Green University of Bangladesh

Department of Computer Science and Engineering (CSE)
Semester: (Summer, Year: 2025), B.Sc. in CSE (Day)

Lab Report 3: Font Changer Android Application With List of Image & Title

Course Title: Mobile Application Development Lab
Course Code: CSE-426
Section: 222-D2

Students Details

Name	ID
Md. Romjan Ali	222002127

Submission Date: 13 Jul 2025
Course Teacher's Name: Sudip Chandra Ghoshal

[For teachers use only: Don't write anything inside this box]

Lab Project Status	
Marks:	**Signature:**
Comments:	**Date:**

1 Objective

The objective of this lab is to develop an Android application that demonstrates the use of custom fonts and ListView. The application will display a list of available fonts, and upon selecting a font from the list, the text in an EditText field will be updated to use the selected font. Each font in the list will be represented by an image and its name.

2 Procedure

The application is built using Android Studio and Gradle. The user interface is defined in XML layout files, and the application logic is implemented in Java.

The main components of the application are:

- **MainActivity.java:** The main activity of the application, responsible for initializing the UI, populating the ListView with font data, and handling user interactions.
- **FontAdapter.java:** A custom adapter to display the list of fonts in the ListView. Each item in the list consists of an image and the font name.
- **FontItem.java:** A simple data class to hold the name and image resource ID for each font.
- **activity_main.xml:** The layout file for the main activity, containing an EditText and a ListView.
- **angila_list.xml:** The layout file for a single item in the font list.
- **Custom Fonts:** The custom font files (TTF) are placed in the ‘assets/fonts‘ directory.

When the application starts, the MainActivity creates a list of FontItem objects, each representing a font with its name and a corresponding image. This list is then used to populate a ListView using a custom FontAdapter. An OnItemClickListener is set on the ListView to detect when a user selects a font. Upon selection, the Typeface of the EditText is changed to the selected font, which is loaded from the assets folder.

3 Implementation

The core logic of the application is implemented in ‘MainActivity.java‘. This class manages the user interface and handles the font selection.

```
package com.romjan.labfour;

import android.graphics.Typeface;
import android.os.Bundle;
import android.view.View;
```

```
import android.widget.AdapterView;
import android.widget.EditText;
import android.widget.ListView;

import androidx.appcompat.app.AppCompatActivity;

import java.util.ArrayList;
import java.util.List;

public class MainActivity extends AppCompatActivity {
    EditText etInput;
    ListView listFont;

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

        etInput = findViewById(R.id.etInput);
        listFont = findViewById(R.id.listFont);

        // Create list of FontItem objects
        List<FontItem> fonts = new ArrayList<>();
        fonts.add(new FontItem("Angila Tattoo", R.drawable.angila));
        fonts.add(new FontItem("Cantate Beveled", R.drawable.cantia));
        fonts.add(new FontItem("Krinkes Decor PERSONAL", R.drawable.krinker));
        fonts.add(new FontItem("Krinkes Regular PERSONAL", R.drawable.kunkes));
        fonts.add(new FontItem("Silent Reaction", R.drawable.silent));

        FontAdapter adapter = new FontAdapter(this, fonts);
        listFont.setAdapter(adapter);

        listFont.setOnItemClickListener(new AdapterView.OnItemClickListener() {
            @Override
            public void onItemClick(AdapterView<?> parent, View view, int position
                FontItem selectedItem = fonts.get(position);
                String fontName = selectedItem.name;

                String fontPath = null;
                switch (fontName) {
                    case "Angila Tattoo":
                        fontPath = "fonts/AngillaTattoo.ttf";
                        break;
                    case "Cantate Beveled":
                        fontPath = "fonts/CantateBeveled.ttf";
                        break;
                    case "Krinkes Decor PERSONAL":
                        fontPath = "fonts/KrinkesDecorPERSONAL.ttf";
```

```
                    break;
                case "Krinkes Regular PERSONAL":
                    fontPath = "fonts/KrinkesRegularPERSONAL.ttf";
                    break;
                case "Silent Reaction":
                    fontPath = "fonts/SilentReaction.ttf";
                    break;
            }

            if (fontPath != null) {
                Typeface tf = Typeface.createFromAsset(getAssets(), fontPath);
                etInput.setTypeface(tf);
            }
        }
    });
  }
}
```

3.1 The FontAdapter is a custom adapter for the ListView.

```
package com.romjan.labfour;

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

import java.util.List;

public class FontAdapter extends BaseAdapter {

    Context context;
    List<FontItem> fontList;

    public FontAdapter(Context context, List<FontItem> fontList) {
        this.context = context;
        this.fontList = fontList;
    }

    @Override
    public int getCount() {
        return fontList.size();
    }
```

```
    @Override
    public Object getItem(int position) {
        return fontList.get(position);
    }

    @Override
    public long getItemId(int position) {
        return position;
    }

    @Override
    public View getView(int position, View convertView, ViewGroup parent) {
        View view = LayoutInflater.from(context).inflate(R.layout.angila_list, par

        ImageView imageView = view.findViewById(R.id.fontIcon);
        TextView textView = view.findViewById(R.id.fontName);

        FontItem item = fontList.get(position);

        textView.setText(item.name);
        imageView.setImageResource(item.imageResId);

        return view;
    }
}
```

3.2 The FontItem class is a simple data holder.

```
package com.romjan.labfour;

public class FontItem {
    String name;
    int imageResId;

    public FontItem(String name, int imageResId) {
        this.name = name;
        this.imageResId = imageResId;
    }
}
```

3.3 The main layout of the application is defined in 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"
```

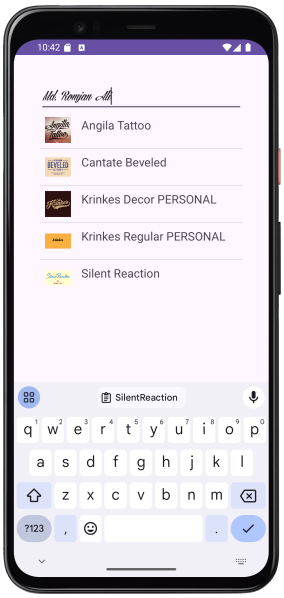
```
    android:id="@+id/main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:padding="40dp"
    android:orientation="vertical"
    tools:context=".MainActivity">

    <EditText
        android:id="@+id/etInput"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:ems="10"
        android:inputType="textPersonName"
        android:hint="Enter a Name"
        />

    <ListView
        android:id="@+id/listFont"
        android:layout_width="match_parent"
        android:layout_height="wrap_content" />

</LinearLayout>
```

3.4 The layout for each item in the ‘ListView‘ is defined in ‘angila_list.xml‘.

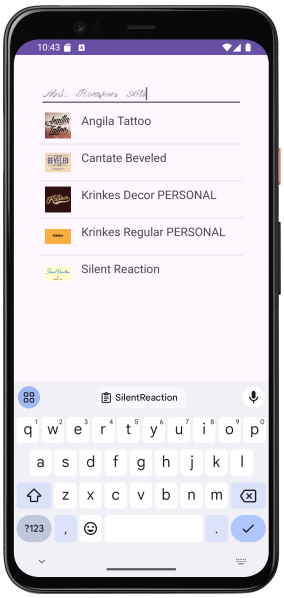
```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="horizontal"
    android:padding="8dp"
    android:layout_width="match_parent"
    android:layout_height="wrap_content">

    <ImageView
        android:id="@+id/fontIcon"
        android:layout_width="40dp"
        android:layout_height="40dp"
        android:layout_marginEnd="16dp"
        android:contentDescription="List Item"
        android:src="@drawable/angila" />

    <TextView
        android:id="@+id/fontName"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textSize="18sp" />
</LinearLayout>
```

4 Output

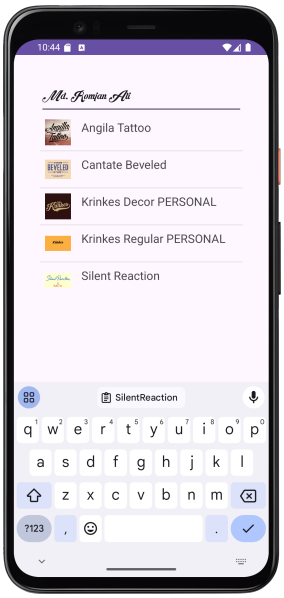
The following screenshots show the application in action.



(a) Screenshot 1

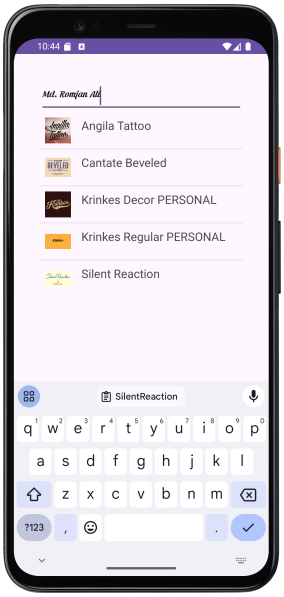


(b) Screenshot 2

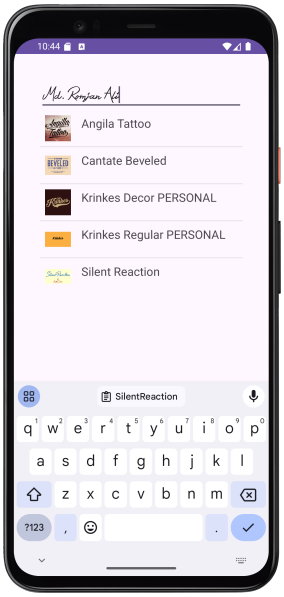


(c) Screenshot 3

Figure 1: Application Screenshots (Row 1)



(a) Screenshot 4



(b) Screenshot 5

Figure 2: Application Screenshots (Row 2)

5 Discussion

This lab successfully demonstrates the implementation of a custom font changer application in Android. The use of a ‘ListView‘ with a custom ‘FontAdapter‘ allows for a dynamic and user-friendly way to display and select fonts. The application effectively loads and applies custom fonts from the ‘assets‘ directory to an ‘EditText‘ widget.

The project is well-structured, with clear separation of concerns between the data (‘FontItem‘), the adapter (‘FontAdapter‘), and the main activity (‘MainActivity‘). This makes the code easy to understand and maintain.

One possible improvement could be to dynamically scan the ‘assets/fonts‘ directory for available fonts instead of hardcoding the font names and paths in the ‘MainActivity‘. This would make the application more scalable and easier to update with new fonts. Additionally, error handling could be improved, for example, by displaying a message to the user if a font file fails to load.

Overall, this lab provides a solid foundation for understanding how to work with custom UI components and resources in Android development.