جامعة المستقبل
AL MUSTAQBAL UNIVERSITY

كلية العلوم

قسم الأنظمة الطبية الذكية
Intelligent Medical Systems Department

Subject: Building User Interfaces
Class: 3rd
Lecturer: Asst.Lect Mustafa Ameer Awadh

Lecture: (7)

Study Year: 2024-2025

Building User Interfaces

The basic building block for a user interface in Android is a View object. Created from the View class, a View occupies a rectangular area on the screen and is responsible for drawing and event handling. It serves as the base class for widgets, which are interactive UI components like buttons, text fields, and more.

Another key element is the ViewGroup. A subclass of View, the ViewGroup acts as an invisible container that holds other Views or even other ViewGroups, defining their layout properties.

At the third level, we have various layouts, which are subclasses of the ViewGroup class. A layout defines the visual structure of an Android user interface. Layouts can be created dynamically at runtime using View/ViewGroup objects, or they can be defined in a simple XML file located in the res/layout folder of the project. For example, a layout file named main_layout.xml would be placed in res/layout to specify the arrangement of UI components.

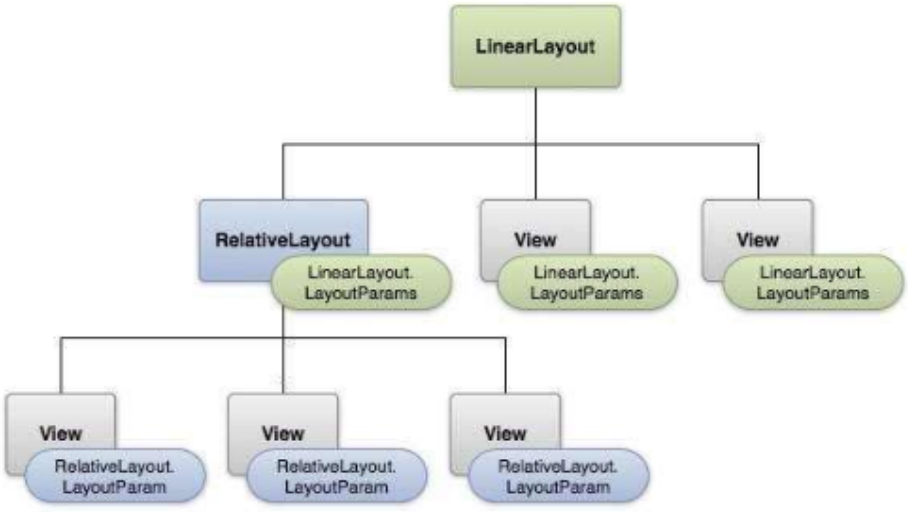

Fig.1 : Layout Structure .

types of layouts

In Android Studio, there are several types of layouts that help organize and structure UI components. Here are some of the main types:

1. **LinearLayout**
   - Arranges child elements in a single row (horizontal) or column (vertical).
   - Useful for simple layouts where components need to be aligned in a straight line.
2. **RelativeLayout**
   - Positions child elements relative to each other or to the parent container.
   - Allows for flexible and dynamic arrangements but can become complex with multiple elements.
3. **ConstraintLayout**
   - A versatile layout that allows for complex positioning and alignment.
   - Uses constraints to position elements, making it ideal for creating responsive designs that adapt to different screen sizes.
4. **FrameLayout**
   - A simple layout that displays one view on top of another, with each view stacked.
   - Useful for overlaying views or displaying a single item, such as a video or image.
5. **TableLayout**
   - Organizes child elements into rows and columns, similar to an HTML table.
   - Useful for data display but can be limiting for more complex layouts.

6. **GridLayout**
    - Divides the layout into a grid of rows and columns, allowing child views to occupy multiple cells.
    - Offers a more structured, grid-like arrangement, suitable for photo galleries or icon grids.

Each layout type has its strengths and is selected based on the specific needs of the app interface.

Layout Attribute

In Android Studio, layouts come with various attributes that control the appearance, size, and behavior of UI components. Here are some commonly used layout attributes:

1. **Width and Height**
    - android:layout_width and android:layout_height: Set the width and height of a view.
    - Common values include match_parent (fills the parent container) and wrap_content (adjusts to fit the content).
2. **Orientation**
    - android:orientation: Used in LinearLayout to specify the layout direction (horizontal or vertical).
3. **Gravity**
    - android:gravity: Controls the alignment of content within a view, such as centering text in a TextView.
    - android:layout_gravity: Aligns the view itself within its parent (e.g., centering a button within a layout).

4. **Padding and Margin**
   - android:padding: Adds space inside the boundaries of a view, pushing its content inward.
   - android:layout_margin: Adds space outside the view's boundaries, separating it from other views.
5. **Weight**
   - android:layout_weight: Used in LinearLayout to distribute space among child views proportionally.
6. **Visibility**
   - android:visibility: Controls whether a view is visible, invisible, or gone (invisible and doesn't take up space).
7. **Constraints (specific to ConstraintLayout)**
   - layout_constraintTop_toTopOf, layout_constraintBottom_to BottomOf: Set constraints to anchor views to each other or to the parent layout.
   - layout_constraintHorizontal_bias and layout_constraintVertic al_bias: Adjust alignment within the constraints.
8. **Column and Row Span (specific to GridLayout and TableLayout)**
   - android:layout_columnSpan and android:layout_rowSpan: Define how many columns or rows a view should span.

These attributes allow precise control over view placement, appearance, and behavior within an Android layout.

Create sign in interface on android studio:

Creating a sign-in interface in Android Studio is a fundamental project for any developer aiming to build interactive and user-secure mobile applications. Sign-in screens provide a gateway for user authentication, enabling apps to store and personalize user experiences based on login credentials. Using Android Studio, developers can design a sign-in interface with user-friendly components such as EditText fields for username and password, buttons for interaction, and a feedback area to guide users during login attempts.

To build a basic sign-in interface, Android developers use XML to define the UI layout and Java or Kotlin to handle functionality, like checking login credentials and providing feedback. Android Studio's Gradle-based environment simplifies managing dependencies and integrating libraries that support authentication, security, and UI customization.

Fig.2 Sign in interface.

Step 1: Create the Layout (XML file)

In the res/layout folder, create an XML file named activity_signin.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<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">

    <EditText
        android:id="@+id/etUsername"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="Username"
        android:inputType="text" />

    <EditText
        android:id="@+id/etPassword"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="Password"
        android:inputType="textPassword" />

    <Button
        android:id="@+id/btnSignIn"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Sign In" />

    <TextView
        android:id="@+id/tvSignInResult"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:padding="8dp"
        android:text=""
        android:textColor="#FF0000" />
</LinearLayout>
```

Or we can use desigin icon to drug the component like Button,TextView...e.g.Fig.3 showing the GUI editor.

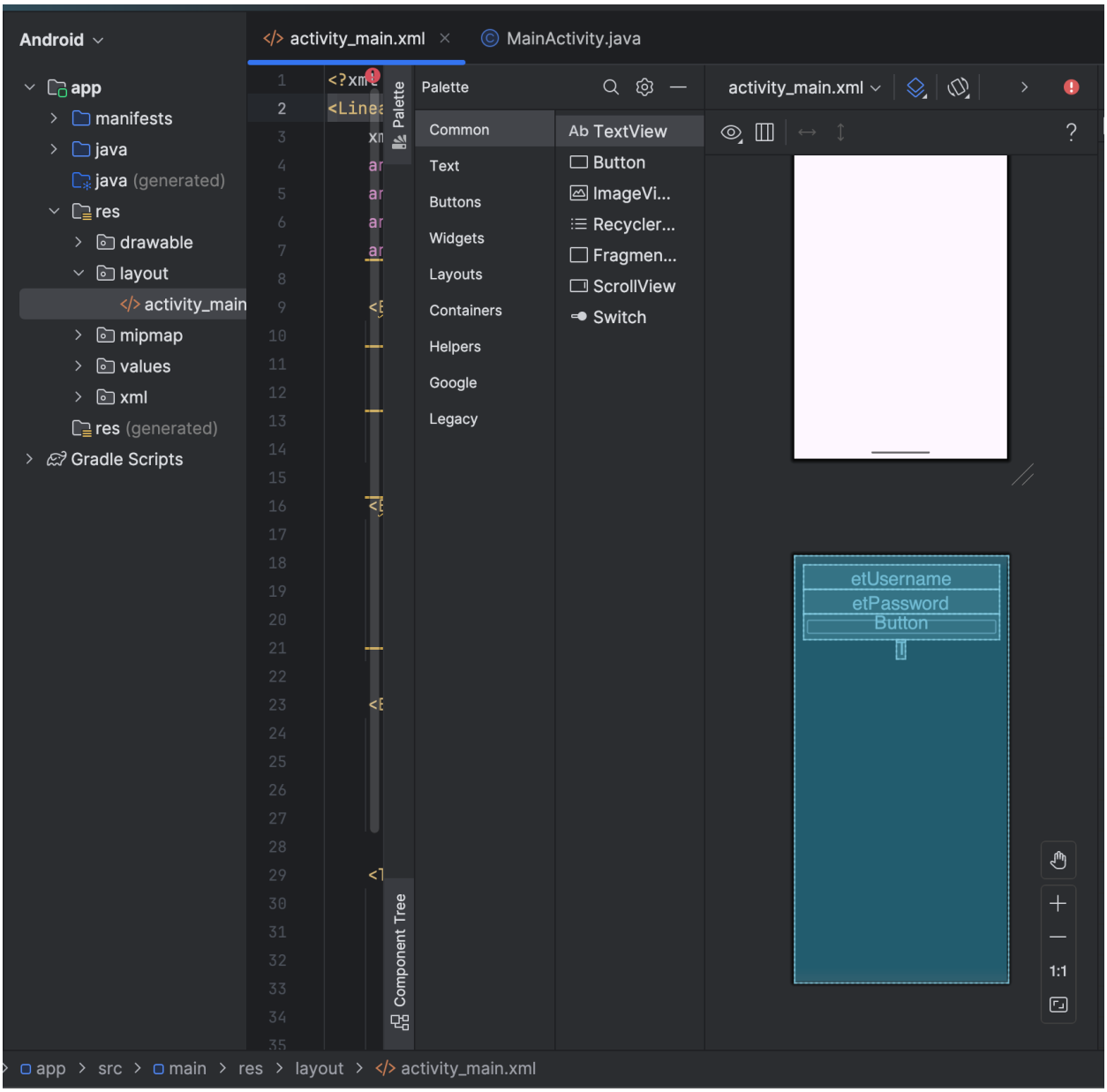

Fig.3 Showing GUI Editor.

Step 2: Create the Java Activity (MainActivity.java)

In the java folder, create a new Java class named MainActivity.java:

```
package com.example.signinapp;

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;
import android.widget.Toast;

public class MainActivity extends AppCompatActivity {

    private EditText etUsername, etPassword;
    private TextView tvSignInResult;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        etUsername = findViewById(R.id.etUsername);
        etPassword = findViewById(R.id.etPassword);
        tvSignInResult = findViewById(R.id.tvSignInResult);
        Button btnSignIn = findViewById(R.id.btnSignIn);

        btnSignIn.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                String username = etUsername.getText().toString().trim();
                String password = etPassword.getText().toString().trim();

                // Simple sign-in logic
                if (username.equals("admin") && password.equals("1234")) {
                    tvSignInResult.setText("Sign-In Successful");
                    Toast.makeText(MainActivity.this, "Welcome " + username,
Toast.LENGTH_SHORT).show();
                } else {
                    tvSignInResult.setText("Invalid Username or Password");
                    Toast.makeText(MainActivity.this, "Sign-In Failed",
Toast.LENGTH_SHORT).show();
                }
            }
        });
    }
}
```

Explanation of Code:

This code is a simple Android sign-in application created using Java. The program uses basic authentication logic to check if the username and password match preset values. Let's break down each part:

2.Package Declaration

```
package com.example.signinapp;
```

This line declares the package name for the application. It helps uniquely identify the app and organizes code files.

2. Imports

```
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;
import android.widget.Toast;
```

These import statements allow the app to use Android classes needed for the user interface, such
as AppCompatActivity, Button, EditText, TextView, and Toast.

3. Class Declaration

```
public class MainActivity extends AppCompatActivity {
```

The `MainActivity` class extends `AppCompatActivity`, which provides compatibility support for different versions of Android and acts as the main activity that handles user interactions.

4. Variable Declaration

```
private EditText etUsername, etPassword;
    private TextView tvSignInResult;
```

Here, the app declares variables for the UI elements:

- etUsername and etPassword are EditText fields where users input their username and password.
- tvSignInResult is a TextView to display the result of the sign-in attempt (success or failure).

5. onCreate() Method

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

The onCreate method is called when the activity starts. Inside this method:

- setContentView(R.layout.activity_main); sets the layout for the activity from activity_main.xml.

6. Initialize Views

```
etUsername = findViewById(R.id.etUsername);
etPassword = findViewById(R.id.etPassword);
tvSignInResult = findViewById(R.id.tvSignInResult);
Button btnSignIn = findViewById(R.id.btnSignIn);
```

These lines link the Java variables with the UI components defined in activity_main.xml by their id.

7. Button Click Listener

btnSignIn.setOnClickListener(new View.OnClickListener()
{ @Override public void onClick(View v) {

A setOnClickListener is set up for the Sign In button. When the button is clicked, the onClick method executes.

8. Retrieve and Trim Input Text

```
String username = etUsername.getText().toString().trim();
String password = etPassword.getText().toString().trim();
```

- username and password are retrieved from the EditText fields.
- trim() removes any leading or trailing whitespace from the inputs.

9. Simple Sign-In Logic

```
                if (username.equals("admin") &&
password.equals("1234")) {
                    tvSignInResult.setText("Sign-In Successful");
                    Toast.makeText(MainActivity.this, "Welcome " +
username, Toast.LENGTH_SHORT).show();
                } else {
                    tvSignInResult.setText("Invalid Username or
Password");
                    Toast.makeText(MainActivity.this, "Sign-In Failed",
Toast.LENGTH_SHORT).show();
                }
            }
        });
    }
}
```

1-This section checks if the entered username is "admin" and the password is "1234".

2- If the condition is true, it means the sign-in was successful:

3- tvSignInResult.setText("Sign-In Successful"); updates the TextView to show "Sign-In Successful".
4- A Toast message displays a welcome message with the username.
5- If the condition is false, the sign-in fails:
6- tvSignInResult.setText("Invalid Username or Password"); updates the TextView to show "Invalid Username or Password".
7- A Toast message displays "Sign-In Failed."