A report on

INTERNSHIP

Google Android Developer Virtual Internship

Submitted in partial fulfillment of the requirements

for the award of the degree of

BACHELOR OF TECHNOLOGY

in

Computer Science and Engineering

(Artificial Intelligence & Machine Learning)

by

K.Rashmi

(224g1a3379)

Department of Computer Science and Engineering

(Artificial Intelligence & Machine Learning)

2023-2024

Srinivasa Ramanujan Institute of Technology

(AUTONOMOUS)

Rotarypuram Village, B K Samudram Mandal, Ananthapuramu - 515 701

Srinivasa Ramanujan Institute of Technology
(AUTONOMOUS)
Rotarypuram Village, B K Samudram Mandal, Ananthapuramu - 515 701

Department of Computer Science & Engineering (AI & ML)

Certificate

This is to certify that the internship report entitled Google Android Developer Virtual Internship is the bonafide work carried out by **K.Rashmi** bearing Roll Number **224g1a3379** in partial fulfilment of the requirements for the award of the degree of **Bachelor of Technology** in **Computer Science and Engineering (Artificial Intelligence & Machine Learning)** for 10 weeks May - July 2023.

Internship Coordinator

Mr. K. Lokeshnath, M.Tech, (Ph.D)
Assistant Professor

Head of the Department

Dr. P. Chitra Lingappa, M.Tech, (Ph.D)
Associate Professor & HOD

Date:

EXTERNAL EXAMINER

Place: Ananthapuramu

PREFACE

All India Council for Technical Education (AICTE) has initiated various activities for promoting industrial internship at the graduate level in technical institutes and Eduskills is a Non-profit organization which enables Industry 4.0 ready digital workforce in India. The vision of the organization is to fill the gap between Academic and Industry by ensuring world class curriculum access to the faculties and students. Formation of the All-India Council for Technical Education (AICTE) in 1945 by the Government of India.

Purpose: With a vision to create an industry-ready workforce who will eventually become leaders in emerging technologies, EduSkills & AICTE launches 'Virtual Internship' program on Process Mining. This field is one of the most in-demand, and this internship will serve as a primer.

Company's Mission Statement: The main mission of these initiatives is enhancement of the employability skills of the students passing out from Technical Institutions

ACKNOWLEDGEMENT

The satisfaction and euphoria that accompany the successful completion of any task would be incomplete without the mention of people who made it possible, whose constant guidance and encouragement crowned our efforts with success. It is a pleasant aspect that I have now the opportunity to express my gratitude for all of them.

It is with immense pleasure that I would like to express my indebted gratitude to my internship coordinator **Mr. K. Lokeshnath, M.Tech, (Ph.D), Assistant Professor, Department Of Computer Science and Engineering**, who has supported me a lot and encouraged me in every step of the internship work. I thank him/her for the stimulating support, constant encouragement and constructive criticism which have made possible to bring out this internship work.

I am very much thankful to **Dr. P. Chitralingappa, M.Tech, (Ph.D),Associate Professor, Department Of Computer Science and Engineering (Artificial Intelligence and Machine Learning),** for his/her kind support and for providing necessary facilities to carry out the internship.

I wish to convey my special thanks to **Dr. G. Balakrishna, Principal** of **Srinivasa Ramanujan Institute of Technology** for giving the required information in doing my internship. Not to forget, I thank all other faculty and non-teaching staff, and my friends who had directly or indirectly helped and supported me in completing my project in time.

I also express our sincere thanks to the Management for providing excellent facilities and support**.**

Finally, I wish to convey my gratitude to my family who fostered all the requirements and facilities that I need.

K.Rashmi

(224g1a3379)

CONTENTS

LIST OF FIGURES

LIST OF ABBREVIATIONS

API	Application Programming Interface
URL	Uniform Resource Locator
SDK	Software Development Kit
SQL	Structured Query Language
XML	Extensible Markup Language
JDK	Java Development Kit
IDE	Integrated Development Environment
UI	User Interface
UX	User Experience
HTTP	Business Process Intelligence
REST	Representational State Transfer

CHAPTER I
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 in flexible platform that works with a variety of devices became of as 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 sand boxing, 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 desert 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 name 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 Whats App 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

5. Live Score Apps:

- Description: Sports apps like ESPN or Score provide real-time updates on live sports events.
- Key Characteristics:
    - Instant score updates
    - Real-time statistics
    - Live commentary

6. Real-Time Collaboration Tools:

- Description: Apps like Slack or Microsoft Teams facilitate real-time communication and collaboration within teams.
- Key Characteristics:
    - Instant messaging and file sharing
    - Real-time status updates
    - Notification alerts

7. Financial Trading Apps:

- Description: Trading platforms like Robinhood or E*TRADE require real-time updates on stock prices and trade execution.
- Key Characteristics:
    - Real-time market data
    - Instant order execution
    - Live portfolio updates

8. Real-Time Gaming Apps:

- Description: Multiplayer games like PUBG Mobile or Among Us involve real-time interactions between players.
- Key Characteristics:
    - Low-latency gameplay
    - Real-time chat and voice communication
    - Synchronized game events

CHAPTER 4

MODULES 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/in/java/technologies/downloads/#jdk22-windows

IDE Download:

https://developer.android.com/studio

Module 2: Building app UI

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

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.

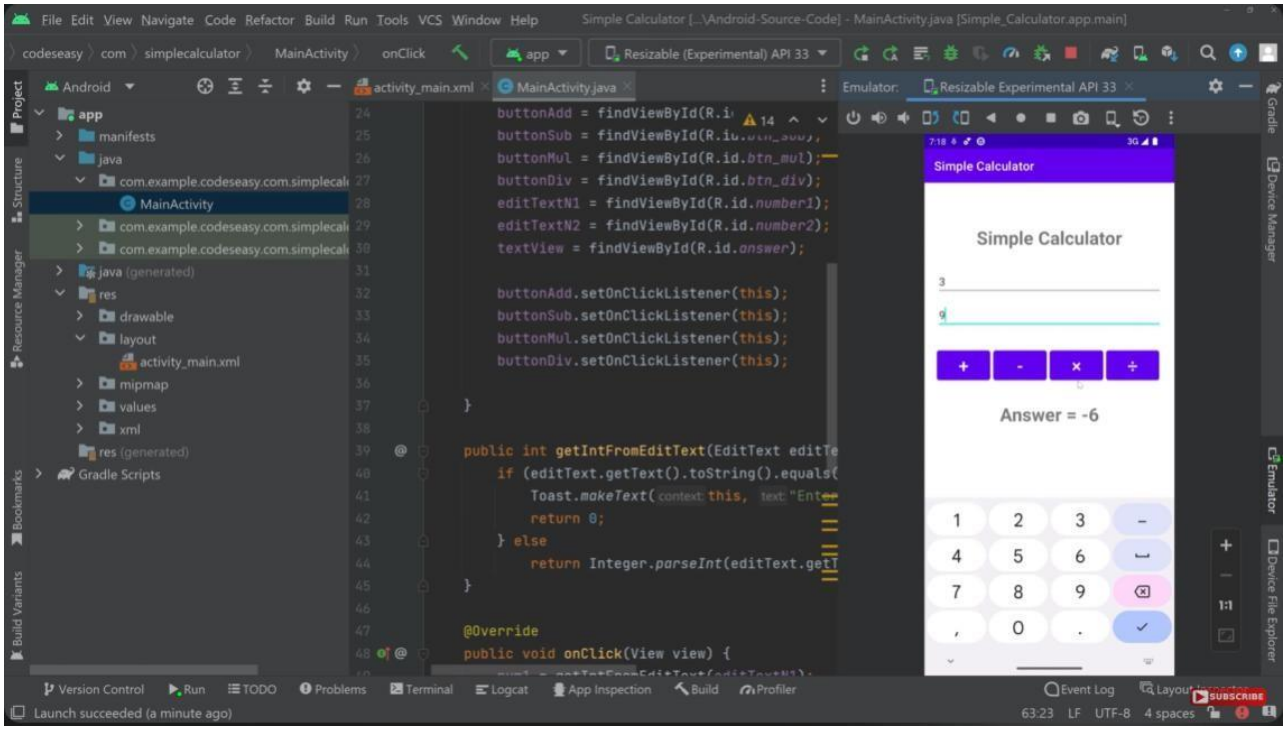

Fig:4.2 Building calculator using button widget

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,learner can develop user-friendly app.

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 co-routines, 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.

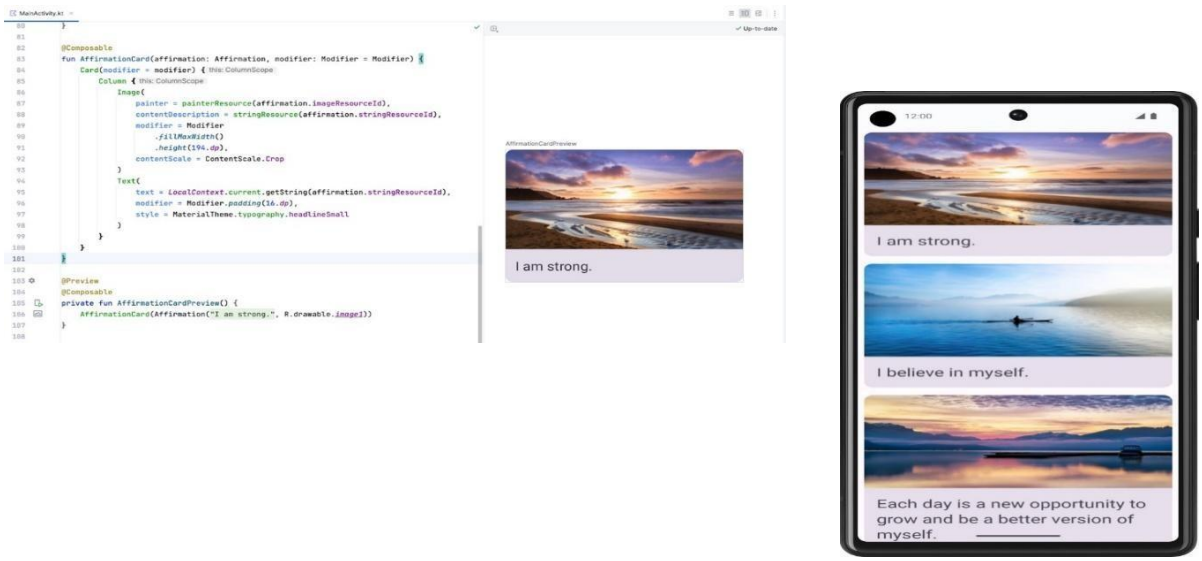

Fig 4.3: Material Design

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 videos 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 Navigation and App Architecture

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 Internet

Connecting to the internet is essential for most modern applications, allowing them to communicate with remote servers and access online services or resources. This functionality enables apps to retrieve data, interact with APIs, and synchronize information across devices.

1. Network Permission

Before an application can access the internet, it needs explicit permission from the operating system. This is typically done by declaring the necessary permissions within the app's configuration files. In Android development, for example, apps must request internet access in the AndroidManifest.xml. This step is mandatory, as it ensures the user is aware that the app will be using the internet, which could involve data usage or privacy implications. Without this permission, the app won't be able to establish any network connection, regardless of the logic written in the code.

2. Network Requests

Once the necessary permissions are in place, the application can begin making network requests to interact with external servers or APIs. There are multiple ways to establish a network connection:

HTTP and HTTPS:

These are the primary protocols used for internet communication. HTTP (Hypertext Transfer Protocol) is commonly used for standard web requests, while HTTPS (HTTP Secure) ensures that the communication is encrypted for security purposes. HTTPS is particularly important when dealing with sensitive data like user credentials or payment information.

HTTPURLConnection:

In Java (and by extension, Android development), HttpURLConnection is a class used for sending requests to a server and handling the responses. This approach allows developers to create a connection, configure it (e.g., setting timeouts or request methods), and then handle the data returned by the server.

HttpClient:

This was another common way of sending network requests in earlier versions of Android, offering more advanced functionality than HttpURLConnection. However, it has since been deprecated, meaning it's no longer recommended for use in new development projects.

3. Fetching Data from the Internet

A common use case for connecting to the internet is to fetch data, whether from a public API, a web service, or a remote server. The process involves the following steps:

Establishing a Connection:

The first step in fetching data is to establish a connection with the server. In Java-based environments, classes like URL and HttpURLConnection are used to create the connection. This involves specifying the URL of the target endpoint and setting up the connection for sending or receiving data.

Handling the Response:

Once the connection is made, the next step is to handle the server's response. This could involve reading the data returned by the server (usually in formats like JSON or XML) and then processing it for use within the application, such as displaying it to the user or storing it in a local database.

4. Choosing the Right HTTP Method

Different types of network requests are made using different HTTP methods. The method you choose depends on the kind of operation you need to perform on the server:

GET: This method is used to retrieve data from a server. It's the most common HTTP method, often used to request data from APIs or display web content.

POST: When you need to send data to the server (such as submitting form data or creating a new resource), the POST method is used. This is a more secure method than GET for sending sensitive information because the data isn't included in the URL.

PUT or PATCH: These methods are used to update existing data on the server. While PUT typically replaces the entire resource, PATCH is used for partial updates.

DELETE: As the name suggests, this method is used to remove a resource from 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.gradle 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 ImageView size to optimize memory usage and improve loading speed.

Fig:4.5 Connect to 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 SharedPreferences, SQLite databases, or file storage.

Remote Storage: Storing data on remote servers, typically accessed through APIs.

Local Data Persistence:

SharedPreferences: 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.

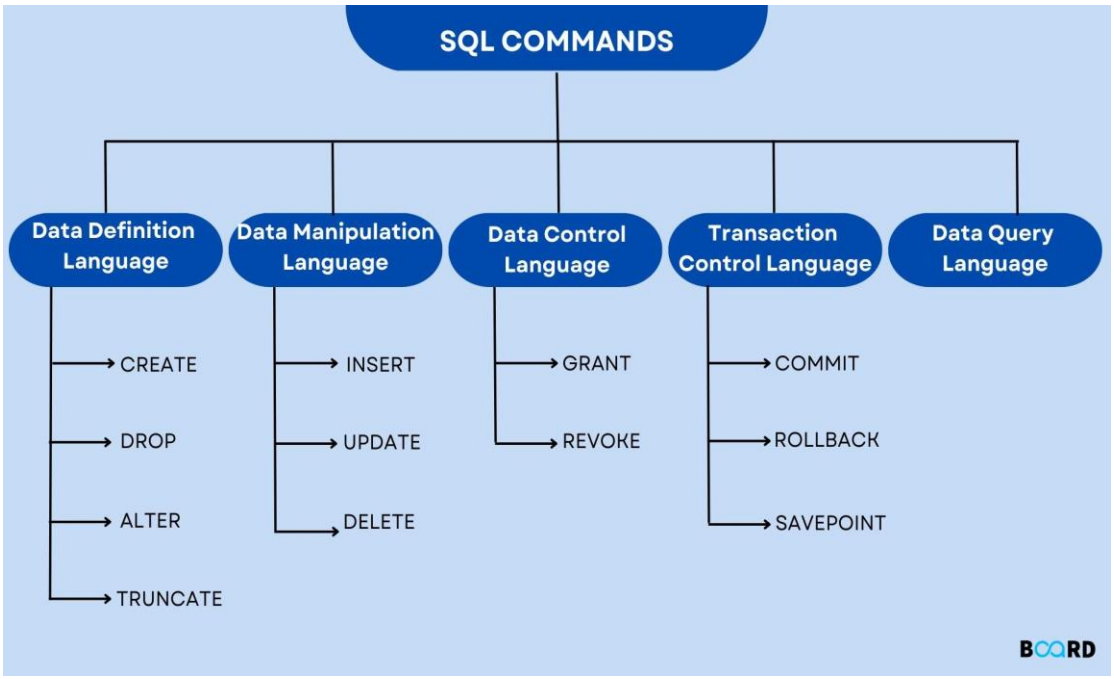

Fig:4.6 Types of SQL Commands

Storing and Accessing Data Using Keys with DataStore

DataStore 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 SharedPreferences, making it well-suited for storing app preferences and small amounts of data.

Fig:4.7 Key Value Pairs

Key-Value Storage:

DataStore 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.

DataStore Types:

There are two main types of DataStore: Preferences DataStore and Proto DataStore.

Preferences DataStore: Suitable for storing simple key-value pairs, similar to SharedPreferences. It supports storing and retrieving data using typed keys.

Proto DataStore: Ideal for complex data structures and objects. It uses Protocol Buffers for serialization, enabling the storage of structured data.

Coroutines Integration:

DataStore 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 SharedPreferences, DataStore 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: 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 EditText class is used to accept the input from users in android apps, which is a subclass of View. Following are the some of common View subclasses that will be used in android applications.

- TextView
- EditText
- Button
- CheckBox
- RadioButton
- ImageButton

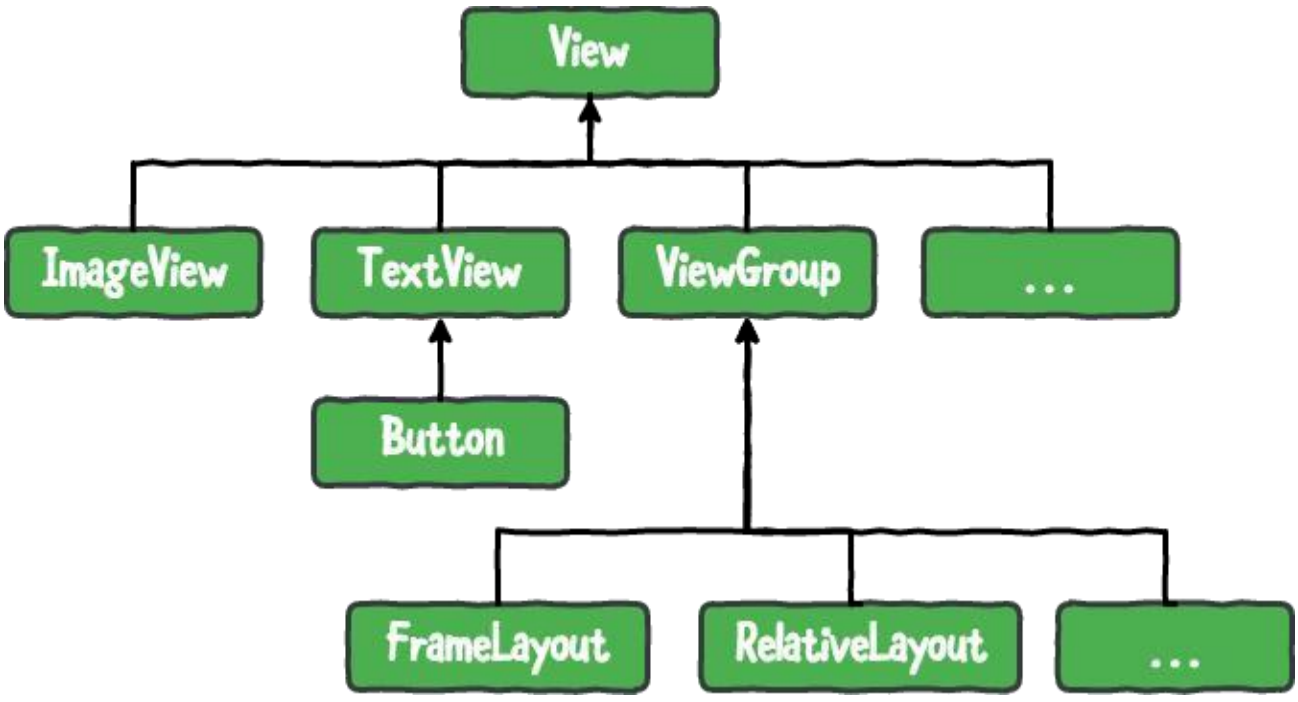

Fig:4.8 Different types of Views

Android Views and Compose in Views

Learning about Android Views involves using XML layouts and UI components like TextViews and Buttons to build interfaces. To create an app, set up Android Studio, design layouts in XML, and develop activities and fragments with Java or Kotlin. Integrate modern Jetpack Compose components for hybrid UIs, then test, debug, and deploy your app.

Compose in Views:

Refers to the ability to integrate or use traditional Android Views (defined in XML) within a Compose-based UI. This approach allows you to use existing View-based components and layouts alongside Compose's declarative UI elements.

CHAPTER 5

REAL TIME EXAMPLES

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

Fig:4.9 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., DoorDash, Grubhub)

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- t ime 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, Runkeeper)

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 course, learners will be able to:

- Manage app resources such as images, strings, and styles for scalable apps across multiple devices.
- Work with Recycler Views to display dynamic data efficiently.
- Handle user interactions through input controls, gestures, and notifications.
- Utilize SQLite for local data persistence and access web-based data via RESTful APIs and JSON.
- Implement Material Design principles for visually appealing and intuitive user interfaces.
- Integrate third-party libraries into Android applications.
- Utilize background services for tasks like notifications and alarms.
- Understand Android's permissions system to ensure user privacy and security.
- Distribute apps through Google Play, including managing app versions and signing apps.
- Implement in-app purchases and advertisements for app monetization.
- Develop, test, and deploy Android apps efficiently with hands-on experience.

CONCLUSION

Enrolling in Google's Android development course offers a clear and thorough introduction to mobile app development. Learners benefit from a curriculum that covers key areas such as programming languages, the Android SDK, and API integration. With a strong emphasis on hands-on learning, the course provides practical experience in building robust applications that meet industry standards. Since the course is backed by Google, students also have access to the latest tools and trends in Android development. Completing the program equips individuals with essential app creation skills and adds a valuable credential from a top player in mobile technology.

Furthermore, the course prepares learners to tackle real-world challenges by focusing on both the technical and creative aspects of app development. Participants gain a deep understanding of designing user-friendly interfaces, optimizing app performance, and ensuring compatibility across a wide range of devices and screen sizes. With guidance from industry experts, students learn how to debug, test, and refine their apps to meet the highest standards of quality. In addition to technical skills, the course fosters problem-solving abilities and encourages innovative thinking, enabling developers to create unique, impactful applications. By the end of the program, graduates are not only equipped with the knowledge to build functional apps but also with the confidence to navigate the evolving Android ecosystem and contribute to the growing mobile app industry.

INTERNSHIP CERTIFICATE

REFERENCES

1. https://developer.android.com/
2. https://developer.android.com/studio
3. https://developer.android.com/courses/android-basics-compose/course