UI related widgets are in Blue

Functionality / Main related widgets are in Green

Important concepts / very very important things are in Orange

**Section 3**

* **MaterialApp()**
  + home: Widget()
* **Center()**
  + child: Widget()
  + backgroundColor: Colors.\_
* **Colors.\_**
  + enter name of the color in place of \_. To add accent, add [int]
* **Text()**
  + ‘String’
  + style: TextStyle()
    - fontSize: double
    - color: Colors.\_
* **Scaffold()**
  + appBar: AppBar()
  + body: Widget()
* **AppBar()**
  + title: Text()
  + backgroundColor: Colors.\_
* **Image()**
  + image: NetworkImage(‘URL’)
  + image: AssetImage(‘images/image.png’)
  + Image.\_
    - asset(‘image path’)
* Wrapping a widget around another widget
* To add an AssetImage, you need to change the pubspec.yaml file, create the directory and put the image inside the folder. Indentation is very important in the pubspec.yaml file.
* How to add icons to the flutter app: Choose your image, go to appicon.co and upload it there. Choose the platforms for which you need the image. Downlaod the zip files. These are the assets which you need to move from the downloads folder to the app folder.
  + Android: app -> src -> main -> res, replace the files
  + iOS: Runner -> Assets.xassets replace the file

**Section 6**

* Hot reload. You need a statelesswidget with a build function. The simulator will look for the nearest build and execute it. Unless you have a build function, hot reload will not work.
* In the main, let the runApp(MyApp) and define a new statelss widget class whose name is MyApp. Change the return of this class.
* Container takes all available space available or child.
* Container()
  + color: Colors.\_
  + child: Widget()
  + height: double
  + width: double
  + margin: EdgeInsets.\_
  + padding: EdgeInsets.\_
* SafeArea()
  + child: Widget()
* EdgeInsets.\_
  + all(double)
  + symmetric(vertical: double, horizontal: double)
  + fromLTRB(\_, \_, \_, \_)
* Column() or Row()
  + children: [Widget() list]
  + verticalDirection: VerticalDirection.\_ (up or down)
  + mainAxisAlignment: MainAxisAlignment.\_
  + crossAxisAlignment: CrossAxisAlignment.\_
* MainAxisAlignment.\_ or CrossAxisAlignment.\_ (This will align the lines of the containers. The end will align the right lines of all the containers.)
  + end
  + center
  + spaceEvenly
  + start
  + stretch
* You can set width: double.infinity to stretch it as much as possible.
* SizedBox() (An empty box to give space between two containers
  + height:
  + width:
* Checkout Flutter Layout cheatsheet. Medium article.
* CircleAvatar()
  + radius:
  + backgroundImage: Imagewidget()
* Incorporating custom fonts: SKIPPED for now.
* Add material icons instead of “images”. No pixelated issue for icons. Use the icon library. Use Icon() class
* Icon()
  + Icons.\_ (add the name of the icon)
  + size: double
* Use card class widget to get nice looking, rounded edges buttons and all. Wrap the row / column / container widget in the card widget.
* Card()
  + color: Colors.\_
  + margin: EdgeInsets.\_
  + child: Widget()
  + #No padding class. To set padding for a card, wrap the card around a padding class
* Padding()
  + padding: EdgeInsets.\_
  + child:()
* ListTile widgets skipped for now.

**Section 7**

* Expanded() – Fit the container / child perfectly in any orientation / size.
  + child: Widget()
* FlatButtom() – Always add this in teh stateful widget.
  + child: Widget()
  + onPressed: () { setState() {} } its a void function which changes the state.
* How does the setState(){} work: First, put this in a stateful widget class. Stateless widget is immutable. When pressed, the onPressed function is called. Which will run the setState function. setState will change first do whatever is said in the function. It will then build from the nearest build function. Lets say we need to update the value of int a to 10 when onPressed. Define the variable a BEFORE THE BUILD. In the setState, put a = 10; when pressed, the setState will update the value of a to 10 and build again from the nearest build. If a is defined under the build, the value will not be changed.
* random -> import ‘dart:math’; then use Random()nextInt... (max number). This will randomly choose a number from 0, max-1

**Section 9**

* Import packages.
* Most of the “todo” is given in the documentation

**Section 10**

* How did you make the scoreChecker using
* Android VCS
* This section is more focused on the Dart concepts. List, objects. I am quite aware of these. And will not be needing that much of complexity in the Beacon app. I will come back to this as and when required.

**Section 11**

* Same as above

**Section 12**

* This module focuses on the UI advanced. Not much of functionality.
* Theme() – You can modify the theme. I will use the dark theme. (Use this theme that they are making)