Web Technologies 2nd Coursework Report

Alexandros Anastasiou
40288545@napier.ac.uk
Edinburgh Napier University - Web Technologies (SET08101)

Contents

List of Figures

1 Introduction

The aim of this coursework is to design and implement a simple blog platform. This blog is running on a local host server, which means that it is not accessible from other computers and is not shared on the world wide web, so the only way to access it is to install Node and Mongo Database. The client of the blog presents the user interface giving the ability to the user to log in, by creating an account first through the registration page, to create, view, edit, delete posts and eventually to log out.

Just like any other blog platform, first you must create an account by filling up your details in a form and pass the validations, so you can create a unique account to access the blog. The user details are getting stored on Mongo Database. After, it redirects you to the login page where you must use the username and password to login to the blog. After successfully logging in, all the posts stored on the database will be shown. The buttons on the top right corner of the site are responsible for the navigation through the site. When logged in there are three buttons dashboard, new post and logout and when not logged in there are two login and register. The main features of this blog are viewing existing posts, create new posts, edit and delete the posts you created.

2 Software Design

After the initial experimentation with Node through the lab practical I realized that I was out of ideas on how to begin this. The first approach was to create a list of requirements according to the descriptor document and a very early navigation diagram which was transformed into a complicated after the completion of the application.

1. A client element to present a user interface:
    - Usable by at least one user.
    - Give the user the ability to:

    (a) register, log in and log out.

    (b) view existing blog post.

    (c) add a new blog post.

    (d) edit an existing blog post.

    (e) delete an existing blog post.
2. A server element to serve up the interface:
    - Persist data related to the blog (Database).
    - Provide a CRUD API to support the client in order to provide the blog's features to the users.
3. A certain level of security for the features that affect date that are stored on the database.

The creation of an early navigation diagram helped me keep in my mind how the pages are related to each other and helped me with redirecting after each function to the desired page. The initial diagram was very simple, and you can see the final on Figure 1 which was created after the completion of the application even though it might still be missing some details.

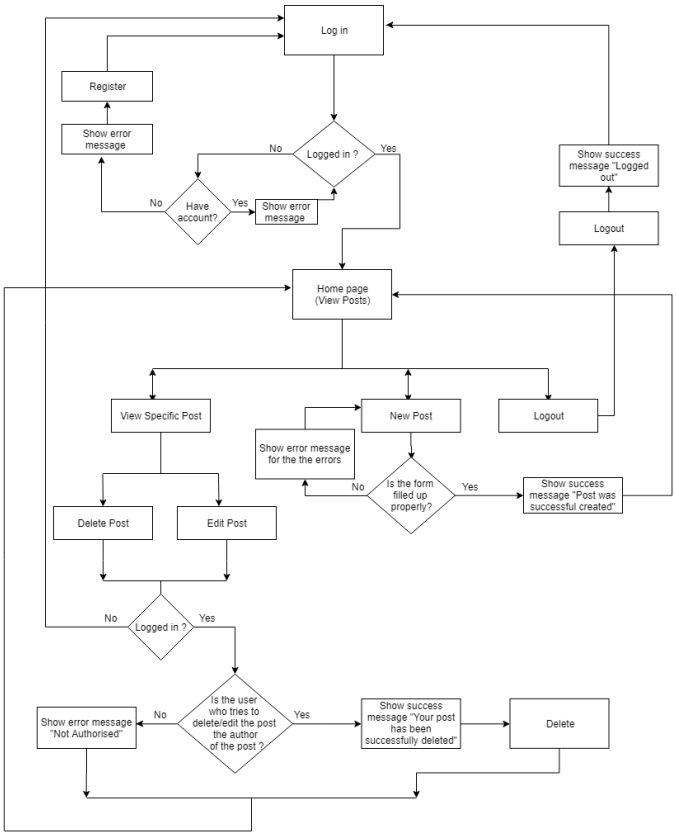

Figure 1: Navigation Diagram

3 Implementation

I started the implementation of the application by creating the register and login pages, because these two pages are important for security and the contained details were needed for further use. Also, it was a great way to test my database for storing data.

So, I created two form pages without any design just for debugging in the first place. After that I created a new model through node and with npm, installed a couple of packages such as passport, bcrypt.js, mongoose, express messages and validation and so on. After the creation of the new model and the forms I created a User Schema to store the data on my Mongo Database. You can see the schema on Figure 2. With passport package and some functions such as getUserbyId, getUserbyUsername etc it helped me authenticate with local authentication strategy that the user credentials are matching and give permission to log in. With bcrypt.js package the password which was passed on the register page was getting encrypted to a string and was getting stored on the database. This helped with securing the data that were getting stored on the database.

```
var mongoose = require('mongoose');
var bcrypt = require('bcryptjs');

//User Schema
var UserSchema = mongoose.Schema({
    username: {
        type: String,
        index: true,
        unique: true
    },
    password: {
        type: String
    },
    email: {
        type: String
    },
    name: {
        type: String

    }
});

var User = module.exports = mongoose.model('User', UserSchema);

```

Figure 2: User Schema

After getting the registration and log in page working Bootstrap framework was used for the design. I included Bootstrap for numerous reasons, and the most important was to save precious time from designing pages from scratch and focus more on the coding part of the project. Bootstrap is reliable providing pre-styled components such as alerts, buttons, dividers, navigation bars and so on. Apart from that, is very easy to use and it is a great approach for people who lack of creativeness and imagination.

Except from the pages stated before there are three more pages included. One page for viewing all the existing post's titles fetched from the database. One page for viewing a single post by clicking on the post title so the whole post can be seen and in addition to that there are two buttons below the post one for editing posts and one for deleting them. The last page missing is the one for creating posts with a simple design of a form just like the registration.

From Figure 3 it can be seen on the top right corner of the page there are two buttons since there is no user logged in now. Those two buttons are being swapped with dashboard, new post and logout when there is a user logged in as you can see on Figures 4,5,6. You can also see the error message below the header mentioning "missing credentials". This is an error message shown when there are no details entered on the login form and restricting you from seeing the rest of the pages.

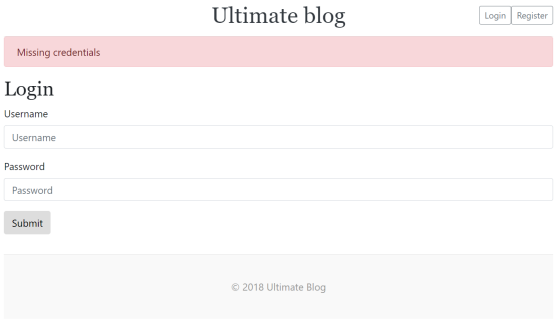

Figure 3: Login

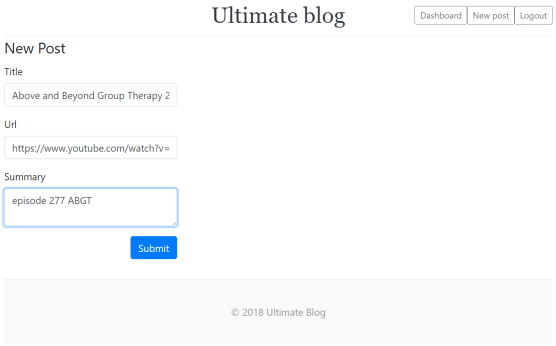

Figure 4: New Post

As it seen from Figure 4 the form for creating a new post is quite simple. There are validations included on this form as well since all the inputs are required to create a new post. The error messages are alerts like the one shown on figure 3. By clicking Submit for the new post it automatically records the time and day the post was created and the user who created the post from their user id which was logged in at that moment. In addition, after clicking Submit button it gets the user redirected to the dashboard where they can choose posts to see and showing a success message alert to notify them that the post was created. So, by clicking on a specific post to see it, you can see who created it and the exact time as it's shown on Figure 6.

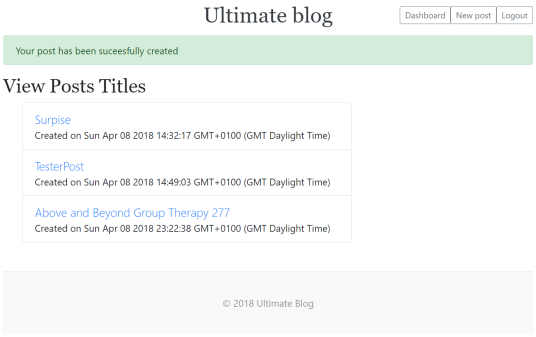

Figure 5: All posts

The dashboard page contains all the existing posts, as you can see on Figure5, that are stored on the database and they are getting fetched by their id. They are in order so the post created last is the one on the bottom of the page. It is basically the index page of the blog and the use of it is to give the ability to the user to choose which posts they want to see or edit. The design is user friendly and straight forward no explanations needed on how to use it and that was what I was aiming for.

Figure 6: Single Post

Figure 6 represents the single post page, where you can see the specific post which the title was clicked from the index page. As you can see the user's name who created the post is displayed as well as the time and the day when the post was created. The only one who can edit and delete the post is the user who created it, otherwise it give an error message "not authorized". This is a great feature since there could be many users viewing each post but it wouldn't be user friendly if every user could delete or edit any post they want. Again this is implemented inside the edit post route and within edit and delete functions where it checks whether the user currently logged in is the author of the post by their user id that is stored in the database.

4 Critical Evaluation

Overall, a solid level was reached by covering all the requirements of my list. There were requirements that were considered extras and I managed to carry them out such as the security features. In my opinion, the essential core features I decided to include to my application were necessary for a well implemented and reliable blog platform. I planed my time accordingly for each part of the application and finished the whole application before the deadline. Again, I don't think that my application has the perfect design, but it is as good to serve the purpose and be user friendly.

When it comes to comparison with other blog platforms, I would guess that there are many features missing and there are numerous improvements that can be made. Creating posts is quite simple and easy and not complicated like any other blogs. Viewing the posts doesn't require anything more than a mouse click since you are logged in. Again, editing and deleting is very basic just two buttons without a big deal. When it comes to access control, only by logging in give you the permission to view and manipulate posts, even if you enter the route on the URL you won't be able to view the posts if you are logged out, so it provides more security and reliability.

Get the application running I would say is the annoying part, but since the application is not deployed on the web then there is no other way accessing it and that's the most necessary improvement that I should be doing for further development. Talking about improvements, there is a non-ending list, but I can point out some vital features that were missed out. Firstly, when you register you can use the same username more than once, this is very confusing because you won't be able to log in because the usernames will be the same and the password wont match and because two users might try to use the same username and the same stands for the email. The easiest way to do this is to run a check if the username or the email already exists.

Another feature that It would be great to include is showing thumbnails for the URL links. When you want to create a new post sharing a link is required, that link might include a YouTube video, or a website and it would be nice to show a picture of what the link includes so you get the idea of what's behind that link before you press it.

Finally, the deleting a post might be easy and functional though it was the easiest way to implement it and there were better ways that provide more security and more control. For example, when you press delete post the post gets deleted without second thought. It would be great if a request was added just to give the user a second chance to think about it or maybe if they click it by mistake. I found a way to do this through jQuery, and ajax request but it was failing, and I was running out of time, so I used the easy way to make is functional.

5 Personal Evaluation

In a nutshell, from my experience with Node I can say that it is a very powerful and useful tool and I will be using it again for further projects. I planned my time accordingly for each features' difficulty. Of course, in every project there are hidden challenges you discover as you go through the process of implementation and so it happened with this one.

I have had some early problems with installing node because I have downloaded the wrong package through zip and I couldn't run global commands. I found the problem by doing further reading and research on the internet and from the help of a classmate.

Another challenge I faced, was with Mongo Database. It was necessary for me to have a database because I didn't want application to be used by only one user and I wanted to have persistence on the data for security reasons. I never used it before and it was the first time that I had to set it up. I just downloaded it and I was expecting it to work without any other actions, but I was mistaken. After I read the documentation from the site I realized that I had to run it through the command prompt. So, I had to run mongod.exe to build the database in first place and then through another command prompt run the command Mongo to get the database up and running. Through the second command prompt I could run commands and queries to make changes to the database as well.

It took me a while to get myself familiar with the way models and routes interact with app.js file and I was in trouble understanding how the redirections were working. I found a couple of tutorials and documentation from the book we were provided from the module website.

When I started working with Node it came to my understanding that I had to use pug templates for the views of my application. Later, I swapped to handlebars because it was closer to html and I had a better understanding of it. Both have the similar way of expressing stuff and do the same job but in my opinion, handlebars were looking better, and they are more organized.

Another challenge I faced was that even if the user should be logged in to see the posts and use the features provided, there was another way to access the pages, by typing the route on the URL manually. I didn't want this to happen, so I decided to add access control to the application. I have discovered a way to do this, by having a function to ensuring that the user is logged in to access all the features of the application. So even if you type in manually the route you get error message alert to log in first.

In brief, I learned a huge amount of stuff from this assignment and in my own point of view I feel I performed excellent on this. The application has very good functionality by offering the user all the appropriate features and persisting data and using a local strategy for login and register authentication. I also feel like I went way beyond the core learning for the module and expanded my knowledge around web development.

6 References

References

Aly Syed, B. (2018). Beginning Node.js. [ebook] Apress. Available at: https://link.springer.com/book/10.1007/978-1-4842-0187-9toc [Accessed 9 Apr. 2018].

Getbootstrap.com. (2018). Blog Template for Bootstrap. [online] Available at: https://getbootstrap.com/docs/4.0/examples/blog/ [Accessed 3 Apr. 2018].

GitHub. (2018). bradtraversy/loginapp. [online] Available at: https://github.com/bradtraversy/loginapp [Accessed 3 Apr. 2018].

Mark Otto, a. (2018). Bootstrap. [online] Getbootstrap.com. Available at: https://getbootstrap.com/ [Accessed 3 Apr. 2018].

MDN Web Docs. (2018). HTTP request methods. [online] Available at: https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods [Accessed 3 Apr. 2018].

Medium. (2018). Starting with Authentication (A tutorial with Node.js and MongoDB). [online] Available at: https://medium.com/of-all-things-tech-progress/starting-with-authentication-a-tutorial-with-node-js-and-mongodb-25d524ca0359 [Accessed 3 Apr. 2018].

npm. (2018). bcryptjs. [online] Available at: https://www.npmjs.com/package/bcryptjs [Accessed 3 Apr. 2018].

Passport.js. (2018). Documentation: Authenticate. [online] Available at: http://www.passportjs.org/docs/authenticate/ [Accessed 3 Apr. 2018].

References GitHub. (2018). bradtraversy/nodekb. [online] Available at: https://github.com/bradtraversy/nodekb [Accessed 8 Apr. 2018].

GitHub. (2018). ctavan/express-validator. [online] Available at: https://github.com/ctavan/express-validator [Accessed 8 Apr. 2018].

GitHub. (2018). expressjs/express-messages. [online] Available at: https://github.com/expressjs/express-messages [Accessed 8 Apr. 2018].

Handlebarsjs.com. (2018). Handlebars.js: Minimal Templating on Steroids. [online] Available at: https://handlebarsjs.com/ [Accessed 8 Apr. 2018].

Makeschool.com. (2018). Reddit Clone in Node.js (2020) — Create a Post. [online] Available at: https://www.makeschool.com/online-courses/tutorials/reddit-clone-in-node-js/create-a-post [Accessed 8 Apr. 2018].

npm. (2018). express-validation. [online] Available at: https://www.npmjs.com/package/express-validation [Accessed 8 Apr. 2018].