Web Applications [2023/2024]

Lab 14: Multi-user Application with Authentication

In this lab, you will make your application multi-user by implementing the login functionality. A modified database is provided in the solution folder, where a new table "user" has been added, and foreign keys (instead of names) are used to identify users in other tables.

1. Login and Logout Users

You will add the possibility of **having multiple users** in your application, enabling them to **authenticate** (i.e., *login* and *logout* functionalities) for managing their films. Non-authenticated users will not see a list of films anymore.

- In the front-end, add a new page (with a dedicated route) with a form, which will be used to log in. 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**. The login form should be validated before its submission, and you must use proper error messages when inconsistencies are found. Specifically, the following checks should be executed:
  - When a field is missing or empty it must be forbidden to send a log-in request to the server.
  - The email should be formally correct. Use, for instance, the HTML5 "type=email" form field or a library to perform email validation: do not write your own validation function.
- In Express, implement the **login** process by exploiting the **Passport** authentication middleware.
  - You can use the database provided in the folder with the code for this lab.
  - Add a new user with username "*u4@p.it*" and password "*pwd*" in the database.
    Beware: do not store **plain text passwords** in the database! Use the **scrypt** approach explained during the lectures (*see the hints*) to generate a hash of each of the passwords before saving them.
  - Associate some films in the database to the newly created user.
- When the login process **fails**, the front-end should display a suitable error message (e.g., "*Incorrect username or password*") and continue to show the login form. Instead, when the login is **successful**, the application redirects the users to their film list page, showing a message like: "*Welcome, {name of the user}*".
- Identify the routes that need to be authenticated (e.g., those to get or modify the list of films) and *protect* them accordingly. Non-authenticated users must not see any film, meanwhile authenticated users must see **only** their films.
- Implement the **logout** functionality, again by exploiting the **Passport** authentication middleware. When the users are logged out, redirect them to the login form.

Hints:

1. *The password of all the users already inserted into the database provided in the folder where the solutions will be put is pwd.*
2. *You can use the following website to generate the hash of a password according to scrypt, as explained during the lectures: https://www.browserling.com/tools/scrypt.*
   If you need to generate a salt by hand, you can use https://www.browserling.com/tools/random-hex, setting the length of the generated hex as the one of the salt (e.g., 16).