Experiment 7

Experiment 7: Implement navigation from the ListView to a dedicated screen displaying detailed restaurant information (name, address)

Lab Objective:
Implement navigation from a ListView to a detail screen that displays the selected restaurant's name and address.

Prerequisites:

1. Kotlin and Android basics
2. Working ListView or local database with restaurant records
3. Android Studio environment

Outcome:
• ListView displays restaurants.
• Tapping a list item opens a detail screen.
• Detail screen shows name and address passed via Intent extras.

Step 1: activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:padding="12dp"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <ListView
        android:id="@+id/listView"
        android:layout_width="match_parent"
        android:layout_height="match_parent" />
</LinearLayout>
```

Step 2: row_restaurant.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
```

```
    android:orientation="vertical"
    android:padding="12dp"
    android:layout_width="match_parent"
    android:layout_height="wrap_content">

    <TextView
        android:id="@+id/text1"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:textSize="16sp"/>

    <TextView
        android:id="@+id/text2"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:textSize="14sp"/>
</LinearLayout>
```

Step 3: activity_detail.xml

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

    <LinearLayout
        android:orientation="vertical"
        android:layout_width="match_parent"
        android:layout_height="wrap_content">

        <TextView
            android:id="@+id/tvName"
            android:textSize="20sp"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:paddingBottom="8dp"/>

        <TextView
            android:id="@+id/tvAddress"
            android:textSize="16sp"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"/>
    </LinearLayout>
</ScrollView>
```

Step 4: MainActivity.kt

```
import android.content.Intent
import android.database.Cursor
import android.os.Bundle
import android.widget.AdapterView
import android.widget.ListView
import android.widget.SimpleCursorAdapter
import androidx.appcompat.app.AppCompatActivity

class MainActivity : AppCompatActivity() {

    private lateinit var listView: ListView
    private lateinit var dbHelper: DatabaseHelper
    private lateinit var cursor: Cursor
    private lateinit var adapter: SimpleCursorAdapter

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)

        listView = findViewById(R.id.listView)
        dbHelper = DatabaseHelper(this)

        val db = dbHelper.readableDatabase
        cursor = db.rawQuery(
            "SELECT id AS _id, name, category, address FROM restaurants ORDER BY name
ASC",
            null
        )

        adapter = SimpleCursorAdapter(
            this,
            R.layout.row_restaurant,
            cursor,
            arrayOf("name", "category"),
            intArrayOf(R.id.text1, R.id.text2),
            0
        )
        listView.adapter = adapter

        listView.onItemClickListener = AdapterView.OnItemClickListener { _, _, position, _ ->
            val itemCursor = listView.adapter.getItem(position) as Cursor
            val name = itemCursor.getString(itemCursor.getColumnIndexOrThrow("name"))
            val address = itemCursor.getString(itemCursor.getColumnIndexOrThrow("address"))
?: ""
```

```
            val intent = Intent(this@MainActivity, DetailActivity::class.java).apply {
                putExtra("name", name)
                putExtra("address", address)
            }
            startActivity(intent)
        }
    }

    override fun onDestroy() {
        super.onDestroy()
        if (::cursor.isInitialized && !cursor.isClosed) cursor.close()
    }
}
```

Step 5: DetailActivity.kt

```
import android.os.Bundle
import android.widget.TextView
import androidx.appcompat.app.AppCompatActivity

class DetailActivity : AppCompatActivity() {

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_detail)

        val tvName = findViewById<TextView>(R.id.tvName)
        val tvAddress = findViewById<TextView>(R.id.tvAddress)

        val name = intent.getStringExtra("name") ?: "Unknown"
        val address = intent.getStringExtra("address") ?: "Address not available"

        tvName.text = name
        tvAddress.text = address
    }
}
```

Step 6: AndroidManifest.xml (register DetailActivity)

```
<application
    ... >
    <activity android:name=".DetailActivity" />
```

```
    <activity android:name=".MainActivity">
        <intent-filter>
            <action android:name="android.intent.action.MAIN" />
            <category android:name="android.intent.category.LAUNCHER" />
        </intent-filter>
    </activity>
</application>
```

Explanation:

1. MainActivity queries the local DB for id, name, category, and address.
2. SimpleCursorAdapter binds name and category to each list row.
3. On item click, the row's Cursor is used to extract name and address.
4. An Intent starts DetailActivity with extras "name" and "address".
5. DetailActivity reads the extras and displays them.

Test Cases:

1. Tap a list row → Detail screen opens and shows correct name and address.
2. Row with null/empty address → Detail shows "Address not available".
3. Large address text → Scrollable UI displays full address.
4. App restart → Navigation still works (persistent DB).

Notes:

• If using an object list instead of Cursor, pass necessary fields via Intent or make the data class Parcelable/Serializable.
• For RecyclerView, use the same intent-passing pattern in the item click listener.