Coursework Report

Tamas Torok
40342265@live.napier.ac.uk
Edinburgh Napier University - Web Technologies (SET08101)

1 Introduction

This is a report for the second coursework in the web tech which aims to introduce and tell the steps taken in order to create this website. The website is a simple blog platform with functionality like authentication, creating, reading, updating and deleting (CRUD) blogs with several topics like art, music, technology, travel and design to choose from, as well as some added CRUD functionality for comments. The scope of this assignment was to merge our knowledge between the client and the server side, front-end and back-end into a functional platform.

In order to be able to achieve these functionality some background reading was necessary like several Medium articles[1] to understand better how the newer JavaScript should be implemented,as well as to learn more about back-end, Typmanus [2] was a great resource to learn more about CSS and Muzli[3] was helpful in getting inspired by the recent trends in the web.

2 Formulating my challenge(s)

Setting up a goal in the beginning of the project was crucial since I knew that I want to learn and get out as much as I can from this, so my first step was to research the possibilities and trends of what you can make during our time constraint. Another challenge was to set up a functional naming system for my CSS since I knew that it usually gets over crowded and messy when I'm not planning ahead with it. Also I was interested in how could I implement some newer stuff that I learned about JavaScript and writing according to the new ECMA standards(e.g. instead of var write let or const).

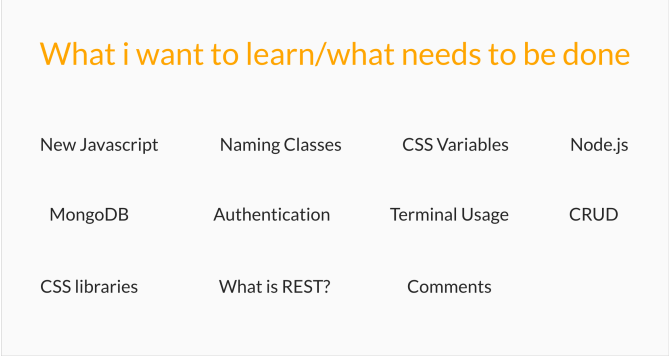

Figure 1: **Ideation phase(Brainstorming)**

In the end I came up with some basic sketches as far as the front-end goes, and I made a basic map on how to navigate around the website. The website will have an authorization function, the user should be able to create a blog with various interchangeable content like images, subtitles, and paragraphs, and also being able edit that blog adding more content if they will. As far as the comments go the user should be able to make a comment and if they are the owner of the comment, they should be able to delete or edit that.

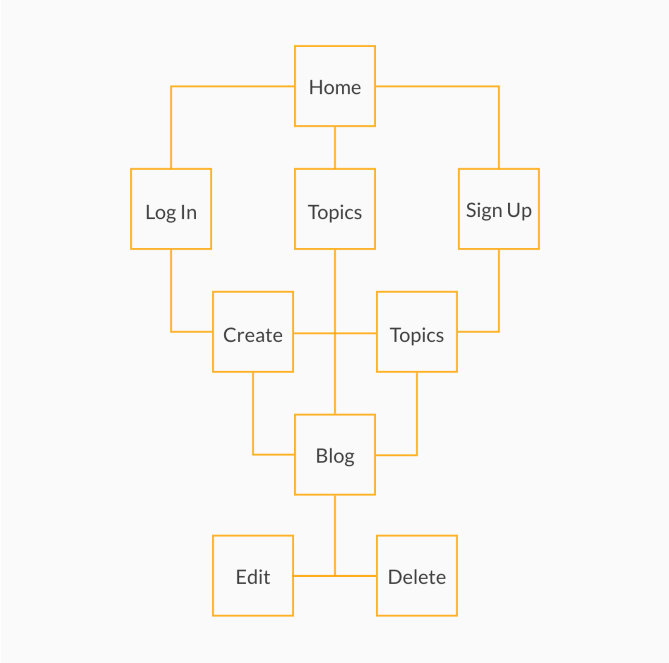

Figure 2: **Navigation Map**

3 Software Design

The next step in this process was to plan how everything will be connected, after reading up about RESTful design and getting familiar with the back-end world i began writing up the routes for the different pages on the website which wasn't that difficult. My basic routes are the index which is the home page, afterwards there is a sign up, log in, the topics are a dynamic route with "/:topic" which changes based on which topic is the user currently on, for creating a blog is "/:topic/blog/new", accessing a single blog is "/:topic/:blogid", editing the blog is "/:topic/:blogid/edit". These are the main ones and I just worked forward with this approach.

Home	/	Show the home page
Topics	/:topic	Show blogs from a single topic
Show	/:topic/:blogid	Show a single blog from a certain topic
New	/:topic/blog/new	Show the new blogs form
Create	/:topic	Create a new blog and redirect to the topics page
Edit	/:topic/:blogid/edit	Show the edit blogs form
Update	/:topic/:blogid	Update a certain blog and redirect to the blog
Delete	/:topic/:blogid	Delete a certain blog and redirect to the topics page

Figure 3: **RESTful Routes**

3.1 Front-end

When the routes were figured out I started to make wire-frames and mockups in Adobe XD which initially looked quite different from the end-outcome, because in the end I ended up using the Semantic UI CSS library to speed up the designing part.

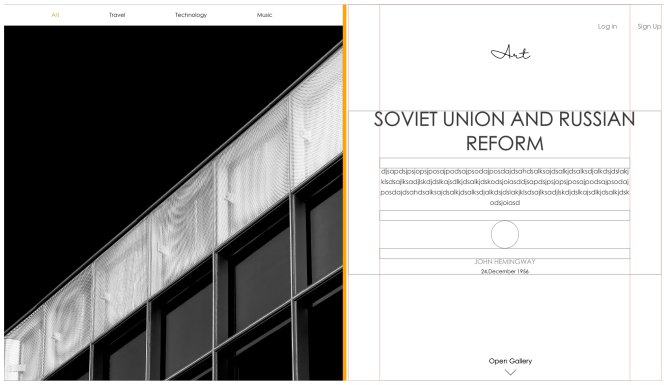

Figure 4: **Web design**

3.2 Other elements

For the typography I chose several fonts from Google Fonts including 'Dawining of a New Day' which is a cursive font, which was used as the logotype because the blog is called 'Readability' and the goal of the website is to create unique content, so I thought this cursive font can express that. I also used the 'Libre Baskerville' serif font for the body text, 'Nunito Sans' and 'Montserrat' for small elements for readability. The highlight colour of the website is FFA500, the orange named color used by the default CSS.

4 Implementation

4.1 EJS

```
<div class="ui comments">
    <h1 class="ui diving header">Comments</h1>
    <% blog.comments.forEach(function(comment){ %>
        <div class="comment">
            <a class="avatar">
            <img src="<%=comment.author.imageurl%>" class="blog__container__right__avatar__image">
        </a>
            <div class="content">
    <p class="author">
        <%=comment.author.username%>
    </p>
    <div class="text">
        <%=comment.text%>
    </div>
</div>
```

Figure 5: **Embedded JavaScript**

During the implementation the sketches and mockups were used as a reference to create the scructure of the website. EJS stands for embedded JavaScript and using it was much easier to create dynamic content for the blog website since writing plain HTML and creating external javascript would've been more difficult to implement.

4.2 CSS

I decided to use the BEM conventions for naming my CSS since I thought that it would make my naming more efficient and readable, hovewer I did not stay with the conventions throughout the website since I implemented the Semantic UI library.

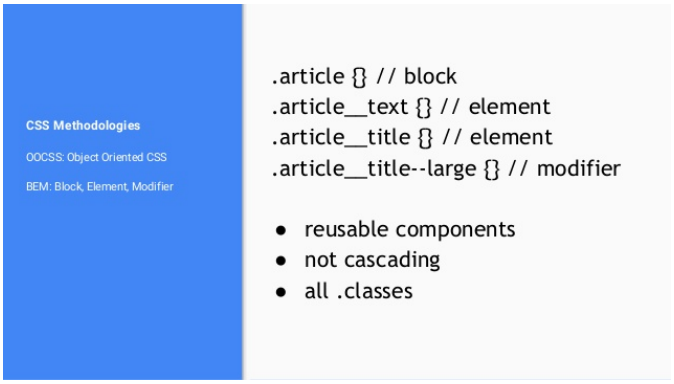

Figure 6: **CSS BEM conventions**

4.3 Javascript

Separate JavaScript was used in order to create a loader on the website, access the Semantic UI features and add extra content form to the create page of the website.

4.4 Back-end

The Beginner Udemy[4] course was really helpful in understanding the back-end.

The app.js contains all the required packages, the passport and passport local mongoose which takes care of authentication, also it includes the paths to the routes used. Since there were more routes following "/:topic", I decided to separate the routes into to files, one of them was used for everything which excluded the topics, like the sign up, log in, and the

homepage route, the rest of it would go to the other route file.

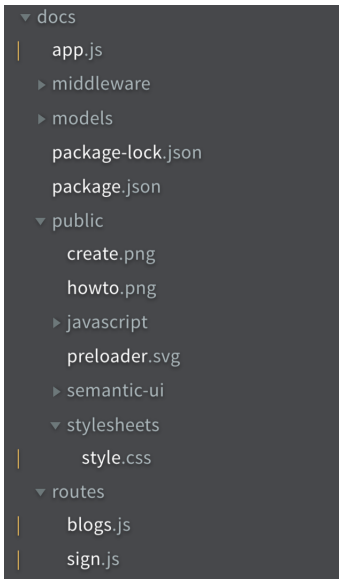

Figure 7: **File Tree leading to routes**

The blogs.js contains all the routes necessary to show the blogs and edit them. If a certain route is called the ejs corresponding to it is called and filled with the necessary information taken from the MongoDB database with certain functions like blogs.find(), blogs.findByID() or blogs.findByIdAndUpdate().

```
router.get("/:id/edit",middleware.checkBlogOwnership, (req, res)=> {
    Blog.findById(req.params.id, (err, foundBlog)=> {
        res.render("blogs/blogedit", {
            blog: foundBlog,
            page: req.baseUrl.substr(1)
        });
    });
});
```

Figure 8: **A MongoDB command to find a blog and edit it**

When a new blog would be created the MongoDB database needs to know where to put the information, for that there's a models folder including models for the users singed up, blogs created and comments which all relate to eachother. Using this instead of MySQL makes our jobs easier to created related content to eachother, e.g. the user can have multiple blogs and multiple comments, but a blog and comment can only have one user assigned to them.

```
//Schema setup
var blogSchema = new mongoose.Schema({
    x: Number,
    topic: String,
    title: String,
    mainimage: String,
    shortdescription: String,
    blogcontent: {},
    author: {
        id: {
            type: mongoose.Schema.Types.ObjectId,
            ref: "User"
        },
        fullname:String,
        username: String,
        imageurl: String
    },
    comments: [
        {
            type: mongoose.Schema.Types.ObjectId,
            ref: "Comment"
        }
    ],


});

module.exports = mongoose.model("Blog", blogSchema);
```

Figure 9: **Model for blogs and their related content**

5 Critical Evaluation

Looking back at the necessary requirements, to create a simple blog platform where the blogs can be created, edited, updated and deleted, this website meets all the necessary requirements, and adding some more functionality like comments and authentication.

The requirements state that we need to create a 'simple blog platform' which isn't specific enough for me to criticize the complexity of this website, but I think it is simple enough to be considered a simple blog platform but complex enough to have some real uses and functionality.

By looking and comparing this blogplatform to other blogplatforms I knew from the beginning that in terms of content uniqueness there will be an issue since the users can only choose from 3 different types of content on their blogs like text, subtitle and images, where the text has a fixed font size and the user can't choose other stylistic customisations. Nevertheless compared to the more basic blog layouts I think that it has a more up do date look, it is made for a niche target audience which in general is more drawn towards this style.

The website is never really done and as the person spending hours, days and weeks with it, I want to leave it unfinished, but since the time constraint and the fact that I have also other modules projects to do I need to. I would've liked to add another type of content to be chosen, a gallery page which would focus more on images, I would also redesign the elements used with Semantic UI, because even though they make the workflow faster, they also tend to look rather unoriginal and plain. Adding some CSS animations, a lazy loading and other stylistic elements would improve the user experience in my opinion.

6 Personal Evaluation

The research before actually creating anything in code was really useful, I found a lot of resources from which to learn also in the future, I already feel like I learned a lot, but I also realised that I need to do a lot more background reading. I feel like my workflow has improved, I'm more productive with my time working and I see more and more how planning ahead and doing actual research into target audience, can help with making a website just right for the customer in the future.

There were definitely a lot of challenges that I needed to overcome, like how to use the Terminal, wrapping my head around the routes and how everything is connected, at first I felt overwhelmed, but in the end I think that the pressure helped me push myself more. Overall I'm quite proud of the website, but I know with the knowledge that I have now I'd do a better job.

6.1 Appendix

All the photos were taken from Unsplash with CC lincense on them - http://unsplash.com

Semantic UI CSS library was used in the making of this website - http://semantic-ui.com

References

[1] T. Anwar, "Teaching myself how to node in 2018,"

[2] A. Galante', "Theming with css variables,"

[3] Muzli, "Ui interaction of the week,"

[4] C. Steele, "The web developer bootcamp,"