Quiz.io

Distributed Systems – Project Proposal

Peter Mueller, Jakob von Kalle, Jonas Mohler, Basil Fuerer, Valentin Kunze
14-914-717, 14-943-880, 14-916-712, 13-918-487, 14-941-413
pemuelle@student.ethz.ch, jvkalle@student.ethz.ch, jmohler@student.ethz.ch,fuererb@student.ethz.ch,
kunzev@student.ethz.ch

ABSTRACT

In this project we are building a distributed Quiz App for android which makes it easy to create quizzes and allows for a unlimited amount of players to compete with each other.

1. INTRODUCTION

Our main idea is to create a distributed android quiz app. The idea is similar to the Clicker option in the ETH EDU App [1], but should be in a more competitive and Quiz Show like style.

Our main approach is that the app is divided into two parts. One part being the Quiz Creator/Moderator part, which is run on one phone and allows the creation of a quiz and then acting as the main display for the Quiz. This phone should be connected to a beamer so that everyone competing can see the questions and results.

The other part is the client side, where every player can join the quiz and gets then displayed with a simple interface to answer the questions like in a game show.

The main aspect of the app is the gameification of the whole quiz, meaning the players get points for answering the questions (more point for fast right answers). A scoreboard is then displayed on the moderators screen.

The main use for this application would be schools and universities to do interactive quizzes during lectures. But it can of course also be used by friends to simulate quiz shows at home.

After we started the Project we realized that similar solutions already exist [3], but we wanted to implement the idea from scratch and think it's the perfect application to apply everything we learned in the previous projects into one real world use case.

2. SYSTEM OVERVIEW

Figure 1: Welcome Screen

Our app is split into two parts, in the welcome screen (Figure 1) you can either choose if you want to create a quiz or join a quiz which then leads to different UI flows which will be described here in more detail.

If you choose to create a new quiz you get presented with the quiz creation screen (Figure 2) where you are able to set the question for the quiz and edit them. Each question has 4 possible answers and you can add as many question to the quiz as you want. The question will be displayed in a list view and clicking on one or adding an additional one presents a simple form to enter the details. (Figure 2)

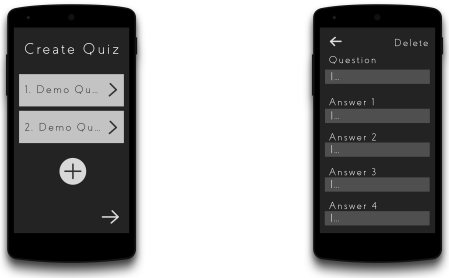

Figure 2: Create Quiz Screens

Figure 3: Join Game Screen

After you are finished with setting up the questions the phone will set up the quiz and assign it a code which can be used to join the game, it now listens for players trying to join the game. From now on the phone should be connected to a beamer so that the participants can see the Quiz Code. The moderator phone now shows the next screen (Figure 3) where at the top the Quiz Code for joining is displayed and below that the names of the players who already joined the game. Once everyone has joined the game the moderator can start the quiz.

Figure 4: Question and Answer/Score screen

Once the moderator starts the quiz the first question is display on the moderators screen (Figure 4) and it sends a signal to the clients that they can now answer the first question. It listens for the incoming responses from the clients. Once the time to answer the question has passed the moderator phone evaluates the answers and assigns points to the different players according to their answering speed, it then presents a screen (Figure 4) showing the right answer and the current Top 3 Players. The moderator then has the option to proceed to the next question. This process continues up to the last question, after which the final scoreboard will be presented. (Figure 5)

Figure 5: Final Scoreboard

The client side of the app will look as follow: After the user select to join a Quiz he gets presented a form (Figure 6) to enter his name and the Quiz Code which is displayed on the moderators display. Once these details are entered the client will try to connect to the server and if a game is found a waiting screen (Figure 6) is presented until the moderator starts the quiz.

Figure 6: Join Quiz and Waiting Screen

While the questions are displayed on the moderator screen, the client phone shows a simple screen with the remaining time to answer the question on top and 4 buttons matching the colors of the answers. (Figure 7) The players can now lock in an answer which sends a request to the server (moderator phone). Once the time has passed to answer the question another screen is presented which shows the player if they were correct and their current score and rank. This process repeats up to the last question.

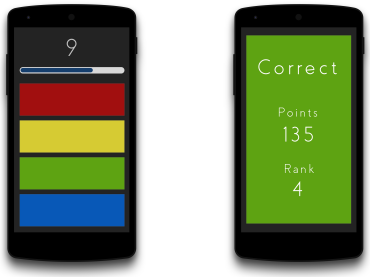

Figure 7: Select Answer and Result Screen

3. REQUIREMENTS

Internet connection, Android phones with the quiz app, one android phone connected to a beamer as the moderator phone. All phones should be connected to the same WiFi Network. Besides Gson [2] for parsing JSON data we're not planning to use any other external libraries.

4. WORK PACKAGES

The project can be broken down in to the following Packages:

- **WP1**: Designing the User Interface
- **WP2**: Set up the Rest API Calls (first with dummy responses so that the client and server side can be worked on at the same time)
- **WP3**: Set up the Quiz Creation (Creating Question Screen)
- **WP4**: Set up the Quiz Connection Mode (Creating the join code and listen for joining Players)
- **WP5**: Set up the Quiz Question Screen (Displaying the question and handle the answer signals from the clients)
- **WP6**: Display the scoreboard (Award points to the players according to their performance and creating the scoreboard)
- **WP7**: Set up the logic to join a game (POST Call)
- **WP8**: Implement the different client screens (Select Answer, Correct Answer, Final Scoreboard)
- **WP9**: Listen to the events from the server and present the according client screen
- **WP10**: Set up the clock sync for consistency in scoring

5. MILESTONES

The main project distribution will be Peter,Valentin,Jakob will do Server Side and Basil,Jonas the Client Side

Peter **WP1**,**WP2**
Valentin **WP3**,**WP4**
Jakob **WP5**,**WP6**
Basil **WP7**,**WP8**
Jonas **WP9**,**WP10**

Schedule:

- **17. November:** Hand in Proposal
- **19. November:** Define API Calls for Client Server Communication
- **21. November:** User Interface completely set up
- **24. November:** Quiz Creation and Connecting Clients finished
- **1. December:** Answering Mechanism and scoring finished
- **8. December:** Clock sync implemented
- **15. December:** Hand in Presentation Slides and Logo
- **17. December:** Hand in Code

6. REFERENCES

[1] ETH EDU App. `https://www.ethz.ch/content/associates/services/de/lehre/lehrbetrieb/it-services-lehrbetrieb/lehrunterstuetzende-applikationen/eduapp-service.html`. Accessed on 15 Nov 2017.
[2] Gson. `https://github.com/google/gson`. Accessed on 15 Nov 2017.
[3] Kahoot. `https://kahoot.it`. Accessed on 15 Nov 2017.