Web Applications Sample instructions (a.y. 2024/2025), version 3.

ALWAYS CHECK the ones specific for your exam, which have priority over this ones.

Project requirements

- User authentication (login and logout) and API access must be implemented with `passport.js` and **session cookies, using the mechanisms explained during the lectures**. Session information must be stored on the server, as done during the lectures. The credentials must be stored in hashed and salted form, as done during the lectures. **Failure to follow this pattern will automatically lead to exam failure.**
- The communication between client and server must follow the multiple-server pattern, by properly configuring CORS, and React must run in "development" mode with Strict Mode activated. **Failure to follow this pattern will automatically lead to exam failure.**
- 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 an SQLite file. **Failure to follow this pattern will automatically lead to exam failure.**
- The evaluation of the project will be carried out by navigating the application, **using the starting URL http://localhost:5173**. Neither the behavior of the "refresh" button, nor the manual entering of a URL (except /) will be tested, and their behavior is not specified. Also, the application should never "reload" itself as a consequence of normal user operations.
- The user registration procedure is not requested *unless specifically required in the text*.
- 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 root directory of the project must contain a README.md file and have the same subdirectories as the template project (`client` and `server`). The project must start by running these commands: "`cd server; nodemon index.js`", and "`cd client; npm run dev`". A template for the project directories is already available in the exam repository. Do not move or rename such directories. You may assume that `nodemon` is globally installed.
- The whole project must be submitted on GitHub in the repository created by GitHub Classroom specifically for this exam.
- The project **must not include** the `node_modules` directories. They will be re-created by running the "`npm ci`" command, right after "`git clone`".
- The project **must include** the `package-lock.json` files for each of the folders (client, server).
- 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` and `package-lock.json` files, so that the `npm ci` command can download and install all of them. Missing required libraries in the json file will lead to mark reductions.

Database requirements

- The database schema must be designed and decided by the student, and it is part of the evaluation. Some content must be preloaded in the database.

- *The exact preloaded content depends on the specific exam specifications. Each exam will have specific instructions here.*

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 of the exchanged objects.
    b. A list of the database tables, with their purpose, and the name of the columns.
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 developed for the project. Minor ones can be skipped.
3. Overall:
    a. A screenshot of the **… (*this depends on the specific exam*)**. The screenshot must be embedded in the README by linking the image committed in the repository.
    b. Usernames and passwords of the users (*potentially with other info, e.g., admin roles: this depends on the specific exam*)

Submission procedure (IMPORTANT!)

To correctly submit the project, you must:

- **Be enrolled** in the exam call.
- **Accept the invitation** on GitHub Classroom, using the link specific for **this** exam, and correctly **associate** your GitHub username with your student ID.
- **Push the project** in the **branch named "main"** 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 whitespaces, and it is a git 'tag', NOT 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
```

NB: **the tag name is "final", all lowercase, no quotes, no whitespaces, no other characters**, and it must be associated with the commit to be evaluated.

Alternatively, you may insert the tag from GitHub's web interface (In section 'Releases' 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 in an empty 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 ci; npm run dev)
(cd server ; npm ci; nodemon index.js)
```

Make sure that all the needed packages are downloaded by the `npm ci` 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 (for instance, in an empty Virtual Machine).

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 upper/lowercase characters, especially of `import` and `require()` statements, and of any filename to be loaded by the application (e.g., the database).