Web Applications I – Exam # 2 (deadline 2021-07-12 at 23:59)

“Meme Generator”

FINAL VERSION – Modifications are reported in “red”

Design and implement a web application to manage the creation and the visualization of *memes*.

The application must satisfy the following requirements.

A *meme* is composed of the following properties:

- a **title**,
- an **image**,
- one or more **sentences** overlaid above the image,
- a **visibility** status (public, or protected),
- and is associated with the **name** of its creator.

For simplicity, assume that the site already has a set of predefined “background” images (this set cannot be modified, and must be served by React or Express), as the starting image for composing memes. Each image also predefines the number of supported text fields (1, 2 or 3), and the position of such fields over the image (different images will position the text in different locations). The definition of a meme, therefore, simply consists in choosing a background image and in defining the content of the text fields allowed for that image. There is no requirement to create a “bitmap” with the composed meme, but just to visualize the image with the text overlaid in the predefined positions.

Memes may be created uniquely by users of type **creator**. Memes with “public” visibility will be visible to all visitors of the web application, while the “protected” ones will be visible to other creators, only.

Anyone who visits the website (i.e., any non-authenticated user) will visualize the list[1] of the *public* memes in the home page; by selecting one of them, he/she will see all its properties: title, image with overlaid text (i.e., the actual meme), and the name of the creator.

Users of type **creator** must authenticate with username/password. Once authenticated, a creator visualizes the list of *public* and *protected* memes (by all creators); by selecting one of them, he/she will see all its properties. Furthermore, from ~~this~~ the page with the list of memes, the creator may:

- Create a new meme, by defining its **title**, the background image and the sentences to be used. In this phase, the possible actions are:
    - Select an **image** (choosing among the predefined ones) to be used as the meme background.
    - Write the **text** of the meme, that will appear in the predefined areas of the image. All text fields should be considered as optional (for example, if the image defines two text fields, the creator may decide to fill only one), but at least one of them must be filled. The creator may choose among at least 2 types of fonts (of predefined size) and 4 different colors, that will apply to *all the text* in the meme.

[1] The title is required, visualizing thumbnails is allowed, but not required.

    - **Save** the just created meme, while specifying its visibility ('public' or 'protected').
- Copy a meme, in two possible ways:
    - If the meme to be copied belongs to the same creator, make a copy and allow changing title, text content, visibility and text attributes (i.e., all properties *except* the background image).
    - If the meme belongs to a different creator, make a copy an allow changing the title and the text (including its attributes). If the copied meme was protected, the new meme must have the same visibility; if it was public, the new one may be saved as public or protected. The copy will belong to the new creator.
- Delete one of his/her own memes.

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` file 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` and `package-lock.json` files, 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 three* creators, who created at least 3 memes each, one of them by copying a meme of another user. At least 6 images need to be available: at least 2 with 1 text field, 2 with 2 fields, and 2 with 3 text fields.

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 meme**. This screenshot must be embedded in the README by linking an image committed in the repository.
6. Username and password of the ~~administrators~~creators, and the list of ~~surveys~~ memes 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 (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 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.