Web Applications I – Exam # 4 (deadline 2021-10-26 at 23:59)

“Forms”

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

The application must implement the following specifications.

A form is composed of a **title** and a list of one or more **questions**, represented as a *grid* of options.

Each grid-like question is composed of a list of row headings and a list of column headings, whose number and contents are customized per each question by the creator of the question. Each grid cell (row, column pair) may be selected or not (e.g., with a checkbox or similar element). As an example, a grid question might have the brands of a mobile phone as row headings, and their strengths as column headings (speed, storage, image quality, cost, …).

For each question, the creator must additionally set a minimum and maximum number of cells that may be selected (by the users), in the range from 0 to rows*columns. The total number of cells selected by the user (counting all cells in the grid) must be included in this minimum-maximum range, as a condition for the validity of the response.

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

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

- Create a new form, by defining its title and grid questions. In this phase, the possible actions are:
    - Creating a new grid question (by specifying all required information: row headings, column headings, minimum and maximum number of options).
    - Deleting a grid question.
    - Publishing the form. From this moment, the form can no longer be modified, and becomes visible to the users, from the public home page of the website.
- View the results of their own published forms. The possible actions are:
    - View the list of forms published by this administrator, by listing their title and number of received responses.
    - By selecting one of these forms, allow viewing the answers given by all users, all on a single page.

The **user** must not authenticate to the website. From the main page, a user may choose one of the published forms, and start responding to it. Initially he/she must insert their *name* (free text field, that needs not be unique), and then he/she may proceed to giving answers. Each question will clearly show and check its validity constraints (min/max). The form may be submitted only if all constraints are satisfied. Once the form 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 form each), and *at least four responses* (at least two to the same form).

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. A list of 'routes' for the React application, with a short description of the purpose of each route
2. A list 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 form, showing two questions**. 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 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 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.