Android Development-Curriculum

1.INTRODUCTION TO ANDROID

1.1 Introduction to Mobile Development

1.2 Native vs Cross Platform

Native apps are available for download exclusively in platform-dedicated app stores, such as Google Play Store for Android apps and Apple App Store for iOS apps. Cross-platform and hybrid apps can be made available and promoted on multiple app stores.

1.3 Installing Android Studio

1.4 Android Studio Overview

- Integrated Development Environment (IDE)
- XML attributes

1.5 Basic Android Application

2.ANDROID COMPONENTS

2.1 Android Activity Lifecycle

- Lifecycle states
- Callback methods

2.2 Android App Components

3.CREATING AN APPLICATION IN ANDROID

SMS Application - Part 1

SMS Application - Part 2

SMS Application - Part 3

4.KOTLIN

4.1 Kotlin vs Java

- Syntax - Kotlin has a modern, concise syntax that reduces bugs and boilerplate. Java's syntax is mostly derived from C and C++.
- Performance - Kotlin and Java have similar performance, but Kotlin may have an advantage for small programs. Java code is typically compiled faster than Kotlin

4.2 Variables in Kotlin

Operators in Kotlin

- Unary operators - These operators require a single operand and operate on it in place.
- Arithmetic operators - These operators include Modulus, Subtraction, Multiplication and Division

4.3 Control Flow Part - 1

4.4 Collections

Android Development-Curriculum

4.5 Control Flow Part - 2 Loops

4.6 Functions

5. ADVANCE USER INTERFACE

5.1 Intents

- Explicit
- Implicit

5.2 Fragments

- Fragments Life Cycle
- Fragments Callbacks

5.3 Shared Preferences

- Allows activities and applications to keep preferences, in the form of key-value pairs similar to a Map that will persist even when the user closes the application.

5.4 List View

- Array
- View Binding

5.5 Recycler View

- Custom Model
- Liner Layout
- Array List

6.WORKING WITH DATABASE

6.1 Intro Android Architecture

- MVP
- MVC
- MVVM

6.2 Entity

6.3 DAO

6.4 Room Database

- Live Data
- Abstract Class
- Companion Object
- Synchronise

Android Development-Curriculum

6.5 Repository
- Clean API
- Note Repository

6.6 ViewModel

7.FIREBASE

7.1 What is Database

7.2 Introduction to Firebase
- Firebase Analytics
- App Connection

7.3 Connect App with Firebase (Manual)

7.4 Connect App with Firebase (Auto)

7.5 Firebase Authentication
- Liner Layout
- Lateinit VAR

7.6 Realtime Database
- How it Works
- How to Use it
- How to Emulate it

7.7 Firebase Storage

7.8 RetroFit

8.JETPACK

8.1 Text Column Row Basic

8.2 Instagram Ui Clone

9.ANDROID WITH MACHINE LEARNING

9.1 Machine Learning - Part 1
- TF HUBS
- Tensorflow Lite Model

9.2 Machine Learning - Part 2

Android Development-Curriculum

CAPSTONE PROJECTS

1 MUSIC APPLICATION DEVELOPED USING KOTLIN

- Kotlin for developing the app's core logic and UI.
- XML for designing the user interface (UI) layouts.
- Recycler View to display the list of songs in an efficient, scrollable format.
- Services for managing background music playback, even when the app is in the background.
- Foreground Tasks for keeping track of the currently playing song with notifications and controls, ensuring seamless user experience when switching between tasks.

2 IMDB CLONE

- Developed using Kotlin for clean and concise code.
- Integrated Retrofit for efficient REST API communication with the backend.
- Employed Coroutines for seamless asynchronous network operations without blocking the UI.
- Adhered to MVVM architecture to ensure scalability and maintainability of the codebase.
- Utilised RecyclerView for optimised and dynamic presentation of movie data in a scrollable list.

3 COIN TRACK

- Real-Time Cryptocurrency Data: Track current prices and trends of all major cryptocurrencies with live data updates.
- Favorites and Watchlist: Users can add coins to their favorites to monitor specific ones more easily.
- Historical Data and Price Trends: Visualize price trends over time to help users make informed decisions.
- MVVM Architecture: Built using the Model-View-ViewModel pattern for clean, maintainable, and scalable code.
- Networking with REST APIs: Fetch live market data through reliable API sources for accurate information.

LIVE PROJECT

1 BLOG APPLICATION

- Developed using Kotlin for smooth and efficient functionality.
- Implemented Retrofit for handling REST API requests.
- Integrated GET, POST, and DELETE methods to fetch, create, and delete blog posts.
- Used Authentication mechanisms to ensure secure access to the application's resources.
- Managed network requests using Coroutines to keep the UI responsive.
- Followed MVVM architecture for a clean, modular, and maintainable codebase.