CADT IDT

# ***W9*** *PRACTICE*

# *QUIZ APP*

## Important

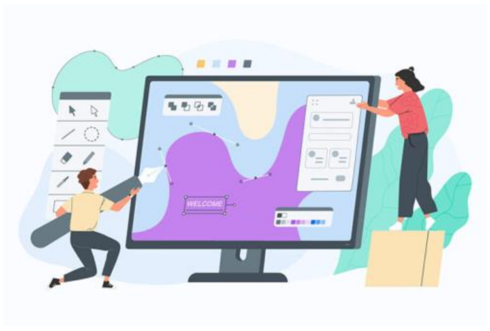
- ✓ The reflection part will be done in **teams of 2** (*designing*) and 4 (*sharing*)
- ✓ The coding part needs to be submitted **individually**

## 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**



## Functional Requirements

### For this practice (W9)

- ✓ The player can **start the quiz** and **answer each question** one by one
- ✓ Only single choice questions
- ✓ Once finished, the app shows the **score and the questions results**

### For Bonus

- ✓ The history of the previous scores can be reviewed
- ✓ The **quiz questions** and **player submission** are persisted in JSON file

### For next practice (W10)

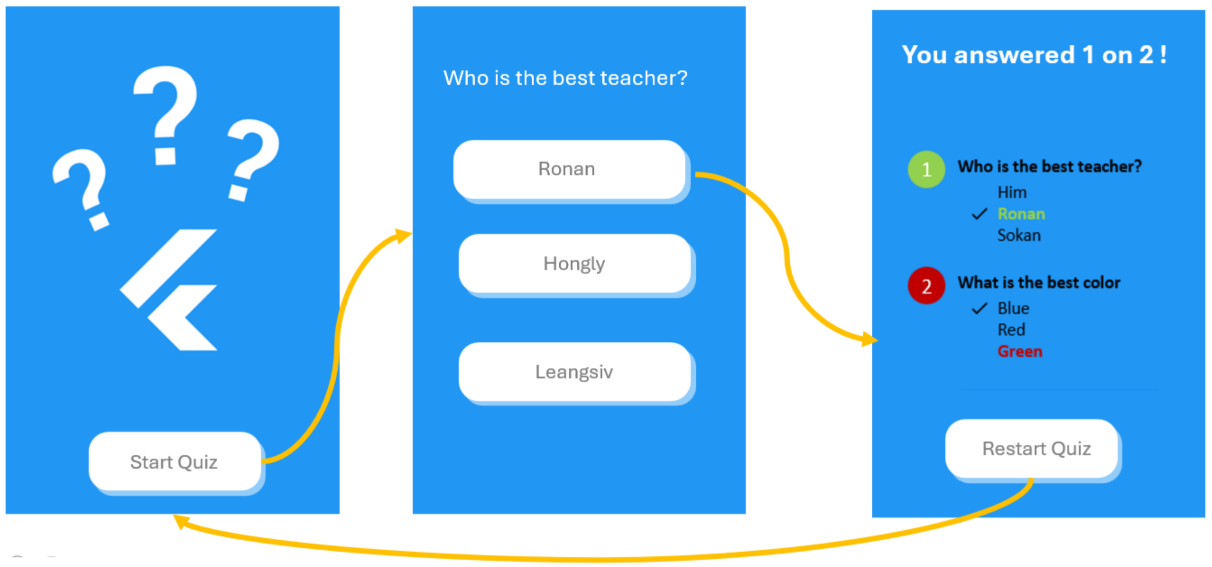
- ✓ The player **enters his/her name** before starting
- ✓ It's possible to **edit the quiz questions**

## Non-Functional Requirements

- ✓ The application must **implement the provided user flow and mockups**

## User Flow

For this practice, the following **user flow**/mockup are required:



*BONUS*

To include the **history of the previous scores**, the **user flow** can evolve as follows:

$$\begin{tikzcd}
\text{Start Screen} \arrow[r] & \text{Question Screen} \arrow[r] & \text{Result Screen} \arrow[r] \arrow[ll, bend left] & \text{History Screen} \arrow[lll, bend left]
\end{tikzcd}$$

## Layer structure

The application is structured around 3 layers: DATA > DOMAIN > UI

| | |
|---|---|
| **data** | Repositories to load **domain objects** from **data sources** |
| **model** | Contain the **domain classes** |
| **ui/screen** | Screen widgets and sub-screen widgets |
| **ui/widgets** | Re usable widgets (button, inputs…) |

Here is an **example** of project structure *(just an example, not the correct one)*

```
lib/
 ├─ data/
 │     └─ repositories/
 │           └─ quiz_json_repository.dart
 │            └─ quiz_mock_repository.dart
 ├─ models/
 │     └─ quiz.dart
 │
 └─ ui/
       ├─ screens/
       │     └─ welcome_screen.dart
       │     └─ question_screen.dart
       └─ widgets/
             └─ app_button.dart
              └─ app_button.dart

     main.dart
```

## Layer interaction

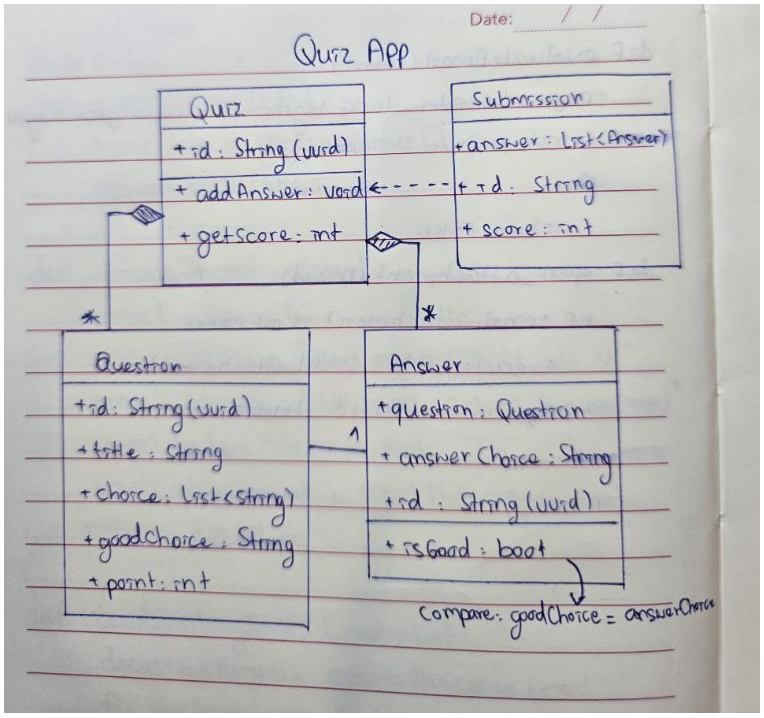
1. The **main** loads the **quiz data** (*from mock data or from a Json file)*
2. The **main** create the **quiz screen**, passing the quiz data as parameter

## PART 1 – REFLECTIONS

### MODEL

To handle the functional requirements for this practice, and be ready for the next practice, how are you going to structure your model?

Q1 – Drop below the **UML diagram of your model**



Q2 – Where do you **keep player submission,** so that you can display the last screen?

### Answer:

{I keep each player's choice inside a List<Answer> answers. Whenever the user picks an option, I just run quiz.answers.add(...) to save it.}

### UI – **Screens**

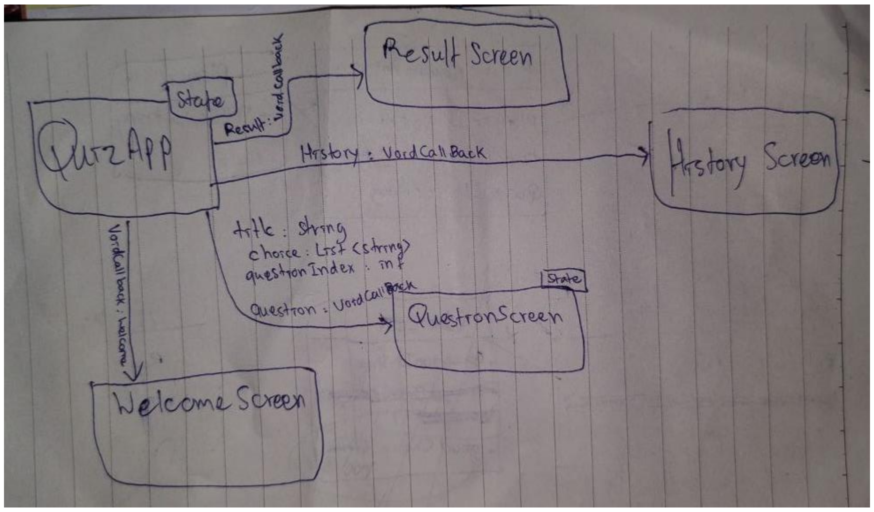
We have 3 screens (start, question and result)

Q3 – Identify for **each widget** their properties

| WIDGET | TYPE (SL / SF) | PARAMETERS | STATES |
|---|---|---|---|

| | | | |
|---|---|---|---|
| welcome | SL | No | No |
| question | SF | Question(title)<br>Question(choice) | Yes |
| result | SL | No | No |
| history | SL | No | No |

Q4 – Draw the **COMPONENT DIAGRAM** of the application



Q5 – Where and How do you **manage the navigation** to the **next questions** and to the **last result screen**?

## Answer:

{ We manage the navigation by checking the index of the questions. If the current question is not the last question in the quiz, we go to the next question, increasing the index by 1. If it is the last question, then we use the callback function from QuizApp and navigate to the result screen.
}

## UI – **Reusable widget**

List down the widget you are **planning to re-use** on different screens (button, card..)

| WIDGET | TYPE (SL / SF) | PARAMETERS | STATES |
|---|---|---|---|
| App_Button | SL | Label, onTap | Yes |

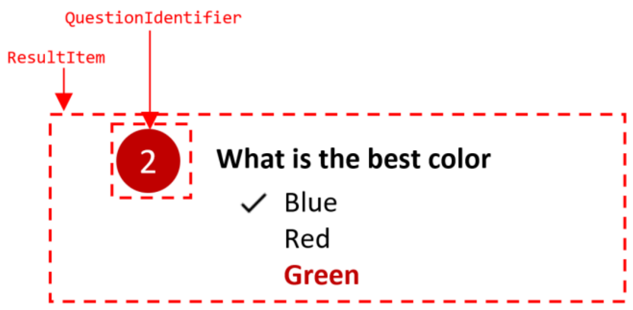
| | | | |
|---|---|---|---|
| | | | |
| | | | |
| | | | |

## *PART 2 – IMPLEMENTATION*

- ✓ The **coding part** needs to be submitted **individually**

*HINTS*

- ✓ Tip: you can divide each screen into many **stateless screen-widgets**, for example:



*This widget takes as parameter a question and a player choice and handle the color computation, the choices highlighting etc..*