Exercise 2 - Keyboard application

Objective

To implement a keyboard application using Android Studio

Algorithm

1. Initialize the Project:
   - Open Android Studio and create a new Android project.
   - Set up the project with a suitable name and package.

2. Design the Keyboard Layout:
   - Create an XML layout file to design the keyboard interface.
   - Use buttons or keys to represent each character or action on the keyboard.

3. Handle Button Clicks:
   - In the Java code, set up event listeners for button clicks.
   - Define actions for each button press, such as appending characters to a text field

4. Implement Basic Keyboard Functions:
   - Include keys for letters, numbers, and common symbols.

5. Toggle Shift State:
   - Implement a shift key that toggles between letters and numbers.
   - Update the key labels dynamically based on the shift state.

Features used

- Dynamic Layout: XML layout to design the keyboard interface.

- Button Click Handling: Event listeners for button clicks.
- Shift State Toggle: Toggle between Characters and numbers/symbols
- Text Input Integration: Linking the keyboard to a text input field.

Source Code

activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
   xmlns:tools="http://schemas.android.com/tools"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   android:orientation="vertical"
   tools:context=".MainActivity">

   <TextView
       android:id="@+id/text_bar"
       android:layout_width="match_parent"
       android:layout_height="wrap_content"
       android:layout_marginLeft="6dp"
       android:layout_marginTop="24dp"
       android:layout_marginRight="6dp"
       android:hint="Enter text..."
       android:text=""
       android:textSize="24sp" />

   <GridLayout
       android:id="@+id/keyboard"
       android:layout_width="match_parent"
       android:layout_height="200dp"
       android:layout_alignParentBottom="true"
       android:background="#202020"
       android:padding="1dp" />
</RelativeLayout>
```

MainActivity.java

```
package com.example.qwerty;
```

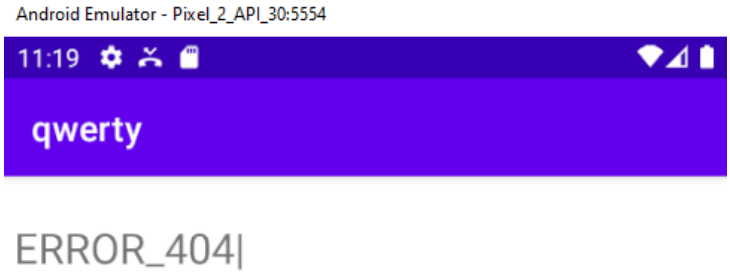
```
import androidx.appcompat.app.AppCompatActivity;
import androidx.core.content.res.ResourcesCompat;
import android.os.Bundle;
import android.util.DisplayMetrics;
import android.view.Gravity;
import android.widget.Button;
import android.widget.GridLayout;
import android.widget.TextView;
import android.graphics.Typeface;

public class MainActivity extends AppCompatActivity {
   private final String[][] letterKeys = {
           {"Q", "W", "E", "R", "T", "Y", "U", "I", "O", "P"},
           {"A", "S", "D", "F", "G", "H", "J", "K", "L"},
           {"↑", "Z", "X", "C", "V", "B", "N", "M", "←"},
           {"?123", ",", " ", ".", "⏎"}
   };
   private final String[][] numberKeys = {
           {"1", "2", "3", "4", "5", "6", "7", "8", "9", "0"},
           {"@", "#", "₹", "_", "&", "-", "+", "(", ")", "/"},
           {"*", "\"", "'", ":", ";", "!", "?", "<", ">", "=", "←"},
           {"ABC", ",", " ", ".", "⏎"}
   };
   private TextView textBar;

   private GridLayout keyboard;
   private DisplayMetrics displayMetrics;
   private Boolean isCaps = false;
   private Typeface customFont;

   private void init() {
       textBar = findViewById(R.id.text_bar);
       textBar.setText("|");
       keyboard = findViewById(R.id.keyboard);

       displayMetrics = new DisplayMetrics();
       getWindowManager().getDefaultDisplay().getMetrics(displayMetrics);

       customFont = ResourcesCompat.getFont(this, R.font.oswald);
   }
   @Override
```

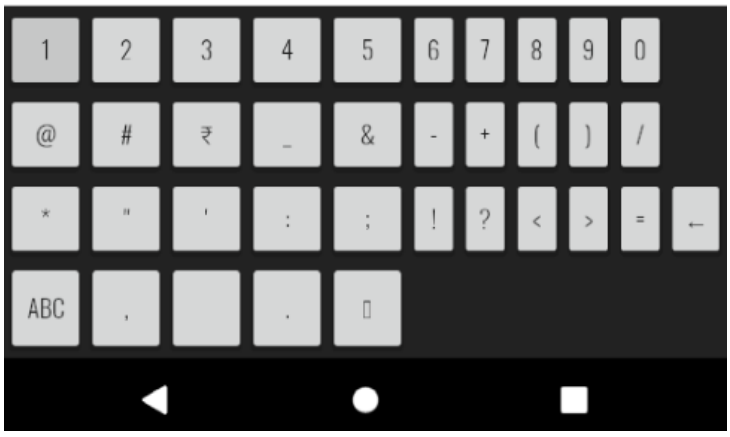
```
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        init();
        initKeyboard(letterKeys);
    }

    private void initKeyboard(String[][] keys) {
        keyboard.removeAllViews();

        int rowIndex = 0;
        for (String[] row : keys) {
            int colIndex = 0;
            int width = displayMetrics.widthPixels / (row.length + 4);

            for (String letter : row) {
                addButtonToKeyboard(letter, width, rowIndex, colIndex);
                colIndex++;
            }
            rowIndex++;
        }
    }

    private boolean isChar(String key) {
        return key.length() == 1 && Character.isLetter(key.charAt(0));
    }

    private void addButtonToKeyboard(String key, int width, int row, int col)
{
        Button button = new Button(this);
        if (isChar(key)) button.setText(key.toLowerCase());
        else {
            button.setText(key);
            button.setTypeface(customFont);
            button.setWidth(0);
            button.setPadding(0, 0, 0, 0);
        }

        GridLayout.LayoutParams params = new GridLayout.LayoutParams();
        params.width = width;
        params.height = GridLayout.LayoutParams.WRAP_CONTENT;
```

```
        params.rowSpec = GridLayout.spec(row);
        params.columnSpec = GridLayout.spec(col);
        params.setGravity(Gravity.FILL);

        button.setLayoutParams(params);
        button.setClickable(true);
        setHandler(button, key);

        keyboard.addView(button);
    }

    private void setHandler(Button button, String key) {
        if (key.length() == 1 && Character.isLetter(key.charAt(0))) {
            button.setOnClickListener(view -> {
                if (isCaps) textBar.append(key.toUpperCase());
                else textBar.append(key.toLowerCase());
            });
            return;
        }

        switch (key) {
            case "↑":
                button.setOnClickListener(view -> isCaps = !isCaps);
                break;
            case "←":
                button.setOnClickListener(view -> {
                    CharSequence currentText = textBar.getText();
                    if(currentText.length()!=0) {
                        CharSequence newText = currentText.subSequence(0,
currentText.length() - 1);
                        textBar.setText(newText);
                    }
                });
                break;
            case "?123":
                button.setOnClickListener(view -> initKeyboard(numberKeys));
                break;
            case "ABC":
                button.setOnClickListener(view -> initKeyboard(letterKeys));
                break;
            case "⍰":
                button.setOnClickListener(view -> textBar.append("\n"));
```

```
                break;
            default:
                button.setOnClickListener(view -> {
                        CharSequence currentText = textBar.getText();
                        CharSequence newText = currentText.subSequence(0,
currentText.length() - 1);
                        textBar.setText(newText);
                        textBar.append(String.valueOf(key.charAt(0)));
                        textBar.append("|");
                    }
                );
                break;
        }

    }
}
```

Output Screenshots





Result

Thus a keyboard was implemented

Best Practices

1. User friendly keyboard design
2. Readable layouts

Learning outcomes

- Basic GUI components and layouts that are available
- Passing data between activities
- Manual keyboard