24/8/23

ASSIGNMENT 1

Unit-1&2 Introduction to Android & Android UI and UI with Views & View Group

Q.1 Explain feature of Android.

—A Android is a mobile operating system developed by Google and it's known for user-friendly interface, customization options and extensive app ecosystem. Here are some features of Android:

i) User Interface (UI): Android features a user-friendly and intuitive interface which uses a combination of touch gestures, buttons and icons to navigate through apps and settings.

ii) Home Screen Customization: Users can customize their home screens by adding widgets, shortcuts and arranging app icons for a personalized and unique user experience.

iii) Multitasking: Android allows for seamless multitasking, enabling users to switch between apps effortlessly. Recent apps can be accessed with the "Recent Apps" button.

iv) Google Play Store: The play store is Android's app distribution platform, offering millions of apps for games, productivity, entertainment and more.

v) Voice Assistant : Android devices feature Google Assistant, a virtual voice-activated assistant that can answer questions, perform tasks and interact with various apps using voice commands.

vi) Multi-User Support : It allows multiple users to have separate profiles on the same device, each with their own apps and settings.

vii) Split-Screen Mode : It enables users to run two apps simultaneously, side by side, enhancing productivity.

* Others features are Battery Optimization, Automatic Backups, Security and Privacy, Notification System.

Q.2 Explain Android Architecture in Detail. (Explain all layers).

—> The Android operating system is built upon a layered architecture that helps separate different component and functionalities, each responsible for specific tasks.

i) System Apps
At the top layer are the user-interactive applications, which are preinstalled system apps or third-party apps downloaded from

the Google Play Store. Android applications are built using Java or Kotlin programming and utilize the Android framework to access various system services and resources. For example contacts, calendar, maps, browser.

ii) Java API Framework
This layer provides the building blocks for developing Android application which includes higher-level components and APIs that make it easier to create apps with various functionality which includes:
- Activity Manager: Manages the lifecycles and navigation of application.
- Location Manager: Provides access to location-based services.
- Telephony Manager: Provides access to telephony-related information and services.
- Package Manager: Manages app installation, permissions and metadata.
- Windows Manager: Manages the windows and user interfaces of applications.
- Notification Manager: Handles notifications and alerts from apps.
- View System: Defines UI components and their interactions.

iii) ① Native C/C++ libraries.
This layer provides essential functions and services for the Android System which are

primarily written in C and C++ and include components for graphics rendering (OpenGL), multimedia playback (OpenSL/Stagefright) and more. They enable efficient and optimized performance for various hardware capabilities.

② Android Runtime (ART)
This is responsible for executing and managing applications which engage Ahead-Of-Time (AOT) compilation. This means that apps are compiled to machine code before they are run, resulting in improved performance and efficiency.

iv) Hadware Abstraction layer (HAL)
This layer provides a standardized interface for device drivers to communicate with the underlying hardware components. It abstracts the difference between various hardware platform, allowing the upper layer of the system to be hardware-agnostic.

v) Linux Kernel
This is the foundation layer which provides essential services like hardware abstraction, memory management, process management, security and device drivers. Android It acts as an intermediary between the hardware and the upper layers of the operating systems.

Q.3 What is Activity in Android? Explain Activity life Cycle in detail.

→ An Activity is a fundamental building block of an application's user interface and represents a single screen with a user interface. It plays a crucial role in defining what the user sees and interacts with on the screen and each activity is a standalone component that performs a specific task or represent a particular UI element.

→ The lifecycle of an Activity refers to the sequence of events that occur as the activity transitions through different states, from creation to destruction.

i) OnCreate(): This method is called when activity is first created, it's where you perform one-time initialisation tasks.

ii) OnStart(): Then the activity enters the "Started" state, this method is called when the activity becomes visible to the user but is not yet interactive.

iii) OnResume(): Then the activity enters "Resumed" state, this method is called when the activity is about to start interacting with the user.

iv) onPause(): When another activity comes to the foreground, the current activity enters "Paused" state. This method is called when the system expects the activity to partially hide and no longer in the foreground.

v) onStop(): When the activity is no longer visible, it enters "Stopped" state and this method is called.

vi) onDestroy(): The final callback in the lifecycle is this method, which is called when the activity is being destroyed because the user is done with the activity.

vii) onRestart(): If the activity is stopped but not destroyed i.e. another activity was on top of it but it has been removed, then this method is called when the activity is to be restarted.

Q.4 What is intent and intent filter? Why we use intent in Android?

→ An Intent is a fundamental component used to facilitate communication between different components within an application or between different applications. It is a messaging object which requests a action from another component or passes data between components.

An Intent filter is a component that defines the types of intents a component can respond to. It declares the actions and data types that the component is capable of handling. When another component wants to perform an action, it can use an implicit intent with certain criteria.

→ Intents are used for various purposes, like:

i) Starting Activities: An activity can be started from another activity using an explicit or implicit intent.

ii) Starting Services: Used to start background services that perform tasks without requiring a user interface.

iii) Broadcasting Messages: Used to broadcast messages within an application or to other applications.

iv) Passing Data Between Activities: Used to allow one activity to send data to another when they are interacting.

v) Opening External Content: Used to open URLs in a web browser, send emails, make phone calls and perform other external actions.

Q.5 Explain Implicit Intent and Explicit Intent with example.

—↓ Implicit Intent

It is an intent that doesn't explicitly specify the target component to be invoked. It describes an ~~intent~~ action and ~~of~~ data and the Android system resolves which component should handle the intent based on the available registered intent filters.

Example:

```
Intent shareIntent = new Intent(Intent.ACTION_SEND);
shareIntent.setType("image/*");
Uri imageUri = Uri.parse("content://path/to/your/
                image");
shareIntent.putExtra(Intent.EXTRA_STREAM, imageUri);
startActivity(Intent.createChooser(shareIntent,
"Share Image"));
```

—↓ Explicit Intent

It is an intent that explicitly specifies the target component to be invoked. This type of intent is used when you know the exact component you want to start or interact with.

Example:

```
Button goToSecondActivityButton = findViewById(R.id.
                                  button_go_to_second);
goToSecondActivityButton.setOnClickListener(new
View.OnClickListener() {
    @Override
```

```
    public void onClick(View v) {
        Intent intent = new Intent(MainActivity.this,
                        SecondActivity.class);
        startActivity(intent);
    }
});
```

Q.6 What is fragment? What are the steps to create new fragment in Android.

→ A Fragment is a modular component that represents a portion of a user interface within an Activity. They are used to create more modular and reusable UI components, allowing you to build flexible and responsive user interfaces, especially for larger screens and tablet layouts. Fragements have their own lifecycle and can be added, removed and replaced within an Activity.

→ Steps to create a new fragment in Android are:

i) Create the Fragment Java class.

```
public class MyFragment extends Fragment {
    @Override
    public View onCreateView(LayoutInflater inflater,
    ViewGroup container, Bundle savedInstanceState) {
        return inflater.inflate(R.layout.fragment_
            layout, container, false);
    }
}
```

ii) Create the Fragment Layout

```
<LinearLayout xmlns:android="http://schemas.
android.com/apk/res/android"
android:layout_width="match_parent"
android:layout_height="match_parent"
android:orientation="vertical">

<!-- Other Views>

<fragment
android:id="@+id/my_fragment"
android:name="com.example.app.MyFragment"
android:layout_width="match_parent"
android:layout_height="wrap_content" />

</LinearLayout>
```

#Also for adding a fragment

Q.7 Explain Fragment life Cycle in Detail.

—↓ The Fragment lifecycle consists of a series of callback methods that are called at various stages of the fragment's existence.

i) onAttach() : Called when the fragment is attached to an Activity. It provides a context that allows the fragment to interact with its hosting Activity.

ii) onCreate() : Called when the fragment is created. Initialize essential components, set up the fragment's initial state and perform any other one-time setup here.

iii) onCreateView(): Responsible for creating the visual representation of the fragment's UI.

iv) onActivityCreated(): After 'onCreate' method has completed, it is a good place to initialize UI components, set listeners and other initialization task that require the Activity to be fully created.

v) onStart(): The fragment becomes visible but not yet interactive.

vi) onResume(): The fragment is fully visible and ready to interact with the user.

vii) onPause(): The fragment is about to lose focus so all the task are paused.

viii) onStop(): The fragment is no longer visible so all the tasks have been stopped.

ix) onDestroyView(): Called when the fragment's UI is being destroyed to release all references to the view and other component to free up memory.

x) onDestroy(): The fragment is being destroyed so final cleanup tasks, release resources are being performed.

xi) onDetach() : The fragment is detached from its hosting Activity.

Q.8 What is View and View Group explain with example.

—> A View is a basic UI element that represents a rectangular area on the screen and is responsible for drawing and handling user interactions. Examples of views are:

- TextView : Displays text
- Button : Represents a clickable button
- ImageView : Displays images
- EditText : Allow users to enter text
- CheckBox : Represents a checkbox for selecting options.

—> View Group is a subclass of view that can contain other views both individual views and view groups. It provides a way to arrange and manage the positioning and layout of its child views. Example of View Groups are:

- LinearLayout : Arrange child views in a linear manner, either horizontally or vertically
- RelativeLayout : Positions relative to one another or to the parent.
- FrameLayout : Displays a single child on top of the others.
- ConstraintLayout : Uses constraints to define position and alignments.

> ScrollView : Allows for scrolling if they exceed the screen's height.

Example:

```
<LinearLayout
  xmlns:android="http://schemas.android.com"
  android:layout_width="match_parent"
  android:layout_height="match_parent"
  android:orientation="vertical">

    <Button
      android:id="@+id/button"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="Click Me" />

</LinearLayout>
```

Q.9 Write difference between View and View Group.
Ans

View	View Group
> View is a simple rectangle box that responds to user's actions	ViewGroup is a invisible container which holds view and viewgroup.
> View is a superclass of all component like Text View, EditText, Button, etc.	ViewGroup is a subclass of views that can contain other views.
> View is a component of the user interface	ViewGroup is a layout i.e. a container

View refers to the android.view.View class.	ViewGroup refers to the android.view.ViewGroup class.
Example : EditText, Button, ImageView, CheckBox, etc	Example : LinearLayout, RelativeLayout, etc.

Q.10 Create an android application to take name from textbox and on button click event display inputted name into text view.

→ layout XML :-

```
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:padding="16dp"
    tools:context=".MainActivity">

    <EditText
        android:id="@+id/editTextName"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:hint="Enter your Name : " />

    <Button
        android:id="@+id/buttonShow"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:text="Show Name" />
```

```
<TextView
  android:id="@+id/textViewResult"
  android:layout_width="match_parent"
  android:layout_height="wrap_content" />
</LinearLayout>
```

Activity Java Class:-

```
import androidx.appcompat.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;
public class MainActivity extends AppCompatActivity {
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        EditText editTextName = findViewById(R.id.editText);
        Button buttonShow = findViewById(R.id.buttonShow);
        TextView textViewResult = findViewById(R.id.
                                  textViewResult);
        buttonShow.setOnClickListener(new View.OnClickListener() {
            public void onClick(View v) {
                String name = editTextName.getText().toString();
                textViewResult.setText("Hello," + name + "!");
            }
        });
    }
}
```

Q.11 Explain following layouts in details with attributes.

a) Linear Layout

—↓ A linear layout arranges its child views in a linear fashion, either horizontally or vertically. Common attributes are:

> android:orientation ⇒ specifies whether the layout should be vertical or horizontal.
> android:layout_gallery ⇒ defines how the child views should be positioned within the layout

b) Relative Layout

—↓ A Relative Layout allows you to position child views relative to one another or relative to the parent layout. Common attributes are,

> android:layout_alignParentTop/Bottom/Left/Right
⇒ positions a child view at the top, bottom, left or right edge of the parent layout.
> android:layout_toRightOf/toLeftOf/below/above
⇒ positions a child view relative to other child views.

c) Constraint Layout

—↓ A Constraint Layout is a flexible layout that uses ~~contraint~~ constraints to position child views. It's used to build responsive designs, and complex layouts.

- app: layout_constraint Top_to Top Of / Bottom_to Bottom Of / Left_to Left Of / Right_to Right Of
  ⇒ Constraint attributes to position a view relative to other views.
- app: layout_constraint Start_to Start Of / End_to End Of
  ⇒ Constraints for positioning views based on start and end of the parent or other views.
- app: layout_constraint Horizontal_bias / Vertical_bias.
  ⇒ Specifies the horizontal or vertical bias of a view within its constraints.

d) Frame Layout:

→ A Frame Layout is a simple layout type that displays a single child view on top of others.

- No specific attributes because it's primarily used to overlap views.

e) Table Layout:

→ A Table Layout organises child views in rows and columns, creating a grid-like structure.

- android: Stretch Columns
  ⇒ Specifies which column should be Streched to fill the available space.
- android: Shrink Columns
  ⇒ Specifies which column should be shrinked.

f) Absolute Layout

—> It allowed you to specify exact positions for child views, which often lead to poor adaptability across different devices.

Q.12 Explain following views in detail with attributes.

a) TextView

—> A TextView is used to display text on the screen for headers, description and more.

> android:text ~ sets the text content
> android:textSize ~ Specifies the size of text
> android:textColor ~ sets the color of text
> android:textStyle ~ defines the style of the text (eg: bold, italics)

b) EditText

—> An EditText is used to allow the user to input text.

> android:hint ~ sets a hint text that appears when the field is empty
> android:inputType ~ defines type of input expected
> android:maxLength ~ Specifies the maximum length of the input

c) Button

—> A Button represents a clickable UI element that performs an action when pressed.

- android:text ~ sets text on the button.
- android:onClick ~ specifies the method to be invoked when clicked.
- android:background ~ defines the background drawable of the button.

d) ImageView

→ An ImageView is used to display images.

- android:src ~ sets the image resource
- android:scaleType ~ specifies how the image should be scaled and position.

e) CheckBox

→ A CheckBox allows the user to select or deselect a binary choice.

- android:text ~ sets the label associated with it.
- android:checked ~ specifies whether the checkbox is initially checked or not

f) RadioGroup

→ A RadioGroup is used to group a set of radio buttons together.

g) RadioButton

→ A RadioButton represents a single option within a RadioGroup.

- android:text ~ sets the label associated with it

> android:checked ~ specifies whether the radio button is initially selected or not.

h) Spinner

—> A Spinner displays a drop-down list of items, allowing the user to select one.

> android:entries ~ sets the array resources containing the items to display.

> android:prompt ~ specifies a prompt to be displayed when it is closed.

i) TimePicker

—> A TimePicker allows the user to select time.

> android:timePickerMode ~ specifies the mode of time picker (spinner or clock)

j) DatePicker

—> A DatePicker allows the user to select date

> android:datePickerMode ~ specifies the mode of date picker (spinner or calendar)

k) Switch

—> A Switch is used to toggle between two states.

> android:text ~ sets label associated with it initially

> android:checked ~ specifies whether it is on or off.

ℓ) ToggleButton

→ A ToggleButton is used to toggle between two states with each click.

- android:textOn:~ sets text when it is "on".
- android:textOff ~ sets text when it is "off".

Q.13 Explain ListView and GridView with Example.

→ A listView is a UI element used to display a vertically-scrollable list of items. Each items in the list is a separate view that can be customized to display text, images or a combination of both for example contacts, messages.

→ A GridView is a UI element used to display a grid of items. It organizes items in a two-dimensional grid. Each item in the grid is a separate view that can be customized similarly to ListView items.

- Example for both:-

XML File :-

```
<LinearLayout xmlns:android="http://schemas.
    android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">
```

```
<ListView
    android:id="@+id/listView"
    android:layout_width="match_parent"
    android:layout_height="match_parent" />

<GridView
    android:id="@+id/gridView"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:numColumns="2" />
```

Activity Java Class :-

```
import android.os.Bundle;
import android.app.Activity;
import android.widget.ArrayAdapter;
import android.widget.ListView;
public class ListViewActivity extends Activity {
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_list_view);
        ListView listview = findViewById(R.id.listView);
        String[] data = {"Item 1", "Item 2", "Item 3"};
        ArrayAdapter<String> adapter = new Array
        Adapter<>(this, android.R.layout.simple_list_item1,
                                                data);
        listview.setAdapter(adapter);
    }
}
```

Q.14 Differentiate Implicit and Explicit intent.

—

Implicit Intent	Explicit Intent
> Do not specify the exact target component	Explicitly names the target component's class
> Useful for assigning tasks to other apps or for allowing multiple components to handle similar actions.	Often used within the same app to navigate between activities or interact with specific services
> Based on action and data, the system determines which component can handle the intent	Used when you know the exact component you want to interact with

> Example of Implicit Intent :-

```
Intent intent = new Intent (Intent.ACTION_VIEW);
intent.setData (Uri.parse ("https://www.example.com"));
startActivity (intent);
```

> Example of Explicit Intent :-

```
Intent intent = new Intent (this, SecondActivity.
                            class);
startActivity (intent);
```

Q.15 C

Q.15 Create an android application in that user can select only one course from given options. Make use of RadioButton.

Layout XML:-

```
<LinearLayout 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"
    android:orientation="vertical"
    android:padding="16dp"
    android:context=".MainActivity">

    <RadioGroup
        android:id="@+id/radioGroupCourses"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:orientation="vertical">

        <RadioButton
            android:id="@+id/radioButtonCourse1"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Course 1" />

        <RadioButton
            android:id="@+id/radioButtonCourse2"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Course 2" />
```

```
<RadioButton
    android:id="@+id/radioButtonCourse3"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Course 3" />
</RadioGroup>
<Button
    android:id="@+id/buttonSubmit"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Submit" />

</LinearLayout>
```

Activity Java Code :~

```
package com.example.radiobuttonexample;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.RadioButton;
import android.widget.RadioGroup;
import android.widget.Toast;

public class MainActivity extends Activity {
    @Override
    protected void onCreate(Bundle savedInstanceState)
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
```

```
RadioGroup radioGroupCourses = findViewById
                (R.id.radioGroupCourses);
Button buttonSubmit = findViewById(R.id.button
                Submit);

buttonSubmit.setOnClickListener(new View.OnClickListener() {
  @Override
  public static onclick(View V) {
    int selectedId = radioGroupCourses.getCheckedRadioButtonId();
    if (SelectedId != -1) {
      RadioButton selectedRadioButton = findViewById
                (SelectedId);
      String selectedCourse = selectedRadioButton.getText().
                toString();
      Toast.makeText(MainActivity.this, "Selected
      Course: " + selectedCourse, Toast.LENGTH_SHORT).
                Show();
    } else {
      Toast.makeText(MainActivity.this, "Please
      select a course", Toast.LENGTH_SHORT).show();
    }
  }
});
}
}
```

~~~ x ~~~ x ~~~
~~~