Assignment Module 1 – Introduction to Mobile Application Development

1. Write a detailed report on the architecture of Android.

➢ Linux Kernel Layer:

- This layer is at bottom of the architecture.
- It is the kernel on which Android is based on.
- This layer provides basic functionalities like process management, memory management, device management and drivers for various hardware.
- Drivers are used to communicate hardware with operating system.

➢ Library layer:

- On the top of the Linux kernel layer, there is a library layer.
- It contains all the classes that provides main features of operating system.
- It supports two types of libraries.
  - Java library (core library)
  - Android library
- For example, SQLite library provides classes for SQLite database, webkit library provides functionality for web browsing.

- ➢ Android Runtime layer:

- This is the third layer available on the same layer as libraries.
- It provides set of core libraries to write android applications using java languages.
- The DVM (Dalvik Virtual Machine) enables android application to run on its own process.
- In android, java classes are first compiled into dalvik executable file (.dex file) and a finally run by DVM.

- ➢ Application Framework Layer:

- This layer provides many high-level services to the application in the form of java classes.
- Android developer can use this service to make their own application.
- At the top layer, you will find all the applications that come with android device.
- It contains all the applications that you have download and install from play store as well as the applications that are developed by users.

2. Explain the key components, including Activities, Services, Broadcast Receivers, and Content Providers.

- ➢ Activities

- Represent the app's UI and a single screen.
- Handle user interactions, such as navigating or performing actions.
- Example: Login screen or settings page.

- **Services**

- Perform long-running tasks in the background without a UI.
- Example: Playing music, fetching data from the internet, or syncing data.

- **Broadcast Receivers**

- Respond to system-wide or app-specific broadcast events.
- Examples: Battery low, network connectivity changes, or custom app events.

- **Content Providers**

- Manage shared application data and provide mechanisms for data access.
- Enable inter-app data sharing.
- Examples: Accessing contacts, media, or files.

- **Fragments**

- Represent portions of the UI within an activity.
- Help create modular and reusable components.

- **Intents**

- Facilitate communication between components or between apps.
- Example: Starting a new activity, sending a broadcast, or opening a web link.

- **Manifest File**

- Declares app components, permissions, and configurations.

- Acts as the blueprint for the application.

3. Compare Native, Web, and Hybrid applications. What are the advantages and disadvantages of each type?

➢ Native Applications

- **Definition**: Apps built for a specific platform (e.g., Android using Kotlin/Java or iOS using Swift/Objective-C).
- **Advantages**:
    - High performance and fast execution.
    - Access to all device features (camera, GPS, etc.).
    - Best user experience (optimized for the platform).
- **Disadvantages**:
    - Higher development cost and effort (separate apps for each platform).
    - Platform-specific expertise required.

➢ Web Applications

- **Definition**: Apps accessed through a browser and built with web technologies (HTML, CSS, JavaScript).
- **Advantages**:
    - Cross-platform compatibility (runs on any device with a browser).
    - Easier to update and maintain.
    - Lower development cost.
- **Disadvantages**:
    - Limited access to device features.
    - Slower performance compared to native apps.
    - Dependence on internet connectivity.

- Hybrid Applications

- **Definition**: Apps that combine web and native components (built using frameworks like React Native, Flutter, or Ionic).

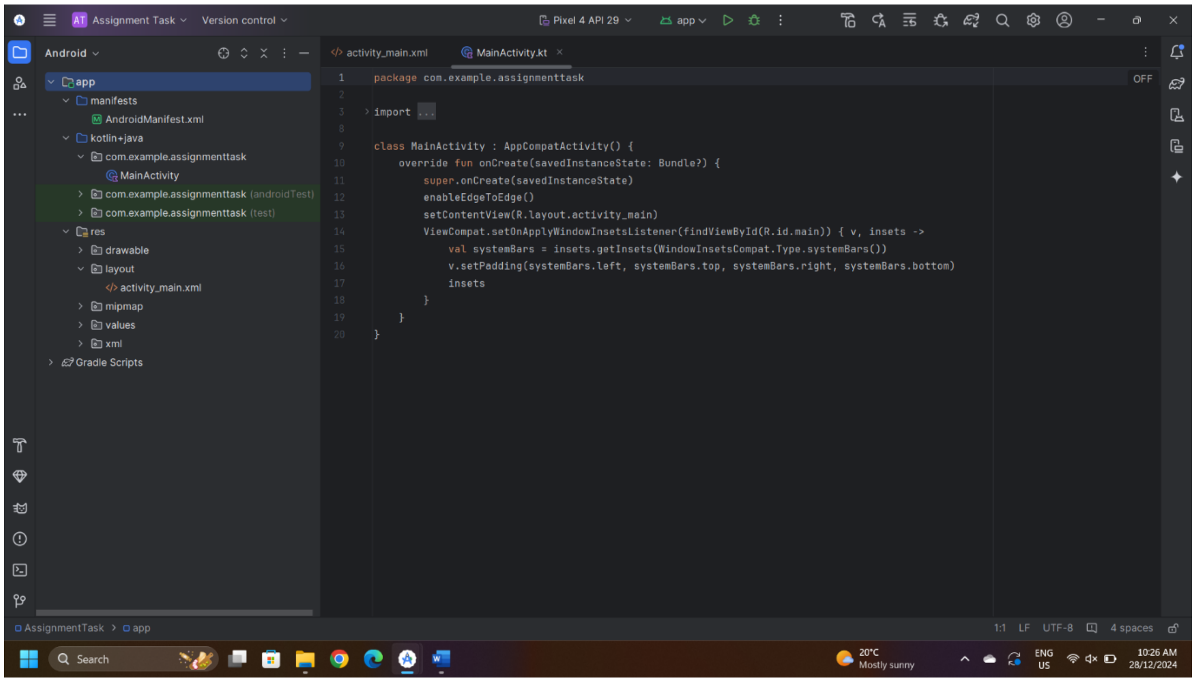
- **Advantages**:
    - Cross-platform development with a single codebase.
    - Access to some native features.
    - Faster development than native apps.

- **Disadvantages**:
    - Performance may not match native apps.
    - Limited access to advanced native features.
    - User experience can feel less polished.

4. Set up Android Studio and build a basic Android project that displays "Hello World" on the screen.

- Select on new project and give name of application.
    - Hear name of application is “Assignment Task”.

- After building Gradle scrips open screen as show below.



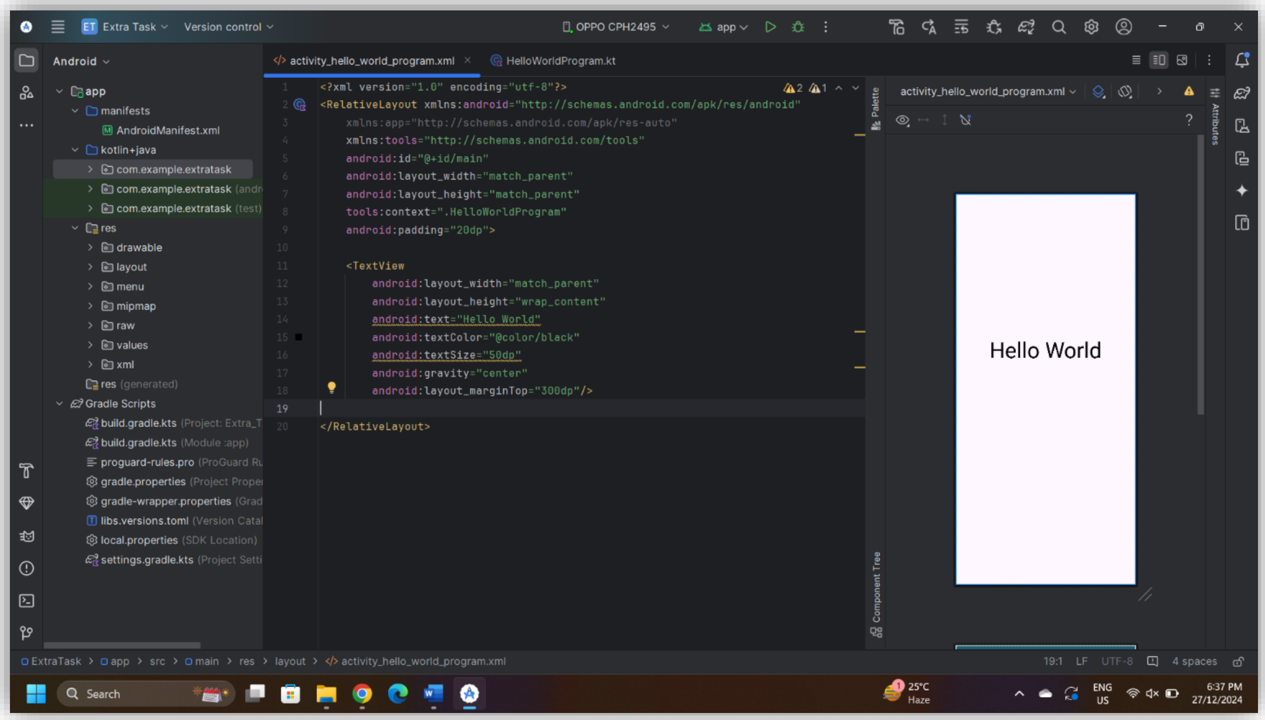
```kotlin
package com.example.assignmenttask

import ...

class MainActivity : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        enableEdgeToEdge()
        setContentView(R.layout.activity_main)
        ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.main)) { v, insets ->
            val systemBars = insets.getInsets(WindowInsetsCompat.Type.systemBars())
            v.setPadding(systemBars.left, systemBars.top, systemBars.right, systemBars.bottom)
            insets
        }
    }
}
```

- In Kotlin + Java folder, there are create file of main activity.kt. which is use for create code of Kotlin.
- In res/layout folder, create a file of activity_main.xml. which is use for create a design for application.

- In activity_main.xml create a design as show below screen



```xml
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".HelloWorldProgram"
    android:padding="20dp">

    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Hello World"
        android:textColor="@color/black"
        android:textSize="50dp"
        android:gravity="center"
        android:layout_marginTop="300dp"/>

</RelativeLayout>
```

  - Create a RelativeLayout as a parent layout
  - For print "Hello World" on screen take a one TextView tag as a child of RelativeLayout.
  - Give some attributes of TextView to create more design of "Hello World" text.