# Champlain College - Lennoxville
## Lab 4: Register/Login in a web app.

| | | |
|---|---|---|
| PROGRAM: | 420.B0 Computer Science Technology | |
| COURSE: | Transactional Web Applications 1 | |
| COURSE CODE: | 420-430-LE | |
| WEIGHT: | 6% of the final score | |
| SEMESTER: | Winter 2023 | |
| INSTRUCTOR: | Francis Gauthier<br>fgauthier@crcmail.net | Office C-239 |

## Objectives

- Practice security authentication mechanism such as password hashing, signing JSON web tokens and validating JWTs.
- Practice building a first robust authentication flow
- Practice client-side validation

## Your task

Produce a full-stack web application that allows:

1. A user to register a new account into the web application
2. A user to login into the web application, after successful registration
3. A user to favorite movies and see their favorite movies first

Each task is broken down below.

### Working in Teams

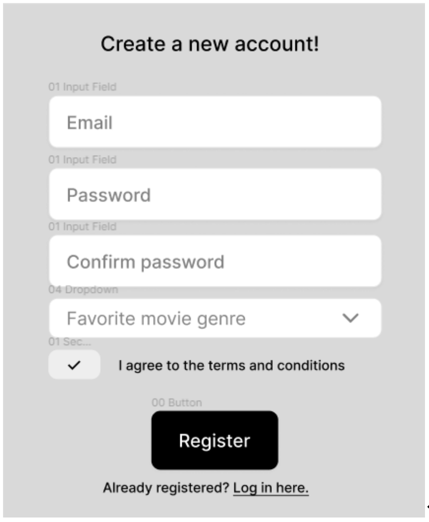
The assignment is meant to be done in a team of 2.

Each task is broken down into client-side and server-side. Each team is encouraged to have a team member that specializes in the client-side portion and the other on the server-side portion.

The assignment can be done individually, but the scope of the assignment will not be reduced.

## User registration (part 1)

### Client-side



← MockUp

Each field is **mandatory**. Fields to validate:

| **Input** | **Type** | **Validation to perform** |
|---|---|---|
| Email | string | Email is a valid email format |
| Password | string | Minimum characters: 8<br>Contains at least one special character<br>Contains at least one lowercase character<br>Contains at least one uppercase character |
| Confirm Password | string | Matches the password exactly |
| Favorite movie genre | string | A choice between:<br>`["Drama", "Comedy", "Action", "Sci-fi", "Animation", "History", "Horror", "Romance"]` |
| Terms and condition | boolean | Is checked |

You must perform client-side validation. The user must receive instant, accurate feedback on the fields with incorrect data format or missing information.

### Logic to perform:

- Validate the form inputs
- Sends an HTTP request to the server with the form input, if valid
- Gives feedback on success/error to the user based on the server's response

## Server-side

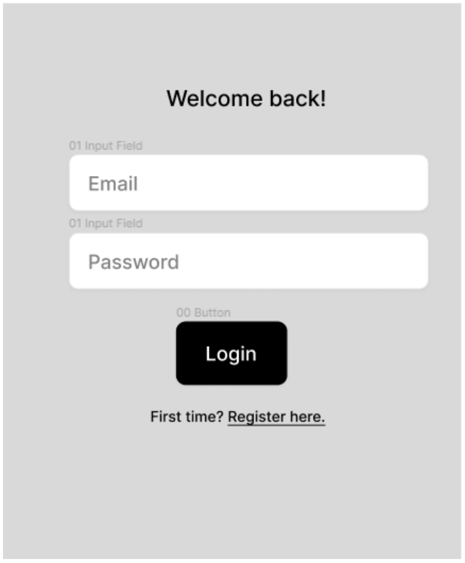
The server must implement a POST /register route.

Requirements:

- POST route
- Expects a JSON body containing the register form values
- Performs server-side validation on the fields (through regex or mongoose schema)
- Hashes the password using *bcrypt*
- Stores the information of the user, along with the password hash, in a ***users*** collection in MongoDB. <u>The original password is never stored</u>.
- Returns a 201 status code on success

## Signing in (part 2)

### Client-side



← MockUp

Each field is **mandatory**. Fields to validate:

| **Input** | **Type** | **Validation to perform** |
|---|---|---|
| Email | string | Email is a valid email format |
| Password | string | No validation done here. |

You must perform client-side validation. The user must receive instant, accurate feedback on the fields with incorrect data format or missing information.

#### Logic to perform:

- Validate the form inputs
- Sends an HTTP request to the server with the form input, if valid
- Gives feedback on success/error to the user based on the server's response
- If a JWT token is received, stores it in the LocalStorage
- Redirects to the home page after 2 seconds on login success.

### Server-side

The server must implement a POST /login route.

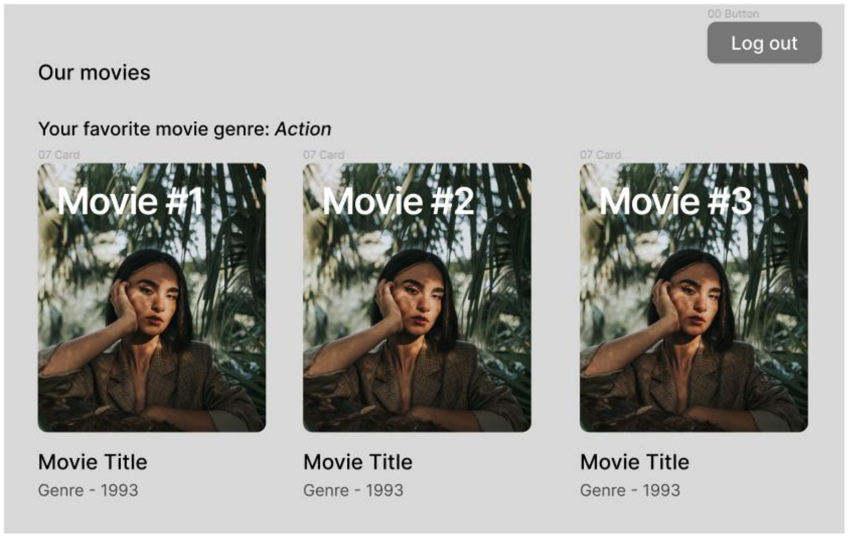
Requirements:

- POST route, expects a JSON body containing the login form values
- Locate the user with the email received
  - Sends back a 4XX status code if the user is not found
- Compares the password with the hashed password found in the database
  - Sends back a 4XX status code if the password does not match
- Signs a JWT with a secret if the password match
  - The payload of the token should contain the favorite movie genre
- Returns a 200 status code on success, sending back the token in the message body.

## Movies home page (part 3)

### Client-side

Mockup:



Components to display:

- A logout button that logs out the user and returns to the login page
- A text displaying the favorite movie genre
- A scrollable list of 25 movies. Each movie displays:
    - The poster
    - The movie title
    - The movie genre
    - The movie year

### Logic to perform:

- Perform a GET /movies request on the server
    - Provide the JWT in the HTTP headers to authenticate properly
- Upon success, use the movies document data to display the movies properly

### Server-side

The movies document are available here: Mflix_movies

The server must implement a GET /movies route.

Requirements:

- GET route
- Expects a JWT to be present in the *authorization* header
    - Sends back a 401 when missing the token
- Decrypt the JWT token to locate the favorite movie genre of the user
- Retrieve 25 movie documents that are matching the favorite genre
- Send back a 200 status code and the movie documents on success

## Submission

The assignment must be pushed to a GitHub repository and given access to the teacher (**frangauthier** username on GitHub).

The deadline for submission is **April 2nd 2024, End of day**. Late submissions are accepted with 10% penalty if no agreement was reached with the teacher.