Coursework Report

Piotr Handkowski
09005378@live.napier.ac.uk
Edinburgh Napier University - Web Technologies (SET08101)

1 Introduction

The aim of this report is to describe the process of creating a blogging platform Bloggeroo. The application has most of the features that are provided by other blogging sites, i.e. it allows you to add a post, edit and delete it. The user can also leave a comment on a post, edit or delete it. To perform those actions the user needs to register with a username and a password. Most of the skills required to build this application I have acquired from a Udemy course [1] titled "The Web Developer Bootcamp". To bridge any gaps in my knowledge I used Google to find answers to most of the issues I came across. Those resources allowed me to build a robust application which I deployed using mLab and Heroku. Bloggeroo can by accessed at: https://agile-plains-17695.herokuapp.com.

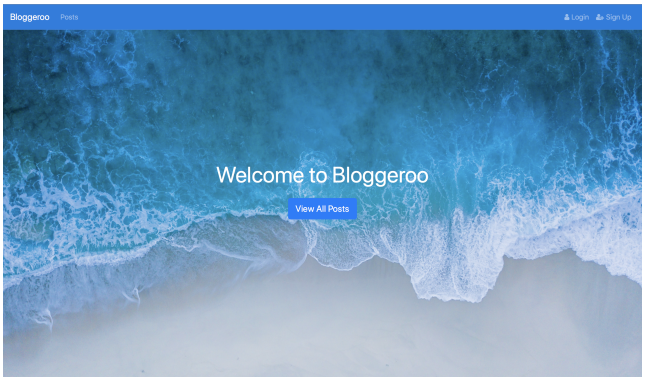

Figure 1: Landing Page

2 Software Design

The design process started with the identification of the features that the application would provide. After deciding that I would like to implement the features mentioned earlier I sketched a rough design of the pages that had to be implemented. Having implemented them I added one more pages where the user can see all his/her posts. At this stage I have not decided on any theme or styling and focused solely on the functionality.

With the initial design I could identify what data will have to be stored to make the application perform the designed features. I have then grouped them into three categories: user, post and comment. At a later stage I have added other attributes such as date created for posts and comments and image URL for the post. With this information I have then

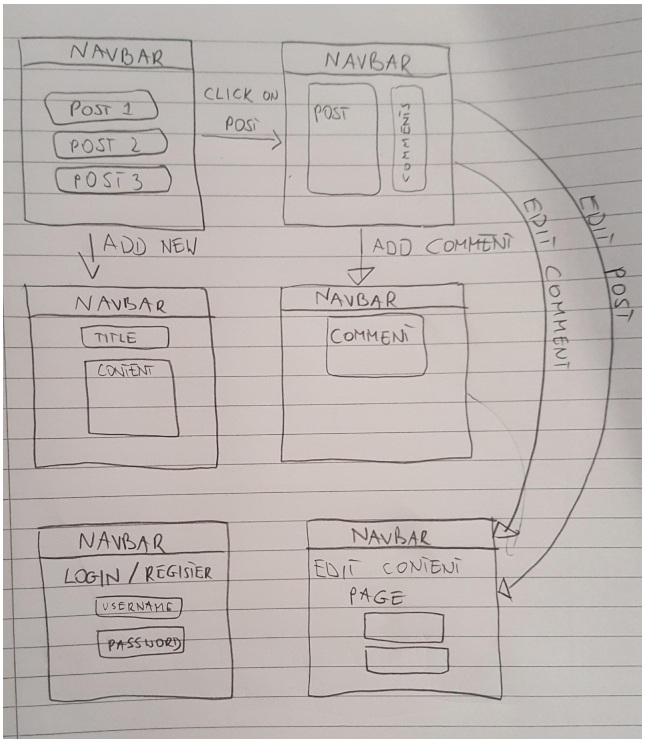

Figure 2: Pages design

designed a small database structure that shows how the information is interrelated, which can be seen in figure 3.

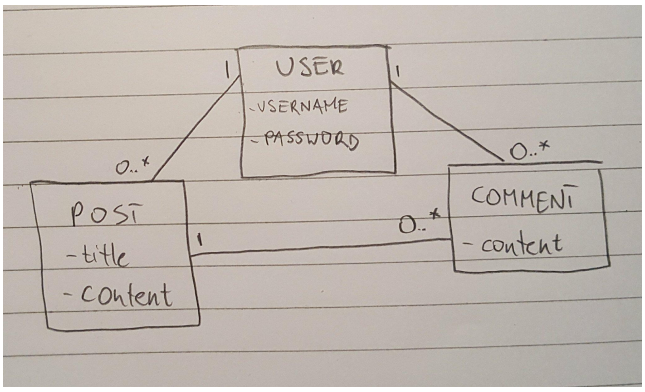

Figure 3: Data Structure

Having decided on the pages that the application would have and the data structure to make it persistent I could start planning the required routes. The Udemy course turned out to be very helpful at this stage, as it has a section with many videos explaining how to design the routes in a restful convention. Completing all the online exercises taught me how to plan the routs for Bloggeroo. All the routes can be

seen in Figure 4. With the initial design I could start working on coding the application.

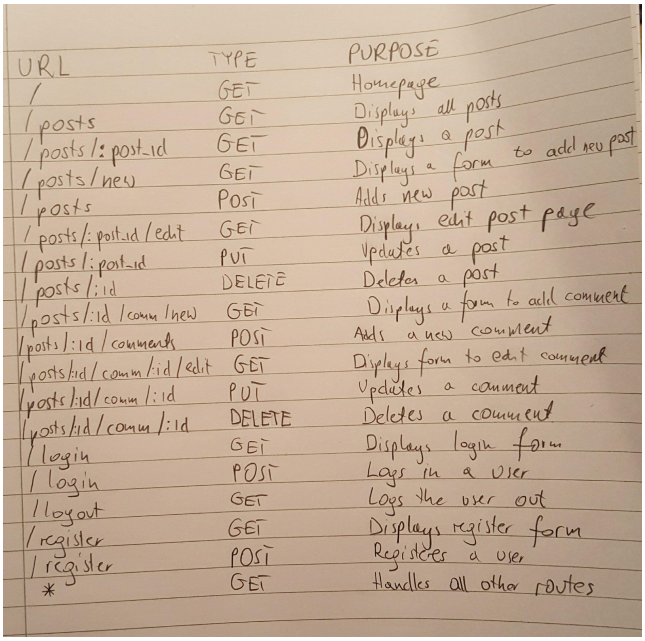

Figure 4: Restful Routes

3 Application Implementation

The implementation started with a basic set up of the application which included the installation of express [2] module. I have also decided on using ejs [3] as the view engine for the application. Firstly, because it allows to render pages dynamically, where you can pass data to the template and use it to create the page content. Additionally, I found it is easy to read and maintain, as it looks like normal HTML with embedded JavaScript. Another library which I decided to use is body-parser [4]. I used this middleware to handle the HTTP POST data, as it extracts the body portion of an incoming request and makes it available in req.body. Since HTML 5 does not support PUT and DELETE methods I have also installed a method-override library which allowed me to use POST method and override it to act as PUT, or DELETE. Before starting the work on the routs I set up data storage. For this purpose I used mongodb and mongoose [5] library. Mongodb seemed like a natural choice for this application, as data is stored in JSON documents and cam be used straight away after retrieving it from the database. I also installed mongoose, as I had clearly defined data structure for the blog and I could create a schema. Additionally, it allows you to create a model abstraction that can be used as an object, which is easier to work with than just data. This made it very easy to add data to the database, as well as retrieve it and dynamically use it in the app. Having installed the mentioned libraries, and creating data storage, I could start implementing the routes. According to the design each page was supposed to contain a navbar, therefore instead of coding it in each file I implemented it as partials (header and footer) and only coding the body of each template. I added a cdn link to bootstrap in the header, as I knew I would be using it to style the application, and scripts to jquery, bootstrap and fontawesome in the footer. Those libraries were necessary to make full use of bootstrap, i.e. responsive navbar which adopts to the page width. On smaller screens some of the options are hidden in a hamburger menu which expands when clicked on. To implement the login feature I used 3 libraries: passport [6], passport-local [7] and passport-local-mongoose [8]. I decided on using those libraries, as authentication on it's own is a topic big enough for a separate project. Those modules make it easy to manage sessions, create hashed password and salt for the users which in turn makes the platform more secure.

After implementing all the routes and making sure they work properly I started working on the styling. First, I found a background photo of sea shore by Nattu Adnan on Unsplash (https://unsplash.com/photos/Ai2TRdvI6gM). I matched the navbar colour using "Color Grab" application, giving it light blue colour. The site name in the navbar is white and all other links are in very light blue. I decided that I will go for simple design and used the default bootstrap style. Most of the buttons are blue with white text. The posts page contains a jumbotron with some information and the posts are displayed as cards. Each card contains the post's title, preview of the content and a photo if the user provide an URL address. If no photo URL is specified then longer preview is displayed to make the cards more or less the same height. There is also a "Read more" button. The cards are displayed responsively. 3 cards in a row on large screen then 2 and 1 on medium and small screen respectively.

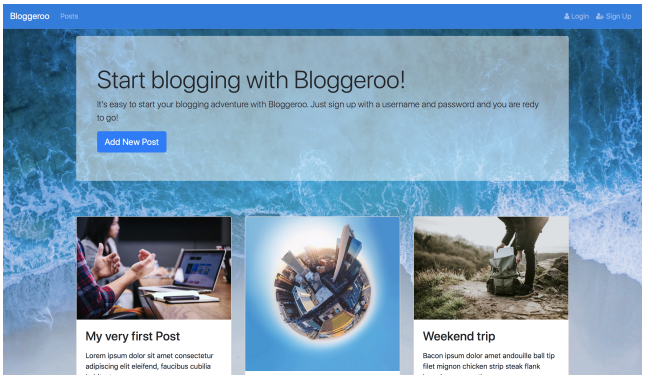

Figure 5: Posts Page

From the posts page the user can either add a new post (the user needs to be logged in to do this), or view any of the posts by clicking on the "Read more" button on the post. Clicking on the "Add New Post" button takes the user to a form where a new post can be added.

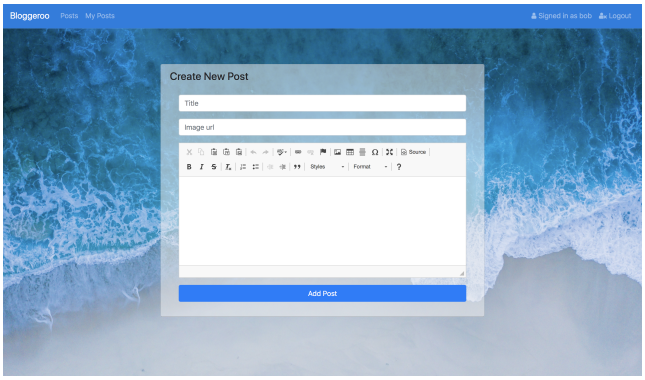

Figure 6: Add New Post Form

Instead of using simple textarea for the post content I used CKEditor to allow the user to style the post. The editor automatically adds the HTML tags, for example the li, strong, or paragraph, that are stored along the text in the database and are rendered when the data is retrieved. The editor is also used for comments input. The user can add an image URL, but this is not necessary to create a post.

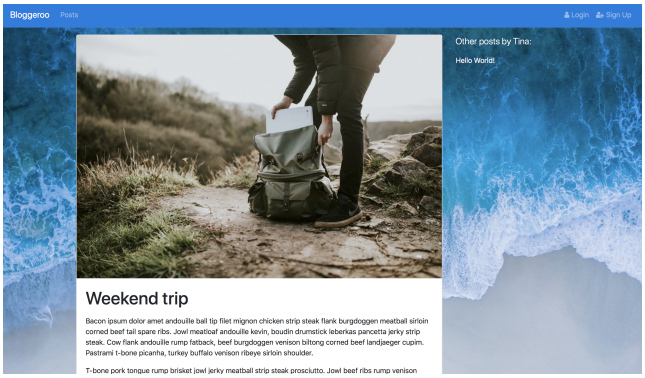

Figure 7: View Post Page

When the "Read more" button on the post preview is clicked the user is taken to a page where the full post is displayed (figure 7). Here logged in users can leave a comment.

The comment section is situated below the post where comments are displayed along with the author username and the date they were written. Additionally, links to other posts of the same user are provided on the right-hand side. On smaller screens the list of other post is displayed below the comments section.

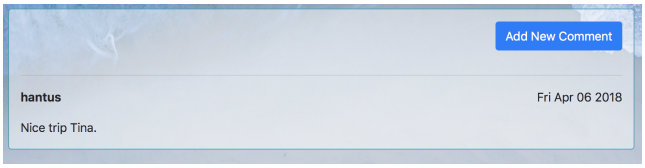

Figure 8: Comment Section

The user who created a particular post or comment is the only one who can edit, or delete it. Edit and delete buttons appear at the bottom of a post or comment when the owner is logged in.
Anybody can use the platform by registering with a username and a password, and then use those details to log in.

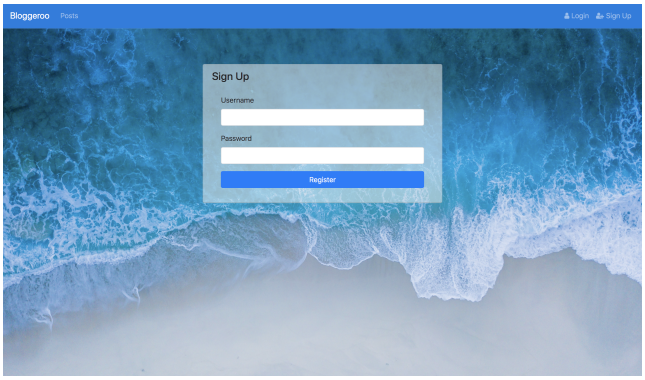

Figure 9: Registration Form

When a user logs in the "Login" and "Sign Up" links on the navbar turn into "Signout" link and "Signed in as + username" message. Furthermore, a new link appears on the navbar "My Posts" where the user sees only their posts. This makes it easy for them to manage their blog. This pages is styled in the same way as the one where all posts are displayed.

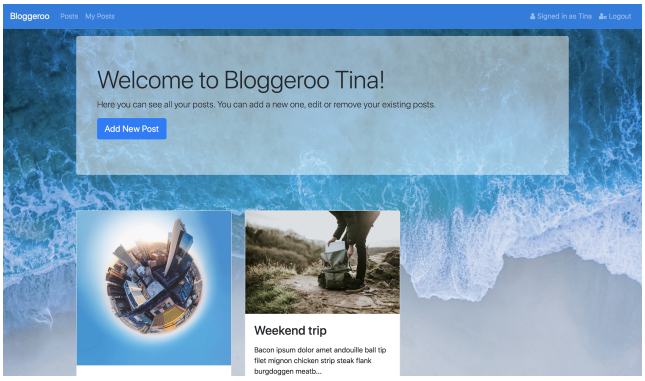

Figure 10: User's Page

To inform the user that they need to login/ register, or that their post or comment has been added/updated/deleted I used connect-flash [9] library. I decided to use it, as it makes the messages appear only once and when the page is refreshed the message disappears. They are also used to displays errors coming from the database, for example when a user tries to register with a username that is already used. Error messages are displayed on red background while confirmation and information messages are displayed on blue background.

Figure 11: Flash Message

The application has been deployed on Heroku [10] using the free plan. In order to do this the database had to be moved from my computer to an online server. For this purpose I used mLab [11] which also offers free accounts with limited space. Bloggeroo memory requirements are small enough to use those limited free resources.

4 Critical Evaluation

Bloggeroo meets all the requirements specified in the coursework description. It allows a user to add a blog post which can also be edited, or deleted by its author. All posts are stored in an external database allowing data persistence. Apart from the core features additional ones have been implemented, i.e. users log in using username and password and are able to leave comments, passwords are encrypted giving extra security. The platform has also been deployed making it a fully operational application which can be used by anybody.
Compared to other popular blogging platforms Bloggeroo seams quite simple. It only allows adding photos by specifying URL links, you cannot follow users and be notified about their recently added posts. It doesn't let you like or rate a post. However, the aim of this platform is not to compete

with other well established applications, but to develop basic skills in web development.
To make the application good enough to attract users many improvements would need to be made. The feature to follow users and create groups would have to be added. Search feature would also be needed so the users could find other users, or search for posts in a certain category. The feature to like a post, or rate it could also be added and then the most liked/rated posts would show first in the feed. Additionally, Facebook authorisation could be added, so the users could log in using Facebook credentials. Another important feature could be the ability to upload photos or videos.

5 Personal Evaluation

This project was a great source of learning a lot of new things and practising my existing stills. Right from the beginning I had to overcome my desire to code straight away and focus on the planning part first. It also made me appreciate how important and helpful the design phase is, as I did not have to change a major part of my code, or start from scratch which was normally a part of my past projects. In terms of technologies I have consolidated my previous knowledge of node.js applications and learned a lot of new things. The most important thing I will remember is the restful routing convention. I knew about it in the past, but I didn't fully understand it. I also learned new libraries which I didn't use before, especially the passport libraries I used for authentication. This was also the first dynamic application that I deployed using Heroku and mLab. It was also a good opportunity for practising command line comments, as I used them for creating files and directories, moving around the project files, testing data in mongodb, as well as using git and pushing the project to Heroku. The biggest challenge was trying to figure out what some of the error messages I was getting meant. Not always they are meaningful, but I am pretty good at looking for solutions online, or finding what could cause the error, as I frequently test the app to see if it works with the new code. When I was starting the project I had this feeling that I do not like developing websites and I would not take advanced web technologies next year. I think this was mainly because my poor understanding of how to design the routes. After completing this project I feel more positive about web technologies. I feel that I have created a good piece of software and I cannot wait to learn more next semester.

6 References

References

[1] C.Steele. "The Web Developer Bootcamp". Available at: https://www.udemy.com/the-web-developer-bootcamp/learn/v4/overview

[2] StrongLoop, IBM, "Express". Available at: http://expressjs.com

[3] M. Eernisse, "Ejs". Available at: http://ejs.co

[4] J. Ong, D. Willson, "Body-parser". Available at: https://www.npmjs.com/package/body-parser

[5] Learn Boost, "Mongoose". Available at: https://github.com/Automattic/mongoose

[6] J. Hanson, "Passport". Available at: http://www.passportjs.org

[7] J. Hanson, "Passport-local". Available at: https://github.com/jaredhanson/passport-local

[8] "Passport-local-mongoose". Available at: https://www.npmjs.com/package/passport-local-mongoose-email

[9] J. Hanson, "Connect-flash". Available at: https://github.com/jaredhanson/passport-local

[10] Salesforce, "Heroku". Available at: https://id.heroku.com

[11] ObjectLabs Corporation, "mLab". Available at: https://mlab.com