Università della Svizzera italiana

Institute of Computing CI

Software Atelier 3 **Fall 2021**

Lovnesh Bhardwaj

First Meeting

Hello, the purpose of this text is to outline what we have discussed during the meeting last night and how we wish to shape the project. As a general intro to the theme, this project would be similar to 'Kahoot' and we have chosen to name it **'USIHoot'**. The following sections define the technicalities involved with the project.

I am not going to define what the project does etc. Rather, this document can be looked to as a blueprint for the more technical part of the project (routes, sync, etc.)

Team Distribution

As you all know, this team has been divided into two with half of us working on the front-end and the back-end each. There would obviously be close communication between the two groups and also assistence to/from the other group whenever required. This is done in order to develop both of the parts of this project in parallel.

Front-end = Alessio, Catarina, and Brian
Back-end = Michele, Lovnesh, Stefano

Object Structure for the views and DB

This was the topic that was heavily discussed during the meeting today. From my understanding, the structure of the objects that would be passed around to/from the backend/DB/Frontend is as follows

```
game = { // Defines a Game Object
    Quiz = { // Quiz object persisted in the DB
        Questions : List<Question>
    },
    Player: List<People>,
    Host: People
}
```

```
Question = {
    q : String, // A question in its string representation
    wrong_answers : List<String>, // Three Wrong answers here
    right_answer : String,
    difficulty : Enum,
    points : { // Depends on the difficulty of the question.. selected by
whoever makes the quiz
        easy = 10,
        medium = 20,
        hard = 30
    }
}

People = {
    username: String
}
```

This above highlights the basic object structure that we discussed during the meeting today.

Routes

After discussing the structure of the objects, we discussed the routes that we would need to have in our application. They are the following:

- **GET /** - Serves the homepage of the application. This homepage has two buttons, one to create a quiz, another to create a game. A game cannot be created without a quiz.
- **GET /quizCreationForm** - Serves the quiz creation form from the homepage. After completing it a **POST** request is send to the backend with the questions, difficulty of each question, and highlighting the correct answer.
- **GET /createGame** - Serves the game creation form for the host of the game, while a person is here the players waiting outside have to be put in a wating room (as discussed in the meeting). After choosing the quiz, it sends a **GET /createGame/:quizID** so that everyone is redirected to the quiz room.
- **GET /question/:id** - Sends the data of the question given its id.

So far we have only discussed these routes, more of them will be added as soon as these basic ones will be completed.