Chapter-1: Introduction to Mobile OSes [7hr]

Prepared By: Hem Raj Bhattarai

Outline:

- Introduction to Mobile OSes
- Build and Structures of Mobile OSes
- Introduction to Development (Native v/s HTML5)
- Introduction to Android
- API levels/versions of Android
- Pros and Cons of Android
- Comparison of Android with other Mobile OSes
- Introduction to Android VM and Runtime (Dalvik and ART)
- Installation and configuration of Android SDKs
- Eclipse IDE Their integration using ADT Plugin
- Running an emulator
- Using the ADB command-line Interface

Introduction To Mobile OSes

- Mobile operating systems (OSes) are specialized software platforms designed to power smartphones, tablets, smartwatches, and other mobile devices
- They provide a framework for device hardware and software applications **to interact**, enabling users to perform various tasks, such as **making calls**, **sending messages**, **browsing the internet**, **running apps**, and more.
- Eg. Android, iOS, Windows

Introduction to Mobile OSes

Windows Phone

iOS

symbian OS

TIZEN

ubuntu touch
Powered by UBports Foundation

BlackBerry

android

Introduction to Mobile OSes

- Android
- iOS
- Ubuntu Touch
- Blackberry
- Tizen
- Firefox OS
- Symbian
- Windows Phone

Android

- Android is a mobile operating system developed by Google.
- It is based on the Linux kernel and designed primarily for touchscreen mobile devices such as smartphones and tablets
- Android OS, developed by Google, powers a vast array of mobile devices worldwide.
- It boasts customizable features, including home screens and widgets, tailored to users' preferences.

Android cont...

- The Google Play Store offers a vast selection of apps, catering to various needs and interests.
- Being open-source, Android encourages innovation and allows for customization by device manufacturers and developers.
- Regular updates ensure that Android devices receive new features and security patches, keeping them up-to-date and protected.

Android cont…

Android History

- 2003: Android Incorporation(California) By Andy Rubin and his Team.
- 2005: Google Acquired android company in 50 million dolor.
- 2007: Google created OHA(Open Handset Alliance) community.
- 2008: First Android mobile was launched, HTC.
- 2010: Google launched, Nexus Series.

Summary Android

- Android OS is an open-source, Linux-based, mobile operating system.

iOS

- iOS is a mobile operating system developed by Apple Inc.
- It's designed exclusively for Apple hardware, including iPhones, iPads, and iPod Touch devices.
- Since its initial release in 2007, iOS has gone through numerous updates and iterations, introducing new features, enhancements, and security improvements.
- It was Programmed in C, C++, and Objective C.
- It was derived from Mac OS
- It was initially released on June 200

iOS cont…


The Evolution Of iOS
iOS 1 - iOS 10
Over the past 10 years, iOS has changed and evolved tremendously. Let's take a look at the journey of iOS till today.
The Beginning of the iOS
Steve Jobs, the former CEO of Apple introduced the first iPhone in January 2007 and The journey of iOS started from that day.
iOS 1
Announced: Macworld 2007 - January 9, 2007
iOS 1, Apple's first-ever OS was launched on 9th January, 2007 with the announcement of iPhone.
It was announced with the features like multi-touch gestures, Safari, a YouTube App and Visual Voicemail.
Release: June 29, 2007 with iPhone
iTunes Music Store
Safari Mobile Web Browser
Email
Multitouch Software Keyboard
iPhone 2G
iOS 2
Announced: WWDC 2008 - June 2008
Release: June 11, 2008 with iPhone 3G
Apple announced the second major OS called iOS 2 on 11th July, 2008 in WWDC 2008 with the launch of iPhone 3G.
iOS 2 was announced with the features like App Store,Maps with GPS, Scientific calculator and 3rd party native apps support.
Support of iPhone OS 2 has ended in 2011.
GPS
App Store
Microsoft Exchange Support
Scientific Calculator
3rd Party Native Apps Support
iPhone 3G
iOS 3
Announced: WWDC 2009 - June 2009
Release: June 19, 2009 with iPhone 3G
Apple announced iOS 3 with the launch of iPhone 3G S on 17th June, 2009.
iOS 3 was announced with the many latest features MMS, spotlight search, a landscape keyboard, in-app purchase, voice control, tap to focus in camera, cut, copy and paste functions.
Support of iPhone OS 3 has ended in late 2012.
MMS
Camcorder
Voice Control
Spotlight Search
Cut, Copy and Paste
iPhone 3G S
iOS 4
Announced: WWDC 2010 - June 7, 2010
Release: June 21, 2010 with iPhone 4
Apple announced iOS 4 with the launch of iPhone 4 and iPad 2 on 21st June, 2010.
iOS 4 was announced with over 100 new features like FaceTime video calls, multitasking support, enhanced search functionality, zooming in camera and iBooks.
Folders
HD Recording + Autofocus
FaceTime
Game Center
Multitasking
iPhone 4
iOS 5
Announced: WWDC 2011 - June 6 , 2011
Release: Oct 12, 2011
Apple announced iOS 5 on 6th June, 2011 with more than 200 new features.
The flagship features of iOS 5 were Siri, Apple's virtual assistant. The other noteworthy features were iMessage, Newsstand, iCloud, Twitter integration, notification center and much more.
Siri
HD Recording + Autofocus
Newsstand
iCloud
iTune Wi-Fi Sync
iPhone 5
iOS 6
Announced: WWDC 2012 - June 11, 2012
Release: septe 19, 2012
Apple announced iOS 6 with the launch of iPhone 5 and iPad Mini at WWDC 2012 on 11th June.
iOS 6 included Apple's own Maps App and Passbook App.It was also included Facebook integration, LTE support, Panorama and many more features.
Panorama
Facetime Video Calling Over 3G/LTE
Apple-Sourced Maps + Turn-by-Turn Navigation
Passbook
Shared Photo Streams
iPhone 5
iOS 7
Announced: WWDC 2013 - June 10, 2013
Release: Septe 18, 2013
Apple announced iOS 7 at WWDC on 10th June, 2013. It was prepackaged with the iPhone 5S, 5C, iPad Air and iPad mini 2.
iOS 7 announced with an entirely new look and feel. iOS 7 introduced us to the iTunes Radio, Control Center, AirDrop, revamped Photos app and CarPlay.
Touch ID
Camera Software Update
Improved Multitasking
AirDrop
FaceTime Audio
iPhone 5S
iPhone 5c
iOS 8
Announced: WWDC 2014 - June 2, 2014
Release: Septe 17, 2014
Apple launched iOS 8 with the announcement of iPhone 6, iPhone 6 Plus and iPad Air 2 at WWDC on 2nd June, 2014
The flagship features of iOS 8 were Apple Pay. The other features were like Health App, QuickType, iCloud Drive, third party keyboard support and much more to point out.
Continuity
Apple Pay
Health App
Wi-Fi Calling
Third Party Keyboard Support Family Sharing, iCloud Drive Message App Update
iPhone 6 iPhone 6 Plus
iOS 9
Announced: WWDC 2015 - June 8, 2015
Release: Septe 16, 2015
Apple launched iOS 9 with an entirely different strategy to work with 3D Touch in WWDC on 8th June, 2015.
iOS 9 announced with some notable features like transit directions in Maps, a revamped Notes app. Siri got smarter with iOS 9 as Apple introduced more intelligence and integrated Spotlight.
Notes
Maps
News
Wallet
Designed to Work with New 3D Touch in iPhone6 and 6splus
iPhone 6s iPhone 6s Plus
iOS 10
Announced: June 13, 2016
Release: Septe 13, 2016
Apple announced the biggest ever iOS OS named iOS 10 at WWDC on 13th June, 2016.
iOS 10 features included new Home app, redesigned Maps and Messages app, revamped widgets screen.
Overhauled Messages App
Passbook icon
Allows texts with a Variety of Special Effects
Redesigned Maps App For Better Navigation
Raise to wake option
iPhone 7 Plus
iPhone 7
Nimblechapps

Ubuntu Touch

- Mobile OS by Canonical: Ubuntu Touch is a mobile operating system developed by Canonical Ltd.
- Convergence Feature: It offers convergence, enabling seamless transition between mobile and desktop interfaces.
- Open Source: Built on open-source principles, fostering community-driven development.

Ubuntu Touch cont…

- Security and Privacy: Prioritizes user security with encryption, sandboxing, and user-controlled permissions.
- App Ecosystem: While growing, its app catalog is not as extensive as mainstream mobile platforms, but supports traditional Linux applications.

Blackberry

- Security Emphasis: BlackBerry OS prioritizes security, appealing to business users and government agencies.
- Physical Keyboards: Known for physical keyboards, offering tactile typing experiences.
- Unified Communication: BlackBerry Hub integrates emails, messages, and social media for streamlined communication.

Blackberry cont…

- Enterprise Solutions: Offers robust enterprise solutions like BlackBerry Enterprise Server (BES) for device and data management.
- Market Decline: Despite past dominance, BlackBerry OS has lost market share to iOS and Android, leading to a shift towards Android-based devices.

Tizen

- Samsung's Creation: Tizen OS is developed by Samsung, primarily for its smartwatches, smart TVs, and other Internet of Things (IoT) devices.
- Linux-Based: It's based on Linux, ensuring compatibility with a wide range of hardware and offering flexibility for developers.

Tizen

- App Ecosystem: While not as extensive as Android or iOS, Tizen has its own app ecosystem, with a growing number of applications available for users.
- Samsung's Alternative: Tizen serves as Samsung's alternative to Google's Android operating system, giving the company more control over its devices and services.
- Open Source: Tizen is open-source, allowing developers to contribute to its development and enabling customization by device manufacturers.

Firefox

- Mozilla's Mobile OS: Firefox OS was Mozilla's attempt at creating an open-source mobile operating system based on web technologies like HTML5, CSS, and JavaScript.
- Focus on Web Apps: It prioritized web apps over traditional native apps, allowing developers to create cross-platform applications.

Firefox cont…

- Low-Cost Devices: Initially targeted at low-cost smartphones, aiming to provide affordable options in emerging markets.
- Discontinued: Despite early promise, Firefox OS struggled to gain traction and was eventually discontinued in 201
- Legacy Lives On: Some of its technologies, like Progressive Web Apps (PWAs), continue to influence the mobile web landscape.

Symbian

- Pioneering(New idea) Mobile OS: Symbian OS was one of the earliest mobile operating systems, powering millions of phones worldwide.
- Customizable Interface: It offered a customizable interface with widgets and themes, allowing users to personalize their devices.
- Native Apps: Symbian had a vast library of native apps, including popular titles like Nokia Maps and Opera Mini.

Symbian cont…

- Challenges with User Experience: Despite its early success, Symbian faced criticism for its outdated user interface and slow performance compared to newer rivals.
- Decline and Discontinuation: Due to the rise of iOS and Android, Symbian gradually lost market share and was eventually discontinued by Nokia in favor of Windows Phone.

History of Symbian

- 1998: Symbian Ltd. is founded as a joint venture between Nokia, Ericsson, Motorola, and Psion.
- 2000: Symbian OS 0 is released, powering early smartphones like Nokia's Communicator series.
- 2001: Nokia introduces the first Symbian-based smartphone, the Nokia 9210 Communicator.
- 2002: Symbian OS 0 is released, bringing improvements in multimedia and connectivity features.

History of Symbian

- 2008: Symbian Foundation is established to oversee the open-sourcing of the Symbian OS.
- 2010: Nokia announces a strategic partnership with Microsoft and begins transitioning away from Symbian to Windows Phone.
- 2013: Nokia discontinues Symbian development and support, marking the decline of the platform.
- 2014: The last Symbian-based device, the Nokia 808 PureView, is released, effectively ending the era of Symbian smartphones.

Windows OS for Mobile

- Windows OS for Windows Phone aimed for a unified ecosystem across devices.
- Its tile-based interface offered live updates and a distinct visual experience.
- Integration with Microsoft services like Office and OneDrive enhanced productivity.

Windows OS for Mobile cont…

- Despite innovation, the platform faced tough competition and struggled to gain market share.
- Microsoft eventually discontinued Windows Phone, shifting focus to other platforms.

Old Questions

- Explain Different types of Mobile Oses. [2017] [8marks]
- What is Mobile OS? Define Android Mobile OS Structure and architecture. [2016] [7]
- Define Mobile Phones and it’s OS and How Android OS is different from iOS and Windows phone.
- Differentiate between native apps and HTML5 apps.

Key Features Android

- Android: Open-source, customizable, large app ecosystem.
- iOS: Closed-source, seamless integration with Apple ecosystem, focus on security.
- Ubuntu Touch: Convergence between desktop and mobile, Linux-based.
- Blackberry: Security-focused, physical keyboard on some models.

Key Features Android

- Tizen: Open-source, Samsung's alternative to Android, used in wearables and smart TVs.
- Firefox OS: Web-based, emphasis on HTML5 apps, discontinued.
- Symbian: Early smartphone OS, Nokia's former flagship, now obsolete.
- Windows Phone: Unique tile-based UI, integration with Microsoft services, discontinued.

Development Environment

- Android: Java, Android Studio, Google Play Store.
- iOS: Swift/Objective-C, Xcode, Apple App Store.
- Ubuntu Touch: HTML5, Qt, Snap Store.
- Blackberry: Java, C++, BlackBerry World.
- Tizen: HTML5, C/C++, Tizen Store.
- Firefox OS: HTML5, WebIDE, Mozilla Marketplace (discontinued).
- Symbian: C++, Qt, Nokia Store (discontinued).
- Windows Phone: C#, Visual Studio, Microsoft Store (discontinued).

UI/UX and Security

- Android: Material Design, Google Play Protect.
- iOS: Human Interface Guidelines, Touch ID/Face ID, App Store review process.
- Ubuntu Touch: Convergent UI, security features inherited from Ubuntu.

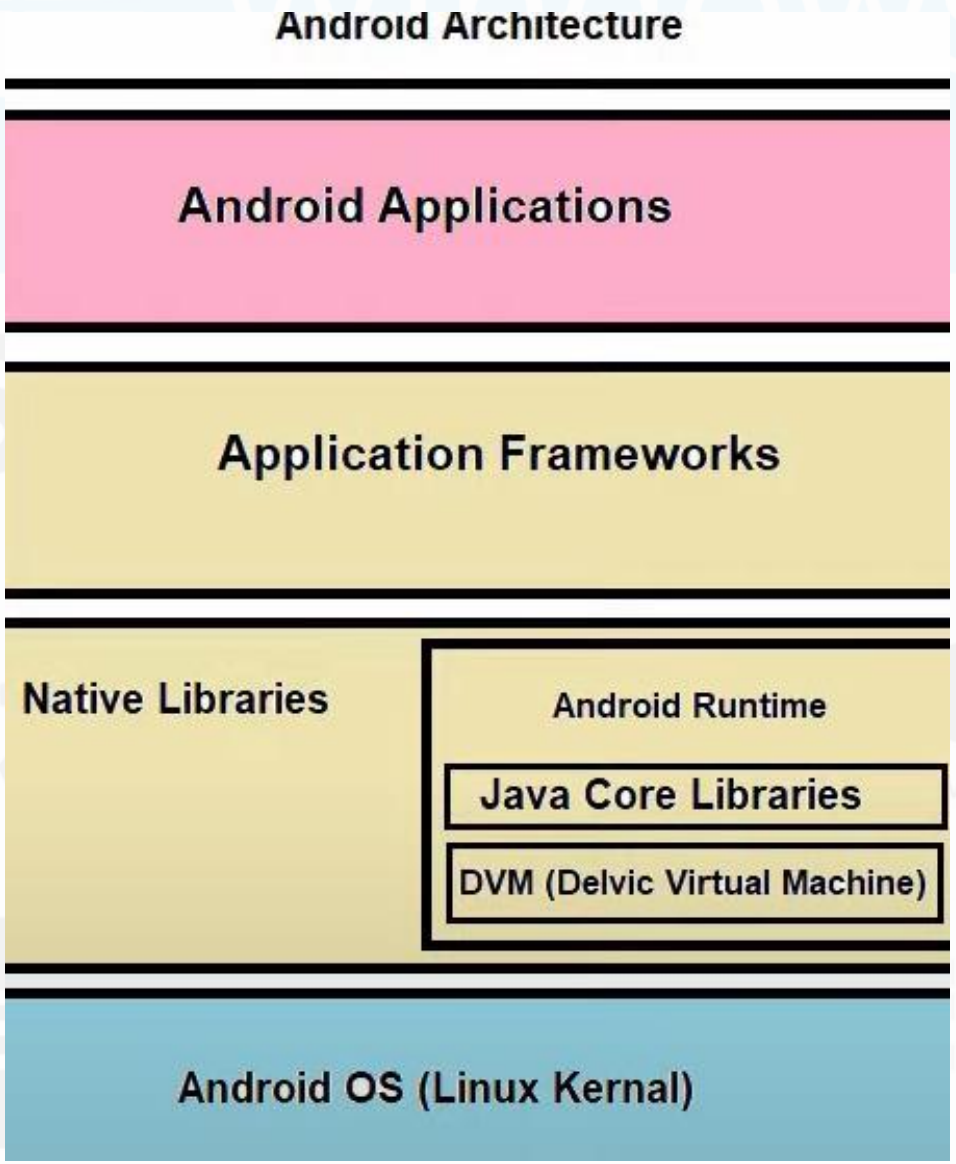
UI/UX and Security

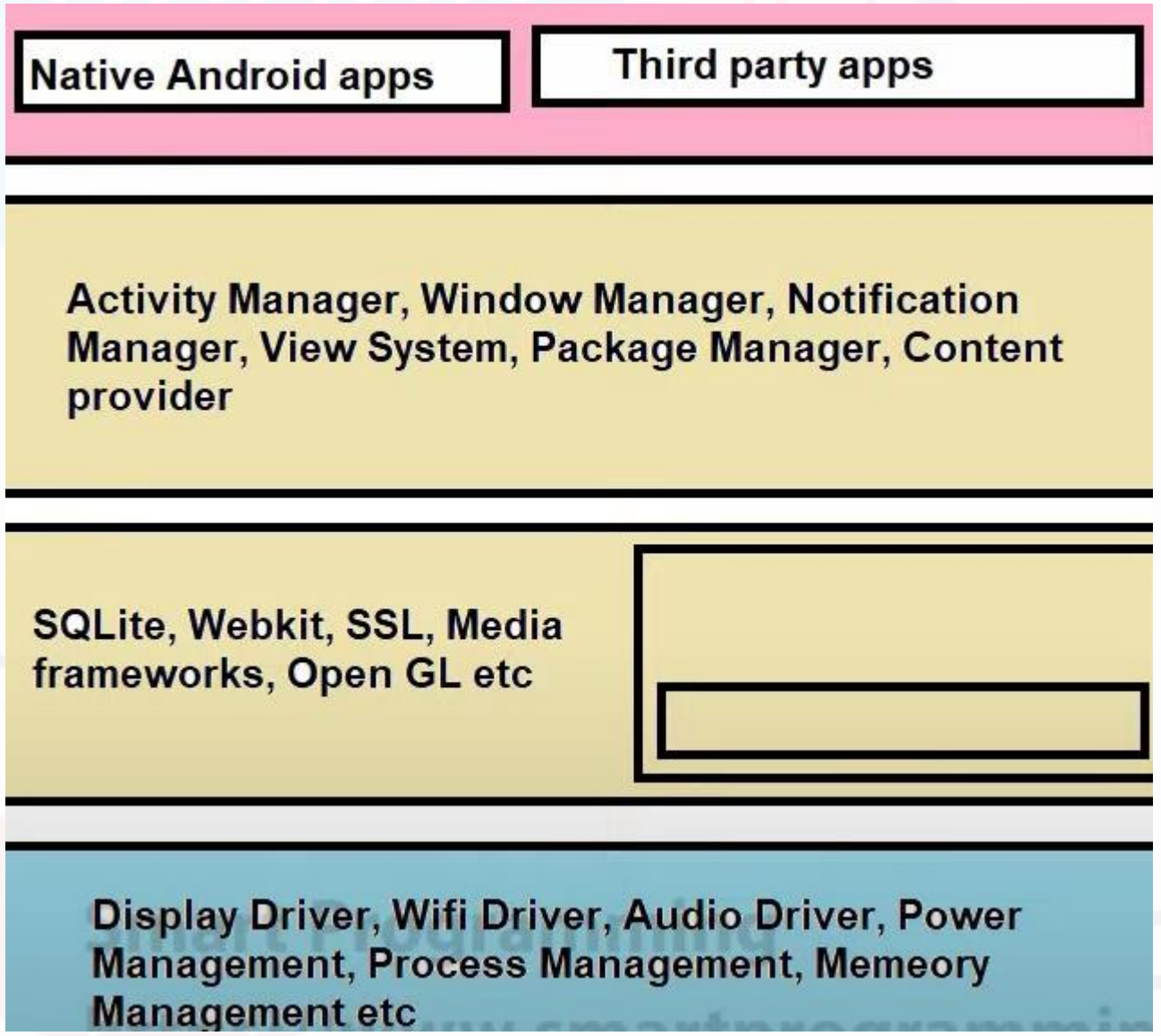
- Blackberry: BB10's gesture-based UI, BlackBerry Balance for work/personal separation.
- Tizen: TouchWiz UI (Samsung devices), Samsung Knox for security.
- Firefox OS: Simple, web-based UI, emphasis on privacy.
- Symbian: S60/Symbian Belle UI, limited security features.
- Windows Phone: Metro UI, BitLocker encryption.

Build and Structures of Mobile OSes

- A mobile operating system serves as a foundational software platform, enabling the execution of various application programs on devices like PDAs, cellular phones, smartphones, and more.
- Basic Function are:
  - User Interface Management
  - Hardware Interaction
  - Memory Management
  - Security Enforcement

Android Architecture





Hardware Manufacturer Device Driver

- Low-level hardware in mobile devices encompasses essential components like the processor, memory, storage, display screen, sensors, connectivity modules, battery, audio components, and camera, which directly interact with the device's software to enable

Hardware Manufacturer Device Driver

- its functionality and performance.
- Processor (CPU)
- Memory (RAM)
- Storage (ROM)
- Display Screen
- Touchscreen and Input Devices
- Sensors
- Connectivity Modules
- Camera
- Battery
- Audio Components

Manufacturer Device Driver

- Manufacturer device drivers are software components developed by the hardware manufacturer to facilitate communication between the operating system and specific hardware components, ensuring compatibility, optimal performance, and functionality of the device.

Device OS Base, Kernel

- The device OS base, or kernel, serves as the core component of the operating system, providing essential functions such as
  - hardware abstraction,
  - process management,
  - memory management,
  - device driver interfaces,
- This facilitate communication between software and hardware components in the device.

OS Libraries

- In the context of mobile operating systems, OS libraries refer to collections of precompiled code modules or functions provided by the operating system to facilitate various tasks and functionalities for application development.
- These OS libraries abstract complex functionalities, simplify app development, and promote code reuse, enabling developers to create powerful and feature-rich mobile applications efficiently.

OS Libraries

- UI Frameworks: Libraries for building user interfaces, including widgets, layouts, and themes, to create visually appealing and responsive applications.
- Networking Libraries: Modules for handling network communication, such as HTTP requests, socket programming, and network protocols, enabling apps to connect to the internet and interact with remote servers.

OS Libraries cont…

- Database Libraries: APIs for accessing and managing local databases, such as SQLite, allowing apps to store and retrieve structured data efficiently.
- Multimedia Libraries: Components for multimedia processing, including audio playback, video playback, image rendering, and camera access, supporting media-rich applications.
- Security Libraries: Modules for implementing security features such as encryption, secure data storage, authentication, and permissions management, ensuring data privacy and app integrity.

OS Libraries

- Location Services Libraries: APIs for accessing GPS, Wi-Fi, and other location sensors, enabling apps to obtain device location information and provide location-based services.
- Sensor Libraries: Interfaces for interacting with onboard sensors like accelerometers, magnetometers, and ambient light sensors, enabling apps to capture environmental data and respond to user interactions.

OS Libraries

- Concurrency Libraries: Tools for managing concurrent execution, threading, and asynchronous programming, facilitating responsive and efficient app behavior.
- Utility Libraries: Miscellaneous modules for common tasks such as file I/O, date/time manipulation, string processing, and mathematical operations, providing developers with reusable utility functions.
- Third-party Libraries: Additionally, developers often leverage third-party libraries and frameworks to extend the functionality of their apps, integrating features such as analytics, advertising, social media integration, and more.

OS Libraries Summary

- UI Libraries: Pre-built components for designing the visual interface of mobile apps.
- Networking Libraries: Tools for enabling communication between mobile apps and remote servers or devices.
- Data Management Libraries: Modules for handling local and remote data storage within apps.

OS Libraries Summary

- Multimedia Libraries: APIs for integrating audio, video, and image content into apps.
- Sensor and Hardware Libraries: Access to device sensors and hardware features like GPS and Bluetooth.
- Security Libraries: Functions for implementing data encryption and user authentication in apps.
- Utility Libraries: Miscellaneous tools for common tasks like math operations and localization.

Applications

- applications refer to software programs or packages designed to perform specific tasks or functions on mobile devices. These applications are typically organized into various layers within the OS architecture:

Applications cont…

- Native Application
  - Phone dialer
  - Messaging app
  - Contacts
  - Camera
  - Calendar
  - Clock
- Third-party Application
  - Social networking apps (e.g., Facebook, Instagram)
  - Gaming apps (e.g., Candy Crush, PUBG Mobile)
  - Productivity apps (e.g., Microsoft Office, Evernote)
  - Health and fitness apps (e.g., Fitbit, MyFitnessPal)
  - E-commerce apps (e.g., Amazon, eBay)

Applications

- applications refer to software programs or packages designed to perform specific tasks or functions on mobile devices. These applications are typically organized into various layers within the OS architecture:
- User Interface Layer: This layer includes the graphical user interface (GUI) elements and components that users interact with directly.
- Application Framework Layer: The application framework provides the runtime environment and libraries that developers use to build and run applications.

Applications cont…

- System Services Layer: Beneath the application framework layer are system services that manage core OS functionalities and provide essential services to applications.
- Kernel Layer: At the lowest level of the OS architecture is the kernel, which serves as the core component responsible for managing hardware resources and providing essential operating system services

Introduction to Development (Native v/s HTML5):

- Native apps refer to software applications developed specifically for a particular operating system (OS) platform, such as iOS for Apple devices or Android for Android devices. These apps are written in programming languages and frameworks that are native to the target platform, allowing them to take full advantage of the device's capabilities and performance.
  - Platform-Specific Development:
  - Access to Native APIs:
  - Optimized Performance:
  - Native Look and Feel:
  - Distribution via App Stores:

Cons. Of Native Application

- Platform Limitation: Requires separate codebases for each platform, leading to increased development time and cost.
- Skill Requirements: Developers need specialized knowledge of platform-specific languages and tools.
- App Store Approval Process: App review process can be time-consuming and unpredictable.
- Limited Reach: Fragmented user bases and limited market reach across different platforms.

Cons. Of Native Application cont…

- Maintenance Challenges: Separate updates and maintenance for each platform version.
- Longer Development Cycle: Involves longer development cycles compared to cross-platform approaches.
- Resource Consumption: Native apps may consume more device resources, impacting performance and battery life.

HTML5 Apps

- HTML5 apps, also known as web apps, are applications developed using web technologies such as HTML5, CSS, and JavaScript, and are designed to run in a web browser on mobile devices. Here's an overview of HTML5 apps in mobile app development:
- HTML5 apps offer a cost-effective and cross-platform solution for mobile app development, suitable for simple applications, content-driven experiences, and scenarios where rapid development and distribution are prioritized over deep integration with device features. However, they may not be suitable for applications requiring high-performance graphics, offline access to device features, or seamless native user experiences.

HTML5 Apps:

- HTML5 apps run on various devices and OSs.
- Development is accessible with web skills.
- No app store installation needed.
- Offline functionality is supported.
- Responsive design ensures device adaptability.
- Limited access to device features.
- Performance may lag compared to native apps.
- Security risks include web-based threats.

Cons. HTML5 Apps :

- Limited device integration.
- Performance may lag.
- Vulnerable to web-based threats.
- Offline functionality limited.
- Challenges in platform compatibility.

Introduction to Android

- Open-Source Platform: Android is an open-source operating system, allowing for customization, flexibility, and community-driven development.
- Wide Device Support: Android powers a vast range of devices, including smartphones, tablets, smartwatches, TVs, and automotive systems, offering versatility and accessibility.

Introduction to Android

- Google Play Store: The primary app distribution platform for Android, Google Play Store, offers millions of apps, games, and digital content, providing users with a rich ecosystem of software.
- Customization Options: Android provides extensive customization options for users and developers, including customizable home screens, widgets, themes, and third-party app launchers.
- Google Services Integration: Android seamlessly integrates with Google services such as Gmail, Google Maps, Google Drive, and Google Assistant, enhancing productivity and connectivity.

Introduction to Android

- Notification System: Android features a robust notification system that provides rich and interactive notifications, including bundled notifications, quick replies, and notification channels for granular control.
- Multitasking and Split-Screen: Android supports multitasking with features like split-screen mode, allowing users to run multiple apps simultaneously and increase productivity.

Introduction to Android cont…

- Google Assistant: Android devices come with Google Assistant, a virtual assistant that provides voice-based interaction, smart home control, and personalized assistance.
- Security and Privacy: Android offers built-in security features such as Google Play Protect, app sandboxing, and regular security updates to protect users' devices and data.

Introduction to Android cont…

- Fragmentation: A challenge for Android is fragmentation, where different devices run different versions of the OS and have varying levels of hardware specifications, leading to compatibility issues and delayed updates.

Android Feature

- Open-Source: Android is an open-source operating system, allowing for customization and flexibility in development.
- Wide Device Support: It powers a variety of devices, including smartphones, tablets, smartwatches, TVs, and automotive systems.

Android Feature cont…

- Google Play Store: The primary app distribution platform, offering millions of apps, games, and digital content.
- Customization: Android provides extensive customization options for users, including customizable home screens, widgets, and themes.
- Google Integration: Seamlessly integrates with Google services like Gmail, Google Maps, Google Drive, and Google Assistant.

Android Feature cont…

- Notification System: Features a robust notification system with bundled notifications, quick replies, and notification channels.
- Multitasking: Supports multitasking with features like split-screen mode, allowing users to run multiple apps simultaneously.
- Google Assistant: Comes with Google Assistant for voice-based interaction, smart home control, and personalized assistance.

Android Feature cont…

- Security: Offers built-in security features like Google Play Protect, app sandboxing, and regular security updates.
- Fragmentation: Challenges include device fragmentation, where different devices run different OS versions and have varying hardware specifications.

Introduction to Android VM and Runtime (Dalvik and ART)

- The Android Virtual Machine (VM), also known as the Dalvik Virtual Machine (DVM) or the Android Runtime (ART), is a key component of the Android operating system architecture. It serves as the runtime environment for executing Android applications (apps) on Android devices.

Android VM and Runtime (Dalvik and ART)

- Initially, Android apps were primarily written in Java and compiled into bytecode. The DVM was designed specifically to run this bytecode efficiently on resource-constrained mobile devices. However, with the introduction of Android 0 (Lollipop), the DVM was replaced by the Android Runtime (ART), which utilizes Ahead-of-Time (AOT) compilation for improved performance.
- Android VM plays a crucial role in the Android ecosystem by providing a runtime environment for executing Android apps efficiently on a wide range of devices, contributing to the platform's versatility and performance.

Introduction to Android VM and Runtime (Dalvik and ART)

- Execution Environment: The Android VM provides the execution environment for Android apps, allowing them to run on Android-powered devices.
- Bytecode Execution: Android apps are typically written in Java or Kotlin and compiled into bytecode. The VM interprets and executes this bytecode to perform the desired operations.
- Memory Management: The VM manages memory allocation and garbage collection, ensuring efficient utilization of resources and preventing memory leaks.

Introduction to Android VM and Runtime (Dalvik and ART)

- Optimization Techniques: The Android VM employs various optimization techniques to enhance app performance, including Just-In-Time (JIT) compilation (in DVM) and Ahead-of-Time (AOT) compilation (in ART).
- Application Isolation: Each Android app runs within its own instance of the VM, providing a level of isolation and security between apps.

Introduction to Android VM and Runtime (Dalvik and ART)

- Compatibility: The Android VM ensures compatibility between apps and different Android devices by providing a standardized runtime environment.
- Debugging and Profiling: Developers can debug and profile Android apps running on the VM using tools like Android Studio and ADB (Android Debug Bridge).

Installation and configuration of Android SDKs

- The Android SDK (Software Development Kit) is a set of tools, libraries, and resources provided by Google for developers to create applications for the Android platform. It includes everything developers need to build, test, and debug Android apps. This includes tools for building the user interface, accessing device sensors and data, managing app lifecycle, handling user input, and more. The Android SDK also provides documentation, sample code, and emulator images to help developers get started with Android app development.

Installation and configuration of Android SDKs

- [Set up the Android 11 SDK | Android Developers]

Get the Android 11 SDK

After you install and open Android Studio, install the Android 11 SDK as follows:

1. Click **Tools > SDK Manager**.
2. In the **SDK Platforms** tab, expand the **Android 11.0 ("R")** section and select the **Android SDK Platform 30** package.
3. In the **SDK Tools** tab, expand the **Android SDK Build-Tools 34** section and select the latest `30.x.x` version.
4. Click **Apply > OK** to download and install the selected packages.

Installation and configuration of Android SDKs

Update your build configuration

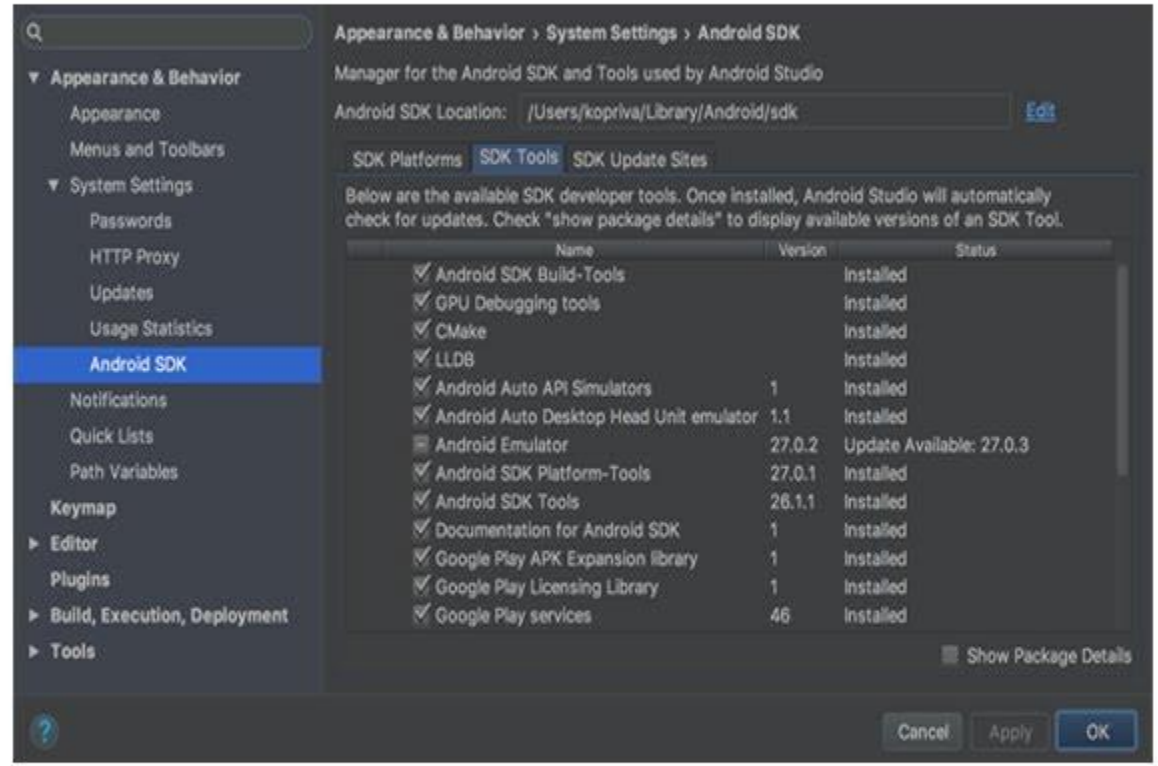
Changing your app's build configuration to target Android 11 gives your app access to the Android 11 APIs and lets you fully test your app's compatibility as you prepare to add full support for Android 11. To do this, open your module-level `build.gradle` file and update the `compileSdkVersion` and `targetSdkVersion`:

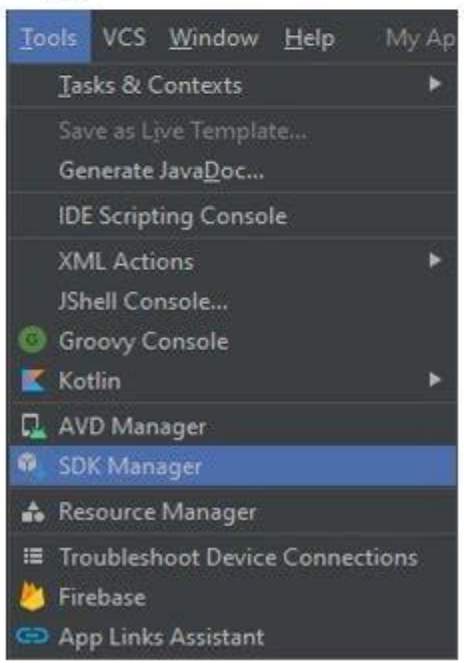
Groovy | Kotlin

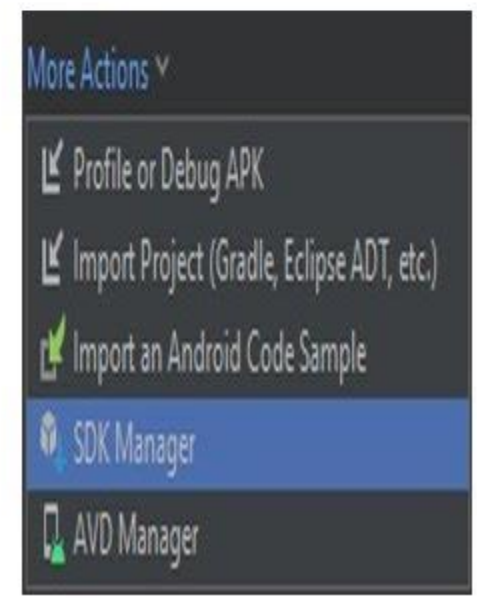
```
android {
    compileSdkVersion 30

    defaultConfig {
        targetSdkVersion 30
    }
    ...
}
```

How to ??







Install SDK Platforms & SDK Tools in Android Studio

Eclipse IDE Their integration using ADT Plugin

- The Eclipse IDE (Integrated Development Environment) was widely used for Android development, especially in the early days of Android app development.
- The ADT (Android Development Tools) Plugin was an essential component for integrating Android development into Eclipse.

Eclipse IDE Their integration using ADT Plugin

- Install Eclipse: Download and install the Eclipse IDE from the official website. Make sure you download the version that is compatible with your operating system.
- Install ADT Plugin: Once Eclipse is installed, you need to install the ADT Plugin. You can do this by going to the Eclipse menu and selecting "Help" > "Eclipse Marketplace." Then search for "Android Development Tools" and install the plugin.

Eclipse IDE Their integration using ADT Plugin

- Configure ADT Plugin: After installation, you'll need to configure the ADT Plugin. Go to "Window" > "Preferences" and navigate to "Android" in the left pane. Here, you'll need to specify the location of the Android SDK on your system.
- Create Android Project: With the ADT Plugin installed and configured, you can now create Android projects within Eclipse. Go to "File" > "New" > "Android Application Project." Follow the wizard to set up your project, including specifying the target Android version, package name, and other project settings.

Eclipse IDE Their integration using ADT Plugin

- Develop Android Apps: Once your project is created, you can start developing your Android app using Eclipse's powerful development features. You can write code in Java or Kotlin, design user interfaces using XML layout files, and utilize various debugging and testing tools provided by Eclipse and the ADT Plugin.
- Build and Run: After writing your code, you can build your Android app within Eclipse. You can run your app on an emulator or a physical Android device directly from Eclipse, making it easy to test your app as you develop it.

Eclipse IDE Their integration using ADT Plugin

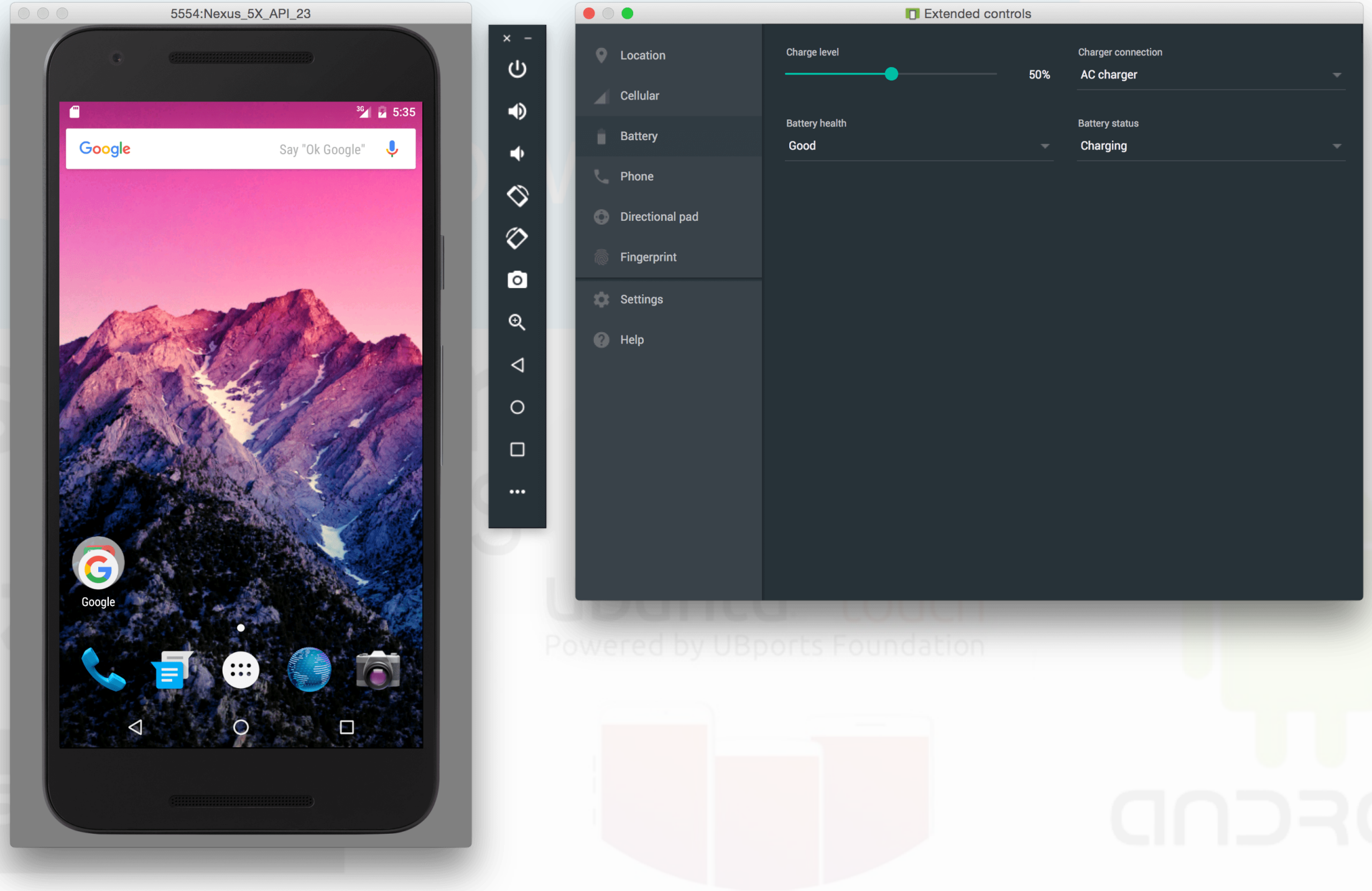
- Debug and Refine: Eclipse provides robust debugging tools that allow you to debug your Android app effectively. You can set breakpoints, inspect variables, and step through your code to identify and fix issues.

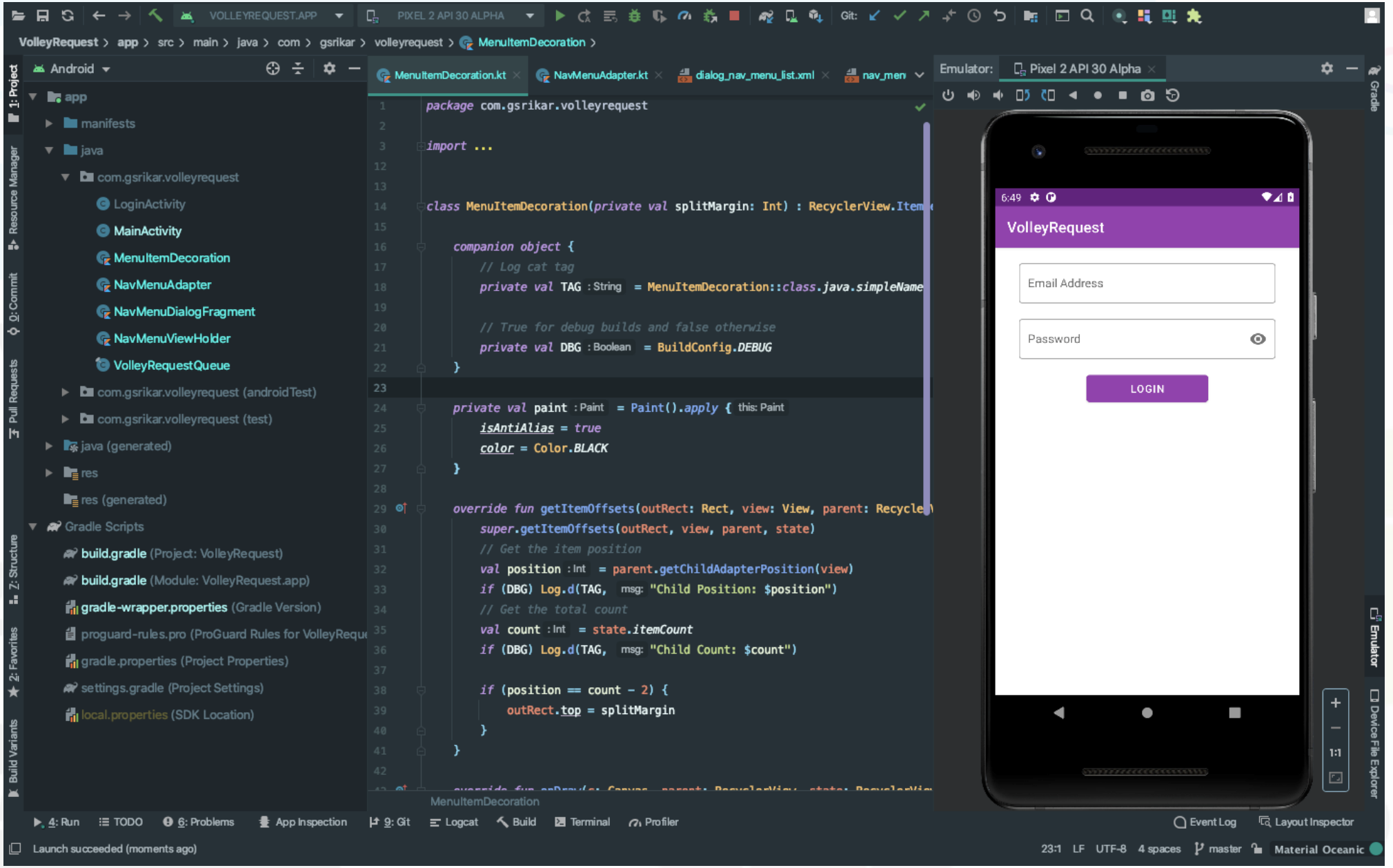
Running an emulator

- An emulator, in the context of software development, is a tool that mimics the behavior of another device or system. Specifically in Android development, an emulator replicates the functionality of an Android device, allowing developers to test their applications without the need for physical hardware.

Running an emulator

- Device Simulation:
- Testing Environment:
- Debugging:
- Cross-Platform Development:
- Integration with IDEs:
- Configuration Options


5554:Nexus_5X_API_23
5:35
Google
Say "Ok Google"
Google
Extended controls
Location
Cellular
Battery
Phone
Directional pad
Fingerprint
Settings
Help
Charge level
50%
Charger connection
AC charger
Battery health
Good
Battery status
Charging


VolleyRequest › app › src › main › java › com › gsrikar › volleyrequest › MenuItemDecoration
MenuItemDecoration.kt
NavMenuAdapter.kt
dialog_nav_menu_list.xml
package com.gsrikar.volleyrequest
import ...
class MenuItemDecoration(private val splitMargin: Int) : RecyclerView.Item
companion object {
// Log cat tag
private val TAG : String = MenuItemDecoration::class.java.simpleName
// True for debug builds and false otherwise
private val DBG : Boolean = BuildConfig.DEBUG
}
private val paint : Paint = Paint().apply { this: Paint
isAntiAlias = true
color = Color.BLACK
}
override fun getItemOffsets(outRect: Rect, view: View, parent: Recycle
super.getItemOffsets(outRect, view, parent, state)
// Get the item position
val position : Int = parent.getChildAdapterPosition(view)
if (DBG) Log.d(TAG, msg: "Child Position: $position")
// Get the total count
val count : Int = state.itemCount
if (DBG) Log.d(TAG, msg: "Child Count: $count")
if (position == count - 2) {
outRect.top = splitMargin
}
}
Emulator: Pixel 2 API 30 Alpha
6:49
VolleyRequest
Email Address
Password
LOGIN
Launch succeeded (moments ago)

Using the ADB command-line Interface

- The Android Debug Bridge (ADB) command-line interface is a versatile tool for interacting with Android devices and emulators from your computer. Here are several common use cases for using ADB:
- Connecting to Devices: ADB allows you to establish a connection between your computer and an Android device or emulator. You can connect via USB or over a network.
- Installing and Managing Apps: ADB enables you to install, uninstall, and manage apps on your Android device or emulator directly from the command line. You can install an APK using the adb install command.

Using the ADB command-line Interface

- Debugging: ADB provides various debugging features, such as logcat, which allows you to view system logs from your device or emulator. You can use the adb logcat command to monitor logs in real-time.
- File Transfer: ADB allows you to push and pull files between your computer and the connected Android device or emulator. You can use commands like adb push and adb pull to transfer files.
- Screen Recording: ADB enables you to capture screen recordings of your Android device or emulator. You can start and stop screen recordings using the adb shell screenrecord command.

Using the ADB command-line Interface

- Accessing Shell: ADB provides access to a shell on the connected Android device or emulator, allowing you to execute commands directly on the device. You can use the adb shell command to access the shell.
- Overall, the ADB command-line interface is a powerful tool for developers, offering a wide range of functionalities for interacting with Android devices and emulators efficiently and effectively.

Thank you