Dry 1 – Part 1

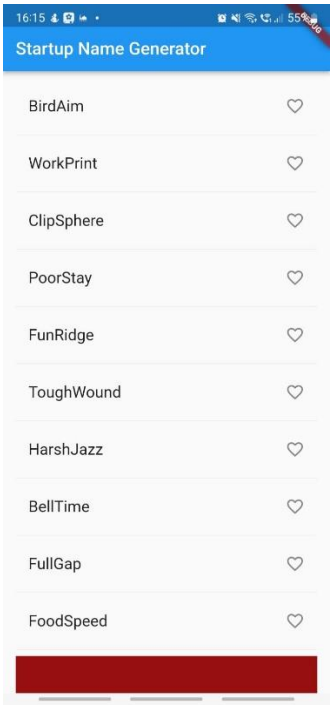
1.

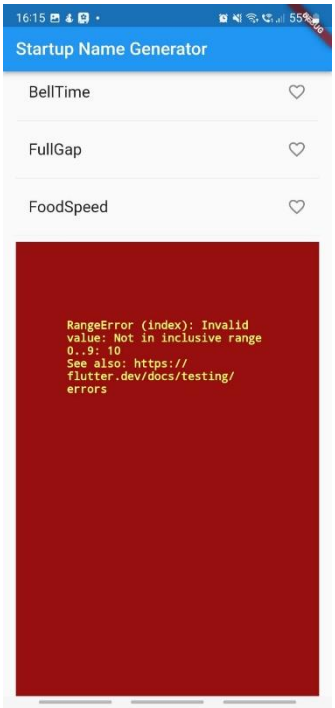
```
if (index >= _suggestions.length) {
  _suggestions.addAll(generateWordPairs().take(10)); /*4*/
}
```

If we remove these two lines we get the result below. I changed the initializing of _suggetions like described in the homework file:

final _suggestions = generateWordPairs() .take(10). toList;

In the previous code (before the changes) each time we got to an index that equals/bigger than the number of words that has been generated, the code will generate 10 more words and so on. Meanwhile, in the new code we generate 10 words at the start and stop generating more. However, the index is still growing but has no words to show so it writes an error message telling us that we are out of range of the list.





2. Another method is using ListView. separated instead of ListView. builder, which creates a fixed-length scrollable linear array of list "items" separated by list item "separators", moreover this constructor is appropriate for list views with a large number of item and separator children because the builders are only called for the children that are actually visible.

   Meanwhile ListView. builder creates a scrollable, linear array of widgets that are created on demand. This constructor is appropriate for list views with a large (or infinite) number of children because the builder is called only for those children that are actually visible.

   so, since we were told to assume that the list is finite and contains only 100 items from the start then the better method is ListView. separated because it can create a fixed-length scrollable linear array of list, while ListView. builder creates scrollable, linear array of widgets that are created on demand.
3. Calling setState() is critical, because this tells the framework that the widget's state has changed, and that the widget should be redrawn.

   when the user presses on the widget it shrinks down in size (meaning the padding becomes less than it was) so it needs to be fixed to its original state, so the setState() tells Flutter to rebuild your widget

Dry 1 – Part 2

1. MaterialApp widget is the starting point of your app, it tells Flutter that you are going to use Material components and follow the material design in your app. MaterialApp is a widget that introduces a number of widgets Navigator, Theme that are required to build a material design app.

   3 Properties of MaterialApp widget:

   1. **color**: It controls the primary color used in the application.
   2. **theme**: This property takes in ThemeData class as the object to describe the theme for the MaterialApp.
   3. **title**: The title property takes in a string as the object to decide the one-line description of the app for the device.

2. You are required to pass key (Key) and child (Widget). The key is the identifier for each widget in the Dismissible list of widgets. Since the widget can be removed from the widget list we need a key to remove it and it has to be a unique one not the index because it can be changed as we remove widgets.