CHAPTER - 1

INTRODUCTION

Launched by Google in 2008, Android is an open-source operating system based on the Linux kernel that encourages community collaboration for continuous improvement. It is a flexible platform that works with a variety of devices because of its adaptability and versatility.

Google Central to Android's development is Google, offering core software, services, and updates. The Google Play Store serves as the official app distribution platform, granting users access to millions of applications. The user interface of Android is tailored for touch gestures, featuring a customizable home screen and support for widgets, ensuring an interactive and intuitive experience.

For application development on Android, the primary programming languages are Java and Kotlin. The Android Software Development Kit (SDK) provides the tools needed for app creation, leveraging a comprehensive set of APIs. The app ecosystem is diverse, covering categories from games and productivity tools to social media and entertainment apps.

Android prioritizes user customization, offering options such as changing wallpapers, themes, and utilizing different launchers. Advanced users can even root their devices to gain deeper access and control over the operating system. Security is a focal point for Android, incorporating features like app sandboxing, regular security updates, and permissions management to safeguard user data.

Regular updates are a hallmark of Android, introducing new features, improvements, and security patches. Notably, each major Android version is named after a dessert or sweet treat, following an alphabetical order. This nomenclature adds a whimsical touch to the platform.

In essence, Android's adaptability, extensive app ecosystem, commitment to regular updates, and playful nomenclature contribute to its widespread popularity in the ever-evolving realm of mobile technology. Its open nature continues to foster innovation and collaboration within the developer community.

CHAPTER – 2

TECHNOLOGY

Android, as an operating system for mobile devices, employs a variety of technologies to provide a robust and versatile platform. To excel in an Android development role, you'll need a combination of technical skills, soft skills, and a good understanding of the Android ecosystem. Here are the key skills that can help you succeed in an Android role:

1.Programming Languages (Java and Kotlin)

A solid grasp of Java is foundational for Android development. Kotlin, introduced by JetBrains, is now officially supported by Google and is increasingly favored for its conciseness, expressiveness, and safety features. Proficiency in both languages allows you to leverage their strengths based on project requirements.

2.Android SDK and API's

In-depth knowledge of the Android Software Development Kit (SDK) and understanding how to work with Android APIs is crucial. This includes expertise in UI components, data storage (Shared Preferences, SQLite, or Room), and utilizing features like notifications and permissions.

3.Android Studio

Android Studio is the official Integrated Development Environment (IDE) for Android development. A thorough understanding of Android Studio's features, including debugging tools, profilers, and the layout editor, is essential for efficient coding and debugging.

4.XML (Extensible Markup Language)

Android uses XML for defining layouts and UI elements. A developer should be proficient in creating and understanding XML layouts to design visually appealing and responsive user interfaces.

5.Version control/git

Git is a widely used version control system in collaborative software development. Proficiency in Git, including branching, merging, and conflict resolution, is crucial for effective collaboration and code management.

6.Gradle build system

Android projects rely on the Gradle build system. Understanding how to configure build scripts, manage dependencies, and optimize build processes ensures smooth development workflows and efficient app builds.

7.Database Management

Android developers should be proficient in working with databases, especially SQLite, and understand data storage options like Room Persistence Library. This skill is vital for managing and retrieving data efficiently in Android applications.

8.Communication and collaboration

Effective communication is vital, especially when working in a team. Clear expression of ideas, providing constructive feedback, and collaborating with other developers, designers, and stakeholders contribute to a positive and productive team environment.

CHAPTER - 3

APPLICATIONS

Android development is a form of software engineering dedicated specifically to creating applications for devices that run on the Android platform. Some of the uses cases of Android Development are-

1.Real-Time Messaging Apps:

- **Description:** Apps like WhatsApp or Telegram provide instant messaging, where users can send and receive messages in real-time.
- **Key Characteristics:**
    - Instant message delivery
    - Real-time presence status
    - Push notifications for new messages

2.Live Streaming Apps:

- **Description:** Platforms like YouTube or Twitch allow users to broadcast and
- watch live streams.
- **Key Characteristics:**
    - Low-latency streaming
    - Real-time viewer interaction (comments, likes)
    - Live updates on concurrent viewers

3.Collaborative Editing Apps:

- **Description:** Applications like Google Docs enable real-time collaboration on documents.
- **Key Characteristics:**
    - Simultaneous editing by multiple users
    - Real-time updates on changes
    - Version history tracking

4.Real-Time Navigation Apps:

- **Description:** Navigation apps like Google Maps provide real-time updates on
- traffic conditions and route changes.
- **Key Characteristics:**
    - Real-time GPS tracking
    - Dynamic route adjustments based on traffic

CHAPTER 4

MODULE EXPLANATION

Module 1: Your first Android app

The module describes about the android and requirements of android. The requirements include Kotlin programming language, setting up the android studio and describes building a basic layout.

Kotlin, a modern and concise programming language, has emerged as a preferred choice for Android app development. Endorsed by Google, Kotlin offers a seamless integration with existing Java code and brings a host of features like null safety, concise syntax, and improved code readability. Its expressive and pragmatic nature accelerates development, making it an excellent fit for Android projects. As the official language for Android app development since 2017, Kotlin enhances developer productivity, reduces boilerplate code, and contributes to building robust, efficient, and more maintainable Android applications.

Fig:4.1 Setting up android studio

Android Studio Installation and Configuration of SDK & JDK JDK download: https://www.oracle.com/java/technologies/downloads/#jdk17-windows IDE download:

https://developer.android.com/studioOperations Perspective

Module 2: Building app UI

This module offers an overview of fundamentals of Kotlin, widgets in android studio, and interacting with UI.

The fundamentals of Kotlin encompass key features of the modern and concise programming language, including robust support for object-oriented programming (OOP) principles. Kotlin seamlessly integrates with existing Java code, providing features like null safety, concise syntax, and improved code readability. Object-oriented programming principles such as encapsulation, inheritance, and polymorphism are foundational concepts in Kotlin, enabling developers to create modular and maintainable code structures. Additionally, Kotlin introduces the concept of lambdas, allowing for concise and expressive functional programming. Lambdas facilitate the creation of anonymous functions, enhancing code conciseness and promoting a more functional programming style within the Kotlin language.

Fig:4.2 Building calculator using button widget

Embarking on the exploration of UI interaction and state management, we delve into the development of a tip calculator app. The app is designed to compute tips based on user input, demonstrating fundamental principles of user interface design and state management. Throughout the development process, we explore techniques for capturing user input, dynamically updating the UI, and managing the application's state to ensure a seamless and responsive user experience. By the end of this module, learners will have gained valuable insights into creating interactive UIs and handling states, providing a solid foundation for developing user-friendly applications.

Module 3: Display lists and use Material Design

Commencing with a succinct overview, this encapsulation outlines key Kotlin programming concepts vital for developers aspiring to craft lively and engaging Android applications.

From Kotlin's clean and readable syntax to its support for asynchronous programming through coroutines, these concepts empower developers to build responsive, efficient, and feature-rich apps. The language's seamless interoperability with Java, coupled with advanced features like extension functions and higher-order functions, promotes expressive coding styles, enhancing the overall development experience. Additionally, Kotlin's robust type system and support for null safety contribute to more reliable code, fostering a foundation for creating sophisticated and enjoyable Android applications.

Fig:4.3 Material Design

In this module, we explore the creation of an app using Compose that showcases a scrollable list containing both text and images. By following the provided guidelines, developers will gain hands-on experience in leveraging Compose, a modern Android UI toolkit, to design and implement dynamic interfaces. The app's functionality includes the seamless integration of text and images within a scrollable layout,

demonstrating essential techniques for creating engaging and visually appealing user experiences. This documentation serves as a valuable resource for developers seeking insights into building versatile and interactive applications through Compose.

Elevate the visual appeal and user experience of your applications by incorporating Material Design principles, animations, and accessibility best practices. This documentation pro vides valuable insights into creating more beautiful and intuitive apps. Material Design ensures a consistent and polished appearance, while animations add a dynamic touch to interactions, enhancing engagement. Following accessibility best practices ensures inclusivity for diverse users. Developers will find this guide essential for implementing these design elements, contributing to the overall aesthetic and usability of their applications.

Module 4: Navigation and app architecture

This module gives knowledge about developers in harnessing the power of the Navigation component to construct intricate Android applications with multiple screens. Learn the art of seamless navigation between different composables while efficiently passing data between screens. By delving into this module, developers will acquire the skills to architect more complex and interconnected apps, enhancing user experiences through intuitive and well- organized navigation flows. Mastering the Navigation component's capabilities ensures a streamlined approach to handling diverse app architectures, fostering the development of dynamic and feature-rich applications.

Efficiently adapt your app to diverse screen sizes and elevate user experiences with this comprehensive documentation. Explore strategies for responsive design that optimize your application's visual appeal across various devices. Gain practical insights into testing and refining your adaptive UI, ensuring seamless interactions for users on different screen dimensions. This guide empowers developers to create versatile applications that dynamically respond to the unique characteristics of various screen sizes, delivering an enhanced and consistent user experience.

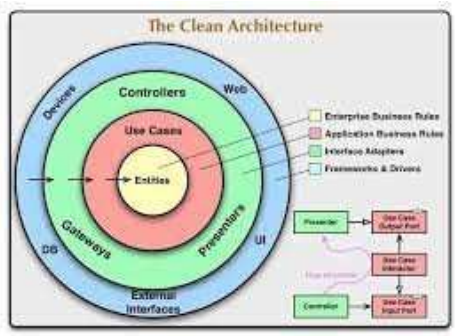

Fig:4.4 Jetpack compose

Embark on a comprehensive exploration of the Navigation component through this detailed documentation, designed to empower developers in constructing sophisticated Android applications featuring multiple screens. Delving into intricate

strategies for seamless navigation between diverse composables, the module provides nuanced insights into the hierarchical structure and design principles conducive to an integrated user journey. Additionally, developers will gain profound knowledge on the effective passing and management of data between screens, emphasizing efficiency and maintaining a resilient architecture. This resource is tailored to help developers master the advanced functionalities of the Navigation component, enabling the creation of dynamic, interconnected applications that respond dynamically to user interactions, ensuring an elevated and personalized user experience.

Module 5: Connect to the internet

Connecting to the internet is a fundamental aspect of modern application development, enabling communication between devices and servers. Here's a concise overview of key concepts when establishing internet connections in software:

Network Permission: Ensure that your application has the necessary network permissions declared in the AndroidManifest.xml file. This is crucial for the app to access the internet.

- **Network Requests:** Use HTTP or HTTPS protocols to initiate network requests from your application to remote servers. Commonly, this is done using the following methods:
- **HTTP URL Connection:** A basic API provided by Java for sending HTTP requests and receiving responses.
- **Http Client:** An alternative for sending HTTP requests, though it has been deprecated in recent Android versions.
- **Get data from internet:** Fetching data from the internet is a common requirement in modern application development. Whether you're retrieving information from a RESTful API, a web service, or a remote server, the process involves several key steps:
- **URL Connection:** Use URL Connection or Http URL Connection classes in Java to establish a connection to a remote server. Construct a URL object with the target endpoint, open a connection, and configure it for reading the response.

Network Permission: Ensure that your Android application has the necessary network permissions declared in the AndroidManifest.xml file to access the internet.

- **HTTP Methods:** Choose the appropriate HTTP method for your request.
- **GET:** Retrieve data from the server.
- **POST:** Send data to the server to create a new resource.
- **PUT or PATCH:** Update an existing resource on the server.
- **DELETE:** Remove a resource on the server.

Loading and Displaying Images from the Internet:

Loading and displaying images from the internet is a common requirement in many applications, especially those dealing with dynamic content or user-generated media.

Below are essential steps and considerations for achieving this in an Android application:

Network Permission: Ensure that your Android application has the necessary internet permissions declared in the AndroidManifest.xml file.

- **Choose Image Loading Library:** Consider using image loading libraries for efficient and optimized image loading. Popular libraries include:
- **Glide:** A fast and efficient open-source image loading library.
- **Picasso:** A widely used library for image loading and caching.
- **Coil:** A lightweight image loading library with modern features.
- **Dependency Integration:** Include the chosen image loading library in your project by adding the corresponding dependency to your app's build grade file.
- **Load Image from URL:** Utilize the library's API to load images directly from a URL. Typically, this involves passing the URL to the library's loading function.
- **Resize and Crop:** Consider resizing or cropping images based on the target Image View size to optimize memory usage and improve loading speed.

Fig:4.5 Connect to the Internet

Module 6: Data persistence

Data persistence in software development refers to the process of storing and retrieving data to and from a persistent storage medium, such as a database or file system. It is a crucial aspect of creating robust and user-friendly applications. Here's a concise summary of key concepts related to data persistence:

Types of Data Persistence:

- **Local Storage:** In-app storage using methods like Shared Preferences, SQLite databases, or file storage.
- **Remote Storage:** Storing data on remote servers, typically accessed through APIs.

Local Data Persistence:

- **Shared Preferences:** Lightweight key-value pairs for storing simple data.
- **SQLite Databases:** Relational databases embedded within the app for structured data storage.

Remote Data Persistence:

- **APIs (Application Programming Interfaces):** Interacting with remote servers through APIs to retrieve and send data.
- **Cloud Storage:** Storing data on cloud platforms, offering scalability and accessibility across devices.
- **Databases:** Databases are organized collections of structured information or data, typically stored electronically in a computer system. They serve as efficient and systematic ways to manage, organize, and retrieve data. Databases play a crucial role in various applications, ranging from small-scale projects to large enterprise systems. Key characteristics of databases include data integrity, security, and the ability to support concurrent access by multiple users.
- **SQL (Structured Query Language):** SQL, or Structured Query Language, is a powerful domain-specific language designed for managing and manipulating

relational databases. It provides a standardized way to interact with databases, allowing users to perform operations such as querying, updating, inserting, and deleting data. SQL is used to define and manipulate the structure of relational databases, create and modify tables, and retrieve information based on specified

- criteria.

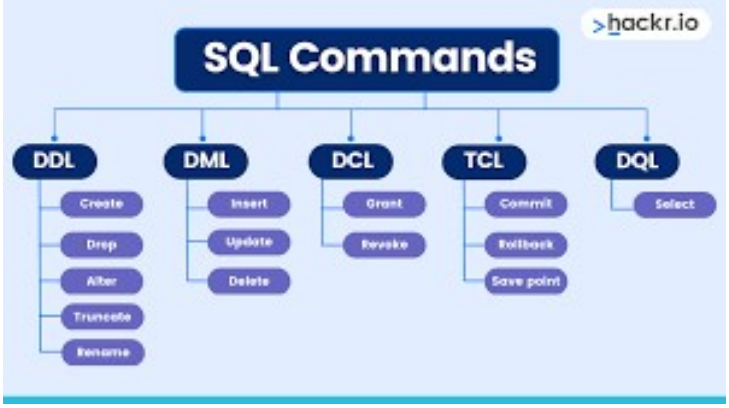

Fig:4.6 Types of SQL Commands

- **Storing and Accessing Data Using Keys with Data Store:** Data Store is a modern data storage solution provided by Android Jetpack for efficiently managing and persisting key-value pairs. It offers a more robust and type-safe alternative to the traditional Shared Preferences, making it well-suited for storing app preferences and small amounts of data.
- **Key-Value Storage:** Data Store operates on the principle of storing data as key-value pairs. Each piece of data is associated with a unique key, allowing for easy retrieval and updates.

- **Data Store Types:** There are two main types of Data Store: Preferences Data Store and Proto Data Store. Preferences Data Store: Suitable for storing simple key-value pairs, similar to Shared Preferences. It supports storing and retrieving data using typed keys.

- **Proto Data Store:** Ideal for complex data structures and objects. It uses Protocol Buffers for serialization, enabling the storage of structured data.
- **Coroutines Integration:** Data Store seamlessly integrates with Kotlin Coroutines, making it easy to perform asynchronous operations for storing and retrieving data without blocking the main thread.
- **Type Safety:** Unlike Shared Preferences, Data Store provides type safety, ensuring that the data retrieved is of the expected type. This helps reduce runtime errors related to data type mismatches.

Module 7: Work Manager

Google's Android Work Manager is a powerful API within the Android Jetpack library designed to simplify and manage background tasks in Android applications. It addresses the need for executing tasks that continue running even when the app is not in the foreground or if the device restarts. Here's a brief summary of Work Manager's key features:

Background Task Management:

Work Manager allows developers to schedule and manage tasks that run in the background, such as data syncing, periodic updates, or content downloads.

Persistent Execution:

Tasks scheduled with Work Manager persist across device reboots and app closures, ensure in greliable execution even in challenging conditions.

Simplified API:

Work Manager provides a simplified and consistent API, abstracting away the complexity of managing background tasks. It is built on top of existing Android background job mechanisms, offering a unified approach.

Flexible Scheduling:

Developers can schedule tasks as one-time or periodic, offering flexibility based on the nature of the background work needed.

Constraints Management: Work Manager allows the specification of constraints, such as network availability or battery status, to control when a task should be executed, optimizing resources and improving efficiency.

Integration with Other Jetpack Components:

It seamlessly integrates with other Android Jetpack components, facilitating the development of robust and modular applications.

Debugging Tools: Work Manager comes equipped with a Background Task Inspector in Android Studio, providing tools for monitoring and debugging background tasks effectively during development.

Result Handling: Developers can handle the results of background tasks using the Result object, allowing for proper handling of success, failure, or retry scenarios.

Module 8: Views and Compose

A View is a fundamental element for any user interface (or design) in android. The View is a base class for all UI components in android. For example, the Edit Text class is used to accept the input from users in android apps, which is a subclass of View.

Following are some of the common View subclasses that will be used in android applications.

- Text View
- Edit Text
- Button
- Check Box
- Radio Button
- Image Button

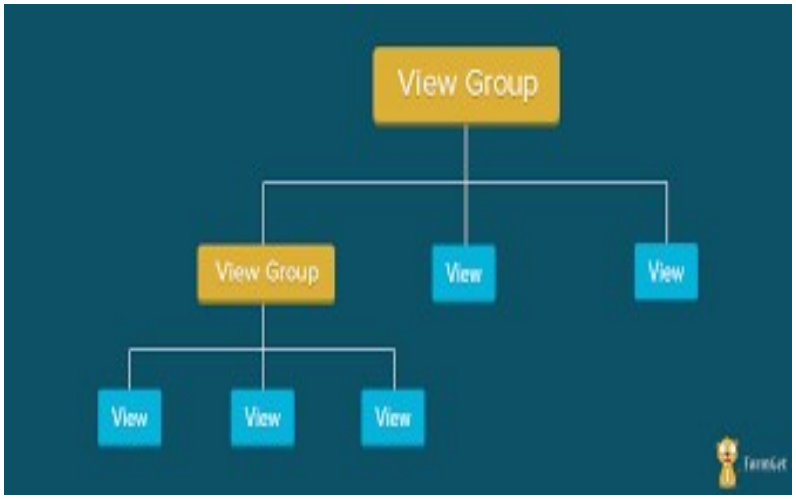

Fig:4.7 Different Views in Android

CHAPTER 5

REAL TIME EXAMPLES

Some top companies that use android in various use cases and industries such as business, healthcare and more. Some of the real time applications are:

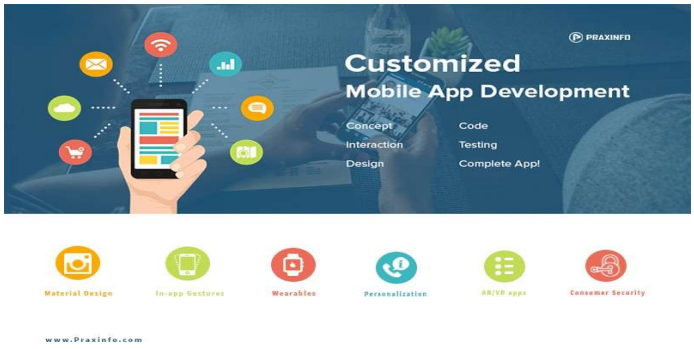

Fig:5.1 Use Cases of Android

1.Ride-Sharing Apps (e.g., Uber, Lyft)

Users can request a ride and track the real-time location of their driver. The app provides continuous updates on the driver's location, estimated time of arrival, and the route taken**.**

2.Food Delivery Apps (e.g., Door Dash, Grub hub)

Customers can place orders for food delivery and receive real-time updates on the status of their order, including confirmation, preparation, and the delivery process.

3.Weather Apps with Real-Time Updates (e.g., AccuWeather)

Weather apps provide real-time updates on current weather conditions, forecasts, and alerts based on the user's location.

4.Home Security Apps with Live Camera Feeds (e.g., Nest, Ring)

Users can view live camera feeds of their home security cameras in real-time, receive alerts for motion detection, and even communicate with visitors through two-way audio.

5.Emergency Services Apps (e.g., SOS Apps)

Apps designed for emergencies can provide real-time location tracking and communication features. Users can send distress signals with their precise location to emergency contacts or services.

6.Health and Fitness Apps with GPS (e.g., Strava, Run keeper)

Fitness apps utilize real-time GPS tracking to monitor and record users' routes, distances, and speeds during activities like running or cycling.

7.Instant Messaging Apps (e.g., WhatsApp, Telegram)

These apps facilitate real-time communication through instant messaging, supporting features like read receipts, typing indicators, and multimedia sharing.

8.Real-Time Language Translation Apps (e.g., Google Translate)

Apps that provide instant translation services, allowing users to translate spoken or written words in real-time.

9.Live Auction Apps (e.g., eBay)

Auction apps allow users to participate in real-time bidding on items, with immediate updates on the current highest bid and auction countdown**.**

10.Live Event Streaming Apps (e.g., Facebook Live, Instagram Live)

Users can broadcast live videos to their followers in real-time, and viewers can engage by sending comments and likes as the stream unfolds.

CHAPTER - 6

LEARNING OUTCOMES

By the end of this we will be able to:

- Gain an overall understanding of basic android.
- Be familiar with shared preferences and layouts.
- Learn the architectural principles of the android.
- Understand and android activities building apps.
- Engage in hands-on practice to create apps**.**

CHAPTER - 7

CONCLUSION

Enrolling in Google's Android development course provides a concise yet comprehensive journey into the world of mobile app creation. With Google's expertise in developing the Android platform, learners can expect a focused curriculum covering essential topics like programming languages, the Android SDK, and API integration. The course's hands-on approach ensures a practical understanding of building robust applications, aligning with industry standards. Google's direct involvement assures access to up-to-date content, reflecting the latest trends and tools in Android development. Completing this course not only equips individuals with the skills necessary for app creation but also stands as a valuable endorsement from a leading authority in the mobile technology landscape.

CERTIFICATE

REFERENCES

[1] https://developer.android.com/courses/android-basics-compose/course

[2] https://internship.aicte-india.org