Web Tech Coursework 2 Report

Kieran Burns
40272382@napier.ac.uk
Edinburgh Napier University - Web Tech (SET08101)

1 Introduction

The purpose of this report is to detail the development process of my simple blog website which I named "Fun Blog!", following the stages going from planning through to completion and post development reflection. The general plan going in to the project was to make a simple and easy to use page where users can log in to add, edit and delete posts under their unique user names using a CRUD (create, read, update, delete) API using HTML, CSS and JavaScript for the client side and Node JS for the local server elements.

My initial hopes were for the site to be based around using only a single web page and using JavaScript in unison with Node to handle dynamic elements within the page such as displaying all posts, sorting posts and changing the display of posts made by a user so they can be edited or deleted.

2 Design

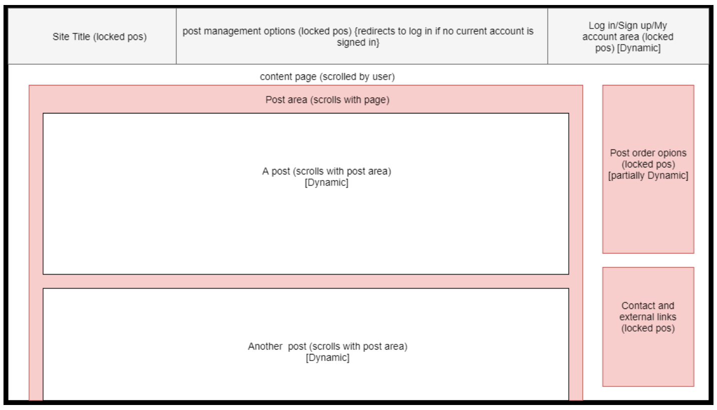
To begin the development process I first had to break down the problem and figure out how each element/section of the site would be made. This was broken down in to the following sections:

- User Interface (UI) - HTML and CSS elements for the site.
- Post Submit and Retrieve - JavaScript and Node with Express.
- User Log In and Sign Up System - JavaScript and Node with Express.
- Post Edit and Delete - JavaScript and Node with Express.

2.1 User Interface

The first thing I began to think about when it came to the planning stages of the project was how I would like the user interface to look. After doing some research on the design of popular social media and blog sites I decided that I would use a hot bar at the top of the screen to host some controls such as navigation options and an account area where a user can log in, sign up, log out and see their logged in status as a whole. See *figure 1* for the initial wireframe diagram I created during this stage. I decided that once I began working on HTML elements to build this page that I would (due to my lack of experience with dynamic websites) build each major feature of the site as separate HTML pages then use JavaScript to merge elements further through the project to unify each of the features into one page.

figure 1.


Site Title (locked pos)
post management options (locked pos) {redirects to log in if no current account is signed in}
Log in/Sign up/My account area (locked pos) [Dynamic]
content page (scrolled by user)
Post area (scrolls with page)
A post (scrolls with post area) [Dynamic]
Another post (scrolls with post area) [Dynamic]
Post order opions (locked pos) [partially Dynamic]
Contact and external links (locked pos)

As can be observed in *figure 1* the site page was planned to consist of four main elements placed onto a canvas, each element being made with different functions and priority levels in mind. The top bar was planned to be a fixed position element that would be the top layer on the site in regards to what gets rendered over the top of what so that the lower elements would scroll under the bar rather than over the top of it. The large central section was planned to be where the posts were hosted and would scroll as part of the page. The two rightmost elements would each also be fixed position elements, the top element providing sorting options for the order in which the central boxes posts would be displayed whilst the bottom right element would contain some external links and contact details to mimic those used in most popular modern sites.

2.2 Post Submitting and Retrieval

For the options to submit and retrieve posts I planned use a dynamic HTML element powered by some JavaScript to bring up a pop up menu where the user can enter a title and some post content to post to the server. When retrieving posts from the server (which would when the page is loaded as well as each time a change is made to the database where the posts would be stored) another set of JavaScript functions would be used to update the box on the page.

During this stage of the planning I had to do a moderate amount of research to determine how I would send information between the site and the server (connected to an external database). After looking at a few different options I decided I would learn how to use AngularJS as it is a widely used and accepted framework that can be used to thoroughly simplify the connection stage between a site and a server in regards to passing information and calling functions among various other options such as having dynamic buttons on a site page. I also preformed a moderate amount of research around JQuery however I decided to stick with angular for my first attempt at developing the site and decided I would later return to JQuery if I was to have too much trouble using AngularJS.

2.3 User System

The site was planned on having a users system where a user could create a new account, log in and create posts from that account, where each new post created by that user would be marked with the users unique user name. Each of the users would be stored within a separate data table and would be submitted and retrieved through similar methods to the way posts were planned on being moved between the server and site. If the user name attached to a post matched the user name of the logged in user the some extra options would appear within that posts box within the post area which would allow the user to amend their post or delete the post. These amend and delete options would also be available through the use of an Admin account that would be created in the users data table and would be hard coded within the sites script.

2.4 Post Editing and Deletion

As mentioned prior, if a user is logged in and a post that they created is on display or if they are an Admin then the options to Edit or Delete a post will be available. These options would be similar in nature to the Submitting and Retrieval of posts however were likely to be more programmatically complex. These functions would be made function with the help of AngularJS and would connect to the NodeJS server just as their sister functions were planned to.

3 Implementation

With a general idea of how I intended to implement my site in mind I moved on to the implementation stage of the project. I began this stage by creating a HTML page with CSS elements to function as the top bar and two side blocks I had planned earlier, giving each intractable element within these areas their own ':hover' CSS function to make the sites elements react to the user to make the site experience more visually appealing and interactive. I then created a box that scales to the size of its contained content to use for the central blog posts area and another scaling box type to be contained within the first to display each individual post.

Once I had made my first page and the majority of the pages CSS elements I created the first of my new temporary pages to be used to add data to the database before being integrated into the first page. This page contained two simple input boxes and a submit button.

Once the submit and display areas were made I moved on to the implementation of the script to add function to the areas. I began by creating the Node 'get' and 'post' requests that I would need for the operations. Once I had created basic versions of each of those I added the AngularJS features within my client side script to allow connections to the server. This was done through the use of the AngularJS '$http' requests.

After creating a working submit and display system I moved back to the HTML and CSS where I decided it was time to merge each of my elements into a single page. I began this by creating a model box in CSS to allow for the user to enter a new post on the main page. At this stage I also preemptively created model boxes for the sign up, log in and amend post options I planned to work on. I also created a new part to the new post server elements where the post contains a user name along side the title and content values each already contained to allow a original poster to be displayed on each post.

After alterations to the post format and HTML and CSS elements I began work on my account system to be used across the website. This process began by creating a second table within the site database to store account user names and passwords. I then created server side elements to allow the new data table to be accessed by the site. Lastly I created post and get requests similar to those of the new post and view post options for blog posts to be used to create new users and access existing accounts. Once the account system was functioning I created a updated version of the post display function where if there is a user logged in and that user matches the user name associated with a displayed post some extra options would appear within that post at the right hand side to allow a user to amend or edit their own posts. An Admin account was also created who has permissions to bypass the account based safety to allow any post to be edited or deleted from that account.

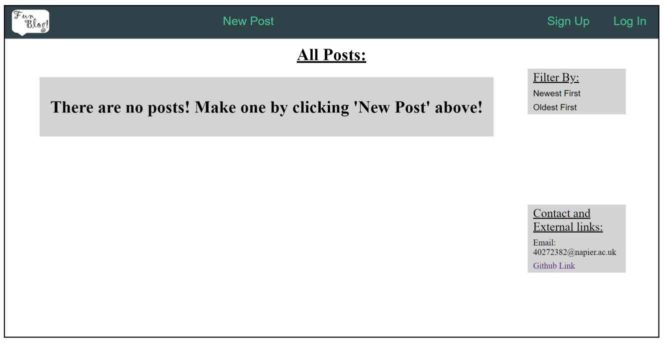
Once the buttons to allow post controlling were added I began work on adding functionality to these options. The delete function was a case of making a simple delete function to remove a post from the database given each posts unique ID tag. The amend option was slightly more complicated than the delete option, rather than taking the easy route and deleting a post then adding an updated version to the database I opted to actually modify an existing database element. This function involved using a different version of the get method to acquire an existing post by ID, using a HTML element to display the details of the acquired post and input an updated title and/or post content then used a put method with the parameters of a post ID and a JSON containing a title and post to be updated. When the put request was called with the given parameters it would access the post assigned to the ID then modify the title and content columns associated with that ID within the database.

One a full CRUD cycle had been established I moved on to add some validation to ensure that nothing could go wrong in that regard, this included adding a duplicate checker to new users to ensure no two users could have the exact same name, a password check to new users to ensure they do not enter a blank password and I added incorrect log in feedback on the log in window to allow users to be informed if they have attempted a log in and were declined access to the account.

The full CRUD cycle of the web site can be observed within *figure 2*.

figure 2.

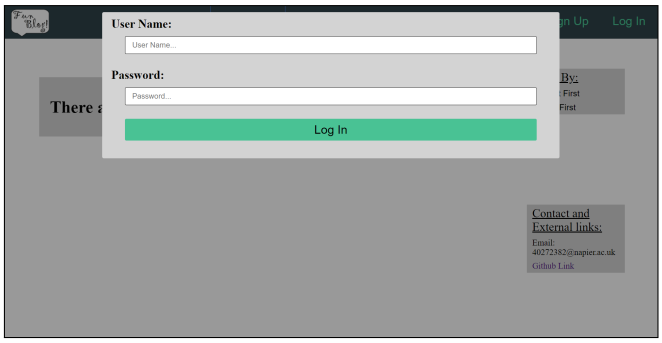
When a user first opens the site and there have been no prior users to add posts to the site, the following screen will be shown:



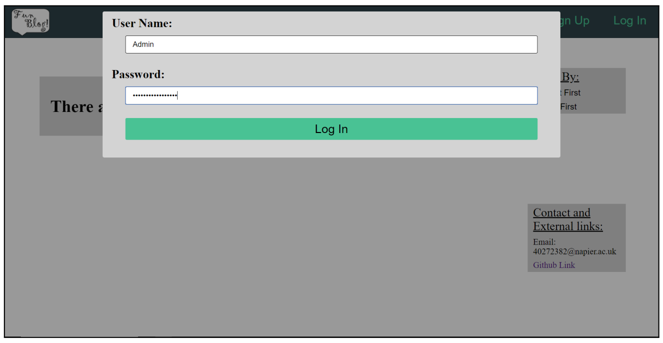
There are no posts! Make one by clicking 'New Post' above!

Email: 40272382@napier.ac.uk

One on the site, if a user clicks either 'Log In' or 'New Post' without being logged in the following box shall open on the page:

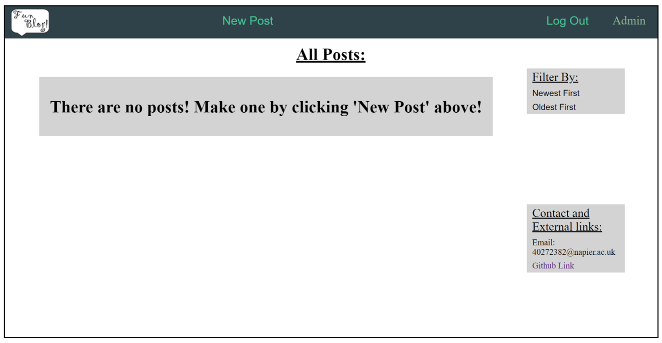


Prior to user interaction, a single account exists within the sites account database, that account being the Admin account, this account has access to the permissions granted to every single account to allow any posts a user makes to be edited or removed by an outside source if necessary, the following image shows the Admin account being logged in:



User Name: Admin

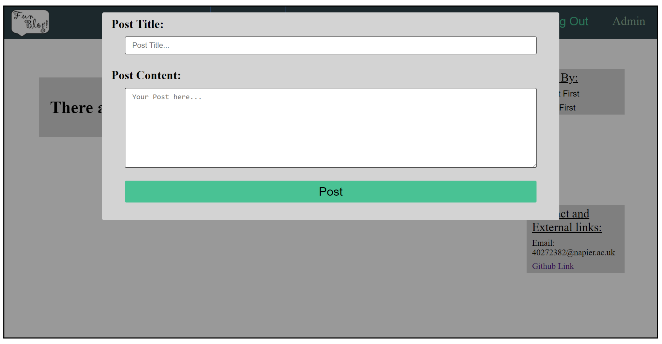
Once a user has logged in the top bar will change as is shown below:



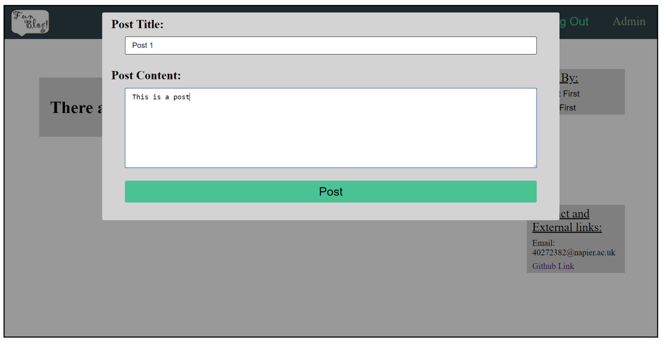
There are no posts! Make one by clicking 'New Post' above!

Email: 40272382@napier.ac.uk

Once logged in, a user can create a new post by clicking 'New Post' on the top bar which will reveal the following box:

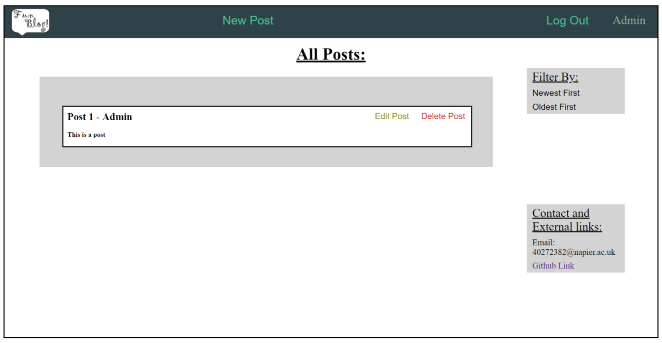


A title can be typed into the top box and some writing can be typed into the bottom box to create a new post which will be stamped with the users user name once they click post as can be seen in the next two images below:



Post Title: Post 1

Post Content: This is a post

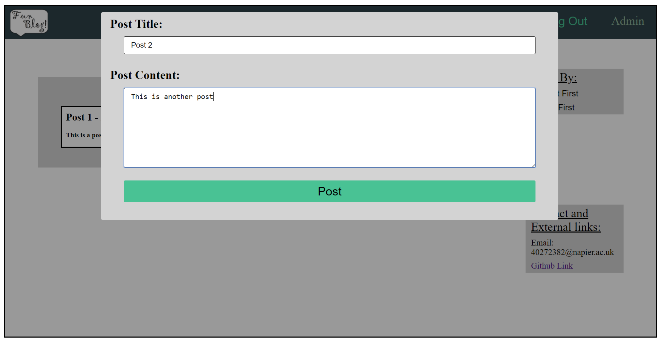


Post 1 - Admin

This is a post

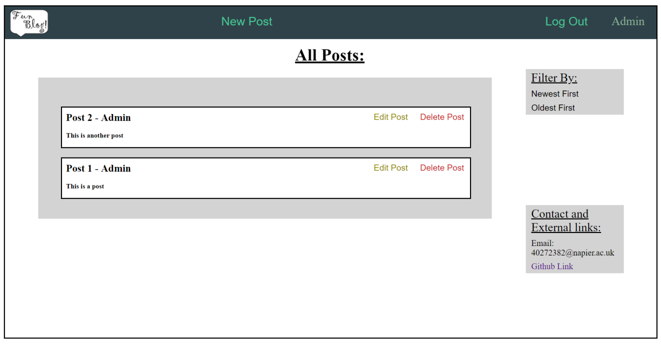
Email: 40272382@napier.ac.uk

Once a second post is created the posts are sorted in 'Newest First' order as can be seen in the next two images below:



Post Title: Post 2

Post Content: This is another post

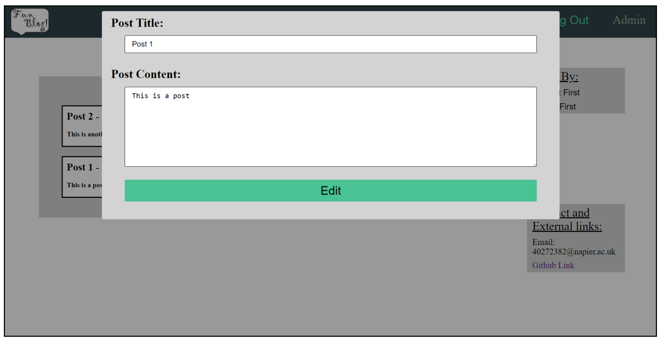


Post 2 - Admin
This is another post

Post 1 - Admin
This is a post

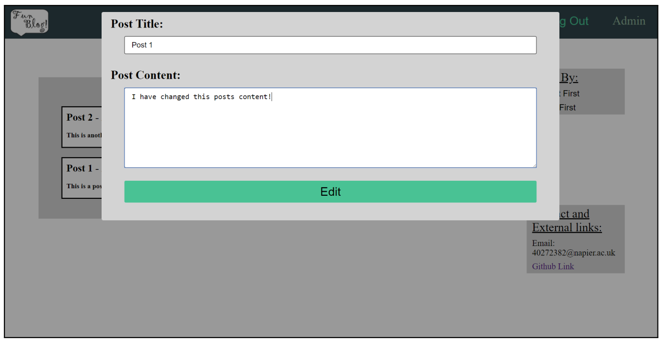
Email: 40272382@napier.ac.uk
Github Link

If the user clicks on the 'Edit Post' option on their post (or any post if the user is the Admin) then a box will open to allow the user to edit and update a post that they have already posted as can be seen int the next three images below:



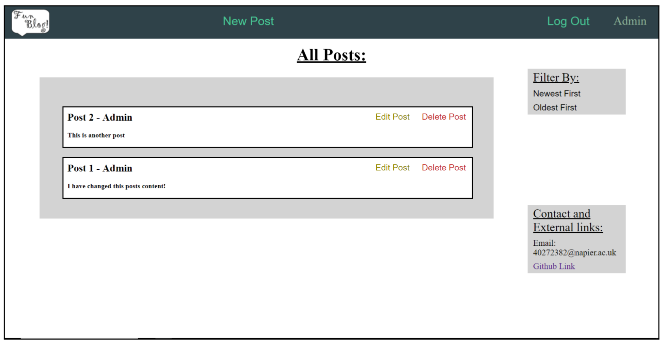
Post Title: Post 1

Post Content: This is a post



Post Title: Post 1

Post Content: I have changed this posts content!

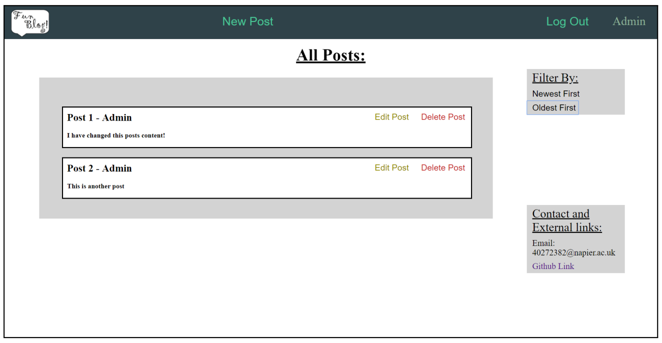


Post 2 - Admin
This is another post

Post 1 - Admin
I have changed this posts content!

Email: 40272382@napier.ac.uk
Github Link

As mentioned prior the posts by default are in 'Newest First' order however this can be changed in the upper left hand box to allow an 'Oldest First' configuration. Edited posts are not counted as new so will display in their original positions dependant on whether the filter is set to 'Newest First' or 'Oldest First', this can be seen in the image below:

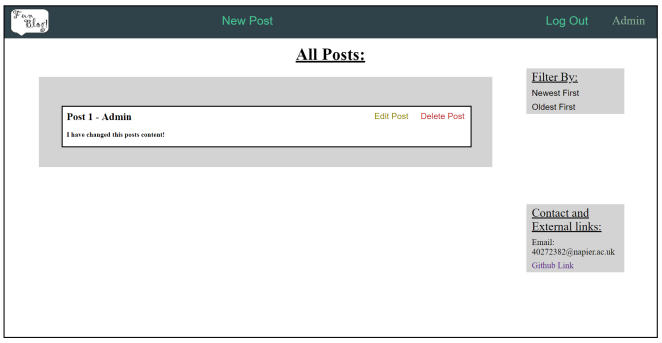


Post 1 - Admin
I have changed this posts content!

Post 2 - Admin
This is another post

Email: 40272382@napier.ac.uk
Github Link

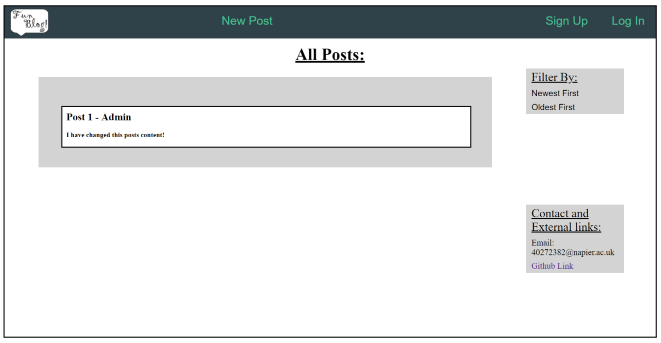
If the 'Delete Post' option is selected on a post on their post (or any post if the user is the Admin) then that post will be permanently deleted from the site as can be seen in the image below:



Post 1 - Admin
I have changed this posts content!

Email: 40272382@napier.ac.uk
Github Link

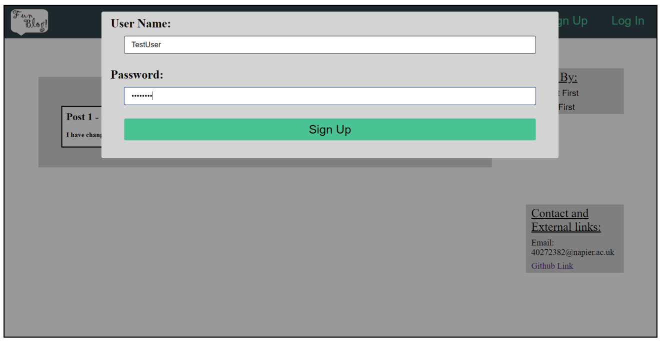
If a signed in user click the 'Log Out' option that appears on the top bar they will no longer be able to edit their posts as can be seen in the image below:



Post 1 - Admin
I have changed this posts content!

Email: 40272382@napier.ac.uk
Github Link

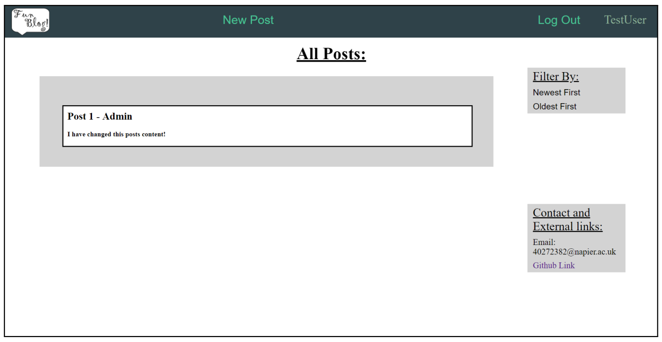
A new user can be created by clicking the 'Sign Up' button and entering a new user name and password as can be seen below:



User Name: TestUser

Password: ••••••••

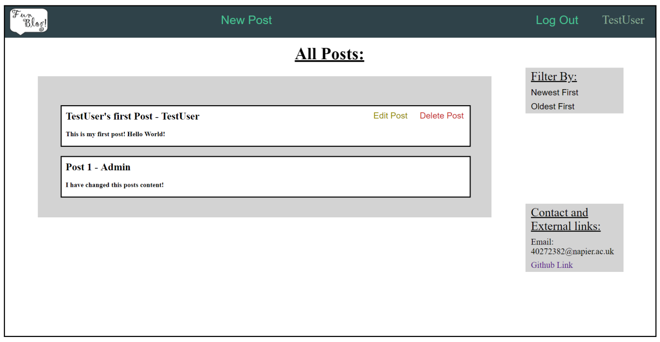
Once Signed up and logged in a new user has full access to create new posts, however as can be seen in the two images below, cannot edit the posts of other users:



All Posts:

Post 1 - Admin
I have changed this posts content!

Email: 40272382@napier.ac.uk



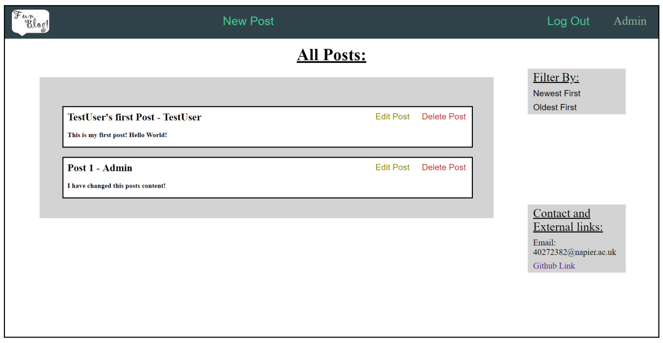
All Posts:

TestUser's first Post - TestUser
This is my first post! Hello World!

Post 1 - Admin
I have changed this posts content!

Email: 40272382@napier.ac.uk

In the image below it can be seen that if the Admin were to log back in, they would still have full access over all posts within the site:



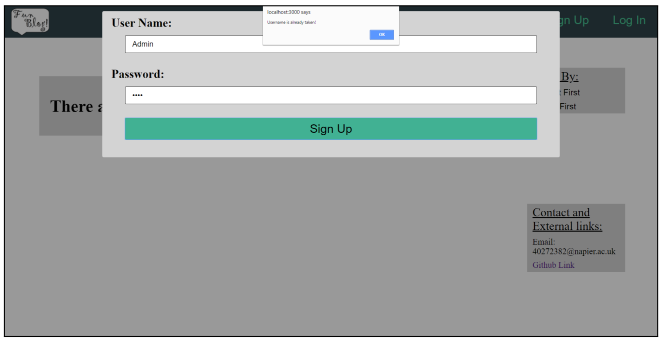
All Posts:

TestUser's first Post - TestUser
This is my first post! Hello World!

Post 1 - Admin
I have changed this posts content!

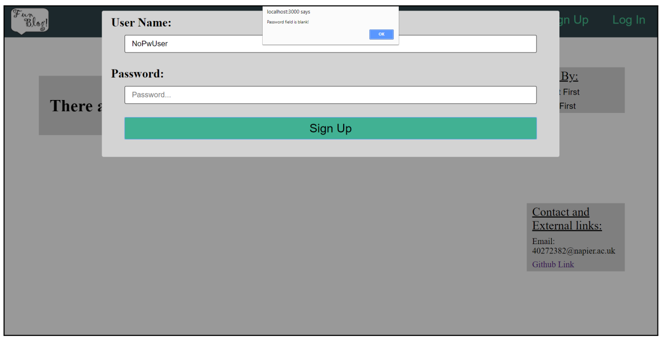
Email: 40272382@napier.ac.uk

Lastly, a few of the different modes of validation can be seen within the following four images:



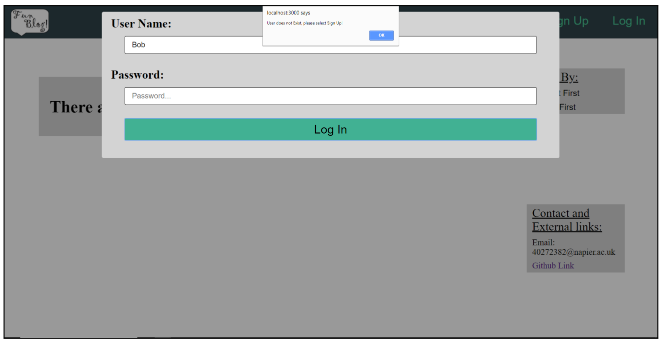
localhost:3000 says
Username is already taken!

User Name: Admin
Password: ....



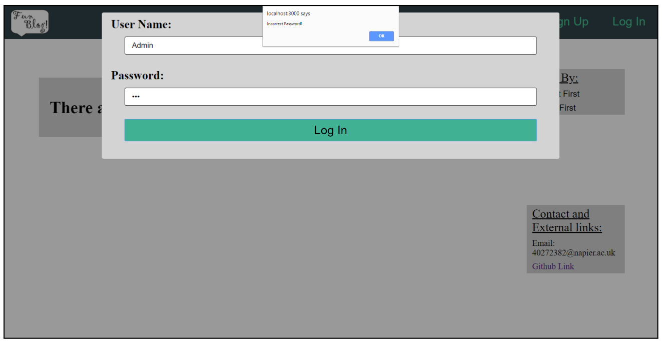
localhost:3000 says
Password field is blank!

User Name: NoPwUser
Password: Password...



localhost:3000 says
User does not Exist, please select Sign Up!

User Name: Bob
Password: Password...



localhost:3000 says
Incorrect Password!

User Name: Admin
Password: ...

As a brief side note, the sites database and each of it's data tables were externally hosted on mlab.com which required me to add a form of server validation when connection to and from the server to ensure the page did not update information before the database and its corresponding information was correct to the latest updates to the given information.

4 Evaluation

Overall I believe the project to be a great success going off of the plan set and I found the project as an overall good learning experience, especially in regards to the new knowledge and skills I gained using basic server-side technologies.

4.1 Critical Evaluation

The overall projects success does not leave it immune to problems and overall improvements that could have been made given more time and if I had had more knowledge on some of the additional features of libraries I used. Overall towards the end of the project I began to regret using AngularJS over JQuery and EJS for the dynamic database oriented features however at that point it was a little too late in to the project which was already under a somewhat tight time frame to where I didn't have the opportunity to backtrack and recreate some of the more inefficient features. Another of the features I wanted to change but did not have the time and opportunity to was to make the input validation (especially that of passwords) on the server side as having this validation on the client side does lead the potential of malicious attacks to the server. I also did not account for Injection Attacks within the input fields used by the database which can also leave a server and it's corresponding database at risk.

In regards to features which I would have liked to add but did not have the time or knowledge to implement there were a few key contenders. The first of the features I would have liked to add was more post filter options, especially a search involving the user looking for titles by being able to type them in to find a post, however more filter options such as alphabetical order would have been a nice feature. Another feature I would have liked to have was a form of 'likes' or up-votes' system similar to those common social media sites to which I would have also added a 'Most Liked' post filter option. Lastly I would have liked to implement an improved design to the site to make the site as a whole look a little less 'bland', especially in regard to the error message system which I ended up using the 'alert()' option in JavaScript as a basic method so I could ensure I had time to complete the project within the given time frame.

4.2 Personal Evaluation

I feel as if this project was personally beneficial in my learning of web technologies as I learned a good number of new skills using features of JavaScript as well as learned a lot about the development and use of server side technologies, especially those revolving around NodeJS. My usage and eventual regret of using AngularJS was an excellent experience about my decision making in regards to selecting the correct software and libraries when starting a project and ensuring I plan further ahead to ensure *every* feature I plan to implement can be done in a consistent fashion using the development environment that I select.

I feel as if the experience working with the technologies I used throughout the project have been a good insight further in to the common tools and methods used within working web development environments, especially those used within smaller scoped projects.

Despite feeling rather disheartened by the project towards the end, where I began to try to rush the project out so that I could have it done and out of my way, stumbling over many small and normally non-existent issues, I still feel as if the project as a whole was a great learning experience and I look forwards to further development tasks that I may have the opportunity to work on in the future.

5 References

Some additional sites used within the development include:

- www.w3schools.com - Learning and additional reading around the use of HTML, JavaScript, CSS, NodeJS and AngularJS.
- expressjs.com - Learning the uses and benefits of Express.
- angularjs.org - Script Source and a general overview of the uses and benefits of Angular.
- youtube.com - Overview of some of the various request types used by Node and Express.
- coolors.co - Site colour selection.
- mlab.com - Database hosting.
- facebook.com & twitter.com - General inspiration for the sites page layout.