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 & Engineering

by

HEMANTH REDDY K **(214G5A0506)**

Department of Computer Science & Engineering

Srinivasa Ramanujan Institute of Technology

(AUTONOMOUS)

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

2023-2024

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

Department of Computer Science & Engineering

Certificate

This is to certify that the internship report entitled Google Android Developer Virtual Internship is the bonafide work carried out by **Hemanth Reddy K** bearing Roll Number 214G5A0506 in partial fulfillment of the requirements for the award of the degree of **Bachelor of Technology** in **Computer Science and Engineering** for three months from January 2024 to March 2024.

Internship Coordinator
Mr. K. Lokeshnath MTech.,(Ph.D.)
Assistant Professor

Head of the Department
Mr. P. Veera Prakash M.Tech, (Ph.D)
Assistant Professor

Date:

EXTERNAL EXAMINER

Place: Anantapuramu

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) in1945by 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 Android Technology, supported by Google.

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 in complete 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 in debted gratitude to my internship coordinator M**r. K. Lokeshnath,** **M.Tech,(Ph.D,)** **Assistant Professor, Computer Science and Engineering** who has supported me a lot and encouraged me in every step of the internship work. I thank him 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 **Mr. P. Veera Prakash,** **M.Tech,(Ph.D),** **Assistant Professor and Head of the Department, Computer Science and Engineering,** for his kind support and for providing necessary facilities to carry out the work.

I wish to convey my special thanks to **Mr. G. Balakrishna,** **M.Tech,Ph.D,** **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 internship 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.

Hemanth Reddy K

(214G5A0506)

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
HTTP	HyperText Transfer Protocol

CHAPTER 1

INTRODUCTION

The Android operating system was developed by Google and first made available in 2008. Its continuous improvement is encouraged by the cooperative development community, since it is built on the Linux kernel. Android is a flexible platform for a variety of apps because it is open source and can be tailored and adapted to a wide range of devices.

The development of Android is greatly aided by Google, which provides services, updates, and necessary software. Customers can browse and download millions of programs through the official Android app distribution platform, the Google Play Store. Android's touch gesture design, customisable home screen, and widget support all contribute to a natural and engaging user experience.

The two primary programming languages utilized in Android application development are Java and Kotlin. The Android Software Development Kit (SDK) facilitates the creation of applications by providing a vast array of APIs. There is a wide range of genres available in the Android app store, from productivity tools and games to social media and entertainment apps.

Android places a high value on allowing users to customize their experience. This includes personalizing the interface using themes, wallpapers, and launchers. To have even more access and control over the operating system, advanced users can even root their devices. Android places a lot of emphasis on security, including measures like permissions control, app sandboxing, and frequent security upgrades to safeguard user information.

Regular updates to Android bring security patches, new features, and improvements. In alphabetical order, each major version of Android is named after a dessert or confection. In essence, Android's large app ecosystem, flexibility, and commitment to regular updates all contribute to its great popularity in the mobile technology sector.

CHAPTER 2
TECHNOLOGY

1. To provide a flexible and reliable base for mobile devices, the Android operating system makes use of a variety of technologies. To succeed in an Android development position, you'll need a combination of hard abilities, soft skills, and in-depth knowledge of the Android ecosystem. The ensuing are crucial proficiencies that will facilitate your success in an Android position:

2. The Coding Languages Kotlin and Java

A solid grasp of Java is essential to Android programming. Because of its safety features, expressiveness, and conciseness, Kotlin has garnered official Google sponsorship since its launch by JetBrains and is growing in popularity. When you are proficient in both languages, you may utilize each one's benefits based on the needs of the project.

3. The APIS and Android SDK

It is essential to have a thorough understanding of the Android Software Development Kit (SDK) and know how to use Android APIs. Proficiency with UI elements, data storage (SharedPreferences, SQLite, or Room), and the use of functionalities like permissions and notifications are all part of this.

4. Android Studio

Fundamental knowledge of the Android Software Development Kit (SDK) and experience with Android APIs are prerequisites. Understanding how to use data storage (SharedPreferences, SQLite, or Room) and features like notifications and permissions is required.

5. XML (Extensible Markup Language)

Android defines its user interface (UI) elements and layouts using XML. A developer must be proficient in both creating and understanding XML layouts in order to produce user interfaces that are both aesthetically beautiful and responsive.

6. Git and Version control

A popular version control system in team-based software development is called Git. Being proficient in Git is essential for efficient code management and collaboration, including branching, merging, and conflict resolution.

7. Gradle build system

The Gradle build system is used for Android projects. App builds that are effective and development workflows that are seamless are guaranteed by knowing how to optimize build processes, manage dependencies, and set build scripts.

8. 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.

9. Collaboration and Communication

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 software engineering with a focus on creating applications for Android-powered devices. The following are some uses for Android development:

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

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
MODULE EXPLANATION

Module-1: Your first Android app

The module provides information about Android and its prerequisites. The prerequisites cover installing the Android Studio, learning the Kotlin programming language, and creating a simple layout.

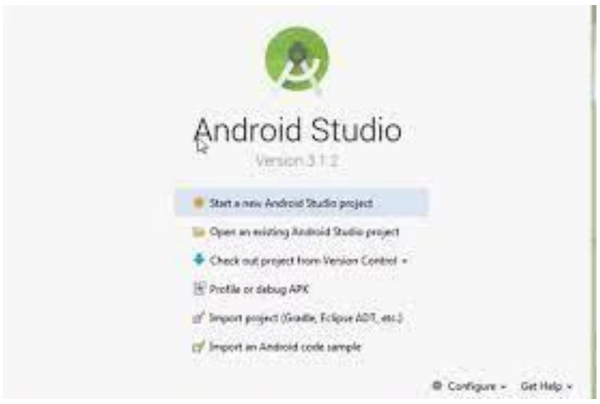

Android Studio
Version 3.1.2
Start a new Android Studio project
Open an existing Android Studio project
Check out project from Version Control
Profile or debug APK
Import project (Gradle, Eclipse ADT, etc.)
Import an Android code sample
Configure
Get Help

Fig:4.1 Setting up android studio

The preferred choice for developing Android apps is now the simplified, cutting-edge programming language Kotlin. Google-approved, Kotlin offers a seamless interface for combining with existing Java code while adding several benefits like more readable code, null safety, and concise syntax. It accelerates development and is a fantastic option for Android projects due to its expressiveness and usefulness. Since 2017, Kotlin has emerged as the standard language for Android app development. It reduces boilerplate code, boosts developer productivity, and facilitates the creation of Android apps that are more dependable, efficient, and manageable.

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.

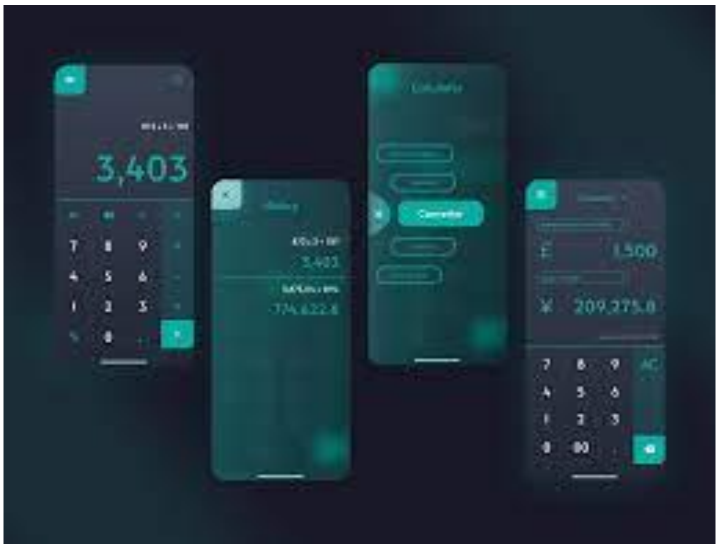


Fig:4.2 Building calculator using button widget

One of the fundamental features of the modern, concise programming language, object-oriented programming (OOP) is strongly supported by Kotlin's key ideas. When integrating with existing Java code, Kotlin offers several advantages such as null safety, readable code, and straightforward syntax. Inheritance, polymorphism, and encapsulation—three fundamental Kotlin concepts—allow programmers to create modular, manageable code. Another concept that Kotlin provides is lambdas, which allow for concise and expressive functional programming. Writing anonymous functions is made simpler by lambdas, which enhances code readability and promotes a more functional approach to Kotlin programming.

The development of a tip calculator application follows, with an emphasis on state management and user interface interaction. The app's capacity to compute suggestions based on user input serves as an example of fundamental concepts in state management and user interface design. We explore techniques for obtaining user input, dynamically updating the user interface, and managing the application's state during development to ensure a seamless and responsive user experience. Students will have a solid basis for creating user-friendly apps after completing this course, along with comprehensive understanding about handling states and creating dynamic designs.

Module 3: Utilize Material Design and Show Listings

Starting with a brief synopsis, this summary presents essential Kotlin programming ideas that developers need know in order to create dynamic and captivating Android apps.

Fig:4.3 Material Design

Kotlin's asynchronous programming capabilities, facilitated by coroutines, and its easily comprehensible syntax enable developers to create feature-rich, fast, and adaptable applications. Together with sophisticated features like higher-order functions and extension functions, the language's smooth interoperability with Java encourages expressive coding styles, which improves the development process overall. Furthermore, Kotlin's strong type system and support for null safety lead to more dependable code, which provides a basis for developing complex and entertaining Android applications.

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: App architecture and Navigation

Fig:4.4 Jetpack compose

This module offers guidance on how to use the features of the Navigation component to create multi-screen, complicated Android applications. Learn the art of seamlessly switching between different elements and moving data between screens. Through the investigation of this module, developers will acquire the capacity to create more complex and networked apps, enhancing user experiences through lucid and systematic navigation procedures. Learning to use the features of the Navigation component ensures a more effective way to manage a range of app architectures, which encourages 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.

Taking a deep dive into the Navigation component with the help of this thorough documentation, which is intended to enable developers to create complex multi-screen Android applications. In-depth techniques for fluidly navigating between various components are covered in this subject, which also offers subtle insights into the design principles and hierarchical structure that support an integrated user journey. Furthermore, developers will get extensive understanding about the efficient transfer and administration of data between displays, stressing effectiveness and upholding a robust architecture. This tool is designed to assist developers in mastering the more complex features of the Navigation component, allowing them to create dynamic, networked applications that react to user input in real time, resulting in a more sophisticated and customized 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:

HTTPURLConnection: A basic API provided by Java for sending HTTP requests and receiving responses.

HttpClient: 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 URLConnection or HttpURLConnection 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.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 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 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.

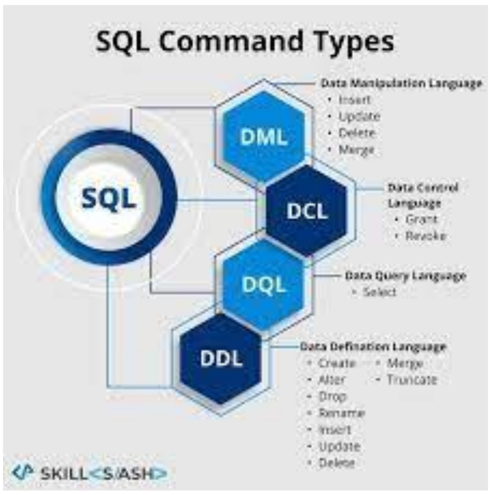


Fig:4.6 Types of SQL Commands

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.

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: WorkManager

Google's Android WorkManager 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 WorkManager's key features:

Background Task Management:

WorkManager 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 WorkManager persist across device reboots and app closures, ensuring reliable execution even in challenging conditions.

Simplified API:

WorkManager 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:

WorkManager 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:

WorkManager 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 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

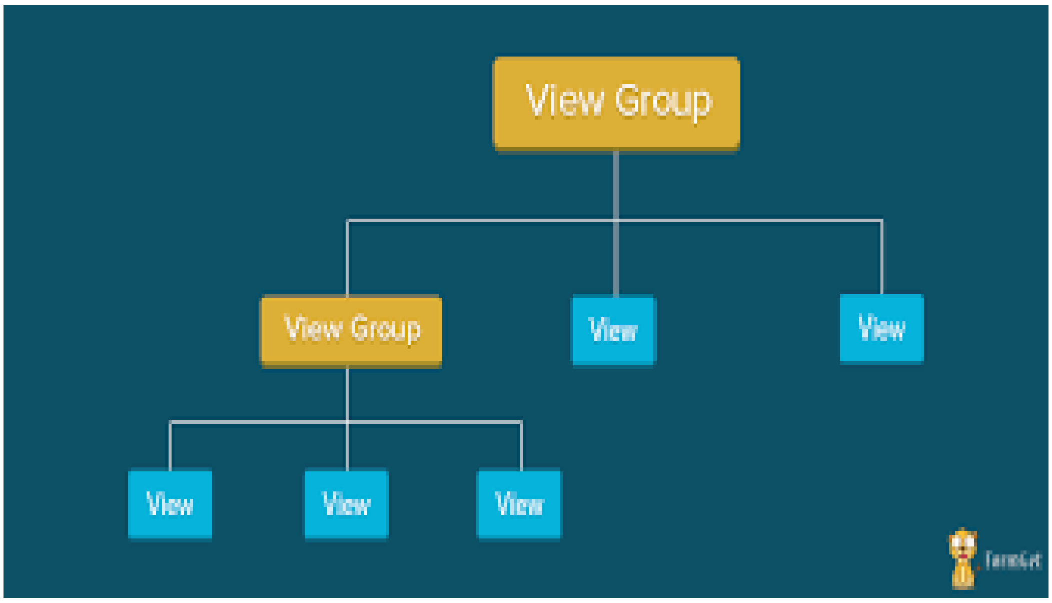


Fig:4.7 Different Views in Android

CHAPTER 5

REAL TIME EXAMPLES

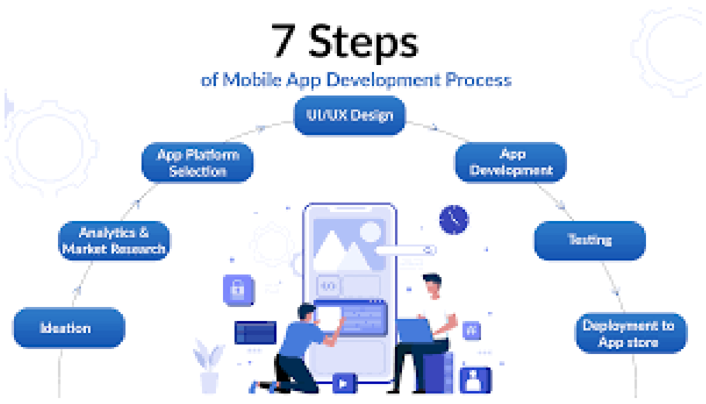


Fig:5.1 Use Cases of Android

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:

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-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, 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

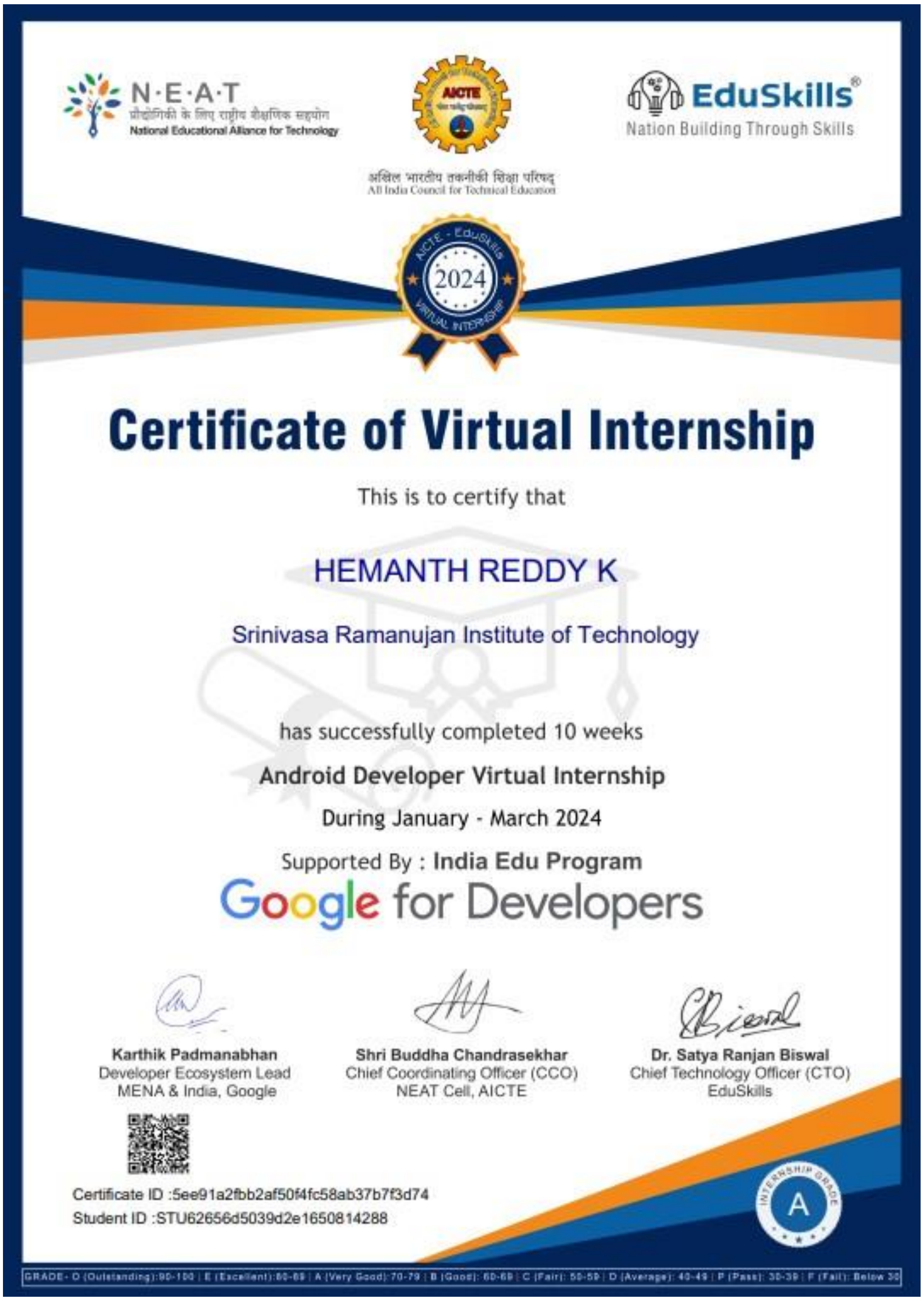
By the end of this we will be able to know about:

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

CONCLUSION

Engaging in an accessible and comprehensive introduction to the world of developing mobile applications may be found in Google's course on Android development. Students may expect a curriculum that is specifically targeted and includes crucial areas like Android SDK, programming languages, and API integration because of Google's experience developing the Android platform. Students will have a practical understanding of creating robust apps that adhere to industry standards thanks to the course's practical approach. Because Google is directly involved, the content is always up to date and represents the latest methods and tools for Android development. After finishing the course, students will not only have the abilities needed to make apps, but they will also obtain an important recommendation from a well-known authority in the mobile technology industry.

CERTIFICATE



N·E·A·T
प्रौद्योगिकी के लिए राष्ट्रीय शैक्षणिक सहयोग
National Educational Alliance for Technology

AICTE

अखिल भारतीय तकनीकी शिक्षा परिषद्
All India Council for Technical Education

EduSkills®
Nation Building Through Skills

AICTE - EduSkills
2024
VIRTUAL INTERNSHIP

Certificate of Virtual Internship

This is to certify that

HEMANTH REDDY K

Srinivasa Ramanujan Institute of Technology

has successfully completed 10 weeks

Android Developer Virtual Internship

During January - March 2024

Supported By : India Edu Program

Google for Developers

[illegible]
Karthik Padmanabhan
Developer Ecosystem Lead
MENA & India, Google

[illegible]
Shri Buddha Chandrasekhar
Chief Coordinating Officer (CCO)
NEAT Cell, AICTE

[illegible]
Dr. Satya Ranjan Biswal
Chief Technology Officer (CTO)
EduSkills

Certificate ID :5ee91a2fbb2af50f4fc58ab37b7f3d74
Student ID :STU62656d5039d2e1650814288

INTERNSHIP GRADE
A

GRADE- O (Outstanding):90-100 | E (Excellent):80-89 | A (Very Good):70-79 | B (Good): 60-69 | C (Fair): 50-59 | D (Average): 40-49 | P (Pass): 30-39 | F (Fail): Below 30

REFERENCES

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

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