Experiment 12: Displaying Pictures

1. Objective

Create an application that uses a content provider to access and display data. This could be integrated into the to-do list app or another application, introducing the concept of content providers and data sharing between apps.

2. Steps to Complete the Experiment

1. Select or Create Image Resources:

   Choose or create a set of images that you want to display in your application. These can be pictures related to your app's theme or any other images you wish to showcase.

   Save these images in your project's res/drawable folder.

2. Design the UI Layout:

   Decide how you want to display the images in your application. Options include using a GridView for a grid layout, an ImageSwitcher for one image at a time with swiping, or a Gallery (deprecated, consider using a RecyclerView with a horizontal layout manager instead) for a horizontal list.

   For this example, let's use a GridView. Define the GridView in your activity's layout XML file (activity_main.xml).

```
<?xml version="1.0" encoding="utf-8"?>
<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"
    tools:context=".MainActivity">

    <GridView
        android:id="@+id/gridViewImages"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:numColumns="auto_fit"
        android:columnWidth="100dp"
        android:horizontalSpacing="10dp"
        android:verticalSpacing="10dp"
        android:gravity="center"
        android:stretchMode="columnWidth" />

</LinearLayout>
```

3. Create a Custom Adapter (if necessary):

   For a simple display, you might use an ArrayAdapter with built-in layouts. For more customization, such as with a GridView, create a custom adapter by extending BaseAdapter.

   Implement required methods in the adapter, such as getCount(), getItem(), getItemId(), and getView(). In getView(), inflate your custom layout for individual grid items and set the image for each item.

4. Implement Image Selection (optional):

   To add functionality for when a user selects an image, set an onItemClickListener on your GridView. In the listener, you could display the image in a larger view, start a new activity with the image details, or any other action you prefer.

   Step 1: Define Grid Item Layout

   First, create an XML layout file for individual grid items. Let's call this grid_item.xml. This layout will contain an ImageView that fills the grid cell.

```
<?xml version="1.0" encoding="utf-8"?>
<ImageView
xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/imageViewItem"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:scaleType="centerCrop" />
```

   Step 2: Create the Custom Adapter

   Next, create a new Java class for the custom adapter. Let's name it ImageAdapter. This class will extend BaseAdapter and override necessary methods.

```
import android.content.Context;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.BaseAdapter;
import android.widget.ImageView;

public class ImageAdapter extends BaseAdapter {
```

```java
private Context context;

private int[] imageIds; // Array of image resource IDs


// Constructor

public ImageAdapter(Context context, int[] imageIds) {

    this.context = context;

    this.imageIds = imageIds;

}


@Override

public int getCount() {

    return imageIds.length;

}


@Override

public Object getItem(int position) {

    return imageIds[position];

}


@Override

public long getItemId(int position) {

    return 0; // Not implemented

}


@Override
```

```
    public View getView(int position, View convertView, ViewGroup parent) {
        ImageView imageView;

        if (convertView == null) {
            // Inflate the layout for each grid item
            imageView = (ImageView) LayoutInflater.from(context).inflate(R.layout.grid_item, parent, false);
        } else {
            imageView = (ImageView) convertView;
        }

        // Set the image for the current position
        imageView.setImageResource(imageIds[position]);
        return imageView;
    }
}
```

Step 3: Use the Adapter in MainActivity

In MainActivity.java, instantiate the ImageAdapter with an array of image resource IDs and set it to the GridView.

```
import androidx.appcompat.app.AppCompatActivity;
import android.os.Bundle;
import android.widget.GridView;

public class MainActivity extends AppCompatActivity {
    GridView gridViewImages;

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

        gridViewImages = findViewById(R.id.gridViewImages);

        // Array of image resource IDs
        int[] imageIds = {
            R.drawable.image1,              R.drawable.image2,
R.drawable.image3, // Add your images
            // More images...
        };

        // Set the adapter to the GridView
        gridViewImages.setAdapter(new       ImageAdapter(this,
imageIds));
    }
}
```

Replace R.drawable.image1, R.drawable.image2, R.drawable.image3, etc., with the actual drawable resource IDs of your images.

5. Testing:

Test your application across different devices and orientations to ensure the images display correctly and the UI scales as expected.

3. Explanation:

LinearLayout: The root layout of the activity, set to a vertical orientation.

GridView (gridViewImages): The main UI component for displaying images in a grid. The attributes are configured to achieve a dynamic grid that adapts to various screen sizes:

android:numColumns="auto_fit": This setting allows the GridView to automatically adjust the number of columns based on the screen width and the specified column width.

android:columnWidth="100dp": Defines the width for each column. Depending on the screen size, more or fewer columns may be displayed to fit the screen width.

android:horizontalSpacing="10dp" and android:verticalSpacing="10dp": These add spacing between items in the grid, both horizontally and vertically.

android:gravity="center": Centers the GridView content within the layout.

android:stretchMode="columnWidth": Stretches items in columns to fill any available space, ensuring that the column width is maintained.