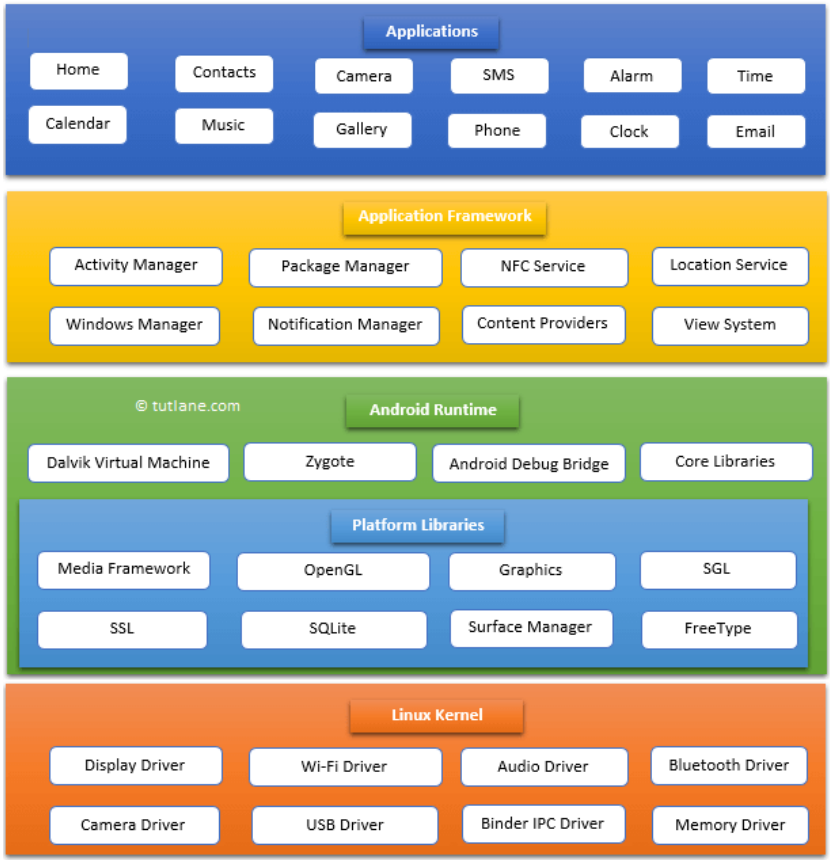
Android OS Architecture

Chapter 3: Understanding Mobile App OS Architecture

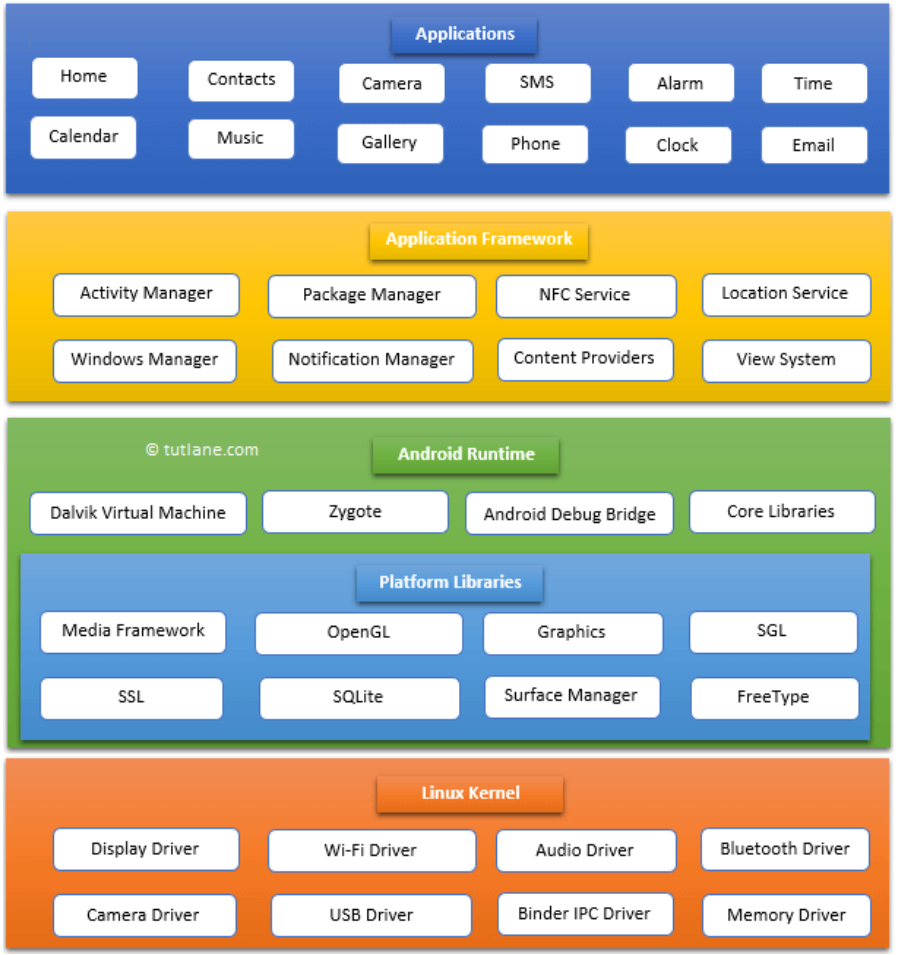
Android architecture

- **Android architecture**, is a comprehensive software stack designed to meet the needs of mobile devices.
- This stack is structured to provide a robust and flexible framework that supports a wide range of applications and services.
- It consists of several key components:
- **Linux Kernel**
- **Application Framework**
- **Android Runtime (ART)**
- **Platform Libraries**



Linux Kernel

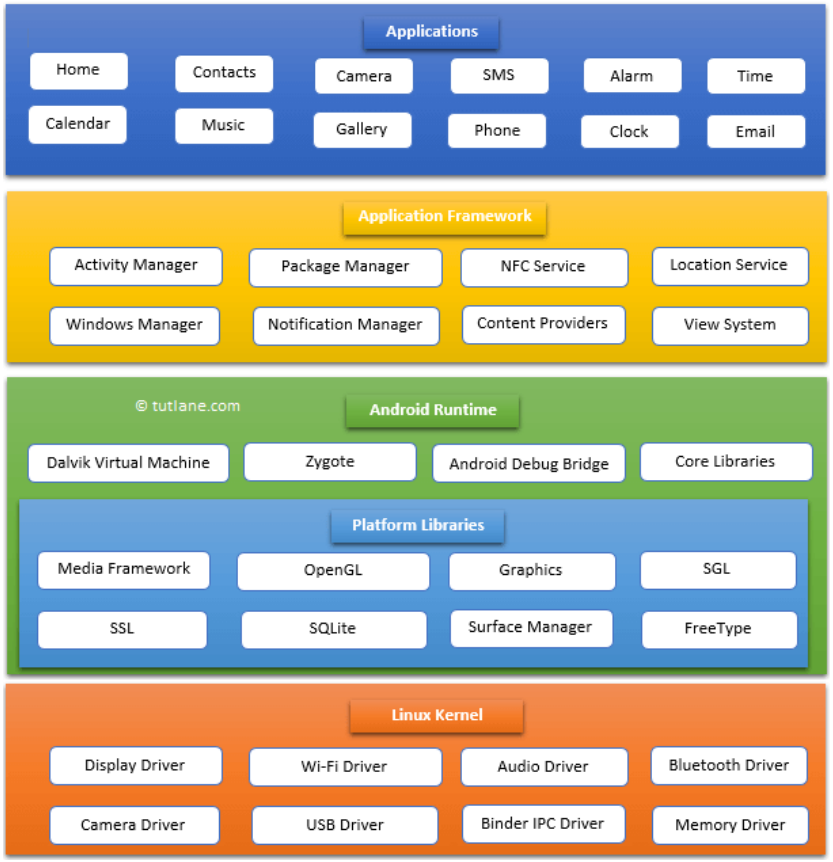
- **Linux kernel** is the foundation of the Android platform.
- It provides core system services such as security, memory management, process management, and networking.
- We benefit its security features and device manufacturers can develop hardware drivers for a well-known kernel.
- This means, it manages the device's hardware and software resources.
- It acts as an intermediary between the hardware and higher-level software layers, providing essential services such as process scheduling, memory management, and device drivers.



Source of the image : tutlane

Platform Libraries

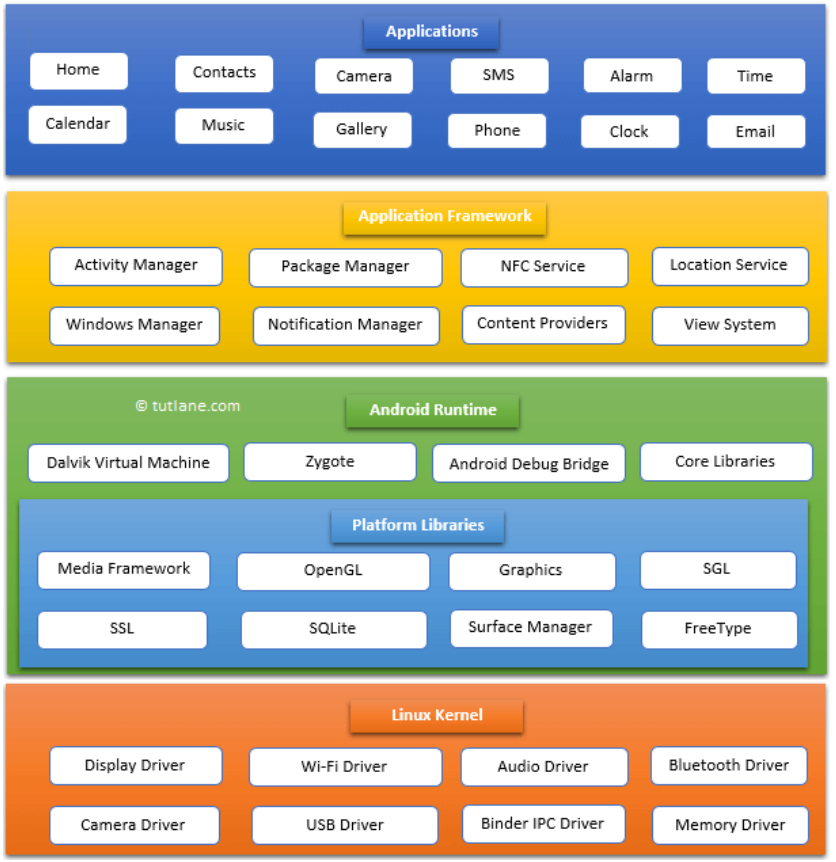
- Platform Libraries in Android development consist of core C/C++ and Java libraries like SSL, libc, Graphics, SQLite, WebKit, Media, Surface Manager, and OpenGL.
- These libraries are crucial for enabling various features and functions in Android applications.
- Key libraries include:
  - **Media Library**: For audio and video playback and recording.
  - **Surface Manager**: Manages display rendering.
  - **SGL and OpenGL**: Provide 2D and 3D graphics support.
  - **SQLite**: Supports local database management.
  - **FreeType**: Enables font rendering.
  - **WebKit**: Allows web browsing capabilities within apps.
  - **SSL**: Ensures secure internet communications.
- The purpose of Platform Libraries in Android development is to provide essential support and functionality for building robust and feature-rich applications.
- These libraries serve as the foundational components that enable various core functionalities required for Android apps.



Source of the image : [tutlane]

Android Runtime (ART)

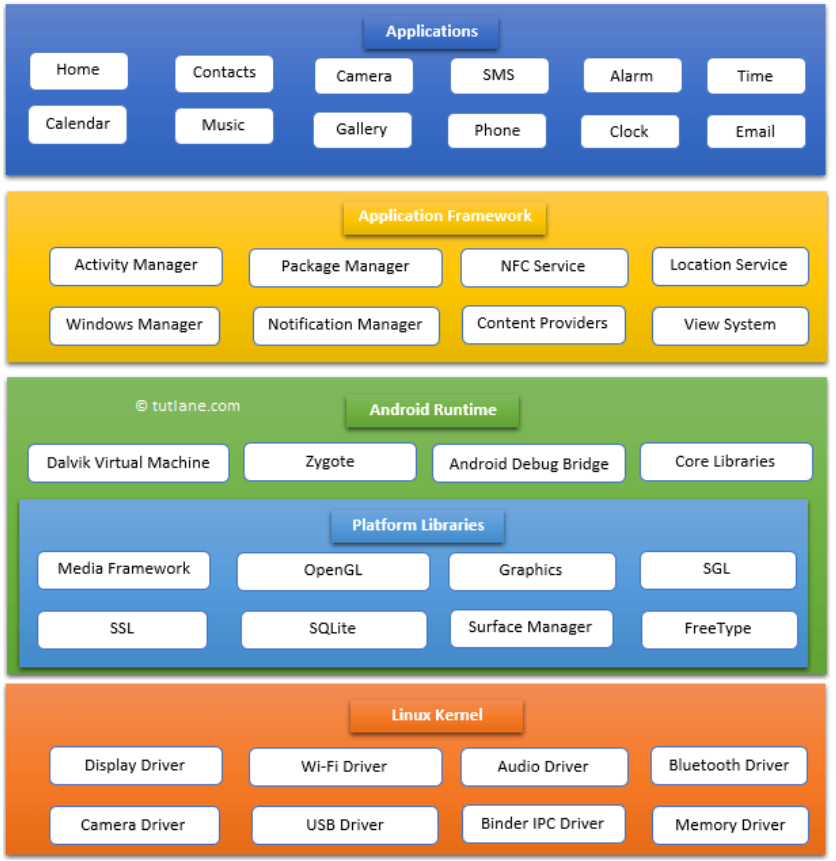
- **ART** is a crucial part of the Android system, consisting of **core libraries** and the **Dalvik virtual machine**.
- It serves as the engine for running applications and is fundamental to the application framework.
- ART is responsible for **managing the execution of applications** on Android devices.
- ART provides better performance and smoother operation compared to its predecessor, Dalvik, through techniques like ahead-of-time (AOT) compilation and just-in-time (JIT) compilation.
- The **Dalvik Virtual Machine (DVM)** is similar to the **Java Virtual Machine (JVM)**, but it is **specifically optimized for Android**.
- **Dalvik** was the original runtime environment for Android.
- It allows **devices to run multiple apps** efficiently by using the Linux kernel for tasks like threading and memory management.
- The **core libraries** in the Android Runtime enable developers to create Android apps using standard Java programming.



Source of the image : [tutlane]

Application Framework

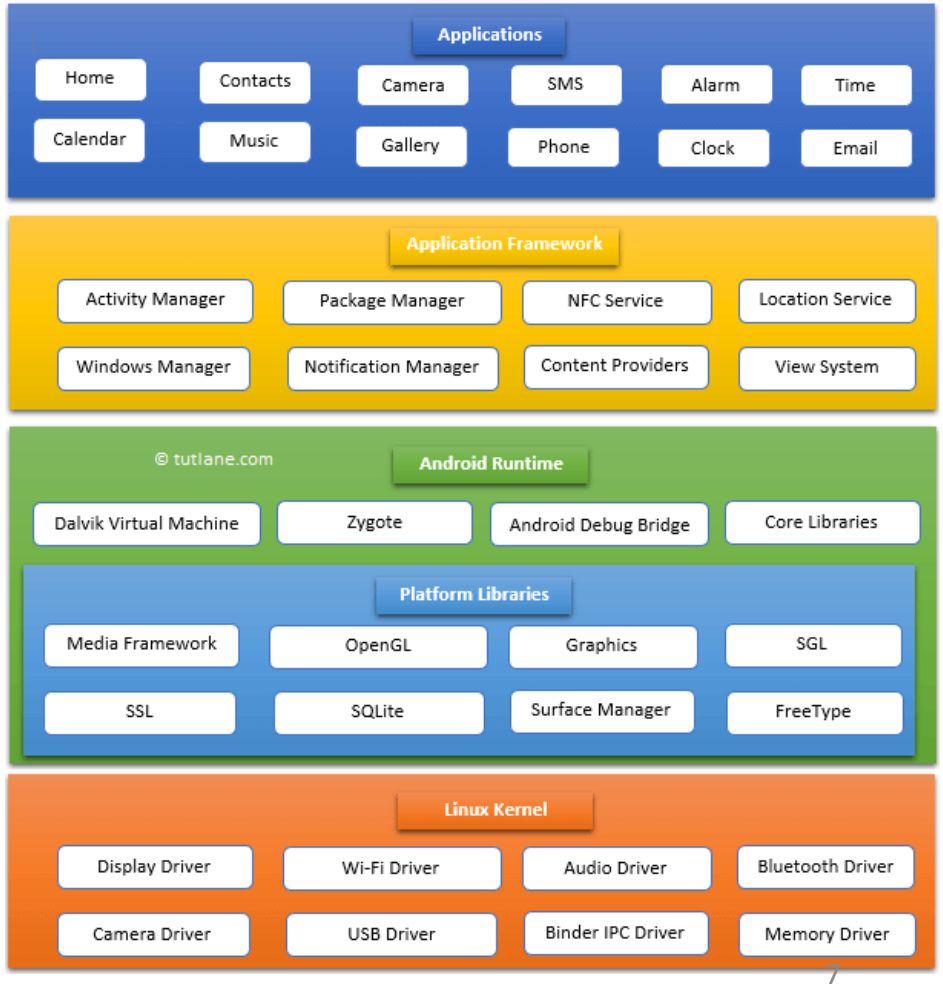
- The Java API framework provides comprehensive set of classes and services required to develop Android applications.
- It facilitates the reuse core system components and services.
- Key elements include:
- **Location Services**: Provides access to the device's geographical location, essential for location-based applications.
  - **NFC Service**: Manages Near Field Communication, enabling apps to interact with NFC tags.
  - **Notification Manager**: It helps in managing notifications.
  - **Activity Manager**: Manages the lifecycle of apps and provides navigation back stack.
  - **Content Providers**: Allow apps to access data from other apps, like Contacts, or to share their own data.



Source of the image : [tutlane]

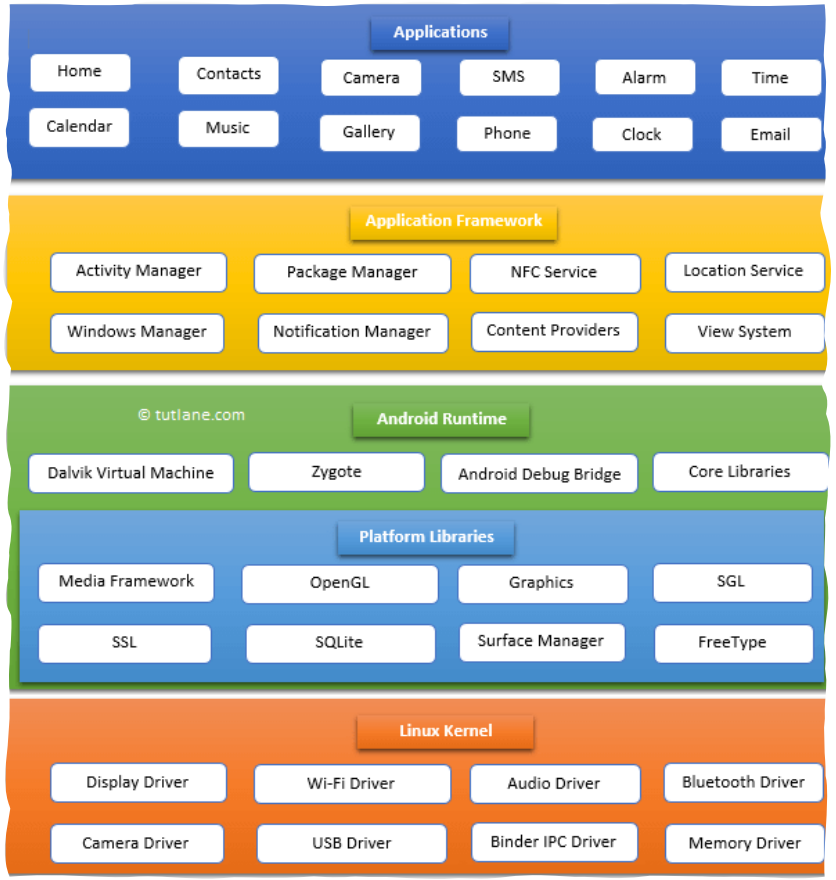
Applications

- **Mobile architecture** refers to the design and structure of mobile applications and the underlying operating systems that support them.
- It involves everything from the underlying kernel and hardware abstraction layers to the runtime environment and application frameworks.
- It contains the various components, layers, and frameworks that enable mobile devices to run applications effectively and efficiently.
- **Mobile architecture** is a multi-layered approach essential for developing robust, efficient, and scalable mobile applications.
- **Mobile architecture** play a crucial role in the functionality and performance of smartphones and tablets.
- **Understanding these components** helps developers design better applications and fully utilize the potential of mobile platforms.



Source of the image : tutlane

Platform Libraries vs Application Framework



Source of the image : tutlane

- **Platform Libraries:**
  - Provide low-level functionalities and support for core system operations.
  - Its functionalities directly related to the system's hardware and core services.
  - Consist of both C/C++ and Java-based libraries.
  - Offer essential tools and frameworks for multimedia processing, graphics rendering, data storage, web content integration, and security.
- **Application Framework:**
  - Provides higher-level APIs and services for application development.
  - Simplifies the development process by offering reusable components and services.
  - Acts as an interface between application developers and the underlying system libraries and hardware.
  - offers APIs and services that simplify and abstract the complexities of the lower-level operations provided by the platform libraries.

Links

- https://blogs.30dayscoding.com/blogs/os/real-world-applications-of-operating-systems/operating-systems-in-mobile-devices/mobile-os-architecture/
- https://www.tutlane.com/tutorial/android/android-architecture

Practice **Permissions**

1

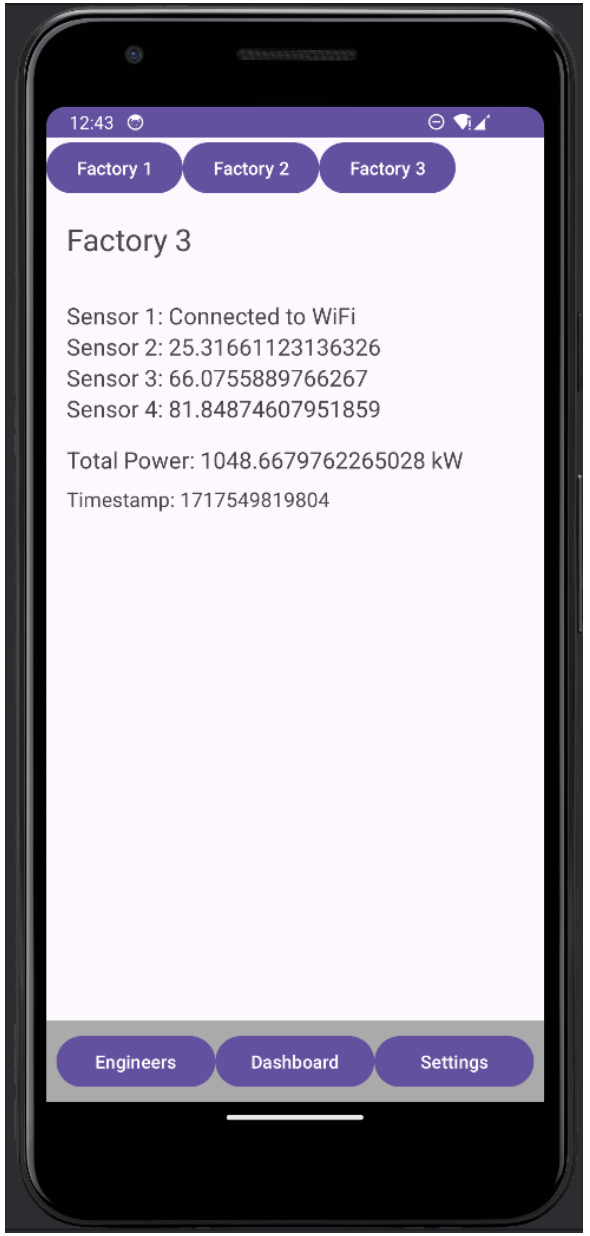
```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
  xmlns:tools="http://schemas.android.com/tools"
  android:layout_width="match_parent"
  android:layout_height="match_parent"
  tools:context=".MainActivity">

  <HorizontalScrollView
    android:id="@+id/horizontalScrollView"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_alignParentTop="true">

    <LinearLayout
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:orientation="horizontal">

      <!-- Factory Buttons -->
      <Button
        android:id="@+id/buttonFactory1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Factory 1" />

      <Button
        android:id="@+id/buttonFactory2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Factory 2" />

      <Button
        android:id="@+id/buttonFactory3"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Factory 3" />

      <!-- Add more buttons as needed -->

    </LinearLayout>
  </HorizontalScrollView>
```

2

```
<TextView
  android:id="@+id/textViewFactoryLabel"
  android:layout_width="wrap_content"
  android:layout_height="wrap_content"
  android:layout_below="@id/horizontalScrollView"
  android:text="Factory 2"
  android:textSize="24sp"
  android:layout_margin="16dp" />

<!-- Dashboard displaying sensor readings -->
<LinearLayout
  android:id="@+id/dashboardLayout"
  android:layout_width="match_parent"
  android:layout_height="wrap_content"
  android:layout_below="@id/textViewFactoryLabel"
  android:orientation="vertical"
  android:layout_margin="16dp">

  <!-- Sensor readings and other details go here -->
  <TextView
    android:id="@+id/textViewSensorReading1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Sensor 1: 0.0"
    android:textSize="18sp" />

  <TextView
    android:id="@+id/textViewSensorReading2"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Sensor 2: 0.0"
    android:textSize="18sp" />

  <TextView
    android:id="@+id/textViewSensorReading3"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Sensor 3: 0.0"
    android:textSize="18sp" />

  <TextView
    android:id="@+id/textViewSensorReading4"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Sensor 4: 0.0"
    android:textSize="18sp" />
```

3

```
<TextView
    android:id="@+id/textViewTotalPower"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Total Power: 1549.7 kW"
    android:textSize="18sp"
    android:layout_marginTop="16dp" />

  <TextView
    android:id="@+id/textViewTimestamp"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Timestamp: 2022-01-01 12:00:00"
    android:textSize="16sp"
    android:layout_marginTop="8dp" />
</LinearLayout>

<!-- Bottom Navigation Bar -->
<LinearLayout
  android:layout_width="match_parent"
  android:layout_height="wrap_content"
  android:layout_alignParentBottom="true"
  android:orientation="horizontal"
  android:background="@android:color/darker_gray"
  android:padding="8dp">

  <Button
    android:id="@+id/buttonEngineers"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:layout_weight="1"
    android:text="Engineers" />

  <Button
    android:id="@+id/buttonDashboard"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:layout_weight="1"
    android:text="Dashboard" />

  <Button
    android:id="@+id/buttonSettings"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:layout_weight="1"
    android:text="Settings" />

</LinearLayout>
</RelativeLayout>
```



SensorsAvailability class

```
public class SensorsAvailability {
    public static boolean isConnectedToWifi(Context context) {

        ConnectivityManager connectivityManager = (ConnectivityManager)
context.getSystemService(Context.CONNECTIVITY_SERVICE);


        NetworkInfo wifiNetwork = connectivityManager.getNetworkInfo(ConnectivityManager.TYPE_WIFI);
        //NetworkInfo networkInfo = connectivityManager.getActiveNetworkInfo();


return wifiNetwork != null && wifiNetwork.isConnected();
        //return wifiNetwork != null && networkInfo.isAvailable();
    }
}
```

MainActivity.jav

1

```
private TextView textViewFactoryLabel;
private TextView textViewSensorReading1;
private TextView textViewSensorReading2;
private TextView textViewSensorReading3;
private TextView textViewSensorReading4;
private TextView textViewTotalPower;
private TextView textViewTimestamp;

public String connected;
```

2

```
textViewFactoryLabel = findViewById(R.id.textViewFactoryLabel);
    textViewSensorReading1 = findViewById(R.id.textViewSensorReading1);
    textViewSensorReading2 = findViewById(R.id.textViewSensorReading2);
    textViewSensorReading3 = findViewById(R.id.textViewSensorReading3);
    textViewSensorReading4 = findViewById(R.id.textViewSensorReading4);
    textViewTotalPower = findViewById(R.id.textViewTotalPower);
    textViewTimestamp = findViewById(R.id.textViewTimestamp);

    Button buttonFactory1 = findViewById(R.id.buttonFactory1);
    Button buttonFactory2 = findViewById(R.id.buttonFactory2);
    Button buttonFactory3 = findViewById(R.id.buttonFactory3);

    Button buttonEngineers = findViewById(R.id.buttonEngineers);
    Button buttonDashboard = findViewById(R.id.buttonDashboard);
    Button buttonSettings = findViewById(R.id.buttonSettings);

    buttonFactory1.setOnClickListener(v -> updateFactoryDashboard("Factory 1"));
    buttonFactory2.setOnClickListener(v -> updateFactoryDashboard("Factory 2"));
    buttonFactory3.setOnClickListener(v -> updateFactoryDashboard("Factory 3"));

    buttonEngineers.setOnClickListener(v -> openEngineersPage());
    buttonDashboard.setOnClickListener(v -> openDashboardPage());
    buttonSettings.setOnClickListener(v -> openSettingsPage());
}
```

3

```
    private void updateFactoryDashboard(String factoryName) {
        textViewFactoryLabel.setText(factoryName);

        // check if the WIFI is connected or not, then update the sensor reading one
        if (SensorsAvailability.isConnectedToWifi(MainActivity.this)) {
            connected = "Connected to WiFi";
        } else {
            connected = "Not connected to WiFi";
        }

        // Update sensor readings and other details for the selected factory

        textViewSensorReading1.setText("Sensor 1: " + connected.toString());
        textViewSensorReading2.setText("Sensor 2: " + Math.random() * 100);
        textViewSensorReading3.setText("Sensor 3: " + Math.random() * 100);
        textViewSensorReading4.setText("Sensor 4: " + Math.random() * 100);
        textViewTotalPower.setText("Total Power: " + Math.random() * 2000 + " kW");
        textViewTimestamp.setText("Timestamp: " + System.currentTimeMillis());
    }

    private void openEngineersPage() {
        Toast.makeText(this, "Open Engineers Page", Toast.LENGTH_SHORT).show();
        // Implement navigation to Engineers Page
    }

    private void openDashboardPage() {
        Toast.makeText(this, "Open Dashboard Page", Toast.LENGTH_SHORT).show();
        // Implement navigation to Dashboard Page
    }

    private void openSettingsPage() {
        Toast.makeText(this, "Open Settings Page", Toast.LENGTH_SHORT).show();
        // Implement navigation to Settings Page
    }
}
```


ANY QUESTION