**Contents**

**Table of Contents**

**1. Analysis - Page 2**

**1.1 Introduction - Page 2**

**1.2 Investigation - Page 3**

**1.3 A general overview of my objectives - Page 11**

**1.4 Specific overview of my objectives - Page 12**

**1.5 Prototyping - Page 13**

**2.Design - Page 17**

**2.1 High level overview - Page 17**

**2.2 Descriptions of modular systems - Page 22**

**2.3 Data Dictionary - Page 23**

**2.4 Validation - Page 30**

**2.5 Database Design - Page 31**

**2.6 Planned SQL Queries - Page 34**

**2.7 UI Design - Page 37**

**2.8 Security - Page 46**

**3.Technical Solution - Page 47**

**4.Testing - Page 123**

**4.1 Postman - Page 123**

**4.2 Testing Strategy - Page 124**

**5.Evaluation - Page 137**

**5.1 Objective Check - Page 137**

**5.2 Feedback from users - Page 139**

**5.3 Potential Improvements - Page 142**

**Analysis – 1**

**Introduction – 1.1**

**Background**

For my NEA project I will be making a social networking website that will have profiles, groups, events and a calendar.

I will be coding in JavaScript and HTML, for the database I'll be using MySQL and using Docker to hold the table in a container.

**Problem**

A lot of the time when you’re working or in school studying, you’ll end up having a very busy schedule and you will find that you and your friends never really have time to get together. My aim is to find an easy way to discover whether you and your friends are free for a night out or not. By creating a calendar that allows for multiple people to input their appointments and schedules all you have to do is look into the calendar and find a day where none of your friends have something planned making meeting up with your buddies exceptionally easy.

**Limitations of Current Situation**

The current limitations of social networks such as snapchat and Instagram is that they tend to stay in their lane in terms of what they're good at. For instance, snapchat does not copy Instagram and have a profile with people's pictures from years ago. Why not have a social network application that can have everything the user wants? You could combine all of the greatest features of every social network out there.

**Target Audience**

My project will be aimed at all ages from teenager and above as everyone starts to use a calendar from roughly secondary school in order to stay organised and on top of work. It can be used by younger people to organize school group projects or meet ups with friends. It can also be used by older people in terms of meetings for work or lunch

**Investigation – 1.2**

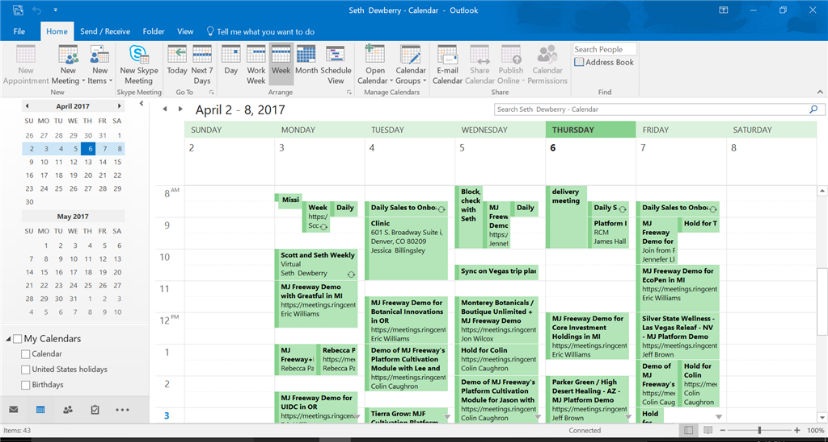
**Preparation**

At the start of year 12 we were learning to code in python which was a language I had a bit of knowledge of since I learned and used it quite a bit during GCSE so naturally when looking at possible ways to code this project, I tried to search up ways python can work with html. I didn’t find many possible options, one of the few was flask which is a web framework for python although I found it all to be rather clunky and in the end I decided to not carry on using this. I decided to look at other languages and quite quickly found that html very easily links up with JavaScript so I looked into using this for my project. Researching JavaScript showed me how easy it was for me to link databases then to my project by maybe using PHP or node js.

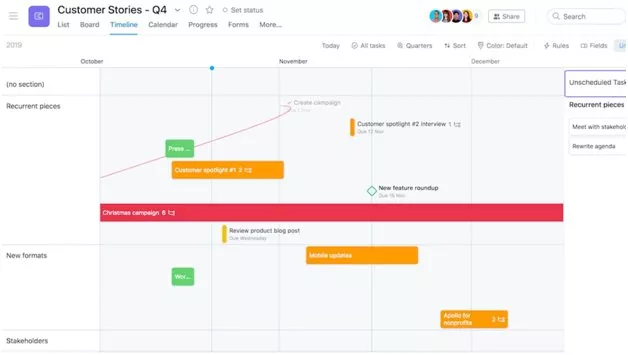
So I came to the conclusion that I was going to use JavaScript and HTML because it led very easily to database accessibility and the code was all very nicely laid out to be understood. There were also tones of learning resources on the internet which I needed because it was a new language to me.

**Inspiration**

**Outlook Calendar**



Outlook calendar is a very well-known calendar. Since a lot of people use outlook as their primary email the ease of having a calendar connected to your email appeals to a lot of people. It also has the ability to share your calendar with co-workers which is the start of becoming a group calendar but it’s not quite there. Another application is that you can create additional calendars for specific projects which influenced my idea of having separate calendar groups for things such as work, friends or school. A disadvantage of the Outlook calendar is that the GUI is close to identical with Googles Calendar showing a lack of thought in terms of the creative side of the calendar. Another disadvantage is that because the calendar for Outlook is almost like a side piece for the email it’s quite difficult to locate it from your email especially if you’re new to Outlook.

**Asana**

Asana puts emphasis on collaboration and productivity, the main feature of Asana is how flexible it can really be. You can use it to keep track of ongoing work and long term projects, you can also use it to coordinate with other teams and distribute workloads among your co-workers. Asana manages to put task management, communication and other helpful tools all within a single interface which shows how useful it really is.

The unique GUI (shown in the picture) is something I want to implement into my own calendar as I want it to stand out and not fall into the group of common calendars such as Outlook or Googles calendars. While by itself Asana doesn’t have a lot of useful features built in such as time tracking or live chat it achieves a lot of functionality through app integrations.

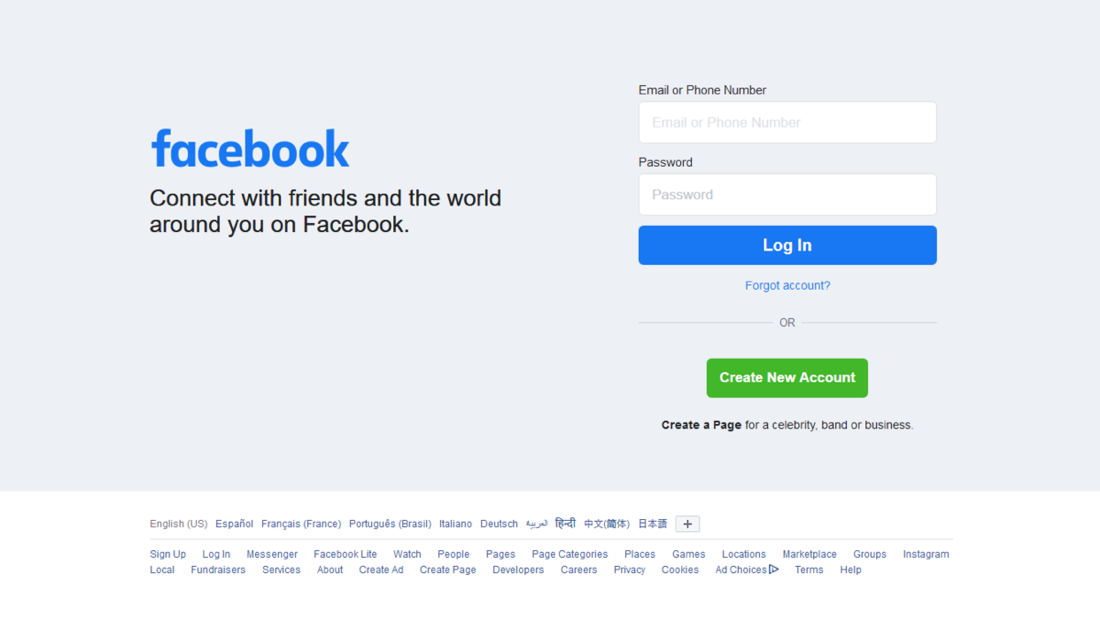
It works with slack and Microsoft teams for live chat and Harvest for time tracking. This could be seen as a disadvantage as it has to rely on other programs other than its own for all its features. This is something that I think I can improve on as these are things that I think should be implemented into a calendar by default.

A useful part of Asana that I plan on taking inspiration from is how easy it is to manage team projects as it has a timeline feature that shows every piece of a project, how it fits together and it helps you track the changes you or your group members make. This would be extremely useful for younger people in school who sometimes leave their projects to the last minute or struggle spreading the workload between the group evenly.

**Facebook**



While the other applications I’ve looked at have been solely calendars, Facebook differs from the trend. A huge part of my own project will be the social networking aspect and Facebook is a perfect example of this. Facebook uses profiles expertly by creating links between mutual friends and by having groups that people can join and chat in.



When looking at the Facebook website I found their login page to be simple and clean, with each of the buttons coloured and laid out to be seen easily. This was something I wanted to replicate on my own project as it’s user friendly and appeals to the eye.

Facebook also has tons of features that would take too long for me to create such as a feed which shows the user their friends’ recent posts and updates. While there are plenty of advantages to this website there are also plenty of disadvantages. They have a lot of privacy issues which when creating a social networking application is a major problem. Normally when creating an app like that a main point in creation would be cyber security but with the time constraints I have, this will be unable to be created but will still be mentioned on where it would normally be added.

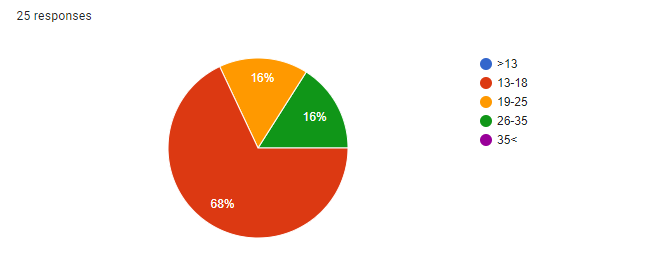
Social networking websites and apps are hugely popular, Facebook has 2.8 billion active users and is considered the biggest social network worldwide. This shows are useful applications like this are, there are multiple different websites and apps who are like Facebook such as Instagram or Snapchat and they’re also hugely popular despite being so similar to each other. This begs the question of what makes a social network website stand out?

* Target market penetration
* Better content display
* Active users

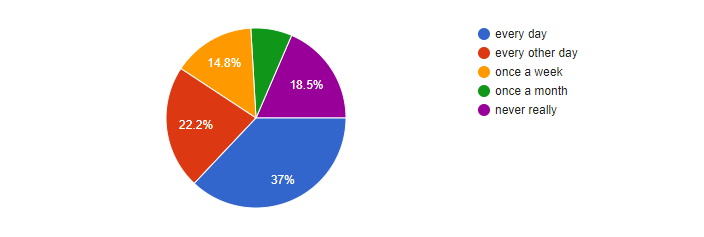
While these three points are very important when it comes to a social network, they're also extremely hard for me to implement but they’re interesting things to consider if one day I created a huge social networking application.

**Peer Survey**

I constructed a quick survey using Google Forms, which uses a simple interface to ask users questions, it also provides a useful analytic graphic of the results. Not all of the questions are mandatory as I felt some of them would require a bit more knowledge on calendars and in case they didn’t have the time to type long answers I left it as optional.

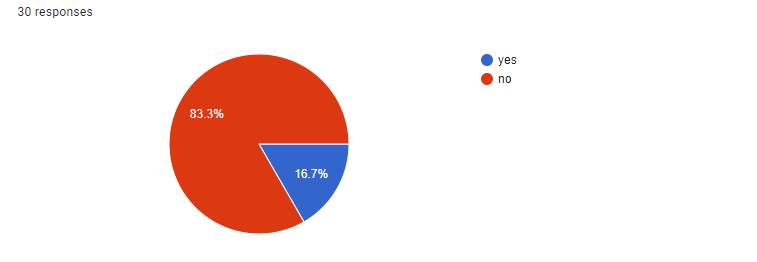
1. What age group are you in?

With most of my participants being teenagers or young adults I can assume they will all be used to new technology and to using calendars.

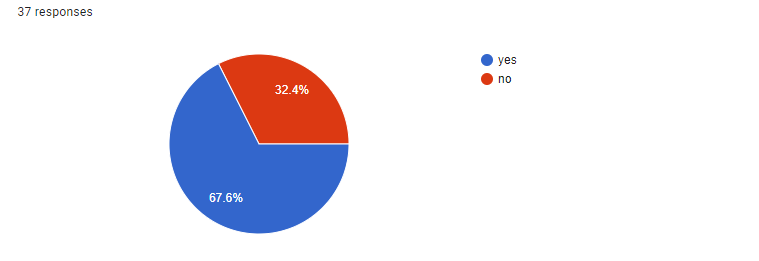
1. How often do you use a calendar?

This question provided a reasonably wide range of answers which the majority saying they check their calendar at least every other day and the next largest group saying they used it every day which shows how vital a calendar is to the ordinary person.

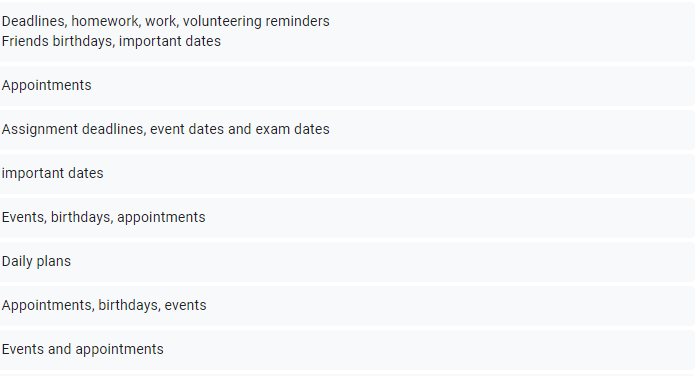
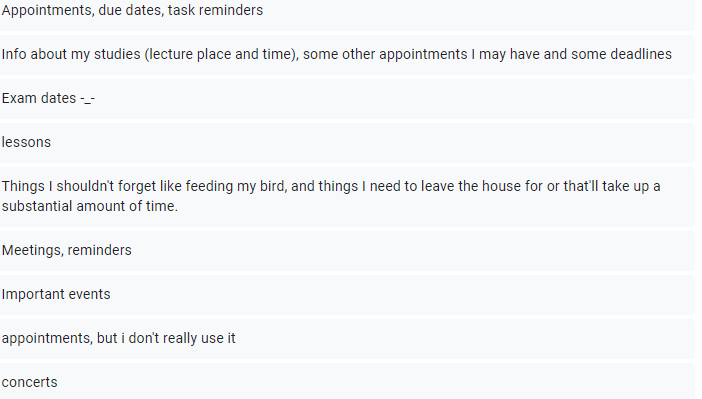
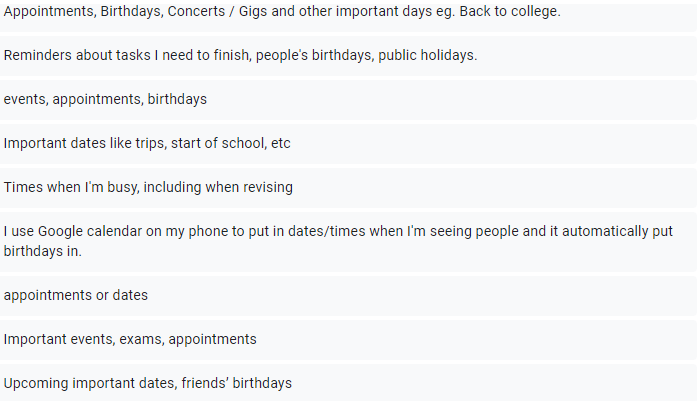
1. Do you have separate calendars for your work and social?

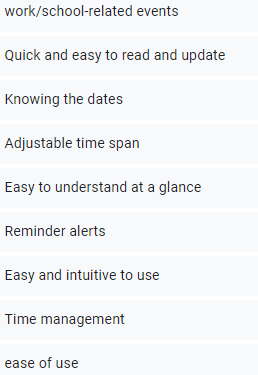
I found the results for this question to be quite interesting as I’d originally thought a lot of people would have separate calendars in order to avoid clutter in their calendars. However, I’ll still need to implement a way to have separate calendars because there could be an instance where the user is working with multiple different groups and needs to have different calendars for each of the groups they are with.

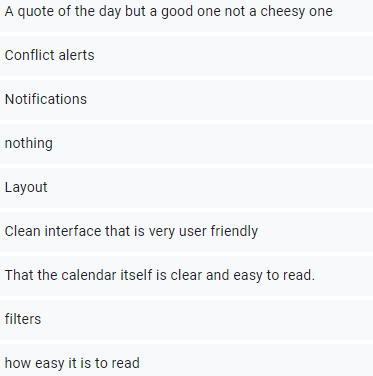
1. Would you benefit from everyone in your work group or social group adding their dates onto the same calendar so everyone could see when you're busy or not?

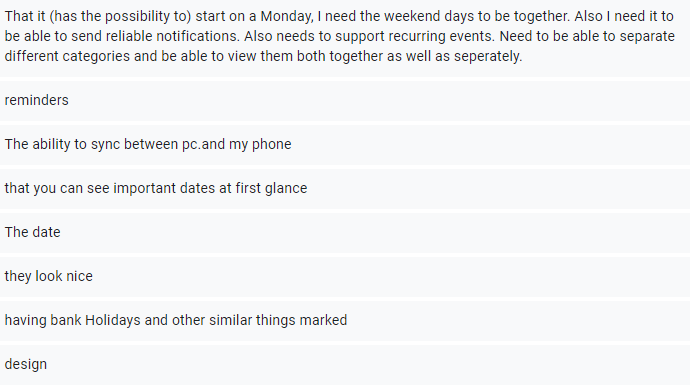
Two thirds of the participants said that they would benefit from having a group calendar in order to see when their work colleagues or friends were busy which shows that there is a market for this type of thing and a lack of a group calendar out there for public use.

5)what do you normally put in your calendar?

With this question the participants gave a large range of answers showing the importance of a calendar as it helps remind these people of appointments, birthdays, concerts and other things that they just can’t forget about.

1. What’s the most important thing for you when it comes to calendars?





“a quote of the day” – this is an interesting suggestion and can definitely be implemented but it’s not needed for the final project.

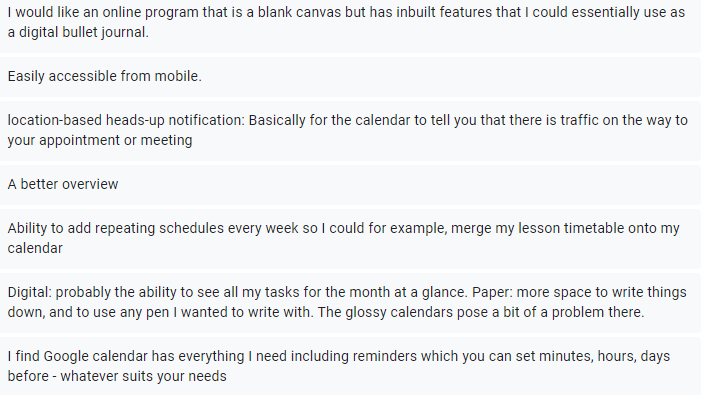
“layout”, “clean interface”, “user friendly” – it’s very clear that a lot of the participants felt that the user interface had to be clean and user friendly which is something I was hoping to implement as I feel that the interface is a main factor in what differentiates one calendar to another.

“customisable design, colour categorising for types of events” – giving the user a choice on the background and other things lets their calendar feel more personalised and they’re more likely to enjoy using it.

“conflict alerts”, “notifications” – this is something I planned on adding. I wanted to have a notification be sent to the user if they had been invited to a event or if one of their most recent changes to their calendar had created an overlap and was now conflicting with another event.

“that you can see important dates at first” – I plan on having the user set priorities on certain events that they add to calendar that will notify the user when it’s coming close to the date in intervals (a week, 3 days, a day, etc).

7) What is something you'd like to see added to make a calendar better or more efficient?



“location-based heads up notification” – this is a particularly good suggestion but could make the project become quite difficult and end up being more about this instead of the calendar.

“repeating schedules” – another good suggestion as to appeals to people who have the same timetable every week or maybe for a worker who has a meeting at the same time on the same day every week. I will definitely think about adding this as a potential feature.

“ability to see all my tasks for the month at a glance” – I plan on making the view of the calendar adjustable so you can view a day, a week, a month or a whole year if that’s your preference.

“reminders that you can set minutes, hours, days” – this is an easy feature that I’ll add which can provide a lot of use to the user.

**General Overview of my objectives – 1.3**

* Create a calendar
* Social networking aspect – profiles – groups
* Database for profiles, groups and events
* List all events and groups and have some sort of search for them
* Login and register page
* Be able to edit and save user, group and event information.

**Specific overview of my objectives – 1.4**

Create a calendar

* Have a working calendar
* Shows the current day

Social networking aspect – profiles – groups

* Each user should have an individual profile and each group will have a separate page.
* No username will be the same
* Save the user and group information in SQL database

Database for profiles, groups and events

* I will use docker to hold the database in a container in order to decrease deployment time
* I’ll also use MySQL to code the database
* Ajax calls and node JS will be used to store and display the data

List all events and groups and have some sort of search for them

* Let the user search an exact name for an event or a group and their searched input comes back with results
* Uses ajax calls and node API
* Search results refresh the table at the press of a button

Login and register page

* Allows user to register a new account
* Allows user to log into their account
* Checks to see if the username and password are correct according to database
* Hashes the password for safety

Be able to edit and save profile, group and event information

* Users can change profile, group and event info after creating each of them
* Saving this information makes a change in the database so when the page is loaded up next the new data will be shown
* Users can also delete their account, event or group which removes it from the database

**Prototyping – 1.5**

Initially when starting the project, I had planned to write in HTML, PHP, SQL and a small bit of JavaScript but after a few tries I found PHP to not be the most effective language to use and instead changed to using JavaScript as the main language. This proved to be much more successful and found my code to be laid out better and be able to be understood easier. Prior to this project I had no JavaScript, PHP or HTML knowledge so there was no loss of time swapping from PHP to JavaScript. Having no prior knowledge did mean a lot of my time was spent on learning the languages but this made the project a very useful learning experience and by the end of it I could confidently use JavaScript and HTML.

Making a website meant I had to host the website and I had trouble finding a suitable way of doing this. At the start I was using WAMP which had a useful SQL GUI that allowed me to create the database easily and also had a connection to PHP, so it was useful at the time. But then I swapped computers and started doing my project on a mac which meant WAMP was no longer available, there was a mac version, but it was not user friendly and couldn’t run on my computer.

At the start I just hosted locally on my computer which was fine as the files were on my computer and they weren't going to be moved but when I was implementing cookies I found issues to arise. This meant I had to find another way to host my website. Eventually I found a solution, by running this command in terminal (python -m SimpleHTTPServer 8000) I was able to create a very basic http server, allowing cookies to be created which are vital for my website. This was required to be done in order to make use of cookies are it couldn’t be done when serving the HTML files directly through the browser.

I then used docker to deploy my SQL database to the cloud. Doing this had several advantages, for example it increased the speed to which I could run queries. The docker containers decrease deployment time to seconds which is very impressive, another advantage of using docker was the mobility, they can run anywhere.

|  |  |
| --- | --- |
| <?php  include("../../includes/databaseconnection.php");    $email = $\_POST['emailInput'];  $password = $\_POST['passwordInput'];  $sql = "INSERT INTO `users` (`UserID`, `Username`, `Password`, `Timestamp`) VALUES (NULL, '$email', '$password', CURRENT\_TIMESTAMP)";    if ($conn->query($sql) === TRUE) {  echo "New record created successfully";  } else {  echo "Error: " . $sql . "<br>" . $conn->error;  }    $conn->close();  ?> | This was my original PHP code to create a user.  Even though I ended up not using PHP for my final project this helped me learn about connecting to databases and implementing SQL into my code. |
| <?php  include("../../includes/databaseconnection.php");    $newemail = $\_POST['newemailInput'];  $userID = $\_POST['userIDInput'];  $sql = "UPDATE `users` SET `Username` = '$newemail' WHERE `users`.`UserID` = '$userID'";    if ($conn->query($sql) === TRUE) {  echo "New record updated successfully";  } else {  echo "Error: " . $sql . "<br>" . $conn->error;  }    $conn->close();  ?> | This was the original PHP code to update a user's information.  I did not like how php laid out the code and when I had added the rest of the functions my code would have been extremely hard to navigate. |
| <?php  include("../../includes/databaseconnection.php");    $sql = "SELECT \* FROM `users` WHERE `UserID`=52";    $result = $conn->query($sql);    if ($result->num\_rows > 0) {  // output data of each row  while($row = $result->fetch\_assoc()) {    echo " - Name: " . $row["Username"];  }  } else {  echo "0 results";  }    $conn->close();  ?> | Prototype php code to get the username of the user. In this code the UserID is just a static number so this function will only work for the UserID 52  Later in my code after I'd researched different ways to pass the UserID dynamically I use cookies which is a much more efficient way.  But this was still a useful learning experience. |
| <?php  $servername = "localhost";  $username = "server\_access";  //this is a local database and a strong password would be used when it's remote  $password = "12345";  $dbname = "group\_calendar";    // Create connection  $conn = new mysqli($servername, $username, $password, $dbname);    // Check connection  if ($conn->connect\_error) {  die("Connection failed: " . $conn->connect\_error);  }    ?> | This prototype PHP code is for the connection to the database.  Recreating the connection in JavaScript was extremely easy as it was similar to this PHP way. |
| ocument.querySelector('form.form-signin').addEventListener('submit', function (e) {  var emailInput = document.getElementById('inputEmail');  var passwordInput = document.getElementById('inputPassword');  //prevent the normal submission of the form  e.preventDefault();    console.log(emailInput.value);  console.log(passwordInput.value);  $.ajax({  type: "POST",  url: "./api/create\_user.php",  data: { 'emailInput': emailInput.value, 'passwordInput': passwordInput.value },  //dataType: 'json', this breaks everything look into why  success: function () {  console.log("it worked");  },  error: function () {  console.log("it did not work");  },  });  }); | This is a prototype of ajax calls. This was my code at the start of my project when I was getting used to using ajax calls and learning how to use them effectively.  This was very useful as seen in my final code, ajax calls are a vital part of how my whole project functions. |
| function getCookie(cname) {  var name = cname + "=";  var decodedCookie = decodeURIComponent(document.cookie);  var ca = decodedCookie.split(';');  for(var i = 0; i <ca.length; i++) {  var c = ca[i];  while (c.charAt(0) == ' ') {  c = c.substring(1);  }  if (c.indexOf(name) == 0) {  return c.substring(name.length, c.length);  }  }  return "";  } function setCookie(cname, cvalue, exdays) {  var d = new Date();  d.setTime(d.getTime() + (exdays\*24\*60\*60\*1000));  var expires = "expires="+ d.toUTCString();  document.cookie = cname + "=" + cvalue + ";" + expires + ";path=/";  }  $(document).ready(function(){  $("#create\_cookie").click(function(){  setCookie("UserID", 1, 2);  console.log(getCookie("UserID"))  });  });  $(document).ready(function(){  $("#create\_cookie2").click(function(){  setCookie("GroupID", 3, 2);  console.log(getCookie("GroupID"))  });  });  $(document).ready(function(){  $("#see\_cookie").click(function(){  console.log(getCookie("UserID"))  console.log("working")  var x = getCookie()  console.log(x)  });  });  $(document).ready(function(){  $("#delete\_cookie").click(function(){  console.log("this button works")  document.cookie = "UserID=; expires=Thu, 01 Jan 1970 00:00:00 UTC; path=/;";  });  }); | When learning about cookies it took me a while to become well versed in using them. In order to practice them I created a test.js and test.html to help me learn.  In the html I created multiple buttons then in the JavaScript I made functions connected to the buttons. I also used console log throughout the code to see how the results came out which I then could see how to handle them properly.  I also practiced deleting cookies, there didn’t seem to be a delete cookie function already so I just simply set the expiry date to sometime in the past so the cookie no longer existed. |

**Design–2**

**High level Overview – 2.1**

Main Menu – Flowchart

Diagram

Description automatically generated

Register - Flowchart

Diagram

Description automatically generated

Login – Flowchart

Diagram

Description automatically generated

Events– Flowchart

Diagram

Description automatically generated

Groups – Flowchart

Diagram

Description automatically generated

**Description of Modular Systems – 2.2**

When the application is launched you will be asked to either login or register an account then after successfully logging in the user will be shown the main menu. This diagram is the different pages that will be included.

Login

Register

Calendar

Profile

Edit Profile

Delete profile

Groups

Add new group

Edit Groups

Group page

Edit event

Delete event

Events

Add new event

Event page

Edit event

Delete event

Log Out

On every page there will be a drop down menu that will allow the user to travel to any of the main pages ( Events, Groups, Profile, Calendar )

**Data Dictionary – 2.3**

User Table

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Field Name | Field Type | Field Size | Purpose | Example Data | Validation |
| UserID | INT | 5 | Assigns a unique identifier to each user which will be used as a primary key. The number will just be automatically incremented each time a new account is registered. | 6 | Needs to be a whole number. |
| Username | VARCHAR | 15 | Stores the username of each account. | Luke | Starts with a letter, not empty, between 4 and 15 character |
| Email | VARCHAR | 50 | Stores the email of each account. | Xyz@email.com | Must be in the format xyz@email.com |
| Hashed\_password | VARCHAR | 150 | Stores each player’s password after it has been salt and hashed. | d1e8a70b5ccab1dc2f56bbf7e |  |
| ImgLoc | N/A | N/A | This will be used to save the location of the image for the profile picture. |  |  |
| FirstName | VARCHAR | 50 | Stores the users first name after they have inputted this information | John | Start with a letter |
| LastName | VARCHAR | 50 | Stores the users last name after they have inputted this information | Smith | Start with a letter |
| bio | VARCHAR | 150 | Stores the users bio after they have inputted this information | This is my long bio | None |
| Plaintext | VARCHAR | 50 | This is the users plaintext password, normally this wouldn’t be saved but I’ve done this so I can input the correct password when logging into multiple accounts | password | None |
| Created\_at | DATETIME | N/A | Stores the exact time that the account was created | 03-12-2012 12:23:16 | None |

Groups table

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Field Name | Field Type | Field Size | Purpose | Example Data | Validation |
| GroupID | INT | 5 | Assigns a unique identifier to each group which will be used as a primary key. The number will just be automatically incremented each time a new group is created. | 7 | Needs to be a whole number. |
| Created\_at | DATETIME |  | Stores the date that the group was first created | 2012- 05-16 12:24:16 | Needs to be in the correct format: YYYY – MM – DD hh:mm:ss |
| Group Name | VARCHAR | 15 | Stores the name of each Group. | Tesco Work. | Starts with a letter |
| bio | VARCHAR | 150 | Stores the Groups bio after they have inputted this information | This is my long bio | None |
| ImgLoc | N/A | N/A | This will be used to save the location of the image for the profile picture. |  |  |

GroupMembers table

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Field Name | Field Type | Field Size | Purpose | Example Data | Validation |
| GroupID | INT | 5 | Stores the ID of the group. Part of a composite key. | 6 | Needs to be a whole number. |
| UserID | INT | 5 | Stores the ID of the user who is in the group. Part of a composite key. | 12 | Needs to be a whole number. |

Events table

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Field Name | Field Type | Field Size | Purpose | Example Data | Validation |
| EventID | INT | 5 | Assigns a unique identifier to each Eventwhich will be used as a primary key. The number will just be automatically incremented each time a new Event is created. | 7 | Needs to be a whole number. |
| Created\_at | DATETIME |  | Stores the date that the group was first created | 2012- 05-16 12:24:16 | Needs to be in the correct format: YYYY – MM – DD hh:mm:ss |
| Event Name | VARCHAR | 15 | Stores the name of each Group. | Tesco Work. | Starts with a letter |
| bio | VARCHAR | 150 | Stores the Events bio after they have inputted this information | This is my long bio | None |
| ImgLoc | N/A | N/A | This will be used to save the location of the image for the profile picture. |  |  |
| TimeOfEvent | TIME | 50 | Saves the time that the event will be happening | 12:02:12 | Needs to be in the correct format: hh:mm:ss |
| DateOfEvent | DATE | 50 | Saves the date that the event will be happening | 12/12/2008 | Needs to be in the correct format: YYYY – MM – DD |
| LocationOfEvent | VARCHAR | 50 | Saves the location that the date will be happening | Local pub |  |

GroupMembers table

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Field Name | Field Type | Field Size | Purpose | Example Data | Validation |
| GroupID | INT | 5 | Stores the ID of the group. Part of a composite key. | 6 | Needs to be a whole number. |
| EventID | INT | 5 | Stores the ID of the Event, part of the composite key. | 12 | Needs to be a whole number. |

**Validation – 2.4**

|  |  |  |  |
| --- | --- | --- | --- |
| **Field Name** | **Validation Checks** | **Description** | **Error Message** |
| Username | 4 < Length <= 15  Matches the expression /^[A-Za-z][A-Za-z0-9]+$/ | Must be more than 4 characters and less than or equal to 15 characters.  Must start with a letter.  Can't have any spaces. | Usernames must be between 4 and 15 characters long.  Username must start with a letter and cannot contain any spaces. |
| Password | 8 <= Length <= 15  Matches the expression ^(?=.\*\d)(?=.\*[a-z])(?=.\*[A-Z])(?=.\*[^a-zA-Z0-9])(?!.\*\s).{8,15}$ | Must be between 8 and 15 characters  Must have a lowercase and uppercase letter.  Must have a number and a symbol. | Password must be between 8 and 15 characters longPassword must contain at least one lowercase letter, one uppercase letter, one numeric digit, and one special character |
| Email Address | Make sure email is in the format of [xyz@email.com](mailto:xyz@email.com) / .co.uk  Matches expression [\w-  ]+@([\w-]+\.)+[\w-]+ | Ensures email addresses are correctly formatted. | Emails must be in the format of [xyz@email.com](mailto:xyz@email.com) |

**Database Design 2.5**

The “User” table stores information on the users account, it stores their UserID, Username, Email, Hashed Password, plaintext password, first name, last name, bio, img loc and created at.

The “Groups” table stores information about the user created groups, it stores their GroupID, GroupName, bio, img loc and created at.

The “Events” table stores information about the user created groups, it stores their EventID, EventName, TimeOfEvent, DateOfEvent, LocationOfEvent, bio, img loc and created at.

Initially these were the only tables I had but there was a problem. A user could have many groups and a group could have many users, this meant that these tables had a many to many relationship which would bring about problems in integrity and problems when updating a user's or groups details.

In order to fix this problem, I had to create a new table to act as a linking table. I made the table “GroupMembers”, this is table that consisted of UserID and GroupID. The primary key of Member\_List is a composite key consisting of UserID and GroupID.

Just like the problem between the Users and the Groups table, Events and Groups have the same problem. So to fix this I created a linking table called EventGroups. The primary key of this table is a composite key consisting of GroupID and EventID.

Diagram

Description automatically generated

**Planned SQL Queries 2.6**

Here are some examples of SQL queries I plan to use.

**Adding a new user into the database after they have registered:**

INSERT INTO users(UserID, Username, Email, Hashed\_password, Salt)

VALUES (NULL, “[Luke”,”Luke@email.com](mailto:Luke), “sags4wv2q54”, “afgva424x2xd”

The value for UserID is null as the user won’t be inputting that as it will be automatically incremented each time a new user is created.

**Adding a new Group into the database**

INSERT INTO Groups(GroupID, , DateStarted, GroupName)

VALUES (NULL, 2018-07-12, “Tesco Work”)

The value for GroupID is null as the user won’t be inputting that as it will be automatically incremented each time a new user is created.

**Adding a member to a group**

INSERT INTO GroupMembers( UserID, GroupID)

VALUES (12, 13)

**User is updating their password**

UPDATE Users

SET Hashed\_password = “AAAAAAA”, SALT = “BBBBBBB”

WHERE UserID = 6

AAAAAAA is the new hashed password of the user and BBBBBBB is the new randomly generated salt.

**Searching for a user :**

SELECT \*

FROM Users

WHERE Username = “Luke”

**Searching for a Group to join**

SELECT \*

FROM Groups

WHERE GroupName = “Waitrose Work”

**Deleting an account:**

DELETE FROM Users

WHERE UserID = 14

**Deleting a group:**

DELETE FROM Groups

WHERE GroupID = 4

**Deleting an Event:**

DELETE FROM Events

WHERE EventID = 4

**Prototype SQL for creating tables**

CREATE DATABASE `group\_calendar`

CREATE TABLE group\_calendar.Users (

UserID INT(6) NOT NULL AUTO\_INCREMENT PRIMARY KEY,

Username VARCHAR(50),

User\_Password VARCHAR(50),

FirstName VARCHAR(30) NOT NULL,

LastName VARCHAR(30) NOT NULL,

Email VARCHAR(50),

Bio VARCHAR(50),

ImgLoc VARCHAR(50),

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP

)

CREATE TABLE group\_calendar.Groups (

GroupID INT(6) NOT NULL AUTO\_INCREMENT PRIMARY KEY,

GroupName VARCHAR(50) NOT NULL,

Bio VARCHAR(50),

ImgLoc VARCHAR(50),

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP

)

CREATE TABLE group\_calendar.Events (

EventID INT(6) NOT NULL AUTO\_INCREMENT PRIMARY KEY,

EventName VARCHAR(50),

TimeOfEvent DATETIME,

DateOfEvent DATE,

LocationOfEvent VARCHAR(30),

Bio VARCHAR(50),

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP

)

CREATE TABLE group\_calendar.GroupMembers (

UserID INT(6) NOT NULL AUTO\_INCREMENT PRIMARY KEY,

GroupID INT(6) NOT NULL,

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP

)

CREATE TABLE group\_calendar.EventGroups (

EventID INT(6) NOT NULL AUTO\_INCREMENT PRIMARY KEY,

GroupID INT(6) NOT NULL,

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP

)

**UI Design 2.7**

**Login/Register Page**

Graphical user interface, application

Description automatically generated

The is the first screen that will be loaded up after the application is launched. One of the main features of this project is the profiles and groups so therefore each user must have a unique login which gives them a unique UserID. This unique ID will be saved to a cookie which will then be used to keep the user signed into their account. If at any time the cookie is deleted or the user logs out then the user will be redirected to this page .

The “Submit” button will send the information to be checked on the database and if the username and password is the same as the data in the database they will be logged in and redirectly to their profile page where all their information will be displayed. If their account has just been registered, then their profile page will be empty.

Graphical user interface, application

Description automatically generated

If the user has entered the wrong credentials, then this warning will come up on their screen telling them they’ve entered the incorrect password or username. They will be able to try and log in again.

Graphical user interface, application

Description automatically generated

This is the screen the user will get shown after they’ve entered a new username and a password of their choosing. They will be told to then log in. Registering adds their username and password into the database which will allow them to log in as their credentials can be checked.

**The pull-down bar**

Graphical user interface, application, Teams

Description automatically generated

This can be pulled down on any page other than the login/register page. This can be done by clicking the button with the three lines in the top right-hand side. After clicking it you see the four main pages which you can get to any of them by clicking their button.

At the top of the pull-down bar you can see the “Log out” button. Since this pull-down bar is on every page you can log out from any page. When you click this button your cookie holding your UserID is deleted, and you are redirected to the login/register page. As your cookie is deleted you need to log in again whenever this log out button is clicked.

**Profile Page**

Graphical user interface, text, application

Description automatically generated

After logging into their account, the user will be presented with their personal profile page which will show all of their information and their groups. If the user has just been registered then they will be shown a screen like this one but with no information in the boxes.

If the user would like to change their information or in the case of a new account add their information then they need to press the edit button.

Graphical user interface, application, Teams

Description automatically generated

This is what the screen looks like when the edit button is pressed. The edit button disappears and three buttons appear, the “save”, “close” and “Delete” buttons. Also, the boxes that were initially greyed out are now white and can be edited. The user is allowed to change their information to whatever they want. Once the user is done changing the data in the boxes they can click save and by doing this an api call will happen which will update the database and their new information will be saved. Alternatively the user can click delete which will delete the users account and their cookie holding their UserID and then redirect them to the login/register page. Finally, by clicking the “close” button the page will return to how it was originally and will hide the 3 new buttons and show the “edit” button again and the boxes will no longer be able to be edited.

**Groups Page**

Graphical user interface, application

Description automatically generated

When you click the “Groups” button on the drop-down menu you will be redirected to this page. On this page all the groups that have been created are displayed in table form. In the top left of the screen there is a box where the user can search for a certain groups or groups beginning with whatever they input.

Graphical user interface, application, Teams

Description automatically generated

When the user searches “tes” it returns with the two groups that start with the user's input. When the user searches for the group's name exactly it will lead the user to that groups group page. The group they search for will have its GroupID saved to a cookie which will be used to get to its unique group page.

If the user made a mistake with their search or they want to see all the groups again they can click the reset button which will reload the page therefore display the full table showing all of the groups information.

When the user clicks the “Create Group” page a new group is created and a cookie is set to the new GroupID of the group. The user is also instantly redirected to the new groups group page.

**Group Page**

Graphical user interface, application, Teams

Description automatically generated

After the user has searched for an individual group or they have created a new group they will be redirected to this page. Similar to the profiles each group has their own separate page that displays their Group name, Bio and Users in the group. Just like the profile page you can click edit and then the three buttons, “save”, “close” and “Delete” will appear with the “Edit” button disappearing.

Pressing the Join button will add the user to the group by utilising the linking table GroupMembers. Vice versa with the Leave button.

Graphical user interface, application, Teams

Description automatically generated

Pressing the delete button will delete the group from the database and redirect you to the groups page displaying the groups. When the Edit button has been clicked you can change the information of the group in the boxes as you will be able to write in them now. Once you have finished editing the boxes the user will press save which will update the groups information on the database with the new user inputted data. When the user is done, they can press the “Close” button which will change the screen back to how it was.

**Calendar**

A picture containing text, wall, white

Description automatically generated

This is a simple calendar that displays all the months and the year. The red box highlights the current day.

Icon

Description automatically generated

By pressing either the left or the right arrow you can go to the next month or the month before. The calendar keeps going and won’t stop just because you're at the next year as seen in this photo.

A picture containing wall, white

Description automatically generated

**Events Page**

Graphical user interface

Description automatically generated

This is the Events page which you can get to by clicking the “Events” button on the drop-down menu. Similar to the Groups page it displays all of the events and each event has its own unique event page as each event is given an EventID. The search function works the same as how the Groups page does.

Graphical user interface, application

Description automatically generated

As we see in the photo once the user has typed something in the search bar and pressed the “search” button the table is updated to only display events that start with the letters that the user has inputted. If they want to go back and see all of the events then they can press the “Reset” button which will reload the page and will show them all of the events displayed in table form again.

If the user wants to create a new event they can press the “Add Event” button. This will run an SQL query to create a new event. A cookie will also be set to the new events EventID which is used to load the events unique event page.

**Event Page**

Graphical user interface, application, Teams

Description automatically generated

After searching for the event or creating a new event you will be redirected to the event page. This shows the events name, date of the event, and the bio of the event. There is also an “Edit” button that when clicking hides the “Edit” button the shows three new buttons.

Graphical user interface, application, Teams

Description automatically generated

Here is what happens after you click the “Edit” button, as seen the “Edit” button is hidden while the “Save”, “Close” and “Delete” button appear. At the same time the three text boxes that were originally greyed out are now white and can be edited. After you have changed the information, you can then press the “Save” button which will send an SQL query to update the events data in the database. You can then press the “Close” button which will hide. The three new buttons and show the “Edit” button again, the three text boxes will also be greyed out and the user will no longer be able to change the text. The “Delete” button when clicked will send an SQL query to delete the event. This will clear the cookie for EventID and also redirect you to the Events page showing all the events left.

**Security – 2.8**

A main part of how I keep the accounts secure is by hashing the passwords from when they are registered. This is done in case of a breach in the database the hacker will only be able to grab the hashed password which on its own is useless as they don’t know the salt or the plaintext password.

In my users table in the database I hold the plaintext password here however in a normal project this would never be done, I've only done this because I need it to check I'm putting the right password in when I'm logging in.

In order to hash the passwords I used a library call bcrypt, to get this I had to run this command in the command line: npm install bcrypt

After installing it I could use the simple hashing command : bcrypt.genSalt(saltRounds, function(err, salt) {

bcrypt.hash(myPlaintextPassword, salt, function(err, hash) {

*// Store hash in your password DB.*

});

});

I decided to not create a hashing function from scratch because it would’ve taken up a lot of my time on something that would have only been a tiny part of my project, so I used a library because I still wanted it to be

**Technical Solution – 3**

|  |  |
| --- | --- |
| **Technique/Algorithm** | **Name of file** |
| Aggregate SQL functions | Database\_api.js |
| calling parameterised web service APIs and parsing JSON/XML to serice a complex client-server model | Database\_api.js |
| Hashing | Database\_api.js |
| Cross table paramatised SQL | Database\_api.js |
| server side scripting using request and response objects and server-side extensions for complex client-server model | Database\_api.js |
| User generated DDL script | Create\_tables.sql |
| Restful api | Database\_api.js |
| JSON parsing | Database\_api.js and profile.js |
| Multi-dimensional arrays | Groups.js |
| Dictionaries / objects | Groups.js |

**First, I create the MySQL Docker to hold my database in a container.**

**All these commands are put into the terminal**

**### Create mySQL Docker DB**

docker run --name test-mysql -p 3306 -e MYSQL\_ROOT\_PASSWORD=my-secret-pw -d mysql:latest

**### Check which port mySQL Docker DB is running on**

docker ps

**### Log into mySQL Docker DB**

mysql -P 32768 --protocol=tcp -u root -p

**Then I run the database code in the terminal as well after connecting to Docker.**

**Create\_tables.sql**

CREATE DATABASE `group\_calendar`;

CREATE TABLE group\_calendar.Users (

UserID INT(6) NOT NULL AUTO\_INCREMENT PRIMARY KEY,

Username VARCHAR(50) UNIQUE,

User\_Password VARCHAR(150),

FirstName VARCHAR(30),

LastName VARCHAR(30),

Email VARCHAR(50),

Bio VARCHAR(50),

ImgLoc VARCHAR(50),

Plaintext VARCHAR(150),

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP

);

CREATE TABLE group\_calendar.Groups (

GroupID INT(6) NOT NULL AUTO\_INCREMENT PRIMARY KEY,

GroupName VARCHAR(50) NOT NULL,

Bio VARCHAR(50),

ImgLoc VARCHAR(50),

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP

);

CREATE TABLE group\_calendar.Events (

EventID INT(6) NOT NULL AUTO\_INCREMENT PRIMARY KEY,

EventName VARCHAR(50),

TimeOfEvent DATETIME,

DateOfEvent DATE,

LocationOfEvent VARCHAR(30),

Bio VARCHAR(50),

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP

);

CREATE TABLE group\_calendar.GroupMembers (

UserID INT(6) NOT NULL ,

GroupID INT(6) NOT NULL,

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

PRIMARY KEY (UserID, GroupID)

);

CREATE TABLE group\_calendar.EventGroups (

EventID INT(6) NOT NULL,

GroupID INT(6) NOT NULL,

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

PRIMARY KEY (EventID, GroupID)

);

**This command is then run on a separate terminal tab in order to set up a http server to run my website on.**

python -m SimpleHTTPServer 8000

**Database\_api.js this is all node js**

var express = require('express')

var app = express()

var mysql = require('mysql');

var cors = require('cors')

var bodyParser = require('body-parser')

var express = require('express');

var session = require('express-session');

var path = require('path');

const jwt = require("jsonwebtoken");

const bcrypt = require('bcrypt');

const saltRounds = 3;

var app = express();

app.use(cors())

const Connection = require('mysql/lib/Connection');

app.use(session({

secret: 'secret',

resave: true,

saveUninitialized: true

}));

app.use(bodyParser.urlencoded({ extended: true }))

app.use(bodyParser.json())

// connecting to the sql database

var con = mysql.createConnection({

host: "localhost",

port:"32768",

user: "root",

password: "my-secret-pw",

database: "group\_calendar"

});

//quick function to test if successful connection

con.connect(function(err) {

if (err) throw err;

console.log("Connected!");

});

//using the usersID to grab the rest of their data

app.get('/Users/:UserID', function(request, response, next) {

con.query('SELECT \* FROM Users WHERE UserID = ' + request.params.UserID, function(err, result, fields)

{

if (err) throw err;

return response.send(result[0]);

});

});

const jwtkey = "insertprivatekeyhere";

const jwtexpiry = 35000;

// the login which also hashes the users plaintext password and compares the hash to the database

app.post('/Login', function(request, response) {

var Username = request.body.Username;

var User\_Password = request.body.User\_Password;

//bcrypt.compare(InputPassword, hash, function(err, res) {

bcrypt.hash(User\_Password, saltRounds, (err, hash) => {

con.query('SELECT \* FROM Users WHERE Username = \''+Username+'\' AND User\_Password = \''+hash+'\'', function(err, res, fields) {

request.session.loggedin = true;

request.session.username = Username;

const token = jwt.sign({

Username}, jwtkey,{

algorithm: "HS256",

expiresIn: jwtexpiry

});

console.log(token)

console.log(verify(token))

response.send(JSON.stringify({"message": "You are logged in", "loggedin": "true", "token": token}))

});

});

});

// the login which also hashes the users plaintext password and compares the hash to the database

app.get('/Login2/:Username', function(request, response) {

var Username = request.body.Username;

var User\_Password = request.body.User\_Password;

con.query('SELECT \* FROM `Users` WHERE Username = \''+request.params.Username+'\'', function(err, result, fields)

{

if (err) throw err;

console.log(result[0])

if (result.length > 0) {

return response.send(result[0]);

console.log(result[0])

response.send(JSON.stringify({"message": "You are logged in", "loggedin": "true", "result": result}))

return response.send(result);

} else {

response.send(JSON.stringify({"message": "Incorrect Username and/or Password!", "loggedin": "false"}));

}

});

});

//function to verify the jwt token

function verify(token){

var verifiedJwt

try {

verifiedJwt = jwt.verify(token,jwtkey);

} catch(err) {

//change this to catch specific errors

return res.status(400).send('invalid token')

}

return verifiedJwt.Username

};

function GetID2(Username){

var result

con.query('SELECT UserID FROM `Users` WHERE Username = \''+Username+'\'', function(err, result, fields)

{

if (err) throw err;

console.log("UserID2:"+parseInt(result[0].UserID));

return (result[0]);

});

}

//register api that adds the users username and password into the database - also hashes the password and saves the hash

app.post('/Register', function(request, response, next) {

var Username = request.body.Username;

var User\_Password = request.body.User\_Password;

bcrypt.genSalt(saltRounds, (err, salt) => {

bcrypt.hash(User\_Password, salt, (err, hash) => {

// Now we can store the password hash in db.

console.log(hash)

if (Username == "") {

return response.send("There has been an error in your username")

}

con.query('INSERT INTO `Users` (`Username`, `User\_Password`, `FirstName`, `LastName`, `email`, `bio`, `Plaintext`, `created\_at`) VALUES ( \''+Username+'\', \''+hash+'\', NULL , NULL, NULL, NULL, \''+User\_Password+'\', CURRENT\_TIMESTAMP);', function(err, result, fields)

{

console.log(err)

if (err) throw err;

return response.send(result[0]);

});

console.log("error!!!!!!!!!!!");

});

});

});

//getting the newest group mades data

app.get('/NewestGroup', function(request, response, next) {

con.query('SELECT \* FROM `Groups` ORDER BY created\_at DESC limit 1', function(err, result, fields)

{

console.log(err)

if (err) throw err;

return response.send(result[0]);

});

console.log("error!!!!!!!!!!!");

});

//creating a new group

app.post('/NewGroup', function(request, response, next) {

con.query('INSERT INTO `Groups` ( `GroupName`, `bio`, `ImgLoc`, `created\_at`) VALUES ( "placeholder", NULL, NULL, CURRENT\_TIMESTAMP);', function(err, result, fields)

{

console.log(err)

if (err) throw err;

return response.send(result[0]);

});

console.log("error!!!!!!!!!!!");

});

//joining a group

app.post('/JoinGroup/:UserID/:GroupID', function(request, response, next) {

con.query('INSERT INTO `GroupMembers` ( `UserID`, `GroupID`) VALUES (' + request.params.UserID + ', ' + request.params.GroupID + ');', function(err, result, fields)

{

console.log(err)

if (err) throw err;

return response.send(result[0]);

});

console.log("error!!!!!!!!!!!");

});

//getting the users in group

app.get('/GroupMembers/:GroupID', function(request, response, next) {

con.query('select UserID from GroupMembers where GroupID =' +request.params.GroupID, function(err, result, fields)

{

console.log(err)

if (err) throw err;

return response.send(result[0]);

});

console.log("error!!!!!!!!!!!");

});

//deleting a group

app.post('/DeleteGroup/:GroupID', function(request, response, next) {

con.query('DELETE FROM `Groups` WHERE GroupID =' + request.params.GroupID, function(err, result, fields)

{

console.log(err)

if (err) throw err;

});

console.log("error!!!!!!!!!!!");

});

//deleting a User

app.post('/DeleteUser/:UserID', function(request, response, next) {

con.query('DELETE FROM `Users` WHERE UserID =' + request.params.UserID, function(err, result, fields)

{

console.log(err)

if (err) throw err;

});

console.log("error!!!!!!!!!!!");

});

//deleting an event

app.post('/DeleteEvent/:EventID', function(request, response, next) {

con.query('DELETE FROM `Events` WHERE EventID =' + request.params.EventID, function(err, result, fields)

{

console.log(err)

if (err) throw err;

});

console.log("error!!!!!!!!!!!");

});

//UPDATING USER INFORMATION

app.put('/Users/:UserID', function(request, response) {

var FirstName = request.body.FirstName;

var LastName = request.body.LastName;

var Email = request.body.Email;

var Bio = request.body.Bio;

con.query('UPDATE Users SET FirstName = \''+FirstName+'\', Lastname = \''+LastName+'\',Email = \''+Email+'\', Bio= \''+Bio+'\' WHERE UserID = '+ request.params.UserID +';', function(err, result, fields)

{

if (err) throw err;

//return response.send(result[0]);

return response.send("success");

});

});

//GET all groups a users is in

app.get('/Users/Groups/:UserID', function(request, response, next) {

con.query('SELECT Groups.GroupName, Groups.GroupID, Groups.Bio, Groups.ImgLoc FROM `Users`, `Groups` WHERE UserID = ' + request.params.UserID, function(err, result, fields)

{

if (err) throw err;

return response.send(result[0]);

});

});

//GET all groups

app.get('/AllGroups', function(request, response, next) {

con.query('SELECT \* FROM `Groups`', function(err, result, fields)

{

if (err) throw err;

return response.send(result);

});

});

//GET all searched groups

app.get('/SearchGroups/:Search', function(request, response, next) {

var Search = request.body.Search;

con.query('SELECT \* FROM `Groups` WHERE GroupName LIKE "' + request.params.Search + '%"', function(err, result, fields)

{

if (err) throw err;

return response.send(result);

});

});

//GET all events

app.get('/AllEvents', function(request, response, next) {

con.query('SELECT \* FROM `Events`', function(err, result, fields)

{

if (err) throw err;

return response.send(result);

});

});

//GET all searched Events

app.get('/SearchEvents/:Search', function(request, response, next) {

var Search = request.body.Search;

con.query('SELECT \* FROM `Events` WHERE EventName LIKE "' + request.params.Search + '%"', function(err, result, fields)

{

if (err) throw err;

return response.send(result);

});

});

//getting the newest event mades data

app.get('/NewestEvent', function(request, response, next) {

con.query('SELECT \* FROM `Events` ORDER BY created\_at DESC limit 1', function(err, result, fields)

{

console.log(err)

if (err) throw err;

return response.send(result[0]);

});

console.log("error!!!!!!!!!!!");

});

//creating a new event

app.post('/NewEvent', function(request, response, next) {

con.query('INSERT INTO `Events` ( `EventName`, `TimeOfEvent`, `DateOfEvent`, `LocationOfEvent`, `Bio`, `created\_at`) VALUES ( "placeholder", NULL, NULL, NULL, NULL, CURRENT\_TIMESTAMP);', function(err, result, fields)

{

console.log(err)

if (err) throw err;

return response.send(result[0]);

});

console.log("error!!!!!!!!!!!");

});

//GET all users in a group

app.get('/Groups/Users/:GroupID', function(request, response, next) {

con.query('SELECT Users.Username, Users.Bio, Users.ImgLoc, Users.UserID FROM `Users`, `Groups` WHERE GroupID = ' + request.params.GroupID, function(err, result, fields)

{

if (err) throw err;

return response.send(result[0]);

});

});

//GET event information from their ID

app.get('/Events/:EventID', function(request, response, next) {

con.query('SELECT \* FROM Events WHERE EventID = ' + request.params.EventID, function(err, result, fields)

{

if (err) throw err;

return response.send(result[0]);

});

});

//updating event information

app.put('/Events/:EventID', function(request, response) {

var EventName = request.body.EventName;

var Bio = request.body.Bio;

var DateOfEvent = request.body.DateOfEvent;

var LocationOfEvent = request.body.LocationOfEvent;

con.query('UPDATE Events SET EventName = \''+EventName+'\', Bio= \''+Bio+'\' WHERE EventID = '+ request.params.EventID +';', function(err, result, fields)

{

if (err) throw err;

//return response.send(result[0]);

return response.send("success");

});

});

//UPDATING GROUP INFORMATION

app.put('/Groups/:GroupID', function(request, response) {

var GroupName = request.body.GroupName;

var Bio = request.body.Bio;

con.query('UPDATE `Groups` SET GroupName = \''+GroupName+'\', Bio= \''+Bio+'\' WHERE GroupID = '+ request.params.GroupID +';', function(err, result, fields)

{

if (err) throw err;

//return response.send(result[0]);

return response.send("success");

});

});

//GET group information from GroupID

app.get('/Groups/:GroupID', function(request, response, next) {

con.query('SELECT \* FROM `Groups` WHERE GroupID = ' + request.params.GroupID, function(err, result, fields)

{

if (err) throw err;

return response.send(result[0]);

});

});

//ALTER USER 'root' IDENTIFIED WITH mysql\_native\_password BY 'my-secret-pw';

**This command in the // can be ran on terminal in a separate tab to check how all the node is doing and if an error comes up then it will show on the terminal tab.**

//DEBUG=myapp:\* npm start

app.listen(3000, function () {

console.log("Listening on port 3000. Go to <http://localhost:3000>");

});

**Login.html**

<!doctype html>

<html lang="en">

<head>

<meta charset="utf-8">

<meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">

<meta name="description" content="">

<meta name="author" content="">

<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>

<!-- Bootstrap core CSS -->

<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/css/bootstrap.min.css" integrity="sha384-Gn5384xqQ1aoWXA+058RXPxPg6fy4IWvTNh0E263XmFcJlSAwiGgFAW/dAiS6JXm" crossorigin="anonymous">

<script src="https://code.jquery.com/jquery-3.1.1.min.js">

</script>

<script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.12.9/umd/popper.min.js" integrity="sha384-ApNbgh9B+Y1QKtv3Rn7W3mgPxhU9K/ScQsAP7hUibX39j7fakFPskvXusvfa0b4Q" crossorigin="anonymous"></script>

<script src="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/js/bootstrap.min.js" integrity="sha384-JZR6Spejh4U02d8jOt6vLEHfe/JQGiRRSQQxSfFWpi1MquVdAyjUar5+76PVCmYl" crossorigin="anonymous"></script>

<!-- Custom styles for this template -->

<link href="css/style.css" rel="stylesheet">

<script type="text/javascript" src="js/login.js"></script>

<body class="text-center">

<form class="form-signin">

<h1 class="h3 mb-3 font-weight-normal">Login / Register</h1>

<label for="inputEmail" class="sr-only">Email address</label>

<input type="email" id="inputEmail" class="form-control" placeholder="Email address" required autofocus>

<label for="inputPassword" class="sr-only">Password</label>

<input type="password" id="inputPassword" class="form-control" placeholder="Password" required>

<div class="checkbox mb-3">

</div>

<button class="btn btn-lg btn-primary btn-block" id="sign\_in\_button" type="button">Sign in</button>

<button class="btn btn-lg btn-secondary btn-block" id= "register\_button" type="button">Register</button>

<h1 class="h4 mb-3 font-weight-normal" id = "registered">Successfully Registered Account - now log in</h1>

<h1 class="h4 mb-3 font-weight-normal" id = "incorrect">Incorrect Password or Username</h1>

<p class="mt-5 mb-3 text-muted">&copy; 2020-2021</p>

</form>

</body>

</html>

**Login.js**

$(document).ready(function() {

$('#registered').hide();

$('#incorrect').hide();

});

//inputing correct data then pressing the register button inserts a new user into the users table

$(document).ready(function(){

$("#register\_button").click(function(){

var Username = document.getElementById('inputEmail').value

var User\_Password = document.getElementById('inputPassword').value

console.log(Username);

$.ajax({

type: "POST",

url: "<http://localhost:3000/Register>",

data: {

"Username": Username,

"User\_Password": User\_Password

},

crossDomain : true,

success: function (res) {

console.log("it worked");

$('#registered').show();

},

error: function (res, err) {

console.log("it did not work");

},

});

});

});

//function to get whats stored inside the cookie

function getCookie(cname) {

var name = cname + "=";

var decodedCookie = decodeURIComponent(document.cookie);

var ca = decodedCookie.split(';');

for(var i = 0; i <ca.length; i++) {

var c = ca[i];

while (c.charAt(0) == ' ') {

c = c.substring(1);

}

if (c.indexOf(name) == 0) {

return c.substring(name.length, c.length);

}

}

return "";

}

function setCookie(cname, cvalue, exdays) {

var d = new Date();

d.setTime(d.getTime() + (exdays\*24\*60\*60\*1000));

var expires = "expires="+ d.toUTCString();

document.cookie = cname + "=" + cvalue + ";" + expires + ";path=/";

}

//prototype data when first using cookies

$(document).ready(function(){

$("#create\_cookie").click(function(){

setCookie("UserID", 1, 2);

console.log(getCookie("UserID"))

});

});

// checking the inputted username and password that the user entered to the data in the database then letting the user into their profile if the data is correct

$(document).ready(function(){

$("#sign\_in\_button").click(function(){

var Username = document.getElementById('inputEmail').value

var User\_Password = document.getElementById('inputPassword').value

console.log(Username);

$.ajax({

type: "GET",

url: "<http://localhost:3000/Login2/>" + Username ,

dataType: 'json',

crossDomain : true,

data: {

},

success: function (res) {

console.log("it worked");

if (res.loggedin == "false") {

console.log("fail")

//displays the incorrect message to the user if they have entered the wrong username or password

$('#registered').hide();

$('#incorrect').show();

} else {

if (res.Plaintext == User\_Password) {

console.log("success")

setCookie("UserID", res.UserID, 2);

$('#registered').hide();

$('#incorrect').hide();

window.location.href = "<http://localhost:8000/profile.html>"

} else{

$('#incorrect').show();

}

}

},

error: function (res, err) {

console.log("it did not work");

console.log(res);

console.log(err);

},

});

});

});

**Profile.html**

<!doctype html>

<html lang="en">

<head>

<meta charset="utf-8">

<meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">

<meta name="description" content="">

<meta name="author" content="">

<title>Profile</title>

<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>

<!-- Bootstrap core CSS -->

<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/css/bootstrap.min.css" integrity="sha384-Gn5384xqQ1aoWXA+058RXPxPg6fy4IWvTNh0E263XmFcJlSAwiGgFAW/dAiS6JXm" crossorigin="anonymous">

<script src="https://code.jquery.com/jquery-3.1.1.min.js"></script>

<script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.12.9/umd/popper.min.js" integrity="sha384-ApNbgh9B+Y1QKtv3Rn7W3mgPxhU9K/ScQsAP7hUibX39j7fakFPskvXusvfa0b4Q" crossorigin="anonymous"></script>

<script src="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/js/bootstrap.min.js" integrity="sha384-JZR6Spejh4U02d8jOt6vLEHfe/JQGiRRSQQxSfFWpi1MquVdAyjUar5+76PVCmYl" crossorigin="anonymous"></script>

<!-- Custom styles for this template -->

<link href="css/Profile.css" rel="stylesheet">

<script type="text/javascript" src="js/profile.js"></script>

</head>

<body>

<nav class="navbar navbar-dark bg-dark">

<a class="navbar-brand" href="#">Logo</a>

<button type="button" id="log\_out" class="btn btn-danger">Log out</button>

<button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#navbarsExample01" aria-controls="navbarsExample01" aria-expanded="false" aria-label="Toggle navigation">

<span class="navbar-toggler-icon"></span>

</button>

<div class="collapse navbar-collapse" id="navbarsExample01">

<ul class="navbar-nav mr-auto">

<li class="nav-item active">

<a class="nav-link" href="profile.html">MyProfile <span class="sr-only">(current)</span></a>

</li>

<li class="nav-item">

<a class="nav-link" href="Calendar.html">MyCalendar</a>

</li>

<li class="nav-item">

<a class="nav-link" href="Groups.html">Groups</a>

</li>

<li class="nav-item">

<a class="nav-link" href="Events.html">Events</a>

</div>

</li>

</ul>

</div>

</nav>

<main role="main" class="container">

<div class="my-3 p-3 bg-white rounded box-shadow">

<h6 class="border-bottom border-gray pb-2 mb-0">MyProfile</h6>

<input type="email" class="form-control" id="username" aria-describedby="emailHelp" placeholder="Enter First Name" disabled>

<div class="container" id="profile\_details">

<div class="row">

<div class="col-4 text-center align-items-center" id="profile\_image">

<div class="pt-4">

<div class="">

<img class="rounded-circle" src="./img/download.png" alt="Generic placeholder image" width="140" height="140">

</div>

<div>

<button type="button" id="upload\_pic" class="btn btn-dark" >Upload</button>

</div>

</div>

</div>

<div class="col-8" id="profile\_name">

<form>

<div class="form-group">

<label for="first\_name">First Name</label>

<input type="email" class="form-control" id="first\_name" aria-describedby="emailHelp" placeholder="Enter First Name" disabled>

</div>

<div class="form-group">

<label for="second\_name">Second Name</label>

<input type="email" class="form-control" id="last\_name" aria-describedby="emailHelp" placeholder="Enter Last Name" disabled>

</div>

<div class="form-group">

<label for="exampleInputEmail">Email address</label>

<input type="email" class="form-control" id="email" aria-describedby="emailHelp" placeholder="Enter Email" disabled>

</div>

</div>

</div>

<div class="row">

<div class="col">

<div class="form-group">

<label for="exampleFormControlTextarea1">Bio</label>

<textarea class="form-control" id="bio" rows="3" disabled></textarea>

</div>

</div>

</div>

<div class="row">

<div class="col">

<div id="edit\_page" class="text-right">

<button type="button" id="edit\_button" class="btn btn-dark">Edit</button>

</div>

<div id="save\_close\_delete\_button" class="text-right">

<button type="button" id="save\_button" class="btn btn-dark">Save</button>

<button type="button" id="close\_button" class="btn btn-dark">Close</button>

<button type="button" id="delete\_button" class="btn btn-danger">Delete</button>

</div>

</div>

</div>

<div class="my-3 p-3 bg-white rounded box-shadow">

<h6 class="border-bottom border-gray pb-2 mb-0">MyGroups</h6>

<div class="list-group">

<a href="#" class="list-group-item list-group-item-action flex-column align-items-start">

<div class="row">

<div class="col col-lg-2" id="profile\_image">

<div class="pt-4">

<div class="">

<img class="rounded-circle" src="./img/download.png" alt="Generic placeholder image" width="140" height="140">

</div>

</div>

</div>

<div class="col" id="group\_info">

<form>

<div class="form-group">

<input type="group\_name" class="form-control" id="group\_name" aria-describedby="emailHelp" placeholder="" disabled>

</div>

<div class="form-group">

<textarea class="form-control" id="group\_bio" rows="3" disabled></textarea>

</div>

</div>

<div class="col-md-auto">

<button type="button" id="leave\_button" class="btn btn-dark">-</button>

</div>

</div>

</a>

<a href="#" class="list-group-item list-group-item-action flex-column align-items-start">

<div class="row">

<div class="col col-lg-2" id="profile\_image">

<div class="pt-4">

<div class="">

<img class="rounded-circle" src="./img/download.png" alt="Generic placeholder image" width="140" height="140">

</div>

</div>

</div>

<div class="col" id="group\_bio">

<form>

<h5 class="text-center">Group Name</h5>

<p class="mb-1">Lorem Ipsum is simply dummy text of the printing and

typesetting industry. Lorem Ipsum has been the industry's

standard dummy text ever since the 1500s, when an unknown

printer took a galley of type and scrambled it to make a

type specimen book. It has survived not only five centuries,

but also the leap into electronic typesetting,

remaining essentially unchanged. </p>

</div>

<div class="col-md-auto">

<button type="button" id="leave\_button" class="btn btn-dark">-</button>

</div>

</div>

</a>

</div>

</div>

</main>

</body>

</html>

**Profile.js**

$(document).ready(function() {

$('#save\_close\_delete\_button').hide();

$('#upload\_pic').hide();

});

function toggle\_readonly(state){

document.getElementById("email").disabled = state;

document.getElementById("first\_name").disabled = state;

document.getElementById("last\_name").disabled = state;

document.getElementById("bio").disabled = state;

}

//Checks to make sure the user has a cookie and logs them out if they dont

$(document).ready(function() {

if (getCookie("UserID") < 1){

window.location.href = "<http://localhost:8000/login.html>"

}

});

//updates the database with the user inputted data

$(document).ready(function(){

var userid = getCookie("UserID");

console.log(userid)

$("#save\_button").click(function(){

var fname = document.getElementById('first\_name').value

var email = document.getElementById("email").value;

var lname = document.getElementById("last\_name").value;

var bio = document.getElementById("bio").value;

console.log("this will save information");

console.log(fname);

var requestbody ={

"FirstName": document.getElementById('first\_name').value,

"Email" : document.getElementById("email").value,

"LastName" : document.getElementById("last\_name").value,

"Bio" : document.getElementById("bio").value

}

$.ajax({

type: "PUT",

url: "[http://localhost:3000/Users/"+userid](http://localhost:3000/Users/%22+userid),

contentType: 'application/json',

// data: JSON.stringify(requestbody),

data: JSON.stringify({

"FirstName": fname,

"Email": email,

"LastName": lname,

"Bio": bio

}),

crossDomain : true,

success: function (res) {

console.log("it worked");

},

error: function (res, err) {

console.log("it did not work:"+ err+ "res:"+res);

},

});

});

// ajax call to put database data into the page

$.ajax({

type: "GET",

url: "[http://localhost:3000/users/"+userid](http://localhost:3000/users/%22+userid),

dataType: 'json',

crossDomain : true,

success: function (res) {

console.log("it worked");

console.log(res.FirstName);

document.getElementById('first\_name').value = res.FirstName;

document.getElementById('last\_name').value = res.LastName;

document.getElementById('email').value = res.Email;

document.getElementById('bio').value = res.Bio;

document.getElementById('username').value = res.Username;

},

error: function (res, err) {

console.log("it did not work");

},

});

//Get Groups that the user is in

$.ajax({

type: "GET",

url: "[http://localhost:3000/Users/Groups/"+userid](http://localhost:3000/Users/Groups/%22+userid),

dataType: 'json',

crossDomain : true,

success: function (res) {

console.log("it worked");

console.log(res.GroupName);

document.getElementById('group\_name').value = res.GroupName;

document.getElementById('group\_bio').value = res.Bio;

},

error: function (res, err) {

console.log("it did not work");

},

});

});

//function to get whats stored inside the cookie

function getCookie(cname) {

var name = cname + "=";

var decodedCookie = decodeURIComponent(document.cookie);

var ca = decodedCookie.split(';');

for(var i = 0; i <ca.length; i++) {

var c = ca[i];

while (c.charAt(0) == ' ') {

c = c.substring(1);

}

if (c.indexOf(name) == 0) {

return c.substring(name.length, c.length);

}

}

return "";

}

$(document).ready(function(){

$("#see\_cookie").click(function(){

console.log(getCookie("UserID"))

console.log("hell23o")

});

});

$(document).ready(function(){

$("#log\_out").click(function(){

document.cookie = "UserID=; expires=Thu, 01 Jan 1970 00:00:00 UTC; path=/;";

console.log("cookie has been deleted")

window.location.href = "<http://localhost:8000/login.html>"

console.log("log out button works")

});

});

//Delete User Ajax call

$(document).ready(function(){

var UserID = getCookie("UserID");;

$("#delete\_button").click(function(){

$.ajax({

type: "POST",

url: "[http://localhost:3000/DeleteUser/"+UserID](http://localhost:3000/DeleteUser/%22+UserID),

dataType: 'json',

crossDomain : true,

success: function (res) {

console.log("it worked");

//delete the cookie then go to the login page

document.cookie = "UserID=; expires=Thu, 01 Jan 1970 00:00:00 UTC; path=/;";

console.log("cookie has been deleted")

},

error: function (res, err) {

console.log("it did not work");

},

});

window.location.href = "<http://localhost:8000/login.html>"

});

});

// hiding and showing of edit, save and close buttons.

$(document).ready(function(){

$("#edit\_button").click(function(){

console.log("hide");

$("#edit\_button").hide();

$("#save\_close\_delete\_button").show();

$("#upload\_pic").show();

toggle\_readonly(false);

});

$("#close\_button").click(function(){

$("#edit\_button").show();

$("#save\_close\_delete\_button").hide();

$("#upload\_pic").hide();

toggle\_readonly(true);

});

});

Profile.css

html,

body {

height: 100%;

}

.box-shadow { box-shadow: 0 .25rem .75rem rgba(0, 0, 0, .05); }

#upload\_button {

display: block;

margin-left: auto;

margin-right: auto;

width: 50%;

}

**Calendar.html**

<!doctype html>

<html lang="en">

<head>

<meta charset="utf-8">

<meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">

<meta name="description" content="">

<meta name="author" content="">

<title>Calendar</title>

<link rel="canonical" href="https://getbootstrap.com/docs/4.0/examples/sign-in/">

<!-- Bootstrap core CSS -->

<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/css/bootstrap.min.css" integrity="sha384-Gn5384xqQ1aoWXA+058RXPxPg6fy4IWvTNh0E263XmFcJlSAwiGgFAW/dAiS6JXm" crossorigin="anonymous">

<script src="https://code.jquery.com/jquery-3.1.1.min.js"></script>

<script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.12.9/umd/popper.min.js" integrity="sha384-ApNbgh9B+Y1QKtv3Rn7W3mgPxhU9K/ScQsAP7hUibX39j7fakFPskvXusvfa0b4Q" crossorigin="anonymous"></script>

<script src="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/js/bootstrap.min.js" integrity="sha384-JZR6Spejh4U02d8jOt6vLEHfe/JQGiRRSQQxSfFWpi1MquVdAyjUar5+76PVCmYl" crossorigin="anonymous"></script>

<!-- Custom styles for this template -->

<link href="css/Calendar.css" rel="stylesheet">

<script type="text/javascript" src="js/calendar.js"></script>

</head>

<link rel="stylesheet" type="text/css" href="https://maxcdn.bootstrapcdn.com/font-awesome/4.7.0/css/font-awesome.min.css">

<link rel="stylesheet" type="text/css" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css">

<script src="https://ajax.googleapis.com/ajax/libs/jquery/2.1.1/jquery.min.js"></script>

<script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/js/bootstrap.min.js"></script>

<nav class="navbar navbar-dark bg-dark">

<a class="navbar-brand" href="#">Logo</a>

<button type="button" id="log\_out" class="btn btn-danger">Log out</button>

<button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#navbarsExample01" aria-controls="navbarsExample01" aria-expanded="false" aria-label="Toggle navigation">

<span class="navbar-toggler-icon"></span>

</button>

<div class="collapse navbar-collapse" id="navbarsExample01">

<ul class="navbar-nav mr-auto">

<li class="nav-item active">

<a class="nav-link" href="profile.html">MyProfile <span class="sr-only">(current)</span></a>

</li>

<li class="nav-item">

<a class="nav-link" href="Calendar.html">MyCalendar</a>

</li>

<li class="nav-item">

<a class="nav-link" href="Groups.html">Groups</a>

</li>

<li class="nav-item">

<a class="nav-link" href="Events.html">Events</a>

</div>

</li>

</ul>

</div>

</nav>

<div class="container">

<i class="prev-month fa fa-chevron-left fa-3x"></i> <i class="next-month fa fa-chevron-right fa-3x"></i>

<br>

<div class="month-year text-center">

<h3></h3></div>

<table class="table table-bordered">

<tr>

<th>S</th>

<th>M</th>

<th>T</th>

<th>W</th>

<th>T</th>

<th>F</th>

<th>S</th>

</tr>

</table>

</div>

**Calendar.js**

//log out function

$(document).ready(function(){

$("#log\_out").click(function(){

document.cookie = "UserID=; expires=Thu, 01 Jan 1970 00:00:00 UTC; path=/;";

window.location.href = "<http://localhost:8000/login.html>"

console.log("log out button works")

});

});

//Checks to make sure the user has a cookie and logs them out if they dont

$(document).ready(function() {

if (getCookie("UserID") < 1){

window.location.href = "<http://localhost:8000/login.html>"

}

});

var CURRENT\_DATE = new Date();

var d = new Date();

var months = 'January February March April May June July August September October November December'.split(' ');

var weekDayName = 'SUN MON TUES WED THURS FRI'.split(' ');

var daysInMonth = [31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31];

// Returns the day of week which month starts (eg 0 for Sunday, 1 for Monday, etc.)

function getCalendarStart(dayOfWeek, currentDate) {

var date = currentDate - 1;

var startOffset = (date % 7) - dayOfWeek;

if (startOffset > 0) {

startOffset -= 7;

}

return Math.abs(startOffset);

}

// Creating the Calendar

function CreateCalendar(startDay, totalDays, currentDate) {

var currentRow = 1;

var currentDay = startDay;

var $table = $('table');

var $week = getCalendarRow();

var $day;

var i = 1;

for (; i <= totalDays; i++) {

$day = $week.find('td').eq(currentDay);

$day.text(i);

if (i === currentDate) {

$day.addClass('today');

}

// +1 next day until Saturday (6), then reset to Sunday (0)

currentDay = ++currentDay % 7;

// Generate new row when the day is Saturday, but only if there are more days to render

if (currentDay === 0 && (i + 1 <= totalDays)) {

$week = getCalendarRow();

currentRow++;

}

}

}

// Clear generated calendar

function clearCalendar() {

var $trs = $('tr').not(':eq(0)');

$trs.remove();

$('.month-year').empty();

}

// Generates table row used when creating Calendar

function getCalendarRow() {

var $table = $('table');

var $tr = $('<tr/>');

for (var i = 0, len = 7; i < len; i++) {

$tr.append($('<td/>'));

}

$table.append($tr);

return $tr;

}

function myCalendar() {

var month = d.getUTCMonth();

var day = d.getUTCDay();

var year = d.getUTCFullYear();

var date = d.getUTCDate();

var totalDaysInMonth = daysInMonth[month];

var counter = 1;

var $h3 = $('<h3>');

$h3.text(months[month] + ' ' + year);

$h3.appendTo('.month-year');

var dateToHighlight = 0;

// Determine if Month && Year are current for Date Highlight

if (CURRENT\_DATE.getUTCMonth() === month && CURRENT\_DATE.getUTCFullYear() === year) {

dateToHighlight = date;

}

//Getting February days including the leasp year

if (month === 1) {

if ((year % 100 !== 0) && (year % 4 === 0) || (year % 400 === 0)) {

totalDaysInMonth = 29;

}

}

// Get Start Day

CreateCalendar(getCalendarStart(day, date), totalDaysInMonth, dateToHighlight);

};

// going back or forward months

function navigationHandler(dir) {

d.setUTCMonth(d.getUTCMonth() + dir);

clearCalendar();

myCalendar();

}

$(document).ready(function() {

// Bind months

$('.prev-month').click(function() {

navigationHandler(-1);

});

$('.next-month').click(function() {

navigationHandler(1);

});

// Generate Calendar

myCalendar();

});

**Calendar.css**

\* {

margin: 0;

padding: 0;

}

.container {

margin-top: 80px;

}

th {

height: 30px;

text-align: center;

font-weight: 700;

}

td {

height: 100px;

}

.today {

background-color: rgb(218, 29, 29);

}

th:nth-of-type(7),

td:nth-of-type(7) {

font-weight: bold;

}

th:nth-of-type(1),

td:nth-of-type(1) {

font-weight: bold;

}

**Groups.html**

<!doctype html>

<html lang="en">

<head>

<meta charset="utf-8">

<meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">

<meta name="description" content="">

<meta name="author" content="">

<title>Profile</title>

<link rel="canonical" href="https://getbootstrap.com/docs/4.0/examples/sign-in/">

<!-- Bootstrap core CSS -->

<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/css/bootstrap.min.css" integrity="sha384-Gn5384xqQ1aoWXA+058RXPxPg6fy4IWvTNh0E263XmFcJlSAwiGgFAW/dAiS6JXm" crossorigin="anonymous">

<script src="https://code.jquery.com/jquery-3.1.1.min.js"></script>

<script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.12.9/umd/popper.min.js" integrity="sha384-ApNbgh9B+Y1QKtv3Rn7W3mgPxhU9K/ScQsAP7hUibX39j7fakFPskvXusvfa0b4Q" crossorigin="anonymous"></script>

<script src="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/js/bootstrap.min.js" integrity="sha384-JZR6Spejh4U02d8jOt6vLEHfe/JQGiRRSQQxSfFWpi1MquVdAyjUar5+76PVCmYl" crossorigin="anonymous"></script>

<!-- Custom styles for this template -->

<link href="css/Groups.css" rel="stylesheet">

<script type="text/javascript" src="js/groups.js"></script>

</head>

<body>

<nav class="navbar navbar-dark bg-dark">

<a class="navbar-brand" href="#">Logo</a>

<button type="button" id="log\_out" class="btn btn-danger">Log out</button>

<button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#navbarsExample01" aria-controls="navbarsExample01" aria-expanded="false" aria-label="Toggle navigation">

<span class="navbar-toggler-icon"></span>

</button>

<div class="collapse navbar-collapse" id="navbarsExample01">

<ul class="navbar-nav mr-auto">

<li class="nav-item active">

<a class="nav-link" href="profile.html">MyProfile <span class="sr-only">(current)</span></a>

</li>

<li class="nav-item">

<a class="nav-link" href="Calendar.html">MyCalendar</a>

</li>

<li class="nav-item">

<a class="nav-link" href="Groups.html">Groups</a>

</li>

<li class="nav-item">

<a class="nav-link" href="Events.html">Events</a>

</div>

</li>

</ul>

</div>

</nav>

<main role="main" class="container">

<div class="my-3 p-3 bg-white rounded box-shadow">

<h6 class="border-bottom border-gray pb-2 mb-0">MyGroups</h6>

<div class="container" id="Groups">

<div class="row">

<div class="col-3">

<input class="form-control" type="text" id = "search\_box" placeholder="Search" aria-label="Search">

</div>

<div class="col-3">

<button type="button" id="search\_button" class="btn btn-dark">Search</button>

</div>

<div class="col-6 text-right">

<button type="button" id="create\_group\_button" class="btn btn-dark">Create Group</button>

</div>

</div>

<div class="col-3">

<button type="button" id="reset\_button" class="btn btn-dark">Reset</button>

</div>

</div>

<div class="my-3 p-3 bg-white rounded box-shadow">

<h6 class="border-bottom border-gray pb-2 mb-0">MyGroups</h6>

<table id="groups\_table" border='1'>

<tr>

<th>GroupID</th>

<th>Group Name</th>

<th>Group Bio</th>

</tr>

</table>

</div>

</div>

</main>

Groups.js

//if the cookie for the userID doesnt exist at any time the user is redirected to the login page

$(document).ready(function() {

if (getCookie("UserID") < 1){

window.location.href = "<http://localhost:8000/login.html>"

}

});

//Checks to make sure the user has a cookie and logs them out if they dont

$(document).ready(function(){

$("#log\_out").click(function(){

document.cookie = "UserID=; expires=Thu, 01 Jan 1970 00:00:00 UTC; path=/;";

window.location.href = "<http://localhost:8000/login.html>"

console.log("log out button works")

});

});

function setCookie(cname, cvalue, exdays) {

var d = new Date();

d.setTime(d.getTime() + (exdays\*24\*60\*60\*1000));

var expires = "expires="+ d.toUTCString();

document.cookie = cname + "=" + cvalue + ";" + expires + ";path=/";

}

//function to get whats stored inside the cookie

function getCookie(cname) {

var name = cname + "=";

var decodedCookie = decodeURIComponent(document.cookie);

var ca = decodedCookie.split(';');

for(var i = 0; i <ca.length; i++) {

var c = ca[i];

while (c.charAt(0) == ' ') {

c = c.substring(1);

}

if (c.indexOf(name) == 0) {

return c.substring(name.length, c.length);

}

}

return "";

}

//creates a new group when the create group button is pressed then redirects the user to the newest group created page

$(document).ready(function(){

$("#create\_group\_button").click(function(){

$.ajax({

type: "POST",

url: "<http://localhost:3000/NewGroup>",

data: {

},

crossDomain : true,

success: function (res) {

console.log("new group made");

},

error: function (res, err) {

console.log("it did not work");

},

});

$.ajax({

type: "GET",

url: "<http://localhost:3000/NewestGroup>",

data: {

},

crossDomain : true,

success: function (res) {

console.log("it worked");

console.log(res)

setCookie("GroupID", res.GroupID, 2)

console.log(getCookie("GroupID"))

window.location.href = "<http://localhost:8000/Group.html>"

},

error: function (res, err) {

console.log("it did not work");

},

});

});

});

//displays all the groups in a table form

$(document).ready(function(){

$.ajax({

type: "GET",

url: "<http://localhost:3000/AllGroups>",

data: {

},

crossDomain : true,

success: function (res) {

console.log("it worked");

console.log(res[0])

var trHTML = '';

$.each(res, function (i, Group) {

trHTML += '<tr><td>' + Group.GroupID + '</td><td>' + Group.GroupName + '</td><td>' + Group.Bio + '</td></tr>';

});

$('#groups\_table').append(trHTML);

},

error: function (res, err) {

console.log("it did not work");

},

});

});

$(document).ready(function(){

$("#reset\_button").click(function(){

location.reload();

});

});

// displays all the groups in a table form that Group Name begin with whatever the user searched

$(document).ready(function(){

$("#search\_button").click(function(){

var search = document.getElementById('search\_box').value

console.log(search)

$.ajax({

type: "GET",

url: "<http://localhost:3000/SearchGroups/>" +search,

data: {

//"Search" : search,

},

crossDomain : true,

success: function (res) {

console.log("it worked");

console.log(res)

var trHTML = '';

$('td').remove();

$.each(res, function (i, Group) {

trHTML += '<tr><td>' + Group.GroupID + '</td><td>' + Group.GroupName + '</td><td>' + Group.Bio + '</td></tr>';

// if there is only one result then go to the groups page

if (res.length == 1){

setCookie("GroupID", Group.GroupID, 2)

window.location.href = "<http://localhost:8000/Group.html>"

}

});

$('#groups\_table').append(trHTML);

},

error: function (res, err) {

console.log("it did not work");

},

});

});

});

Groups.css

html,

body {

height: 100%;

}

.box-shadow { box-shadow: 0 .25rem .75rem rgba(0, 0, 0, .05); }

Group.html

<!doctype html>

<html lang="en">

<head>

<meta charset="utf-8">

<meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">

<meta name="description" content="">

<meta name="author" content="">

<title>Profile</title>

<link rel="canonical" href="https://getbootstrap.com/docs/4.0/examples/sign-in/">

<!-- Bootstrap core CSS -->

<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/css/bootstrap.min.css" integrity="sha384-Gn5384xqQ1aoWXA+058RXPxPg6fy4IWvTNh0E263XmFcJlSAwiGgFAW/dAiS6JXm" crossorigin="anonymous">

<script src="https://code.jquery.com/jquery-3.1.1.min.js"></script>

<script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.12.9/umd/popper.min.js" integrity="sha384-ApNbgh9B+Y1QKtv3Rn7W3mgPxhU9K/ScQsAP7hUibX39j7fakFPskvXusvfa0b4Q" crossorigin="anonymous"></script>

<script src="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/js/bootstrap.min.js" integrity="sha384-JZR6Spejh4U02d8jOt6vLEHfe/JQGiRRSQQxSfFWpi1MquVdAyjUar5+76PVCmYl" crossorigin="anonymous"></script>

<!-- Custom styles for this template -->

<link href="css/Group.css" rel="stylesheet">

<script type="text/javascript" src="js/group.js"></script>

</head>

<body>

<nav class="navbar navbar-dark bg-dark">

<a class="navbar-brand" href="#">Logo</a>

<button type="button" id="log\_out" class="btn btn-danger">Log out</button>

<button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#navbarsExample01" aria-controls="navbarsExample01" aria-expanded="false" aria-label="Toggle navigation">

<span class="navbar-toggler-icon"></span>

</button>

<div class="collapse navbar-collapse" id="navbarsExample01">

<ul class="navbar-nav mr-auto">

<li class="nav-item active">

<a class="nav-link" href="profile.html">MyProfile <span class="sr-only">(current)</span></a>

</li>

<li class="nav-item">

<a class="nav-link" href="Calendar.html">MyCalendar</a>

</li>

<li class="nav-item">

<a class="nav-link" href="Groups.html">Groups</a>

</li>

<li class="nav-item">

<a class="nav-link" href="Events.html">Events</a>

</div>

</li>

</ul>

</div>

</nav>

<main role="main" class="container"></main>

<div class="my-3 p-3 bg-white rounded box-shadow">

<h6 class="border-bottom border-gray pb-2 mb-0">MyGroup</h6>

<div class="container" id="profile\_details">

<div class="row">

<div class="col-4 text-center align-items-center" id="profile\_image">

<div class="pt-4">

<div class="">

<img class="rounded-circle" src="./img/download.png" alt="Generic placeholder image" width="140" height="140">

</div>

<div>

<button type="button" id="upload\_pic" class="btn btn-dark" >Upload</button>

</div>

</div>

</div>

<div class="col-8" id="profile\_name">

<form>

<div class="form-group">

<label for="group\_name">Group Name</label>

<input type="email" class="form-control" id="group\_name" aria-describedby="emailHelp" placeholder="Enter Group Name" disabled>

</div>

<div class="form-group">

<label for="exampleFormControlTextarea1">Bio</label>

<textarea class="form-control" id="bio" rows="3" disabled></textarea>

</div>

</div>

</div>

<div class="row">

<div class="col">

<div id="edit\_page" class="text-right">

<button type="button" id="edit\_button" class="btn btn-dark">Edit</button>

</div>

<div id="join\_leave" class="text-left">

<button type="button" id="join\_button" class="btn btn-dark">Join</button>

<button type="button" id="leave\_button" class="btn btn-dark">Leave</button>

</div>

<div id="save\_close\_delete\_button" class="text-right">

<button type="button" id="save\_button" class="btn btn-dark">Save</button>

<button type="button" id="close\_button" class="btn btn-dark">Close</button>

<button type="button" id="delete\_button" class="btn btn-danger">Delete</button>

</div>

</div>

</div>

<div class="my-3 p-3 bg-white rounded box-shadow">

<h6 class="border-bottom border-gray pb-2 mb-0">Users In Group</h6>

<div class="list-group">

<a href="#" class="list-group-item list-group-item-action flex-column align-items-start">

<div class="row">

<div class="col col-lg-2" id="profile\_image">

<div class="pt-4">

<div class="">

<img class="rounded-circle" src="./img/download.png" alt="Generic placeholder image" width="140" height="140">

</div>

</div>

</div>

<div class="col" id="group\_bio">

<div class="form-group">

<input type="group\_name" class="form-control" id="users\_name" aria-describedby="emailHelp" placeholder="" disabled>

</div>

<div class="form-group">

<textarea class="form-control" id="users\_bio" rows="3" disabled></textarea>

</div>

</div>

</div>

</a>

<a href="#" class="list-group-item list-group-item-action flex-column align-items-start">

<div class="row">

<div class="col col-lg-2" id="profile\_image">

<div class="pt-4">

<div class="">

<img class="rounded-circle" src="./img/download.png" alt="Generic placeholder image" width="140" height="140">

</div>

</div>

</div>

<div class="col" id="group\_bio">

<form>

<h1 class="text-center">User Name</h1>

</div>

</div>

</a>

</div>

</div>

</div>

</div>

</main>

**Group.js**

//hides the save+close and the upload pic button

$(document).ready(function() {

$('#save\_close\_delete\_button').hide();

$('#upload\_pic').hide();

});

//Checks to make sure the user has a cookie and logs them out if they dont

$(document).ready(function() {

if (getCookie("UserID") < 1){

window.location.href = "<http://localhost:8000/login.html>"

}

});

//function to get whats stored inside the cookie

function getCookie(cname) {

var name = cname + "=";

var decodedCookie = decodeURIComponent(document.cookie);

var ca = decodedCookie.split(';');

for(var i = 0; i <ca.length; i++) {

var c = ca[i];

while (c.charAt(0) == ' ') {

c = c.substring(1);

}

if (c.indexOf(name) == 0) {

return c.substring(name.length, c.length);

}

}

return "";

}

function toggle\_readonly(state){

document.getElementById("group\_name").disabled = state;

document.getElementById("bio").disabled = state;

}

//ajax call to put the groups data into the page

var groupid = getCookie("GroupID");

$.ajax({

type: "GET",

url: "[http://localhost:3000/Groups/"+groupid](http://localhost:3000/Groups/%22+groupid),

dataType: 'json',

crossDomain : true,

success: function (res) {

console.log("it worked");

console.log(res.EventName);

document.getElementById('group\_name').value = res.GroupName;

document.getElementById('bio').value = res.Bio;

},

error: function (res, err) {

console.log("it did not work");

},

});

//updates the database with the new inputted data

$(document).ready(function(){

var GroupID = getCookie("GroupID");;

console.log(GroupID)

$("#save\_button").click(function(){

var groupname = document.getElementById('group\_name').value;

var bio = document.getElementById("bio").value;

console.log("this will save information");

console.log(groupname);

var requestbody ={

"Groupname": document.getElementById('group\_name').value,

"Bio" : document.getElementById("bio").value,

}

$.ajax({

type: "PUT",

url: "[http://localhost:3000/Groups/"+GroupID](http://localhost:3000/Groups/%22+GroupID),

contentType: 'application/json',

data: JSON.stringify({

"GroupName": groupname,

"Bio": bio

}),

crossDomain : true,

success: function (res) {

console.log("it worked");

},

error: function (res, err) {

console.log("it did not work:"+ err+ "res:"+res);

},

});

});

//displays the users username in the group and their bio

$.ajax({

type: "GET",

url: "[http://localhost:3000/Groups/Users/"+GroupID](http://localhost:3000/Groups/Users/%22+GroupID),

dataType: 'json',

crossDomain : true,

success: function (res) {

console.log("it worked");

console.log(res.GroupName);

document.getElementById('users\_name').value = res.Username;

document.getElementById('users\_bio').value = res.Bio;

},

error: function (res, err) {

console.log("it did not work");

},

});

});

//deletes the group then redirects you to the groups page

$(document).ready(function(){

var GroupID = getCookie("GroupID");;

$("#delete\_button").click(function(){

$.ajax({

type: "POST",

url: "[http://localhost:3000/DeleteGroup/"+GroupID](http://localhost:3000/DeleteGroup/%22+GroupID),

dataType: 'json',

crossDomain : true,

success: function (res) {

console.log("it worked");

//delete the cookie

document.cookie = "GroupID=; expires=Thu, 01 Jan 1970 00:00:00 UTC; path=/;";

console.log("cookie has been deleted")

},

error: function (res, err) {

console.log("it did not work");

},

});

console.log("cookie has been deleted")

window.location.href = "<http://localhost:8000/Groups.html>"

});

});

$(document).ready(function(){

$("#join\_button").click(function(){

var GroupID = getCookie("GroupID");;

var UserID = getCookie("UserID");

$.ajax({

type: "POST",

url: "[http://localhost:3000/JoinGroup/"+UserID+'/'+GroupID](http://localhost:3000/JoinGroup/%22+UserID+'/'+GroupID),

dataType: 'json',

crossDomain : true,

success: function (res) {

},

error: function (res, err) {

console.log("it did not work");

},

});

});

});

// when the log out button is clicked the cookie is set to be expired therefore deleting the cookie

$(document).ready(function(){

$("#log\_out").click(function(){

document.cookie = "UserID=; expires=Thu, 01 Jan 1970 00:00:00 UTC; path=/;";

window.location.href = "<http://localhost:8000/login.html>"

console.log("log out button works")

});

});

//when the edit button is clicked the save and close buttons appear while the edit button disapears vice verser when clicking the close button

$(document).ready(function(){

$("#edit\_button").click(function(){

console.log("hide");

$("#edit\_button").hide();

$("#save\_close\_delete\_button").show();

$("#upload\_pic").show();

toggle\_readonly(false);

});

$("#close\_button").click(function(){

$("#edit\_button").show();

$("#save\_close\_delete\_button").hide();

$("#upload\_pic").hide();

toggle\_readonly(true);

});

});

**Group.css**

html,

body {

height: 100%;

}

.box-shadow { box-shadow: 0 .25rem .75rem rgba(0, 0, 0, .05); }

Events.html

<!doctype html>

<html lang="en">

<head>

<meta charset="utf-8">

<meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">

<meta name="description" content="">

<meta name="author" content="">

<title>Profile</title>

<link rel="canonical" href="https://getbootstrap.com/docs/4.0/examples/sign-in/">

<!-- Bootstrap core CSS -->

<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/css/bootstrap.min.css" integrity="sha384-Gn5384xqQ1aoWXA+058RXPxPg6fy4IWvTNh0E263XmFcJlSAwiGgFAW/dAiS6JXm" crossorigin="anonymous">

<script src="https://code.jquery.com/jquery-3.1.1.min.js"></script>

<script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.12.9/umd/popper.min.js" integrity="sha384-ApNbgh9B+Y1QKtv3Rn7W3mgPxhU9K/ScQsAP7hUibX39j7fakFPskvXusvfa0b4Q" crossorigin="anonymous"></script>

<script src="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/js/bootstrap.min.js" integrity="sha384-JZR6Spejh4U02d8jOt6vLEHfe/JQGiRRSQQxSfFWpi1MquVdAyjUar5+76PVCmYl" crossorigin="anonymous"></script>

<!-- Custom styles for this template -->

<link href="css/Events.css" rel="stylesheet">

<script type="text/javascript" src="js/events.js"></script>

</head>

<body>

<nav class="navbar navbar-dark bg-dark">

<a class="navbar-brand" href="#">Logo</a>

<button type="button" id="log\_out" class="btn btn-danger">Log out</button>

<button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#navbarsExample01" aria-controls="navbarsExample01" aria-expanded="false" aria-label="Toggle navigation">

<span class="navbar-toggler-icon"></span>

</button>

<div class="collapse navbar-collapse" id="navbarsExample01">

<ul class="navbar-nav mr-auto">

<li class="nav-item active">

<a class="nav-link" href="profile.html">MyProfile <span class="sr-only">(current)</span></a>

</li>

<li class="nav-item">

<a class="nav-link" href="Calendar.html">MyCalendar</a>

</li>

<li class="nav-item">

<a class="nav-link" href="Groups.html">Groups</a>

</li>

<li class="nav-item">

<a class="nav-link" href="Events.html">Events</a>

</div>

</li>

</ul>

</div>

</nav>

<main role="main" class="container">

<div class="my-3 p-3 bg-white rounded box-shadow">

<h6 class="border-bottom border-gray pb-2 mb-0">MyEvents</h6>

<div class="container" id="Events">

<div class="row box-shadow">

<div class="col">

<h2 class="">Events</h2>

</div>

<div class="col-3">

<input class="form-control" type="text" id = "search\_box" placeholder="Search" aria-label="Search">

</div>

<div class="col-3">

<button type="button" id="search\_button" class="btn btn-dark">Search</button>

<button type="button" id="reset\_button" class="btn btn-dark">Reset</button>

</div>

<div class="col text-right">

<button type="button" id="add\_button" class="btn btn-dark">Add Event</button>

</div>

</div>

<table id="events\_table" border='1'>

<tr>

<th>EventID</th>

<th>Event Name</th>

<th>Date Of Event</th>

<th>Event Bio</th>

</tr>

</table>

</div>

</div>

Events.js

//function to get whats stored inside the cookie

function getCookie(cname) {

var name = cname + "=";

var decodedCookie = decodeURIComponent(document.cookie);

var ca = decodedCookie.split(';');

for(var i = 0; i <ca.length; i++) {

var c = ca[i];

while (c.charAt(0) == ' ') {

c = c.substring(1);

}

if (c.indexOf(name) == 0) {

return c.substring(name.length, c.length);

}

}

return "";

}

function setCookie(cname, cvalue, exdays) {

var d = new Date();

d.setTime(d.getTime() + (exdays\*24\*60\*60\*1000));

var expires = "expires="+ d.toUTCString();

document.cookie = cname + "=" + cvalue + ";" + expires + ";path=/";

}

// gets all the events and displays them in a table

$(document).ready(function(){

$.ajax({

type: "GET",

url: "<http://localhost:3000/AllEvents>",

data: {

},

crossDomain : true,

success: function (res) {

console.log("it worked");

console.log(res[0])

var trHTML = '';

$.each(res, function (i, event) {

trHTML += '<tr><td>' + event.EventID + '</td><td>' + event.EventName + '</td><td>' + event.DateOfEvent + '</td><td>' + event.Bio + '</td></tr>';

});

$('#events\_table').append(trHTML);

},

error: function (res, err) {

console.log("it did not work");

},

});

});

//Checks to make sure the user has a cookie and logs them out if they dont

$(document).ready(function() {

if (getCookie("UserID") < 1){

window.location.href = "<http://localhost:8000/login.html>"

}

});

//when log out is clicked the user is sent to the login screen

$(document).ready(function(){

$("#log\_out").click(function(){

document.cookie = "UserID=; expires=Thu, 01 Jan 1970 00:00:00 UTC; path=/;";

window.location.href = "<http://localhost:8000/login.html>"

console.log("log out button works")

});

});

//when the reset button is pressed the page is reloaded

$(document).ready(function(){

$("#reset\_button").click(function(){

location.reload();

});

});

//gets all the groups according to the search then puts them in a table form

$(document).ready(function(){

$("#search\_button").click(function(){

var search = document.getElementById('search\_box').value

console.log(search)

$.ajax({

type: "GET",

url: "<http://localhost:3000/SearchEvents/>" +search,

data: {

},

crossDomain : true,

success: function (res) {

console.log("it worked");

console.log(res)

var trHTML = '';

$('td').remove();

$.each(res, function (i, event) {

trHTML += '<tr><td>' + event.EventID + '</td><td>' + event.EventName + '</td><td>' + event.DateOfEvent + '</td><td>' + event.Bio + '</td></tr>';

//if there is only one result then go to that events page

if (res.length == 1){

setCookie("EventID", event.EventID, 2)

window.location.href = "<http://localhost:8000/Event.html>"

}

});

$('#events\_table').append(trHTML);

},

error: function (res, err) {

console.log("it did not work");

},

});

});

});

//pressing the add event button creates a new event then goes to the event page of the newest event made

$(document).ready(function(){

$("#add\_button").click(function(){

$.ajax({

type: "POST",

url: "<http://localhost:3000/NewEvent>",

data: {

},

crossDomain : true,

success: function (res) {

console.log("new event made");

},

error: function (res, err) {

console.log("it did not work");

},

});

$.ajax({

type: "GET",

url: "<http://localhost:3000/NewestEvent>",

data: {

},

crossDomain : true,

success: function (res) {

console.log("it worked");

console.log(res)

setCookie("EventID", res.EventID, 2)

console.log(getCookie("EventID"))

window.location.href = "<http://localhost:8000/Event.html>"

},

error: function (res, err) {

console.log("it did not work");

},

});

});

});

Events.css

html,

body {

height: 100%;

}

.box-shadow { box-shadow: 0 .25rem .75rem rgba(0, 0, 0, .05); }

Event.html

<!doctype html>

<html lang="en">

<head>

<meta charset="utf-8">

<meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">

<meta name="description" content="">

<meta name="author" content="">

<title>Profile</title>

<link rel="canonical" href="https://getbootstrap.com/docs/4.0/examples/sign-in/">

<!-- Bootstrap core CSS -->

<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/css/bootstrap.min.css" integrity="sha384-Gn5384xqQ1aoWXA+058RXPxPg6fy4IWvTNh0E263XmFcJlSAwiGgFAW/dAiS6JXm" crossorigin="anonymous">

<script src="https://code.jquery.com/jquery-3.1.1.min.js"></script>

<script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.12.9/umd/popper.min.js" integrity="sha384-ApNbgh9B+Y1QKtv3Rn7W3mgPxhU9K/ScQsAP7hUibX39j7fakFPskvXusvfa0b4Q" crossorigin="anonymous"></script>

<script src="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/js/bootstrap.min.js" integrity="sha384-JZR6Spejh4U02d8jOt6vLEHfe/JQGiRRSQQxSfFWpi1MquVdAyjUar5+76PVCmYl" crossorigin="anonymous"></script>

<!-- Custom styles for this template -->

<link href="css/Event.css" rel="stylesheet">

<script type="text/javascript" src="js/event.js"></script>

</head>

<body>

<nav class="navbar navbar-dark bg-dark">

<a class="navbar-brand" href="#">Logo</a>

<button type="button" id="log\_out" class="btn btn-danger">Log out</button>

<button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#navbarsExample01" aria-controls="navbarsExample01" aria-expanded="false" aria-label="Toggle navigation">

<span class="navbar-toggler-icon"></span>

</button>

<div class="collapse navbar-collapse" id="navbarsExample01">

<ul class="navbar-nav mr-auto">

<li class="nav-item active">

<a class="nav-link" href="profile.html">MyProfile <span class="sr-only">(current)</span></a>

</li>

<li class="nav-item">

<a class="nav-link" href="Calendar.html">MyCalendar</a>

</li>

<li class="nav-item">

<a class="nav-link" href="Groups.html">Groups</a>

</li>

<li class="nav-item">

<a class="nav-link" href="Events.html">Events</a>

</div>

</li>

</ul>

</div>

</nav>

<main role="main" class="container">

<div class="my-3 p-3 bg-white rounded box-shadow">

<h6 class="border-bottom border-gray pb-2 mb-0">Event</h6>

<div class="container" id="Events">

<div class="row">

<div class="col-4">

<label for="event\_name">Event Name</label>

<input type="email" class="form-control" id="event\_name" placeholder="Enter Event Name" disabled>

</div>

<div class="col-4">

<label for="date\_of\_event">Date Of Event</label>

<input type="email" class="form-control" id="date\_of\_event" placeholder="Enter Date of Event" disabled>

</div>

<div class="col-4">

<label for="bio">Bio</label>

<textarea class="form-control" id="bio" rows="3" disabled></textarea>

</div>

</div>

<div class="row">

<div class="row">

<div class="col">

<div id="edit\_page" class="text-right">

<button type="button" id="edit\_button" class="btn btn-dark">Edit</button>

</div>

<div id="save\_close\_button" class="text-right">

<button type="button" id="save\_button" class="btn btn-dark">Save</button>

<button type="button" id="close\_button" class="btn btn-dark">Close</button>

<button type="button" id="delete\_button" class="btn btn-danger">Delete</button>

</div>

</div>

</div>

</div>

</div>

</main

Event.js

//function to get whats stored inside the cookie

function getCookie(cname) {

var name = cname + "=";

var decodedCookie = decodeURIComponent(document.cookie);

var ca = decodedCookie.split(';');

for(var i = 0; i <ca.length; i++) {

var c = ca[i];

while (c.charAt(0) == ' ') {

c = c.substring(1);

}

if (c.indexOf(name) == 0) {

return c.substring(name.length, c.length);

}

}

return "";

}

//creating a function to be used to disable and enable the text boxes without reusing code

function toggle\_readonly(state){

document.getElementById("event\_name").disabled = state;

document.getElementById("date\_of\_event").disabled = state;

document.getElementById("bio").disabled = state;

}

//Checks to make sure the user has a cookie and logs them out if they dont

$(document).ready(function() {

if (getCookie("UserID") < 1){

window.location.href = "<http://localhost:8000/login.html>"

}

});

//this puts the database information into the page

$(document).ready(function() {

$('#save\_close\_button').hide();

});

var eventid = getCookie("EventID");

console.log(eventid)

$.ajax({

type: "GET",

url: "[http://localhost:3000/Events/"+eventid](http://localhost:3000/Events/%22+eventid),

dataType: 'json',

crossDomain : true,

success: function (res) {

console.log("it worked");

console.log(res.EventName);

document.getElementById('event\_name').value = res.EventName;

//fix date of event sql table datafield

document.getElementById('bio').value = res.Bio;

},

error: function (res, err) {

console.log("it did not work");

},

});

// updates the database with the new inputted data

$(document).ready(function(){

var EventID = getCookie("EventID");

$("#save\_button").click(function(){

var eventname = document.getElementById('event\_name').value;

var bio = document.getElementById("bio").value;

var dateofevent = document.getElementById("date\_of\_event").value;

console.log("this will save information");

console.log(eventname);

var requestbody ={

"Groupname": document.getElementById('event\_name').value,

"Bio" : document.getElementById("bio").value,

//"DateOfEvent" : document.getElementById("date\_of\_event").value,

}

$.ajax({

type: "PUT",

url: "[http://localhost:3000/Events/"+EventID](http://localhost:3000/Events/%22+EventID),

contentType: 'application/json',

// data: JSON.stringify(requestbody),

data: JSON.stringify({

"EventName": eventname,

"Bio": bio,

//"DateOfEvent" : dateofevent,

}),

crossDomain : true,

success: function (res) {

console.log("it worked");

},

error: function (res, err) {

console.log("it did not work:"+ err+ "res:"+res);

},

});

});

});

// when the log out button is clicked the cookie is set to be expired therefore deleting the cookie

$(document).ready(function(){

$("#log\_out").click(function(){

document.cookie = "UserID=; expires=Thu, 01 Jan 1970 00:00:00 UTC; path=/;";

window.location.href = "<http://localhost:8000/login.html>"

console.log("log out button works")

});

});

//when the edit button is clicked the save and close buttons appear while the edit button disapears vice verser when clicking the close button

$(document).ready(function(){

$("#edit\_button").click(function(){

console.log("hide");

$("#edit\_button").hide();

$("#save\_close\_button").show();

toggle\_readonly(false);

});

$("#close\_button").click(function(){

$("#edit\_button").show();

$("#save\_close\_button").hide();

toggle\_readonly(true);

});

});

//Delete Event Ajax call

$(document).ready(function(){

var EventID = getCookie("EventID");;

$("#delete\_button").click(function(){

$.ajax({

type: "POST",

url: "[http://localhost:3000/DeleteEvent/"+EventID](http://localhost:3000/DeleteEvent/%22+EventID),

dataType: 'json',

crossDomain : true,

success: function (res) {

console.log("it worked");

//delete the cookie then go to the events page

document.cookie = "EventID=; expires=Thu, 01 Jan 1970 00:00:00 UTC; path=/;";

},

error: function (res, err) {

console.log("it did not work");

},

});

console.log("cookie was delted")

window.location.href = "<http://localhost:8000/Events.html>"

});

});

Event.css

html,

body {

height: 100%;

}

.box-shadow { box-shadow: 0 .25rem .75rem rgba(0, 0, 0, .05); }

**Testing–4**

**Postman 4.1**

Graphical user interface, text, email, website

Description automatically generated

To do a lot of the API testing I used an application called Postman. This tool lets you send and receive a request from a service. It was a very useful tool as it let me check to make sure my API worked before using it in my code. Another reason why it was useful was because if there was an error in my code, I wouldn’t know whether it was the API or the Ajax call that was incorrect. But by checking postman it let me know that the API was either sending the right information or not narrowing down where the error in my code was.

The UI was very easy to use and it let me change from GET, POST and PUT letting me test any API I could've possibly used. It also let me input a body to test when passing through a username for example to ensure that it would work with a user’s input.

**Test Strategy 4.2**

In order to make sure the system works I will be testing it. This will include testing that the game functions properly (black box testing) and testing the user interface (eg. Buttons, input boxes) with white box testing.

To test the input fields, I will use valid and erroneous data to check that the system can handle them properly and not crash.

I will be recording on QuickTime Player.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| # | Description of test | How to test/ test data used | Expected outcome | Actual outcome | Time stamp |
| 1 | Test if registering an account works | Testname – username  123 – password  Then press register | Notification of successful register. Check sql users table to make sure | Success: user has been successful registered | 0:09 |
| 2 | Test login if wrong username but correct password creates an error | Testname1 – username  123 - password | Notification of incorrect password or username. Shouldn't be redirected to profile page | Success: Notification of incorrect password or username. | 0:19 |
| 3 | Test login if correct username but incorrect password creates an error | Testname – username  1234 - password | Notification of incorrect password or username. Shouldn't be redirected to profile page | Success: Notification of incorrect password or username. | 0:26 |
| 4 | Test login with correct password and username | Testname – username  123 - password | User should be redirected to their profile page | Success: User has been redirected to the profile page | 0:35 |
|  | **Profile Page** |  |  |  |  |
| 5 | Test to see if the profile is the accounts profile | The box under the “MyProfile” will display the username of the account | Box should display the username that the user logged in with. | Success: box displays username that the user logged in with | 0:36 ( under “MyProfile”) |
| 6 | Test to see if the cookie has been created for the UserID | I will check the console log for the UserID in the cookie showing the code as well. | Console should log the correct UserID from the cookie | Success: correct UserID has been logged | 0:52 |
| 7 | Test to make sure user cannot edit text without clicking edit button first | Click on the box and try to type. | Won't be able to type and change the information in the box. | Success: user cannot change the data in the boxes | 0:46 |
| 8 | Test to see if edit button makes text boxes editable | Click on the edit button | All the boxes will no longer be greyed out and can be edited. | Success: when the button is clicked the boxes are editable | 1:03 |
| 9 | Test to see if when changed data is saved it updates the database and page | Change the data then press the save button then reload the page to prove it has updated the database | When the page is reloaded the data in the boxes should be the same as what they’ve been changed to. | Success: when the page is reloaded the new data remains in the boxes | 1:33 |
| 10 | Test to see if the close button works | Click on the close button | Should turn the boxes to grey again and they can no longer be edited | Success: the boxes cannot be edited anymore and are grey | 1:45 |
| 11 | Test to see if the log out button returns user to the login/register page | Click log out button | User should be redirected to the login/register page | Success: the user is now on the login page | 1:50 |
| 12 | Test to see if the log out button deletes the cookie | Check console log | Console log should say if the cookie has been deleted | Success: console log shows that the cookie has been deleted | 1:49 pause |
| 13 | Test to see if the delete button returns user to login/register page | Click the delete button | User should be redirected to the login/register page | Success: the user is now on the login page | 2:10 |
| 14 | Test to see if the delete button deletes the account | Check sql table for the account | The account should no longer be on the Users table | Success: the account has been deleted and is no longer on the Users table |  |
| 15 | Test to see if the delete button deletes the cookie | Check console log | Console log should say if the cookie has been deleted | Success: console log shows that the cookie has been deleted | 2:10 |
| 16 | Test to see if the button with the 3 lines brings the pull-down menu up | Click the button in the top right corner | Pull-down menu should appear | Success: Pull-down menu appears | 2:29 |
| 17 | Test to see if going to the calendar works | Click MyCalendar | User should be redirected to the calendar page | Success: user was redirected to the calendar page | 2:33 |
|  | **Calendar Page** |  |  |  |  |
| 18 | Test to see if the calendar shows the current day | Open new tab to search for current day | The red box should be over the current day | Success: the red box highlights the current day | 2:46 |
| 19 | Test to see if the arrows go back and forward a month | Click on both arrows | The month should go back and forth | Success: the month goes back and forth | 2:51 |
| 20 | Test to see if the year can change | Click on the arrows until the year changes | The year should increase or decrease depending on which way you went | Success: the year changes | 2:58 |
| 21 | Test if the groups button works | Click groups button | Groups page should load up | Success: groups page is loaded up | 3:03 |
|  | **Groups Page** |  |  |  |  |
| 22 | Write a groups name in the search box that won’t bring anything up | Type aaa then press search | The table should display no groups | Success: the table displays no groups | 3:11 |
| 23 | Test if the reset button works | Press the reset button | The page should reload and all the groups are displayed again | Success: page is reloaded and all the groups are displayed again | 3:16 |
| 24 | Test if the search function works | Search “Tes” then press search | The table should display all groups who's group name begins with tes | Success: all groups beginning with tes are displayed | 3:29 |
| 25 | Test if when searching for the whole name of a group it goes to their page | Search “tesco2” | User should be redirected to the tesco2 group page | Success: user was redirected to the tesco2 group page | 3:35 |
| 26 | Test if a cookie has been created for the GroupID of whatever the user searched | Check the console log then show the code | Cookie should be created for the groupID of the group the user searched for | Success: cookie was created with the same ID as the group the user searched for | 3:45 |
| 27 | Test if create group button redirects to group page | Press button | User should be redirected to group page of this new group | Success: user is redirected to the new page | 3:56 |
| 28 | Test if create group button creates a new group | Check group table before and after clicking the button | A new group should be created | Success: new group has been created |  |
|  | **Group Page** |  |  |  |  |
| 29 | Test to make sure user cannot edit text without clicking edit button first | Click on the box and try to type. | Won't be able to type and change the information in the box. | Success: user cannot change the data in the boxes | 4:05 |
| 30 | Test to see if edit button makes text boxes editable | Click on the edit button | All the boxes will no longer be greyed out and can be edited. | Success: when the button is clicked the boxes are editable | 4:09 |
| 31 | Test to see if when changed data is saved it updates the database and page | Change the data then press the save button then reload the page to prove it has updated the database | When the page is reloaded the data in the boxes should be the same as what they’ve been changed to. | Success: when the page is reloaded the new data remains in the boxes | 4:50 |
| 32 | Test to see if the close button works | Click on the close button | Should turn the boxes to grey again and they can no longer be edited | Success: the boxes cannot be edited anymore and are grey | 5:03 |
| 33 | Test to see if the delete button returns user to groups page | Click the delete button | User should be redirected to the groups page | Success: the user is now on the groups page | 5:08 |
| 34 | Test to see if the delete button deletes the group | Check sql table for the group | The group should no longer be on the groups table | Success: the group has been deleted and is no longer on the groups table |  |
| 35 | Test to see if the delete button deletes the cookie | Check console log | Console log should say if the cookie has been deleted | Success: console log shows that the cookie has been deleted | 5:08 pause |
| 36 | Test to see if the events button works | Click events | User should be redirected to events | Success: user is redirected to events | 5:17 |
|  | **Events Page** |  |  |  |  |
| 37 | Write a events name in the search box that won’t bring anything up | Type aaa then press search | The table should display no events | Success: the table displays no events | 5:23 |
| 38 | Test if the reset button works | Press the reset button | The page should reload and all the events are displayed again | Success: page is reloaded and all the events are displayed again | 5:26 |
| 39 | Test if the search function works | Search “Tes” then press search | The table should display all groups whose event name begins with tes | Success: all events beginning with tes are displayed | 5:30 |
| 40 | Test if when searching for the whole name of a group it goes to their page | Search “meeting” | User should be redirected to the meeting event page | Success: user was redirected to the meeting event page | 5:40 |
| 41 | Test if a cookie has been created for the EventID of whatever the user searched | Check the console log then show the code | Cookie should be created for the EventID of the group the user searched for | Success: cookie was created with the same ID as the event the user searched for | 6:47 |
| 42 | Test if add event button redirects to group page | Press button | User should be redirected to event page of this new group | Success: user is redirected to the new page | 6:25 |
| 43 | Test if add event button creates a new event | Check event table before and after clicking the button | A new event should be created | Success: new event has been created | 6:35 |
|  | **Event Page** |  |  |  |  |
| 44 | Test to make sure user cannot edit text without clicking edit button first | Click on the box and try to type. | Won't be able to type and change the information in the box. | Success: user cannot change the data in the boxes |  |
| 45 | Test to see if edit button makes text boxes editable | Click on the edit button | All the boxes will no longer be greyed out and can be edited. | Success: when the button is clicked the boxes are editable | 5:45 |
| 46 | Test to see if when changed data is saved it updates the database and page | Change the data then press the save button then reload the page to prove it has updated the database | When the page is reloaded the data in the boxes should be the same as what they’ve been changed to. | Success: when the page is reloaded the new data remains in the boxes | 6:06 |
| 47 | Test to see if the close button works | Click on the close button | Should turn the boxes to grey again and they can no longer be edited | Success: the boxes cannot be edited anymore and are grey | 6:01 |
| 48 | Test to see if the delete button returns user to events page | Click the delete button | User should be redirected to the events page | Success: the user is now on the events page | 6:13 |
| 49 | Test to see if the delete button deletes the event | Check sql table for the event | The account should no longer be on the groups table | Success: the group has been deleted and is no longer on the groups table | 6:23 |
| 50 | Test to see if the delete button deletes the cookie | Check console log | Console log should say if the cookie has been deleted | Success: console log shows that the cookie has been deleted | 6:!2 pause |

Link for the video:

https://youtu.be/QYSyO7jYakQ

**Evaluation–5**

**Objective Check 5.1**

At the beginning of my project, I wrote some objectives that I wanted to complete in order to have a working project. I felt like by completing these my project would be of a high level and would create a web application I was proud of.

1. Create a calendar
2. Social networking aspect – profiles – groups
3. Database for profiles, groups and events
4. List all events and groups and have some sort of search for them
5. Login and register page
6. Be able to edit and save user, group and event information.
7. Create a calendar

I have created a calendar and shows all the months of the year and can carry on going to further years or can go into the past years as well. It also caters for leap years. The calendar also highlights in red the current day which was everything I wanted it to do originally.

1. Social networking aspect – profiles – groups

As seen in my testing each user has their own separate profile. I’ve done this utilising cookies, they are holding the UserID from when they log in. Similarly, each group has their own unique page this is also done by allocating the GroupID to a cookie.

1. Database for profiles, groups and events

Using MySQL I successfully created a database with 5 tables. I then used Docker which is a software that puts my database into a container that can be accessed from anywhere making my database mobile. Docker also decreases deployment times making my project run quicker as a vital part of my code is SQL queries.

I also used node js as it’s useful for back-end API services this is due to its single threaded nature. A lot of my code is just something happening when a button is pressed or when a page is loaded so it’s very useful.

Initially I had the three main tables, Users, Groups and Events but since they share a many to many relationships between themselves, I had to create two new linking tables: GroupMembers and EventGroups which link the respective tables together one to many relationships.

1. List all events and groups and have some sort of search for them

When the groups page or the events page is loaded the user is greeted with a table full of all the groups/events that have been created. On each of these pages is also a search bar with a search button so when they user enters a group/event name then press search the table will display all the groups/events that begin with the users input. If they users search also returns only one result, then they will be redirected to that group or events unique page.

This is all done through Ajax calls, API, and SQL queries. Ajax calls take the user inputted data and send it to the API for it to use the data to run the custom SQL query. Also by pressing the reset button the page is reloaded and the original table with all the results is shown again.

1. Login and register page

Users having individual accounts is a major part of my project so having a login and register page is especially important. A user can enter their own username and password then click register to create a new account. They can then re-enter this information to log in, this will send their username and password to the API to check it with the database to ensure the credentials are correct in order to get redirected to their profile.

When registering an account, the user's plaintext password is hashed using a salt then saved as the password in the database. This is done in case the database is breached then the hacker still cannot get the users actual password as we only save the hashed.

Throughout the project there is a log out button located at the top of the screen at all times, whenever this button is pressed it will first delete the user’s cookie holding their UserID then redirect them to the login page.

Also, if at any time the cookie holding the UserID is deleted or expires then the user is sent straight to the login page to log in again. This stops people leaving themselves logged in then someone else going on their account.

1. Be able to edit and save profile, group and event information

As seen in the tests the user can edit their profile page and change data such as their first name, last Name, bio or email then when they click the save button their data will be sent to an API through an ajax call which will then start an SQL query to update the database with the users new inputted data. This is done exactly the same in the groups page as well as the events page.

When clicking the edit button, a delete button also shows up which when clicked will delete the profile, event or group from the database, delete the cookie holding the ID information then redirect the user to either the login page, the groups page or the events page depending on what they have deleted.

**Conclusion**

After finishing my project I'm glad to say I completed all the objectives that I originally gave myself. The project was quite difficult for me, but I found it to be a great learning experience and I learnt important skills that I can definitely say I will be using later in my career.

**Feedback from Users 5.2**

Chart, bar chart

Description automatically generated

The first question I asked users was what is the most useful features. Winning by a lot was the Calendar and I'd agree, not only is a calendar very useful but it also has the most potential to be a huge part of the social network app. I think if I had more time, I would definitely focus on the calendar and make it something really special.

An outlier in the data shown was the Events, they only received two votes. Events do not do a lot in my project, but they also have the potential to be something special. I think by developing the calendar the events would also develop as you could link the events to specific dates in the calendar also you could link the dates to specific groups making them group exclusive or you could have multiple groups contain the same event as if it were a group lunch between companies.

Chart, pie chart

Description automatically generated

When investigating the different websites and apps in my analysis chapter I noticed how nice all of the user interfaces were and how simple they were to follow. Also when doing my survey a huge number of participants mentioned that the clean and simple website is something that pulls them in to using it and especially coming back to it.

So when I received the overwhelming amount of yeses to this question I was very happy with how I did. The reason my interface was so clean was down to using bootstrap to help with my html, they simplify the html code and also have a lot of templates to take inspiration from as well as tutorials and resources to learn from.

Chart, pie chart

Description automatically generated

The search bar was something I wanted to add in instead of just having a button next to each possible group to get their page. I thought it would have enhanced the users experience by speeding up how they find the groups they want. The search bar also becomes more effective the more groups or events there are which might mean I wasn’t able to utilise it as much as I possibly could due to my low amount of groups of events but If I was to launch this app globally this chart shows me how the users agree with how useful it would be.

Chart, pie chart

Description automatically generated

The results of this question came back quite average which is understandable for a unfinished project. Compared to big sites like Facebook, Instagram or snapchat a simple website doesn’t match up but with a similar yes to no’s shows how there's potential for a very promising website.

As I discovered from my research in the analysis chapter the biggest differences between the big social networks is the content at display, and the UI. I can proudly say the interface of my website was clean but can definitely be improved.

**Potential improvements 5.3**

One of the things I wanted to add but did not have enough time to do was the implementation of google location API with my event page. By doing this I could have shown exactly on google maps where the event location was and also provided directions to the event from the users current location.

Similar to that idea I could've had a new page that was a map of the local area and it had pins of all the events that were happening nearby. This also could have been used to show a user’s location after they had given permission, similar to how snap maps work on snapchat.

I could have also maybe added a chat room feature to the groups page or maybe a feed where you can post a small message from your account onto the group page or a feed similar to how facebooks feed works.

Another thing I could have added was maybe some customisation to the users profile page, possibly letting them do their own css like oldstyle myspace did or giving them colours and background pictures that they can either pick from or take from the internet themselves.

**References to external sources of information**

|  |  |
| --- | --- |
| https://www.w3schools.com/js/default.asp | Used to learn javascript |
| https://www.w3schools.com/html/default.asp | Used for html |
| https://getbootstrap.com/docs/4.0/examples/ | Used for inspiration for my html designs |
| https://www.w3schools.com/js/js\_cookies.asp | Used to learn to use cookies |