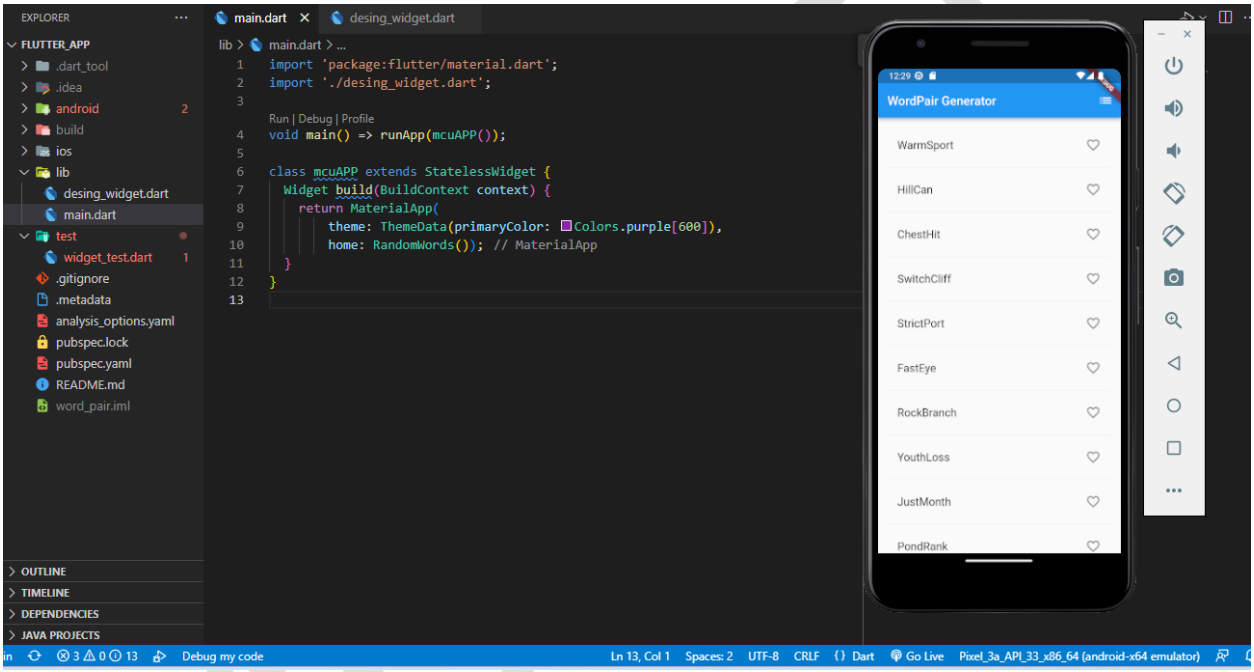
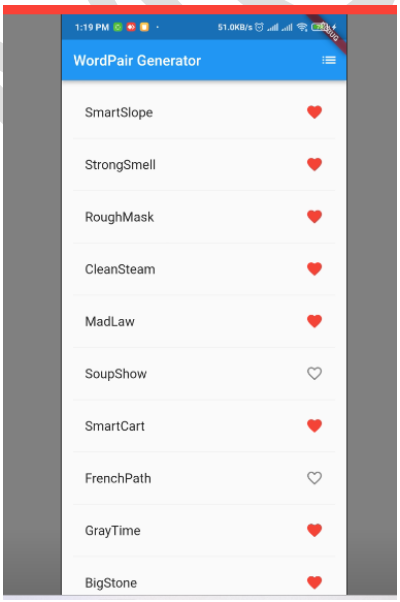
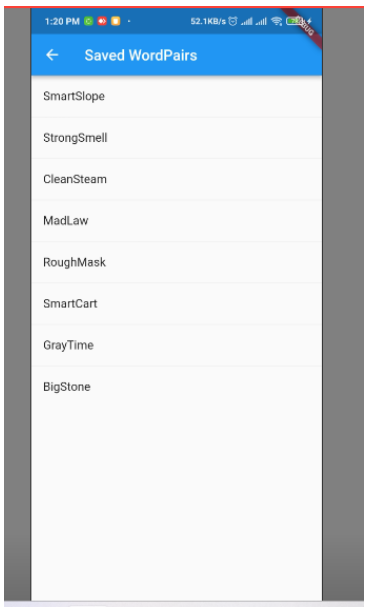
Name: Md. Mohacel Hosen
ID: 193071002
Semester: 10
Department: CSE

Flutter Word Pair







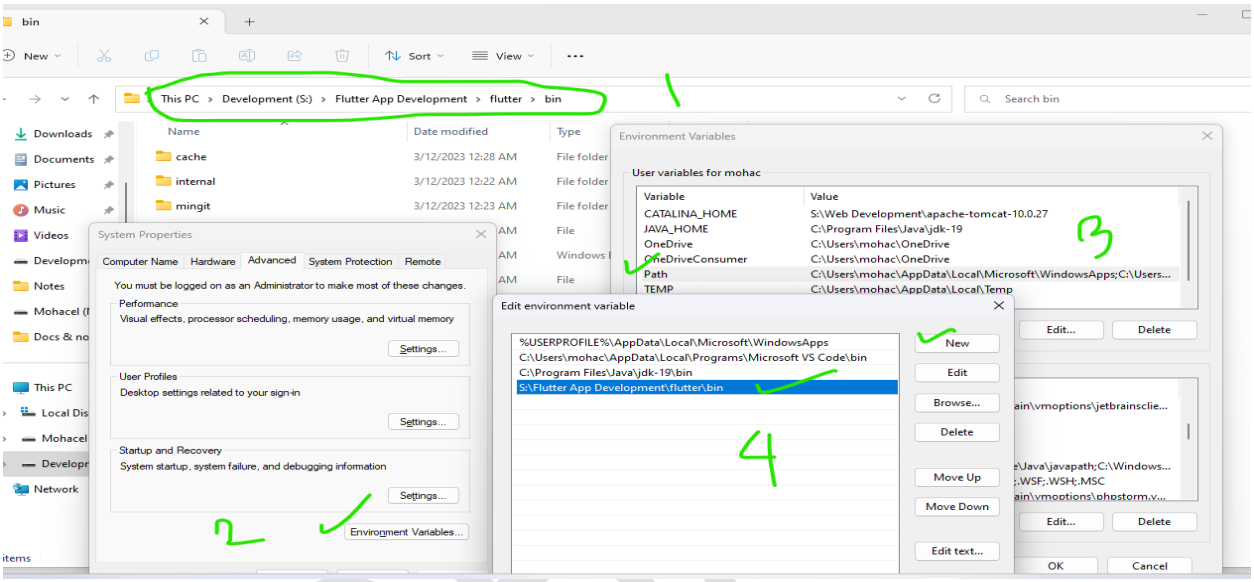
Step-1: First Download and Install Flutter SDK and Android SDK

[Install | Flutter]

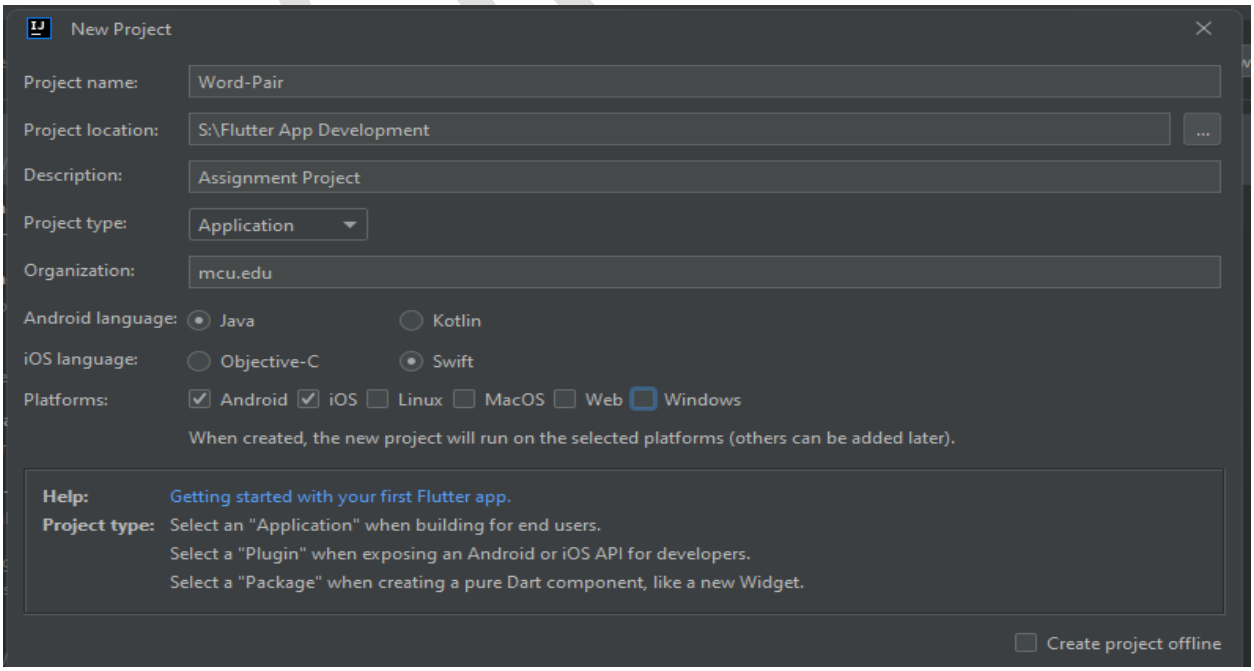
[Download Android Studio & App Tools - Android Developers]

[for light weight you can you]

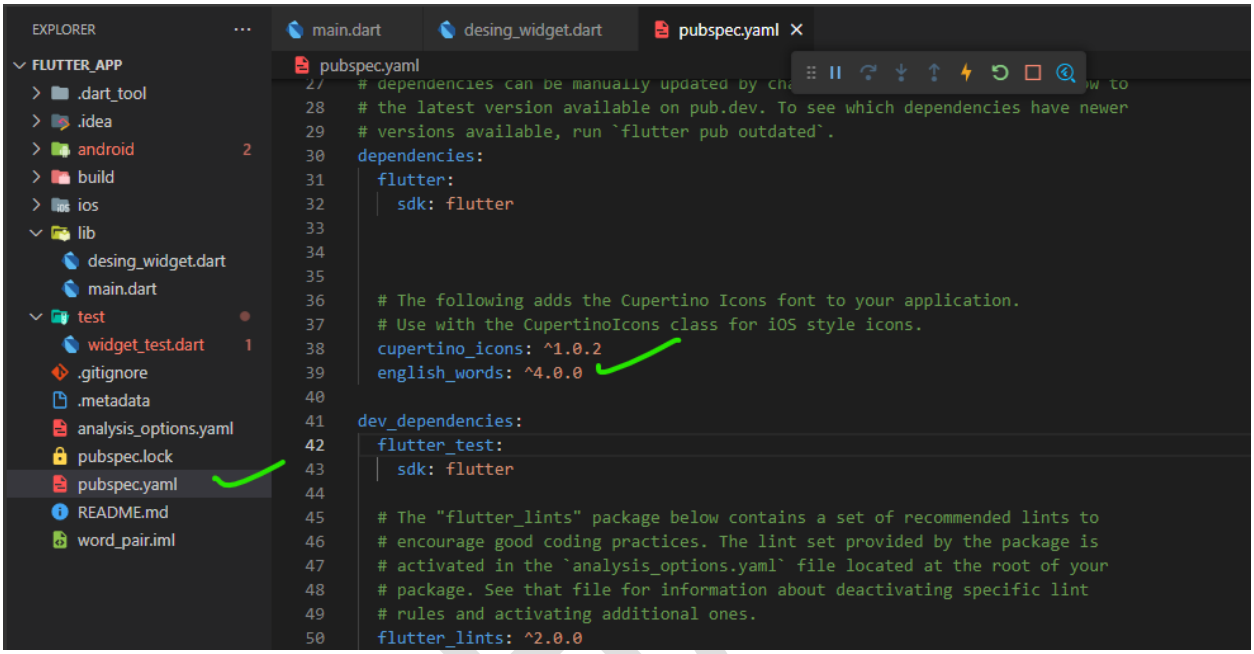
Step-2: Set Environment Variable of your Flutter SDK bin folder.



Step-3: Create a Flutter Project using command line or IDE.



Step-4: For word pair generator import "english_words: ^4.0.0" inside the "pubspec.yaml" file. Or open cmd and run this code "dart pub add english_words" it will automatically add the dependency .



Step-4: For the root or main we are going to use 'MaterialApp' widget.

The **MaterialApp** widget is used to implement Material Design in a Flutter app, and it provides several pre-built widgets and styles to create a modern and sleek user interface.

In this code, the **MaterialApp** widget is configured with a theme that sets the primary color to a shade of purple. The home widget for the app is a custom widget called **RandomWords**, which is not included in the snippet.

The **RandomWords** widget is most likely a custom widget that generates a list of random words and displays them on the screen. The **build** method of the **mcuAPP** class returns an instance of the **MaterialApp** widget with the specified theme and home widget.

Overall, this code sets up the basic structure for a Flutter app with Material Design elements and a custom home widget.

main.dart desing_widget.dart ×

lib > desing_widget.dart > RandomWords

```
1  import 'package:flutter/material.dart';
2  import 'package:english_words/english_words.dart';
3
4  class RandomWords extends StatefulWidget {
5    @override
6    RandomWordsState createState() => RandomWordsState();
7  }
8
```

It defines a class called **RandomWords** that extends the **StatefulWidget** class. **StatefulWidget** is a widget that has mutable state, which means its properties can change during the lifetime of the widget.

The **RandomWords** class overrides the **createState** method, which returns an instance of **RandomWordsState**. **RandomWordsState** is a separate class that extends the **State** class and defines the mutable state for the **RandomWords** widget.

```
Widget _buildList() {
  return ListView.builder(
    padding: const EdgeInsets.all(16.0),
    itemBuilder: (context, item) {
      if (item.isOdd) return Divider();

      final index = item ~/ 2;

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

      return _buildRow(_randomWordPairs[index]);
    },
  ); // ListView.builder
}
```

It defines a private method **_buildList()** that returns a **ListView.builder** widget. **ListView.builder** is a widget that creates a scrollable, linear array of widgets based on the data that is provided to it.

The **itemBuilder** parameter is a callback function that is called for each item in the list. It takes two arguments: **context**, which is the build context for the widget, and **item**, which is the index of the current item in the list.

The function first checks if **item** is odd, and if so, it returns a **Divider()** widget. This creates a visual separation between the items in the list.

If **item** is even, the function calculates the **index** of the item in the **_randomWordPairs** list by dividing **item** by 2 using integer division (the **~/** operator).

If **index** is greater than or equal to the length of **_randomWordPairs**, the function generates 10 new random word pairs using the **generateWordPairs()** function (not shown in this code snippet) and adds them to the **_randomWordPairs** list using the **addAll()** method.

Finally, the function returns the result of calling another private method **_buildRow()** with **_randomWordPairs[index]** as an argument. The **_buildRow()** method is presumably defined elsewhere in the code and builds the UI for a single row in the list.

```
Widget _buildRow(WordPair pair) {
  final alreadySaved = _savedWordPairs.contains(pair);

  return ListTile(
    title: Text(pair.asPascalCase, style: TextStyle(fontSize: 18.0)),
    trailing: Icon(alreadySaved ? Icons.favorite : Icons.favorite_border,
        color: alreadySaved ? Colors.red : null), // Icon
    onTap: () {
      setState(() {
        if (alreadySaved) {
          _savedWordPairs.remove(pair);
        } else {
          _savedWordPairs.add(pair);
        }
      });
    }); // ListTile
}
```

It defines a private method **_buildRow(WordPair pair)** that takes a **WordPair** object as an argument and returns a **ListTile** widget that displays the word pair and a favorite icon.

The **alreadySaved** variable is a boolean value that indicates whether the current **WordPair** is already saved in the **_savedWordPairs** list. **_savedWordPairs** is presumably a class-level set of **WordPair** objects that have been favorited by the user.

The **ListTile** widget has a **title** property that displays the word pair using the **asPascalCase** method, which formats the words in PascalCase (where each word is capitalized and there are no spaces).

The **trailing** property is an **Icon** widget that displays either a filled red heart (if the **WordPair** is already saved) or an outlined heart (if it is not saved). The color of the heart icon is set to red if the **WordPair** is already saved, or null otherwise.

The **onTap** property is a callback function that is called when the user taps the **ListTile** widget. It toggles the value of **alreadySaved** by adding or removing the **WordPair** from the **_savedWordPairs** set. Finally, it calls **setState()** to update the UI and rebuild the widget tree with the new saved **WordPair**.

```
void _pushSaved() {
  Navigator.of(context)
      .push(MaterialPageRoute(builder: (BuildContext context) {
    final Iterable<ListTile> tiles = _savedWordPairs.map((WordPair pair) {
      return ListTile(
        title: Text(pair.asPascalCase, style: TextStyle(fontSize: 16.0)));
    });

    final List<Widget> divided =
        ListTile.divideTiles(context: context, tiles: tiles).toList();

    return Scaffold(
        appBar: AppBar(title: Text('Saved WordPairs')),
        body: ListView(children: divided)); // Scaffold
  })); // MaterialPageRoute
}
```

It defines a private method **_pushSaved()** that navigates to a new screen showing the list of saved word pairs. It uses the **Navigator** widget to manage the navigation stack and push the new screen onto the stack.

The **Navigator** widget is a widget that manages a stack of pages and transitions between them. It provides methods for pushing, popping, and replacing pages on the stack.

The **MaterialPageRoute** is a built-in Flutter widget that defines a standard transition animation between two screens in a material design app.

The **builder** property of **MaterialPageRoute** takes a callback function that builds the new screen. In this case, it builds a **Scaffold** widget with an **AppBar** and a **ListView** widget.

The **Iterable<ListTile>** variable **tiles** is created by mapping the **_savedWordPairs** set to a list of **ListTile** widgets. Each **ListTile** widget displays a **WordPair** using the **asPascalCase** method.

The **ListTile.divideTiles** method is used to create dividers between each **ListTile** widget. The **context** property is set to the build context of the current widget, and the **tiles** property is set to the **Iterable<ListTile>** created earlier.

Finally, the method returns the **Scaffold** widget with an **AppBar** widget that has a title of 'Saved WordPairs', and a **ListView** widget that displays the list of saved **WordPair** objects. When the user taps the back button in the app bar, the **Navigator** widget automatically pops the current screen off the navigation stack and returns to the previous screen.

```
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text('WordPair Generator'),
        actions: <Widget>[
          IconButton(icon: Icon(Icons.list), onPressed: _pushSaved)
        ], // <Widget>[]
      ), // AppBar
      body: _buildList()); // Scaffold
  }
}
```

It defines the **build** method of the **RandomWordsState** class. This method returns a **Scaffold** widget, which is a built-in Flutter widget that provides a basic framework for creating material design apps.

The **Scaffold** widget has an **appBar** property, which is set to an **AppBar** widget with a title of 'WordPair Generator'. The **AppBar** widget also has an **actions** property, which is set to a list containing a single **IconButton** widget.

The **IconButton** widget has an icon of a list and an **onPressed** property set to the **_pushSaved** method. When the user taps the list icon, it calls the **_pushSaved** method and navigates to a new screen showing the list of saved word pairs.

The **Scaffold** widget also has a **body** property, which is set to the **_buildList** method. The **_buildList** method returns a **ListView.builder** widget that displays a list of randomly generated word pairs.

Overall, this **build** method returns a **Scaffold** widget with an **AppBar**, a list of word pairs generated by **_buildList**, and an action icon that navigates to a screen showing the list of saved word pairs when tapped.