Unit-2 :Android User Interface Design Essentials

What is Android UI?
Android is a mobile operating system that runs on 75% of mobile phones in the world and is based on a modified version of Linux kernel and other open source software. Android is primarily designed for touch screen mobile devices such as smartphones, smart watches, tablets, Android TV, and automotives. The Android operating system (OS) allows users to interact with it through different devices and a medium called the User Interface (UI).

The UI is what the end user is able to see and interact with to navigate and use different components of the Android OS on an Android application. They are presented to the user in different forms, such as the following

Graphical User Interface (GUI): This helps users interact with visual representations on digital devices or smartphones such as a Samsung phone.

Voice-controlled interfaces: These allow users to interact through voice commands, such as Siri and Alexa.

Gesture-based interface: Users are able to interact with the interface through body motions

AIDL: Android Interface Definition Language, it helps to create bridge between user and application of the android.

Top UI Controls:

Button: It is used to display text on the button to perform an Action.
Check Box: A Checkbox is the UI control that has two states that are either checked or unchecked. It helps to choose more than one option.

Image Button: An Image Button is an Absolute Layout which enables you to specify the exact location of its children. This shows a button with an image (instead of text) that can be pressed or clicked by the user.

Toggle button: An on/off button with a light indicator.

Radio Button: The Radio Button has two states: either checked or unchecked. It helps to choose only one option.

Progress Bar: The Progress Bar view provides visual feedback about some ongoing tasks, such as when you are performing a task in the background.
Spinner: A drop-down list that allows users to select one value from a set.
Time Picker: The Time Picker view enables users to select a time of the day, in either 24-hour mode or AM/PM mode.
Date Picker: The Date Picker view enables users to select a date of the day.

User Interface Screen elements

Screen elements help to apply specific action on mobile screen. The following are elements of screen:
1.Main Action Bar
2.View Control
3.Content Area
4.Split Action Bar

1.Main Action Bar:

It has detail about the action which is performed by application like opening, closing, downloading etc.

2.View control:

A View is an object that draws something on the screen that the user can interact
ViewGroup is an object that holds two or more views together in order to define the layout of the user interface.
It helps to show your content in particular order

The following are different types of views in android studio:

- **TextView** is one of the most fundamental View types. It's used to display text to the user. You can customize the appearance of the text by changing its color, size, font, and alignment.
- **ImageView** it is used for displaying images. You can manipulate images in ImageView, such as resizing them or changing their aspect ratio.
- **EditText** is an interactive TextView that allows users to input and edit text.

It's commonly used in forms, search bars, and messaging apps. You can use various input types like text, number, and password, and set various properties to control the text, like limiting the number of characters or allowing only numbers.

- **ListView** is a view which groups several items and display them in vertical scrollable list.
- **GridView** is a type of Adapter View that displays items in a two-dimensional scrolling grid. Items are inserted into this grid layout from a database or from an array.
- **Web View** class is an extension of Android's View class that lets you display web pages as a part of your activity layout.
- **ScrollView** is a view group that is used to make vertically and horizontally scrollable views.

3. Content area:

Content area manages access to a central repository of data. It is part of an Android application, which often provides its own UI for working with the data.

4.Split action bar:

It helps to divide application action by two or more ways that can be done with the help of OK or CANCEL actions, YES or NO actions etc.

Designing Interface with Layout

- Layouts in Android serve as the cornerstone for designing and arranging user interfaces.
- It is important because they provide a systematic way to arrange different interface components.
- A well-structured layout can significantly reduce the complexity of application maintenance.

The following are most frequently used layout in android studio:

LinearLayout:organizes Views in a linear order, horizontally or vertically. It is straightforward and efficient for arranging elements in a single row or column. This layout is particularly useful for creating forms, toolbars, or any interface where a sequential arrangement of elements is needed.

TableLayout: as the name suggests, it arranges Views in a grid-like format, similar to a table. It's useful for displaying rows of data or elements that need to be aligned in columns and rows. Each row in a TableLayout can contain multiple Views, and each View is placed in a cell. This layout is beneficial when creating calculators, timetables, or any interface with a tabular structure.

AbsoluteLayout enables you to specify the exact location of child Views. Each View can be positioned with specific x and y coordinates. However, it's not recommended for use as it doesn't adapt well to different screen sizes and orientations. Due to its lack of flexibility and the complexity of maintaining it across various screen sizes, it's generally avoided in modern Android development.

Relative Layout: It enables relationship between two different layouts. Whenever the changes made in one layout that will get affected in another layout also.

ConstraintLayout: It enables to create complex application in android studio. It allows to apply specific condition on layout for creating complicate UI design.

Working with Layouts:

Layout combines two or more controls together to design User Interface design. The following are attributes of layout:

layout_width: Specifies the width of a View or ViewGroup. Common values include match_parent, wrap_content, and specific dimensions.

layout_height : Specifies the height of a View or ViewGroup, Like layout width, it can be match_parent, wrap_content, or a specific size.
layout gravity: Used in certain ViewGroups like LinearLayout or FrameLayout to specify the position of a View within the ViewGroup.
Padding or margin: padding defines the space inside the View boundaries, while margin defines the space outside, around the View.
ID: Assigns a unique identifier to a View, which is crucial for referencing the View in your Java or Kotlin code.
Src: Used in ImageView to define the source image.
text: Sets the text displayed by a TextView or a Button.

Drawing and Working With animation

Animation is the process of adding a motion effect to any view, image, or text. With the help of an animation, you can add motion or can change the shape of a specific view. Animation in Android is generally used to give your UI a rich look and feel. The animations are basically of three types as follows:
1. Property Animation
2. View Animation
3. Drawable Animation

Property Animation: Property Animation is one of the robust frameworks which allows animation almost everything. This is one of the powerful and flexible animations which was introduced in Android 3.0.
View Animation : This animation is slower and less flexible. An example of View animation can be used if we want to expand a specific layout in that place we can use ViewAnimation.

Drawable Animation : Drawable Animation is used if you want to animate one image over another Attributes or Methods of animation.
startAnimation (): This method will start the animation.
clearAnimation (): This method will clear the animation running on a specific view.
Rotation(): This method helps to apply rotation on objects, the rotation can be of two types clockwise or anti-clockwise rotations.
StartPoint(): This method helps to specify the starting point of animation.
EndPoint(): This method helps to specify the ending point of animation.

Drawing an animation:
Drawing an animation can be of three types,
1.Custom animation
2.Timing and Sequencing animation
3.Creating complex animation

Custom animation: This type of animation allows users to make movement on object. Steps to create custom animation:
Step 1: Set up android environment. In this step we need to set two things dependencies and libraries.
Step 2: Define animator resources. In this step we need to set XML file and calling animate class. This animate class helps to apply basic animation in android studio. The following are examples of animate class:
Translation: It helps to translate object from one position to another position
Rotation: It helps to rotate an object either clockwise or anti-clockwise.
Scaling: It helps to change the size of object, size can be increased or decreased
Alpha: It helps to combine any two animations on single object.
Step 3: Implement animation in your application. In this step we should do apply animation in your application and referencing animation resources for the animation application.

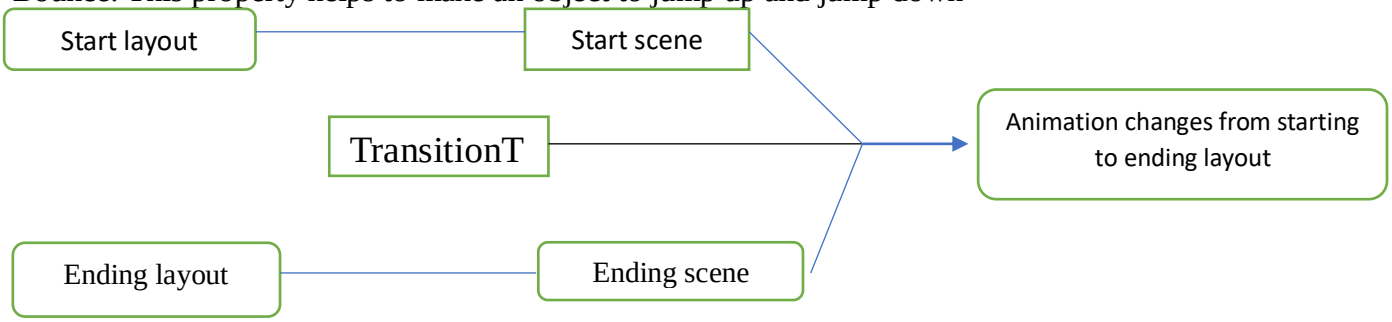
Timing and Sequencing animation:

This type of animation helps to control the animation and helps to change the sequence order of the animation. This animation can be implemented with the help of duration class.

Steps to create timing and sequencing animation:
Step 1: Using animator set to apply order of animation, duration of animation and helps to set delay which makes delay on animation transition
Step 2: Setting animation duration. This helps to control the speed of the animation.
Longer duration helps to show lower speed in animation
Shorter duration helps to make high speed in animation
Step 3: Applying delay that helps to create staggered effect on your animation The following are examples of duration class:
Start time(): It helps to set the starting time of the animation
Stop time(): It helps to set the stop time of the animation
Fixed time(): It helps to set specific time of the animation

Creating Complex animation

It can be created with help of interpolator class. The following are properties of interpolator class:
Accelerate: This property helps to increase the speed of animation
Decelerate: This property helps to decrease the speed of animation
Rotate: This property helps to rotate the object on specific number of times
Bounce: This property helps to make an object to jump up and jump down



Step 1: Starting and ending layout helps to set layout always it should be constraint layout
Step 2: Staring scene and ending scene helps to set sequence of the animation
Step 3: Transition will help to set the specific animation on object and Transition manager helps to control the flow of animation.

Testing an Android Application.

Testing an android application involves insuring it function correctly is bug-free and provides a Seamless user experience. This can be done through various methods including manual testing, automated testing & using different tools and frame works.

Types of Testing.

1. Manual Testing : Involves testers manually interacting with the app to identify issues.
2. Automated Testing : uses scripts, tools to automate repetitive testing tasks, saving time & resource.
3. Unit tests : Test individual component or function in isolation.
4. UI tests : verify and interactions! the user interface and interactions.
5. Integration tests: Ensure different part of the app work together correctly.
6. performance tests: Evaluate the app's performance under different condition.
7. Security tests : Identify potential vulnerabilities.

Publishing Android application

Publishing an Android application involves several key steps, including creating a developer account, preparing the app for release, and uploading it to the Google Play Store. This process also includes configuring store listings, handling app updates, and ensuring compliance with Google's policies.
Here's a more detailed breakdown:

1. **Creating a Developer Account and Setting up the Google Play Console:**
   - Register for a developer account: Sign up for a Play Console developer account.

- Accept the agreement: Accept the Developer Distribution Agreement and pay the one-time registration fee.
- Set up your account: Choose a developer account type, verify your identity, and ensure you meet any testing or device verification requirements.

2. Preare the app for release

- Configure your app:

Disable logging, set debuggable to false, and configure your app's version information.

- Build and sign:

Build a release version of your app using Gradle and sign it with a release key.

- Build a release version of your app using Gradle and sign it with a release key.
- Test thoroughly:
- Test the app on various devices and configurations to ensure it functions correctly, according to Android Developers.
- Prepare resources and servers:

Ensure all resources are updated and included, and that any external servers or services are production-ready.

3. Uploading to the Google Play Store:

- Create a release: In the Play Console, create a release for your app.
- Upload your app: Upload the signed APK or AAB file to the Play Console.
- Fill in details: Provide the app's title, description, screenshots, and other store listing information.
- Set content rating and pricing: Determine the appropriate content rating and set the app's price.
- Send for review: Submit the app for review by Google.

4. Post-Release Activities:

- Monitor app analytics: Track app performance and user engagement to optimize the app, according to Appy Pie.
- Handle app updates: Prepare and release updates to address issues and add new features.
- Promote your app: Use social media and other channels to promote your app and increase downloads, according to Brainvire.

Using android preferences managing application resource in a hierarchy:

One of the most Interesting Data Storage options **Android** provides its users is **Shared Preferences**. **Shared Preferences** is the way in which one can store and retrieve small amounts of primitive data as key/value pairs to a file on the device storage such as String, int, float, Boolean that make up your preferences in an XML file inside the app on the device storage. **Shared Preferences** can be thought of as a dictionary or a key/value pair.

Shared Preferences are suitable for different situations. For example, when the user's settings need to be saved or to store data that can be used in different activities within the app. As you know, onPause() will always be called before your activity is placed in the background or destroyed, So for the data to be saved persistently, it's preferred to save it in onPause(), which could be restored in onCreate() of the activity. The data stored using shared preferences are kept private within the scope of the application.

How are Shared Preferences different from Saved Instance State?

Shared Preferences	Saved Instance State
Persist Data across user sessions, even if the app is killed and restarted, or the device is rebooted	Preserves state data across activity instances in the same user session.
Data that should be remembered across sessions, such as the user's preferred settings or game score.	Data that should not be remembered across sessions, such as the currently selected tab or current state of activity.

Shared Preferences	Saved Instance State
A common use is to store user preferences	A common use is to recreate the state after the device has been rotated

How to Create Shared Preferences?

The first thing we need to do is to create one shared preferences file per app. So name it with the package name of your app- unique and easy to associate with the app. When you want to get the values, call the getSharedPreferences() method. Shared Preferences provide modes of storing the data (private mode and public mode). It is for backward compatibility- use only MODE_PRIVATE to be secure.
public abstract SharedPreferences getSharedPreferences (String name, int mode)
This method takes two arguments, the first being the name of the SharedPreference(SP) file and the other is the context mode that we want to store our file in.

- *MODE_PUBLIC will make the file public which could be accessible by other applications on the device*
- *MODE_PRIVATE keeps the files private and secures the user's data.*
- *MODE_APPEND is used while reading the data from the SP file.*

Nested classes of Shared Preferences

1. SharedPreferences.Editor: Interface used to write(edit) data in the SP file. Once editing has been done, one must commit() or apply() the changes made to the file.
2. SharedPreferences.OnSharedPreferenceChangeListener(): Called when a shared preference is changed, added, or removed. This may be called even if a preference is set to its existing value. This callback will be run on your main thread.

Functions of Shared Preferences

1. contains(String key): This method is used to check whether the preferences contain a preference.
2. edit(): This method is used to create a new Editor for these preferences, through which you can make modifications to the data in the preferences and atomically commit those changes back to the SharedPreferences object.
3. getAll(): This method is used to retrieve all values from the preferences.
4. getBoolean(String key, boolean defValue): This method is used to retrieve a boolean value from the preferences.
5. getFloat(String key, float defValue): This method is used to retrieve a float value from the preferences.
6. getInt(String key, int defValue): This method is used to retrieve an int value from the preferences.
7. getLong(String key, long defValue): This method is used to retrieve a long value from the preferences.
8. getString(String key, String defValue): This method is used to retrieve a String value from the preferences.
9. getStringSet(String key, Set defValues): This method is used to retrieve a set of String values from the preferences.
10. registerOnSharedPreferencechangeListener(SharedPreferences.OnSharedPreferencechangeListener listener): This method is used to register a callback to be invoked when a change happens to a preference.
11. unregisterOnSharedPreferencechangeListener(SharedPreferences.OnSharedPreferencechangeListener listener): This method is used to unregister a previous callback.

Working with Different types of resources:

In mobile app development you work with various resources like image, layouts, strings and like res/ to ensure, compatibility across different device and configurations.

Types of resources :

Animation resources

Define pre-determined animations.

Tween animations are saved in res/anim/ and accessed from the R.anim class.

Frame animations are saved in res/drawable/ and accessed from the R.drawable class.

Color state list resource
Define a color resource that changes based on the View state.
Saved in res/color/ and accessed from the R.color class.

Drawable resources
Define various graphics with bitmaps or XML.
Saved in res/drawable/ and accessed from the R.drawable class.

Layout resource
Define the layout for your application UI.
Saved in res/layout/ and accessed from the R.layout class.

Menu resource
Define the contents of your application menus.
Saved in res/menu/ and accessed from the R.menu class.

String resources
Define strings, string arrays, and plurals and include string formatting and styling.
Saved in res/values/ and accessed from the R.string, R.array, and R.plurals classes.

Style resource
Define the look and format for UI elements.
Saved in res/values/ and accessed from the R.style class.

Font resources
Define font families and include custom fonts in XML.
Saved in res/font/ and accessed from the R.font class.

More resource types
Define other primitive values as static resources, including the following:

Bool
XML resource that carries a boolean value.

Color
XML resource that carries a hexadecimal color value.

Dimension
XML resource that carries a dimension value with a unit of measure.

ID
XML resource that provides a unique identifier for application resources and components.

Integer
XML resource that carries an integer value.

Integer array
XML resource that provides an array of integers.

Typed array
XML resource that provides a TypedArray, which you can use for an array of drawables.