W7-S1 *PRACTICE*

QUIZ PROJECT STARTER

This practice serves as a foundation for building **a simple single choice quiz app** and will prepare you for your **upcoming Quiz Micro Project**.

In your Quiz Micro Project, you'll extend this basic quiz app by adding advanced features*, such as data persistence, multi players, quiz editor, different question types etc.…*

Because it's important to start with a good project structure, your first mission is to complete this starter practice:

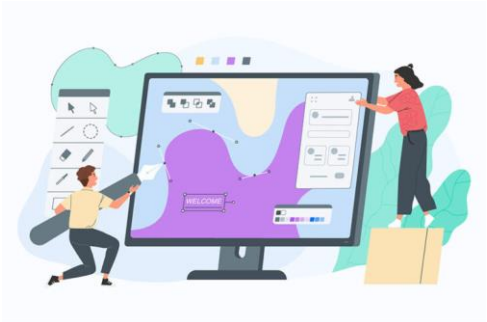
You'll establish **a core structure and essential skills** that will enable you to tackle **more complex features in your project**.

Learning objectives

- ✓ Handle **navigation** between **multiple screens** – *Using a state (not router for now…)*
- ✓ **Pass data** between screens
- ✓ Separate **UI logic** from **business logic**: using a model folder
- ✓ Reflect on the best approaches (***data, states, widgets***) to maintain a clean architecture

How to submit?

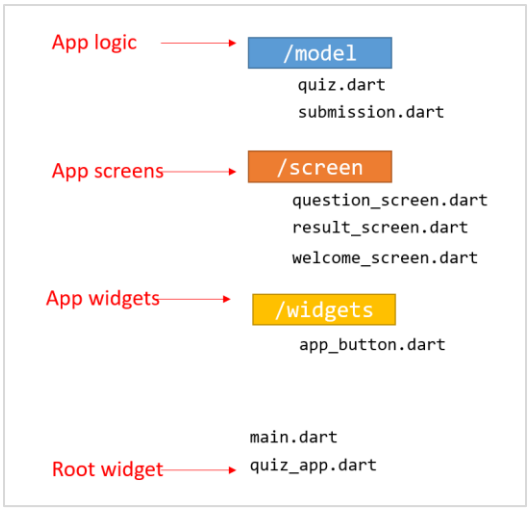
- ✓ **Push** your final code on **your GitHub repository**
- ✓ Then **attach the GitHub path** to the MS Team assignment and **turn it in**



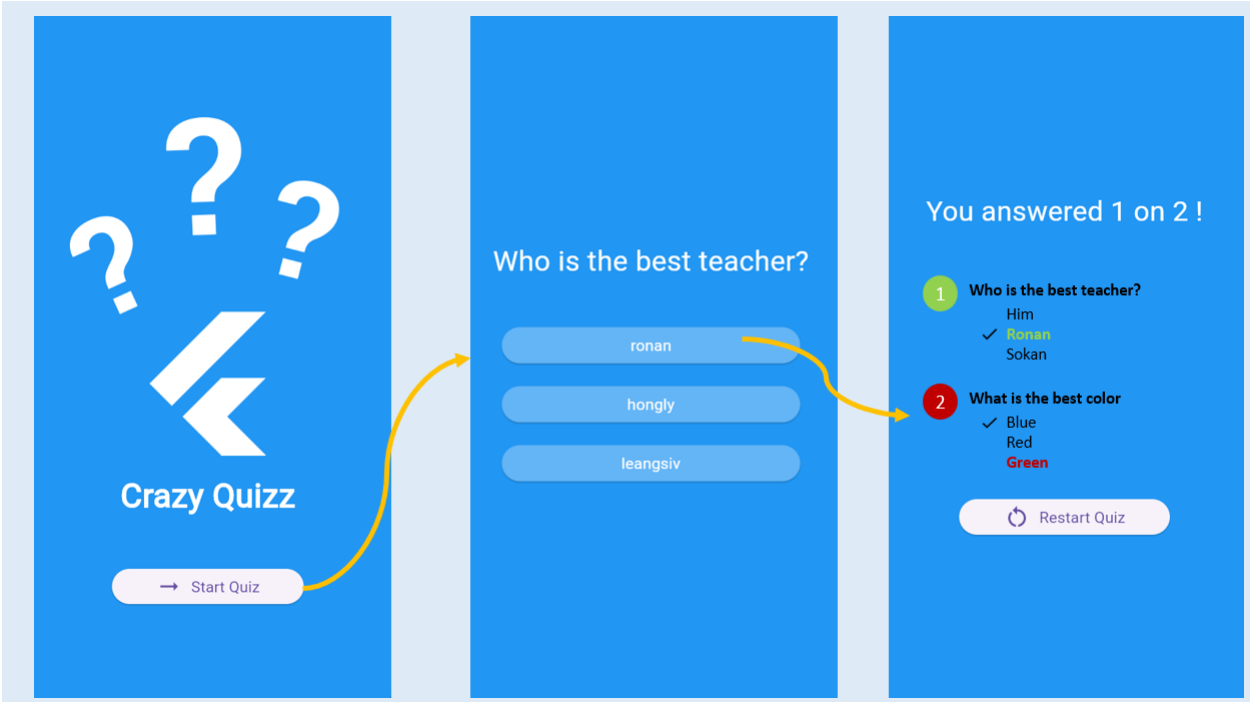
APP OVERVIEW

The application separates classes and widgets into 3 folders:

Model	Contains the app data structure and business logic
Screens	Contains the screens and their sub widgets
Widgets	Contains the widgets re-usable *(widget, form, checkboxes.)*



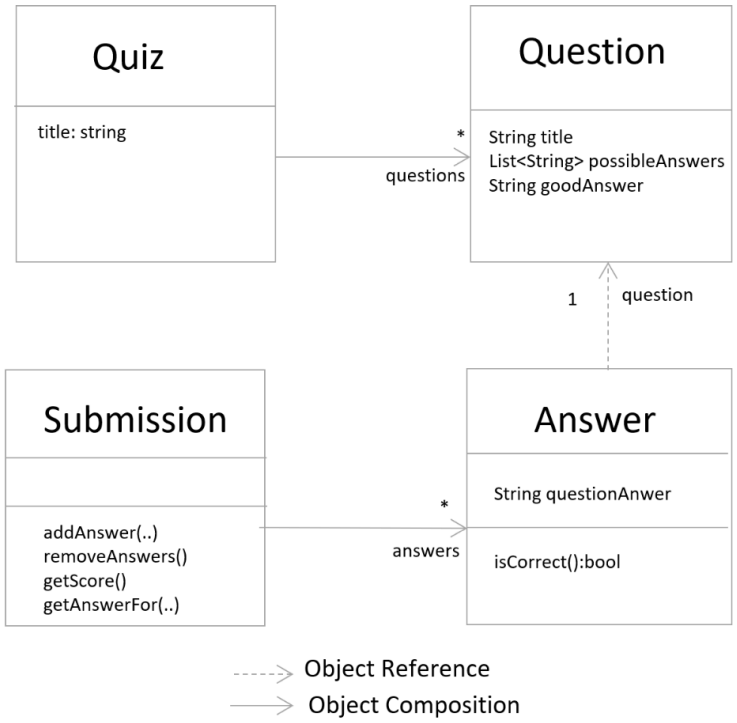
This is a start structure! might be subject to change according to your own project needs



The app just consists of a list of question and a result view

PART 1 – MODEL

The model is dedicated to handle a single choice-based quiz.
The start code already contains the classes **Quiz** and **Question** in `/model` folder.



Q1 – Implement the following classes in `model/submission.dart` file.

- Class **Answer**
  - `bool isCorrect()`
    - *Return true if the answer is correct*

- Class **Submission**
  - `int getScore()`
    - *Calculate the submission total score*
  - `Answer? getAnswerFor(Question question)`
    - *Retrieve the answer related to given question*
    - *Return null if no match*
  - `void addAnswer(Question question, String answer)`
    - *Add or update an answer*
    - *RULE: only 1 answer per question*
  - `void removeAnswers()`
    - Remove all answers.

Q2 – Test the model using a dedicated **test file**

PART 2 – QUIZ APP

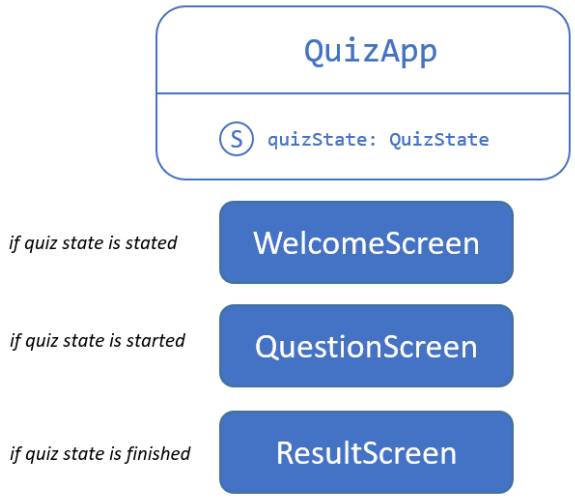
The next step is to create the main widget, and manage the **navigation between the screens**.
The 3 screens (welcome, question, result) are already created in /`screen` folder

Note that for now we don't use yet a Router. Instead, we use a state to manage the screen conditional display.

- ✓ Create the **QuizState** enum composed of 3 values: *not started, started, finished.*
- ✓ Create the **QuizApp** stateful widget.

TYPE	ARGUMENT	STATE
STATEFULL	quiz: **Quiz**	quizState

- ✓ Depending on the quiz state () the QuizApp widget shall display 1 of the 3 screens, as follows:

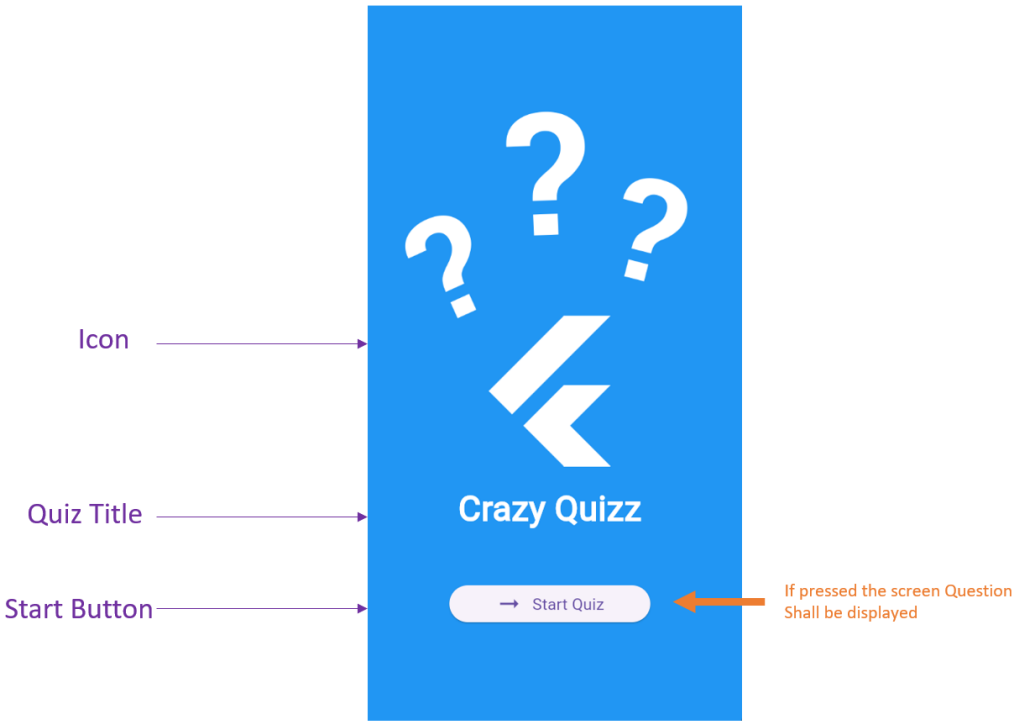


- ✓ Test the app by manually changing the state.

PART 3 – WELCOME SCREEN

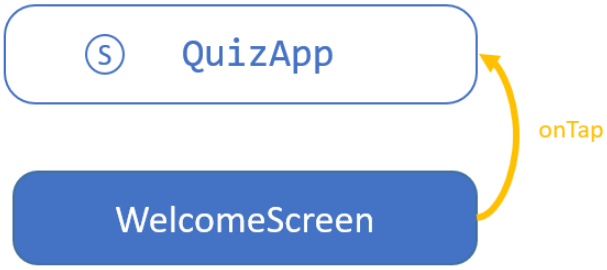
TYPE	ARGUMENT	STATE
STATELESS	- onStart: **Callback** - quizTitle: **String**	

✓ The **Welcome** screen shall be displayed as bellow:



<u>Note</u> The icon image and the app button are provided in the start code

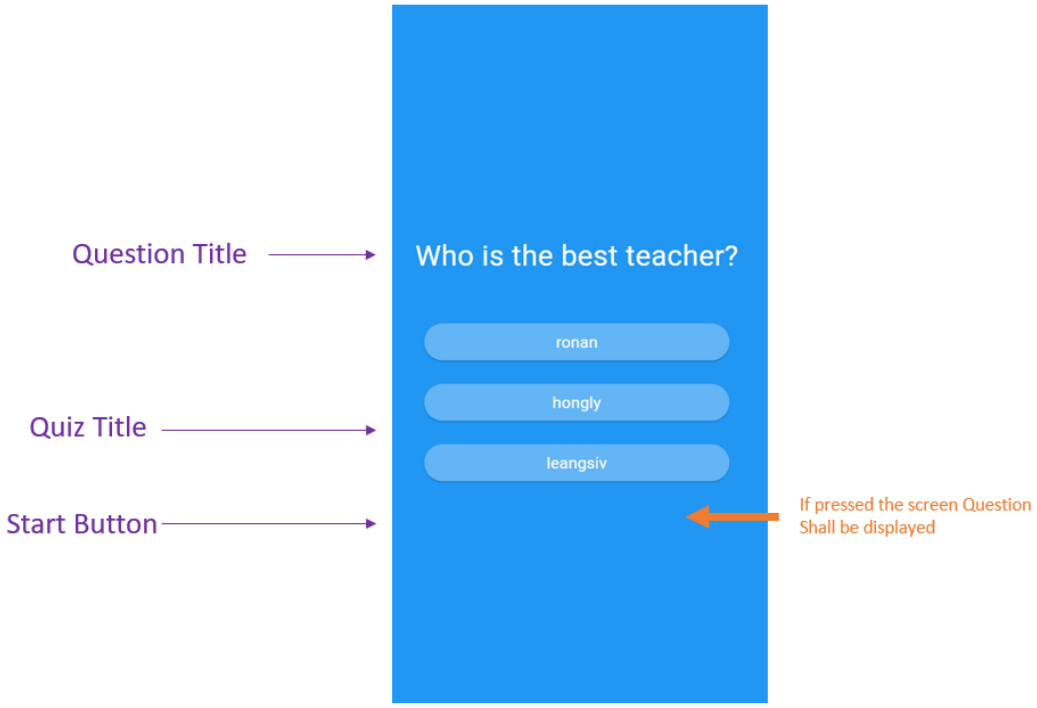
✓ When clicking on the start button, the app should **switch to the question screen.**



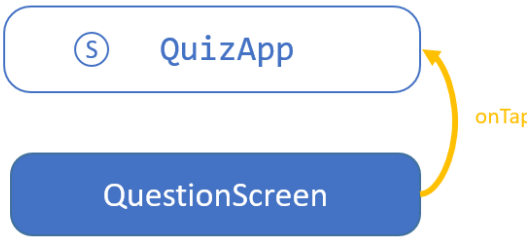
PART 4 – QUESTION SCREEN

TYPE	ARGUMENT	STATE
STATELESS	- onTap: **Callback** - question: **Question**	

✓ The **Question** screen shall be displayed as bellow:



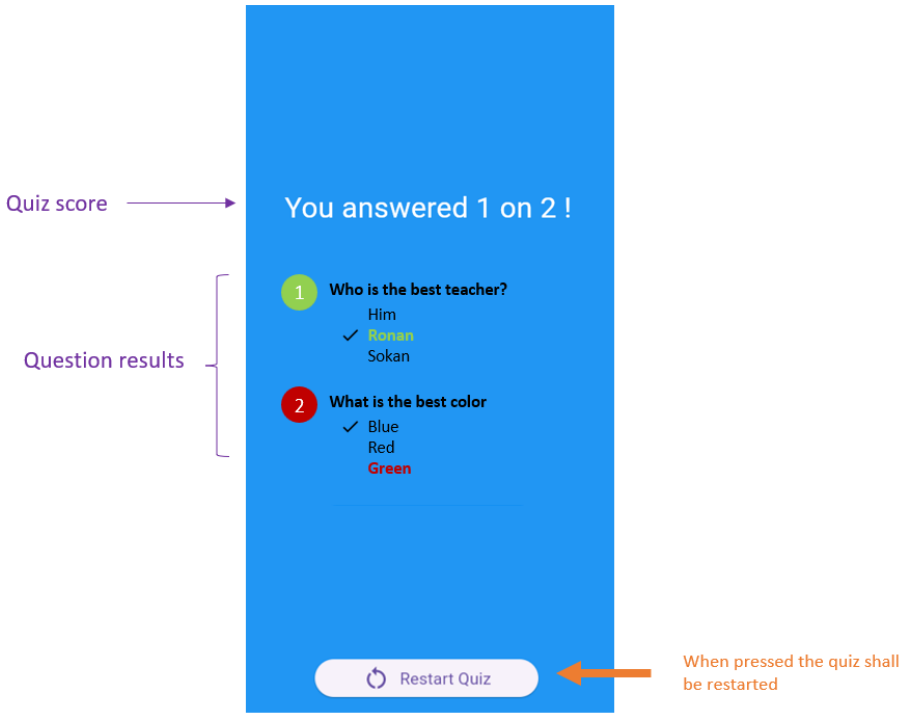
✓ When user click on any choice:
- If the quiz is finished, **go to the result view**
- If the quiz is not finished, **go to the next question**



PART 5 – RESULT SCREEN

TYPE	ARGUMENT	STATE
STATELESS	- onRestart: **Callback** - submission: **Submission** - quiz: **Quiz**	

✓ The **Result** screen shall be displayed as bellow:



✓ When clicking on the Restart button, the app should **restart**
- The quiz state should switch to NOT_STARTED
- All submission answers should be cleared

✓ Tip: you can divide the work into many sub-widgets, as example: