Web Applications I – Exam # 3 (deadline 2020-09-09 at 23:59)

"Scheduling oral tests"

FINAL VERSION – Additions and corrections are shown in red

Design and implement a web application to allow teachers and students to seamlessly define the scheduling for oral tests, where a large number of students must be allocated in individual time slots over a number of days.

There are two types of users of the system: teachers and students. Teachers must always be authenticated to the system (all operations available to teachers must be authenticated). Operations carried on by students don't need to be authenticated (they are always available).

We assume that every teacher is associated with exactly one course, for which the list of students is stored in the database. After login, the teacher has access to the following functionality:

- **Create an exam and a schedule**. This action requires 4 different phases:
    1. The teacher selects a subset of the students of the course (may select only students who haven't yet passed the exam): the selected students will have to take the oral test.
    2. Later, the teacher defines T1, i.e., the duration of a "slot" corresponding to a single oral test (e.g., 15 minutes).
    3. The teacher then defines one or more "sessions" (defined by: day, starting time, duration – with the duration multiple of T1). While sessions are defined, the system shows (in real time) the number of students selected in step 1, and the number of available slots (may be a positive or negative difference).
    4. If the difference is positive or zero, the teacher may save and close the schedule.
- **Execute an oral test**. The teacher selects one of the slots (for ease of debug, we assume that it corresponds to the current time and date, without checking), where a student is scheduled. The teacher will specify whether the student is Present or Absent. If he/she is present, then the teacher will specify a mark: 18-30, 30L, Fail, Withdraw. Students who get a positive mark (18 to 30L) will not be able to book other slots in the same exam.
- **View results overview**. The teacher views the list of all selected students. For each student, either the scheduled slot date and time is shown (for students booked and not already passed), or the assigned mark (for orals already taken), or missing for students who haven't booked yet.

The functionality visible to students does not require authentication, but just needs the student to enter his/her Student ID number (no password). Students may:

- **Book one available slot** (not already taken by other students). To see the list of slots, the student must first choose one of the possible exams.
- **View his/her currently booked slots**. If the oral is already taken, it shows the result. If the oral is not already taken, it shows the date/time, and gives the possibility of deleting the appointment.

Project requirements

- The application architecture and the source code should be developed by adopting the best practices in software design, in particular for single page applications using React and REST.
- The project must be realized as a React Application, interacting with a REST API implemented in Node+Express. The database should be stored in an SQLite file.
- The communication between client and server must follow the "React Development Proxy" pattern and React must run in development mode.
- The top-level directory must have a `README.md` file and contain two sub-directories (`client` and `server`). The project must be run with the following commands: "`cd server; nodemon server.js`" and "`cd client; npm start`". A skeleton of the project directories is provided.
- The whole project must be submitted on GitHub, in the repository created by GitHub Classroom.
- The project must not include the `node_modules` directories. They shall be re-populated by running "`npm install`", immediately after "`git clone`".
- The project may use popular and commonly adopted libraries (such as `moment.js`, `react-bootstrap`, etc.), if applicable and useful. Such libraries must be correctly declared in `package.json` so that `npm install` may download them.
- User login and access should be protected with a JWT token, stored in an http-only cookie. No additional protection is required.
- You may assume that a single student will be using the system at a time, therefore you don't have to consider possible concurrent operations my multiple users.
- The project data-base must be defined by the student and must be preloaded with at least 2 example teachers/courses (the registration procedure is not required), at least 10 students for each course (partially overlapping), and at least 3 exams defined, and some students already booked.

Contents of README.md

The README.md file must contain the following information (a template is available in the project skeleton – in general each explanation should be no longer than 1-2 lines):

1. A list of the React Application Routes, with a short description of each route's purpose
2. A list of REST APIs offered by the server, with a short description of the parameters and the exchanged entities
3. A list of the database tables, with their purpose
4. A list of the main React components adopted in the application.
5. A screenshot of the **page for defining a new slot** (embedding a picture stored in the repository).
6. The usernames and passwords of the 2 test teachers. The student IDs of 3-4 students for each course.

Submission procedure (important!)

For successfully submitting the project, you need to:

- be **enrolled** for the exam
- **push the project** to the `master` branch of the repository that GitHub Classroom generated for you. The last commit (the one that you want corrected) must be **tagged** with the `final` tag

Note: for tagging 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 a tag from the GitHub web interface (follow the link 'Create a new release').

If you want to check your submission, these are the commands that WE will use to download your project... you might want to try them in an empty directory:

```
git clone ...yourCloneURL...
cd ...yourProjectDir...
git pull origin master  # just in case the default branch is not master
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 will be retrieved with the `npm install` commands.

The project will be tested under Linux (beware that Linux is case-sensitive in file names, while MacOS-X and Windows are not).