Coursework Report

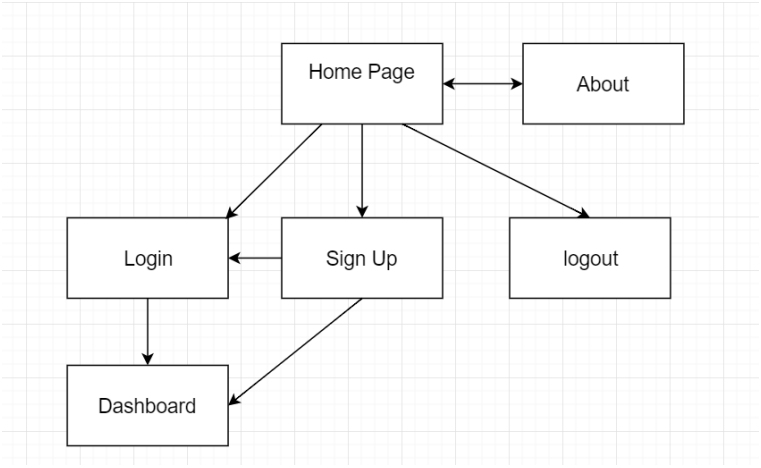
Christopher Jones
40274924@napier.ac.uk
Edinburgh Napier University - Web Technologies (SET00801)

1 Introduction

The main point of this coursework was to create a blog website using Node.js and the packages that comes with it. Node.js is a cross-platform JavaScript run-time environment that allows the execution of server-side JavaScript code. We had to use Node.js to implement a blog which could create posts, edit, delete posts and store those posts so that the user could come back and read them again. My blog was called "Wargaming News Blog" which is a company who is very controversial in the gaming community as they "love a good cash grab" and will do things that net them the most money. My blog contains the ability for the user to register and account and use that account to log in and make a post. The user can also edit and delete there posts if they so wish. Just for the user convenience I added a feature which allows them to quickly get to the dashboard to make a post rather than having to log in each time if they wish to make a post which takes effort. There is also a logout button so that the user can destroy their session on the website so no one else can use their account while they are away. The main page on my blog just shows example blogs that have been hard coded into the program so that the user can see what they can do when they create a post. I didnt do any background reading as such but instead watched a fantastic tutorial on YouTube by a guy called Manthan Dave YouTube channel https://www.youtube.com/channel/UCHbPSsEyu5__6nnOjHtUyTQ He gave me the fundamentals of Node.js in an easy to understand 20 part series. However, he uses a program called "Postman" to make GET and POST requests to the server but that wasn't hard to change in the website. I did do a lot of searching Stack Overflow for what commands I could use on server-side JavaScript vs client-side JavaScript as they vary quite a bit.

2 Software Design

I planned my website out on Drow.io https://www.draw.io/ which has a lot of tools to create quite a good and detailed plan of the blog so that i could see how i think pages were going to link etc. In this blog what I really wanted in it was to have a login system setup which only allowed the user to write a post to the blog if they were actually signed in.



Above you can see the design that i wanted to implement with the user starting on the homepage and being able to navigate between several different pages such as the about me page, Login, Sign Up (Register) and Logout page which of course should do nothing if the user isnt actually logged in. As can be seen from the desgin the user has to either create and account or Login to be able to access the dashboard so that they cant create a post if they aren't logged in. This design I think will be the most optimal for my blog as it wont require too much knowledge of JavaScipt to be able to link together the pages

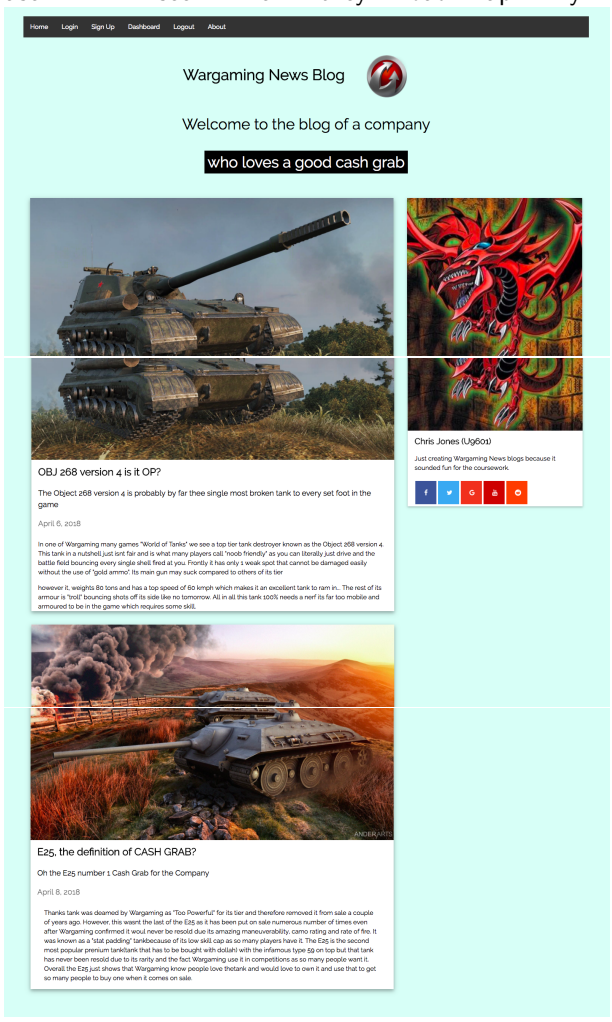
Above is my design for the navigation bar which includes all web pages event if they aren't linked together. The dashboard and homepage aren't linked but once a user has been created or logged in then that means they should be able to click the dashboard button on the home page and get redirected to their dashboard. If a user attempts to click it and they aren't currently a user then i think they should be redirected to the login page so that they can login. While doing some research about Node.js I found the social media icons which i thought were really cool and I really wanted to have them in the blog for a bit of shameless promotion of my social media which is so active :P. The social media buttons can be seen below.

Of course I don't have every kind of social media on that list so my plan is just to use the ones that i actually use e.g YouTube, Facebook, Twitter etc. One of the major things i realized before writing anything is that Node.js uses something called "Jade" instead of HTML for its web pages and there are some very key differences between the 2. One of them is that Jade doesn't use end tags only start tags but it knows whats in the lag by the use of indentation if

a tag is placed and nothing is indented inside of it like and if statement then nothing will use that tag. For example Head h1 This is an Example Body From the example above you can see that h1 is indented inside the head if it was at the same indentation then it wouldn't work. This was interesting and strange to learn about and understand as HTML is all about tags and this changed the ball game. So my plan before I started was to try and learn a bit of Jade so when it came to doing the blog I knew how to use it. I soon realized that there are many HTML to Jade converters on the internet so actually I dint need to learn Jade as much as I thought but still did a bit before starting so I had a bit of knowledge but having the converters would make life for me very easy as I could just write in HTML and covert it to Jade rather than having to do the conversion myself. Before actually writing the blog as I mentioned earlier I did use Manthan Dave's tutorials so I had a very good understanding for the Server-side JavaScript that I would be using for this project. This guys videos were fundamental and gave me a lot of things I wanted to have in my blog such as login system and the ability to write posts (sort of took a bit of changing) etc.

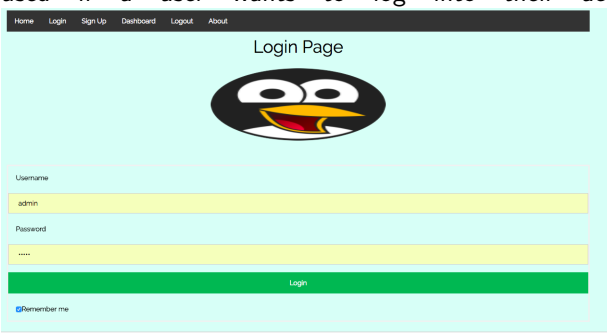
3 Implementation

The first thing I implemented was the Home page which of course is the first page the user will see when they load up my blog.



The above screen shot is what the page looks like however because of how large it is I had to save it in 3 images. But as you can see from the pictures "Wargaming" created "World of Tanks" which my blog is based upon. Everything on that homepage contains some sort of CSS to make the posts look like an actual blog type post which I thought was pretty cool. You can also see my Implementation of the social media buttons underneath my little image and a little description about the web page. Also the navigation bar at the top can be seen which I showed the design for above and is basically the exact same thing as I thought that was the best way of doing it. The 2 example posts on the front have been hard coded into the program to give the user and example of what can be done on this blog but other than that serve no real other purpose apart from looking good on the front page. Other than navigating to other pages the home page does nothing other than being the central node in the web of web pages.

The next page is the Login page which is used if a user wants to log into their account

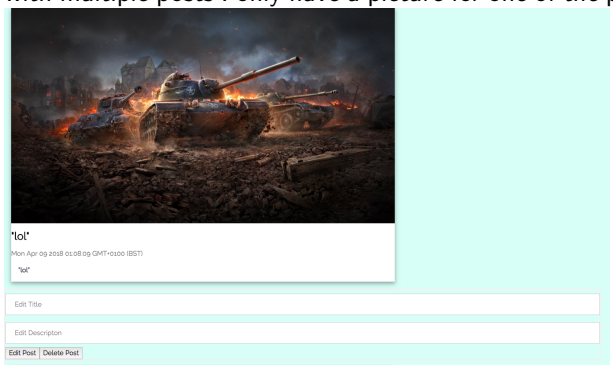


The login page is just a generic one with a little avatar at the top which is a penguin because that was one of the first things that came up on Google images and I thought it was funny. Then there is again the navigation bar at the top so that user can navigate back to the home page or what not etc. Then on the homepage there is a Username box and Password box which of course require your username and password to log in. In the screen shot there is currently already a user just because of testing purposes i need a quick login so got chrome to keep a hold of my login details. The 2 boxes must be occupied by some sort of text if they are left blank then the user will be requested to input data. Once the user clicks the Login button it sends a POST request to my JavaScript. Which can be seen below

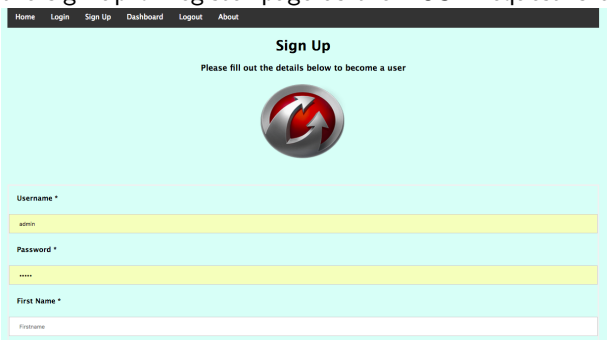
```
1 router.post('/login', function(req, res){
2   var username = JSON.stringify(req.body.username);
3   var password = JSON.stringify(req.body.password);
4
5
6   User.findOne({username: username}, function(err, user){
7     if(err){
8       console.log(err);
9       return res.status(500).send();
10    }
11
12    if(!user) {
13      res.render("../views/login.jade");
14      return res.status(404).send();
15    }
16
17
18    user.comparePassword(password, function(err, isMatch){
19      if (isMatch && isMatch == true){
20        req.session.user = user;
21        var postTitle = req.session.user.posttitle;
22        var postDesc = req.session.user.postdescription;
23        var date = req.session.user.date;
24        var postTitle1 = req.session.user.posttitle1;
```

```
25        var postDesc1 = req.session.user.postdescription1;
26        var date1 = req.session.user.date1;
27        var postTitle2 = req.session.user.posttitle2;
28        var postDesc2 = req.session.user.postdescription2;
29        var date2 = req.session.user.date2;
30        var postTitle3 = req.session.user.posttitle3;
31        var postDesc3 = req.session.user.postdescription3;
32        var date3 = req.session.user.date3;
33        var postTitle4 = req.session.user.posttitle4;
34        var postDesc4 = req.session.user.postdescription4;
35        var date4 = req.session.user.date4;
36        var postTitle5 = req.session.user.posttitle5;
37        var postDesc5 = req.session.user.postdescription5;
38        var date5 = req.session.user.date5;
39        if(req.session.user.posttitle && req.session.user.↩
          postdescription != null){
40          return res.render('../views/index3.jade', { newPostTitle: ↩
          postTitle, newPostDescription: postDesc, date: date, ↩
          newPostTitle1: postTitle1, newPostDescription1: postDesc1↩
          , date1: date1,
41            newPostTitle2: postTitle2, newPostDescription2: ↩
          postDesc2, date2: date2,  newPostTitle3: postTitle3, ↩
          newPostDescription3: postDesc3, date3: date3,  ↩
          newPostTitle4: postTitle4, newPostDescription4: postDesc4↩
          , date4: date4,
42          newPostTitle5: postTitle5, newPostDescription5: ↩
          postDesc5, date5: date5 });
43
44        }else if (isMatch && isMatch == true){
45          req.session.user = user;
46          res.render('../views/dashboard.jade');
47        }else{
48          res.render("../views/login.jade");
49          return res.status(404).send()
50        }
51      }
52    })
53  });
54 });
```

So once a POST request to /Login is received then it checks to see if a user name matches a username we have on record in the MongoDB Database which stores all of my data about uses and scrambles their passwords. I use mongoose as the middle man to send and receieve data from the database. If the user does exist and then next we check their password against the database. To compare a hashed password I use Bcrypt to compare the hashed password to make sure the passwords are indeed the same. If the password is the same well then they are a user and I can then get all their session information for them including their posts. Now to get multiple posts to work I have a Schema(the columns in a database) which can contain 6 posts not the ideal way of doing it but it will just have to do for now. Of course if they have never written a post then we check to see if the first post is = null telling us that their isn't a post. Once we have all the posts then we render the dashboard with the posts on it or if there isn't any posts then just render the dashboard. The next page is of Course the dashboard however has it can get quite long with multiple posts I only have a picture for one of the posts.



As you can see above the Post that the user has made has been posted to the page. The post button is located above and is out of shot but you get the point. So the user can edit that post and it will update the time stamp however, my edit buttons require you to copy and paste the post back in and then edit the but you want to edit due to some time constraints I had with other CW's but mainly my fault. I wanted to have a text editor appear and I know this is possible in Jade but instead had to go with this method which will edit the post no problem it will just require a quick copy paste of the old post not ideal again but it does do the job. Because my Schema allows the user to have a max of 6 posts on the page they can create 6 but once they reach 6 it will begin to over right their oldest post which again isn't the greatest of design but I had problems with multiple posts so went with a way that would work but not practical what so ever. The next page I will show is the sign up or register page as the POST request is called.



This is the sort of final page that the user will most likely look at. It allows a user of the website to become a user of the blog by requesting they select a username, password and add their first and second name into the boxes. Currently if the user tries to leave any the boxes blank when they click the Sign up button they will be prompted to add a value into it. Once they click Sign up a new entry into the MongoDB database is made and now the user can log into the blog, once the data entry is made the user is redirected to the dashboard where they can make their first post. The actual Sign up is basically the exact same design and logic as the login page just except their is more boxes to receive data from and we send the POST request to a different window but apart from that it is the exact same. Lastly for implementation I would like to discuss how my blog stores data as I have mentioned it uses MongoDB database which runs on NoSQL. So there is no need to try and build a table using SQL commands as that's what the User Schema is used for. A Schema allows the database to understand what each column in the database is and what input it should receive for it. To actually connect to the Database we need a sort of middle man as the JavaScript cant directly communicate with the Database so this is where the Mongoose Package comes in and it allows our JavaScript to pass data through it and straight on to the database. This is the same when the JavaScript requires data from the Database it uses mongoose.

4 Critical Evaluation

So the main thing that got the cut from the requirements was the about me page as on the home page I already have a sort of about me so I thought there wasn't really much point in

adding a whole about it as that is just sort of pointless. But apart from that everything else that I wanted to have in my blog made it in the main one being the login in system which I'm am very proud that it actually worked. I was also happy to get the MongoDB to work fully as I initially had multiple problems with understanding how I could connect that to my blog but eventually with the help of Manthan Dave I was able to do that successfully. I think one huge thing that I would have liked to have fixed and I wanted to have was multiple posts because at the moment my Schema allows the user to have a max of 6 posts and no more. This is a sort of problem as I had initially looked into saving an array to the Database but that proved to be a lot of hassle of which I couldn't make work so made the decision to abandon that idea even though it was one thing I required to have. Another thing that I would have liked to have added would have been the ability to edit the post with a built in text editor rather than having the user copy and paste their posts back into the box and then having them edit what they need to edit kind of a slight design flow but it does technically work. I would have also liked to have the users name be displayed on each of the user on each page as long as they have signed in but just sort of got pushed back as time became low. Another thing is I would have liked to have the current logged in users posted to the home page a long with the example posts but just ran into issues with multiple posts so again that kind of got pushed back. One thing my blog is missing is the ability to add comments to a post and read more about the post but I think that was going into a very detailed and complex zone as trying to store each users comments would have required another Schema and I'm not sure how you would do the connections because you can't have multiple connections to the database at the same time and that is what I would have needed.

5 Personal Evaluation

I have learned so much from this coursework and actually really enjoyed the Server-Side JavaScript as it was pretty cool to learn how to use. I have also learned how Jade works and how annoyingly temperamental it can be when you mess up your indentation and break half your web page honestly so fun. A challenge that I managed to overcome was actually being able to send GET and POST requests without the use of a 3rd party program because to start off with I never really understood that Node.js was server sided and thought I could get the data inside text boxes using client side commands which of course isn't the case. Another challenge was trying to get the Database to display what was actually in it e.g the posts. This was a problem because I couldn't find a way of passing the data from JavaScript to the Jade file like you would with client side so after some searching I found that you use the "extends dashboard" and then with Jade you can actually use small snippets of code like if's and elses meaning I could set the value from the database and then set it to a variable in Jade which I could then display on screen. I also found out that you can nest the Blocks (Jade methods effectively) inside each other as long as you create another Jade file and extend the last one so effectively you can nest methods inside other methods which meant on the Jade page that loaded the Dashboard there is hardly any Jade as all it needs to do is call 1 block and it gets all of the data. Overall I'm actually happy with how my website turned out even though there is some much needed changes I think in the time I actually used to work on it was sufficient. I mean it has a functioning database, posts and a user which looks good overall and doesn't crash thankfully. I think I have followed and fulfilled the specification to quite a high degree and I am content with how I have done :D.

References

[1] Manthan Dave Lad Node.Js Tutorials, `https://www.youtube.com/user/themanthandave`

[2] Stack OverFlow for General Stuff, `https://stackoverflow.com/`

[3] w3Schools for some HTML and CSS, https://www.w3schools.com/

[4] HTML to Jade Converter, http://html2jade.vida.io/