Individual Extension Report
Web Technologies (E23)
T510048101-1-E23

Daniel Bermann Schmidt
University of Southern Denmark
GitHub Repository

December 28, 2023

1 Introduction

In this report I'll reflect on the group project, and then I'll describe my individual extension, which is React. I will also explain why I chose React, how I implemented it and how it has affected the project. Furthermore, I'll also reflect on the security, performance and scalability of the project.

2 Reflections

2.1 Front-end

The UI in the group project was done with Blade templates and most importantly not done with components. Therefore it isn't modular or resuable, and not generally a great approach for most applications as it limits the maintainability and scalability of the UI. Although it isn't the best implementation, it did do its job for displaying our data on the page. In terms of readability, it's generally readable for people who know Blade and if the UI isn't that complex or large.

2.2 Resource Management

When it comes to resource management. We actually started by moving all of the static files, such as CSS and JavaScript, over to the public folder instead of the resources folder, as the group thought this would make more sense. I didn't agree with this, as the resources folder is meant for things like CSS and JavaScript files, and the public folder is meant for things like images, fonts and other static files. But for the sake of the group, I went along with it. The most important part was that they were still in each their folder and organized.

2.3 Authentication & Authorization

We have basic auth functionality and middleware. Although we didn't have the best implementation, it did solve the problem at hand, wanting to block certain pages for users who weren't logged in or authorized to access certain pages. We were somewhat happy with the implementation and that it worked the way we wanted it to for the most part.

3 Individual Extension

In my invidual extension I have chosen to implement React into the project. React is a JavaScript library for building user interfaces.

3.1 Why React?

Building web UIs, it can be a challenge to keep the code organized as the project grows. React solves

this problem by allowing you to split the UI into independent, reusable pieces. This can make the code more organized.

3.2 How and what did I implement with React?

I started implementing React by installing the npm packages 'react' and 'react-dom' into the project. Then I included the JSX page file into the Blade template, and a div with a specific ID, where the React component would be rendered.

A problem was getting data from the server into the React component without calling the API from the client-side, which would require an access_token for the server to authenticate and know who called that API endpoint. I solved this in the Blade template with `json_encode()` in the data attribute of the HTML div for the React component, then parsing it again in the component.

3.3 Reflection on the implementation

I split the UI into components and made them reusable. This will make it easier to maintain. And by injecting the data into the data attribute, the React components will still display the correct data without calling the API on page load.

4 Security Reflections

In terms of security, we don't store any passwords in plain text, we salt and then hash them with bcrypt.

We have middleware that checks if user is logged in and is authorized to access certain pages. On top of this we also have CSRF protection, a thing that's built into Laravel and easy to implement.

We do have one major security flaw. On the site for a specific movie, we have comments. We didn't setup the authentication and authorization properly, so we don't have any access_tokens on the user. This also means that there was no way of authenticating the user when called an API endpoint, that would require user data. We ended up making a hidden HTML div with a data-attribute with the user ID, and then using that ID on the client. The problem with this approach is that if you just change the user ID in the HTML, you can post comments as other users. Obviously if this was a real application, we would have to fix this and setup access_token, authentication and authorization properly.

5 Performance & Scalability Reflections

As it relates to performance the system's reaction time from the POV of the user, we have optimistic updates of comments, loading states for comments, loading states for adding to watchlist and other things. In that aspect, at least from the user's POV, the system is very responsive. There are other things that could be done to improve performance, like caching, better database queries, pagination and so on.

In terms of scalability, we've not done a lot of things that make it easier to scale. In this individual extension, as I have implemented React, I have made the UI more modular and reusable, which makes the UI easier to maintain. But other things that could have been done to make it easier to scale, such as caching, load balancing, database replication etc, has not been done.

6 Conclusions

In conclusion, I have implemented React into the project, which makes the UI more modular and reusable. This, in return, makes the UI generally easier to maintain.

In terms of security, we have basic auth and CSRF protection, but we have one major security flaw, where you can post comments as other users. And in terms of performance and scalability, there has not been a lot of work done to help in that regard. Although there are certain aspects in the application that makes if feel more reponsive from the user's POV. Scalability would probably also be the most important thing to do, after fixing the security flaw.