**MAJOR PROJECT REPORT ON**

Tourism Recommendation System

*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 System” 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 System” 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 System”** 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

**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 unlabeled 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

**Classification:**

Classification is a type of supervised learning where the algorithm learns to assign input data to a specific category or class based on input features. The output labels in classification are discrete values. Classification algorithms can be binary, where the output is one of two possible classes, or multiclass, where the output can be one of several classes. The different Classification algorithms in machine learning are: Logistic Regression, Naive Bayes, Decision Tree, Support Vector Machine (SVM), K-Nearest Neighbours (KNN), etc.

**(keras, dense layer, )**

**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:**

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

Android device: 8GB Ram, Version: 13

**3.2 Software Requirements:**

Android Studio

Java

Android Virtual Device or Emulator

XML for UI

Database (SQLite or Firebase)

Google Maps API

Version Control System (Git)

**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.

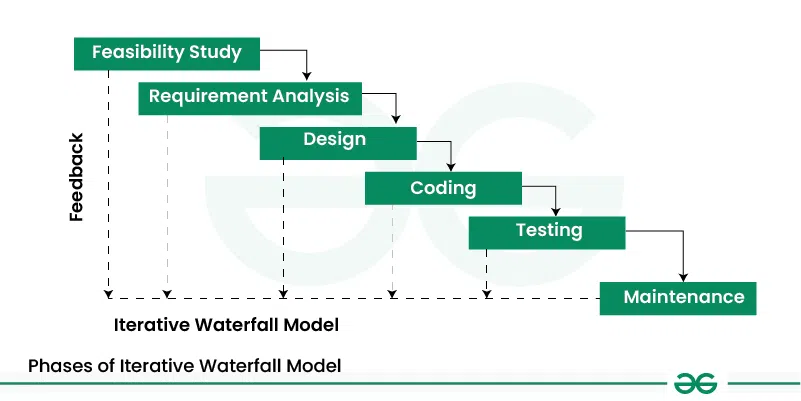
The feedback paths allow the phase to be reworked in which errors are committed and these changes are reflected in the later phases.

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.



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:**

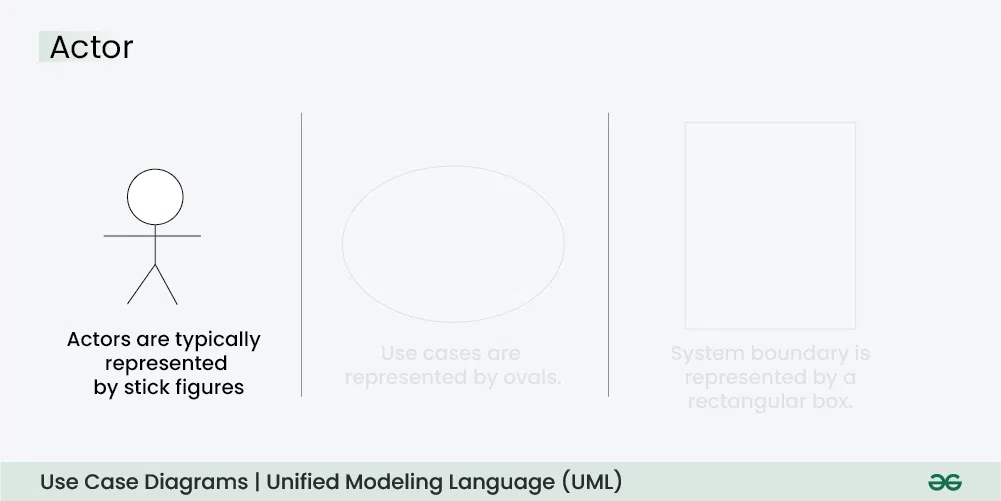
A Use Case Diagram is a type of Unified Modeling Language (UML) diagram that represents the interaction between actors (users or external systems) and a system under consideration to accomplish specific goals. It provides a high-level view of the system’s functionality by illustrating the various ways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

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.

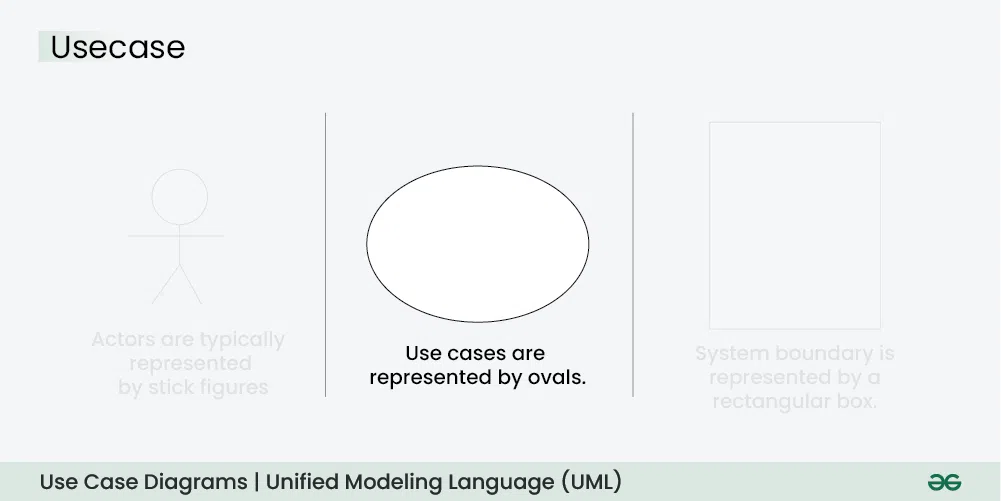
1.1. Actors

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.



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.



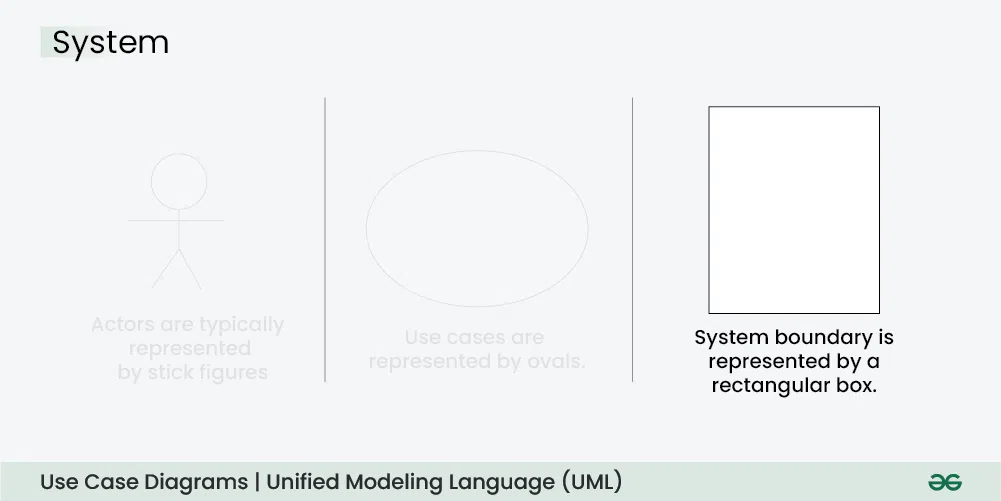
1.3. System Boundary

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.



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.

3.1. Association Relationship

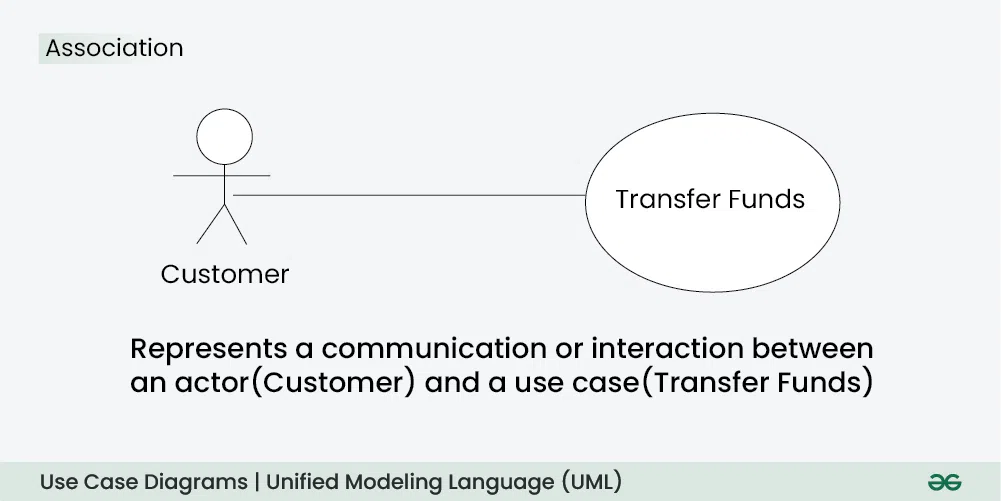
The Association Relationship represents a communication or interaction between an actor and a use case. It is depicted by a line connecting the actor to the use case. This relationship signifies that the actor is involved in the functionality described by the use case.

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.



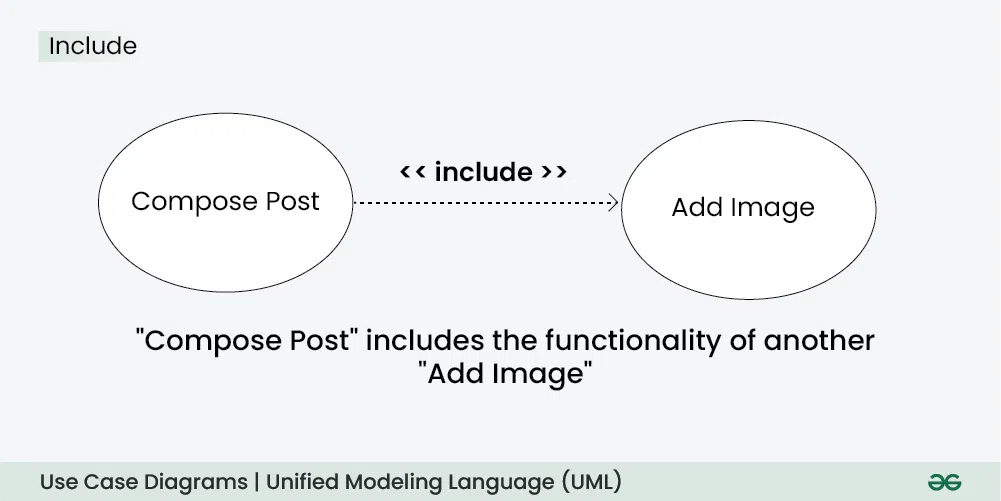
3.2. Include Relationship

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.



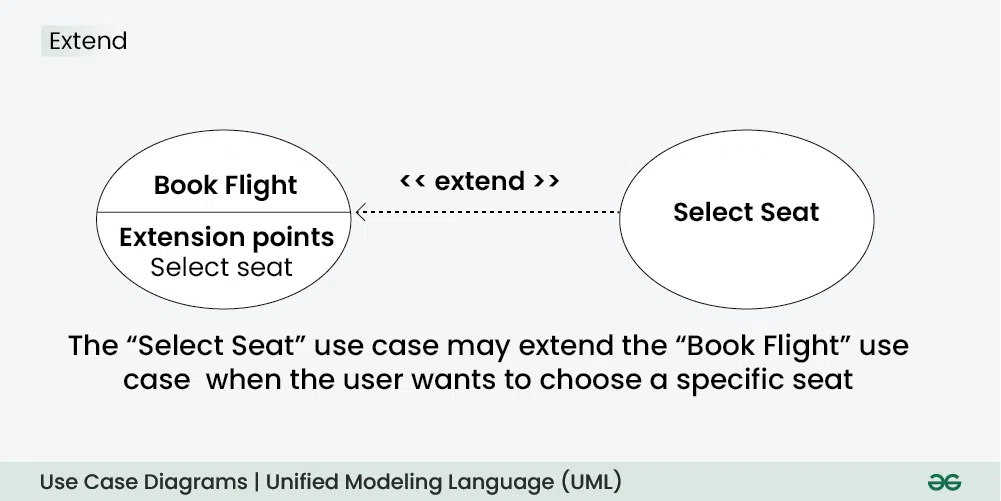
3.3. Extend Relationship

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.



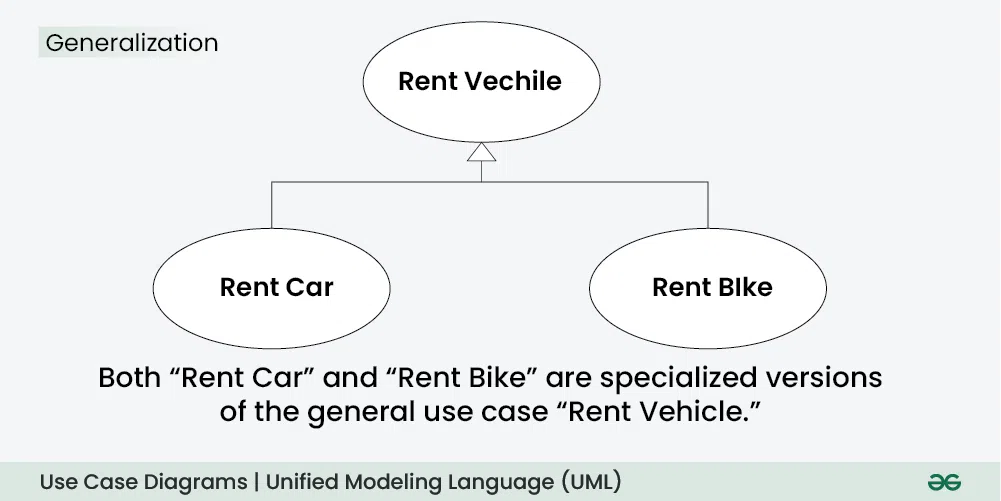
3.4. Generalization Relationship

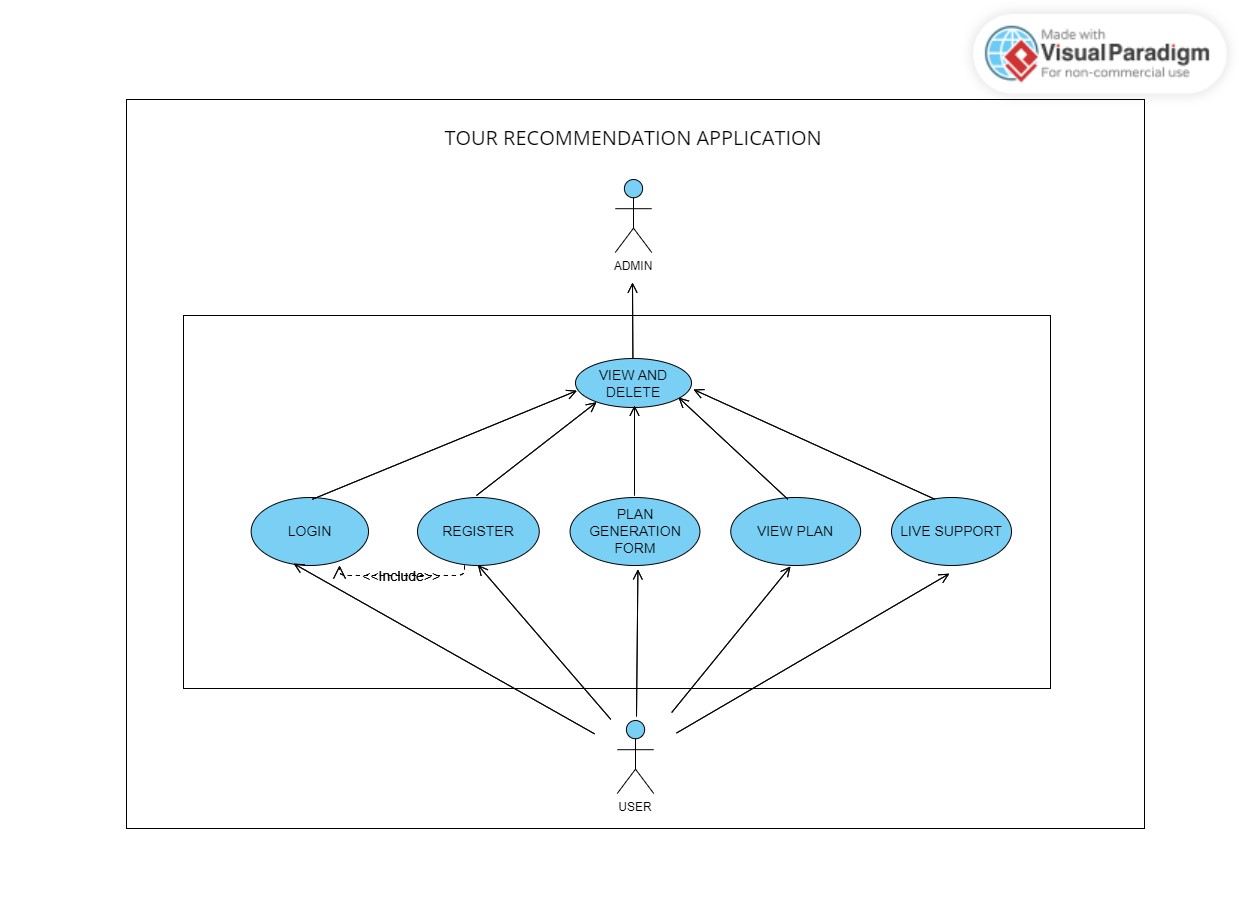
The Generalization Relationship establishes an “is-a” connection between two use cases, indicating that one use case is a specialized version of another. It is represented by an arrow pointing from the specialized use case to the general use case.

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.”





**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.

PERT helps management in deciding the best possible resource utilization method.

PERT take advantage by using time network analysis technique.

PERT 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.

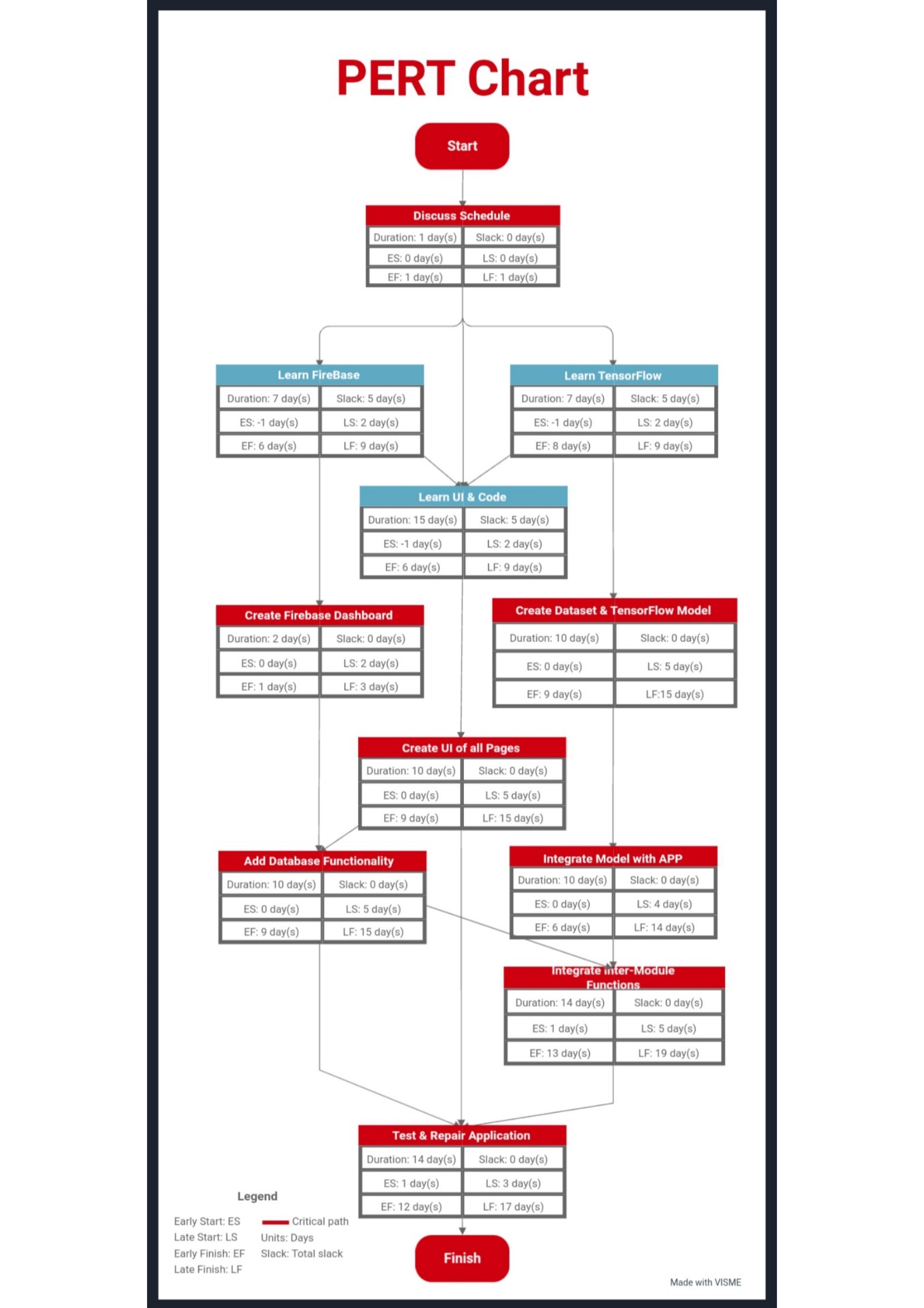
**Components of PERT Chart:**

**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.



**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:



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

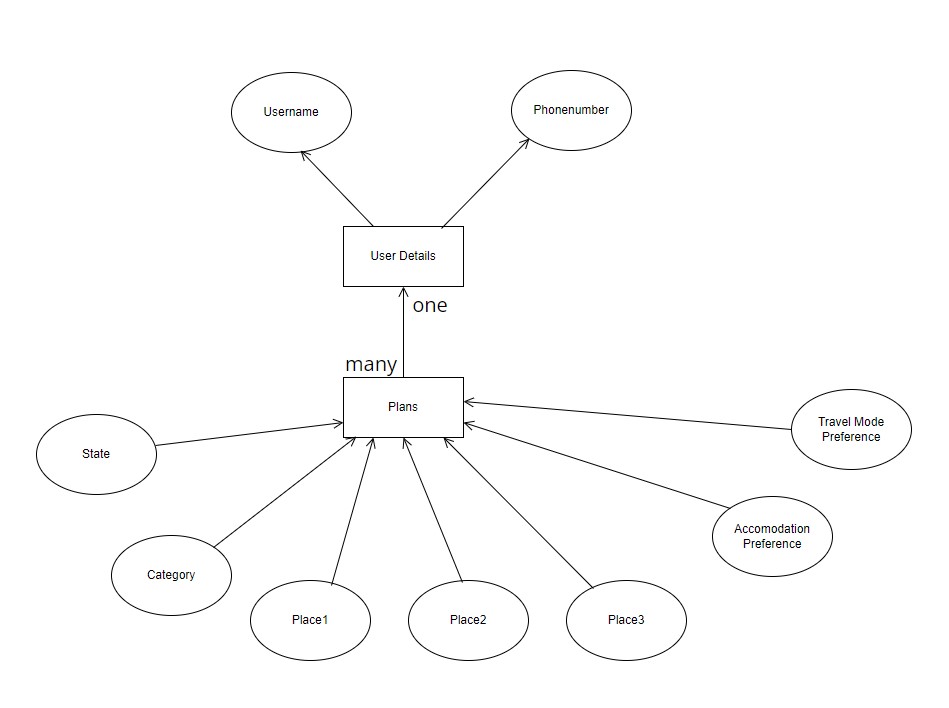
When more than one instance of the entity on the left, and only one instance of an entity on the right associates with the relationship then it is known as a many-to-one relationship.

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.



**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

**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

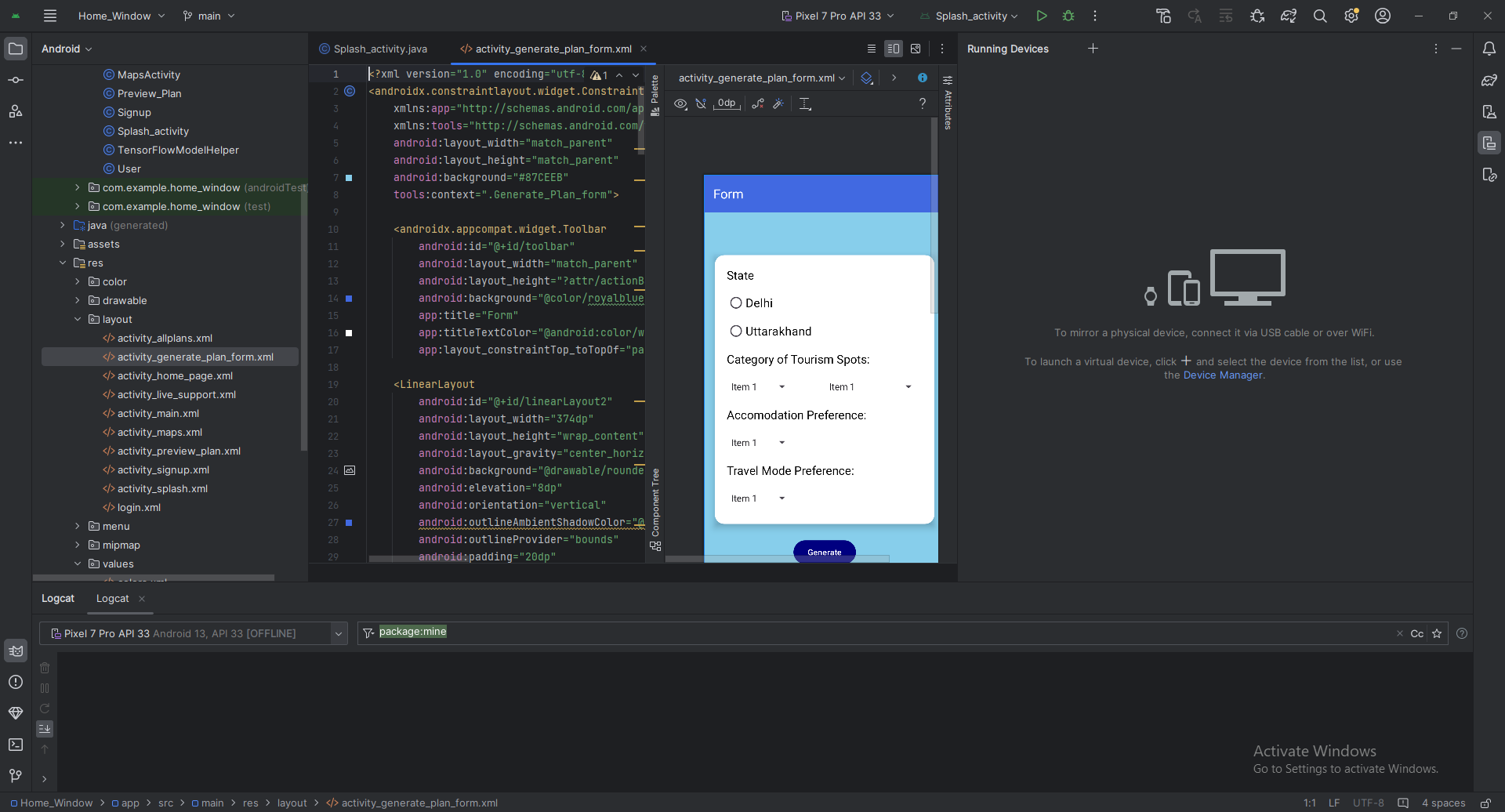
These are the XML files that provide graphics to elements like custom background for the buttons and its ripple effects, also various gradients can be created. This also holds the vector graphics like icons. Using these files custom layouts can be constructed for EditTexts.

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.



**7.4 Process Involved**

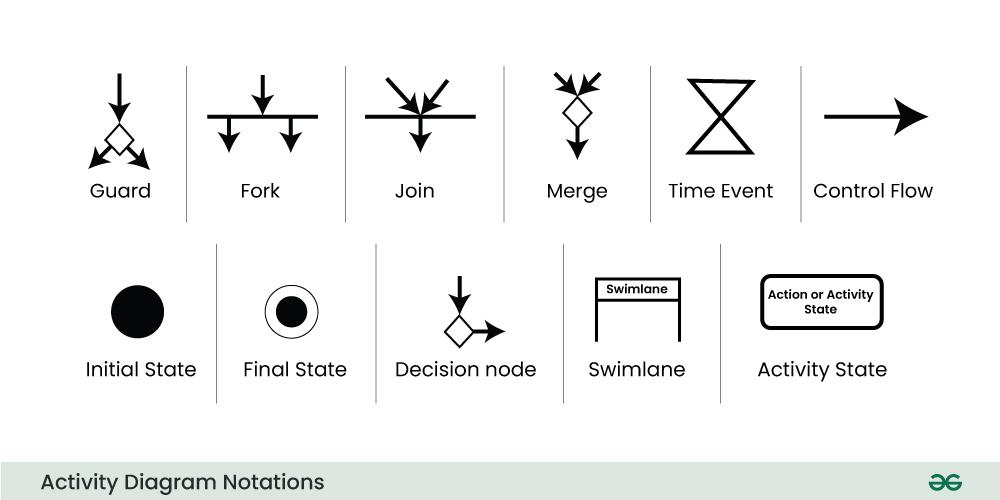
Activity Diagrams are used to illustrate the flow of control in a system and refer to the steps involved in the execution of a use case. We can depict both sequential processing and concurrent processing of activities using an activity diagram ie an activity diagram focuses on the condition of flow and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



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.

**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.

References:  
<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/