Edinburgh Napier UNIVERSITY

Coursework Report

Namo Najem
40313888@napier.ac.uk
18th April 2019
Web Tech CW2 - Edinburgh Napier University

1 Introduction

1.1 Introduction

Web design is among the most flexible platforms in computing so much so that it's very important for every developer to fully grasp and wrap their head around. The first coursework was intriguing but developing a full server side clientside secure website/service will be the most useful in broadening our understanding of WebTech.

1.2 Goals

The goal of this assignment is to build upon the cyphers and website from the first coursework and create a website that handled all important data on the server side for vast security and potential performance gains. I chose to use dependencies available on NPM to create a simple real time messaging lobby that also includes the ability to send the messages as encoded from our cyphers. Alongside this there is a user registration and login enviroment, a user must be logged in to access the chat elements of the site.

2 Software Design

2.1 UI

The UI for the project is rather rudimentary and very little time was spent on it, bootstrap is used for all the registration and login with some custom CSS for the chat window and messaging form.
Jade is used for every page on the website, it's simpler to work with and the indentation for divisions work very well and clearly.

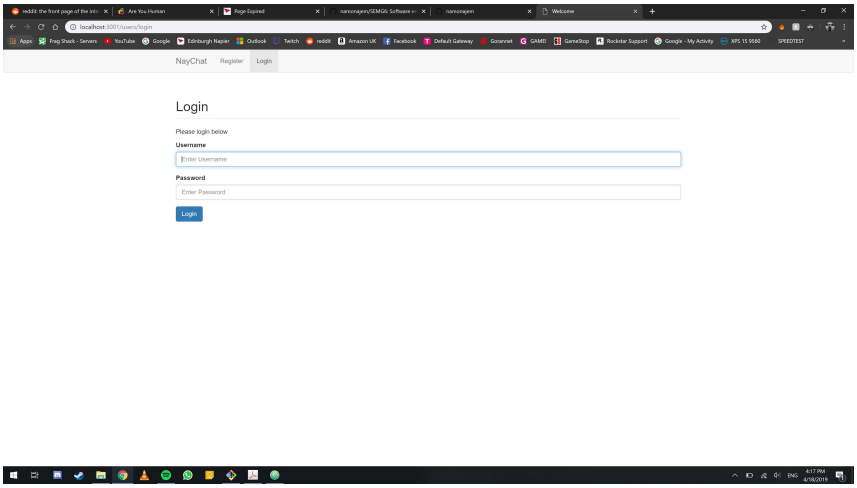
2.2 Users and MongoDB

A database system is necessary to obtain proper user registration and login functionality. We use a simple user schema off mongodb to save and keep all user data persistently between sessions. The implementation allows for server to be setup easily on new machines as all that's required is for mongodb to be installed on the server machine, if no collection is available the server automatically creates a collection and adds all the users information. Passport is used as a dependecy to keep track of user sessions authentication.

```
// Schema for user registration and login using mongo db
var UserSchema = mongoose.Schema({
  username: {
    type: String,
    index: true
  },
  password: {
    type: String
  },
  email: {
    type: String
  },
  name: {
    type: String
  },
  profileimage:{
    type: String
  }
});


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

module.exports.getUserById = function(id, callback){
  User.findById(id, callback);
}

module.exports.getUserByUsername = function(username, callback){
  var query = {username: username};
  User.findOne(query, callback);
}

module.exports.comparePassword = function(candidatePassword, hash, callback){
  bcrypt.compare(candidatePassword, hash, function(err, isMatch) {
      callback(null, isMatch);
  });
}

module.exports.createUser = function(newUser, callback){
  bcrypt.genSalt(10, function(err, salt) {
      bcrypt.hash(newUser.password, salt, function(err, hash) {
        newUser.password = hash;
        newUser.save(callback);
      });
  });
}
```

2.3 bcrypt

Bcrypt is a dependency on NPM. It's been used to encrypt user passwords before they are saved to the mongodb collection. Encryption has evolved and become a leading industry standard and so using this dependency is a more secure than writing my own encryption function from an algorithm as it's opensource. Also no need to reinvent the wheel.

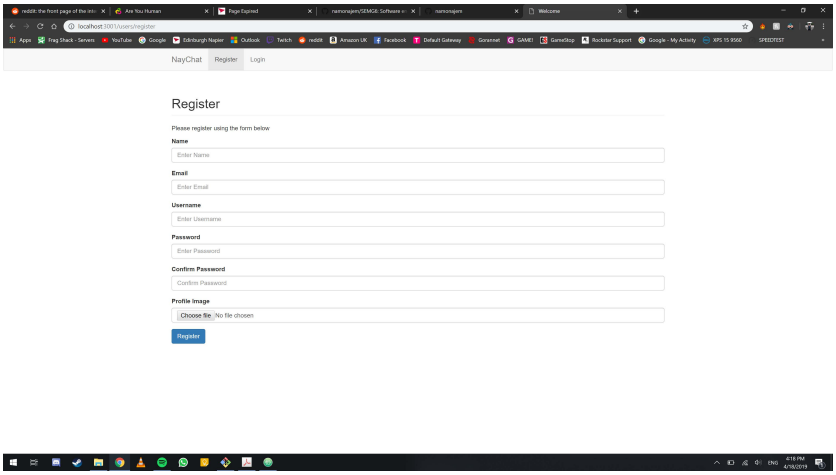


3 Implementation

When the website is first opened the first page is a registration form. Above is the navigation links to the rest of the pages, some have been disaabled as they can't be accessed without a login. After a user has logged in/registered they are redirected to the live chat page where they are placed in the lobby. Each user's username is then added to the list of active users. If a user wants to leave the chat they can logout in the top right corner ot simply close their browser/tab, this automatically removes their username from the list.

3.1 Registration and login

Alot of this code is written with the help of stackskills. All proper validation is in place for someone to register ie: checks if username is taken. And the same goes for logging in. The code responsible for this functionality is mostly under the users.js in the routes folder and the user.js schema responsible for creating and adding new users information to the database.

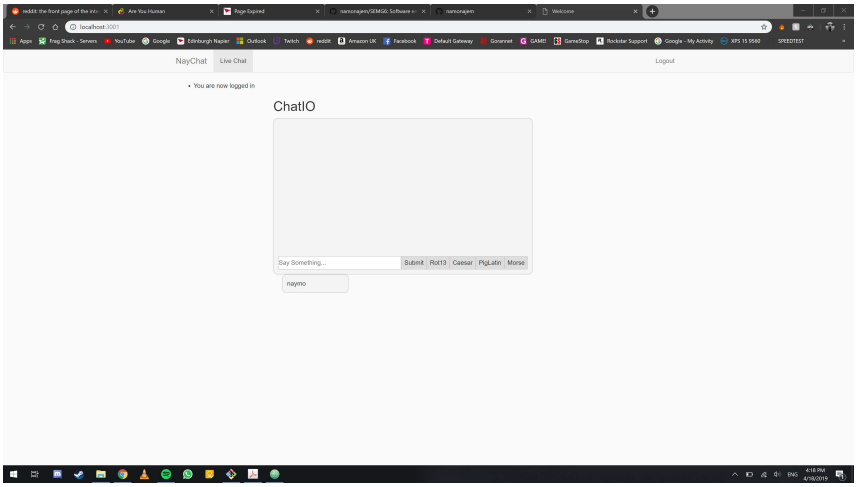


3.2 Top Navigation

To navigate between the websites pages some sort of navigation layout is required. layout.jade is a file that all pages extend from and it houses the neccesarry code to make it work, including hiding certain elements depending on the current page.

4 Socket io

A huge part of making this project viable is the inclusion of the socket io dependecy on NPM. Using socket we can quickly setup a connection between the client and the sever and relay data back and forth in real time. With this kind of functionality easily obtained, the rest of the proccess of creating the live chat falls into place. Once the server recieives a message it relays it to all other connected sockets. Socket is perfect for this part of the project as it was designed and built specifically to do this. Further functionality can be easily added from what is already working for example private messaging or persitent messages.

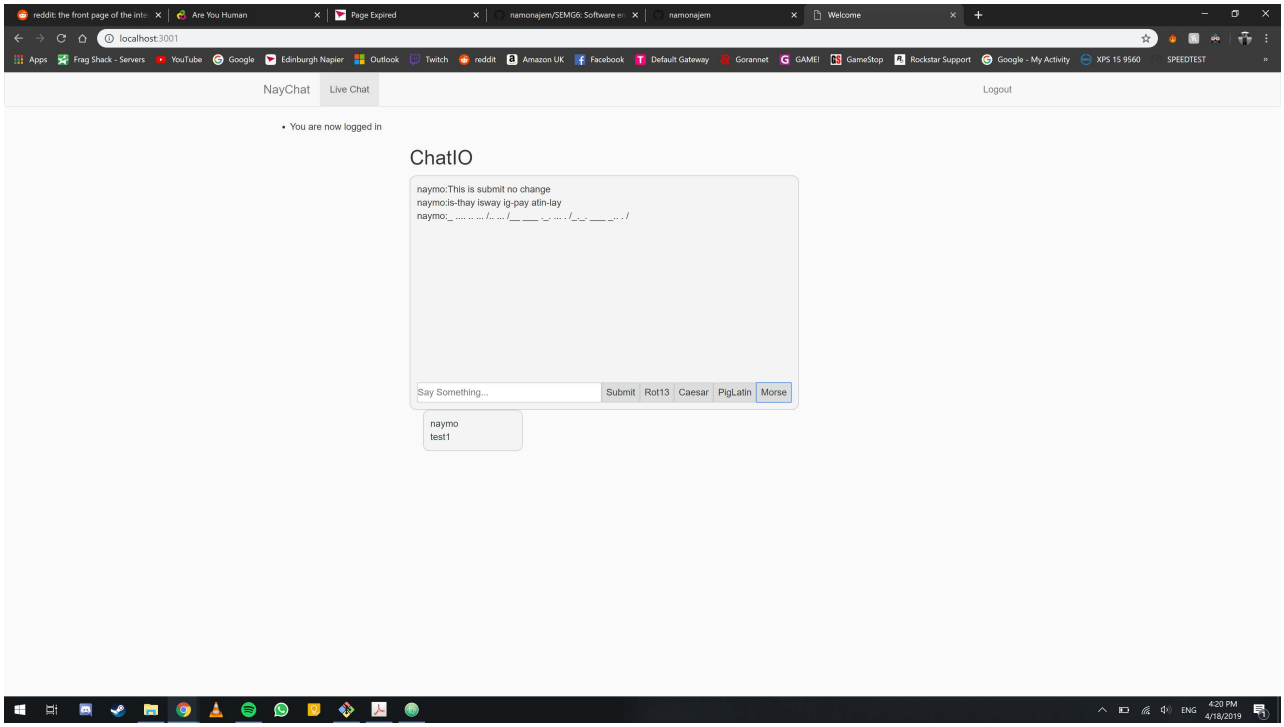


4.1 Cyphers

The cyphers from the first CourseWork have made a return on this project. All of them are implemented on the client side(right before messages are relayed back to server). This isn't the ideal configuration,due to time constraints, but it works.

5 Critical Evaluation

The requirements for this project started off clear but then got a bit broad. The goal was to extend off our first website: which i have done to and extend with my cyphers,
To create a server-side client-side enviroment using nodejs and express: I believe i'm very familiar with these after doing the realtime messaging,
robust reliable and secure: i have a very safe user registration, some things clientside are very makeshift but not unsafe.
user interface is unexciting and bland but functional.



6 Ciphers

(these are used so i copied from last report)

All Ciphers apart from the ROT13 from the practicals has been written up from scratch, there are a few simple flags to deter from invalid input.

6.1 ROT13 Cipher

The ROT13 Cypher is an ancient roman Cypher designed around the latin alphabet and as it's name suggests rotates the index by 13 charecters. Therefore the same technique used to encode the message can also be used to decode.

6.2 Caesar Cipher

Caesar Cipher is a cypher very similar to the ROT13 Cypher in that charecters are shifted a certain indexes to the right, some Caesar Ciphers can include a setting to select how much to shift the index.

6.3 Morse Cipher

Morse Cypher can encode from english to Morse Code and decode Morse Code back into English. Techniques used include spaces to split Morse letters and '/' to split Morse words from one another. This allows for precise and easy decoding where might otherwise not be possible

6.4 Pig Latin Cipher

The Pig Latin Cypher translates English into pig latin english by adding '-way' to the end of every word that starts with a vowel. If a word starts with a constant or

more than one constanant then all constanants until the first vowel and moved to the end of the word followed by a 'ay'.
The decision had to be made to include '-' in the pig latin as decoding would otherwise not be properly possible.

7 Conclusion and Reflection

7.1 Strengths and Weakness

Stengths of my project include that i got realtime messaging working on server integrated platform, encrypted and secure user registration, user freindly interface.

Weaknesses are that the UI is boring and uninventive, no private messaging, messaging is not persistent. and ofcourse that i didn't have the time to put decoding in aswell.

7.2 Personal Evaluation

I had alot more fun doing this Coursework compared to the first one, mostly from the realtime server thing(where most of my time went). To be honest i dont see my self ever doing something like web full time in the future and i am therefore not that passionate about it but i recognize the importance of understanding and familiarizing ourselves with as many languages as possible and web is one of the most key platforms.

8 References

https://en.wikipedia.org/wiki/Caesar_cipher

https://en.wikipedia.org/wiki/ROT13

http://www.snowcrest.net/donnelly/piglatin.html

https://morsecode.scphillips.com/translator.html

https://en.wikipedia.org/wiki/Morse_code

http://www.learnmorsecode.com/

https://socket.io/get-started/chat/#The-web-framework

https://stackskills.com