Web Applications I – Exam # 1 (deadline 2021-06-28 at 23:59)

"Survey"

FINAL VERSION – Modifications are reported in "red"

Design and implement a web application to manage the creation of an online survey and the collection of responses through that survey.

The application must implement the following specifications.

A survey is composed of a **title** and a list of one or more **questions**. The questions may be of two different types:

- **Closed-answer** question. It is composed of a **question title** and a list of texts for possible **answers**. The question is customized by specifying the *minimum number* and the *maximum number* of answers that will be accepted[1] in filling the response. Questions may be limited to a maximum of 10 answers.
- **Open-ended** question. It is composed of a **question title** and a **free text area** for the answer (max 200 characters). The question may be flagged as **mandatory** or **optional**.

There are two types of users: **administrator** (who may create surveys and view their responses) and **user** (who may respond to a survey).

The **administrator** must authenticate with a username/password pair. Once authenticated, an administrator may:

- Create a new survey, by defining its title and questions. In this phase, the possible actions are:
  - Creating a new question (by collecting all required information and options for that question).
  - Modifying the order of the inserted questions (with up/down actions available for each question).
  - Deleting a question.
  - Publishing the survey. From this moment, the survey can no longer be modified, and becomes visible to the users, from the home page of the website.
- View the results of their published surveys. The possible actions are:
  - View the list of surveys published by this administrator, by listing their title and number of received responses.

[1] By setting these values, we may create different types of questions:
- min = 0, max = 1 → optional question, single-choice
- min = 1, max = 1 → mandatory question, single-choice
- min = 0, max > 1 → optional question, multiple-choice
- min = 1, max > 1 → mandatory question, multiple-choice

- By selecting one of these surveys, allow navigation through the answers given by the users. The page will show the name of the user, followed by all given responses[2]. The page will allow navigating (forward/backward) across the received responses ~~of other users~~.

The **user** must not authenticate to the website. From the main page, a user may choose one of the published surveys, and start responding to it. Initially he/she must insert their *name* (free text field), and then he/she may proceed to giving answers. Each question will clearly show its validity constraints (min/max/mandatory). The survey may be submitted only if all constraints are satisfied. Once the survey is submitted, it may no longer be modified, and the user is brought back to the main page.

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 "React Development Proxy" pattern, and React must run in "development" mode.
- The root directory of the project must contain a `README.md` fine, and have two subdirectories (`client` and `server`). The project must be started by running the two commands: "`cd server; nodemon server.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 already installed globally.
- 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 API access must be implemented with `passport.js` and session cookies. No further protection mechanism is required. The user registration procedure is not requested.
- The project database must be implemented by the student, and must be pre-loaded with *at least two* administrators (who created at least 1 survey each), and *at least four responses* (at least two to the same survey).

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.

[2] **Hint**: for this functionality, you may re-use the components used for creating the survey, configured in read-only mode.

1. A list of 'routes' for the React application, with a short description of the purpose of each route
2. A liste of the HTTP APIs offered by the server, with a short description of the parameters and o the exchanged objects
3. A list of the database tables, with their purpose
4. A list of the main React components
5. A screenshot of **the page for creating a survey, showing two questions** (one per type). This screenshot must be embedded in the README by linking an image committed in the repository.
6. Username and password of the administrators, and the list of surveys created by each of them.

Submission procedure (important!)

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 (fron 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 submisison, 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 PC, 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.