

# Mobile Application Development

## Chapter Two: Android UI, Intents and Services

Ewnetu E. (MSC.)

May 17, 2024

- Android UI Types & Components, Activities, Intents and Services
- Activity & Service Life Cycles
- Foreground Vs Background Service
- Pure Service Vs IntentService
- Characteristics of Mobile Applications
- Successful Mobile Development

# User Interface Types

- User interface (UI) design is the process designers/developers use to build interfaces in software or computerized devices, **focusing on look & feel or style**.
- UI design refers to graphical user interfaces and other forms, e.g., voice-controlled interfaces, GUI & gesture based
- User interfaces are the **access points** where users interact with the system.
- They come in three formats:
  - ▶ **Graphical user interfaces (GUIs)**: users interact with visual representations on digital control panels. e.g; smart phone UI
  - ▶ **Voice-controlled interfaces (VUIs)**: users interact with these through their voices. e.g; Smart assistants, such as Siri on iPhone & Alexa on Amazon are VUIs.
  - ▶ **Gesture-based interfaces**: users engage with 3D design spaces through bodily motions: e.g., Virtual reality (VR) games.

# Android UI

- User interface design in Android is a graphical representation of **views** displayed on a **smartphone or tablet**.
- It allows users to interact with the **features & contents** of the smartphone application.
- To design UI, you need no prior programming knowledge, although it is nice to have web development or programming skills.
- Most applications have a user interface with which users can interact.
- Android provides various **pre-built UI components** that allow you to build a GUI for your application.

- The core building blocks/ fundamental components of android are:
- **Views**: are used to customize user interface (UI) design.
  - ▶ A view is considered as a building block for a proper user interface created from the view class.
  - ▶ Are UI elements that are drawn on screen including **buttons, forms, check-boxes, text-views etc...**
  - ▶ Anything that you see on the app. screen is a view.
- **Layouts**: view hierarchies that control screen format and appearance of the views.
  - ▶ Layouts are kept in the folder called **resources**.
  - ▶ The layout can be created with a simple XML layout file located in the **resource layout folder** of any project you work on.
  - ▶ Android studio creates a default XML layout file in the resources layout folder that is extremely useful.

# Advantages of XML layout

- There are many advantages of XML layout.
- Some of the advantages are:
  - ▶ It is an immensely popular and widely used format.
  - ▶ It helps to provide the UI component from the app. logic, & provides the **flexibility** to change one component without affecting the other.
  - ▶ It is much **easier to generate than writing direct code**.
  - ▶ It allows an **easier drag and drop** to generate interfaces for the android application.
- Units of measurement
  - ▶ DP → Density independent pixel
  - ▶ SP → Scale independent pixel
  - ▶ PX → Pixel
  - ▶ PT→ Point

# Layout Types

- Constraint
- Linear
- Relative
- Table
- Absolute
- Frame

- **Constraint Layout:** a view group that allows us to position and size widgets in a flexible way.

```
<androidx.constraintlayout.widget.ConstraintLayout 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:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <Button
        android:id="@+id/onClickInsert"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:onClick="onClickInsert"
        android:text="Go To Insert Activity"
        android:textSize="24sp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />
</androidx.constraintlayout.widget.ConstraintLayout>
```



- **Linear Layout:** a view group that aligns all children in a single direction vertically or horizontally.

```
<androidx.appcompat.widget.LinearLayoutCompat 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:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <Button
        android:id="@+id/onclickinsert"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:onClick="onClickInsert"
        android:text="Go To Insert Activity"
        android:textSize="24sp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />
</androidx.appcompat.widget.LinearLayoutCompat>
```

- Android Layout attributes
  - ▶ android: id
  - ▶ android: layout\_width
  - ▶ android: layout\_height
  - ▶ android: layout\_margin
  - ▶ android: layout\_padding
  - ▶ android: layout\_weight
  - ▶ android: layout\_gravity. . . .

- Android Layout controls
  - ▶ Text input fields
  - ▶ Buttons
  - ▶ Checkboxes
  - ▶ Seek bars
  - ▶ Toggle buttons
  - ▶ Zoom buttons

# Reading Assignment

- Relative layout
- Table layout
- Absolute layout
- Frame layout

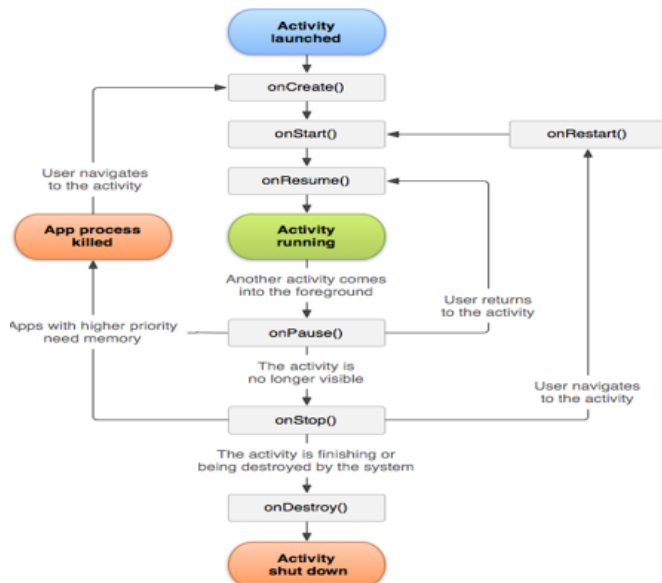
- **Fragments**: represent a behavior or a portion of user interface in an activity.
- **Resources**: represents elements, such as strings, constants and drawable pictures.
- Sub-folders in resources folder;
  - ▶ drawable
  - ▶ layout
  - ▶ mipmap
  - ▶ values
- **Manifest**: configuration file for the application.
- Configurable items include;
  - ▶ Permissions → call, sms, Internet, storage access
  - ▶ Setting launcher activity
  - ▶ Meta-data....

# Activity

**Activity:** is a class that represents a single screen.

- Provides user interaction.
- An application can have multiple activities.
- An activity has essentially four states:
  - 1 If an activity is in the foreground of the screen (at the top of the stack), it is **active or running**.
  - 2 If an activity has lost focus but is still visible (that is, a new non-full-sized or transparent activity has focus on top of it), it is **paused**.
  - 3 If an activity is **completely obscured by another activity**, it is **stopped**.
  - 4 If an activity is **paused or stopped**, the system can **destroy the activity** from memory by either asking it to finish, or simply **killing** its process.

# Android life-cycle of Activity



# Creating Activity

- An **activity** represents a single screen in an app.

```
protected void onCreate(Bundle savedInstanceState) {  
    super.onCreate(savedInstanceState);  
    setContentView(R.layout.activity_main);  
}
```



# Starting New Activity

- You can start a **new activity** by passing an Intent to **startActivity()**.
- The Intent describes the activity to start and carries any necessary data.

```
public void onClickInsert(View view){  
    Intent Insert=new Intent(getApplicationContext(), InsertActivity.class);  
    startActivity(Insert);  
}
```

# Running Android app. on a Real Device

- Set up your device

- ▶ Plug in your device to your development machine with a USB cable.
- ▶ You might need to install the appropriate USB driver for your device.
- ▶ Enable USB debugging on your device.
  - ★ On Android 4.0 and newer, go to Settings → Developer options.
  - ★ Note: On Android 4.2 and newer, Developer options is hidden by default.
  - ★ To make it available, go to Settings → About phone and tap Build number seven times.
  - ★ Return to the previous screen to find Developer options.

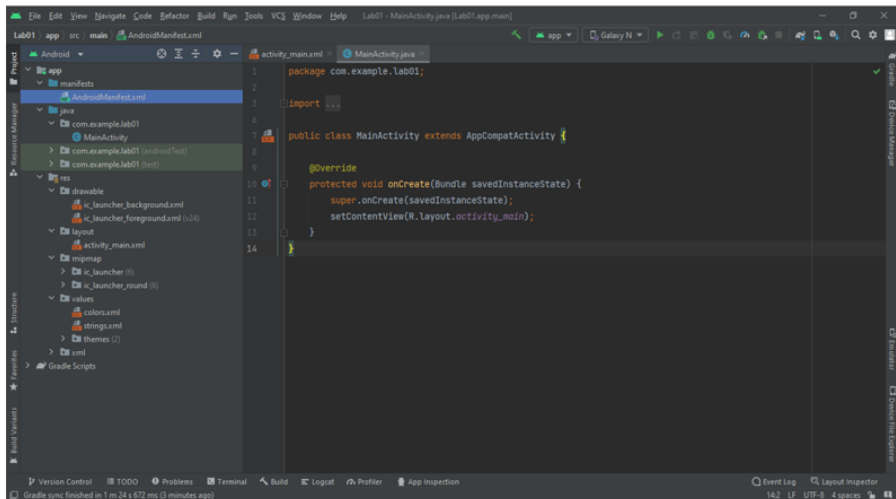
- Run the app from Android Studio

- ▶ Select one of your project's files and click Run from the toolbar.
- ▶ In the Choose Device window that appears, select the running device, and click OK .
- ▶ Android Studio installs the app on your connected device and starts it.

# Internal Details of Hello Android Example

- Here, we are going to learn the internal details or working of hello android example.
- Android application contains different components such as [java source code](#), [string resources](#), [images](#), [manifest file](#) etc.
- Let's understand the project structure of android application.
  - ▶ onCreate method: is called when Activity class is first created.
  - ▶ setContentView(R.layout.activity\_main): gives information about our layout resource.
  - ▶ Here, our layout resources for main activity are defined in activity\_main.xml file.

# ...Cont'd



# Intents

- An Intent is a messaging object you can use to request an action from another app component.
- It is a component used to invoke other components, make activity wirings etc.
- Bind individual components to each other at runtime.
- There are two types of intents.
  - ▶ **Explicit Intents**: specify which application will satisfy the intent, by supplying either the target app's package name or a fully-qualified component class name.
  - ▶ You'll typically use an explicit intent to start a component in your own app, because you know the class name of the activity or service you want to start.
    - ★ e.g, you might start a new activity within your app in response to a user action.
  - ▶ **Implicit Intents**: don't name a specific component, but instead declare a general action to perform, which allows a component from another app to handle it.
    - ★ e.g, if you want to show the user a location on a map, you can use an implicit intent to request that another capable app show a specified

```
public void onClickInsert(View view){  
    Intent Insert=new Intent(getApplicationContext(), InsertActivity.class);  
    startActivity(Insert);  
}
```

- Service: is a background process that can run for a long time (e.g. music).
- A service is a component that performs operations in the **background** usually without a user interface.
- Should be used if something needs to be done while the user is not interacting with application.
- Needs to be declared in manifest file.
- There are two main ways of using service:
  - ▶ **Foreground**: a service that will let the user know about what is happening in the background.
    - ★ e.g, in Music app., the user can see the ongoing song on the device as a form of notification.
  - ▶ **Background**: the user will never know about what is happening in the background of the application.
    - ★ e.g, while sending some images over Whatsapp, Whatsapp compresses the image file to reduce the size.

# Difference between Service and IntentService

- **Usage:**

- ▶ If you want some background task to be performed for a long time, then you should use the **IntentService**.
- ▶ You can use **service** for the tasks that don't require any UI and also it is not a very long running task.

- **How to Start?:**

- ▶ To start a service, call the **onStartService()** method
- ▶ To start IntentService, use Intent → i.e. start the IntentService by calling **Context.startService(Intent)**.

- **Running Thread:** service always runs on the **main thread** while the IntentService runs on a **separate worker thread** that is triggered from the main thread.

- **Triggering Thread:** service can be **triggered** from any thread while the IntentService can be triggered only from the main thread.



- Main Thread Blocking:

- ▶ If you are using [service](#), then there are chances that your [main thread will be blocked](#) because service runs on the main thread.
- ▶ In case of [IntentService](#), there is no involvement of the main thread.
  - ★ Here, the tasks are performed in the form of Queue.

- Stop Service:

- ▶ To stop a service, you have to use [stopService\(\)](#) or [stopSelf\(\)](#).
- ▶ In case of [IntentService](#), there is [no need of stopping the service](#) because the service will be automatically stopped once the job is done.

- Interaction with the UI:

- ▶ If you are using [IntentService](#), then you will find it difficult to interact with the UI of the application.
- ▶ If you want to display some result of the [IntentService](#) in your UI, then you have to take help of some [Activity](#).

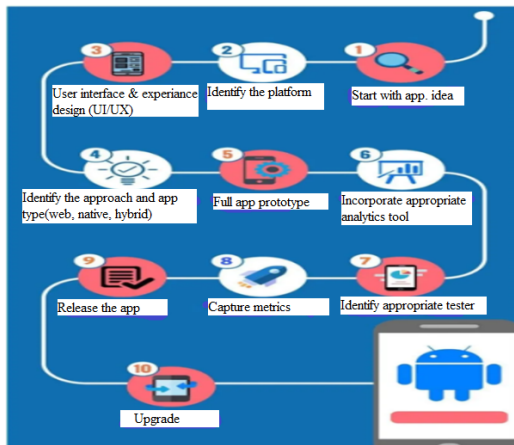
Service	Intent Service
If the task doesn't require any and also not a very long task you can use service.	If the Background task is to be performed for a long time we can use the intent service.
we use the method <b>onStartService()</b> to start the service	we use the method <b>Context.startService(Intent)</b> to start the intent service
Service will always run on the main thread.	intent service always runs on the worker thread triggered from the main thread.
There is a chance of blocking the main thread.	tasks will be performed on a queue basis i.e, first come first serve basis.
To stop service we have to use <b>stopService()</b> or <b>stopSelf()</b>	No need to stop the service, it will stop automatically.
Easy to interact with the UI of the application.	Difficult to interact with the UI of the application.

# Characteristics of Mobile Apps

- Run on resource constrained devices
- User friendly/simplicity for the end user
- High performance requirement
- Security sensitive
- Offline work
- Regular update
- Social media integration

# Successful Mobile Application Development

- To create a successful mobile application, you need to follow a **systematic approach** to the mobile app development lifecycle.
- Let you follow the following 10 steps to create a successful mobile application.



- Step 1: Starts with an app idea
- To create a successful mobile application, the first thing you need to keep in mind is:
  - ▶ Identify a problem which can be resolved by your app
  - ▶ Decide the features of your app
  - ▶ The app should provide customer with tangible benefits including reducing costs via productivity enhancements, new revenue or improving the customer experience.
- Step 2: Identification /clarification of the platform
  - ▶ Application target users
  - ▶ Mobile platforms and devices to be supported
  - ▶ Revenue model

- Step 3: UI/UX design

- ▶ Designing your app is yet another **significant factor** which is a key for the success of an app. in the market.
- ▶ Good UX design and good UI-UX means good discoverability.
- ▶ Designing an app is becoming increasingly popular as it create an instant impact on the mind of the user while ensuring usability of an app.



- Step 4: Identify approach to develop the app – native, web or hybrid
  - ▶ A number of app developers prefer to follow the agile methodology.
  - ▶ **Native:** apps. enables in delivering the best user experience but require significant time and skill to be developed.
    - ★ These apps are basically platform specific and require expertise along with knowledge.
    - ★ Native apps are costly as well as time taking to be developed and deliver the highest user experience amongst all the approaches.
    - ★ Downloaded from App Store/Play Store
    - ★ Done for specific platform. e.g; WhatsApp
  - ▶ **Web:** apps are quick and cheap to develop and can run on multiple platforms.
    - ★ Developed using HTML5, CSS and JavaScript code.
    - ★ Less powerful than native apps.
    - ★ Accessed through web browsers. e.g; Gmail

- Step 4: ...Cont'd

- ▶ **Hybrid**: this approach is the latest approach to develop any app.
  - ★ It combines pre-built native containers with on-the-fly web coding in order to achieve the best of both worlds.
  - ★ In this approach, the developer augments the web code with native language to create unique features and access native APIs which are not yet available through JavaScript. e.g; Instagram

- Step 5: App prototype

- ▶ It is actually the process of taking your idea and turning it into an application with some **basic functionality**.
- ▶ A prototype makes it quite easier to sell your idea to potential buyers who can now actually view the tangible benefits instead of just visualizing or reading product description.
- ▶ Helpful in attracting investors.



- Step 6: Develop & Integrate an appropriate analytics tool
  - ▶ There is also a need to incorporate appropriate analytics which gives you a detailed picture of;
    - ★ How many visitors will use your apps/web apps,
    - ★ How they will arrive to your site and
    - ★ How can they keep coming back.
- Some of the mobile analytics tools that are used in this process:
  - ▶ Google Analytics
  - ▶ Firebase
  - ▶ Mixpanel
  - ▶ Preemptive



- Step 7: Identify your testers: listen to them and incorporate relevant feedback
  - ▶ Beta testing is the first opportunity to get feedback from your target limited customers.
  - ▶ It is especially important as it enhances your visibility in the app store.
  - ▶ It not only reduce product risk but get you that initial push in the app store.
- Step 8: Release/Deploy the app: make your app. available to the market
- Step 9: Capture the metrics: to know the number of your app. users

- Step 10: Upgrade your app with improvements and new features
  - ▶ After capturing the metrics it becomes important to upgrade your app. with **improvements and innovative** features.
  - ▶ A mobile app. without innovative features loses its usability in long run.
  - ▶ Upgrading your app. with innovative features enhances its visibility along with downloads of an app.
  - ▶ Also ensure you keep updating your app. to meet new guidelines offered by the various platforms, don't let your apps. stagnate.



# Thank you!!!