PROJECT

1. INTRODUCTION

Quiz games are a very popular nowadays and there are a few places where you can go to play them. Students of IRC were requested to develop an online quiz game, so that players can enjoy the experience anywhere.

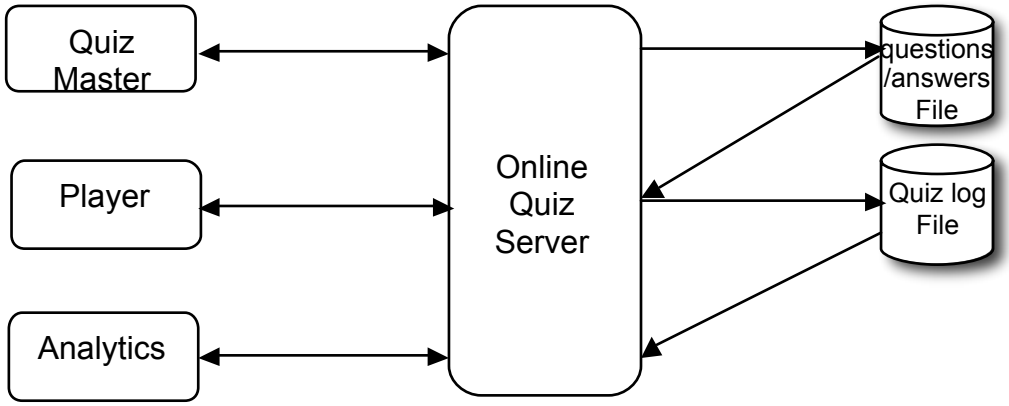
1. PROJECT SPECIFICATION

1.1 OBJECTIVES

The objective of the project is to implement an online quiz gaming system.

1.2 SPECIFICATION

The online quiz gaming quiz system should be based on a client-server architecture.



Three different clients should exist:

- a client for managing the game operations, called **quizmaster**;
- a client for managing players operations, called **player**;
- a client for getting system statistics, called **analytics**.

A quizmaster can create a new quiz, add questions to a quiz and list existing quizzes. Each quiz has its own unique name. Quizzes can only contain multiple-choice questions with 2 to 4 possible answers and a single correct answer. When adding a question to a quiz, the question text should be provided along with the text of the possible answers and the number of the correct answer. New questions can be added to a quiz at any time.

Players can start answering a quiz of their choice. They are identified by their name. Questions are presented to the player sequentially for that quiz. With each question, the existing possible answers are also presented. The player is given 60 seconds to provide an answer. If the player provides an answer within the time limit, the system informs him if the answer is correct or not. If the answer is wrong, the correct choice should be provided. The player can advance to the next question or quit the quiz at any time. If no answer is given within the time limit or a question is skipped, it counts as not answered and the correct choice should be provided.

Statistics can be provided per quiz or per player. For quiz statistics, the name of the quiz is provided by the user and the following statistics should be provided:

- total questions placed;
- total questions correctly answered;
- total questions incorrectly answered;
- total questions not answered within the time limit (timed-out);
- for each question in the quiz:
    - total answers provided
    - total answers provided for each existing choice

For players statistics, the name of the player and quiz is provided by the user and the following statistics should be provided:

- total questions placed;
- total questions correctly answered;
- total questions incorrectly answered;
- total questions not answered within the time limit (timed-out);

The server should manage the existing quizzes, the logs of answers provided. The server should store all information in text files. The server should provide application layer confirmations of all messages received from clients. The server should support multiple simultaneous clients of each type without any client blocking other clients.

An example of possible text files format representing a hypothetical state of the quiz system is:

File "networking-quiz.txt":
Format: Question text: answer 1:...: answer n: correct answer number
TCP is a reliable transport protocol: True, False, 1 The address 255.255.255.255 is?: the broadcast MAC address: the broadcast IP address: the Internet IP address, none of the above answers: 2

File "networking-log.txt":
Format: Player name: question id: number of answer provided/timeout=0: correct=1/incorrect or no answer=0
Maria: 1:1:1 Maria: 2:1:0 Paulo: 1:0:0 Paulo: 2:2:1

The file "networking-quiz.txt" contains a line for each existing question in the "networking" quiz. For each question, several fields are separated with ":". The first field

is the text of the question. Then the possible answers are listed (minimum 2, maximum 4). The last field in the line presents the number of the correct answer (1-4). Since the field separator is a ":", it is assumed that the text fields have no ":".

The file "networking-log.txt" contains a line for each existing question of the "networking" quiz that was asked. For each question asked, four fields are separated with commas. The first field is the name of the student to whom the question was asked. The second field has the question id, corresponding to the line number in the "networking-quiz.txt" file with the question. The third field has the number of the answer option provided by the student (1-4), or "0" if no answer was provided within the timeout. The fourth field has "1" if the answer provided by the student was correct or "0" if the answer provided was incorrect (or timed-out).

3. Organization of work and Rules

In order to fully commit each member of the group with this work, the responsibilities of each one of the 3 components must be assigned to a different member of the group. Group members should agree on this assignment between themselves. Each group member should specify, implement and test the functionality assigned to him

If the group has less than 3 members, students should implement only the functionality corresponding to the number of group members. No additional grade will be given for specifying or implementing functionalities of more clients than group members.

Finally, group members should integrate their work into a single project.

During the first phase of the project, team members must report the distribution of the work, according to the template provided.

A student must sign his (her) own report (by hand-writing). This signature means that he (she) is responsible for the content provided and the author of the work. In order be compliant with the university regulation and ethical principles students are advised to not sign the report if they cannot claim the authorship of the work.

2. Project Activity Plan

2.1 Activity 1: Communication Protocol Specification

A report with the protocol specification should be presented during the class in the week of 2-6 April. The group must deliver a single specification of the project, as a ZIP file through Fenix system, with the individual contributions of each member of the group. The due date is Friday, 6st April, 23:59h. Score 6/20.

Students are advised to prepare the specification before the laboratory classes so that they will be ready for discuss it and get feedback from the professor.

Each member of the group must produce a report using the template provided in Fenix system (project-specification-report-template). It should include:

- Transport protocol to be used and justification.
- Format of the messages to be used.

- Time diagrams identifying the possible sequences of actions, or state machines of the client and server applications.

2.2 Activity 2: Application Implementation – Standalone Test

Each client's functionality should be implemented separately by each group member and shown separately during the classes in the week of 9-13 April. Score 4/20.

During the standalone test, the students should perform a set of tests to show that their individual component of the application is working properly:

Some feedback can be provided to students during the classes.

The code should:

- Be developed according the client-server architecture rules.
- Be easily readable, modular and with appropriate comments if found necessary.

2.3 Activity 3: Application Implementation – Integrated Test

Students should integrate the different functionalities they implemented.

During the integration, the students should perform a set of tests that illustrates the different use cases and enables them to assess the status of integration. Each student will only be evaluated by the component that he (she) is responsible for.

- In case of not being able to successfully integrate the different parts, each student may illustrate the complete functionalities by defining and realizing a complete set of tests that illustrate the different use cases independently of other parts.
- In case of partial or fully integration, both test and results will be used for evaluation.

The final implementation of the applications, together with the list of tests (in case of not having an integrated version), should be delivered in a ZIP file through the Fenix system by Monday 16th April, 23:59h. Score 4/20.

The code should:

- Respect the specification delivered. If, during implementation, the specification was changed, an updated version of the specification can be delivered with the code.

2.4 Activity 4: Application Demonstration

The final version of the project should be demonstrated by the students to the professor during the laboratory class in the week of 16-20 April. Score 6/20.

During the demonstration, the students should perform a set of tests to show their application is working properly:

- Show the functionality implemented by each group member.
- Show the integrated functionality that the group successfully implemented, if it exists.
- You may also show partial, incomplete functionality, or functionality not integrated.
- Show exceptional communication situations.
- Examples of tests to consider: i) normal application operation; ii) communication with a server that is not available; iii) wrong number of parameters provided by the user;

iv) invalid parameters provided by the user (e.g. invalid answer choice, statistics for a quiz that does not exist); v) multiple simultaneous users.

- Other situations not shown during the demonstration may be tested afterwards.

2.4 ACTIVITY 5: INDIVIDUAL EVALUATION

Students must be prepared to an oral discussion with the professor, where they must be able to answer questions related to their design and implementation options. Additionally, the professor may ask students to solved individually a client-server programming individually, using the PC lab. This activity will be realized in the class during the week of 23-27 April, according to a schedule that will be defined later.