Web Applications I – Exam # 2 (deadline 2022-07-14 at 23:59)

“SolveMyRiddle”

FINAL VERSION - MODIFICATIONS FROM THE INITIAL VERSION ARE MARKED IN RED.

Design and implement a web application for managing a gaming platform based on posting and solving “riddles”.

Note: in the following, ‘user’ means any authenticated and logged-in user, while ‘anonymous’ means any user that is not logged in.

Every user of the application may post one or more *riddles* that other users may try to solve.

A riddle is characterized by a question (a text string), a level of difficulty (easy, average, difficult), a duration (an integer number of seconds, from 30 to 600), the correct response (a text string), and 2 hints (text strings). The duration is specific for every riddle, and the duration period is measured starting from the first response submitted by a user (the creation time of the riddle is irrelevant). A riddle may be posted by any user, and all other users may attempt to answer it, within its defined duration. The first user that provides the correct answer gets a score, the riddle becomes “closed”, and no further answers may be given. If no user provides the correct answer within the defined duration, the riddle automatically becomes “closed”.

In particular, the system should support the following functionalities:

- a user may create a new riddle, by providing all necessary information. A newly created riddle is in the “open” state, that allows entering replies.
- the user may see the status of any of their riddles that they posted as author
    - If the riddle is “closed”, the author will see all the given answers, the correct response, and possibly the identity of the winner user.
    - If the riddle is “open”, the author will see all current answers (updated every second), and a count-down showing the remaining time.
- any user may see the list of riddles, divided between “open” and “closed” ones, by viewing the question and the difficulty
    - selecting a “closed” riddle shows all the given answers, the correct response, and possibly the indication of the winner.
    - selecting an “open” riddle, gives him/her the opportunity to reply (without showing any of the other replies, of course). Any user may provide at most one reply to each riddle, and it cannot be modified after submission. In the reply to an open riddle, a count-down shows the remaining time. If the remaining time is less than 50%, the first hint is shown. If the remaining time is less than 25%, also the second hint is shown. If the reply is correct, the riddle immediately becomes “closed” and the user will get a score of 3, 2 or 1 points, depending on the difficulty of the riddle (difficult, average, easy, respectively).
- anonymous visitors will see the list of the riddles (question and difficulty), but they can’t see the replies nor the correct answer. All riddles will be shown, and their status (open, closed) should be reported.

- anonymous visitors and logged-in users may see the “top-3” ranking of the users with the highest score (in case of ties, show all the users with the 3 highest scores).

The organization of these functionalities in different screens (and possibly on different routes) is left to the student.

Project requirements

- The application architecture and source code must be developed by adopting the best practices in software development, in particular those relevant to single-page applications (SPA) using React and HTTP APIs.
- The project must be implemented as a React application that interacts with an HTTP API implemented in Node+Express. The database must be stored in a SQLite file.
- The communication between client and server must follow the “two servers” pattern, by properly configuring CORS, and React must run in “development” mode with Strict Mode activated.
- The evaluation of the project will be carried out by navigating the application. Neither the behavior of the “refresh” button, nor the manual entering of a URL (except /) will be tested, and their behavior is undefined. Also, the application should never “reload” itself as a consequence of normal user operations.
- The root directory of the project must contain a README.md file, and have two subdirectories (client and server). The project must be started by running the two commands: “cd server; nodemon index.js” and “cd client; npm start”. A template for the project directories is already available in the exam repository. You may assume that nodemon is globally installed.
- The whole project must be submitted on GitHub, on the same repository created by GitHub Classroom.
- The project **must not include** the node_modules directories. They will be re-created by running the “npm install” command, right after “git clone”.
- The project may use popular and commonly adopted libraries (for example day.js, react-bootstrap, etc.), if applicable and useful. Such libraries must be correctly declared in the package.json file, so that the npm install command might install them.
- User authentication (login and logout) and API access must be implemented with passport.js and session cookies. The credentials should be stored in encrypted and salted form. The user registration procedure is not requested.

Database requirements

- The project database must be implemented by the student, and must be pre-loaded with *at least five users,* of which three of them in the top-3 ranking. All the users should have, as authors, at least two closed riddles and an open one.

Contents of the README.md file

The README.md file must contain the following information (a template is available in the project repository). Generally, each information should take no more than 1-2 lines.

1. Server-side:
    a. A list of the HTTP APIs offered by the server, with a short description of the parameters and o the exchanged objects

    b. A list of the database tables, with their purpose
2. Client-side:
    a. A list of 'routes' for the React application, with a short description of the purpose of each route
    b. A list of the main React components
3. Overall:
    a. A screenshot of the **riddle answering page with the first hint showed.** This screenshot must be embedded in the README by linking an image committed in the repository.
    b. Username and password of the defined users.

Submission procedure

To correctly submit the project, you must:

- **Be enrolled** in the exam call.
- **Accept the invitation** on GitHub Classroom, and correctly **associate** your GitHub username with your student ID.
- **Push the project** in the **main branch** of the repository created for you by GitHub Classroom. The last commit (the one you wish to be evaluated) must be **tagged** with the tag **final**.

Note: to tag a commit, you may use (from the terminal) the following commands:

```
# ensure the latest version is committed
git commit -m "...comment..."
git push

# add the 'final' tag and push it
git tag final
git push origin --tags
```

Alternatively, you may insert the tag from GitHub's web interface (follow the link 'Create a new release').

To test your submission, these are the exact commands that the teachers will use to download the project. You may wish to test them on a clean directory:

```
git clone ...yourCloneURL...
cd ...yourProjectDir...
git pull origin main  # just in case the default branch is not main
git checkout -b evaluation final # check out the version tagged with
'final' and create a new branch 'evaluation'
(cd client ; npm install)
(cd server ; npm install)
```

Ensure that all the needed packages are downloaded by the npm install commands. Be careful: if some packages are installed globally, on your computer, they might not be listed as dependencies. Always check it in a clean installation.

The project will be tested under Linux: be aware that Linux is case-sensitive for file names, while Windows and macOS are not. Double-check the case of import and require() statements.