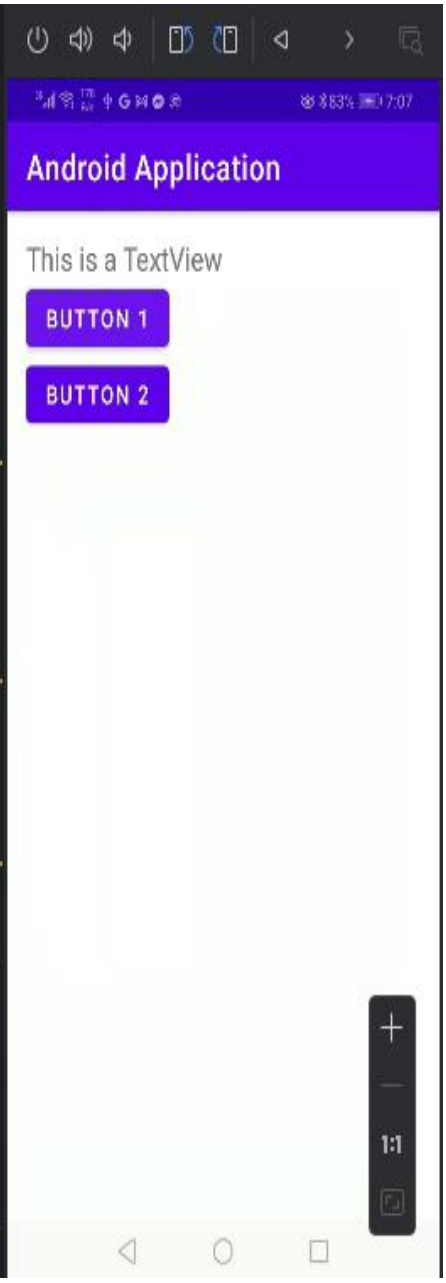
Unit 3: Designing the User Interface (5 Hours)

1. Android Layout Types

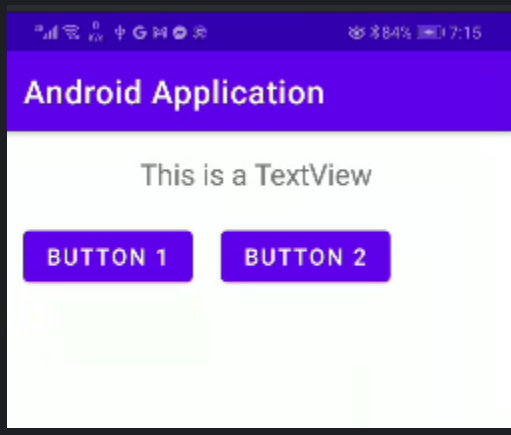
- **Linear Layout**
  - Aligns all children in a single direction, either vertically or horizontally.
  - Attributes:
    - `orientation`: Defines the direction (horizontal or vertical).
    - `layout_weight`: Distributes the extra space in the layout.
    - `gravity`: Defines the alignment of children within the layout.

```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">

        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="This is a TextView"
            android:textSize="18sp" />

        <Button
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Button 1" />

        <Button
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Button 2" />

</LinearLayout>
```



- **Relative Layout**
  - Positions children relative to each other or the parent.
  - Attributes:
    - `layout_alignParentTop`, `layout_alignParentBottom`, `layout_alignParentLeft`, `layout_alignParentRight`: Aligns the view relative to the parent.
    - `layout_below`, `layout_above`, `layout_toLeftOf`, `layout_toRightOf`: Positions the view relative to another view.

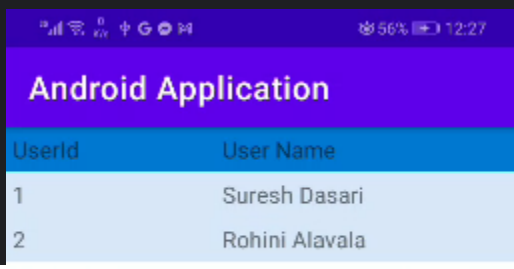
```xml
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:padding="16dp">

        <TextView
            android:id="@+id/textView"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="This is a TextView"
            android:textSize="18sp"
            android:layout_alignParentTop="true"
            android:layout_centerHorizontal="true" />

        <Button
            android:id="@+id/button1"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Button 1"
            android:layout_below="@id/textView"
            android:layout_marginTop="20dp"
            android:layout_alignParentLeft="true" />

        <Button
            android:id="@+id/button2"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Button 2"
            android:layout_below="@id/textView"
            android:layout_marginTop="20dp"
            android:layout_toRightOf="@id/button1"
            android:layout_marginLeft="20dp" />

</RelativeLayout>
```



- **Table Layout**
  - Organizes children into rows and columns, similar to an HTML table.
  - Attributes:
    - `stretchColumns`: Stretches specified columns to take up available space.
    - `shrinkColumns`: Shrinks specified columns if the content exceeds the available space.

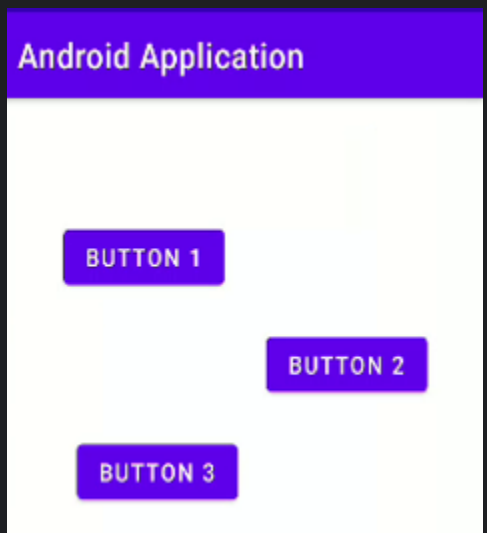
```xml
<?xml version="1.0" encoding="utf-8"?>
<TableLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_width="match_parent"
    android:stretchColumns="*">
        <TableRow android:background="#0079D6" android:padding="5dp">
                <TextView
                    android:layout_width="wrap_content"
                    android:layout_height="wrap_content"
                    android:text="UserId" />
                <TextView
                    android:layout_width="wrap_content"
```

```
                    android:layout_height="wrap_content"
                    android:text="User Name" />
        </TableRow>
        <TableRow android:background="#DAE8FC" android:padding="5dp">
                <TextView
                    android:layout_width="wrap_content"
                    android:layout_height="wrap_content"
                    android:text="1" />
                <TextView
                    android:layout_width="wrap_content"
                    android:layout_height="wrap_content"
                    android:text="Suresh Dasari" />
        </TableRow>
        <TableRow android:background="#DAE8FC" android:padding="5dp">
                <TextView
                    android:layout_width="wrap_content"
                    android:layout_height="wrap_content"
                    android:text="2" />
                <TextView
                    android:layout_width="wrap_content"
                    android:layout_height="wrap_content"
                    android:text="Rohini Alavala" />
        </TableRow>
</TableLayout>
```



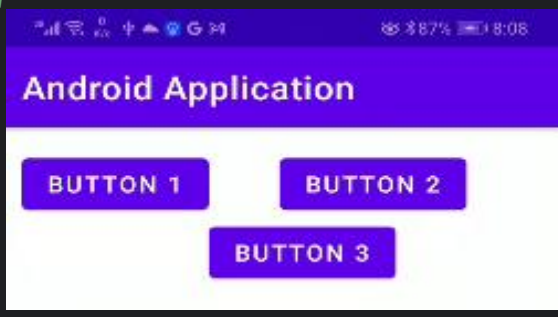
- **Absolute Layout** (Deprecated)
  - Allows you to specify the exact location of its children.
  - Attributes:
    - `layout_x`, `layout_y`: Specifies the exact X and Y coordinates of the view.

```
<?xml version="1.0" encoding="utf-8"?>
<AbsoluteLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent">
        <Button
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Button 1"
            android:layout_x="50dp"
            android:layout_y="80dp" />
        <Button
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Button 2"
            android:layout_x="200dp"
            android:layout_y="150dp" />
        <Button
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Button 3"
            android:layout_x="60dp"
            android:layout_y="220dp" />
</AbsoluteLayout>
```



- **Constraint Layout**
  - Flexible and powerful layout that allows you to position and size widgets in a flat hierarchy.
  - Attributes:
    - `layout_constraintLeft_toLeftOf`, `layout_constraintRight_toRightOf`: Defines constraints relative to other views or the parent.
    - `layout_constraintTop_toTopOf`, `layout_constraintBottom_toBottomOf`: Positions the view relative to other views or the parent.

```xml
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:padding="16dp">
        <Button
            android:id="@+id/button1"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Button 1"
            app:layout_constraintStart_toStartOf="parent" />
        <Button
            android:id="@+id/button2"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Button 2"
            app:layout_constraintStart_toEndOf="@id/button1"
            app:layout_constraintEnd_toEndOf="parent" />
        <Button
            android:id="@+id/button3"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Button 3"
            app:layout_constraintLeft_toRightOf="@id/button1"
            app:layout_constraintTop_toBottomOf="@id/button1" />
</androidx.constraintlayout.widget.ConstraintLayout>
```



2. Layout Attributes

- Common attributes applicable across all layouts:
  - `layout_width`, `layout_height`: Defines the width and height of the view.
  - `padding`: Adds space inside the view, between the view's content and its boundary.
  - `margin`: Adds space outside the view, between the view's boundary and the adjacent elements.

Attribute	Description	Example Value
`android:id`	Unique identifier for the widget, used to reference it in the code.	`@+id/myTextView`
`android:layout_width`	Specifies the width of the widget.	`wrap_content`, `match_parent`, `200dp`
`android:layout_height`	Specifies the height of the widget.	`wrap_content`, `match_parent`, `200dp`
`android:text`	Text to be displayed in the widget (e.g., TextView, Button).	`"Hello World!"`
`android:textColor`	Sets the color of the text.	`#FF0000`
`android:textSize`	Sets the size of the text.	`18sp`
`android:padding`	Adds space inside the widget between its edges and content.	`10dp`
`android:paddingLeft`	Adds padding to the left side of the widget.	`5dp`
`android:paddingRight`	Adds padding to the right side of the widget.	`5dp`
`android:paddingTop`	Adds padding to the top of the widget.	`5dp`
`android:paddingBottom`	Adds padding to the bottom of the widget.	`5dp`
`android:background`	Sets the background color or drawable of the widget.	`@color/blue`, `@drawable/bg_image`
`android:gravity`	Specifies how content (e.g., text) is aligned inside the widget.	`center`, `left`, `right`, `top`, `bottom`
`android:layout_margin`	Adds space outside the widget, separating it from other widgets or the parent container.	`16dp`
`android:layout_marginLeft`	Adds margin to the left side of the widget.	`10dp`
`android:layout_marginRight`	Adds margin to the right side of the widget.	`10dp`
`android:layout_marginTop`	Adds margin to the top side of the widget.	`10dp`

android:layout_marginBottom	Adds margin to the bottom side of the widget.	10dp
android:layout_gravity	Specifies how the widget itself is aligned within its parent container.	center, left, right
android:visibility	Controls whether the widget is visible, invisible, or gone.	visible, invisible, gone
android:clickable	Determines whether the widget can be clicked by the user.	true, false
android:enabled	Determines whether the widget is enabled, which affects user interaction (e.g., clickability).	true, false
android:hint	Sets a hint to be displayed in widgets like EditText when the text field is empty.	"Enter your name"
android:src	Sets the image source for an ImageView.	@drawable/icon, @mipmap/ic_launcher
android:scaleType	Controls how the image is scaled or resized in an ImageView.	center, fitCenter, fitXY
android:orientation	Specifies the orientation of child views in a LinearLayout (either horizontal or vertical).	horizontal, vertical
android:inputType	Sets the type of input that a user can enter in an EditText.	text, number, phone, password
android:singleLine	Restricts an EditText or TextView to a single line of text.	true, false
android:maxLines	Sets the maximum number of lines for text in widgets like TextView.	2, 3, 4
android:ellipsize	Controls how text is truncated when it exceeds the available width.	end, middle, start, marquee
android:ems	Sets the width of a TextView or EditText in terms of ems (a unit based on the width of the letter “M”).	10, 12, 15
android:focusable	Determines whether the widget can receive focus.	true, false
android:focusableInTouchMode	Determines whether the widget can receive focus in touch mode.	true, false
android:minWidth	Sets the minimum width of the widget.	100dp, 200dp
android:minHeight	Sets the minimum height of the widget.	50dp, 100dp
android:maxWidth	Sets the maximum width of the widget.	300dp, 400dp
android:maxHeight	Sets the maximum height of the widget.	200dp, 300dp

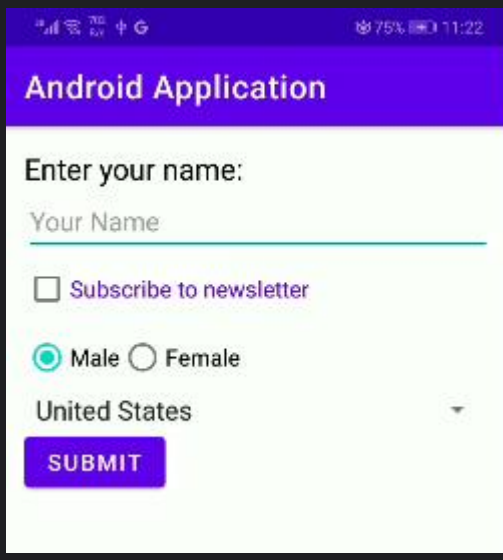
3. Android Widgets

- **TextView**
  - Displays text to the user.
  - Attributes: `text`, `textSize`, `textColor`, `gravity`.
- **EditText**
  - A user-editable text box.
  - Attributes: `hint`, `inputType`, `text`.
- **Checkbox**
  - A two-state button that can be either checked or unchecked.
  - Attributes: `checked`, `text`.
- **RadioButton**
  - Allows the user to select one option from a set.
  - Attributes: `checked`, `text`.
- **Spinner**
  - A drop-down list for selecting an item from a list.
  - Attributes: `entries`, `prompt`.

Example:

Activity_main.xml

```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">
        <TextView
            android:id="@+id/textView"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Enter your name:"
            android:textSize="18sp"
            android:textColor="#000000"
            android:gravity="left" />
        <EditText
            android:id="@+id/editTextName"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:hint="Your Name"
            android:inputType="textPersonName"
            android:textSize="16sp" />
        <CheckBox
            android:id="@+id/checkBox"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Subscribe to newsletter"
            android:checked="false" />
        <RadioGroup
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:orientation="horizontal">
```



```
                <RadioButton
                    android:id="@+id/radioButtonMale"
                    android:layout_width="wrap_content"
                    android:layout_height="wrap_content"
                    android:text="Male"
                    android:checked="true" />

                <RadioButton
                    android:id="@+id/radioButtonFemale"
                    android:layout_width="wrap_content"
                    android:layout_height="wrap_content"
                    android:text="Female" />
        </RadioGroup>
        <Spinner
            android:id="@+id/spinner"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:entries="@array/country_array"
            android:spinnerMode="dropdown" />
</LinearLayout>
```

String.xml

```
<resources>
    <string name="app_name">Android Application</string>
    <string-array name="country_array">
        <item>United States</item>
        <item>United Kingdom</item>
        <item>Canada</item>
        <item>Australia</item>
        <item>India</item>
    </string-array>
</resources>
```

4. Event Handling

- Events are the actions performed by the user in order to interact with the application, for e.g. pressing a button or touching the screen. The events are managed by the android framework in the FIFO manner i.e. First In – First Out. Handling such actions or events by performing the desired task is called Event Handling.
- Event Handling is the Process of responding to user interactions such as clicks, touches, etc.

To include the Event Handler in your application, you should know the following three concepts:

- Event Listeners
- Event Handlers
- Event Listener Registration

1. Event Listeners

An event listener is an interface in the View class of Android. It has a single callback method. This method will be called when the View that is registered with the Listener is activated by the user's action.

2. Event Handlers

Event handles are the actual methods that have the action that is to be taken. After an event has happened and event listeners are registered, event listeners call Event Handler. These are useful to define some callback methods.

3. Event Listener Registration

Event Registration is the process in which Event Listeners are registered with Event Handlers. This is important as Handler will work only after the Listener fires an Event.

Event Listeners and their respective event handlers

- **OnClickListener()** – This method is called when the user clicks, touches, or focuses on any view (widget) like Button, ImageButton, Image, etc. **Event handler used for this is onClick().**

```
Button button = findViewById(R.id.button);
button.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        // Handle the button click
    }
});
```

- **OnLongClickListener()** – This method is called when the user presses and holds a particular widget for one or more seconds. **Event handler used for this is onLongClick().**

```
button.setOnLongClickListener(new View.OnLongClickListener() {
    @Override
    public boolean onLongClick(View v) {
        // Handle the long click
        return true;
    }
});
```

- **OnKeyListener()** – This method is called when the user presses or releases a hardware key on the device. **Event handler used for this is onKey().**

```
button.setOnKeyListener(new View.OnKeyListener() {
    @Override
    public boolean onKey(View v, int keyCode, KeyEvent event) {
        if (event.getAction() == KeyEvent.ACTION_DOWN && keyCode == KeyEvent.KEYCODE_ENTER) {
            // Handle key press
            return true;
        }
        return false;
    }
});
```

- **OnFocusChangeListener()** – This method is called when the user goes away from the view item. **Event handler used for this is onFocusChange().**

```java
editText.setOnFocusChangeListener(new View.OnFocusChangeListener() {
    @Override
    public void onFocusChange(View v, boolean hasFocus) {
        if (hasFocus) {
            // Handle focus gained
        } else {
            // Handle focus lost
        }
    }
});
```

- **OnMenuItemClickListener()** – This method is called when the user selects a menu item. **Event handler used for this is onMenuItemClick().**

```java
menuItem.setOnMenuItemClickListener(new MenuItem.OnMenuItemClickListener() {
    @Override
    public boolean onMenuItemClick(MenuItem item) {
        // Code to handle menu item click
        return true;
    }
});
```

- **OnTouchListener()** – This method is called either for a movement gesture on the screen or a press and release of an on-screen key. **Event handler used for this is onTouch().**

```java
view.setOnTouchListener(new View.OnTouchListener() {
    @Override
    public boolean onTouch(View v, MotionEvent event) {
        // Code to handle touch event
        return true;
    }
});
```

- **OnItemSelectedListener()** – This method is called to handles item selections in a Spinner or ListView. **Event handler used for this is onItemSelected().**

```java
spinner.setOnItemSelectedListener(new AdapterView.OnItemSelectedListener() {
    @Override
    public void onItemSelected(AdapterView<?> parent, View view, int position, long id) {
        // Code to handle item selected
    }

    @Override
    public void onNothingSelected(AdapterView<?> parent) {
        // Code to handle no item selected
    }
});
```

- **OnCheckedChangedListener()** – This method is called to responds the changes in the state of CheckBoxes or RadioButtons. **Event handler used for this is onCheckedChanged().**

```
checkBox.setOnCheckedChangeListener(new CompoundButton.OnCheckedChangeListener() {
    @Override
    public void onCheckedChanged(CompoundButton buttonView, boolean isChecked) {
        // Code to handle check change
    }
});
```

Example:

XML Code:

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

JAVA Code:

```
Button button = findViewById(R.id.button);
button.setOnClickListener(new View.OnClickListener() {
@Override
public void onClick(View v) {
Toast.makeText(getApplicationContext(),"Button Clicked",Toast.LENGTH_SHORT).show();
}
});
```

5. Working with Strings, String Arrays, and Colors

- **Strings**:
  - Defined in the `res/values/strings.xml` file.
  - Accessed in the layout using `@string/string_name`.

String.xml

```
<resources>
<string name="semester">Sixth </string>
</resources>
```

Activity_main.xml:

```
<TextView
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:text="@string/semester" />
```

- **String Arrays**:
  - Defined in `res/values/strings.xml` using `<string-array>` tag.
  - Used for populating Spinner or ListView.

String.xml

```
<resources>
<string-array name="semesters">
        <item>First</item>
        <item>Second</item>
        <item>Third</item>
        <item>Fourth</item>
</string-array>
</resources>
```

Activity_main.xml

```
<Spinner
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:entries="@array/semesters"/>
```

- **Colors**:
  - Defined in `res/values/colors.xml`.
  - Used in the layout using `@color/color_name`.

Define colors in `res/values/colors.xml`:

```
<resources>
<color name="primaryColor">#FF6200EE</color>
<color name="secondaryColor">#FF03DAC5</color>
</resources>
```

Use them in layout (activity_main.html)

```
<TextView
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="Colored Text"
android:textColor="@color/primaryColor" />
```

6. Working with Resources and Drawables

- **Resources**:
  - Defined in the `res` folder and accessed using `@resource_type/resource_name`.
  - Examples: `@string/app_name`, `@color/primaryColor`.
- **Drawables**:
  - Images or shapes used in your app.
  - Stored in the `res/drawable` directory.
  - Accessed in the layout using `@drawable/drawable_name`.

With an image saved at res/drawable/myimg.png , this layout XML applies the image to a View.

```
<ImageView
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:src="@drawable/myimg" />
```

The following application code retrieves the image as a Drawable:

```
Resources res = getResources();
Drawable drawable = ResourcesCompat.getDrawable(res,
R.drawable.myimg, null);
```

7. Adding Icon to the Project

1. Place your icon image in the `res/drawable` directory.
2. Reference the icon in your layout or activity using `@drawable/icon_name`.
3. Set the icon for the app in the `AndroidManifest.xml` using the `android:icon` attribute within the `<application>` tag.

```
<application
      android:icon="@drawable/myicon "
       android:roundIcon="@drawable/myicon_round "
       android:label ="@string/app_name "
      ... >
       ...
 </application>
```

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.myapp">

    <application
        android:allowBackup="true"
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@drawable/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <activity android:name=".MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>

</manifest>
```