****

**Department of Computer Science and Engineering (CSE)**

**Faculty of Sciences and Engineering**

**Semester: (Summer, Year:2025), B.Sc. in CSE (Day)**

**LAB REPORT NO: 2**

**Course Title: Mobile Application Development Lab**

**Course Code: CSE 426**

**Section: 221\_D1**

**Lab Experiment Name: Implementing ListView in Android**

**Student Details**

| **Name** | **ID** |
| --- | --- |
| **Tajmin Khanam** | **221002566** |

**Lab Date : 16.07.2025**

**Submission Date : 22.07.2025**

**Course Teacher’s Name : Mr. Montaser Abdul Quader**

**[For Teachers use only: Don’t Write Anything inside this box]**

| **Lab Report Status**  **Marks: ………………………………… Signature: .....................**  **Comments: .............................................. Date: ..............................** |
| --- |

# **TITLE OF THE LAB EXPERIMENT:**

Implementing ListView in Android

# **INTRODUCTION:**

This report details the theoretical concepts and practical implementation of ListView within an Android mobile application designed to manage a dynamic list of owners. The application allows users to add and delete owner 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 patterns, data management techniques, and inter-activity communication.

# **OBJECTIVES:**

The objectives of this lab report are:

# **PROCEDURE / ANALYSIS / DESIGN:**

ListView is designed to display a vertically scrolling list of views. It is particularly useful when the number of items is not fixed and can grow or shrink dynamically. Its core strength lies in its ability to efficiently manage a large dataset by recycling views, meaning it only creates enough views to fill the screen and reuses them as the user scrolls.

For the "Fruits List" application, ListView was chosen to present a scrollable list of fruit names, each accompanied by its corresponding image and a delete button. The design involved:

1. **Main Layout (activity\_main.xml):** Incorporating a ListView widget to occupy a significant portion of the screen, allowing ample space for displaying fruit items. The ListView itself was placed within a CardView to provide a modern, elevated visual appearance with rounded corners and a shadow.
2. **List Item Layout (list\_item\_fruit.xml):** Designing a custom layout for each row of the ListView. This layout included an ImageView for the fruit icon, a TextView for the fruit name, and a Button for deleting the item. This custom layout ensures that each list entry is visually rich and interactive.
3. **Data Management:** Utilizing an ArrayList<String> to hold the names of the fruits and a HashMap<String, Integer> to map these fruit names to their respective image resource IDs (R.drawable.fruit\_name). This allows for dynamic retrieval of images based on the fruit name.
4. **User Interaction:** Implementing mechanisms for adding new fruits via an EditText and a "Add" button, and for deleting existing fruits via a "Delete" button within each list item. Crucially, clicking on a list item (the fruit name or image) was designed to navigate to a separate detail screen, passing the selected fruit's name.

# **IMPLEMENTATION:**

**activity\_main.xml (ListView Integration)**

The ListView is declared in the main layout file:

*<ListView  
    android:id="@+id/listViewFruits"  
    android:layout\_width="match\_parent"  
    android:layout\_height="0dp"  
    android:layout\_weight="1"  
    android:padding="12dp"  
    android:divider="@android:color/transparent"  
    android:dividerHeight="12dp"/>*

It is wrapped in a androidx.cardview.widget.CardView for enhanced aesthetics, providing rounded corners and elevation.

**list\_item\_fruit.xml (Custom Item Layout)**

This XML defines the structure of each item displayed in the ListView:

*<LinearLayout  
    android:layout\_width="match\_parent"  
    android:layout\_height="wrap\_content"  
    android:orientation="horizontal"  
    android:padding="12dp"  
    android:background="@drawable/rounded\_list\_item\_bg"  
    android:gravity="center\_vertical">  
  
    <ImageView  
        android:id="@+id/ivFruitIcon"  
        android:layout\_width="48dp"  
        android:layout\_height="48dp"  
        android:scaleType="centerCrop"  
        android:layout\_marginEnd="12dp"  
        android:contentDescription="Fruit Icon"  
        android:src="@drawable/placeholder\_fruit"/>  
  
    <TextView  
        android:id="@+id/tvFruitNameItem"  
        android:layout\_width="0dp"  
        android:layout\_height="wrap\_content"  
        android:layout\_weight="1"  
        android:text="Fruit Name"  
        android:textSize="18sp"  
        android:textColor="#333333"  
        android:textStyle="medium"/>  
  
    <Button  
        android:id="@+id/btnDeleteItem"  
        android:layout\_width="wrap\_content"  
        android:layout\_height="wrap\_content"  
        android:text="Delete"  
        android:background="@drawable/gradient\_button\_secondary"  
        android:textColor="#FFFFFF"  
        android:paddingVertical="8dp"  
        android:paddingHorizontal="16dp"  
        android:textSize="14sp"  
        android:stateListAnimator="@null"/>  
</LinearLayout>*

**MainActivity.java (ListView Logic)**

The MainActivity.java class orchestrates the ListView's behavior:

1. **Initialization:**  
   *listViewFruits = findViewById(R.id.listViewFruits);  
   fruitList = new ArrayList<>();  
   fruitImageMap = new HashMap<>();  
   R.drawable IDs  
   fruitAdapter = new FruitAdapter(this, R.layout.list\_item\_fruit, fruitList);  
   listViewFruits.setAdapter(fruitAdapter);*
2. **Adding Items:** The *addFruit()* method adds a new fruit name (converted to lowercase) to fruitList and calls *fruitAdapter.notifyDataSetChanged()* to refresh the ListView.
3. **Deleting Items:** Within the FruitAdapter's getView() method, an OnClickListener is set for *btnDeleteItem*. When clicked, it removes the item from fruitList and calls *notifyDataSetChanged()*.
4. **Item Click for Navigation:** An AdapterView.OnItemClickListener is set on listViewFruits:  
   *listViewFruits.setOnItemClickListener(new AdapterView.OnItemClickListener() {  
       @Override  
       public void onItemClick(AdapterView<?> parent, View view, int position, long id) {  
           String selectedFruit = fruitList.get(position);  
           Intent intent = new Intent(MainActivity.this, FruitDetailActivity.class);  
           intent.putExtra("FRUIT\_NAME", selectedFruit);  
           startActivity(intent);  
       }  
   });*
5. **Custom FruitAdapter:** This inner class extends ArrayAdapter<String> and overrides getView() to inflate list\_item\_fruit.xml, set the fruit name, load the corresponding image from fruitImageMap, and attach the delete button's listener. The ViewHolder pattern is used to optimize performance by caching view lookups.

# **OUTPUT:**

.

# **CONCLUSION:**

# This laboratory exercise successfully demonstrated the creation of a dynamic list application using ListView in Android, specifically for managing a list of owners. 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 pattern for efficient list rendering. Furthermore, the application incorporated attractive 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.