Adelina Advanced Database Assignment

GitHub Repository link: https://github.com/AdelinaCenusa/Database-Assignment.git

Heroku link: https://whispering-badlands-66119.herokuapp.com/

Introduction

System Overview

I have database called "Assignment_Adelina". In the database, I have two collections named: "users" and "todos". The "users" collection, contains the registered users in my application, and the other("todos"), stores all of the todo lists created.

The view engine, I use is ejs(embeded javascript) and my key views are: "header.ejs", "footer.ejs" these two are included in every page; "index.ejs" my home page; "loginUser.ejs", the log in page; "registerUser.ejs" the register page.

You are able to register using the required fields, after the registration is successful a cookie is being created which expires within one day. I use that cookie then to log out the user, by replacing it with another wich expires pretty much immediately after it's creation. And when you log in with the correct credentials the same cookie as the one after the registration is being created.

The home page shows us all of the lists we've created. We can read them, update them and delete them. In order to make the update and delete functions, I had to use the "method-override" middleware, which allows me to use "PUT" and "DELETE" requests.

Key Design Decisions

I decided to use ejs for my view engine, because we used it in class, and I got used to it a bit. Also ejs is really easy to understand, and by that, I mean that it's script tags are fairly simple, and they are easily distinguishable from one another. And there is a lot of information about it, so researching a problem was easy enough.

For my project structure, I tried to follow a task that we've done in one of our seminar classes. Having folders for the files they correspond to, makes is a bit more organizied, and easy to follow and read hopefully.

Database Design

My database design is really simple, it's a "normal" database with some collections in it. I definitely could've extended it, but I haven't spent much time on it. Ultimately I would've liked to connect the two collections, and maybe add some more collections, but I didn't have enough time for that.

Security and Scalability

The security, I have implemented is that, I hash the password that the user provides, and then save the hashed password in the database.

I don't think that my project is very scalable at the moment, because I've already experienced some slow load times. That's probably caused, because of the inefficiency of some parts of my code.

Conclusion and Relfection

The authentication and the error handling took me a lot of time, because, I wanted to make them as good as, I can. Debugging was also a big time consumer, because, I encountered some problems during the implementation of the authentication. Overall my project is "ok", it's not perfect, and I don't think that it's bad as well, so it's kind of in the middle. Everything works as intended, of course some parts work better than others,but I managed to apply most of the things, I had in mind. If I had a bit more time, I would've tried to connect the two collections that, I have, and display the data accordingly. I would've tried to make it so, an user would see the lists that that he created, instead of all users seeing all created lists. Also, I would've liked to make the list that an user has created, exprotable into an excel format somehow, so he won't have to use the application everytime he needs to see it. I will try to implement all of these features anyway, because it will help me for future projects.

I learned a lot of new technologies, methods and strategies during this project and module, of course there is still a lot to learn, but slow and steady wins the race.