## Web Applications I 2024/25 – Exam #2 (deadline 2025-07-06 at 23:59)

# “Class Assignments”

FINAL VERSION. Modifications from the draft text are shown in red.

A set of NT teachers of a class of NS students must organize and evaluate a series of *assignments* to be carried by groups of students.

An *assignment* consists of a *‘question’*, a *‘group’* of students who are involved, an *‘answer’* given by the students, and a final *‘evaluation’* (an integer score from 0 to 30).

A logged-in *teacher* may perform the following operations:

- Define a new *assignment*, by writing the *question* and selecting the *group* of participating students (among all the students). The system must prevent the creation of a group where any pair of students already participated together in at least 2 previous assignments among those created by the logged-in teacher (both open and closed ones). Immediately, after creation, the assignment becomes “*open*”. A group must include between 2 and 6 students.
- View the *answer* given to the assignment and enter an *evaluation score,* only if an answer has already been provided by the students. After entering the evaluation score, an assignment is “*closed*” and cannot be further modified (neither the answer nor the score can be modified).
- View the overall status of the class, by listing, for each student and referring to his/her own assignments, how many open assignments, how many closed assignments, and the average score that he/she got. This list can be sorted by alphabetical order, by the number of (total) assignments for the student, and by the average score.

A logged-in *student* may perform the following operations:

- View the open *assignments* in which he/she is involved.
- Enter and submit the *answer*. Any student in the group may submit an answer, and the answer may be changed/updated by the same student or by any other student of the same group, until the teacher evaluates it. The answer to the assignment is unique for the group.
- View the evaluation scores that he/she received in all closed assignments he/she was involved in, as well as his/her overall average score.

Notes:

- At any time, any number of assignments may be open.
- All NT teachers supervise the same set of NS students.
- All NT teachers see and evaluate their own assignments (and not those by the other teachers).
- Each assignment is given to a single group of students; in case the teacher wants to give the same assignment to different groups, he/she must create several ~~(identical)~~ assignments (with the same text, but different student groups).
- The average score of each student is computed as weighted average, where weights are the inverse of the number of students in the group (a score in a group of 6 students counts ½ than the score in a group of 3 students).

- For simplicity, both the question and the answer are just arbitrary-length text blocks.

The organization of these specifications 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. APIs should be carefully protected and the front-end should not receive unnecessary information.
- The application should be designed for a desktop browser. Responsiveness for mobile devices is not required nor evaluated.
- The project must be implemented as a React 19 application that interacts with an HTTP API implemented in Node+Express. The Node version must be the one used during the course (22.x, LTS). The database must be stored in a SQLite file. The programming language must be JavaScript.
- 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.mjs`" and "`cd client; npm run dev`". A template for the project directories is already available in the exam repository. You may assume that nodemon is globally installed. No other global modules will be available.
- 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. All required 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 an encrypted and salted form. The user registration procedure is not requested nor evaluated.

## Quality requirements

In addition to the implementation of the required application functionality, the following quality requirements will be evaluated:

- Database design and organization.
- Design of the HTTP APIs.
- Organization of React components and routes.
- Correct usage of React patterns (functional behavior, hooks, state, context, and effects). Avoiding direct manipulation of the DOM is included in these rules.
- Code clarity.

- Absence of errors (and warnings) in the client console (except those caused by errors in the imported libraries).
- Absence of application crashes or unhandled exceptions.
- Essential data validation (in Express and React).
- Basic usability and user-friendliness.
- Originality of the solution.

## Database requirements

- The project database must be realized by the student and must be preloaded with a class of at least 20 students and 2 teachers. One of the teachers must have created at least 2 assignments (with different student groups), one still open and one already closed. As a simplification in the construction of the database, you may assign the same password to all students.

## 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 item 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 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. Two screenshots of the **application, one with the creation of an assignment and the other with the overall status of the class**, respectively. The screenshots must be embedded in the README by linking two images committed in the repository.
    b. Usernames and passwords of the registered users.

## Submission procedure

To correctly submit the project, you must:

- **Be enrolled** in the exam call.
- Use the provided **link** to **join the classroom** on GitHub Classroom (i.e., correctly **associate** your GitHub username with your student ID) and **accept the assignment**.
- **Push the project** into 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: `final` is all-lowercase with no spaces, and it is a git 'tag', nor a 'commit message').

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 and run 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; npm run dev)
(cd server ; npm install; nodemon index.mjs)
```

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.

Be aware that Linux is case-sensitive for file names, while Windows and macOS are not. Double-check the case of import statements and all file names.