Assignment

[Module 3]

Que-1) Write a detailed report on the architecture of Android. Explain the key components, including Activities, Services, Broadcast Receivers, and Content Providers.

Ans:

Overview of Android Architecture

Android is an open-source operating system based on the Linux kernel. It is designed primarily for touchscreen devices like smartphones and tablets. Android architecture is organized into five key layers, each serving a specific role in the operation of the system:

1. Linux Kernel
2. Hardware Abstraction Layer (HAL)
3. Android Runtime (ART)
4. Native C/C++ Libraries
5. Java API Framework
6. Applications Layer

Each layer provides various components and services that enable the development and execution of Android applications.

Key Layers and Their Components

1. Linux Kernel

- Acts as the foundation of the Android OS.
- Handles core system services like process management, memory management, security, network stack, and hardware drivers.
- Provides abstraction between hardware and the software stack.

2. Hardware Abstraction Layer (HAL)

- Provides standard interfaces for hardware components such as cameras, sensors, audio, and more.
- Allows Android to communicate with hardware without needing to know the details of the underlying drivers.

3. Android Runtime (ART)

- Replaces the older Dalvik runtime.
- Executes bytecode for Android apps compiled using the Java/Kotlin programming languages.
- Features include:

- Ahead-of-Time (AOT) Compilation: Improves app performance by pre-compiling bytecode.
- Garbage Collection: Manages memory by reclaiming unused objects.
- Optimized memory usage and application start-up times.

4. Native C/C++ Libraries

- Includes core libraries used by various components of Android.
- Examples:
  - SQLite: Lightweight database engine.
  - OpenGL/ES: Graphics rendering for 2D/3D.
  - WebKit: Browser engine for rendering web pages.
  - Media Framework: For audio and video playback and recording.

5. Java API Framework

- Provides a rich set of APIs that developers use to build Android apps.
- Key components include:
  - Activity Manager: Manages the lifecycle of applications.
  - Notification Manager: Controls system notifications.
  - Content Providers: Facilitates data sharing between applications.
  - Location Manager: Provides location-based services.

6. Applications Layer

- The topmost layer where all user-facing applications exist.
- Includes both system apps (e.g., Phone, Contacts) and third-party apps developed by developers.

Que-2) Compare Native, Web, and Hybrid applications. What are the advantages and disadvantages of each type?

Ans:

1. **Native:**
   - Built specifically for a single operating system (like Android or iOS). They are installed directly onto the device and can access all device resources.

Advantages:

- **Best performance:** Optimized for the OS, leading to fast and smooth operation.
- **Full access to device features:** Can use GPS, camera, contacts, etc., seamlessly.
- **Enhanced user experience:** Follow platform-specific UI/UX guidelines for a familiar feel.
- **Offline functionality:** Can work without an internet connection (depending on the app).

Disadvantages:

- **Higher development cost:** Requires separate codebases for each platform (e.g., Swift for iOS, Java/Kotlin for Android).
- **Longer development time:** Building and maintaining separate apps takes more effort.
- **Updates can be fragmented:** Users on different platforms may receive updates at different times.

2. **Web:**
    - Accessed through a web browser on any device with an internet connection. They are essentially websites that look and function like apps.

Advantages:

- **Cross-platform compatibility:** Works on any device with a browser, regardless of OS.
- **Lower development cost:** One codebase for all platforms.
- **Easy to maintain and update:** Changes are made on the server-side and immediately available to all users.
- **No installation required:** Users access them directly through a URL.

Disadvantages:

- **Performance limitations:** Can be slower and less responsive compared to native apps.
- **Limited access to device features:** Cannot fully utilize device hardware or software.
- **Requires internet connection:** Mostly dependent on a stable internet connection.
- **Less immersive experience:** May not feel as integrated into the device as native apps.

3. **Hybrid:**
    - A blend of native and web apps. They are built using web technologies (HTML, CSS, JavaScript) but are packaged within a native container, allowing them to be installed on devices.

Advantages:

- **Cross-platform compatibility:** Can run on multiple operating systems from a single codebase.
- **Reduced development cost and time:** Compared to building separate native apps.
- **Access to some device features:** Can access certain device functionalities through plugins.

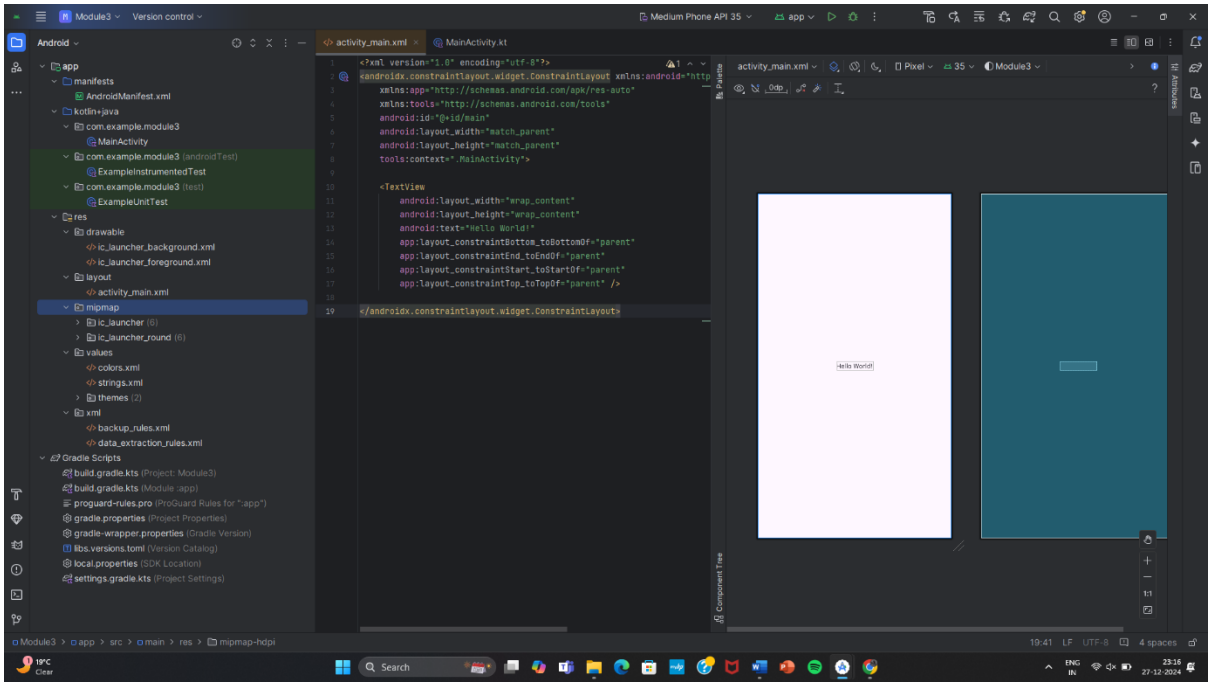
- **Easier to update:** Updates can be pushed through app stores or sometimes even without requiring a new download.

Disadvantages:

- **Performance can be inconsistent:** May not be as smooth or fast as native apps, especially for graphic-intensive tasks.
- **Limited access to device features:** Access to device hardware and software may be restricted compared to native apps.
- **Reliance on plugins:** Functionality may depend on third-party plugins, which can sometimes cause compatibility issues.
- **User experience may vary:** Can feel less native than true native apps.

Que-3) Set up Android Studio and build a basic Android project that displays "Hello World" on the screen. Take a screenshot of your project in Android Studio and describe each part of the project structure.

Ans:



Structure:

1. **app:**

- Contains all the files related to your app, including code, resources, and configurations.

2. **java:**

- Contains the source code for the app, including the Main Activity class where your app's logic resides.

3. **res:**

- Stores app resources such as layouts, drawable, strings, and styles.
- **Subfolders**:
    - layout/: Contains XML files defining the app's UI (e.g., activity_main.xml).
    - drawable/: Stores images and vector assets.
    - values/: Stores resource values like strings (strings.xml), colors (colors.xml), and themes (themes.xml).
    - Mitmap/: In this you have inbuilt imagies.
    - XML/: It has XML code.

4. **AndroidManifest.xml**:

- The manifest file defines essential app information, such as the app name, activities, permissions, and themes.

5. **Gradle Scripts**:

- **build.gradle (Module: app)**: Specifies app-level dependencies, compile options, and SDK versions.
- **build.gradle (Project)**: Configures settings for the entire project.