01UDFOV/01TXYOV – WEB APPLICATIONS I

BIGLAB 2D: NOW WITH AUTHENTICATION

WHAT ARE WE BUILDING THIS WEEK?

We will add the possibility of **having multiple users** to both our back-end and front-end applications, enabling them to **authenticate** (login and logout functionalities) and manage their tasks. Access to the page showing the list of tasks **will then be refused** to non-authenticated users.

STEP-BY-STEP INSTRUCTIONS:

- In the front-end, add a new page (with a dedicated route) with a form, which will be used to log in a user. The page should be well structured in terms of needed components and appropriate states. The login form will have two *mandatory* fields: **email** and **password**, both **validated** properly by the client.
- Implement the **login** process by exploiting the **Passport** authentication middleware in express, as shown in the lectures, and the "users" table of the provided database. You are free to add new tasks and users inside your database.
  Beware: *do not store* **plain text passwords** *in the database*! Use **bcrypt** (see the hints) to generate a hash of the passwords before saving them. For testing the application, list the usernames and the plain text passwords of the example users in the project README.md file included in the repository.
- When the login process **fails**, the front-end should display a suitable error message (e.g., "*Incorrect username and/or password*" or similar) and continue to show the login form. Instead, when the login is **successful**, the application redirects the user to her task list page, showing a message like: "*Welcome, {name_of_the_user}*".
- Implement the **logout** functionality, again by exploiting the **Passport** authentication middleware. When the user is logged out, redirect her to the login form.
- Identify the routes that need to be authenticated (e.g., those to get the list of tasks) and *protect* them accordingly. Non-authenticated users must not see any task, meanwhile authenticated users must see only their tasks.

Hints:

1. *The general specification of the second BigLab can be found at: https://github.com/polito-WA1-AW1-2021/course-materials/blob/main/labs/BigLab2/BigLab2.pdf*
2. *You can use the following website to generate the hash of a password, according to bcrypt:* https://www.browserling.com/tools/bcrypt.
   If the generated hash starts with $2y$, replace that part with $2b$, as the node bcrypt module does not support $2y$ hashes.
3. *You can install the bcrypt node module with npm. Documentation is available at https://github.com/kelektiv/node.bcrypt.js*