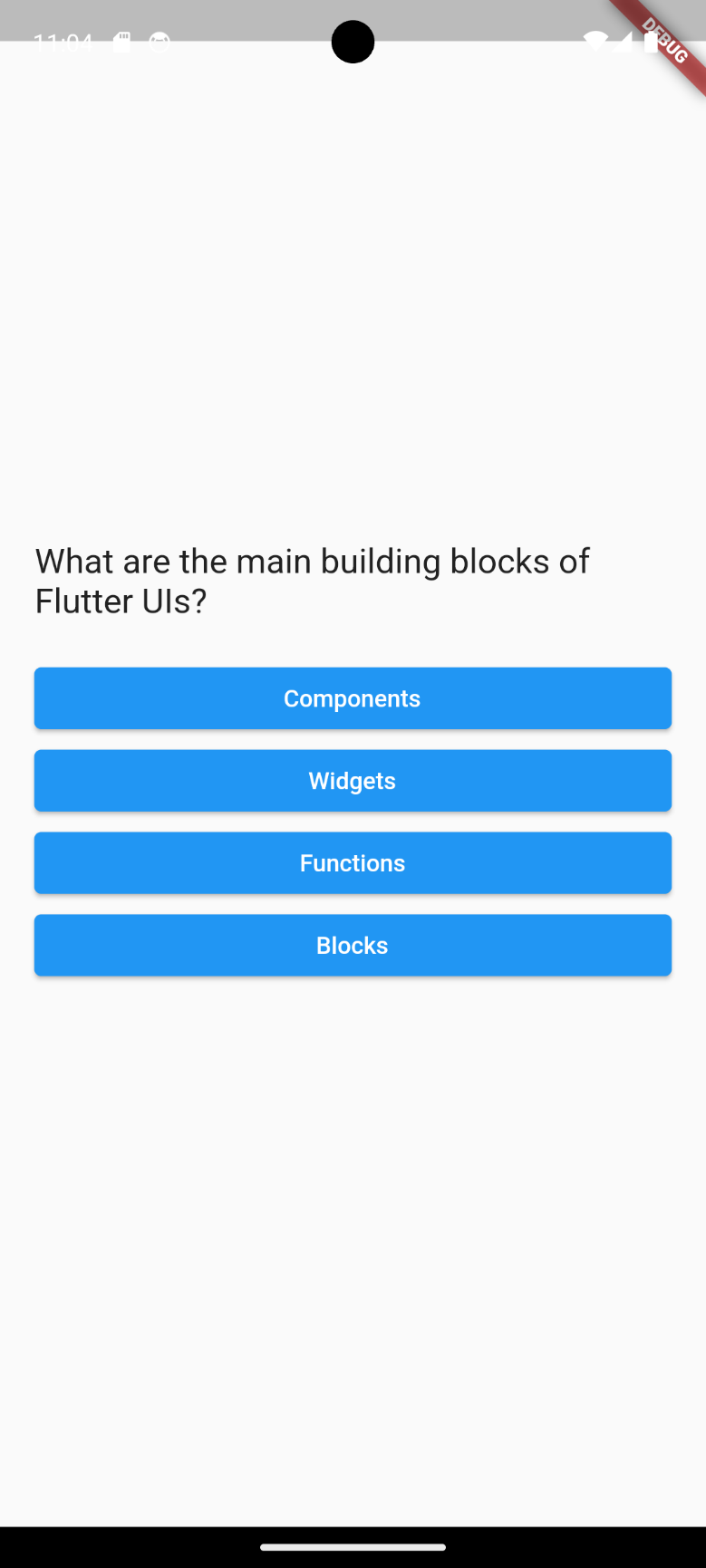
**Quizz**

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

import 'package:flutter/material.dart';

import 'package:my\_app/question\_screen.dart';

void main() {

  runApp(

    const MaterialApp(

      home: Scaffold(

        body: QuestionScreen(),

      ),

    ),

  );

}

1. Setup a question\_screen.dart file that stores the frame of the application. There will be no navigation to other screen (that use a **navigation.push** function) but it will be one stateful widget with a variable than stores a widget class that can be used to change the current widget.
2. There will be 3 widgets. **start\_widget, questions\_widget, and result\_widget.** start\_widget and result\_widget will have a stateless widget since no rendering needed. But question\_widget will have a widget component that renders each question one after another.
3. Start form the start\_widget, use a stateless widget and name it StartWidget. Here, it will display a centered text and a textbutton widgets, where when the button is pressed, it will render the screen to the question\_screen. Now, how to we change the state of the UI if start\_screen is not a stateful widget? Here question\_screen.dart will be the anchor to all the 3 widgets mentioned in **d).**
4. In the question\_screen.dart file, make a variable that has a ‘widget’ type, let’s name it ‘currentWidget’. Inside the \_questionScreenState class, make a void function that sets the currentWidget to a StartWidget(), lets name the void function goToStartWidget(). So that in the build method, we will return the currentWidget which is a start\_widget.

A screenshot of a cell phone

Description automatically generated

1. Then, how to connect the textbutton onTap function to the question\_screen.dart file, when both function was in a different file? Use a constructor function to pass a value **or a function**. To get and run the function of other class. In the start\_widget.dart file, make a constructor that needs a variable with a format of:

const StartWidget(this.startQuiz, {super.key});

  final void Function() startQuiz;

where the class will required a function passed from other file or else the class will not run/code red. State the onTap text button to startQuiz. Then in the question\_screen.dart file, make a function that will go to the questionsWidget. It means that the UI needs to be rendered, so use setstate and change the currentWidget to QuestionsWidget, lets name the function goToQuestionWidget(). Then in the goToStartWidget, the startWidget now requires a function that need to be passed, hence use the goToQuestionWidget function.

void goToStartWidget() {

    setState(() {

      currentWidget = StartWidget(goToQuestionWidget);

    });

  }

Now, when the button is pressed, it will run goToQuestionWidget Funtcion even if it is in a different file. And since it is in a set state function, the UI will render the currentWidget to QuestionsWidget

1. To be able to show the questions, we must make the questions first. So first, make a model file named quiz\_question.dart where it will consist of a class, and what required value that is needed to be able to store the questions. Here, we will need a question with a String data type and a **List<String>** answer. **List<String>** works like an array in other programming language where it will store an arrayed string, it means that there will be a multiple choises answer. Then, to shuffle the answer, use .shuffle method like this format:

List<String> getShuffledAnswers() {

    final shuffledList = List.of(answer);

    shuffledList.shuffle();

    return shuffledList;

  }

1. Make a new file named question.dart where it will store the question and the answers. It must be a const -> variable name -> a square bracket where inside stores a string, and another square bracket that stores the answer. We can store more than 4 answers and more than 2 questions since the code is dynamic. And to make it easier, the answer for the question will always be in the 1st index of the List of answers.

const questions = [

  QuizQuestion(

    'What are the main building blocks of Flutter UIs?',

    [

      'Widgets',

      'Components',

      'Blocks',

      'Functions',

    ],

  ),

  QuizQuestion('How are Flutter UIs built?', [

    'By combining widgets in code',

    'By combining widgets in a visual editor',

    'By defining widgets in config files',

    'By using XCode for iOS and Android Studio for Android',

  ],

),

1. In the questions\_widget.dart, make a centered widget that will show a question text, and the answers below it. For now, make the hard code first, then we will change it dynamicly. Make 2 variable outside the build method which is a ‘**final List<String> answers = []’** where it will stores the answers chosen, and a ‘**var currentQuestionIndex = 0**’ that stores the question index, this will be responsible to show which questions that is shown in the screen.

Note: a List<String> is the same as [‘apple’, ‘banana’, ‘orange’], where the index will start from 0-2.

final List<String> answers = [];

  var currentQuestionIndex = 0;

1. Import the questions.dart file, then inside the build method, make a ‘**var currentQuestion = questions[currentQuestionIndex]**’, where currentQuestion will store a QuizQuestion class type made from quiz\_question.dart file.

 Widget build(BuildContext context) {

    var currentQuestion = questions[currentQuestionIndex];

1. In the question text, we can change the hard code text to a dynamic one by changing it to **currentQuestion.question** which will show the question string from the questions.dart data.

Text(

            currentQuestion.question,

            style: const TextStyle(fontSize: 20),

          ),

1. To show all of the answers, we can use a ListView.builder or a map. Remember that we store the answer to the question on the first index and we need to shuffle the answers randomly, which means the answers now will not be on the first index. How to solve this? The solution is using map and spread method, where map makes a copy of the list and modify the List without changing the original list. To make a map, we use a ‘.map’, ‘.getShuffledAnswers()’ to make a map and shuffle the answers. But when we implement it, it is still in red line/ error occurred. It is because we need to spread the map using a ‘**…**’ operator in front of the currentQuestion.

Note: map of a list = [[a, b, c, d]], we cant shuffle it.

‘**…**’ + map of a list = [a, b, c, d], we can now shuffle it.

...currentQuestion.getShuffledAnswers().map((answer) {

            return SizedBox(

              width: double.infinity,

              child: ElevatedButton(

                onPressed: () {

                  answerQuestion();

                  answers.add(answer);

                },

                child: Text(answer),

              ),

            );

          }),

1. On the code above, it will show the answer buttons based on the amount of the answers made in the question.dart data. Now, how do we store the user’s answer and move to the next question every time the button is clicked? We need to re-render the widget and change the index to change the question and answers in the UI by using a void function, and use .add method the store the answer in the answers list<string>.
2. We can also pass the user’s answer by using similar method. Using a constructor to be able to use a function from another class in another file and run it in that file. Here, the questionWidget requires a void function with a List<String> parameters to be able to accsess the class.

const QuestionWidget({super.key, required this.quizResult});

  final void Function(List<String>) quizResult;

1. In the question\_screen.dart file, make a function that requires a parameter of List<String> and a void function named goToResultWidget(List<String> user’s answer). That stores the user’s answer and pass the answers to the results\_widget.

void goToResultWidget(List<String> answer) {

    setState(() {

      currentWidget = ResultWidget(

        answers: answer,

        startWidget: goToStartWidget,

      );

    });

  }

startWidget and answers here it made from the constructor of the results\_widget. Challenge: ‘try to find out how to make constructor for answers and a startWidget function

1. In the answerQuestion void, which will change the question once user input is clicked, use a setState method to add the index of the question. Now remember, the index cannot be more than the questions available or it will cause an error in the code. It means we need to make a **Conditional Statement** that checks if the index is equals to the total questions, then reset the currentQuestionIndex to 0 and run the quizResult(answer), where the function allows us to run the function in the question\_screen.dart file and pass the list<value> of user’s answer from the question\_widget file by a parameter.

 void answerQuestion() {

      if (currentQuestionIndex == questions.length - 1) {

        currentQuestionIndex = 0;

        widget.quizResult(answers);

      } else {

        setState(() {

          currentQuestionIndex++;

        });

      }

    }

1. In the result\_widget.dart file, which is a stateless element since no UI is rendered and the component is fixed, remember to make a contructor to pass the List<String> of user’s answer and a function to reset the quiz.

const ResultWidget(

      {super.key, required this.answers, required this.startWidget});

  final List<String> answers;

  final void Function() startWidget;

1. To be able to show the user’s answer, use a **ListView.builder** since mapping the list is not required. A Listview.builder will need in itemCount, and itemBuilder which stores a context and an index where to show the answers, we just need a Text widget and declare the text as answers[index].

ListView.builder(

              itemCount: answers.length,

              itemBuilder: (context, index) {

                Color trueFalseColor;

                if (questions[index].answer[0] == answers[index]) {

                  trueFalseColor = Colors.blue;

                } else {

                  trueFalseColor = Colors.red;

                }

                return Text(

                  answers[index],

                  textAlign: TextAlign.left,

                  style: TextStyle(fontSize: 18, color: trueFalseColor),

                );

              },

            ),

To be able to change the color based on the correct/ false statement, we can use a conditional statement like above.