Project in Android Development 236272 // Spring 2022

Dry 1

Exercise 1 - Dry**:**

1. The 2 lines of code that make the list infinitely scrolling are:

```
if (index >= _suggestions.length) {
                _suggestions.addAll(generateWordPairs().take(10));
 }
```

   If we remove these lines and scroll to the end of the list, no new lines will be generated other that the 10 wordPairs that was generated in the first place. So that when we will try to access the next line, we won't get any new lines and specifically here we will get an exception. That is because the next line is generated by next index that doesn't exist in the list range anymore as we didn't generate the next 10 lines to be suggested in the list.

2. We can use a ListView.separated constructor instead for this list, where the itemBuilder will build all child items with separator children between them.
   This way is better because there is no need to handle the divider after each child item, especially when there is a fixed number of children (given that that the list is finite and contains 100 items from the start).

3. SetState is one of the methods to render the changes that are made dynamically into the UI. We need the setState() call inside the onTap() handler for when we want to reflect the change in a variable on the UI of the screen after the user had clicked on the icon. In our case, it's for when we add or remove a wordPair to the saved words list.

Exercise 2 - Dry**:**

1. The purpose of the MaterialApp widget is to be the "Root widget" in our widgets tree, which contains all other widgets and navigates through-out the material design app, the over-all themes and required structure.
   For example, here are 3 of the MaterialApp widget properties that are in use:
   - **title:** 'Startup Name Generator', this is the description of the app for the device.
   - **theme:** by using the ThemeData class we can describe the theme for the MaterialApp and make it more homogeneous.
   - **home:** the widget that will be shown on the default route of the app. In our case, we chose the RandomWords() widget.

2. The Dismissible widget has a key property which specifies the widgets we want to dismiss under this action. It is required to uniquely identify the widgets within the collection and sync their state correctly after the dismiss was made. The key property must not be null or else, the default behavior would be to sync widgets based on their index in the list. This means that the item after the dismissed item would be synced with the state of the dismissed item. Therefore, we use the keys to avoid this unsuspected issue.