**Lab Report: Dynamic Flowers List Application with Image Visualization**

**TITLE**

Lab Report: Understanding and Implementing ListView in Android for a Dynamic Flowers List Application

**INTRODUCTION**

This report details the theoretical concepts and practical implementation of ListView within an Android mobile application designed to manage a dynamic list of owers. The application allows users to add and delete ower names, visualize associated images directly within the list, and view larger images and details on a separate screen. This exercise serves to illustrate fundamental Android UI development paerns, data management techniques, and inter-activity communication.

**OBJECTIVES**

The objectives of this lab report are:

● To understand the core concepts of ListView and its role in displaying dynamic lists in Android.

● To explain the necessity and function of Adapters (specically ArrayAdapter) in populating a ListView with custom item layouts.

● To demonstrate how to dene custom layouts for individual list items, including both text and images.

● To illustrate the implementation of dynamic adding and deleting of ower names from the list.

● To describe how ListView item clicks can trigger navigation to a separate detail activity, passing relevant data (ower name) for display.

● To utilize aractive UI design principles, including CardView and custom drawables, to enhance user experience.

**PROCEDURE / ANALYSIS / DESIGN**

**Analysis of ListView**

ListView is a UI component used for displaying a vertically scrolling collection of items. It is ecient in handling large datasets by recycling views, which means views that scroll o-screen are reused for new items that scroll into view, reducing memory consumption and improving performance. For this application, ListView was chosen to

present the dynamic collection of owers.

**Design Considerations**

The "Flowers List" application was designed with two main screens: a list view and a detail view.

1. **Main List Screen (activity\_main.xml):**

○ **Layout:** A LinearLayout serves as the root container, arranging elements vertically.

○ **Title:** A prominent TextView displays "My Flowers List."

○ **Input Section:** A horizontal LinearLayout contains an EditText (etFlowerName) for entering ower names and a Buon (btnAddFlower) for adding them. This section is visually grouped with padding, a rounded background (@drawable/rounded\_list\_bg), and a subtle shadow

(android:elevation).

○ **List Display:** A ListView (listViewFlowers) is used to show the list of owers. It is aesthetically enhanced by being nested within a

androidx.cardview.widget.CardView, providing rounded corners and a distinct elevated appearance. The ListView uses transparent dividers with increased height (android:dividerHeight="12dp") for beer visual separation between items.

2. **List Item Layout (list\_item\_ower.xml):** Each item in the ListView is a custom layout.

○ It's a horizontal LinearLayout that holds an ImageView (ivFlowerIcon) for the ower's picture, a TextView (tvFlowerNameItem) for its name, and a Buon (btnDeleteItem) for removal.

○ Custom drawables like rounded\_list\_item\_bg are applied for a consistent, aractive look.

3. **Detail Screen (activity\_ower\_detail.xml):**

○ A simple vertical LinearLayout displays a large TextView (tvDetailFlowerName) for the ower's name and a large ImageView (ivDetailFlowerImage) for its full picture. The ImageView also uses a rounded background for visual appeal. 4. **Data Management:**

○ An ArrayList<String> (owerList) stores the names of the owers added by the user.

○ A HashMap<String, Integer> (owerImageMap) is used to store the mapping between lowercase ower names (e.g., "rose") and their corresponding drawable resource IDs (e.g., R.drawable.rose). This allows for dynamic image loading based on the entered name.

5. **Navigation:** Intents are used to transition from MainActivity to FlowerDetailActivity, passing the selected ower's name.

**IMPLEMENTATION**

**activity\_main.xml (Main List Screen Layout)**

The provided activity-main-owers-list-xml immersive artifact denes the layout. Key elements include:

● Root LinearLayout with android:padding="24dp" and

android:background="#e0f2f7".

● Input LinearLayout with android:padding="12dp",

android:background="@drawable/rounded\_list\_bg", and android:elevation="4dp". ● EditText (@id/etFlowerName) and Buon (@id/btnAddFlower) for adding owers. ● ListView (@id/listViewFlowers) wrapped in a androidx.cardview.widget.CardView with app:cardCornerRadius="12dp" and app:cardElevation="6dp".

**list\_item\_ower.xml (Custom List Item Layout)**

This layout denes each row in the ListView. It includes an ImageView (@id/ivFlowerIcon), a TextView (@id/tvFlowerNameItem), and a Buon (@id/btnDeleteItem). Styling uses rounded\_list\_item\_bg and

gradient\_buon\_secondary.

**MainActivity.java (Main List Screen Logic)**

The core logic for the main list screen is handled in MainActivity.java.

● **Initialization of UI Components and Data Structures:**

public class MainActivity extends AppCompatActivity {

private EditText etFlowerName;

private Buon btnAddFlower;

private ListView listViewFlowers;

private List<String> owerList;

private FlowerAdapter owerAdapter;

private Map<String, Integer> owerImageMap;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_main);

etFlowerName = ndViewById(R.id.etFlowerName);

btnAddFlower = ndViewById(R.id.btnAddFlower);

listViewFlowers = ndViewById(R.id.listViewFlowers);

owerList = new ArrayList<>();

owerImageMap = new HashMap<>();

// Populate owerImageMap (e.g., owerImageMap.put("rose", R.drawable.rose);)

owerAdapter = new FlowerAdapter(this, R.layout.list\_item\_ower, owerList);

listViewFlowers.setAdapter(owerAdapter);

// ... (Listeners for Add buon and ListView items) ...

}

// ... (addFlower method and FlowerAdapter class) ...

}

● Custom FlowerAdapter (getView() method for item display and listeners): This custom ArrayAdapter is crucial for displaying the image and text in each list item and handling the delete buon.

private class FlowerAdapter extends ArrayAdapter<String> {

// ... (constructor and other methods) ...

@Override

public View getView(nal int position, View convertView, ViewGroup parent) { ViewHolder holder;

if (convertView == null) {

convertView = LayoutInater.from(getContext()).inate(layoutResource, parent, false);

holder = new ViewHolder();

holder.ivFlowerIcon = convertView.ndViewById(R.id.ivFlowerIcon); holder.tvFlowerNameItem =

convertView.ndViewById(R.id.tvFlowerNameItem);

holder.btnDeleteItem = convertView.ndViewById(R.id.btnDeleteItem); convertView.setTag(holder);

} else {

holder = (ViewHolder) convertView.getTag();

}

nal String ower = owers.get(position);

String displayFlowerName = ower.substring(0,

1).toUpperCase(Locale.getDefault()) +

ower.substring(1).toLowerCase(Locale.getDefault());

holder.tvFlowerNameItem.setText(displayFlowerName);

Integer imageResId = owerImageMap.get(ower);

if (imageResId != null) {

holder.ivFlowerIcon.setImageResource(imageResId);

} else {

holder.ivFlowerIcon.setImageResource(R.drawable.placeholder\_ower); }

holder.btnDeleteItem.setOnClickListener(new View.OnClickListener() { @Override

public void onClick(View v) {

owers.remove(position);

notifyDataSetChanged();

Toast.makeText(getContext(), displayFlowerName + " removed.", Toast.LENGTH\_SHORT).show();

}

});

return convertView;

}

private class ViewHolder { // Optimizes view recycling

ImageView ivFlowerIcon;

TextView tvFlowerNameItem;

Buon btnDeleteItem;

}

}

● **Adding Flowers (btnAddFlower OnClickListener):**

btnAddFlower.setOnClickListener(new View.OnClickListener() { @Override

public void onClick(View v) {

String owerName = etFlowerName.getText().toString().trim(); if (owerName.isEmpty()) {

Toast.makeText(MainActivity.this, "Please enter a ower name.",

Toast.LENGTH\_SHORT).show();

return;

}

String lowerCaseFlowerName =

owerName.toLowerCase(Locale.getDefault());

if (!owerList.contains(lowerCaseFlowerName)) {

owerList.add(lowerCaseFlowerName);

owerAdapter.notifyDataSetChanged();

etFlowerName.setText("");

Toast.makeText(MainActivity.this, owerName + " added to list.",

Toast.LENGTH\_SHORT).show();

} else {

Toast.makeText(MainActivity.this, owerName + " is already in the list.", Toast.LENGTH\_SHORT).show();

}

}

});

● **Item Click for Navigation (listViewFlowers OnItemClickListener):** listViewFlowers.setOnItemClickListener(new AdapterView.OnItemClickListener() { @Override

public void onItemClick(AdapterView<?> parent, View view, int position, long id) {

String selectedFlower = owerList.get(position);

Intent intent = new Intent(MainActivity.this, FlowerDetailActivity.class); intent.putExtra("FLOWER\_NAME", selectedFlower); // Pass data to detail activity

startActivity(intent);

}

});

**activity\_ower\_detail.xml (Detail Screen Layout)**

This layout displays the selected ower's details, featuring a TextView (@id/tvDetailFlowerName) for the name and a large ImageView

(@id/ivDetailFlowerImage) for the picture, both styled with rounded backgrounds.

**FlowerDetailActivity.java (Detail Screen Logic)**

The FlowerDetailActivity.java handles displaying the details of the selected ower.

● **Initialization and Data Retrieval:**

public class FlowerDetailActivity extends AppCompatActivity {

private TextView tvDetailFlowerName;

private ImageView ivDetailFlowerImage;

private Map<String, Integer> owerImageMap; // Same map as in MainActivity

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_ower\_detail);

tvDetailFlowerName = ndViewById(R.id.tvDetailFlowerName);

ivDetailFlowerImage = ndViewById(R.id.ivDetailFlowerImage);

// Populate owerImageMap again (must match MainActivity's map) owerImageMap = new HashMap<>();

owerImageMap.put("rose", R.drawable.rose);

// ... add all other ower mappings ...

String owerName = getIntent().getStringExtra("FLOWER\_NAME"); // Retrieve passed data

if (owerName != null && !owerName.isEmpty()) {

String displayFlowerName = owerName.substring(0,

1).toUpperCase(Locale.getDefault()) +

owerName.substring(1).toLowerCase(Locale.getDefault());

tvDetailFlowerName.setText(displayFlowerName);

Integer imageResId = owerImageMap.get(owerName);

if (imageResId != null) {

ivDetailFlowerImage.setImageResource(imageResId);

} else {

ivDetailFlowerImage.setImageResource(R.drawable.placeholder\_ower); }

} else {

tvDetailFlowerName.setText("No Flower Selected");

ivDetailFlowerImage.setImageResource(R.drawable.placeholder\_ower); }

}

}

**Drawable Resources**

Various XML drawable les (rounded\_ediext\_bg.xml, gradient\_buon\_primary.xml, gradient\_buon\_secondary.xml, rounded\_list\_bg.xml, rounded\_list\_item\_bg.xml, placeholder\_ower.xml) are used to provide consistent and aractive styling across the application's UI elements, including rounded corners, gradient buons, and subtle shadows. Actual ower image les (e.g., rose.png, lily.jpg) are placed in res/drawable/.

**CONCLUSION**

This laboratory exercise successfully demonstrated the creation of a dynamic list application using ListView in Android, specically for managing a list of owers. The objectives of dynamically adding and deleting list elements, displaying images within the list items, and navigating to a detail screen with full image visualization were met. The implementation highlighted the importance of custom ArrayAdapter and the ViewHolder paern for ecient list rendering. Furthermore, the application incorporated aractive UI design principles through the use of CardView and custom XML drawables, resulting in a visually appealing and user-friendly experience. This project reinforced core Android development concepts essential for building interactive and data-driven mobile applications.