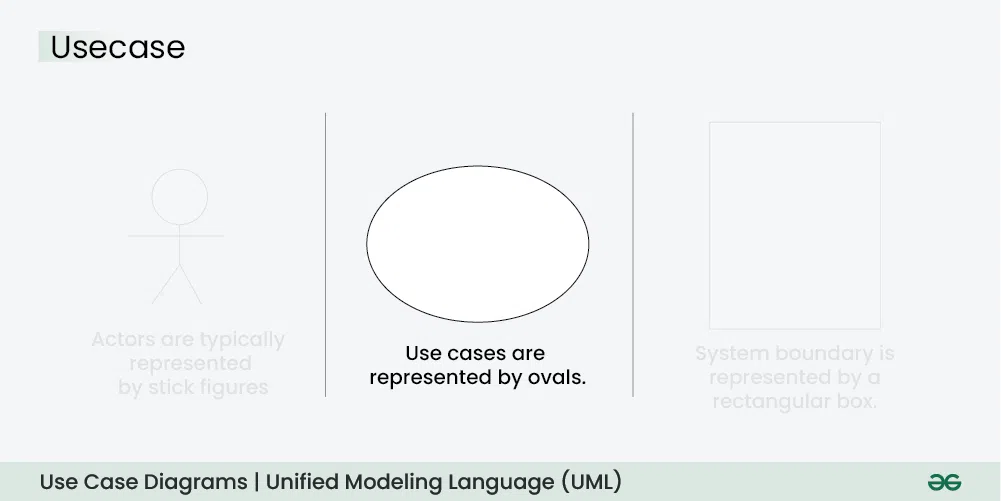


Figure 3: Use case representation

**1.3 System Boundary**

The system boundary is a visual representation of the scope or limits of the system you are modeling. It defines what is inside the system and what is outside. The boundary helps to establish a clear distinction between the elements that are part of the system and those that are external to it. The system boundary is typically represented by a rectangular box that surrounds all the use cases of the system.

**Purpose of System Boundary:**

Scope Definition: It clearly outlines the boundaries of the system, indicating which components are internal to the system and which are external actors or entities interacting with the system.

Focus on Relevance: By delineating the system’s scope, the diagram can focus on illustrating the essential functionalities provided by the system without unnecessary details about external entities.

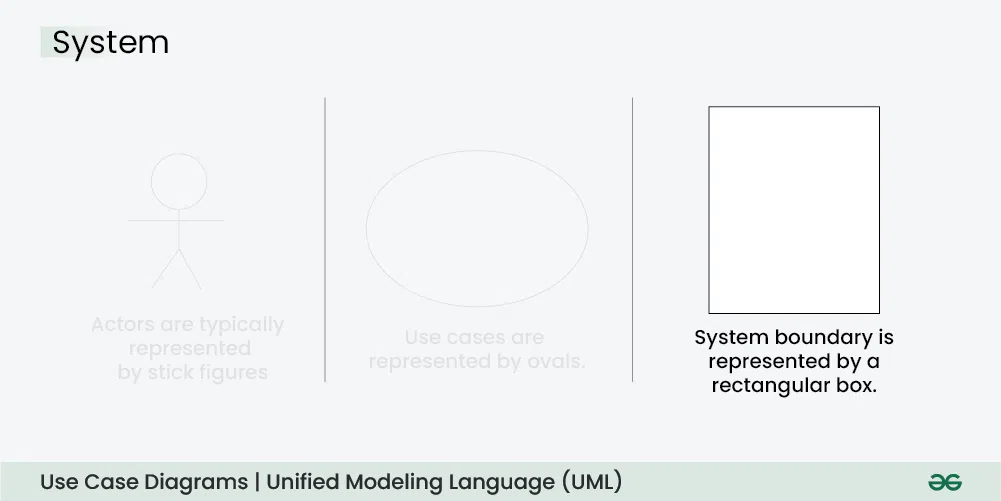


Figure 4: System boundary representation

**2. Use Case Diagram Relationships**

In a Use Case Diagram, relationships play a crucial role in depicting the interactions between actors and use cases. These relationships provide a comprehensive view of the system’s functionality and its various scenarios. Let’s delve into the key types of relationships and explore examples to illustrate their usage.

**2.1. Association Relationship**

The Association Relationship represents a communication or interaction between an actor and a use case. It is depicted by a line connecting the actor to the use case. This relationship signifies that the actor is involved in the functionality described by the use case.

Example: Online Banking System

Actor: Customer

Use Case: Transfer Funds

Association: A line connecting the “Customer” actor to the “Transfer Funds” use case, indicating the customer’s involvement in the funds transfer process.

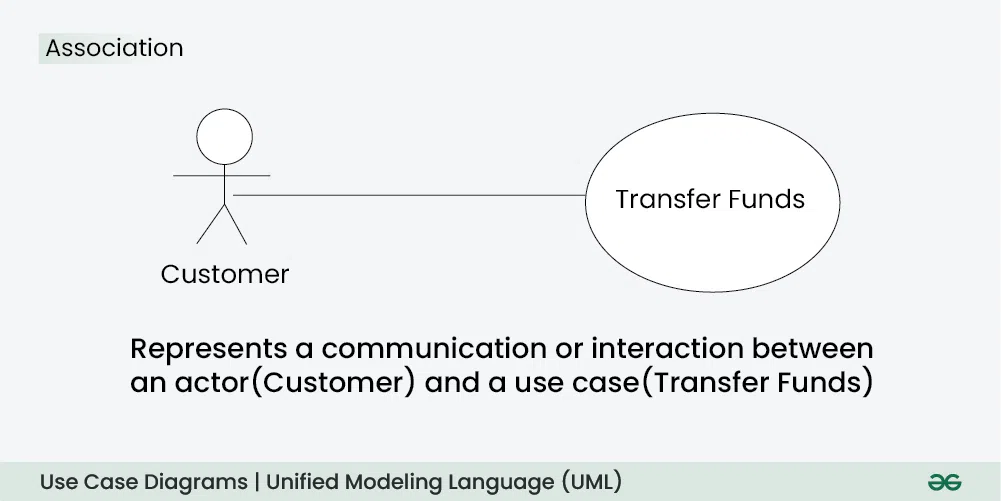


Figure 5: Association Relationship

**2.2. Include Relationship**

The Include Relationship indicates that a use case includes the functionality of another use case. It is denoted by a dashed arrow pointing from the including use case to the included use case. This relationship promotes modular and reusable design.

Example: Social Media Posting

Use Cases: Compose Post, Add Image

Include Relationship: The “Compose Post” use case includes the functionality of “Add Image.” Therefore, composing a post includes the action of adding an image.

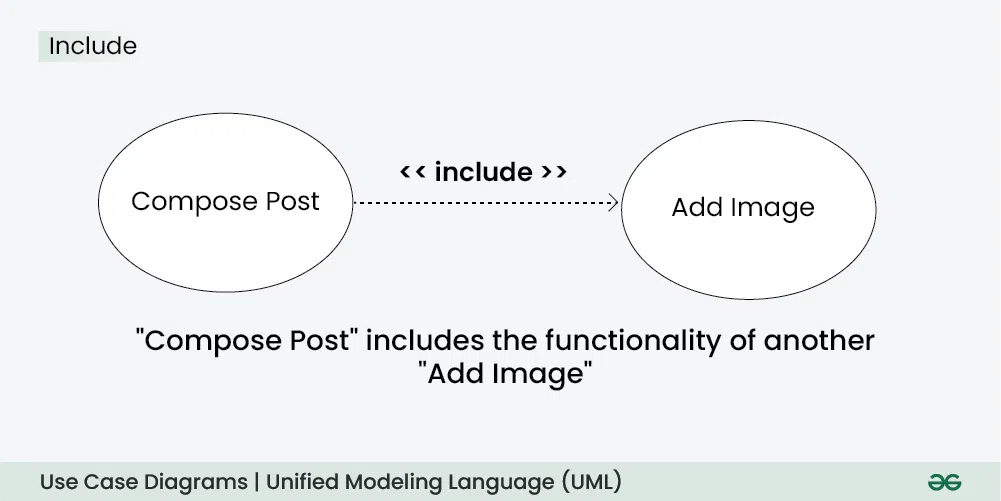


Figure 6: Include relationship

**2.3. Extend Relationship**

The Extend Relationship illustrates that a use case can be extended by another use case under specific conditions. It is represented by a dashed arrow with the keyword “extend.” This relationship is useful for handling optional or exceptional behavior.

Example: Flight Booking System

Use Cases: Book Flight, Select Seat

Extend Relationship: The “Select Seat” use case may extend the “Book Flight” use case when the user wants to choose a specific seat, but it is an optional step.

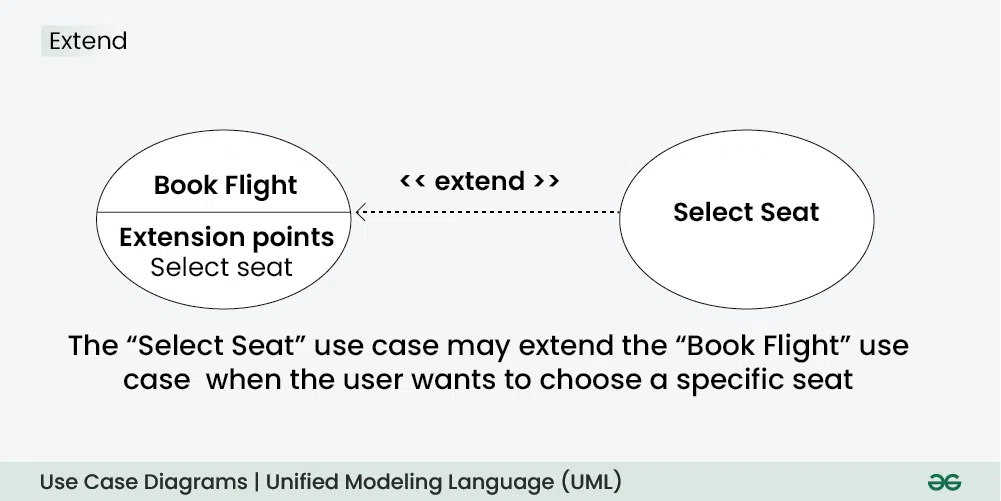


Figure 7:Extend Relationship

**3.4. Generalization Relationship**

The Generalization Relationship establishes an “is-a” connection between two use cases, indicating that one use case is a specialized version of another. It is represented by an arrow pointing from the specialized use case to the general use case.

Example: Vehicle Rental System

Use Cases: Rent Car, Rent Bike

Generalization Relationship: Both “Rent Car” and “Rent Bike” are specialized versions of the general use case “Rent Vehicle.”

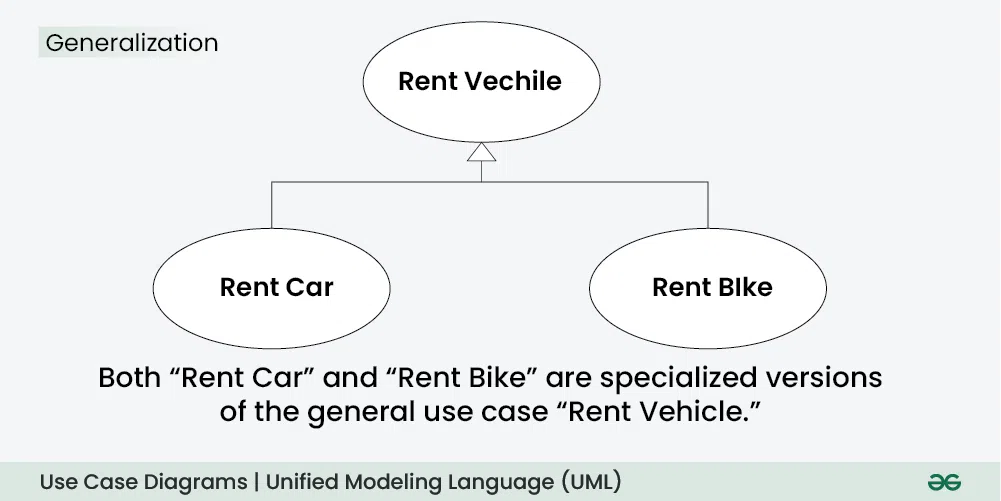


Figure 8: Generalization

Use Case Diagram of Tour Recommenation Application

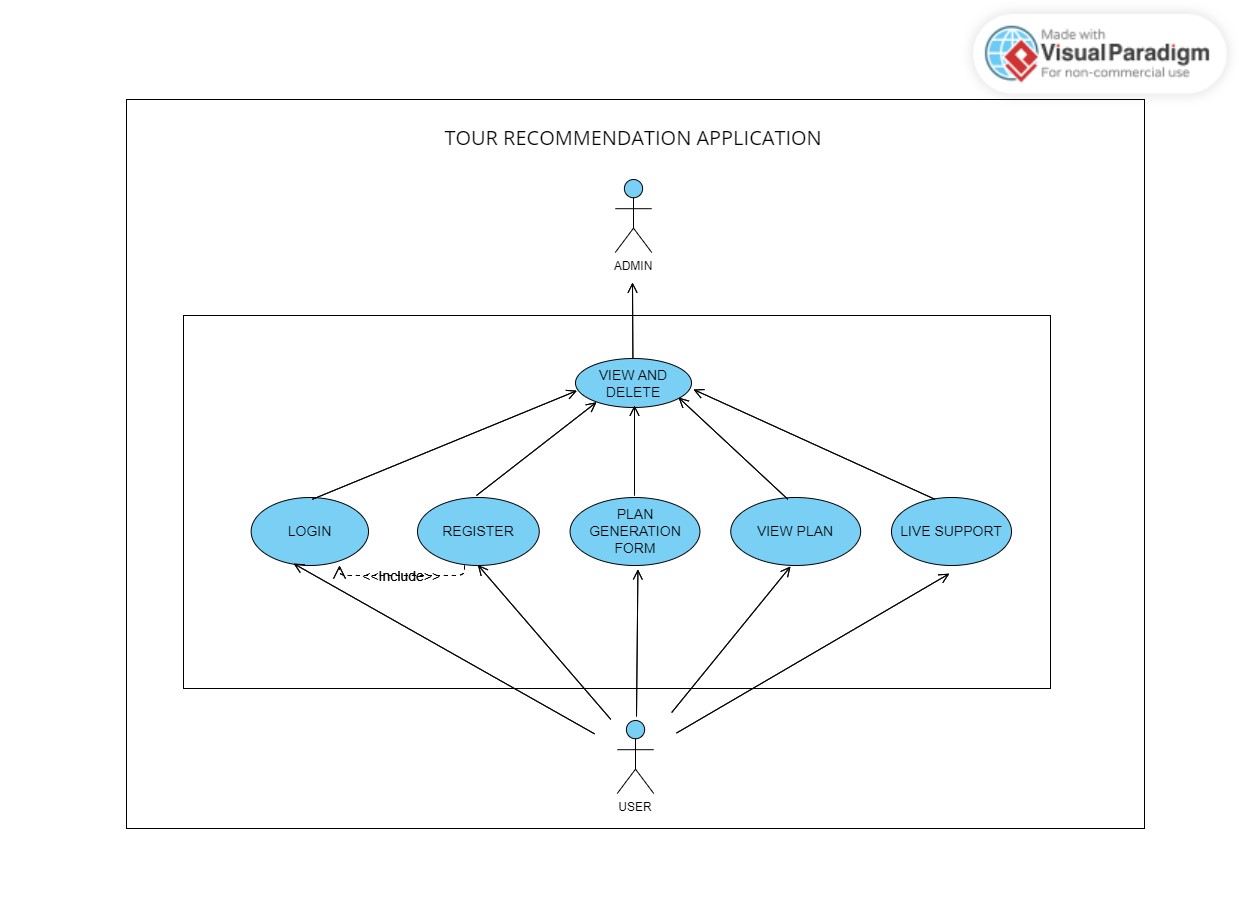


Figure 9 Use Case Diagram

# **Chapter 5: System Planning (PERT CHART)**

A PERT chart, sometimes called a PERT diagram, is a project management tool used to schedule, organize and coordinate tasks within a project. It provides a graphical representation of a project's timeline that enables project managers to break down each individual task in the project for analysis.

The PERT chart template uses nodes -- drawn as rectangles or circles -- to represent events and milestones throughout the project. The nodes are connected by vectors -- drawn as lines -- which represent the various tasks that need to be completed.

PERT charts provide project managers with an estimation of the minimum amount of time needed to complete a project. Managers can also analyze the work breakdown and task connections, as well as assess the risk associated with the project. The breakdown structure makes it easy to organize a complex project with a variety of moving parts by visualizing the dependencies between each step of the process.

PERT stands for Program Evaluation Review Technique, a methodology developed by the U.S. Navy in the 1950s to manage the Polaris submarine missile program. A similar methodology, the critical path method (CPM) was developed for project management in the private sector at about the same time.

**Characteristics of PERT:**

* It serves as a base for obtaining the important facts for implementing the decision-making.
* It forms the basis for all the planning activities.
* It helps management in deciding the best possible resource utilization method.
* It took advantage by using time network analysis technique.
* It presents the structure for reporting information.
* It helps the management in identifying the essential elements for the completion of the project within time.
* It specifies the activities that from the critical path.
* It describes the probability of completion of project before the specified date.
* It describes the dependencies of one or more tasks on each other.
* It represents the project in graphical plan form.

**Components of PERT Chart:**

**Arrows:** Connects one activity to the next.

**Table:** Contains the various measure of regarding the duration of the activity and the activity’s name.

**Start:** Represents the start of the project:

**Finish:**  Represents the end the of the project.

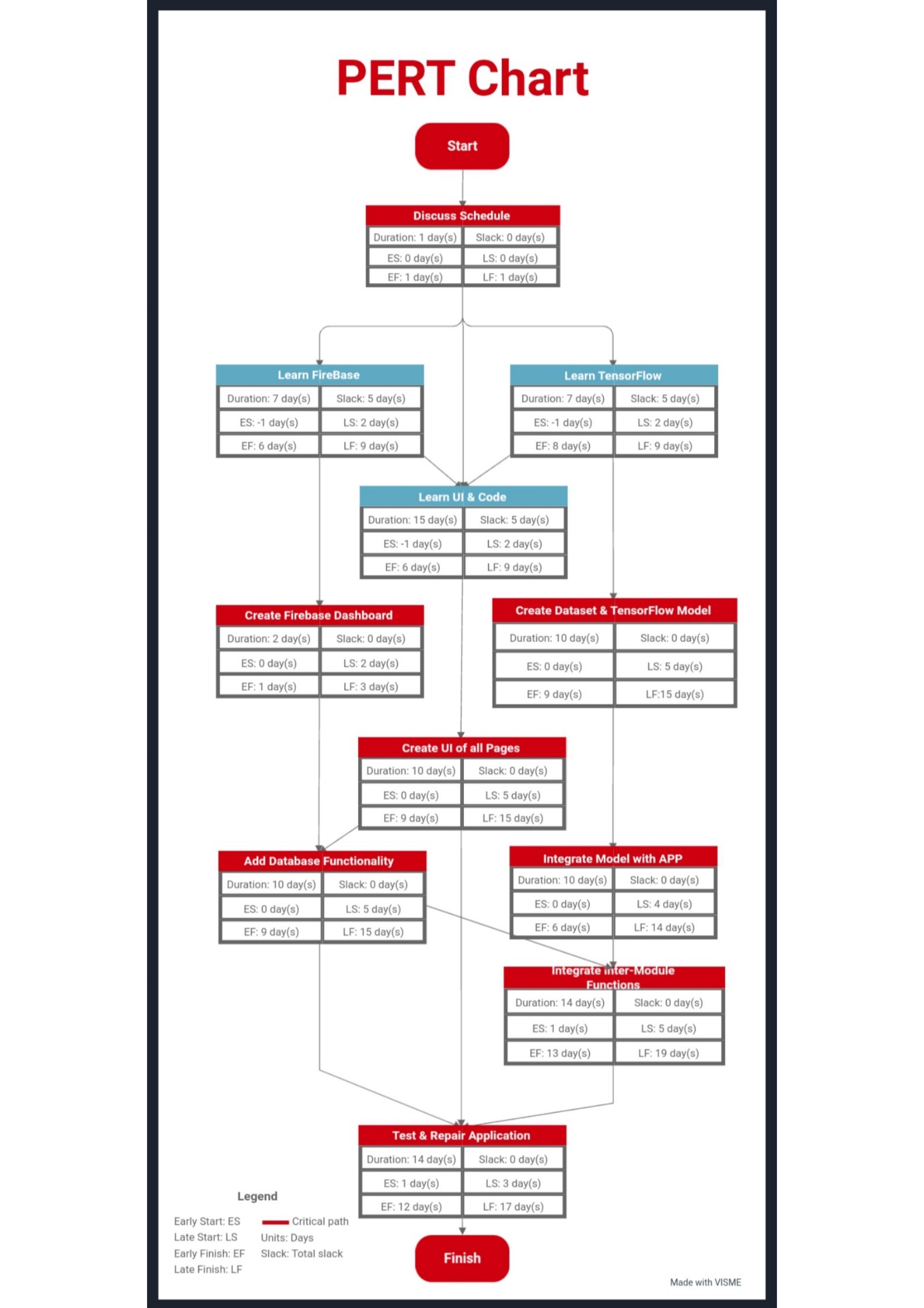


Figure 10 Pert Chart

# **Chapter 6: Methodology**

1. **Data Collection & Creation:** Procure data regarding tourism plans, pricing, places, cities etc., from the internet as well as creating it.
2. **Machine Learning Model:** Develop a machine learning model which provides the best suiting recommendation to the user, incorporating various inputs developed during data collection and creation.
3. **Recommendation Generation:** Utilize the model’s filtering algorithm to generate individualized plan recommendation for each user, incorporating the various details they provide such as their budget, place of travel, number of days, possible route etc.
4. **User Interface:** Make a user-friendly and interactive interface that is intuitive to use and attractive to look at, it should be able to incorporate all planned functions in an organized and smooth manner.
5. **Functionality:** The user interface will be implemented with required functionality after being planned and created.

# **Chapter 7: Detail life cycle of the project**

## **7.1 ERD (Entity Relationship Diagram)**

ER model stands for an Entity-Relationship model. It is a high-level data model. This model is used to define the data elements and relationship for a specified system.

It develops a conceptual design for the database. It also develops a very simple and easy to design view of data.

In ER modeling, the database structure is portrayed as a diagram called an entity-relationship diagram.

**Components of ERD:**



Figure 11 ER Components chart

**1. Entity:**

An entity may be any object, class, person or place. In the ER diagram, an entity can be represented as rectangles.

Consider an organization as an example- manager, product, employee, department etc. can be taken as an entity.

**Weak Entity:**

An entity that depends on another entity called a weak entity. The weak entity doesn't contain any key attribute of its own. The weak entity is represented by a double rectangle.

**2. Attribute**

The attribute is used to describe the property of an entity. Eclipse is used to represent an attribute.

For example, id, age, contact number, name, etc. can be attributes of a student.

**a. Key Attribute**

The key attribute is used to represent the main characteristics of an entity. It represents a primary key. The key attribute is represented by an ellipse with the text underlined.

**b. Composite Attribute**

An attribute that composed of many other attributes is known as a composite attribute. The composite attribute is represented by an ellipse, and those ellipses are connected with an ellipse.

**c. Multivalued Attribute**

An attribute can have more than one value. These attributes are known as a multivalued attribute. The double oval is used to represent multivalued attribute.

For example, a student can have more than one phone number.

**d. Derived Attribute**

An attribute that can be derived from other attribute is known as a derived attribute. It can be represented by a dashed ellipse.

For example, A person's age changes over time and can be derived from another attribute like Date of birth.

**3. Relationship**

A relationship is used to describe the relation between entities. Diamond or rhombus is used to represent the relationship.

**a. One-to-One Relationship**

When only one instance of an entity is associated with the relationship, then it is known as one to one relationship.

For example, A female can marry to one male, and a male can marry to one female.

**b. One-to-many relationship**

When only one instance of the entity on the left, and more than one instance of an entity on the right associates with the relationship then this is known as a one-to-many relationship.

For example, Scientist can invent many inventions, but the invention is done by the only specific scientist.

**c. Many-to-one relationship**

When more than one instance of the entity on the left, and only one instance of an entity on the right associates with the relationship then it is known as a many-to-one relationship.

For example, Student enrolls for only one course, but a course can have many students.

**d. Many-to-many relationship**

When more than one instance of the entity on the left, and more than one instance of an entity on the right associates with the relationship then it is known as a many-to-many relationship.

For example, Employee can assign by many projects and project can have many employees.

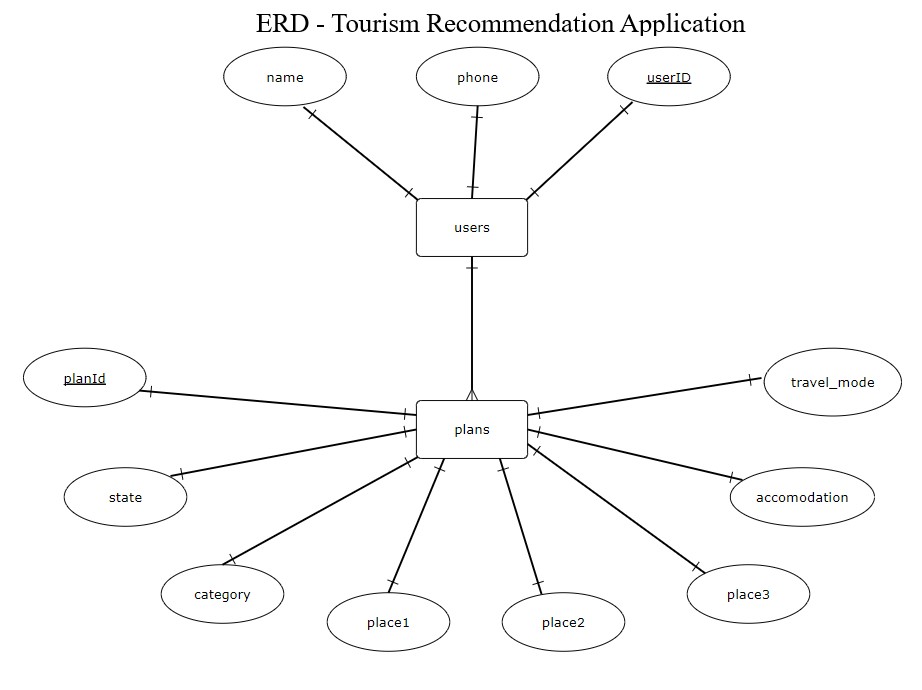


Figure 12 ER Diagram

## **7.2 DFD (Data Flow Diagram)**

Also known as DFD, Data flow diagrams are used to graphically represent the flow of data in a business information system. DFD describes the processes that are involved in a system to transfer data from the input to the file storage and reports generation.

Data flow diagrams can be divided into logical and physical. The logical data flow diagram describes flow of data through a system to perform certain functionality of a business. The physical data flow diagram describes the implementation of the logical data flow.

DFD Symbols

There are four basic symbols that are used to represent a data-flow diagram.

Process

A process receives input data and produces output with a different content or form. Processes can be as simple as collecting input data and saving in the database, or it can be complex as producing a report containing monthly sales of all retail stores in the northwest region.

Every process has a name that identifies the function it performs.

The name consists of a verb, followed by a singular noun.

Example:

Apply Payment

Calculate Commission

Verify Order

Data Flow

A data-flow is a path for data to move from one part of the information system to another. A data-flow may represent a single data element such the Customer ID or it can represent a set of data element (or a data structure).

Example:

Customer\_info (LastName, FirstName, SS#, Tel #, etc.)

Order\_info (OrderId, Item#, OrderDate, CustomerID, etc.).

**Data Store**

A data store or data repository is used in a data-flow diagram to represent a situation when the system must retain data because one or more processes need to use the stored data in a later time.

**External Entity**

An external entity is a person, department, outside organization, or other information system that provides data to the system or receives outputs from the system. External entities are components outside of the boundaries of the information systems. They represent how the information system interacts with the outside world.

A rectangle represents an external entity.

They either supply data or receive data.

They do not process data

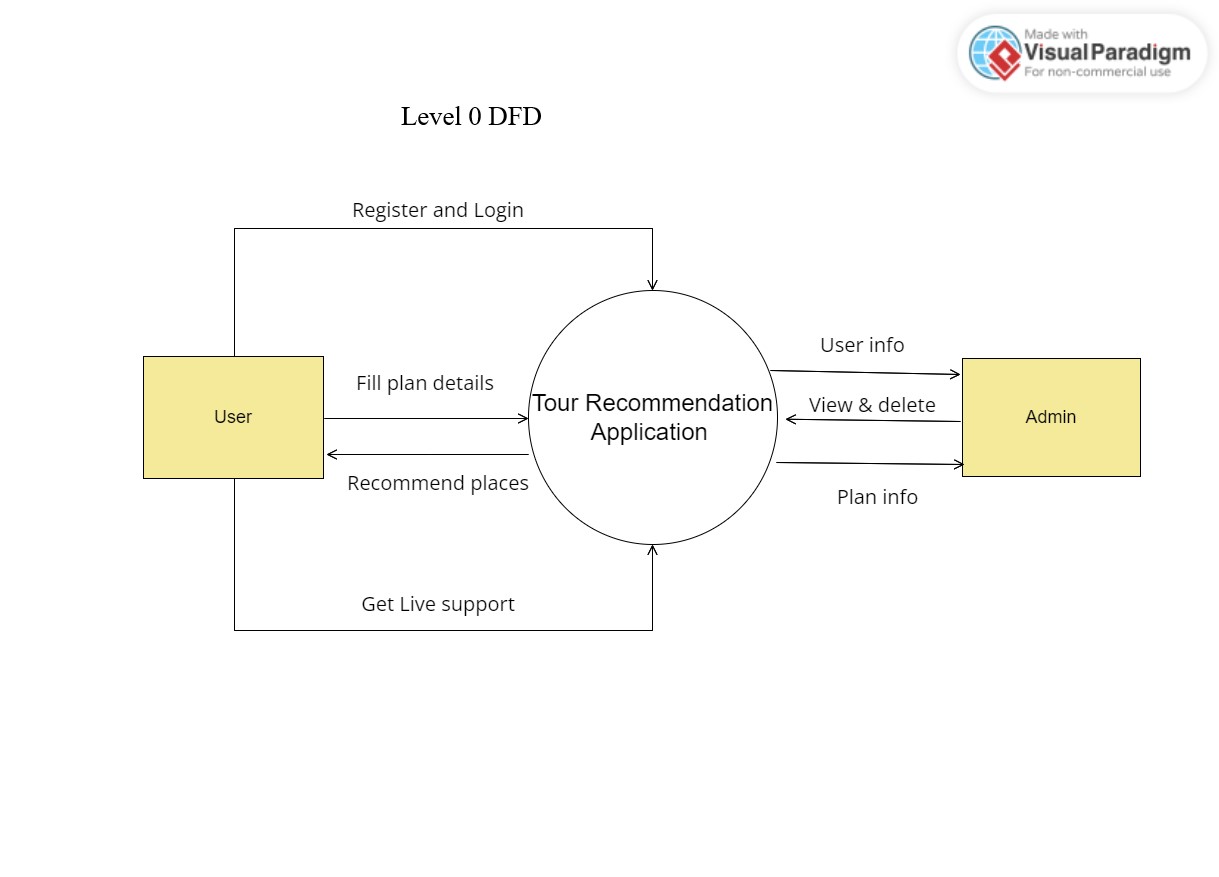


Figure 13: Level 0 DFD

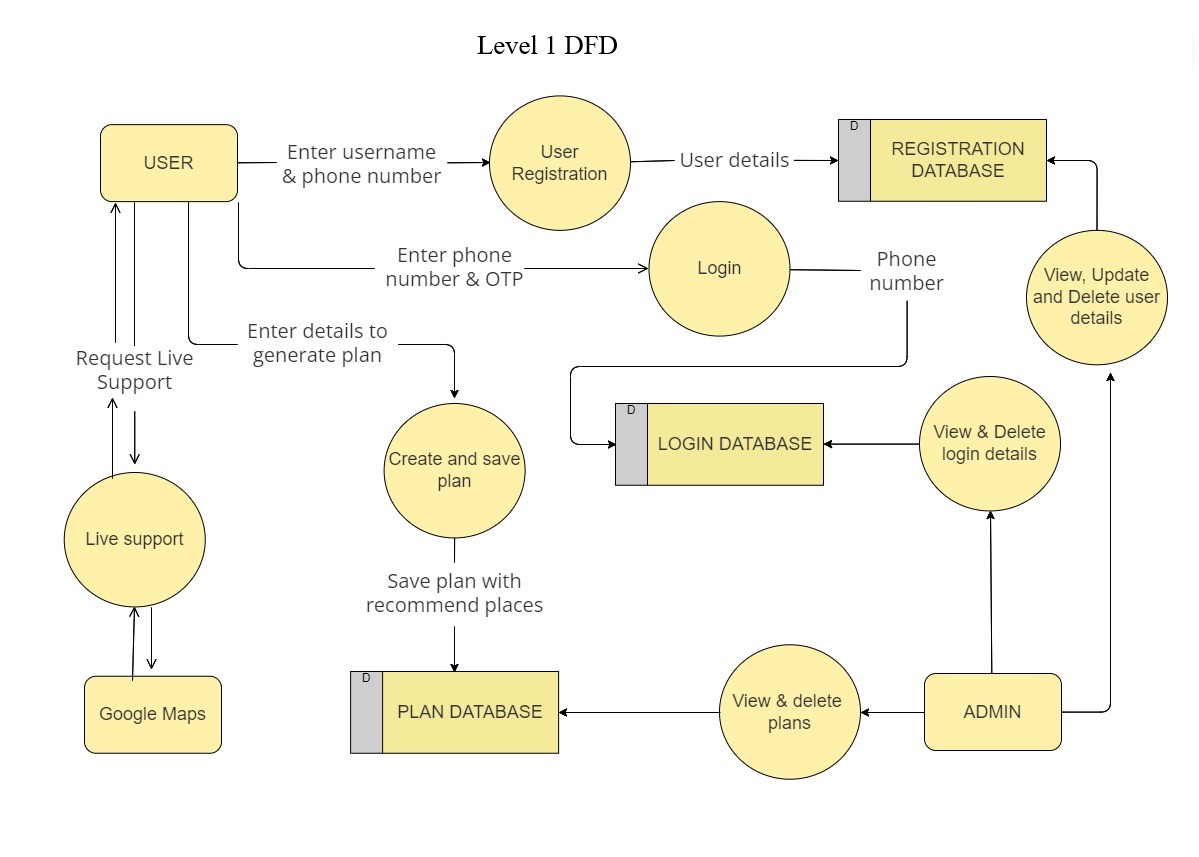


Figure 14:Level 1 DFD

## **7.3 Input and output screen design**

XML stands for Extensible Markup Language. XML is a markup language much like HTML used to describe data. It is derived from Standard Generalized Markup Language(SGML). Basically, the XML tags are not predefined in XML. We need to implement and define the tags in XML. XML tags define the data and used to store and organize data. It’s easily scalable and simple to develop. In Android, the XML is used to implement UI-related data, and it’s a lightweight markup language that doesn’t make layout heavy. XML only contains tags, while implementing they need to be just invoked.

In Android XML is used to implement the UI-related data. So understanding the core part of the UI interface with respect to XML is important. The User Interface for an Android App is built as the hierarchy of main layouts, widgets. The layouts are ViewGroup objects or containers that control how the child view should be positioned on the screen. Widgets here are view objects, such as Buttons and text boxes.

Different Types of XML Files Used in Android Studio

Different XML files serve different purposes in Android Studio. The list of various XML files in Android Studio with their purposes is discussed below.

**1. Layout XML files in android**

The Layout XML files are responsible for the actual User Interface of the application. It holds all the widgets or views like Buttons, TextViews, EditTexts, etc. which are defined under the ViewGroups.

**2. AndroidManifest.xml file**

This file describes the essential information about the application’s, like the application’s package names which matches code’s namespaces, a component of the application like activities, services, broadcast receivers, and content providers. Permission required by the user for the application features also mentioned in this XML file.

**3. strings.xml file**

This file contains texts for all the TextViews widgets. This enables reusability of code, and also helps in the localization of the application with different languages. The strings defined in these files can be used to replace all hardcoded text in the entire application.

**4. themes.xml file**

This file defines the base theme and customized themes of the application. It also used to define styles and looks for the UI(User Interface) of the application. By defining styles we can customize how the views or widgets look on the User Interface.

**5. Drawable XML files**

These are the XML files that provide graphics to elements like custom background for the buttons and its ripple effects, also various gradients can be created. This also holds the vector graphics like icons. Using these files custom layouts can be constructed for EditTexts.

**6. colors.xml file**

The colors.xml file is responsible to hold all the types of colors required for the application. It may be primary brand color and its variants and secondary brand color and its variants. The colors help uphold the brand of the applications. So the colors need to be decided cautiously as they are responsible for the User Experience. The colors need to be defined in the hex code format.

**7. dimens.xml file**

As the file name itself suggests that the file is responsible to hold the entire dimensions for the views. it may be the height of the Button, padding of the views, the margin for the views, etc. The dimensions need to in the format of pixel density(dp) values. Which replaces all the hard-coded dp values for the views. This file needs to be created separately in the values folder.

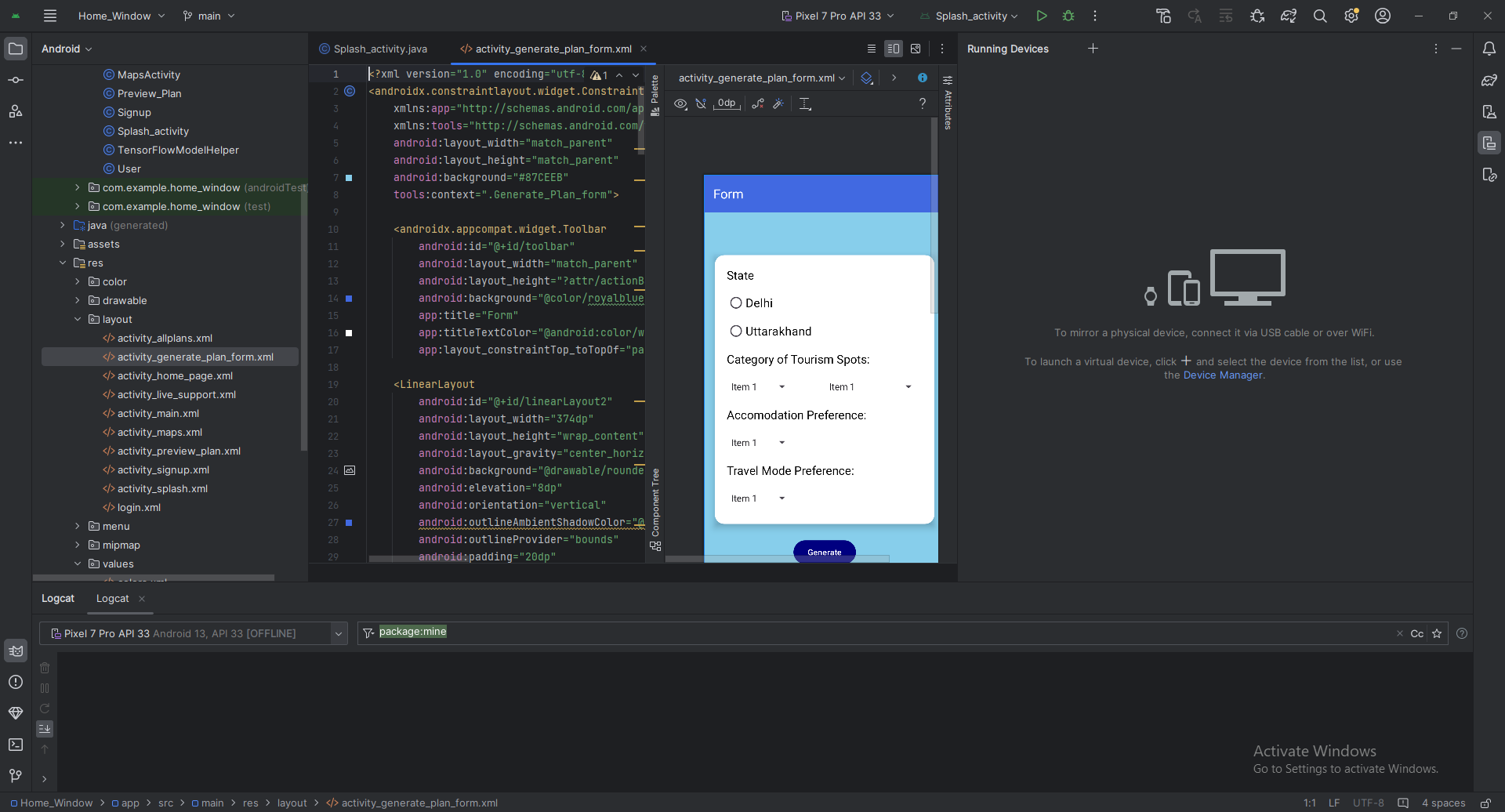


Figure 15: XML layout in Android Studio

## **7.4 Process Involved**

Activity Diagrams are used to illustrate the flow of control in a system and refer to the steps involved in the execution of a use case. We can depict both sequential processing and concurrent processing of activities using an activity diagram ie an activity diagram focuses on the condition of flow and the sequence in which it happens.

We describe what causes a particular event using an activity diagram.

An activity diagram portrays the control flow from a start point to a finish point showing the various decision paths that exist while the activity is being executed.

They are used in business and process modeling where their primary use is to depict the dynamic aspects of a system.

Activity Diagram Notations

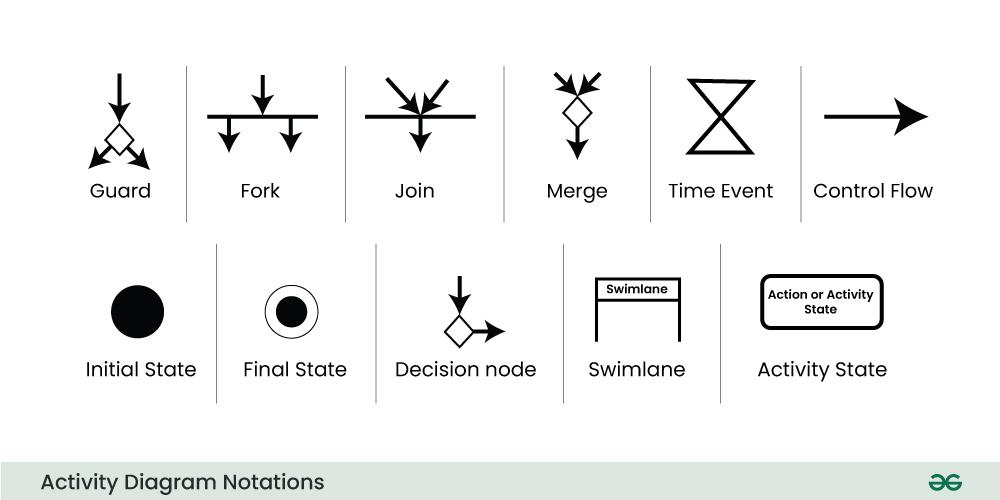


Figure 16: Activity Diagram Notations

**Initial State**

The starting state before an activity takes place is depicted using the initial state.

A process can have only one initial state unless we are depicting nested activities. We use a black filled circle to depict the initial state of a system. For objects, this is the state when they are instantiated. The Initial State from the UML Activity Diagram marks the entry point and the initial Activity State.

**Action or Activity State**

An activity represents execution of an action on objects or by objects. We represent an activity using a rectangle with rounded corners. Basically any action or event that takes place is represented using an activity.

**Action Flow or Control flows**

Action flows or Control flows are also referred to as paths and edges. They are used to show the transition from one activity state to another activity state.

An activity state can have multiple incoming and outgoing action flows. We use a line with an arrow head to depict a Control Flow. If there is a constraint to be adhered to while making the transition it is mentioned on the arrow.

**Decision node and Branching**

When we need to make a decision before deciding the flow of control, we use the decision node. The outgoing arrows from the decision node can be labelled with conditions or guard expressions. It always includes two or more output arrows.

**Guard**

A Guard refers to a statement written next to a decision node on an arrow sometimes within square brackets.

**Fork**

Fork nodes are used to support concurrent activities. When we use a fork node when both the activities get executed concurrently i.e. no decision is made before splitting the activity into two parts. Both parts need to be executed in case of a fork statement. We use a rounded solid rectangular bar to represent a Fork notation with incoming arrow from the parent activity state and outgoing arrows towards the newly created activities.

**Join**

Join nodes are used to support concurrent activities converging into one. For join notations we have two or more incoming edges and one outgoing edge.

**Merge or Merge Event**

Scenarios arise when activities which are not being executed concurrently have to be merged. We use the merge notation for such scenarios. We can merge two or more activities into one if the control proceeds onto the next activity irrespective of the path chosen.

**Swimlanes**

We use Swimlanes for grouping related activities in one column. Swimlanes group related activities into one column or one row. Swimlanes can be vertical and horizontal. Swimlanes are used to add modularity to the activity diagram. It is not mandatory to use swimlanes. They usually give more clarity to the activity diagram. It’s similar to creating a function in a program. It’s not mandatory to do so, but, it is a recommended practice.

**Time Event**

This refers to an event that stops the flow for a time; an hourglass depicts it. We can have a scenario where an event takes some time to completed.

**Final State or End State**

The state which the system reaches when a particular process or activity ends is known as a Final State or End State. We use a filled circle within a circle notation to represent the final state in a state machine diagram. A system or a process can have multiple final states.

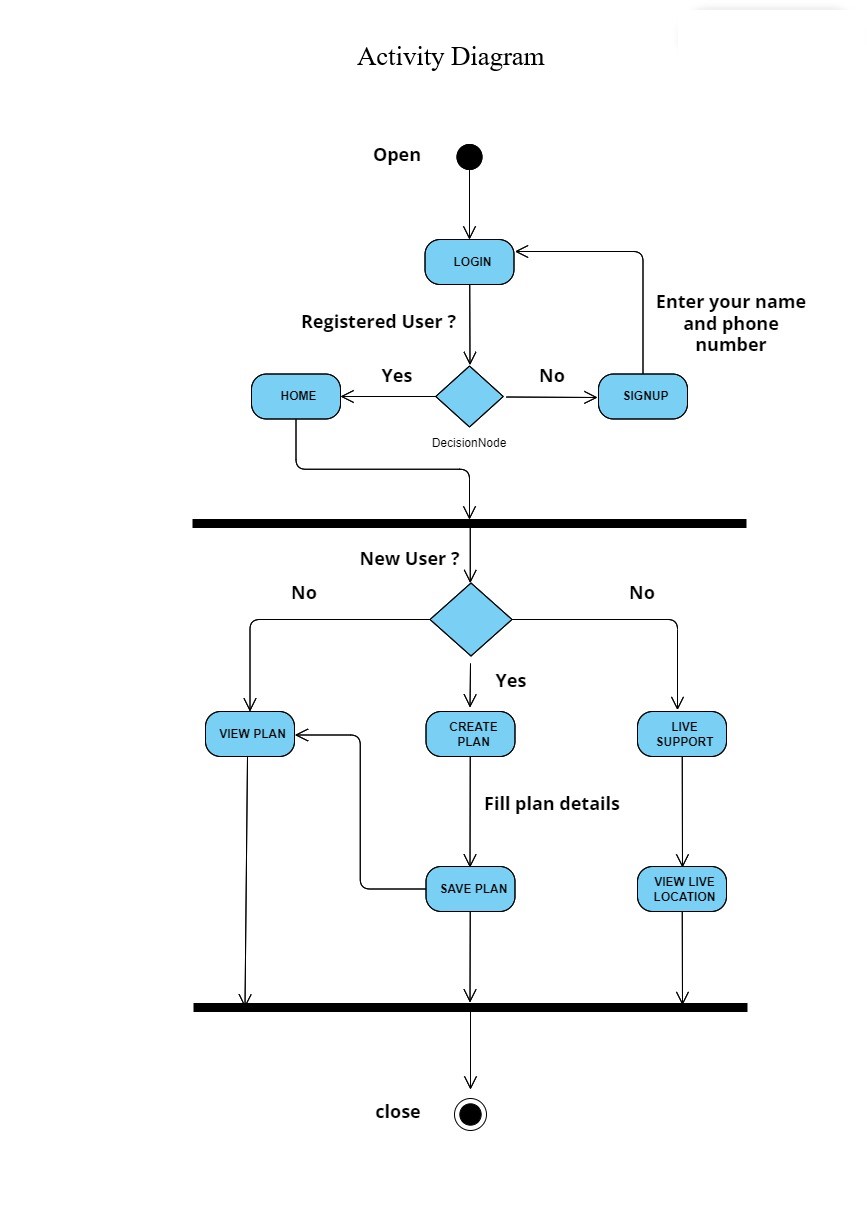


Figure 17: Activity Diagram

## **7.5 Methodology used in testing:**

Software Testing is a method to assess the functionality of the software program. The process checks whether the actual software matches the expected requirements and ensures the software is bug-free. The purpose of software testing is to identify the errors, faults, or missing requirements in contrast to actual requirements. It mainly aims at measuring the specification, functionality, and performance of a software program or application.

**Software testing can be divided into two steps:**

**Verification:** It refers to the set of tasks that ensure that the software correctly implements a specific function. It means “Are we building the product right?”.

**Validation:** It refers to a different set of tasks that ensure that the software that has been built is traceable to customer requirements. It means “Are we building the right product?”.

**Importance of Software Testing:**

**Defects can be identified early:** Software testing is important because if there are any bugs they can be identified early and can be fixed before the delivery of the software.

**Improves quality of software:** Software Testing uncovers the defects in the software, and fixing them improves the quality of the software.

**Increased customer satisfaction:** Software testing ensures reliability, security, and high performance which results in saving time, costs, and customer satisfaction.

**Helps with scalability:** Software testing type non-functional testing helps to identify the scalability issues and the point where an application might stop working.

**Saves time and money:** After the application is launched it will be very difficult to trace and resolve the issues, as performing this activity will incur more costs and time. Thus, it is better to conduct software testing at regular intervals during software development.

**Whitebox testing:**

White box testing techniques analyze the internal structures the used data structures, internal design, code structure, and the working of the software rather than just the functionality as in black box testing. It is also called glass box testing or clear box testing or structural testing. White Box Testing is also known as transparent testing or open box testing.

White box testing is a software testing technique that involves testing the internal structure and workings of a software application. The tester has access to the source code and uses this knowledge to design test cases that can verify the correctness of the software at the code level.

White box testing is also known as structural testing or code-based testing, and it is used to test the software’s internal logic, flow, and structure. The tester creates test cases to examine the code paths and logic flows to ensure they meet the specified requirements.

## **7.6 Test Report:**

**Signup:**

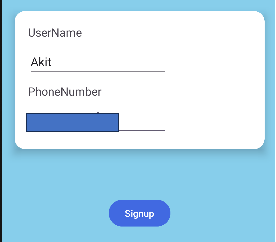


Figure 18: Test Report 1

**Expected Output:**User data is input into firebase database.

**Output:**



Figure 19: Test Report 2

**Result: Pass**

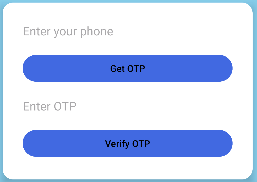
**Login:**

Figure 20: Test Report 3

**Expected Output:**

User enters phone number -> user gets OTP -> user enters OTP -> user logs in successfully.

**Output:**



Figure 21: Test Report 4

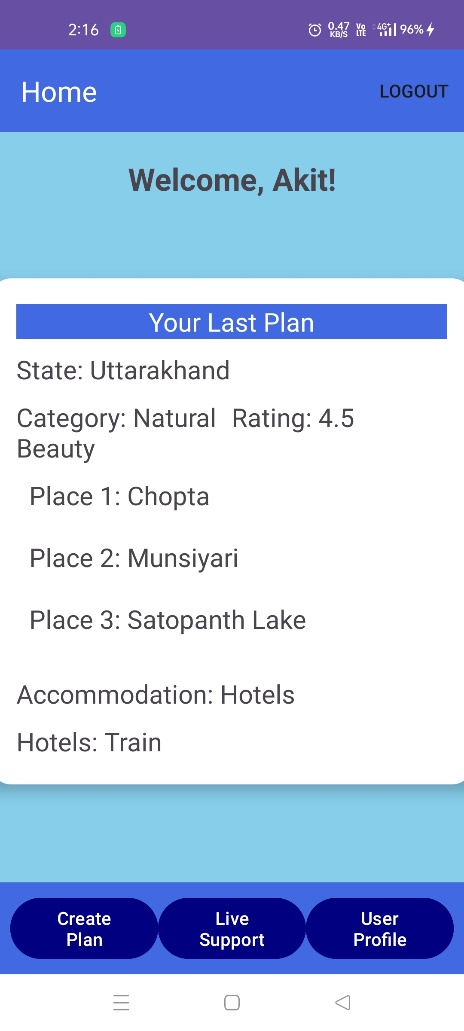


Figure 22: Test Report 5

**Create Plan:**

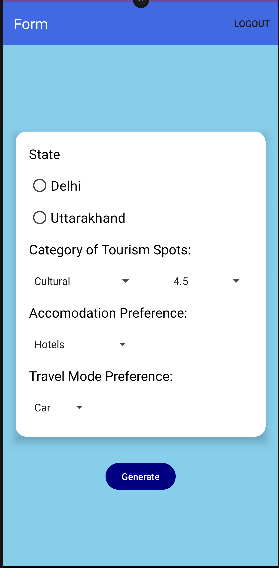


Figure 23: Test Report 6

**Expected Output:**  
User enters Plan Requirements -> User clicks generate -> taken to preview plan get

**Output:**

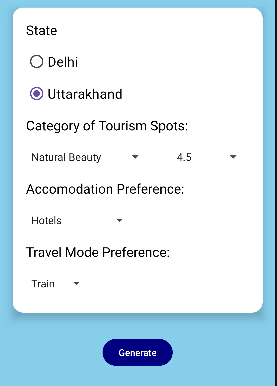


Figure 24: Test Report 7

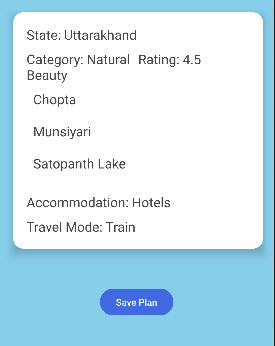


Figure 25: Test Report 8

**Result:** **PASS**

**Preview Plan:**

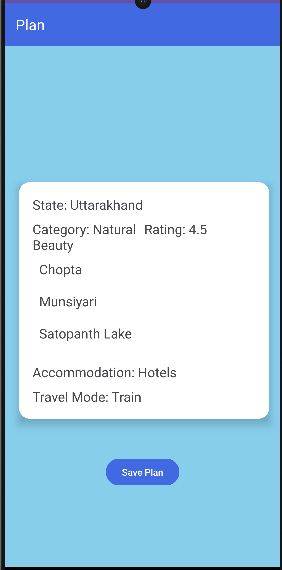


Figure 26: Test Report 9

**Expected Result:**

User clicks save plan -> Plan is saved in database.

**Output:**



Figure 27 Test Report 10

**Result: Pass**

**Live Support:**

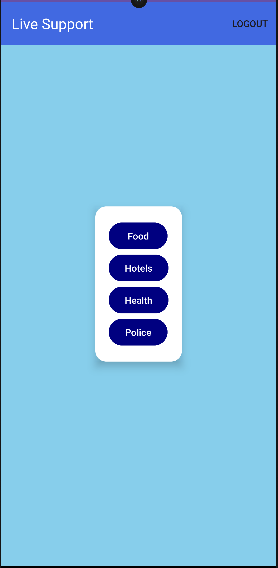


Figure 28: Test Report 11

**Expected Result:**

Pressing any of the buttons will send and intent to google maps, opening google maps and showing the nearby location of the things specified by the buttons.

**Output:**

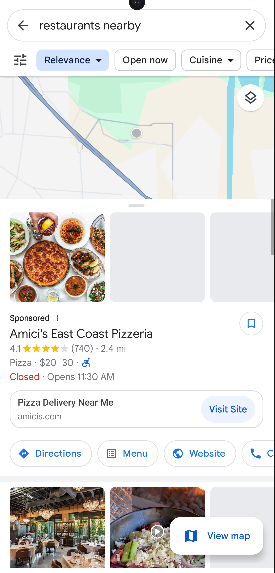


Figure 29: Test Report 12

**Result: Pass**

# **Chapter 8: Code and Screenshots**

**Splash Activity:**

**Class:**

import androidx.appcompat.app.AppCompatActivity;

import android.content.Intent;

import android.os.Bundle;

import android.os.Handler;

public class Splash\_activity extends AppCompatActivity {

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_splash);

Intent intent = new Intent(Splash\_activity.this, MainActivity.class);

new Handler().postDelayed(new Runnable() {

@Override

public void run() {

startActivity(intent);

finish();

}

},4000);

}

}

**Layout:**

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

<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"

xmlns:app="http://schemas.android.com/apk/res-auto"

xmlns:tools="http://schemas.android.com/tools"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

android:background="#87CEEB"

tools:context=".Splash\_activity">

<ImageView

android:id="@+id/imageView3"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

app:layout\_constraintBottom\_toBottomOf="parent"

app:layout\_constraintEnd\_toEndOf="parent"

app:layout\_constraintHorizontal\_bias="0.0"

app:layout\_constraintStart\_toStartOf="parent"

app:layout\_constraintTop\_toTopOf="parent"

app:layout\_constraintVertical\_bias="0.0"

app:srcCompat="@drawable/d" />

<TextView

android:id="@+id/textView2"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:background="@android:color/transparent"

android:padding="0dp"

android:text="Tour Recommendation App"

android:textAlignment="center"

android:textColor="@color/black"

android:textSize="30sp"

android:textStyle="bold"

android:visibility="visible"

app:layout\_constraintBottom\_toBottomOf="parent"

app:layout\_constraintEnd\_toEndOf="parent"

app:layout\_constraintHorizontal\_bias="0.627"

app:layout\_constraintStart\_toStartOf="parent"

app:layout\_constraintTop\_toTopOf="parent"

app:layout\_constraintVertical\_bias="0.146" />

</androidx.constraintlayout.widget.ConstraintLayout>

**Output:**



Figure 30: Output 1

**Main Activity:**

**Class:**

import androidx.appcompat.app.AppCompatActivity;

import android.content.Intent;

import android.os.Bundle;

import android.view.View;

import android.widget.Button;

public class MainActivity extends AppCompatActivity {

Button loginbtn, signupbtn;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_main);

loginbtn = findViewById(R.id.bt1);

signupbtn = findViewById(R.id.bt2);

loginbtn.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View v) {

// Start the Login activity

Intent intent = new Intent(MainActivity.this, Login.class);

startActivity(intent);

}

});

signupbtn.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View v) {

Intent intent = new Intent(MainActivity.this, Signup.class);

startActivity(intent);

}

});

}

}

**Layout:**

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

<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"

xmlns:app="http://schemas.android.com/apk/res-auto"

xmlns:tools="http://schemas.android.com/tools"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

android:background="#87CEEB"

tools:context=".MainActivity">

<ImageView

android:id="@+id/imageView"

android:layout\_width="418dp"

android:layout\_height="488dp"

android:src="@drawable/b"

app:layout\_constraintEnd\_toEndOf="parent"

app:layout\_constraintHorizontal\_bias="0.571"

app:layout\_constraintStart\_toStartOf="parent"

app:layout\_constraintTop\_toTopOf="parent"

tools:ignore="MissingConstraints" />

<TextView

android:id="@+id/textView"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_marginTop="32dp"

android:text="\t\tWelcome To The App\n Please Login to Proceed."

android:textColor="#000000"

android:textSize="12pt"

android:textStyle="bold"

app:layout\_constraintEnd\_toEndOf="parent"

app:layout\_constraintStart\_toStartOf="parent"

app:layout\_constraintTop\_toBottomOf="@+id/imageView"

tools:ignore="MissingConstraints" />

<TextView

android:id="@+id/textView3"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Tourism\nRecommendation\nApplication"

android:textAlignment="center"

android:textColor="#000000"

android:textSize="10pt"

android:textStyle="bold"

app:layout\_constraintBottom\_toBottomOf="@+id/imageView"

app:layout\_constraintEnd\_toEndOf="parent"

app:layout\_constraintHorizontal\_bias="0.497"

app:layout\_constraintStart\_toStartOf="parent"

app:layout\_constraintTop\_toTopOf="parent"

app:layout\_constraintVertical\_bias="0.637"

tools:ignore="MissingConstraints" />

<LinearLayout

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:orientation="horizontal"

app:layout\_constraintBottom\_toBottomOf="parent"

app:layout\_constraintEnd\_toEndOf="@+id/textView"

app:layout\_constraintStart\_toStartOf="@+id/textView"

app:layout\_constraintTop\_toBottomOf="@+id/textView">

<Button

android:id="@+id/bt1"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:backgroundTint="@color/royalblue"

android:text="Login"

android:textColor="@color/black"

android:textSize="12pt"

android:textStyle="bold" />

<Space

android:layout\_width="65dp"

android:layout\_height="wrap\_content"

android:layout\_weight="1" />

<Button

android:id="@+id/bt2"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:backgroundTint="@color/royalblue"

android:text="SignUp"

android:textColor="@color/black"

android:textSize="12pt"

android:textStyle="bold" />

</LinearLayout>

</androidx.constraintlayout.widget.ConstraintLayout>

**Output:**



Figure 31: Output 2

**Signup:**

**Class:**

public class Signup extends AppCompatActivity {

private EditText editTextName, editTextPhone;

private Button buttonSignup;

private DatabaseReference databaseReference;

private ValueEventListener userListener; // Listener to check if user already exists

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_signup);

databaseReference = FirebaseDatabase.getInstance().getReference("users");

editTextName = findViewById(R.id.editTextText);

editTextPhone = findViewById(R.id.editTextPhone);

buttonSignup = findViewById(R.id.button2);

buttonSignup.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View v) {

checkUserExists();

}

});

}

private void checkUserExists() {

final String name = editTextName.getText().toString().trim();

final String phone = editTextPhone.getText().toString().trim();

userListener = new ValueEventListener() {

@Override

public void onDataChange(DataSnapshot dataSnapshot) {

boolean userExists = false;

for (DataSnapshot userSnapshot : dataSnapshot.getChildren()) {

User user = userSnapshot.getValue(User.class);

if (user != null && user.getPhone().equals(phone)) {

userExists = true;

break;

}

}

if (userExists) {

Toast.makeText(Signup.this, "You are already registered, go to login page", Toast.LENGTH\_SHORT).show();

} else {

signupUser();

}

}

@Override

public void onCancelled(DatabaseError databaseError) {

Toast.makeText(Signup.this, "Database error: " + databaseError.getMessage(), Toast.LENGTH\_SHORT).show();

}

};

databaseReference.addListenerForSingleValueEvent(userListener);

}

private void signupUser() {

String name = editTextName.getText().toString().trim();

String phone = editTextPhone.getText().toString().trim();

if (name.isEmpty()) {

editTextName.setError("Name is required");

editTextName.requestFocus();

return;

}

if (phone.isEmpty()) {

editTextPhone.setError("Phone number is required");

editTextPhone.requestFocus();

return;

}

String userId = databaseReference.push().getKey();

if (userId == null) {

Toast.makeText(this, "Failed to create user. Please try again.", Toast.LENGTH\_SHORT).show();

return;

}

User user = new User(userId, name, phone);

databaseReference.child(userId).setValue(user);

finish();

}

@Override

protected void onDestroy() {

super.onDestroy();

// Remove the ValueEventListener to prevent memory leaks

if (userListener != null) {

databaseReference.removeEventListener(userListener);

}

}

}

**Layout:**

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

<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"

xmlns:app="http://schemas.android.com/apk/res-auto"

xmlns:tools="http://schemas.android.com/tools"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

android:background="#87CEEB"

tools:context=".Signup">

<LinearLayout

android:id="@+id/linearLayout"

android:layout\_width="374dp"

android:layout\_height="wrap\_content"

android:layout\_gravity="center\_horizontal"

android:background="@drawable/rounded\_rectangle"

android:elevation="8dp"

android:orientation="vertical"

android:outlineAmbientShadowColor="@color/royalblue"

android:outlineProvider="bounds"

android:padding="20dp"

app:layout\_constraintBottom\_toBottomOf="parent"

app:layout\_constraintEnd\_toEndOf="parent"

app:layout\_constraintStart\_toStartOf="parent"

app:layout\_constraintTop\_toTopOf="parent">>

<TextView

android:id="@+id/textView8"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="UserName"

android:textSize="17sp" />

<Space

android:layout\_width="match\_parent"

android:layout\_height="10dp" />

<EditText

android:id="@+id/editTextText"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:ems="10"

android:hint="Enter your name"

android:inputType="text" />

<Space

android:layout\_width="match\_parent"

android:layout\_height="10dp" />

<TextView

android:id="@+id/textView9"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="PhoneNumber"

android:textSize="17sp" />

<Space

android:layout\_width="match\_parent"

android:layout\_height="10dp" />

<EditText

android:id="@+id/editTextPhone"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:ems="10"

android:hint="Enter your phone number"

android:inputType="phone" />

</LinearLayout>

<Button

android:id="@+id/button2"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Signup"

android:elevation="8dp"

android:outlineAmbientShadowColor="@color/royalblue"

android:outlineProvider="bounds"

android:backgroundTint="@color/royalblue"

android:textColor="@color/white"

app:layout\_constraintBottom\_toBottomOf="parent"

app:layout\_constraintEnd\_toEndOf="parent"

app:layout\_constraintStart\_toStartOf="parent"

app:layout\_constraintTop\_toBottomOf="@+id/linearLayout"

app:layout\_constraintVertical\_bias="0.265" />

</androidx.constraintlayout.widget.ConstraintLayout>

**Output:**

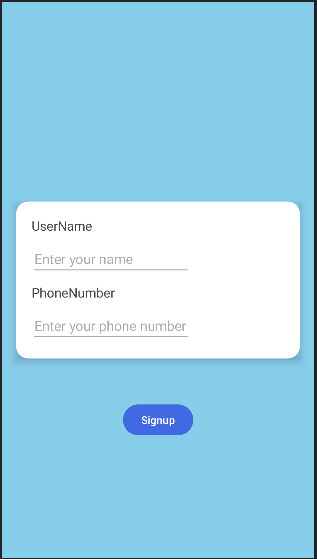


Figure 32: Output 3

**Login:**

**Class:**

import android.content.Intent;

import android.os.Bundle;

import android.text.TextUtils;

import android.util.Log;

import android.view.View;

import android.widget.Button;

import android.widget.EditText;

import android.widget.Toast;

import androidx.annotation.NonNull;

import androidx.appcompat.app.AppCompatActivity;

import com.google.android.gms.tasks.OnCompleteListener;

import com.google.android.gms.tasks.Task;

import com.google.firebase.FirebaseApp;

import com.google.firebase.FirebaseException;

import com.google.firebase.FirebaseTooManyRequestsException;

import com.google.firebase.auth.AuthResult;

import com.google.firebase.auth.FirebaseAuth;

import com.google.firebase.auth.FirebaseAuthInvalidCredentialsException;

import com.google.firebase.auth.FirebaseAuthMissingActivityForRecaptchaException;

import com.google.firebase.auth.FirebaseUser;

import com.google.firebase.auth.PhoneAuthCredential;

import com.google.firebase.auth.PhoneAuthOptions;

import com.google.firebase.auth.PhoneAuthProvider;

import com.google.firebase.appcheck.FirebaseAppCheck;

import com.google.firebase.appcheck.playintegrity.PlayIntegrityAppCheckProviderFactory;

import com.google.firebase.database.DataSnapshot;

import com.google.firebase.database.DatabaseError;

import com.google.firebase.database.DatabaseReference;

import com.google.firebase.database.FirebaseDatabase;

import com.google.firebase.database.ValueEventListener;

import java.util.concurrent.TimeUnit;

public class Login extends AppCompatActivity {

private static final String TAG = "Login";

private String enteredPhoneNumber;

private FirebaseAuth mAuth;

private EditText edtPhone, edtOTP;

private Button verifyOTPBtn, generateOTPBtn;

private String mVerificationId;

private PhoneAuthProvider.ForceResendingToken mResendToken;

private PhoneAuthProvider.OnVerificationStateChangedCallbacks mCallbacks;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.login);

// Initialize Firebase App Check

FirebaseApp.initializeApp(this);

FirebaseAppCheck firebaseAppCheck = FirebaseAppCheck.getInstance();

firebaseAppCheck.installAppCheckProviderFactory(

PlayIntegrityAppCheckProviderFactory.getInstance()

);

mAuth = FirebaseAuth.getInstance();

edtPhone = findViewById(R.id.idEdtPhoneNumber);

edtOTP = findViewById(R.id.idEdtOtp);

verifyOTPBtn = findViewById(R.id.idBtnVerify);

generateOTPBtn = findViewById(R.id.idBtnGetOtp);

generateOTPBtn.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View v) {

String phoneNumber = edtPhone.getText().toString().trim();

if (TextUtils.isEmpty(phoneNumber)) {

Toast.makeText(Login.this, "Please enter a valid phone number.", Toast.LENGTH\_SHORT).show();

} else {

String formattedNumber = "+91" + phoneNumber;

sendVerificationCode(formattedNumber);

}

}

});

verifyOTPBtn.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View v) {

String otp = edtOTP.getText().toString().trim();

if (TextUtils.isEmpty(otp)) {

Toast.makeText(Login.this, "Please enter OTP", Toast.LENGTH\_SHORT).show();

} else {

verifyPhoneNumberWithCode(mVerificationId, otp);

}

}

});

mCallbacks = new PhoneAuthProvider.OnVerificationStateChangedCallbacks() {

@Override

public void onVerificationCompleted(@NonNull PhoneAuthCredential credential) {

Log.d(TAG, "onVerificationCompleted:" + credential);

String code = credential.getSmsCode();

if (code != null) {

edtOTP.setText(code);

verifyPhoneNumberWithCode(mVerificationId, code);

} else {

signInWithPhoneAuthCredential(credential);

}

}

@Override

public void onVerificationFailed(@NonNull FirebaseException e) {

Log.w(TAG, "onVerificationFailed", e);

if (e instanceof FirebaseAuthInvalidCredentialsException) {

Toast.makeText(Login.this, "Invalid request. " + e.getMessage(), Toast.LENGTH\_LONG).show();

} else if (e instanceof FirebaseTooManyRequestsException) {

Toast.makeText(Login.this, "The SMS quota for the project has been exceeded.", Toast.LENGTH\_LONG).show();

} else if (e instanceof FirebaseAuthMissingActivityForRecaptchaException) {

Toast.makeText(Login.this, "reCAPTCHA verification attempted with null Activity.", Toast.LENGTH\_LONG).show();

} else {

Toast.makeText(Login.this, "Verification failed. " + e.getMessage(), Toast.LENGTH\_LONG).show();

}

}

@Override

public void onCodeSent(@NonNull String verificationId,

@NonNull PhoneAuthProvider.ForceResendingToken token) {

Log.d(TAG, "onCodeSent: verificationId = " + verificationId);

mVerificationId = verificationId;

mResendToken = token;

Toast.makeText(Login.this, "OTP sent successfully", Toast.LENGTH\_SHORT).show();

}

};

}

@Override

public void onStart() {

super.onStart();

FirebaseUser currentUser = mAuth.getCurrentUser();

updateUI(currentUser);

}

private void sendVerificationCode(final String number) {

Log.d(TAG, "sendVerificationCode: number = " + number);

enteredPhoneNumber = number;

DatabaseReference usersRef = FirebaseDatabase.getInstance().getReference("users");

ValueEventListener phoneListener = new ValueEventListener() {

@Override

public void onDataChange(@NonNull DataSnapshot dataSnapshot) {

boolean isNumberRegistered = false;

for (DataSnapshot userSnapshot : dataSnapshot.getChildren()) {

String phone = userSnapshot.child("phone").getValue(String.class);

if (number.equals(phone)) {

isNumberRegistered = true;

break;

}

}

if (isNumberRegistered) {

startPhoneNumberVerification(number);

} else {

Toast.makeText(Login.this, "Phone number not registered. Please proceed with SignUp.", Toast.LENGTH\_SHORT).show();

}

}

@Override

public void onCancelled(@NonNull DatabaseError databaseError) {

Toast.makeText(Login.this, "Database error: " + databaseError.getMessage(), Toast.LENGTH\_SHORT).show();

}

};

usersRef.addListenerForSingleValueEvent(phoneListener);

}

private void startPhoneNumberVerification(String phoneNumber) {

Log.d(TAG, "startPhoneNumberVerification: phoneNumber = " + phoneNumber);

PhoneAuthOptions options = PhoneAuthOptions.newBuilder(mAuth)

.setPhoneNumber(phoneNumber)

.setTimeout(60L, TimeUnit.SECONDS)

.setActivity(this)

.setCallbacks(mCallbacks)

.build();

PhoneAuthProvider.verifyPhoneNumber(options);

}

private void verifyPhoneNumberWithCode(String verificationId, String code) {

Log.d(TAG, "verifyPhoneNumberWithCode: verificationId = " + verificationId + ", code = " + code);

PhoneAuthCredential credential = PhoneAuthProvider.getCredential(verificationId, code);

signInWithPhoneAuthCredential(credential);

}

private void resendVerificationCode(String phoneNumber, PhoneAuthProvider.ForceResendingToken token) {

PhoneAuthOptions options = PhoneAuthOptions.newBuilder(mAuth)

.setPhoneNumber(phoneNumber)

.setTimeout(60L, TimeUnit.SECONDS)

.setActivity(this)

.setCallbacks(mCallbacks)

.setForceResendingToken(token)

.build();

PhoneAuthProvider.verifyPhoneNumber(options);

}

private void signInWithPhoneAuthCredential(PhoneAuthCredential credential) {

Log.d(TAG, "signInWithPhoneAuthCredential");

mAuth.signInWithCredential(credential)

.addOnCompleteListener(this, new OnCompleteListener<AuthResult>() {

@Override

public void onComplete(@NonNull Task<AuthResult> task) {

if (task.isSuccessful()) {

Log.d(TAG, "signInWithCredential:success");

FirebaseUser user = task.getResult().getUser();

fetchUserNameAndNavigate(user);

} else {

Log.w(TAG, "signInWithCredential:failure", task.getException());

if (task.getException() instanceof FirebaseAuthInvalidCredentialsException) {

Toast.makeText(Login.this, "The verification code entered was invalid.", Toast.LENGTH\_LONG).show();

}

updateUI(null);

}

}

});

}

private void fetchUserNameAndNavigate(FirebaseUser user) {

DatabaseReference usersRef = FirebaseDatabase.getInstance().getReference("users");

usersRef.orderByChild("phone").equalTo(enteredPhoneNumber).addListenerForSingleValueEvent(new ValueEventListener() {

@Override

public void onDataChange(@NonNull DataSnapshot dataSnapshot) {

if (dataSnapshot.exists()) {

for (DataSnapshot userSnapshot : dataSnapshot.getChildren()) {

String name = userSnapshot.child("name").getValue(String.class);

// Pass the user's name to the homepage

Intent i = new Intent(Login.this, Home\_page.class);

i.putExtra("USER\_NAME", name);

startActivity(i);

finish();

}

}

}

@Override

public void onCancelled(@NonNull DatabaseError databaseError) {

Toast.makeText(Login.this, "Database error: " + databaseError.getMessage(), Toast.LENGTH\_SHORT).show();

}

});

}

private void updateUI(FirebaseUser user) {

if (user != null) {

// User is signed in

Toast.makeText(Login.this, "Signed in successfully", Toast.LENGTH\_SHORT).show();

// Navigate to home page or update the UI accordingly

} else {

// User is signed out

Toast.makeText(Login.this, "Signed out", Toast.LENGTH\_SHORT).show();

}

}

}

**Layout:**

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

<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"

xmlns:app="http://schemas.android.com/apk/res-auto"

xmlns:tools="http://schemas.android.com/tools"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

android:background="#87CEEB">

<LinearLayout

android:layout\_width="374dp"

android:layout\_height="wrap\_content"

android:layout\_gravity="center\_horizontal"

android:background="@drawable/rounded\_rectangle"

android:orientation="vertical"

android:elevation="8dp"

android:outlineAmbientShadowColor="@color/royalblue"

android:outlineProvider="bounds"

android:padding="20dp"

app:layout\_constraintBottom\_toBottomOf="parent"

app:layout\_constraintEnd\_toEndOf="parent"

app:layout\_constraintStart\_toStartOf="parent"

app:layout\_constraintTop\_toTopOf="parent">

<EditText

android:id="@+id/idEdtPhoneNumber"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:layout\_centerHorizontal="true"

android:layout\_margin="10dp"

android:background="@color/white"

android:hint="Enter your phone"

android:importantForAutofill="no"

android:inputType="phone" />

<Button

android:id="@+id/idBtnGetOtp"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:layout\_below="@id/idEdtPhoneNumber"

android:layout\_margin="10dp"

android:text="Get OTP"

android:textAllCaps="false"

android:backgroundTint="@color/royalblue"

android:textColor="@color/black"/>

<EditText

android:id="@+id/idEdtOtp"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:layout\_below="@id/idBtnGetOtp"

android:layout\_margin="10dp"

android:background="@color/white"

android:hint="Enter OTP"

android:importantForAutofill="no"

android:inputType="phone" />

<Button

android:id="@+id/idBtnVerify"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:layout\_below="@id/idEdtOtp"

android:layout\_margin="10dp"

android:text="Verify OTP"

android:textAllCaps="false"

android:backgroundTint="@color/royalblue"

android:textColor="@color/black"/>

</LinearLayout>

</androidx.constraintlayout.widget.ConstraintLayout>

**Output:**

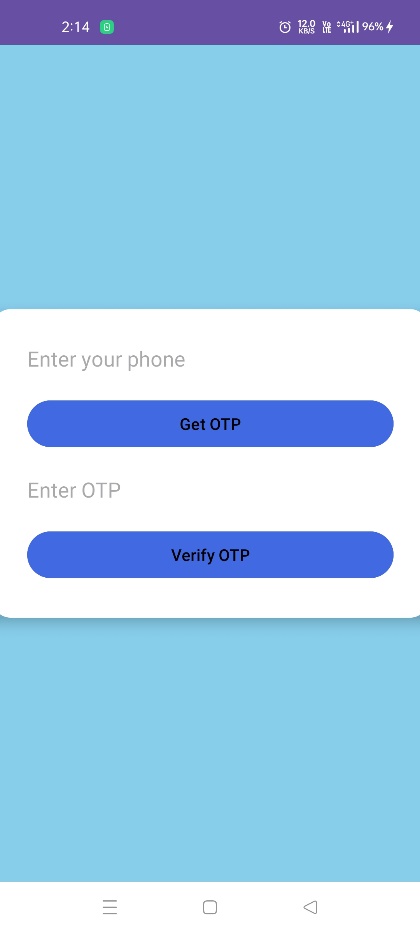


Figure 33: Output 4

**Home Page:**

**Class:**

import static android.content.ContentValues.TAG;

import androidx.annotation.NonNull;

import androidx.appcompat.app.AppCompatActivity;

import androidx.appcompat.widget.Toolbar;

import android.content.Intent;

import android.net.Uri;

import android.os.Bundle;

import android.util.Log;

import android.view.Menu;

import android.view.MenuInflater;

import android.view.MenuItem;

import android.view.View;

import android.widget.Button;

import android.widget.TextView;

import android.widget.Toast;

import com.google.firebase.auth.FirebaseAuth;

import com.google.firebase.database.DataSnapshot;

import com.google.firebase.database.DatabaseError;

import com.google.firebase.database.DatabaseReference;

import com.google.firebase.database.FirebaseDatabase;

import com.google.firebase.database.Query;

import com.google.firebase.database.ValueEventListener;

public class Home\_page extends AppCompatActivity {

private FirebaseAuth firebaseAuth;

private TextView state, category, rating, place1, place2, place3, accommodation, travel\_mode;

private String user\_name;

private TextView un;

private Button gen\_plan\_form, livesupport, allplans, userprofile;

private Generate\_plans plan;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_home\_page);

Toolbar toolbar = findViewById(R.id.toolbar);

setSupportActionBar(toolbar);

firebaseAuth = FirebaseAuth.getInstance();

state = findViewById(R.id.State);

category = findViewById(R.id.Category);

rating = findViewById(R.id.Rating);

place1 = findViewById(R.id.Place1);

place2 = findViewById(R.id.Place2);

place3 = findViewById(R.id.Place3);

accommodation = findViewById(R.id.Accommodation);

travel\_mode = findViewById(R.id.Travel\_Mode);

gen\_plan\_form = findViewById(R.id.btnRecomGen);

livesupport = findViewById(R.id.btnLiveSupport);

userprofile = findViewById(R.id.btnUserProfile);

un = findViewById(R.id.user\_name);

user\_name = getIntent().getStringExtra("USER\_NAME");

if (user\_name != null) {

// Update your UI with the user's name

un.setText("Welcome, " + user\_name + "!");

}

else {

Log.e(TAG, "user\_name is null in Home\_page");

Toast.makeText(this, "Error: No user found", Toast.LENGTH\_SHORT).show();

finish();

return;

}

fetchLatestPlan();

gen\_plan\_form.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View v) {

Intent intent = new Intent(Home\_page.this, Generate\_Plan\_form.class);

intent.putExtra("USER\_NAME",user\_name);

startActivity(intent);

}

});

livesupport.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View v) {

Intent intent = new Intent(Home\_page.this, Live\_Support.class);

startActivity(intent);

}

});

userprofile.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View v) {

Intent intent = new Intent(Home\_page.this, User\_Profile.class);

intent.putExtra("USER\_NAME",user\_name);

startActivity(intent);

}

});

place1.setOnClickListener(v -> {

navigateToPlace(place1.getText().toString(), state.getText().toString());

});

place2.setOnClickListener(v -> {

navigateToPlace(place2.getText().toString(), state.getText().toString());

});

place3.setOnClickListener(v -> {

navigateToPlace(place3.getText().toString(), state.getText().toString());

});

}

public boolean onCreateOptionsMenu(Menu menu){

MenuInflater inflater = getMenuInflater();

inflater.inflate(R.menu.home\_menu, menu);

return true;

}

public boolean onOptionsItemSelected(MenuItem item){

if (item.getItemId() == R.id.menu\_logout) {

logout();

return true;

}

return super.onOptionsItemSelected(item);

}

private void logout(){

firebaseAuth.signOut();

Intent intent = new Intent(Home\_page.this, MainActivity.class);

startActivity(intent);

finish();

}

private void fetchLatestPlan() {

DatabaseReference plansRef = FirebaseDatabase.getInstance().getReference("plans").child(user\_name);

plansRef.orderByKey().limitToLast(1).addListenerForSingleValueEvent(new ValueEventListener() {

@Override

public void onDataChange(@NonNull DataSnapshot dataSnapshot) {

if (dataSnapshot.exists()) {

for (DataSnapshot snapshot : dataSnapshot.getChildren()) {

Generate\_plans plan = snapshot.getValue(Generate\_plans.class);

if (plan != null) {

state.setText(String.format("State: %s", plan.getState()));

category.setText(String.format("Category: %s", plan.getCategory()));

rating.setText(String.format("Rating: %s", plan.getRating()));

place1.setText(String.format("Place 1: %s", plan.getPlace1()));

place2.setText(String.format("Place 2: %s", plan.getPlace2()));

place3.setText(String.format("Place 3: %s", plan.getPlace3()));

accommodation.setText(String.format("Accommodation: %s", plan.getAccommodation()));

travel\_mode.setText(String.format("Hotels: %s", plan.getTravelMode()));

}

}

} else {

Toast.makeText(Home\_page.this, "No plans found.", Toast.LENGTH\_SHORT).show();

}

}

@Override

public void onCancelled(@NonNull DatabaseError databaseError) {

Log.w(TAG, "fetchLatestPlan:onCancelled", databaseError.toException());

Toast.makeText(Home\_page.this, "Failed to load your last plan.", Toast.LENGTH\_SHORT).show();

}

});

}

private void navigateToPlace(String place,String state) {

String travelMode = travel\_mode.getText().toString();

String mode = "d";

switch (travelMode) {

case "bus":

mode = "transit";

break;

case "car":

mode = "driving";

break;

case "train":

mode = "transit";

break;

}

Log.d("travel preference",travelMode);

Uri gmmIntentUri = Uri.parse("google.navigation:q=" + Uri.encode(place+","+state+",India") + "&mode=" + mode);

Intent mapIntent = new Intent(Intent.ACTION\_VIEW, gmmIntentUri);

mapIntent.setPackage("com.google.android.apps.maps");

if (mapIntent.resolveActivity(getPackageManager()) != null) {

startActivity(mapIntent);

}

}

}

**Layout:**

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

<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"

xmlns:app="http://schemas.android.com/apk/res-auto"

xmlns:tools="http://schemas.android.com/tools"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

android:background="#87CEEB"

tools:context=".Home\_page">

<!-- Top Navigation -->

<androidx.appcompat.widget.Toolbar

android:id="@+id/toolbar"

android:layout\_width="match\_parent"

android:layout\_height="?attr/actionBarSize"

android:background="@color/royalblue"

app:title="Home"

app:titleTextColor="@android:color/white"

app:layout\_constraintTop\_toTopOf="parent"/>

<!--bottom Navigation-->

<LinearLayout

android:id="@+id/bottomNavigation"

android:layout\_width="0dp"

android:layout\_height="wrap\_content"

android:orientation="horizontal"

android:background="@color/royalblue"

app:layout\_constraintBottom\_toBottomOf="parent"

app:layout\_constraintStart\_toStartOf="parent"

app:layout\_constraintEnd\_toEndOf="parent"

android:padding="8dp">

<Button

android:id="@+id/btnRecomGen"

android:layout\_width="0dp"

android:layout\_height="wrap\_content"

android:backgroundTint="#000080"

android:layout\_weight="1"

android:text="Create Plan"

android:textColor="@color/white" />

<Button

android:id="@+id/btnLiveSupport"

android:layout\_width="0dp"

android:layout\_height="wrap\_content"

android:backgroundTint="#000080"

android:layout\_weight="1"

android:text="Live Support"

android:textColor="@color/white"/>

<Button

android:id="@+id/btnUserProfile"

android:layout\_width="0dp"

android:layout\_height="wrap\_content"

android:backgroundTint="#000080"

android:layout\_weight="1"

android:text="User Profile"

android:textColor="@color/white" />

</LinearLayout>

<TextView

android:id="@+id/user\_name"

android:layout\_width="193dp"

android:layout\_height="35dp"

android:layout\_marginTop="20dp"

android:text="Username"

android:textAlignment="center"

android:textSize="24sp"

android:textStyle="bold"

app:layout\_constraintBottom\_toTopOf="@+id/bottomNavigation"

app:layout\_constraintEnd\_toEndOf="parent"

app:layout\_constraintStart\_toStartOf="parent"

app:layout\_constraintTop\_toBottomOf="@+id/toolbar"

app:layout\_constraintVertical\_bias="0.0" />

<LinearLayout

android:id="@+id/hpplanll"

android:layout\_width="374dp"

android:layout\_height="wrap\_content"

android:layout\_gravity="center\_horizontal"

android:background="@drawable/rounded\_rectangle"

android:elevation="8dp"

android:orientation="vertical"

android:outlineAmbientShadowColor="@color/royalblue"

android:outlineProvider="bounds"

android:padding="20dp"

app:layout\_constraintBottom\_toBottomOf="parent"

app:layout\_constraintEnd\_toEndOf="parent"

app:layout\_constraintHorizontal\_bias="0.567"

app:layout\_constraintStart\_toStartOf="parent"

app:layout\_constraintTop\_toTopOf="parent"

app:layout\_constraintVertical\_bias="0.547">

<TextView

android:id="@+id/textView11"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:text="Your Last Plan"

android:textAlignment="center"

android:textSize="20sp"

android:background="@color/royalblue"

android:textColor="@color/white"/>

<Space

android:layout\_width="match\_parent"

android:layout\_height="10sp" />

<TextView

android:id="@+id/State"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:text="State"

android:textSize="20sp" />

<Space

android:layout\_width="match\_parent"

android:layout\_height="10sp" />

<LinearLayout

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

android:orientation="horizontal">

<TextView

android:id="@+id/Category"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:layout\_weight="1"

android:text="Category"

android:textSize="20sp" />

<TextView

android:id="@+id/Rating"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:layout\_weight="1"

android:text="Rating"

android:textSize="20sp" />

</LinearLayout>

<Space

android:layout\_width="match\_parent"

android:layout\_height="10sp" />

<TextView

android:id="@+id/Place1"

android:layout\_width="wrap\_content"

android:layout\_height="48dp"

android:layout\_marginStart="10dp"

android:clickable="true"

android:text="Place1"

android:textSize="20sp" />

<TextView

android:id="@+id/Place2"

android:layout\_width="wrap\_content"

android:layout\_height="48dp"

android:layout\_marginStart="10dp"

android:clickable="true"

android:text="Place2"

android:textSize="20sp" />

<TextView

android:id="@+id/Place3"

android:layout\_width="wrap\_content"

android:layout\_height="48dp"

android:layout\_marginStart="10dp"

android:clickable="true"

android:text="Place3"

android:textSize="20sp" />

<Space

android:layout\_width="match\_parent"

android:layout\_height="10sp" />

<TextView

android:id="@+id/Accommodation"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:text="Accommodation"

android:textSize="20sp" />

<Space

android:layout\_width="match\_parent"

android:layout\_height="10sp" />

<TextView

android:id="@+id/Travel\_Mode"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:text="Travel\_Mode"

android:textSize="20sp" />

</LinearLayout>

</androidx.constraintlayout.widget.ConstraintLayout>

**Output:**

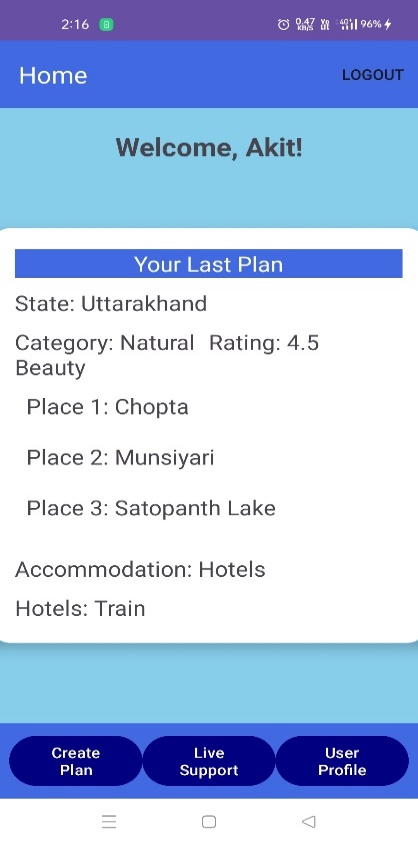


Figure 34: Output 5

**Generate Plan form:**

**Class:**

import android.content.Intent;

import android.os.Bundle;

import android.util.Log;

import android.view.Menu;

import android.view.MenuInflater;

import android.view.MenuItem;

import android.widget.Button;

import android.widget.RadioButton;

import android.widget.Spinner;

import org.tensorflow.lite.Interpreter;

import androidx.activity.OnBackPressedCallback;

import androidx.appcompat.app.AppCompatActivity;

import androidx.appcompat.widget.Toolbar;

import android.widget.ArrayAdapter;

import android.widget.Toast;

import com.google.firebase.auth.FirebaseAuth;

import com.google.firebase.database.DatabaseReference;

import com.google.firebase.database.FirebaseDatabase;

import java.io.FileInputStream;

import java.io.IOException;

import java.nio.MappedByteBuffer;

import java.nio.channels.FileChannel;

import java.util.Arrays;

import java.util.HashMap;

import java.util.Map;

public class Generate\_Plan\_form extends AppCompatActivity {

private RadioButton radioButton1, radioButton2;

private Spinner spinner1, spinner2, spinner3, spinner4;

private FirebaseAuth firebaseAuth;

private Interpreter interpreter;

private TensorFlowModelHelper modelHelper;

private String user\_name;

private FirebaseDatabase db;

private DatabaseReference reference;

private Button generateButton;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_generate\_plan\_form);

Toolbar toolbar = findViewById(R.id.toolbar);

setSupportActionBar(toolbar);

firebaseAuth = FirebaseAuth.getInstance();

user\_name = getIntent().getStringExtra("USER\_NAME");

radioButton1 = findViewById(R.id.radioButton);

radioButton2 = findViewById(R.id.radioButton2);

spinner1 = findViewById(R.id.spinner);

spinner2 = findViewById(R.id.spinner2);

spinner3 = findViewById(R.id.spinner3);

spinner4 = findViewById(R.id.spinner4);

generateButton = findViewById(R.id.button3);

radioButton1.setOnClickListener(v -> updateCategories(R.array.categories1));

radioButton2.setOnClickListener(v -> updateCategories(R.array.categories2));

generateButton.setOnClickListener(v -> generatePlan());

try {

interpreter = new Interpreter(loadModelFile());

} catch (IOException e) {

e.printStackTrace();

}

getOnBackPressedDispatcher().addCallback(this, new OnBackPressedCallback(true) {

@Override

public void handleOnBackPressed() {

Intent intent = new Intent(Generate\_Plan\_form.this, Home\_page.class);

intent.putExtra("USER\_NAME",user\_name);

startActivity(intent);

finish();

}

});

}

private MappedByteBuffer loadModelFile() throws IOException {

FileInputStream fileInputStream = new FileInputStream(getAssets().openFd("my\_model.tflite").getFileDescriptor());

FileChannel fileChannel = fileInputStream.getChannel();

long startOffset = getAssets().openFd("my\_model.tflite").getStartOffset();

long declaredLength = getAssets().openFd("my\_model.tflite").getDeclaredLength();

return fileChannel.map(FileChannel.MapMode.READ\_ONLY, startOffset, declaredLength);

}

private void updateCategories(int arrayResId) {

ArrayAdapter<CharSequence> adapter = ArrayAdapter.createFromResource(this,

arrayResId, android.R.layout.simple\_spinner\_item);

adapter.setDropDownViewResource(android.R.layout.simple\_spinner\_dropdown\_item);

spinner1.setAdapter(adapter);

}

private void generatePlan() {

final String[] place1 = {"Akshardham Temple", "Ambience Mall", "Appu Ghar Express", "Askot Sanctuary",

"Auli", "Badrinath", "Bangla Sahib Market", "Bhimtal lake", "Bhulla Lake", "Chakrata", "Chamoli",

"Chandni Chowk", "Chhatarpur Temple", "Chopta", "Connaught Place", "Corbett National Park", "DLF CyberHub",

"DLF Emporio", "Dehradun", "Delhi Haunted House", "Delhi Ridge Forest", "Dilli Haat INA", "Dronagiri",

"Gandhi Smriti", "George Everest Peak", "Gurudwara Bangla Sahib", "Haldwani", "Haridwar", "Hauz Khas Village",

"Humayun’s Tomb", "ISKCON Temple", "India Gate", "India Habitat Centre", "India International Centre", "Jageshwar",

"Jama Masjid", "Janpath Market", "Jantar Mantar", "Joshimath", "Kafni Glacier", "Kainchi Dham", "Kausani",

"Kedarnath", "Khari Baoli", "Lakshman Jhula", "Lansdowne", "Lotus Temple", "Madame Tussauds", "Mughal Gardens",

"Mukteshwar", "Munsiyari", "Mussoorie", "Nag Tibba", "Nainital", "National Gallery of Modern Art",

"National Museum", "National Zoological Park", "Neelkanth Mahadev Temple", "Neer Garh Waterfall",

"Nizamuddin Dargah", "Palika Bazaar", "Panch Kedar", "Pauri", "Pindari Glacier", "Pithoragarh",

"Purana Qila", "Qutub Minar", "Raj Ghat", "Rajaji Wildlife Sanctuary", "Rajouri Garden Market",

"Rani Jheel", "Red Fort", "Roopkund", "Sacred Heart Cathedral", "Sahastradhara", "Sarojini Nagar Market",

"Sattal", "Select Citywalk", "Talkatora Garden", "Tapovan", "Tehri", "Tughlaqabad Fort",

"Valley of Flowers National Park"};

final String[] place2 = {

"Akshardham Temple", "Appu Ghar Express", "Badrinath", "Bangla Sahib Market",

"Bhavishya Badri", "Bhimtal lake", "Bhulla Lake", "Binsar", "Chakrata", "Chamoli",

"Chhatarpur Temple", "Connaught Place", "DLF Emporio", "Dehradun",

"Delhi Ridge Biodiversity Park", "Delhi Ridge Forest", "Delhi-6", "Devprayag",

"Dhanaulti", "Dilli Haat INA", "Dronagiri", "Gangotri", "George Everest Peak",

"Gomukh", "Gurudwara Bangla Sahib", "Hanuman Temple", "ISKCON Temple", "Jageshwar",

"Janpath Market", "Jharipani Falls", "Kafni Glacier", "Kainchi Dham", "Kanatal",

"Kedarnath", "Khan Market", "Khirsu", "Kingdom of Dreams", "Lakshman Jhula",

"Lansdowne", "Laxminarayan Temple", "Mukteshwar", "Munsiyari", "Mussoorie",

"Nag Tibba", "National Gallery of Modern Art", "National Rail Museum",

"Neelkanth Mahadev Temple", "Neer Garh Waterfall", "Pacific Mall", "Palika Bazaar",

"Panch Kedar", "Pangot And Kilbury Bird Sanctuary", "Pauri", "Pithoragarh",

"Rajaji Wildlife Sanctuary", "Rajouri Garden Market", "Rani Jheel", "Ranikhet",

"Roopkund", "Roorkee", "Sacred Heart Cathedral", "Sahastradhara",

"Sarojini Nagar Market", "Sattal", "Select Citywalk", "St. James’ Church",

"Talkatora Garden", "Tapovan", "Tungnath Temple", "Valley of Flowers National Park",

"Art Spice Gallery", "Delhi Haunted House", "Feroz Shah Kotla Fort",

"Fun N Food Village", "Humayun’s Tomb", "India Gate", "Lodi Art District",

"Madame Tussauds", "Nehru Planetarium", "Okhla Bird Sanctuary", "Purana Qila",

"Qutub Minar", "Raj Ghat", "Rashtrapati Bhavan", "Rishikesh", "Tughlaqabad Fort"

};

final String[] place3 = {

"Raj Ghat", "Almora", "Badrinath", "Bhavishya Badri", "Bhim Ghasutri", "Bhowali", "Bhulla Lake",

"Binsar", "Binsar Wildlife Sanctuary", "Chakrata", "Chamoli", "Connaught Place", "Crafts Museum",

"DLF Emporio", "Dayara Bugyal", "Delhi Ridge Biodiversity Park", "Delhi-6", "Devprayag", "Dhanaulti",

"Dharali", "Dronagiri", "Ganga Aarti at Triveni Ghat", "Gangotri", "Gomukh", "Greater Kailash M-Block Market",

"Gurudwara Bangla Sahib", "Gurudwara Sis Ganj Sahib", "Gwaldam", "Hanuman Temple", "Hemkund Sahib",

"ISKCON Temple", "Jageshwar", "Janpath Market", "Jhandewalan Temple", "Jharipani Falls", "Kafni Glacier", "Kainchi Dham",

"Kanatal", "Karnaprayag", "Karol Bagh Market", "Kausani", "Kempty Falls", "Khan Market", "Khirsu",

"Kingdom of Dreams", "Kunjapuri Temple", "Lakshman Jhula", "Lansdowne", "Laxminarayan Temple", "Lodhi Gardens",

"Munsiyari", "Nag Tibba", "National Rail Museum", "Neer Garh Waterfall", "Pacific Mall", "Palika Bazaar",

"Panch Kedar", "Pangot And Kilbury Bird Sanctuary", "Raj Ghat", "Rani Jheel", "Roopkund",

"Roorkee", "Sahastradhara", "Sarojini Nagar Market", "Satopanth Lake", "Sattal", "St. James’ Church",

"St. Stephen’s Church", "Surkanda Devi Temple", "Swarg Ashram", "Talkatora Garden", "Tapkeshwar Temple",

"Tapovan", "The Great India Place", "Tungnath Temple", "Vashishta Gufa", "Worlds of Wonder", "Adventure Island",

"Agrasen ki Baoli", "Benog Wildlife Sanctuary", "Delhi Haunted House", "Delhi War Cemetery", "Feroz Shah Kotla Fort",

"Fun N Food Village", "Gaumukh Glacier", "Kathputli Colony", "Purana Qila", "Qutub Minar", "Rashtrapati Bhavan",

"Shankar’s International Dolls Museum", "Surajkund Mela", "Tughlaqabad Fort", "Yamuna biodiversity Park"

};

String State = radioButton1.isChecked() ? radioButton1.getText().toString() : radioButton2.getText().toString();

String Category = spinner1.getSelectedItem().toString();

String spinner2Value = spinner2.getSelectedItem().toString();

float Rating = Float.parseFloat(spinner2Value);

String Accomod = spinner3.getSelectedItem().toString();

String TravelPf = spinner4.getSelectedItem().toString();

int encodedState = encodeState(State);

int encodedCategory = encodeCategory(Category);

int encodedRating = encodeRating(Rating);

float[][] input = new float[1][3];

input[0][0] = encodedState;

input[0][1] = encodedCategory;

input[0][2] = encodedRating;

float[][] output1 = new float[1][83];

float[][] output2 = new float[1][86];

float[][] output3 = new float[1][93];

Map<Integer, Object> outputs = new HashMap<>();

outputs.put(0, output1);

outputs.put(1, output2);

outputs.put(2, output3);

interpreter.runForMultipleInputsOutputs(new Object[]{input}, outputs);

String place1Name = decodePlace(place1, output1[0]);

String place2Name = decodePlace(place2, output2[0]);

String place3Name = decodePlace(place3, output3[0]);

String[] results = {place1Name,place2Name,place3Name};

Intent intent = new Intent(Generate\_Plan\_form.this, Preview\_Plan.class);

intent.putExtra("KEY1", State);

intent.putExtra("KEY2", Category);

intent.putExtra("KEY3", spinner2Value);

intent.putExtra("KEY4", Accomod);

intent.putExtra("KEY5", TravelPf);

intent.putExtra("KEY6", results);

intent.putExtra("USER\_NAME",user\_name);

startActivity(intent);

}

private String decodePlace(String[] places, float[] output) {

int maxIndex = 0;

for (int i = 1; i < output.length; i++) {

if (output[i] > output[maxIndex]) {

maxIndex = i;

}

}

return places[maxIndex];

}

public static int encodeState(String state) {

switch (state) {

case "Delhi":

return 0;

case "Uttarakhand":

return 1;

}

return 0;

}

public static int encodeRating(float rating) {

if (rating == 4.0f) {

return 0;

} else if (rating == 4.5f) {

return 1;

} else {

return 0;

}

}

public static int encodeCategory(String category) {

switch (category) {

case "Adventure sports":

return 0;

case "Culture":

return 1;

case "Entertainment":

return 2;

case "Hill Station":

return 3;

case "Historical":

return 4;

case "Mall":

return 5;

case "Market":

return 6;

case "Museum":

return 7;

case "Natural Beauty":

return 8;

case "Religious":

return 9;

case "Wildlife":

return 10;

}

return 0;

}

public boolean onCreateOptionsMenu(Menu menu){

MenuInflater inflater = getMenuInflater();

inflater.inflate(R.menu.home\_menu, menu);

return true;

}

public boolean onOptionsItemSelected(MenuItem item){

if (item.getItemId() == R.id.menu\_logout) {

logout();

return true;

}

return super.onOptionsItemSelected(item);

}

private void logout(){

firebaseAuth.signOut();

Intent intent = new Intent(Generate\_Plan\_form.this, MainActivity.class);

startActivity(intent);

finish();

}

}

**Layout:**

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

<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"

xmlns:app="http://schemas.android.com/apk/res-auto"

xmlns:tools="http://schemas.android.com/tools"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

android:background="#87CEEB"

tools:context=".Generate\_Plan\_form">

<androidx.appcompat.widget.Toolbar

android:id="@+id/toolbar"

android:layout\_width="match\_parent"

android:layout\_height="?attr/actionBarSize"

android:background="@color/royalblue"

app:title="Form"

app:titleTextColor="@android:color/white"

app:layout\_constraintTop\_toTopOf="parent"/>

<LinearLayout

android:id="@+id/linearLayout2"

android:layout\_width="374dp"

android:layout\_height="wrap\_content"

android:layout\_gravity="center\_horizontal"

android:background="@drawable/rounded\_rectangle"

android:elevation="8dp"

android:orientation="vertical"

android:outlineAmbientShadowColor="@color/royalblue"

android:outlineProvider="bounds"

android:padding="20dp"

app:layout\_constraintBottom\_toBottomOf="parent"

app:layout\_constraintEnd\_toEndOf="parent"

app:layout\_constraintStart\_toStartOf="parent"

app:layout\_constraintTop\_toTopOf="parent">

<TextView

android:id="@+id/textView4"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="State"

android:textColor="@color/black"

android:textSize="20sp"

tools:layout\_editor\_absoluteX="33dp"

tools:layout\_editor\_absoluteY="56dp" />

<Space

android:layout\_width="match\_parent"

android:layout\_height="10dp" />

<RadioGroup

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content">

<RadioButton

android:id="@+id/radioButton"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Delhi"

android:textColor="@color/black"

android:textSize="20sp"

tools:layout\_editor\_absoluteX="105dp"

tools:layout\_editor\_absoluteY="45dp" />

<RadioButton

android:id="@+id/radioButton2"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Uttarakhand"

android:textColor="@color/black"

android:textSize="20sp"

tools:layout\_editor\_absoluteX="105dp"

tools:layout\_editor\_absoluteY="104dp" />

</RadioGroup>

<Space

android:layout\_width="match\_parent"

android:layout\_height="10dp" />

<TextView

android:id="@+id/textView5"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Category of Tourism Spots:"

android:textSize="20sp"

android:textColor="@color/black"

tools:layout\_editor\_absoluteX="7dp"

tools:layout\_editor\_absoluteY="190dp" />

<Space

android:layout\_width="match\_parent"

android:layout\_height="10dp" />

<LinearLayout

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

android:orientation="horizontal">

<Spinner

android:id="@+id/spinner"

android:layout\_width="wrap\_content"

android:layout\_height="48dp"

android:textColor="@color/black"

android:entries="@array/categories1"

tools:layout\_editor\_absoluteX="261dp"

tools:layout\_editor\_absoluteY="237dp" />

<Space

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_weight="1" />

<Spinner

android:id="@+id/spinner2"

android:layout\_width="wrap\_content"

android:layout\_height="48dp"

android:layout\_weight="1"

android:textColor="@color/black"

android:entries="@array/ratings"/>

</LinearLayout>

<Space

android:layout\_width="match\_parent"

android:layout\_height="10dp" />

<TextView

android:id="@+id/textView6"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Accomodation Preference:"

android:textSize="20sp"

android:textColor="@color/black"

tools:layout\_editor\_absoluteX="7dp"

tools:layout\_editor\_absoluteY="282dp" />

<Space

android:layout\_width="match\_parent"

android:layout\_height="10dp" />

<Spinner

android:id="@+id/spinner3"

android:layout\_width="wrap\_content"

android:layout\_height="48dp"

android:layout\_weight="1"

android:textColor="@color/black"

android:entries="@array/Accomodation\_types"/>

<Space

android:layout\_width="match\_parent"

android:layout\_height="10dp" />

<TextView

android:id="@+id/textView7"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Travel Mode Preference:"

android:textSize="20sp"

android:textColor="@color/black"

tools:layout\_editor\_absoluteX="7dp"

tools:layout\_editor\_absoluteY="345dp" />

<Space

android:layout\_width="match\_parent"

android:layout\_height="10dp" />

<Spinner

android:id="@+id/spinner4"

android:layout\_width="wrap\_content"

android:layout\_height="48dp"

android:layout\_weight="1"

android:textColor="@color/black"

android:entries="@array/Travel\_Modes"/>

</LinearLayout>

<Button

android:id="@+id/button3"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_marginTop="8dp"

android:text="Generate"

android:textColor="@color/white"

android:backgroundTint="#000080"

app:layout\_constraintBottom\_toBottomOf="parent"

app:layout\_constraintEnd\_toEndOf="parent"

app:layout\_constraintStart\_toStartOf="parent"

app:layout\_constraintTop\_toBottomOf="@+id/linearLayout2"

app:layout\_constraintVertical\_bias="0.2" />

</androidx.constraintlayout.widget.ConstraintLayout>

**Output:**

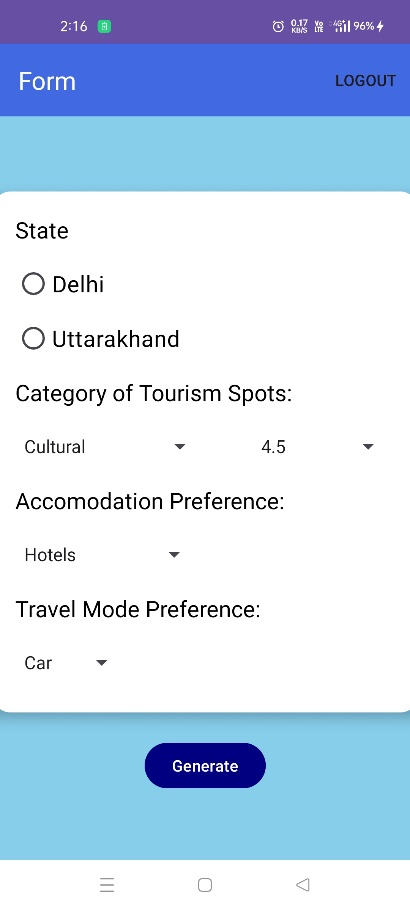


Figure 35: Output 6

**Preview Plan:**

**Class:**

import androidx.appcompat.app.AppCompatActivity;

import androidx.appcompat.widget.Toolbar;

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.TextView;

import android.widget.Toast;

import androidx.activity.OnBackPressedCallback;

import com.google.firebase.database.DatabaseReference;

import com.google.firebase.database.FirebaseDatabase;

public class Preview\_Plan extends AppCompatActivity {

private TextView state, category, rating, place1, place2, place3, accommodation, travel\_mode;

private Button savebtn;

private Intent intent;

private String data1, data2, data3, data4, data5, user\_name;

private String[] data6;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_preview\_plan);

Toolbar toolbar = findViewById(R.id.plantoolbar);

setSupportActionBar(toolbar);

intent = getIntent();

state = findViewById(R.id.State);

category = findViewById(R.id.Category);

rating = findViewById(R.id.Rating);

place1 = findViewById(R.id.Place1);

place2 = findViewById(R.id.Place2);

place3 = findViewById(R.id.Place3);

accommodation = findViewById(R.id.Accommodation);

travel\_mode = findViewById(R.id.Travel\_Mode);

savebtn = findViewById(R.id.savebutton);

user\_name = intent.getStringExtra("USER\_NAME");

data1 = intent.getStringExtra("KEY1");

data2 = intent.getStringExtra("KEY2");

data3 = intent.getStringExtra("KEY3");

data4 = intent.getStringExtra("KEY4");

data5 = intent.getStringExtra("KEY5");

data6 = intent.getStringArrayExtra("KEY6"); // Assuming KEY6 contains comma-separated values

state.setText(String.format("State: %s", data1));

category.setText(String.format("Category: %s", data2));

rating.setText(String.format("Rating: %s", data3));

accommodation.setText(String.format("Accommodation: %s", data4));

travel\_mode.setText(String.format("Travel Mode: %s", data5));

place1.setText(data6[0]);

place2.setText(data6[1]);

place3.setText(data6[2]);

place1.setOnClickListener(v -> {

navigateToPlace(data6[0],data1);

});

place2.setOnClickListener(v -> {

navigateToPlace(data6[1],data1);

});

place3.setOnClickListener(v -> {

navigateToPlace(data6[2],data1);

});

savebtn.setOnClickListener(v -> saveDataToFirebase());

getOnBackPressedDispatcher().addCallback(this, new OnBackPressedCallback(true) {

@Override

public void handleOnBackPressed() {

Intent intent = new Intent(Preview\_Plan.this, Home\_page.class);

intent.putExtra("USER\_NAME",user\_name);

startActivity(intent);

finish();

}

});

}

private void saveDataToFirebase() {

FirebaseDatabase database = FirebaseDatabase.getInstance();

DatabaseReference myRef = database.getReference("plans");

String planId = myRef.push().getKey();

Generate\_plans plan = new Generate\_plans(user\_name, data1, data2, data3, data4, data5, data6[0], data6[1], data6[2]);

if (planId != null) {

myRef.child(user\_name).child(planId).setValue(plan).addOnCompleteListener(task -> {

if (task.isSuccessful()) {

Toast.makeText(Preview\_Plan.this, "Plan saved successfully", Toast.LENGTH\_SHORT).show();

} else {

Toast.makeText(Preview\_Plan.this, "Failed to save plan", Toast.LENGTH\_SHORT).show();

}

});

}

}

private void navigateToPlace(String place,String state) {

String travelMode = data5.toLowerCase();

String mode = "d";

switch (travelMode) {

case "bus":

mode = "transit";

break;

case "car":

mode = "driving";

break;

case "train":

mode = "transit";

break;

}

Log.d("travel preference",travelMode);

Uri gmmIntentUri = Uri.parse("google.navigation:q=" + Uri.encode(place+","+state+",India") + "&mode=" + mode);

Intent mapIntent = new Intent(Intent.ACTION\_VIEW, gmmIntentUri);

mapIntent.setPackage("com.google.android.apps.maps");

if (mapIntent.resolveActivity(getPackageManager()) != null) {

startActivity(mapIntent);

}

}

}

**Generate Plans Class:**

package com.example.home\_window;

public class Generate\_plans {

public String state;

public String user\_name;

public String category;

public String rating;

public String accommodation;

public String travel\_mode;

public String place1;

public String place2;

public String place3;

public Generate\_plans(){

}

public Generate\_plans(String user\_name, String state, String category, String rating, String accommodation, String travel\_mode, String place1, String place2, String place3) {

this.user\_name = user\_name;

this.state = state;

this.category = category;

this.rating = rating;

this.accommodation = accommodation;

this.travel\_mode = travel\_mode;

this.place1 = place1;

this.place2 = place2;

this.place3 = place3;

}

public String getState() {

return state;

}

public String getCategory() {

return category;

}

public String getAccommodation() {

return accommodation;

}

public String getTravelMode() {

return travel\_mode;

}

public String getRating() {

return rating;

}

public String getPlace1(){return place1;}

public String getPlace2(){return place2;}

public String getPlace3(){return place3;}

}

**Layout:**

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

<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"

xmlns:app="http://schemas.android.com/apk/res-auto"

xmlns:tools="http://schemas.android.com/tools"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

android:background="#87CEEB"

tools:context=".Preview\_Plan">

<androidx.appcompat.widget.Toolbar

android:id="@+id/plantoolbar"

android:layout\_width="match\_parent"

android:layout\_height="?attr/actionBarSize"

android:background="@color/royalblue"

app:title="Plan"

app:titleTextColor="@android:color/white"

app:layout\_constraintTop\_toTopOf="parent" />

<LinearLayout

android:id="@+id/linearLayout3"

android:layout\_width="374dp"

android:layout\_height="wrap\_content"

android:layout\_gravity="center\_horizontal"

android:background="@drawable/rounded\_rectangle"

android:elevation="8dp"

android:orientation="vertical"

android:outlineAmbientShadowColor="@color/royalblue"

android:outlineProvider="bounds"

android:padding="20dp"

app:layout\_constraintBottom\_toBottomOf="parent"

app:layout\_constraintEnd\_toEndOf="parent"

app:layout\_constraintHorizontal\_bias="0.567"

app:layout\_constraintStart\_toStartOf="parent"

app:layout\_constraintTop\_toTopOf="parent"

app:layout\_constraintVertical\_bias="0.547">

<TextView

android:id="@+id/State"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:text="State"

android:textSize="20sp" />

<Space

android:layout\_width="match\_parent"

android:layout\_height="10sp" />

<LinearLayout

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

android:orientation="horizontal">

<TextView

android:id="@+id/Category"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:layout\_weight="1"

android:text="Category"

android:textSize="20sp" />

<TextView

android:id="@+id/Rating"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:layout\_weight="1"

android:text="Rating"

android:textSize="20sp" />

</LinearLayout>

<Space

android:layout\_width="match\_parent"

android:layout\_height="10sp" />

<TextView

android:id="@+id/Place1"

android:layout\_width="wrap\_content"

android:layout\_height="48dp"

android:layout\_marginStart="10dp"

android:clickable="true"

android:text="Place1"

android:textSize="20sp" />

<TextView

android:id="@+id/Place2"

android:layout\_width="wrap\_content"

android:layout\_height="48dp"

android:layout\_marginStart="10dp"

android:clickable="true"

android:text="Place2"

android:textSize="20sp" />

<TextView

android:id="@+id/Place3"

android:layout\_width="wrap\_content"

android:layout\_height="48dp"

android:layout\_marginStart="10dp"

android:clickable="true"

android:text="Place3"

android:textSize="20sp" />

<Space

android:layout\_width="match\_parent"

android:layout\_height="10sp" />

<TextView

android:id="@+id/Accommodation"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:text="Accommodation"

android:textSize="20sp" />

<Space

android:layout\_width="match\_parent"

android:layout\_height="10sp" />

<TextView

android:id="@+id/Travel\_Mode"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:text="Travel\_Mode"

android:textSize="20sp" />

</LinearLayout>

<Button

android:id="@+id/savebutton"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Save Plan"

android:backgroundTint="@color/royalblue"

android:textColor="@color/white"

app:layout\_constraintBottom\_toBottomOf="parent"

app:layout\_constraintEnd\_toEndOf="parent"

app:layout\_constraintStart\_toStartOf="parent"

app:layout\_constraintTop\_toBottomOf="@+id/linearLayout3"

app:layout\_constraintVertical\_bias="0.321" />

</androidx.constraintlayout.widget.ConstraintLayout>

**Output:**

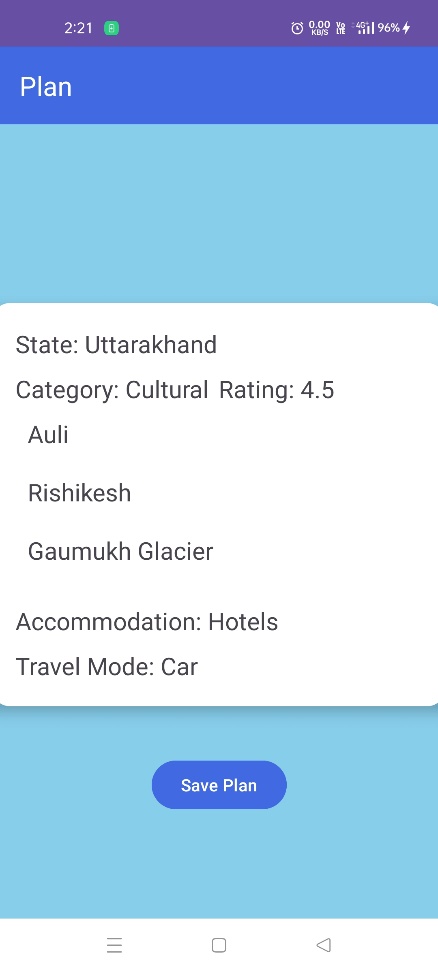


Figure 36: Output 7

**Live Support:**

**Class:**

import androidx.appcompat.widget.Toolbar;

import androidx.core.app.ActivityCompat;

import androidx.appcompat.app.AppCompatActivity;

import android.Manifest;

import android.annotation.SuppressLint;

import android.content.Context;

import android.content.Intent;

import android.content.pm.PackageManager;

import android.location.Location;

import android.location.LocationManager;

import android.net.Uri;

import android.os.Bundle;

import android.view.View;

import android.widget.Button;

import android.widget.Toast;

import android.os.Looper;

import android.provider.Settings;

import androidx.annotation.NonNull;

import com.google.android.gms.location.FusedLocationProviderClient;

import com.google.android.gms.location.LocationCallback;

import com.google.android.gms.location.LocationRequest;

import com.google.android.gms.location.LocationResult;

import com.google.android.gms.location.LocationServices;

import com.google.android.gms.tasks.OnCompleteListener;

import com.google.android.gms.tasks.OnSuccessListener;

import com.google.android.gms.tasks.Task;

import android.view.Menu;

import android.view.MenuInflater;

import android.view.MenuItem;

import com.google.firebase.auth.FirebaseAuth;

public class Live\_Support extends AppCompatActivity {

private FirebaseAuth firebaseAuth;

private Button restaurantButton, hotelsButton, hospitalsButton, policeButton;

private FusedLocationProviderClient fusedLocationClient;

private static final int PERMISSION\_ID = 44;

private String supportQuery;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_live\_support);

Toolbar toolbar = findViewById(R.id.toolbar);

setSupportActionBar(toolbar);

firebaseAuth = FirebaseAuth.getInstance();

fusedLocationClient = LocationServices.getFusedLocationProviderClient(this);

restaurantButton = findViewById(R.id.Restaurant);

hotelsButton = findViewById(R.id.Hotels);

hospitalsButton = findViewById(R.id.Hospitals);

policeButton = findViewById(R.id.Police);

restaurantButton.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View v) {

supportQuery = "restaurants+nearby";

getLastLocation();

}

});

hotelsButton.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View v) {

supportQuery = "hotels+nearby";

getLastLocation();

}

});

hospitalsButton.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View v) {

supportQuery = "hospitals+nearby";

getLastLocation();

}

});

policeButton.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View v) {

supportQuery = "police+station+nearby";

getLastLocation();

}

});

}

@SuppressLint("MissingPermission")

private void getLastLocation() {

if (checkPermissions()) {

if (isLocationEnabled()) {

fusedLocationClient.getLastLocation().addOnCompleteListener(new OnCompleteListener<Location>() {

@Override

public void onComplete(@NonNull Task<Location> task) {

Location location = task.getResult();

if (location == null) {

requestNewLocationData();

} else {

double latitude = location.getLatitude();

double longitude = location.getLongitude();

String geoUri = "geo:" + latitude + "," + longitude + "?q=" + supportQuery;

Uri gmmIntentUri = Uri.parse(geoUri);

Intent mapIntent = new Intent(Intent.ACTION\_VIEW, gmmIntentUri);

mapIntent.setPackage("com.google.android.apps.maps");

startActivity(mapIntent);

}

}

});

} else {

Toast.makeText(this, "Please turn on your location...", Toast.LENGTH\_LONG).show();

Intent intent = new Intent(Settings.ACTION\_LOCATION\_SOURCE\_SETTINGS);

startActivity(intent);

}

} else {

requestPermissions();

}

}

@SuppressLint("MissingPermission")

private void requestNewLocationData() {

LocationRequest mLocationRequest = new LocationRequest();

mLocationRequest.setPriority(LocationRequest.PRIORITY\_HIGH\_ACCURACY);

mLocationRequest.setInterval(5);

mLocationRequest.setFastestInterval(0);

mLocationRequest.setNumUpdates(1);

fusedLocationClient.requestLocationUpdates(mLocationRequest, mLocationCallback, Looper.myLooper());

}

private LocationCallback mLocationCallback = new LocationCallback() {

@Override

public void onLocationResult(LocationResult locationResult) {

Location mLastLocation = locationResult.getLastLocation();

double latitude = mLastLocation.getLatitude();

double longitude = mLastLocation.getLongitude();

String geoUri = "geo:" + latitude + "," + longitude + "?q=" + supportQuery;

Uri gmmIntentUri = Uri.parse(geoUri);

Intent mapIntent = new Intent(Intent.ACTION\_VIEW, gmmIntentUri);

mapIntent.setPackage("com.google.android.apps.maps");

startActivity(mapIntent);

}

};

private boolean checkPermissions() {

return ActivityCompat.checkSelfPermission(this, Manifest.permission.ACCESS\_COARSE\_LOCATION) == PackageManager.PERMISSION\_GRANTED &&

ActivityCompat.checkSelfPermission(this, Manifest.permission.ACCESS\_FINE\_LOCATION) == PackageManager.PERMISSION\_GRANTED;

}

private void requestPermissions() {

ActivityCompat.requestPermissions(this, new String[]{Manifest.permission.ACCESS\_COARSE\_LOCATION, Manifest.permission.ACCESS\_FINE\_LOCATION}, PERMISSION\_ID);

}

private boolean isLocationEnabled() {

LocationManager locationManager = (LocationManager) getSystemService(Context.LOCATION\_SERVICE);

return locationManager.isProviderEnabled(LocationManager.GPS\_PROVIDER) || locationManager.isProviderEnabled(LocationManager.NETWORK\_PROVIDER);

}

@Override

public void onRequestPermissionsResult(int requestCode, @NonNull String[] permissions, @NonNull int[] grantResults) {

super.onRequestPermissionsResult(requestCode, permissions, grantResults);

if (requestCode == PERMISSION\_ID) {

if (grantResults.length > 0 && grantResults[0] == PackageManager.PERMISSION\_GRANTED) {

// Permission granted, get last location

getLastLocation();

} else {

Toast.makeText(this, "Permission denied", Toast.LENGTH\_SHORT).show();

}

}

}

public boolean onCreateOptionsMenu(Menu menu){

MenuInflater inflater = getMenuInflater();

inflater.inflate(R.menu.home\_menu, menu);

return true;

}

public boolean onOptionsItemSelected(MenuItem item){

if (item.getItemId() == R.id.menu\_logout) {

logout();

return true;

}

return super.onOptionsItemSelected(item);

}

private void logout(){

firebaseAuth.signOut();

Intent intent = new Intent(Live\_Support.this, MainActivity.class);

startActivity(intent);

}

}

**Layout:**

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

<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"

xmlns:app="http://schemas.android.com/apk/res-auto"

xmlns:tools="http://schemas.android.com/tools"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

android:background="#87CEEB"

tools:context=".Live\_Support">

<androidx.appcompat.widget.Toolbar

android:id="@+id/toolbar"

android:layout\_width="match\_parent"

android:layout\_height="?attr/actionBarSize"

android:background="@color/royalblue"

app:title="Live Support"

app:titleTextColor="@android:color/white"

app:layout\_constraintTop\_toTopOf="parent"/>

<LinearLayout

android:id="@+id/btnav"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_gravity="center\_horizontal"

android:background="@drawable/rounded\_rectangle"

android:elevation="8dp"

android:orientation="vertical"

android:outlineAmbientShadowColor="@color/royalblue"

android:outlineProvider="bounds"

android:padding="20dp"

app:layout\_constraintBottom\_toBottomOf="parent"

app:layout\_constraintEnd\_toEndOf="parent"

app:layout\_constraintStart\_toStartOf="parent"

app:layout\_constraintTop\_toTopOf="parent">

<Button

android:id="@+id/Restaurant"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:backgroundTint="#000080"

android:layout\_weight="1"

android:text="Food"

android:textColor="@color/white" />

<Button

android:id="@+id/Hotels"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:backgroundTint="#000080"

android:layout\_weight="1"

android:text="Hotels"

android:textColor="@color/white" />

<Button

android:id="@+id/Hospitals"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:backgroundTint="#000080"

android:layout\_weight="1"

android:text="Health"

android:textColor="@color/white" />

<Button

android:id="@+id/Police"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:backgroundTint="#000080"

android:layout\_weight="1"

android:text="Police"

android:textColor="@color/white" />

</LinearLayout>

</androidx.constraintlayout.widget.ConstraintLayout>

**Output:**

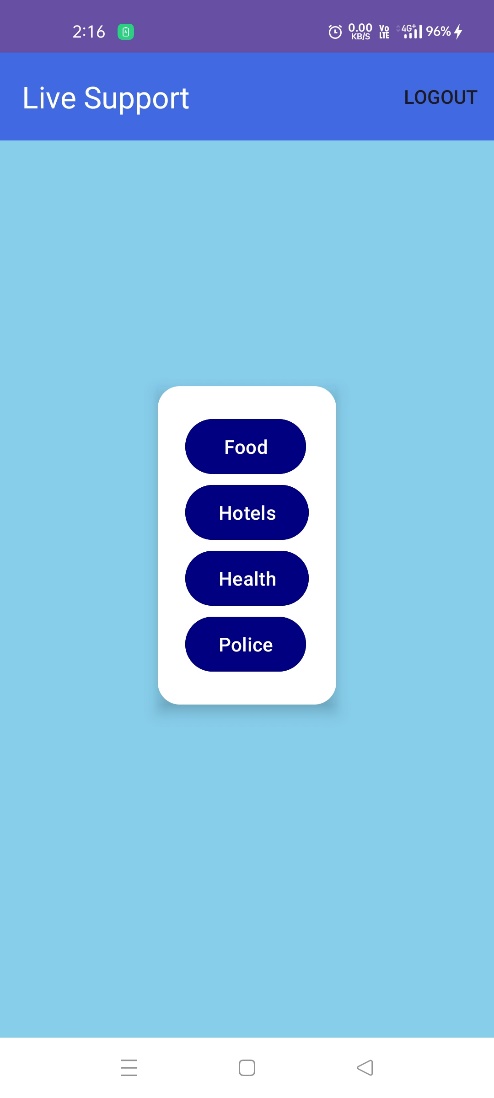


Figure 37: Output 8

**User Profile:**

**Class:**

package com.example.home\_window;

import androidx.annotation.NonNull;

import androidx.appcompat.app.AlertDialog;

import androidx.appcompat.app.AppCompatActivity;

import android.content.DialogInterface;

import android.content.Intent;

import android.os.Bundle;

import android.text.TextUtils;

import android.view.View;

import android.widget.Button;

import android.widget.EditText;

import android.widget.TextView;

import android.widget.Toast;

import com.google.firebase.database.DataSnapshot;

import com.google.firebase.database.DatabaseError;

import com.google.firebase.database.DatabaseReference;

import com.google.firebase.database.FirebaseDatabase;

import com.google.firebase.database.ValueEventListener;

public class User\_Profile extends AppCompatActivity {

private TextView user\_name, Phone;

private Button chng1, chng2;

private String un;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_user\_profile);

user\_name = findViewById(R.id.name);

Phone = findViewById(R.id.number);

chng1 = findViewById(R.id.changename);

chng2 = findViewById(R.id.changenum);

Intent intent = getIntent();

un = intent.getStringExtra("USER\_NAME");

if (un != null) {

user\_name.setText(un);

fetchPhoneNumber(un);

}

chng1.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View v) {

showChangeNameDialog();

}

});

chng2.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View v) {

showChangePhoneDialog();

}

});

}

private void fetchPhoneNumber(String userName) {

DatabaseReference usersRef = FirebaseDatabase.getInstance().getReference("users");

usersRef.orderByChild("name").equalTo(userName).addListenerForSingleValueEvent(new ValueEventListener() {

@Override

public void onDataChange(@NonNull DataSnapshot dataSnapshot) {

if (dataSnapshot.exists()) {

for (DataSnapshot snapshot : dataSnapshot.getChildren()) {

String phone = snapshot.child("phone").getValue(String.class);

if (phone != null) {

Phone.setText(phone);

} else {

Toast.makeText(User\_Profile.this, "Phone number not found.", Toast.LENGTH\_SHORT).show();

}

}

} else {

Toast.makeText(User\_Profile.this, "User not found.", Toast.LENGTH\_SHORT).show();

}

}

@Override

public void onCancelled(@NonNull DatabaseError databaseError) {

Toast.makeText(User\_Profile.this, "Database error: " + databaseError.getMessage(), Toast.LENGTH\_SHORT).show();

}

});

}

private void showChangeNameDialog() {

AlertDialog.Builder builder = new AlertDialog.Builder(this);

builder.setTitle("Change Name");

final EditText input = new EditText(this);

input.setHint("Enter new name");

builder.setView(input);

builder.setPositiveButton("Change", new DialogInterface.OnClickListener() {

@Override

public void onClick(DialogInterface dialog, int which) {

String newName = input.getText().toString().trim();

if (!TextUtils.isEmpty(newName)) {

updateName(newName);

} else {

Toast.makeText(User\_Profile.this, "Name cannot be empty", Toast.LENGTH\_SHORT).show();

}

}

});

builder.setNegativeButton("Cancel", null);

builder.show();

}

private void showChangePhoneDialog() {

AlertDialog.Builder builder = new AlertDialog.Builder(this);

builder.setTitle("Change Phone Number");

final EditText input = new EditText(this);

input.setHint("Enter new phone number");

builder.setView(input);

builder.setPositiveButton("Change", new DialogInterface.OnClickListener() {

@Override

public void onClick(DialogInterface dialog, int which) {

String newPhone = input.getText().toString().trim();

if (!TextUtils.isEmpty(newPhone)) {

updatePhoneNumber(newPhone);

} else {

Toast.makeText(User\_Profile.this, "Phone number cannot be empty", Toast.LENGTH\_SHORT).show();

}

}

});

builder.setNegativeButton("Cancel", null);

builder.show();

}

private void updateName(final String newName) {

final DatabaseReference usersRef = FirebaseDatabase.getInstance().getReference("users");

usersRef.orderByChild("name").equalTo(un).addListenerForSingleValueEvent(new ValueEventListener() {

@Override

public void onDataChange(@NonNull DataSnapshot dataSnapshot) {

if (dataSnapshot.exists()) {

for (DataSnapshot snapshot : dataSnapshot.getChildren()) {

snapshot.getRef().child("name").setValue(newName);

user\_name.setText(newName);

un = newName;

Toast.makeText(User\_Profile.this, "Name updated successfully", Toast.LENGTH\_SHORT).show();

}

} else {

Toast.makeText(User\_Profile.this, "User not found.", Toast.LENGTH\_SHORT).show();

}

}

@Override

public void onCancelled(@NonNull DatabaseError databaseError) {

Toast.makeText(User\_Profile.this, "Database error: " + databaseError.getMessage(), Toast.LENGTH\_SHORT).show();

}

});

}

private void updatePhoneNumber(final String newPhone) {

final DatabaseReference usersRef = FirebaseDatabase.getInstance().getReference("users");

usersRef.orderByChild("name").equalTo(un).addListenerForSingleValueEvent(new ValueEventListener() {

@Override

public void onDataChange(@NonNull DataSnapshot dataSnapshot) {

if (dataSnapshot.exists()) {

for (DataSnapshot snapshot : dataSnapshot.getChildren()) {

snapshot.getRef().child("phone").setValue(newPhone);

Phone.setText(newPhone);

Toast.makeText(User\_Profile.this, "Phone number updated successfully", Toast.LENGTH\_SHORT).show();

}

} else {

Toast.makeText(User\_Profile.this, "User not found.", Toast.LENGTH\_SHORT).show();

}

}

@Override

public void onCancelled(@NonNull DatabaseError databaseError) {

Toast.makeText(User\_Profile.this, "Database error: " + databaseError.getMessage(), Toast.LENGTH\_SHORT).show();

}

});

}

}

**Layout:**

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

<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"

xmlns:app="http://schemas.android.com/apk/res-auto"

xmlns:tools="http://schemas.android.com/tools"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

android:background="#87CEEB"

tools:context=".User\_Profile">

<ImageView

android:id="@+id/imageView2"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_marginTop="16dp"

app:layout\_constraintBottom\_toTopOf="@+id/userprofilell"

app:layout\_constraintEnd\_toEndOf="parent"

app:layout\_constraintStart\_toStartOf="parent"

app:layout\_constraintTop\_toTopOf="parent"

tools:srcCompat="@tools:sample/avatars" />

<LinearLayout

android:id="@+id/userprofilell"

android:layout\_width="374dp"

android:layout\_height="wrap\_content"

android:layout\_gravity="center\_horizontal"

android:background="@drawable/rounded\_rectangle"

android:elevation="8dp"

android:orientation="vertical"

android:outlineAmbientShadowColor="@color/royalblue"

android:outlineProvider="bounds"

android:padding="20dp"

app:layout\_constraintBottom\_toBottomOf="parent"

app:layout\_constraintEnd\_toEndOf="parent"

app:layout\_constraintHorizontal\_bias="0.567"

app:layout\_constraintStart\_toStartOf="parent"

app:layout\_constraintTop\_toTopOf="parent"

app:layout\_constraintVertical\_bias="0.547">

<TextView

android:id="@+id/name"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:text="TextView"

android:textSize="24sp"/>

<Space

android:layout\_width="match\_parent"

android:layout\_height="10sp" />

<TextView

android:id="@+id/number"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:text="TextView"

android:textSize="24sp" />

<LinearLayout

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

android:orientation="horizontal">

<Button

android:id="@+id/changename"

android:layout\_width="0dp"

android:layout\_height="wrap\_content"

android:backgroundTint="@color/royalblue"

android:layout\_weight="1"

android:text="change user name"

android:textColor="@color/white" />

<Button

android:id="@+id/changenum"

android:layout\_width="0dp"

android:layout\_height="wrap\_content"

android:backgroundTint="@color/royalblue"

android:layout\_weight="1"

android:text="change number"

android:textColor="@color/white" />

</LinearLayout>

</LinearLayout>

</androidx.constraintlayout.widget.ConstraintLayout>

**Output:**

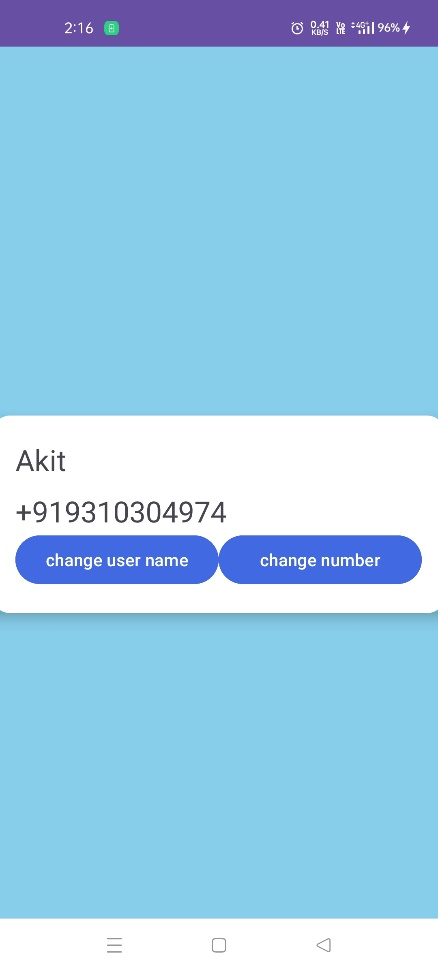


Figure 38: Output 9

# **Chapter 9: Conclusion and Future Scope**

In its current form the application will allow the user to navigate through the traveling process by themselves although it might not bring all information from within but it is able to guide the user in not recommending them some great places to visit but also provide them with routes.

Live support is also key a function of this application, helping the user navigates during their travels.

The future scope of this application lies in upgrading the machine learning model as well as adding a collaborative model to work along side the current model so that the application is able to refine its output in two layers. Another future part of this application would be the addition of further functions in the application in order to bring all information that the user might need inside the app so that all points related to the traveling and touring can be covered inside the application.

# **Chapter 10: References**

https://www.tensorflow.org/learn

<https://www.tensorflow.org/guide/keras>

<https://www.geeksforgeeks.org/introduction-to-java/>

<https://www.geeksforgeeks.org/xml-basics/>

<https://www.geeksforgeeks.org/android-studio-tutorial/>

<https://www.educative.io/answers/core-features-of-android-studio>

<https://www.geeksforgeeks.org/software-engineering-iterative-waterfall-model/>

<https://www.geeksforgeeks.org/use-case-diagram/>

<https://www.javatpoint.com/dbms-er-model-concept>

<https://www.visual-paradigm.com/guide/data-flow-diagram/what-is-data-flow-diagram/>

<https://www.geeksforgeeks.org/a-complete-guide-to-learn-xml-for-android-app-development/>

https://www.geeksforgeeks.org/unified-modeling-language-uml-activity-diagrams/