Hristo Panayotov Advanced Database Assignment

Heroku: https://fathomless-anchorage-80930.herokuapp.com/

Github Repository: https://github.com/EzzoPanayotov/Database_Assignment

Introduction

The inspiration for this project is that by the end of 2021 I would like to create and publish my own web application, which would look something like the one I've done for this assignment. The project I have in mind is going to allow you to write your food recipe and publish it so that everyone can see it. Of course, I would like to have, comments and ratings for example. These are the two things that, I couldn't implement in my project, because I haven't done anything like that before, so I found it a bit difficult. So that's pretty much what I tried to create here.

System Overview

I use one database called "Project", inside it, I have two collections called "articles" and "users". The collections store the registered users and the created articles. My view engine is ejs(embeded javascript) and my key views are:

- Home page(index.ejs)
- Register page(register.ejs)
- Login Page(login.ejs)
- New Article(newArticle.ejs).

The application allows you to register, login, and create, read, update and delete an article. The users and the articles created or updated are being stored in MongoDB. To register you have to provide: First Name, Last Name, Email, and a Password. In order to store the users' password safely, I used a hash, which adds a level of security to my pretty much non-secure application. The login page requires you to provide an email and a password in order to get into your "account". After the log in a cookie is being created, this cookie then helps me to log out the user by just replacing it with another one which lasts one millisecond. The home page shows all created articles(if any) and after they are created, we can read more about them, edit them and delete them. For the delete and edit operations, I use a middleware called "method-override", which allows me to override the default methods GET/POST.

Key Design Decisions

At the beginning, I used "express-handlebars", but I had some problems with it, when I was trying to display data from node.js into the front-end. I decided to use "express-handlebars", because it looked easy when, I was reading it's documentation, but that just made my life harder. That's when I swapped to "ejs", I was already familiar with "ejs", because we used it in class quite a bit, and I knew everything, that I needed, in order to display the information, that I had in mind. Embeded Javascript is pretty simple to understand, and the "EJS Language support" extension, makes it a lot more easier.

I structured my source files, based on an activity we had in our classes, plus I researched it quite a bit, and most results had the same file structure. The "config" folder contains my "Secret", that's why I've added it in

the ".gitignore" file. The "controllers", of course control the connection between the database and the front-end. In them, I have the register and login authentication, and the article creation. In the "model", folder are the collections, I created, using mongoose. The "node_modules" folder is added in ".gitignore" as well, because it contains sensitive information about my project, so I think that's a best practice. I have one static folder which is "public", it contains my css file. The "routes" folder has the routes for my "views", I've also included my "models" in it, because that helps me send the information, I need to the front-end. And the "views" folder contains the "ejs" files, which are my web pages. Everything is connected to the "app.js" which is the main file.

Database Design

My database design is really simple, I have two collections within a database called "Project". The collections are not connected. I tried to connect them at some point, but I couldn't make it work. I wanted to connect them, because I wanted to, save and display the user that created this "Article". For the implementation of "comments" and "ratings" as collections, I didn't even try to make them, because of the connection, I was talking about.

Security and Scalability

The security measures, I've taken are, hashing the password inputted by the user, using ".gitignore" to ignore my "sensitive" files, and of course making it so the user is not able to create or save blank documents in my collections.

The scalability of my application is not good, because the strategy, I use for the authentication is long and slow, thus having a lot of users registering, will probably make the application a lot slower. Also displaying all "Articles" in one page is not a great idea, because if we have a thousand for example, it might take some time to render the main page.

Conclusion and Reflection

The project is working as, I wanted it to, although it still has some errors. I had some problems while, I was trying to handle the duplicate error, which was coming up when the same email was being used for two "accounts". That happened because, I wanted the email to be unique, because we don't want the same email to be related to two different "accounts". The authentication was pretty easy to make, because of REGEX(Regular expression), but that's definitely not the best solution. I, also had some problems with css, which I didn't think would be the case. The problem was that, some pages wouldn't load the css file, and so I had to add " base="/" ", to my main.ejs. The " base="" ", tag specifies the base URL for all relative URLs in a document. CRUD operations work, as they should, I believe. It was a bit hard to figure out how to "edit" and "delete" an "Article", but then I found out about the "method-override", middleware. Method-override, lets you use HTTP verbs such as PUT or DELETE in places where the client doesn't support it.

Putting the problems aside, I managed to make things work one way or another. This module and it's assignment, helped me learn and understand a lot of new technologies. In the future, I would like to improve my project, and add the parts that, I initially wanted. The project is pretty small, and simple, but it's the best work I've done so far.

References

npm. (n.d.). method-override. [online] Available at: https://www.npmjs.com/package/method-override [Accessed 28 May 2021].

npm. (n.d.). express-handlebars. [online] Available at: https://www.npmjs.com/package/express-handlebars [Accessed 28 May 2021].

developer.mozilla.org. (n.d.). : The Document Base URL element - HTML: HyperText Markup Language | MDN. [online] Available at: https://developer.mozilla.org/en-US/docs/Web/HTML/Element/base [Accessed 28 May 2021].