Ramzi eleryan id.211412101

1. **Identify the 2 lines of code that make the list infinitely scrolling. What happens if we remove these lines, and scroll to the end of the list?**

   A:

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

   when you remove them, and you scroll down you will get an error of index out of range since we reached a tile index that is beyond the length of the _suggestion list.

   ---

2. **Our list is separated with Divider widgets. Visit the ListView widget documentation here: https://api.flutter.dev/flutter/widgets/ListView-class.html; Give a different method to construct such a list with dividers. Which way do you think is better, and why? You may assume for this question only that the list is finite and contains 100 items from the start.**

   A:

```
76      body: ListView.separated(
77      padding: const EdgeInsets.all(16.0),
78      itemCount: 100,
79      itemBuilder: (context, i) {
80        /// so that the even list view items contain the values - word pairs -
81        /// while the odd list view items are dividers
82        // if (i.isOdd) return const Divider();
83        final index = i ~/ 2;
84        if (index >= _suggestions.length) {
85          _suggestions.addAll(generateWordPairs().take(10));}
86        final alreadySaved = _saved.contains(_suggestions[index]); // NEW
87        return ListTile(
88          title: Text(_suggestions[index].asPascalCase, style: _biggerFont),
89          trailing: Icon(
90
91            alreadySaved ? Icons.favorite : Icons.favorite_border,
92            color: alreadySaved ? Colors.red : null,
93            semanticLabel: alreadySaved ? 'Remove from saved' : 'Save',
94          ),  // Icon
95 >        onTap: () {···
04        ); // ListTile
05      },
06      separatorBuilder: (BuildContext context, int index) => const Divider(),) // ListView.separated
```

the other method is to use ListView.separated with separator builder, this method divides the tiles automatically without having to deal with it manually as we did, in our solution we chose to have the odd number of tiles to be the dividers, thus we only used half of the tiles for content / company names. But on the other hand, it made it possible to have an infinite number of tiles, unlike the builder which needed a given finite number of tiles. So, in terms of which is preferable, it depends on the situation. Ps. I tried to check with copilot and Claude for ways to have separated builder with infinite number of tiles, but it didn't work.

---

3. _buildRow contains a call to setState() inside the onTap() handler. Why do we need it there?

A: to maintain the reactivity of the application. When we Tap on the heart icon of any tiles, we meant to change the state of the application and add this tile's text's name to favorites, so by having SetState() there we can change the state of the application and notify all the relevant widgets about it.

Answer the following questions:

Q1. What is the purpose of the MaterialApp widget? Provide examples of 3 of its properties followed by a short explanation.

A: the materialApp widget is an important foundation widget that follows Material design for flutter applications, so it wraps our entire application and configures the overall visual and settings for Material applications just like an android application. 3 properties are : *

theme: which controls the visual aspect / theme of the application. * home: specifies our home screen once the app is launched * title: gives a title to the app

Q2. The Dismissible widget has a key property. What does it mean and why is it required?

A: Every widget has a key, which differentiates it from the others, which helps with maintaining the app's state clear especially after rebuilds and updates. For dismissible widgets, the key (which i defined for example as : ValueKey<WordPair>(pair), helps with identifying the widget for instance when the list updates or and we want to know the tiles that are swiped, or when the list rebuilds and we want to identify the pairs of widgets with their contents, or when the tile is currently being swiped and to provide clear continuous animation instantly for the current tile.
