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

Daniel Bermann Schmidt
University of Southern Denmark
GitHub Repository

December 9, 2023

1 Introduction

2 Reflections

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

6 Conclusions