CEN 308 SOFTWARE ENGINEERING

PROJECT DOCUMENTATION

Movie Review Site

Prepared by:
Benjamin Pasic
Abdulkarem Khamis

Proposed to:
Nermina Durmić, Assist. Prof. Dr.
Aldin Kovačević, Teaching Assistant

24 June 2022

TABLE OF CONTENTS

Generate your table of contents here.

Contents

1. Introduction

1.1. About the Project

Describe the project/application you were working on in a few sentences and provide *a link* to where it is deployed.

Our project was inspired by the **IMDB website**. It's a place where people can check out different reviews for different movies, as well as some basic details about different actors. Our version is a sort of "lite" version of the IMDB website, where we don't include ALL of the feature the website has, rather the main idea behind it.

The website includes a fully functioning **authentication** and **authorization** process.
We also included **JWT tokens** sent back to the users after a successful login as a cookie.

There are two roles on our website: **User** and **Editor.**

The editor has complete access to the website, where they can create new movie reviews, edit them afterwards or delete them. They can do the same things for actors.

The users are forbidden from creating new reviews and adding new actors, but every other route is accessible.

Our project is hosted on Heroku, which you can find here: https://se-project-burch.herokuapp.com/login

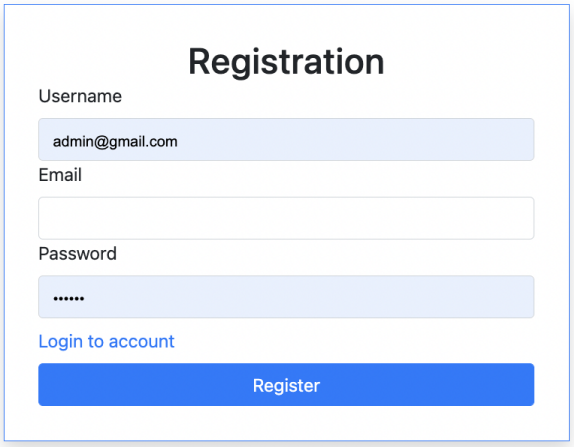
And a small note, since we're using a freely hosted db by Heroku we are limited to only 5mb of database storage, so all our movies contain the same poster (Toy Story) and the same picture for actors (Will Smith). We wanted to do something like storing imgur links in the database and using that, but imager lacks movie posters

1.2. Project Functionalities and Screenshots

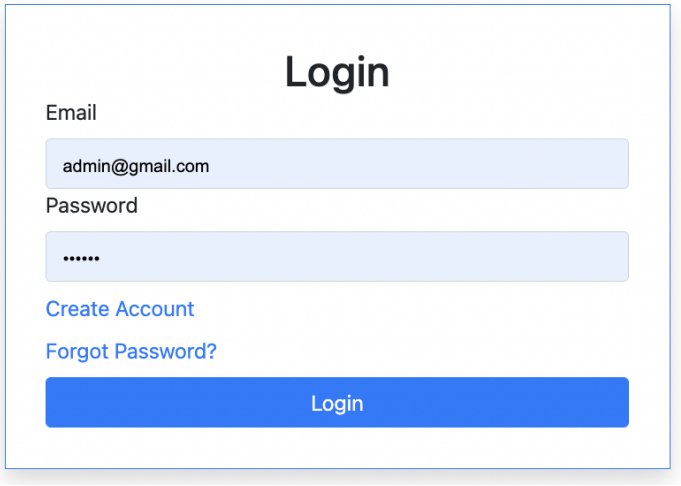
The main features are as follows:

1. Login/Register/Forgot Password

- Basic forms for logging in, registering and forgot password. The login and register forms also include error popups that tell the user which part of their login/register process is incorrect.
- The forgot password feature works by setting up a gmail SMTP account which sends out recovery links for users (with expiry time). When the user enters a correct email a unique password reset token is generated with the date of creation. When the user clicks on the recovery link, they get taken to a page with two fields for the new password and password repeat.



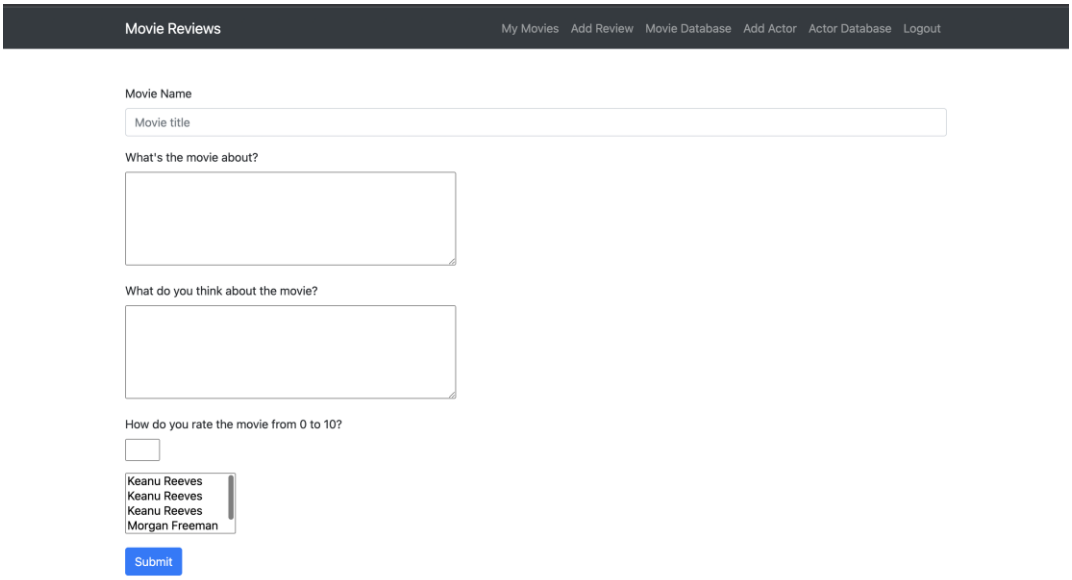
Registration Form



Login form

2. Create/Edit/Delete reviews (only as editor)

- In this part an editor can create a new movie review for everyone to see. The editor types in the basic information about the movie such as the movie name, what's it about and what do they think about the movie. They also give the movie a rating from 0 to 10 and include what actors are in the



Add Review Page

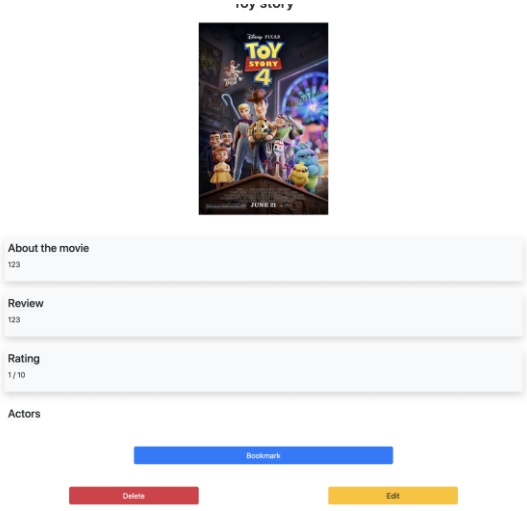
movie that are available in the database

3. Check movie/actor details, bookmark and delete or edit current movie (editor only)

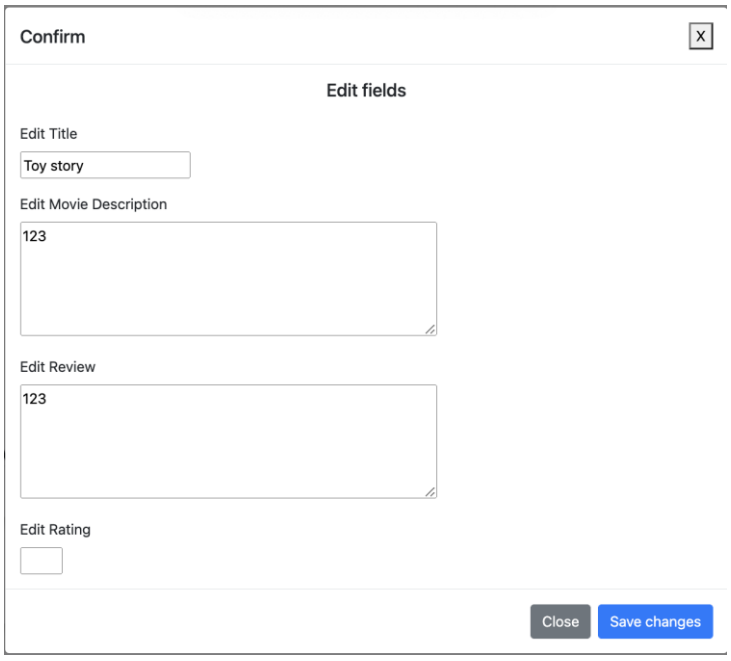
- On the homepage or if the user searched for a specific movie, once they click on that movie they can see the details about the movie. If the current user is an editor the “Delete” and “Edit” buttons appear, otherwise they don’t.

- If the user likes the movie, they can choose to bookmark it for future reference. It gets saved in the “My movies” page

- If the editor clicks on the “Edit” button, a modal appears with the current values of the review. The editor can then change the current fields.



Movie review page

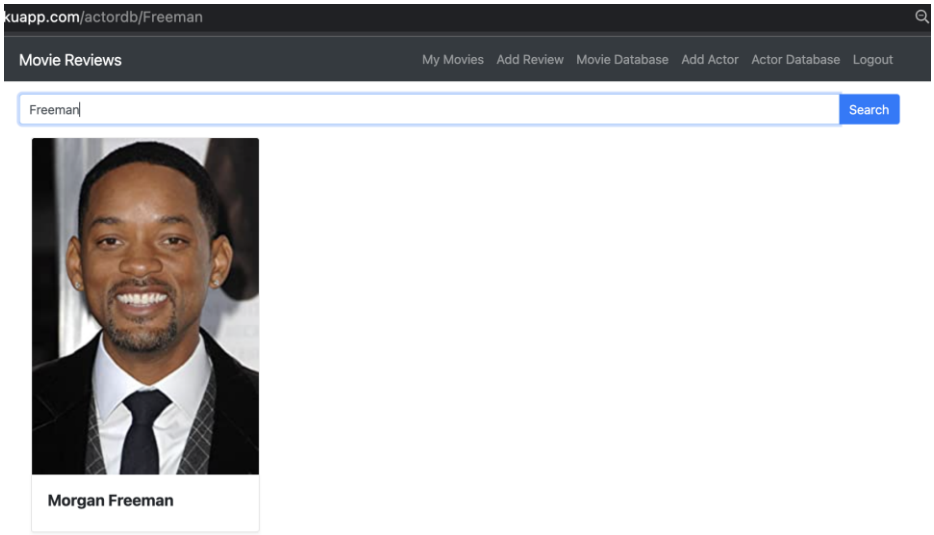


Edit Review Modal

- I only included the movie review screenshots here, but the same principle applies for actors, only the fields differ.

4. Search for movies/actors

- If a user has a specific movie or actor in mind they can search for them in the Movie or Actor database shown in the navigation bar, they click on the found movies and get taken to the movie



Search for actor

details page.

- Only included the actor database screenshot, the same applies to the movie database

2. Project Structure

2.1. Technologies

Describe or list *what technologies* (programming languages/frameworks) you used in your project for backend, frontend and the database. If you also used some other technologies or third-party tools, you could list them, as well.

Afterwards, specify which *coding standard* you used and in which part of your project (was it on the backend, frontend, both, etc.). If you are unclear about coding standards, refer to Week 2 and Week 3 on LMS.

For our **frontend** we didn't use any frameworks. It consisted of basic HTML/CSS/JavaScript and bootstrap.

For our **backend** we used Node.js with the Express framework written with JavaScript.

As our **database** we used MySQL

As well as different npm packages which can be found in the package. Json file on our GitHub

We followed Google's JavaScript coding standard which can be found here (https://google.github.io/styleguide/jsguide.html). We followed the coding standard for all parts of our project.

2.2. Database Entities

Provide a list of *tables or entities* you have in your database/schema. If it is not obvious from the name of the table/entity what it is used for, also provide a brief explanation next to it. For example:

- Actors
- Users
- Movies
- Actors_Movies - used as an intermediary table between actors and movies
- Bookmarked_Movies - used to store data about which user bookmarked which movie

2.3. Architectural Pattern

The architectural pattern that we used is the **MVC Pattern.** We used it because it provides the most clear (in our opinion) separation between three main parts of our project (Database, Views, Logic) and it provides a clear idea of the order in which things get executed.

2.4. Design Patterns

- Module pattern - This is the most common design pattern in our project. The reason we used it is because it allows each component of our project to be cleanly separated and organized. It also provides encapsulation for each module. You can find this pattern in almost every file where we export classes or functions.
- Change of Responsibility pattern - Next design pattern we use is the Chain of Responsibility pattern, we predominantly used this in our Models folder, where we have a generalized Dao JavaScript class with all commonly used database requests. We also have models for each entity in our database. So the idea was to add very specific database requests in the entity models, and each time a more generalized request had to be made, the responsibility would be passed on the Dao model. (You can find this pattern in the Models folder)

3. Conclusion

- We're happy how the project turned out. It's not particularly feature heavy, but the main idea is there. Rather than implementing a lot of features, our focus was more on the coding side, where we tried to follow the coding standards, trying to think about which design patterns to use, picking our architectural pattern, code organization and so on.
- There was nothing majorly complicated to implement. These are just basic CRUD operations with authentication and authorization, however, deciding what's the best WAY to implement something (is it the most efficient way?, is this a security risk?, etc..) is the hardest part of any project I imagine, and that just takes a lot of experience. Another "issue" would maybe be writing regular HTML/CSS in. EJS. It's very limited in terms of a dynamic UI (For one example, we wanted to create a navigation bar where the regular user wouldn't see the "Create new review" option, only the editors of the page)

- The things that we would improve on later on: more features, db capacity upgrade, better and more dynamic ui