**Dicee**

Flutter Preview:



Code Explanation (step by step):

1. Setup Flutter Projects -> go to main.dart files -> delete all values
2. Set up the main Function

void main() {

  runApp(

    const MaterialApp(

      home: Screen(),

    ),

  );

}

1. Setup a screen.dart file to store a stateless widget ‘Screen’ because the screen will stay constant/ never changes.
2. Use SafeArea as the primary widget content so that the content inside would not overlap the screen-border. Then since the colors is equally split into 7 parts vertically, we use column and **Expanded** widget.

A computer screen with white text

Description automatically generated

1. To simplify the code, make another class widget named ‘Piano’ that extends a stateful widget since we will use ‘async’ and ‘await’ to specify a future widget.
2. Based on step ‘d’, the Piano widget will require a number which will specify what sound should be played, and a color to colorize the container.

**How use AudioPlayer Extension by Flutter**

1. Go to https://pub.dev/packages/audioplayers and in the installing section, copy

flutter pub add audioplayers

and paste it in the VSCode terminal

1. Add the .mp3 or .wav audio format into the VSCode Project (It is recommended to not put an folder named ‘assets’/’asset’, try to name it for instance ‘audio’), then in the pubspec.yaml file, uncomment the assets and paste the location inside the assets.

assets:

    - assets/note1.wav

    - assets/note2.wav

    - assets/note3.wav

    - assets/note4.wav

    - assets/note5.wav

    - assets/note6.wav

    - assets/note7.wav

A screenshot of a computer

Description automatically generated

1. Use the import below to be able to use audioplayer class.

import 'package: audioplayers/audioplayers.dart';

1. In the documentation for audioplayers extension, to start an audio, use this format.

Future \_play() async {

    final player = AudioPlayer();

    await player.play(AssetSource('note${widget.number}.wav'));

  }

1. It is a future widget that is used to create Flutter widgets based on the results of an underlying future. Such as when the audio is pressed, we do not want any sort of distraction happening, so the code will work again after the sound is done playing.

async and await is a must in a future widget, await means it will wait player.play to play until it is finished, then return something ‘it will not return anything if not specified’.

1. Use gesture detector inside a container and use the ontap parameter to call \_play() based on the container pressed.

GitHub Link: