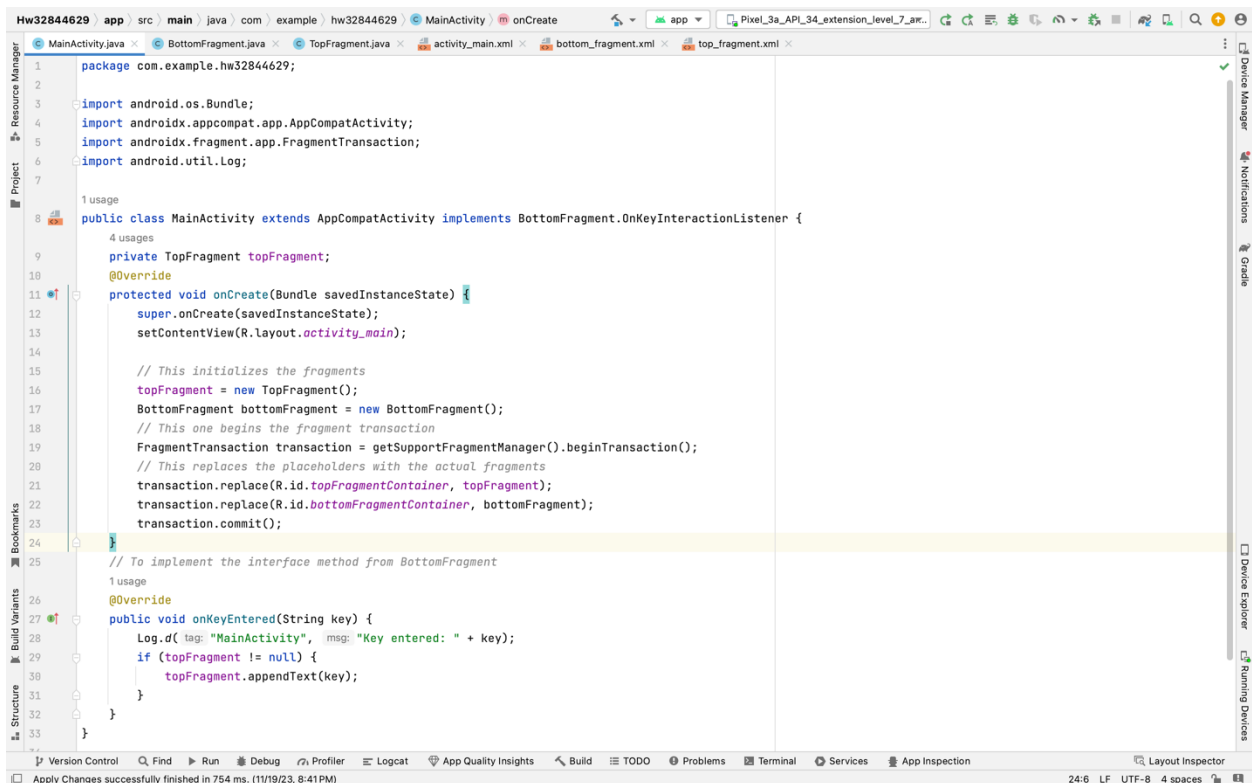


Develop a notepad app with custom keypad:

- Top fragment: display the note entered.
- Bottom fragment: display several rows of letters, numbers, and symbols (each as a button) for text input; when a user pushes a button, the corresponding input is appended in the top fragment.

This assignment is about creating an app which has two fragment the top fragment and bottom fragment where the bottom fragment has the all the button with letters, numbers and symbols which when type will be displayed on the top fragment.

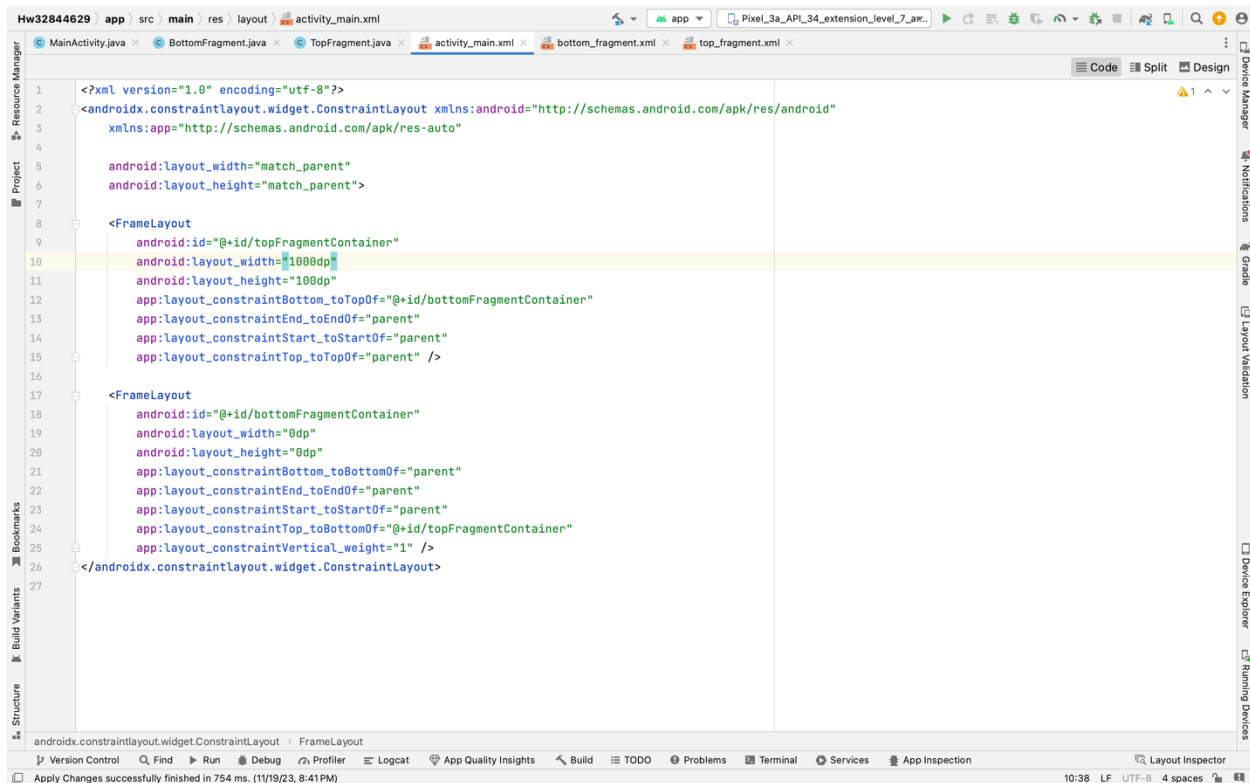
**MainActivity.java:** This is the app's primary class; it oversees managing the fragments, communicating between the UI and the java classes between the top and bottom fragments, and serving as the entry point for engaging with the user. When a user presses or enters a key or provides input to the bottom fragment, the MainActivity receives the input and, using the method "onkeyEntered," updates the top fragment with each new input the user inputs in the bottom fragment. I'm also utilizing several other methods, such as "onCreate," "FragmentTransaction," and "OnKeyInteraction," to help with initializing the activity, embedding the top and bottom fragments, and handling keypresses from the bottom fragment.



```
1 package com.example.hw32844629;
2
3 import android.os.Bundle;
4 import androidx.appcompat.app.AppCompatActivity;
5 import androidx.fragment.app.FragmentTransaction;
6 import android.util.Log;
7
8 1 usage
9 public class MainActivity extends AppCompatActivity implements BottomFragment.OnKeyInteractionListener {
10     4 usages
11     private TopFragment topFragment;
12     @Override
13     protected void onCreate(Bundle savedInstanceState) {
14         super.onCreate(savedInstanceState);
15         setContentView(R.layout.activity_main);
16
17         // This initializes the fragments
18         topFragment = new TopFragment();
19         BottomFragment bottomFragment = new BottomFragment();
20         // This one begins the fragment transaction
21         FragmentTransaction transaction = getSupportFragmentManager().beginTransaction();
22         // This replaces the placeholders with the actual fragments
23         transaction.replace(R.id.topFragmentContainer, topFragment);
24         transaction.replace(R.id.bottomFragmentContainer, bottomFragment);
25         transaction.commit();
26
27         // To implement the interface method from BottomFragment
28         1 usage
29         @Override
30         public void onKeyEntered(String key) {
31             Log.d("MainActivity", "msg: " + key);
32             if (topFragment != null) {
33                 topFragment.appendText(key);
34             }
35         }
36     }
37 }
```

**MainActivity.xml :** In mainactivity.xml, we design the main activity's user interface. We divide the screen into two halves by using "TopFragmentContainer" for the top fragment and "BottomFragmentContainer" for the bottom fragment. We also set the size of the fragments using these methods and we use "ConstraintLayout" as the root. To achieve a responsive UI, the frame layouts are positioned and enlarged relative to one another. This configuration effectively

produces a split screen within the UI in which the top and bottom fragments work independently but cooperatively within the same activity.



**BottomFragment.java** : The primary function of the “BottomFragment” is to create a custom keypad with in a “GridLayout” . When attached to an activity, this fragment creates an interface “OnKeyInteractionListener” for signaling key presses to the activity. The “onCreateView” method dynamically adds buttons representing letters (A-Z), numerals (0-9), and different symbols (such as semicolon, @, #, and so on) to the “GridLayout”. When a button is clicked, the linked activity is notified via "mListener.onKeyEntered" and the relevant character is sent as input. I'm generating and configuring this button with the required text and click listeners within the grid using the "addButton" method.

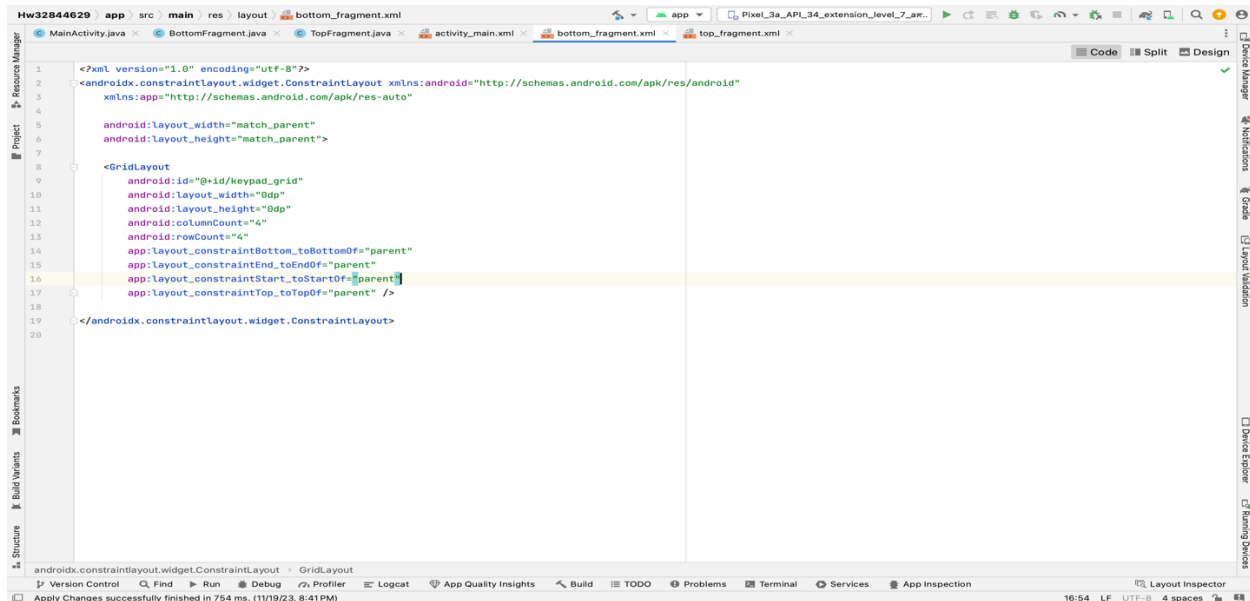
```

1 package com.example.hw32844629;
2
3 import android.content.Context;
4 import android.os.Bundle;
5 import android.view.LayoutInflater;
6 import android.view.View;
7 import android.view.ViewGroup;
8 import android.widget.Button;
9 import androidx.annotation.NonNull;
10 import androidx.annotation.Nullable;
11 import androidx.fragment.app.Fragment;
12 import android.widget.GridLayout;
13
14 public class BottomFragment extends Fragment {
15     private OnKeyInteractionListener mListener;
16
17     public interface OnKeyInteractionListener {
18         void onKeyEntered(String key);
19     }
20
21     @Override
22     public void onAttach(@NonNull Context context) {
23         super.onAttach(context);
24         if (context instanceof OnKeyInteractionListener) {
25             mListener = (OnKeyInteractionListener) context;
26         } else {
27             throw new RuntimeException(context.toString() + " must implement OnKeyInteractionListener");
28         }
29     }
30
31     @Nullable
32     @Override
33     public View onCreateView(@NonNull LayoutInflater inflater, @Nullable ViewGroup container, @Nullable Bundle savedInstanceState) {
34         View view = inflater.inflate(R.layout.bottom_fragment, container, attachToRoot: false);
35         GridLayout gridLayout = view.findViewById(R.id.keypad_grid);
36
37         // To add letter buttons
38         for (char letter = 'A'; letter <= 'Z'; letter++) {
39             addButton(String.valueOf(letter), gridLayout);
40         }
41
42         // To add number buttons
43         for (int number = 0; number <= 9; number++) {
44             addButton(String.valueOf(number), gridLayout);
45         }
46
47         // now adding the symbols and space
48         addButton(text: ";", gridLayout);
49         addButton(text: "@", gridLayout);
50         addButton(text: "#", gridLayout);
51         addButton(text: "$", gridLayout);
52         addButton(text: "%", gridLayout);
53         addButton(text: "&", gridLayout);
54         addButton(text: ":", gridLayout);
55         addButton(text: "_", gridLayout);
56         addButton(text: "-", gridLayout);
57         addButton(text: "+", gridLayout);
58         addButton(text: " ", gridLayout);
59
60         return view;
61     }
62
63     private void addButton(String text, GridLayout gridLayout) {
64         Button button = new Button(getContext());
65         button.setText(text);
66         button.setOnClickListener(v -> mListener.onKeyEntered(text));
67         gridLayout.addView(button);
68     }
69 }

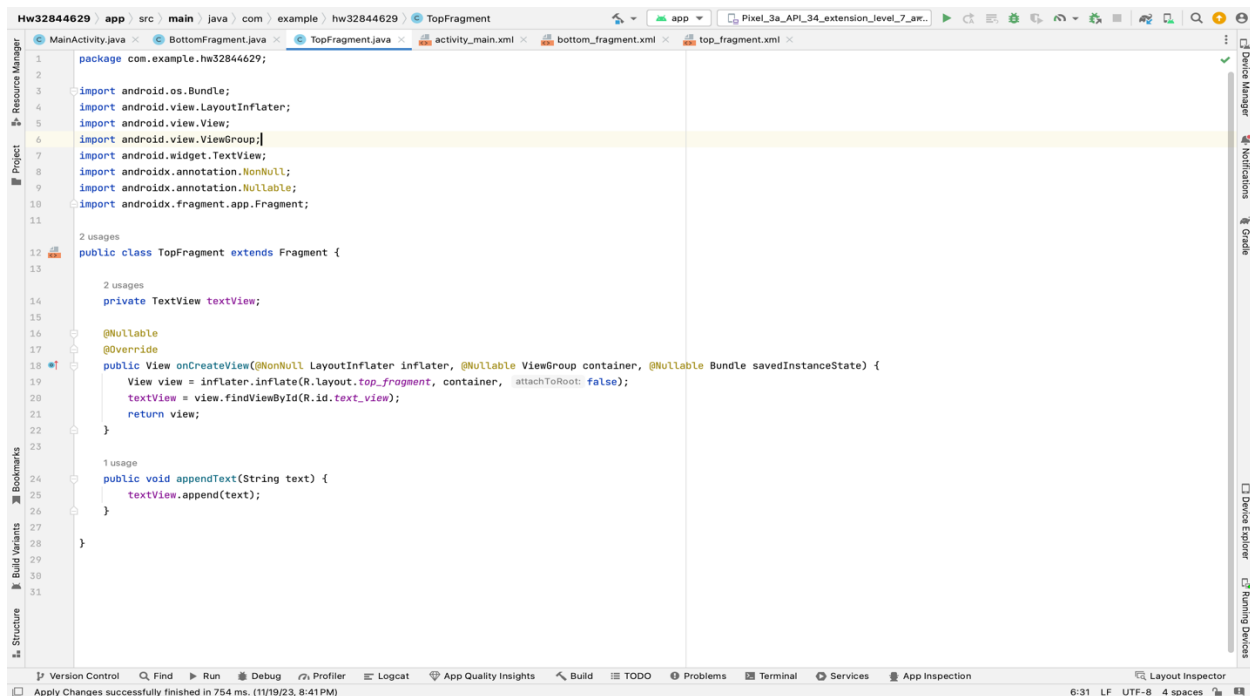
```

**Bottomfragment.xml** : A 'GridLayout' is utilized within a 'ConstraintLayout' in this XML layout to create a grid-based layout . The 'GridLayout' is set to span the whole available space

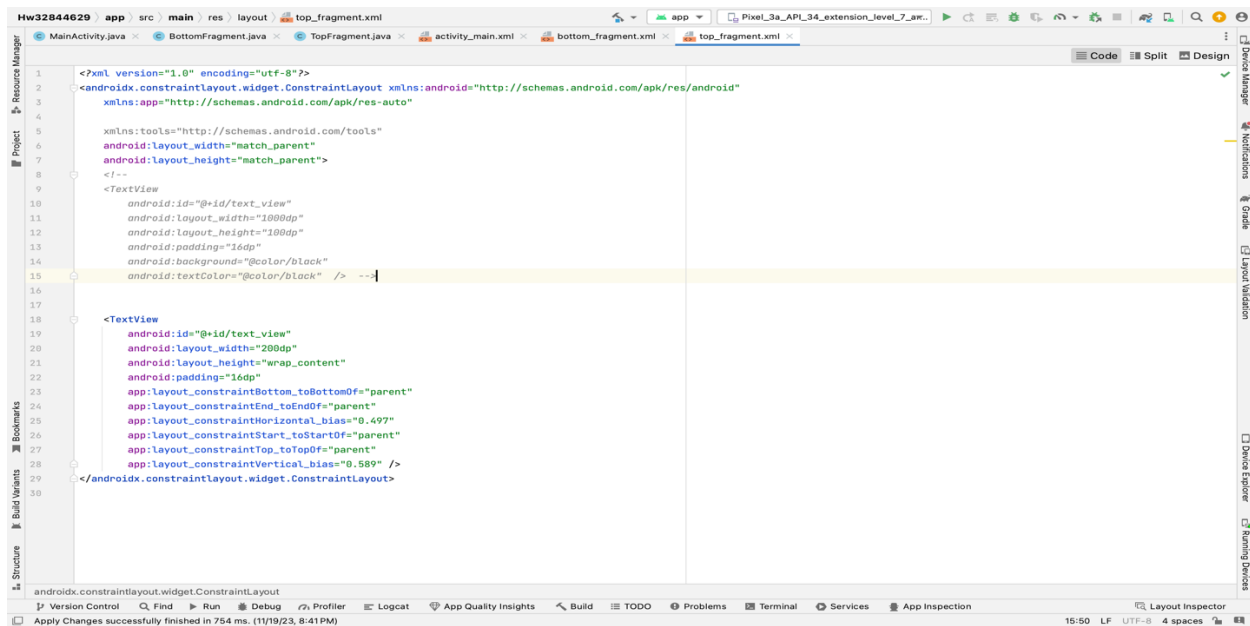
“match\_parent” for both width and height and has a given number of columns and rows  
 “columnCount=“4””. The 'GridLayout' is limited to align with the boundaries of the parent  
 'ConstraintLayout', ensuring that it takes up the whole area of the parent layout.



**TopFragment.java** : this class is responsible for displaying text and to manage and update a  
 “TextView” within its layout. here we are using methods like  
 “onCreateView”, “appendText”, “TextViewInitialization” for creating and setting up the fragment  
 view, to display the text in the fragment and to append “text” to the “TextView”.



x : This XML layout defines a user interface “ConstraintLayout”. It has a single “TextView” that has a fixed width 200dp and a variable height “wrap\_content”. This arrangement is commonly used to show text in a visually centered manner on a screen. The TextView is centered in the parent ConstraintLayout using constraint settings, and it includes padding around its content for improved reading and appearance.



Below are the screenshots of the applications



Screenshot 1(main activity)



Screenshot 2(input using the keyboard)