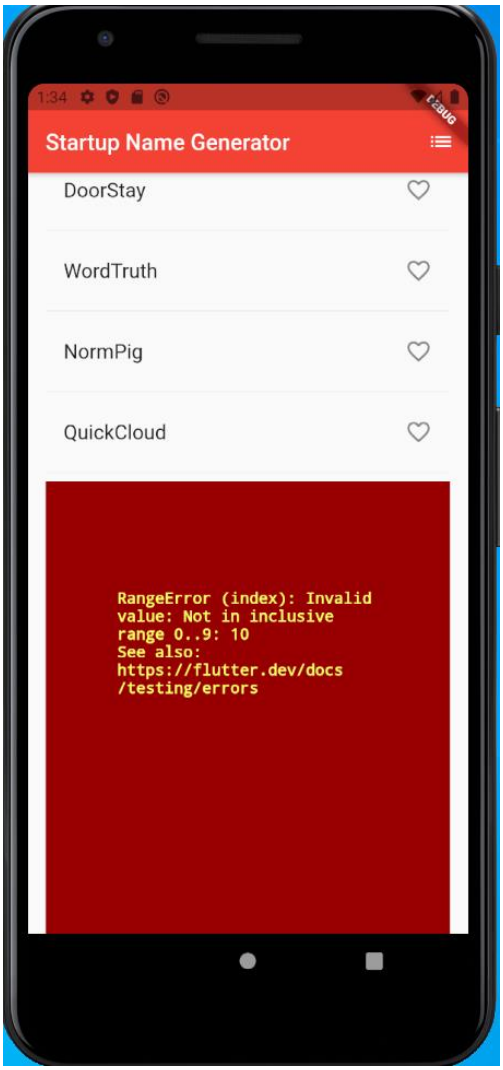
Exercise 1, dry part:

Question 1:

The 2 lines of code are
if (index >= _suggestions.length) {
  _suggestions.addAll(generateWordPairs().take(10));
}

These lines ensure that if we reach the last pair of words while scrolling (index >= _suggestions.length) , then we add another 10 pairs. Removing them and adding only 10 pairs of words in the initialization results in an index out of range exception after scrolling to the end:



Question 2:

A different method would be to use the “ListView” constructor, which takes a fixed list of widgets (in our case the same list will contain the word pairs and dividers), and constructs the elements of all the items in the given list (including those that are not visible yet). “ListView.builder” constructs elements lazily as they become visible for the user, which is a better and more efficient approach for lists with a large number of items (100 in our case) since the children of the list are created on demand as they become visible, which prevents creating all the elements from the start even though the user might not use most of them.

Creating all the elements from the start (as in the “ListView” constructor approach) causes the initial load time of the app to be slower, and holds memory even though the user might not need to interact with all the elements of the list, so in short it does unnecessary work in many cases. “ListView.builder” solves this problem by creating elements on demand as they become visible for the user.

Question 3:

In Flutter's reactive style framework, calling **setState()** triggers a call to the **build()** method for the **State** object, resulting in an update to the UI.

We want the UI to be updated as soon as we interact (click on a row) with the app, which is exactly what setState does. Without setState we would click on a row, the wordPair would be added to favorites but we won’t see the “full heart” unless we refresh/ hot reload the app, which makes setState a must to make the app interactive.

Exercise 2, dry part:

1. I used “Navigator.of(context).push( );” in order to navigate to a different page, as explained in part 2 section 6 of Google’s Codelab on the subject . A different method is to use “pushNamed()” where we define names route for every page. For example the home page is ‘/’, favorites page is ‘/favorites’, and when we click on the favorites button, we use Navigator.pushNamed(context, ‘/favorites’), which achieves the same result. Ref: https://flutter.dev/docs/cookbook/navigation/named-routes

2. I used “ScaffoldMessenger.*of*(context).showSnackBar(snackBar);” to show snackbar. Scaffold is required in order to use this method. We define and show the SnackBar inside Scaffold’s body.