COMP 4513
Assignment #2:
Node + MongoDB + Simple Authentication

Due Sat April 8th at midnight (since final exams start on Apr 11, I recommend finishing earlier)
Version 1.0, March 9, 2023

Overview

This assignment will consist of two deliverables. The first is creating and hosting a Node and MongoDB based API. The second will be a simple authentication system using Node and MongoDB-based. This group assignment provides an opportunity to display your front-end, back-end, and dev-ops capabilities. You will be working alone or with a partner.

Submitting

You will not be submitting your source code. Instead, I will mark your code from a GitHub repository. I will mark the functionality by visiting some type of server. Thus, you will submit your assignment by emailing me a short message consisting of the group member names, the GitHub repository URL, and the web address where I will be able to find your working assignment.

Grading

The grade for this assignment will be broken down as follows:

Visual Design and Styling	10%
Programming Design and Documentation	10%
Functionality (follows requirements)	80%

Data Files

I will be providing you with a variety of JSON data files that you will import into MongoDB. The github repo for the start files will be at:

https://github.com/mru-comp4513-archive/comp4513-w2023-assign2

Requirements

1. In this assignment, you will implement (and host) an expanded version of the movie API using Node and MongoDB.
2. You must host your assignment's Node components on a server.

   For this assignment, I am going to recommend glitch.com, which provides a free option for hosting simple Node applications. Do note that glitch projects will go to sleep after 5 minutes, so be aware that the first request of a slept Node application will take some time to awaken.

   You are quite welcome to try a different hosting platform; last semester's COMP3612 students reported that it was pretty straightforward to implement hosting on glitch.

   Rather than hosting MongoDB on glitch, I would strongly recommend using a third-party Mongo service such as Mongo Atlas. In such a case, your Node application will simply connect to your MongoDB database that is running on Atlas's servers.

   Regardless of your approach, it will take time to get this working so please don't leave this to the last few days!

   When your glitch hosting is working and the assignment is ready to be marked, then send me an email with the following information:

   - The URL of the github repo so that I can mark the source code. If your repo is private, then add me as a collaborator.
   - In the readme.md file for your repo, provide a link to each of the APIs on glitch so I can test them (see details below).
   - The URL of the home page.

3. API Functionality

You must create the following APIs with the specified routes and functionality.

/api/movies	Returns all movies. The file sample_movie.json illustrates the expected format for all these apis.
/api/movies/limit/num	Returns the first num movies. The num field must be between 1 and 200.
/api/movies/id	Returns JSON for the single painting whose id matches the provided id.
/api/movies/tmdb/id	Returns movie whose tmdb_id matches the provided id.
/api/movies/year/min/max	Returns all movies whose year is between the two supplied values. If min is larger than max, then return error message.
/api/movies/ratings/min/max	Returns all movies whose average rating is between the two supplied values. If min is larger than max, then return error message.
/api/movies/title/text	Returns movies whose title contains (somewhere) the provided text. This search should be case insensitive.
/api/movies/genre/name	Returns movies that have a genre name that matches the provided value. This should be case insensitive.

For each of the requests that take parameters, your API needs to handle a No Matches Found condition. For instance, if an id doesn't exist, return a JSON message that indicates the requested request did not return any data.

4. Login System.

You must implement a very simple login system using Node and MongoDB.

Your login page will be implemented in EJS, which is a common templating system in Node. The login form must provide a way for the user to enter their email and password. Make the default email "al@ace.ca" and the default password "mypassword". (Be sure to test with some of the other emails in the users.json file; every user has the same mypassword password).

If successful, it should redirect to the home page. This can be a very simple page that says "Home Page" and "Login Successful". This page must also provide a way logout the user (which will redirect to the login page). If the user isn't logged in yet, redirect to login form.

If unsuccessful, the form page should display again with some type of message indicating why the login failed.

Your login page must also include a header that indicates that this is Assignment #2 for COMP4513, and a footer that displays your names and the github repo link.

You are going to need some system for "remembering" whether the user is logged in. I recommend using the Passport package in Node to implement your authentication. By default, Passport uses sessions to maintain login status. Lab13b and 16b provides you with guidance for working with EJS and Passport.

Do note that you are not creating a complete security system. We just don't have quite enough time in the semester to do so!