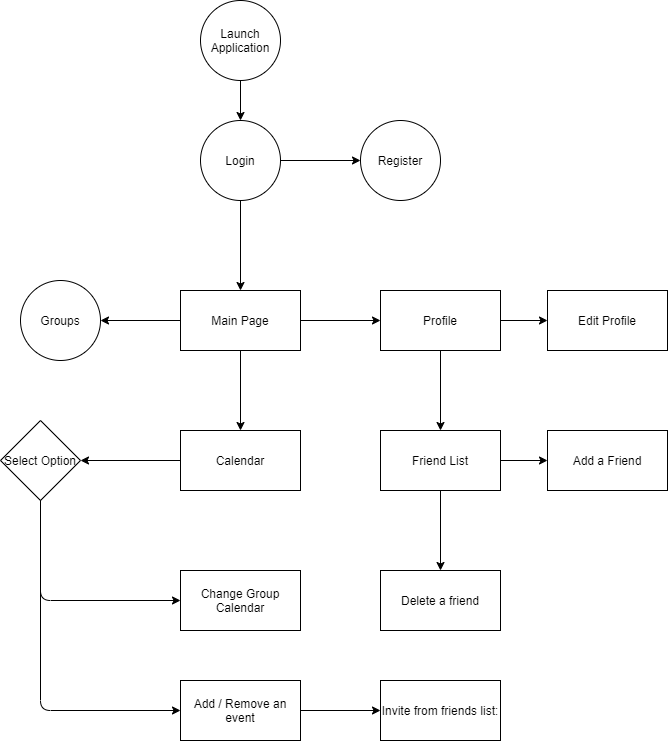
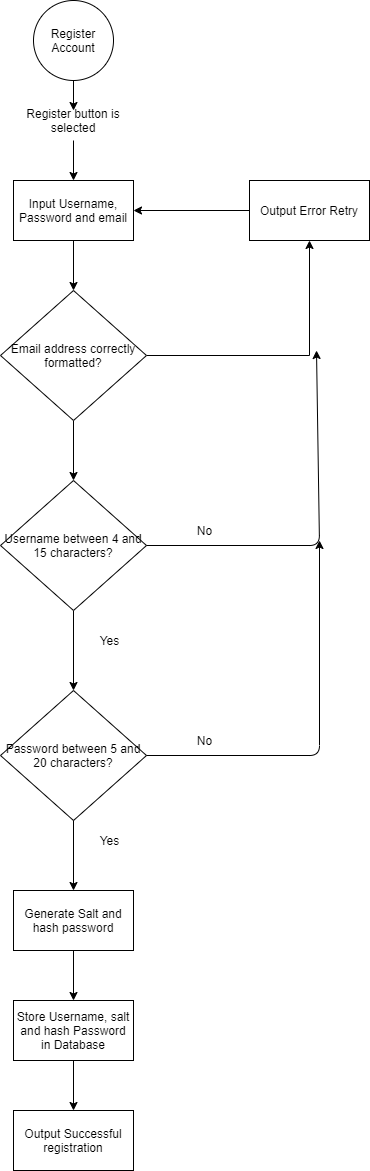
**Design–2**

**High level Overview – 2.1**

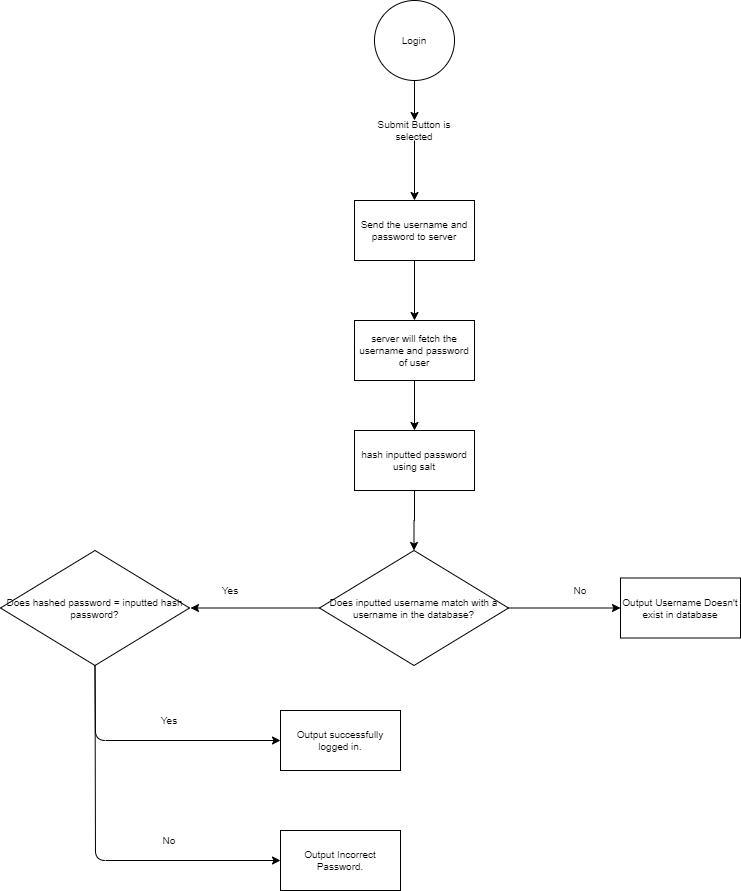
Main Menu – Flowchart



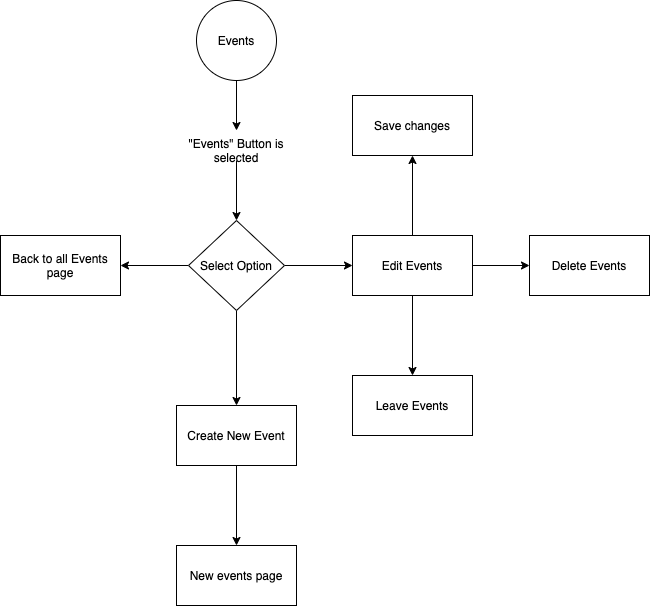
Register - Flowchart



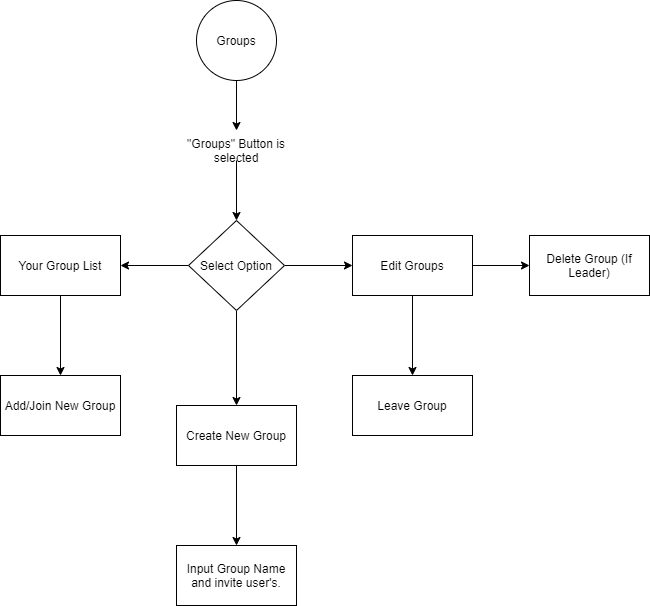
Login – Flowchart



Events– Flowchart



Groups – Flowchart



**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.

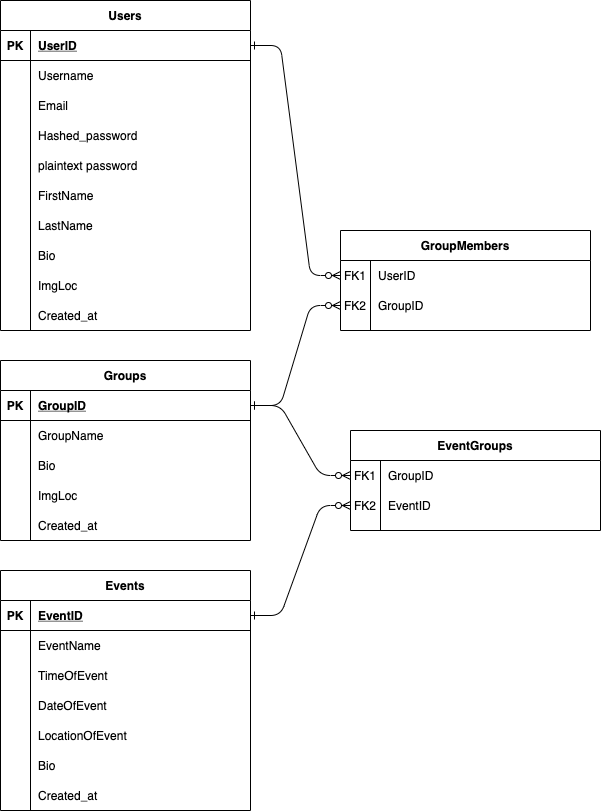
The “Groups” table stores information about the user created groups, it stores their GroupID, GroupName, bio, img loc and created at.

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.



**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”,”Luke@email.com), “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,

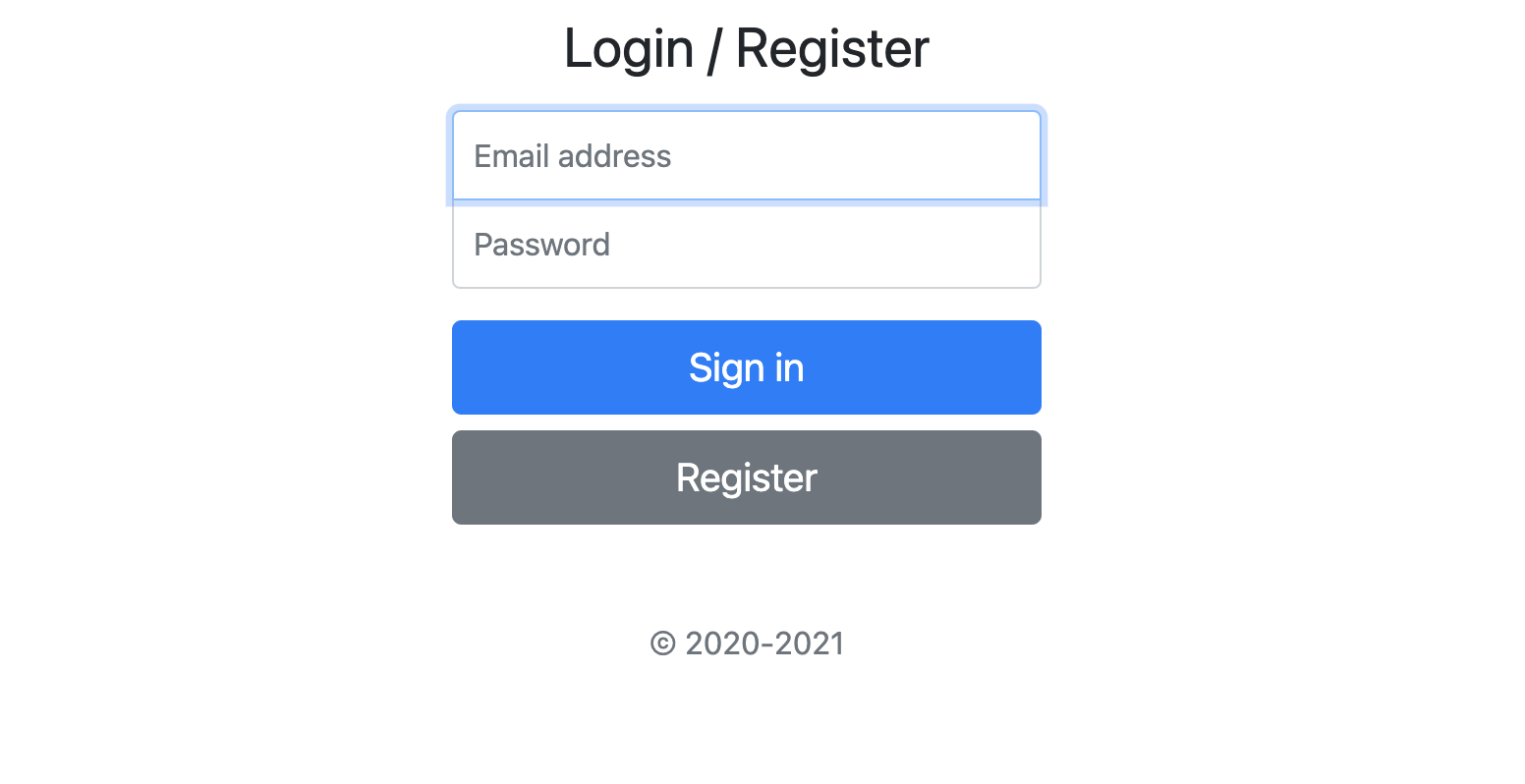
GroupID INT(6) NOT NULL,

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP

)

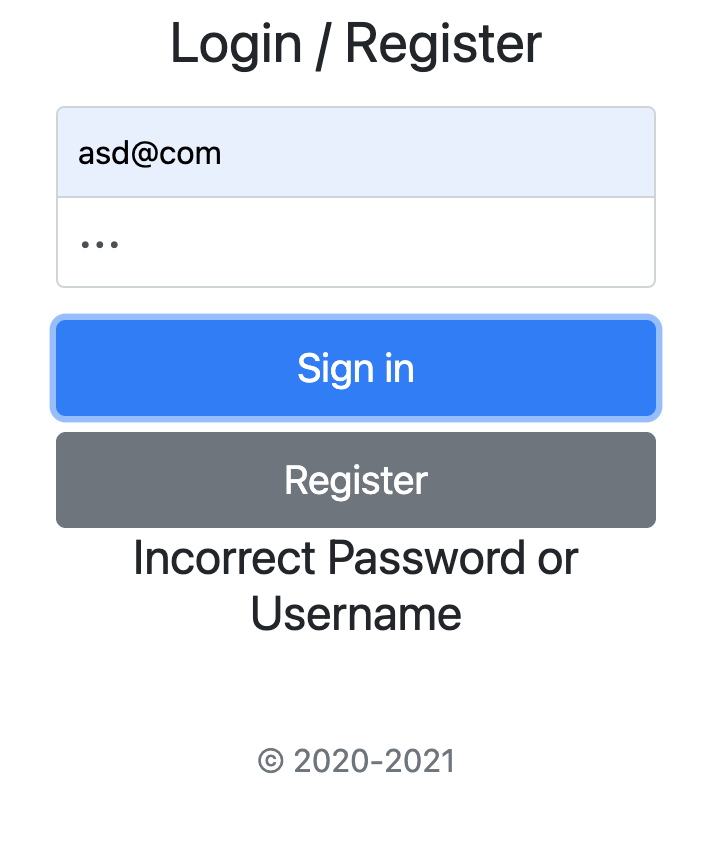
**UI Design 2.7**

**Login/Register Page**

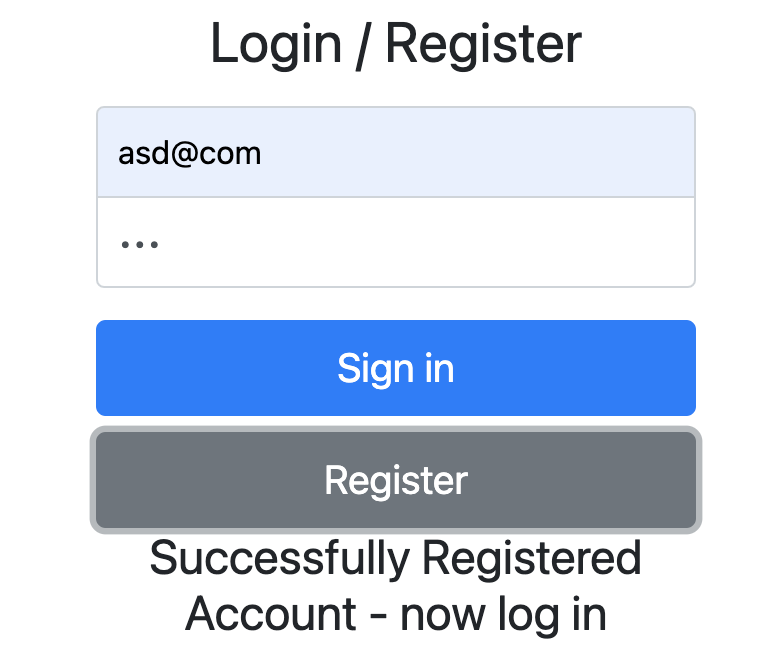


The is the first screen that will be loaded up after the application is launched. One of the main features of this project is the profiles and groups so therefore each user must have a unique login which gives them a unique UserID. This unique ID will be saved to a cookie which will then be used to keep 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.

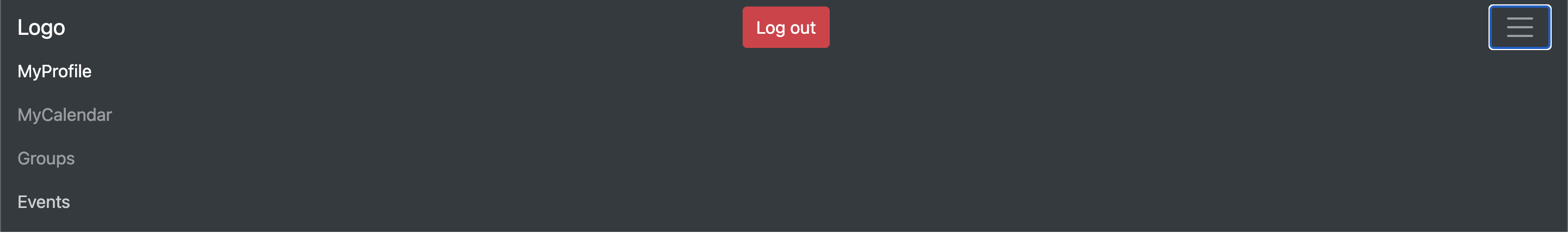


If the user has entered the wrong credentials, then this warning will come up on their screen telling them they’ve entered the incorrect password or username. They will be able to try and log in again.



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.

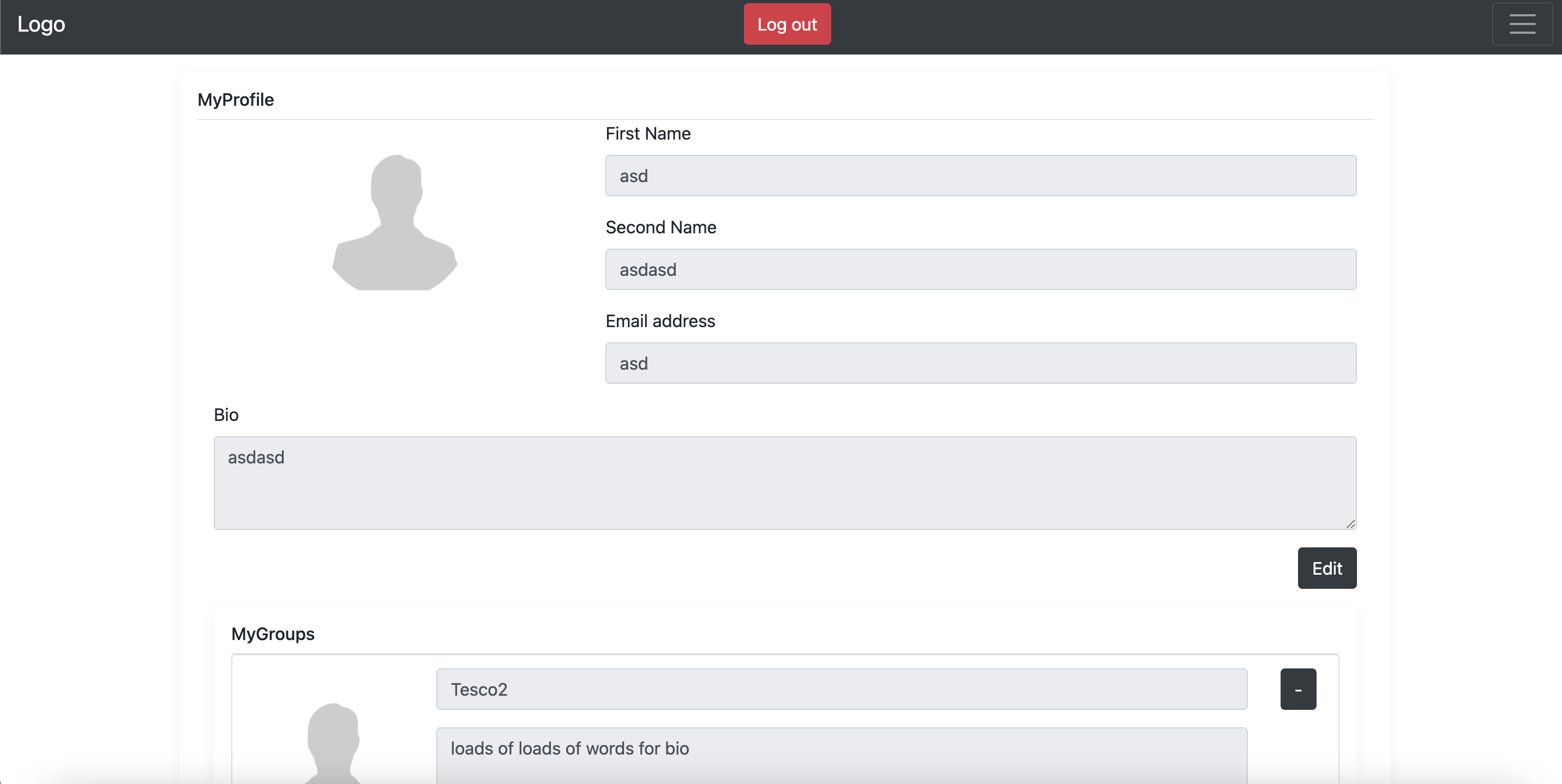
**The pull-down bar**



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.

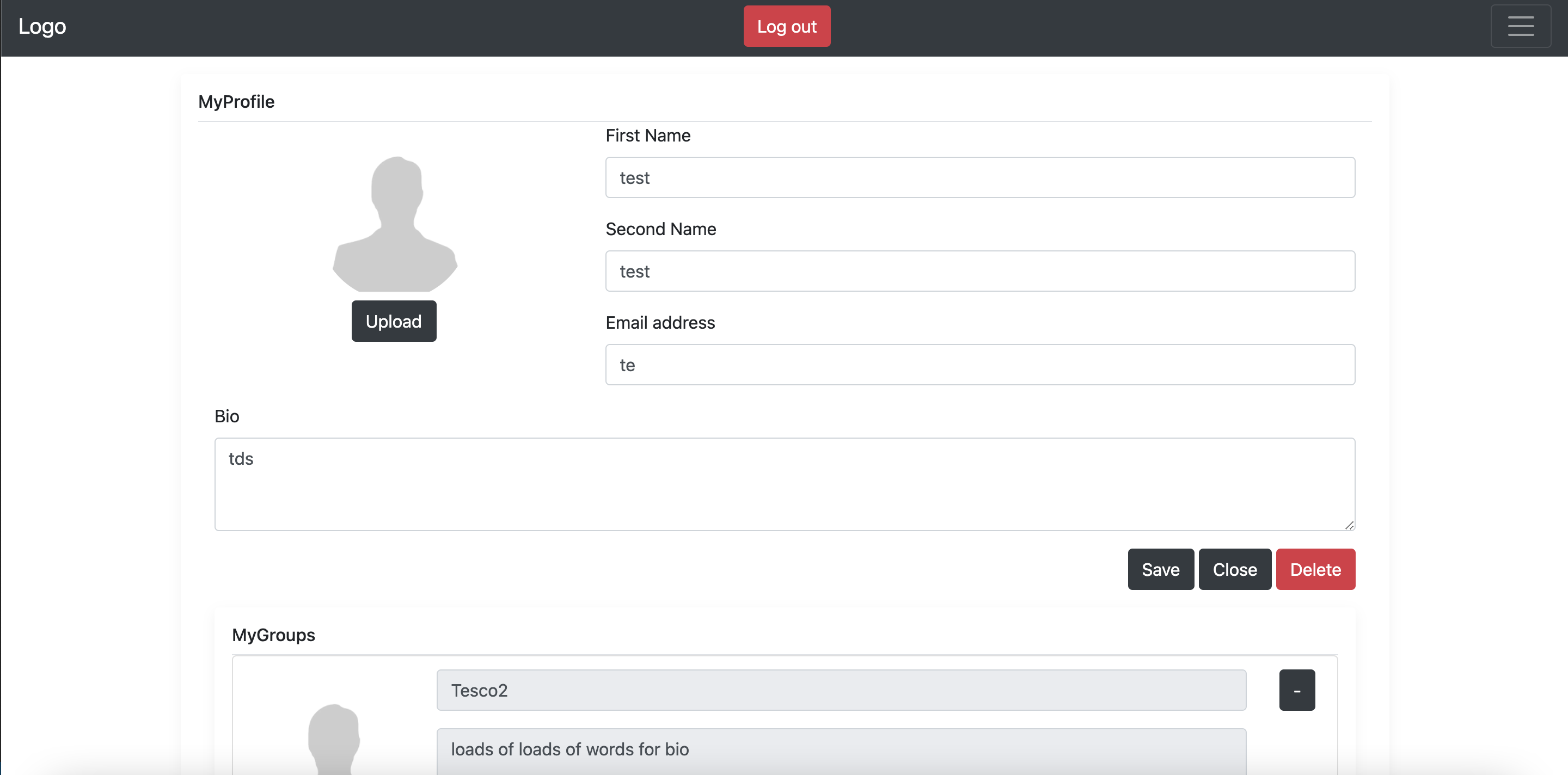
At the top of the pull-down bar you can see the “Log out” button. Since this pull-down bar is on every page you can log out from any page. When you click this button your cookie holding your UserID is deleted, and you are redirected to the login/register page. As your cookie is deleted you need to log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

**Profile Page**



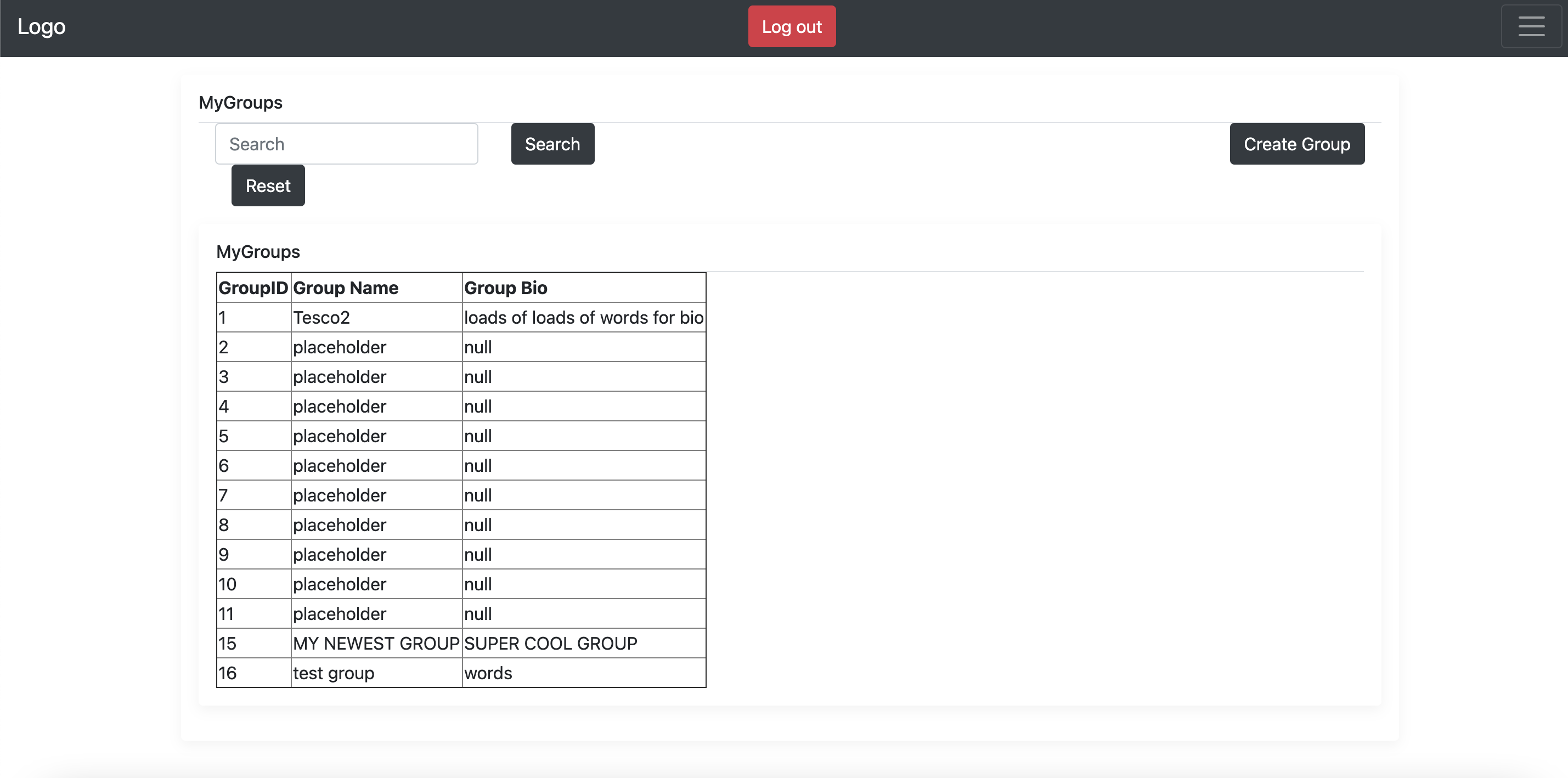
After logging into their account, the user will be presented with their personal profile page which will show all of their information and their groups. If the user has just been registered then they will be shown a screen like this one but with no information in the boxes.

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.

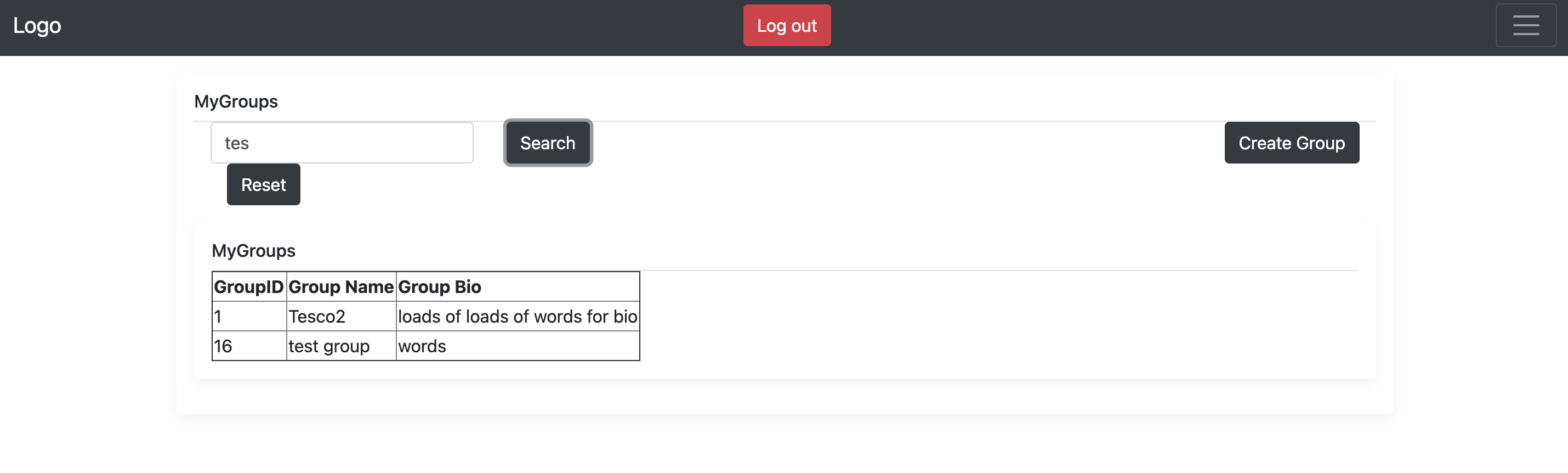


This is what the screen looks like when the edit button is pressed. The edit button disappears and three buttons appear, the “save”, “close” and “Delete” buttons. Also, the boxes that were initially greyed out are now white and can be edited. The user is allowed to change their information to whatever 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**



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.

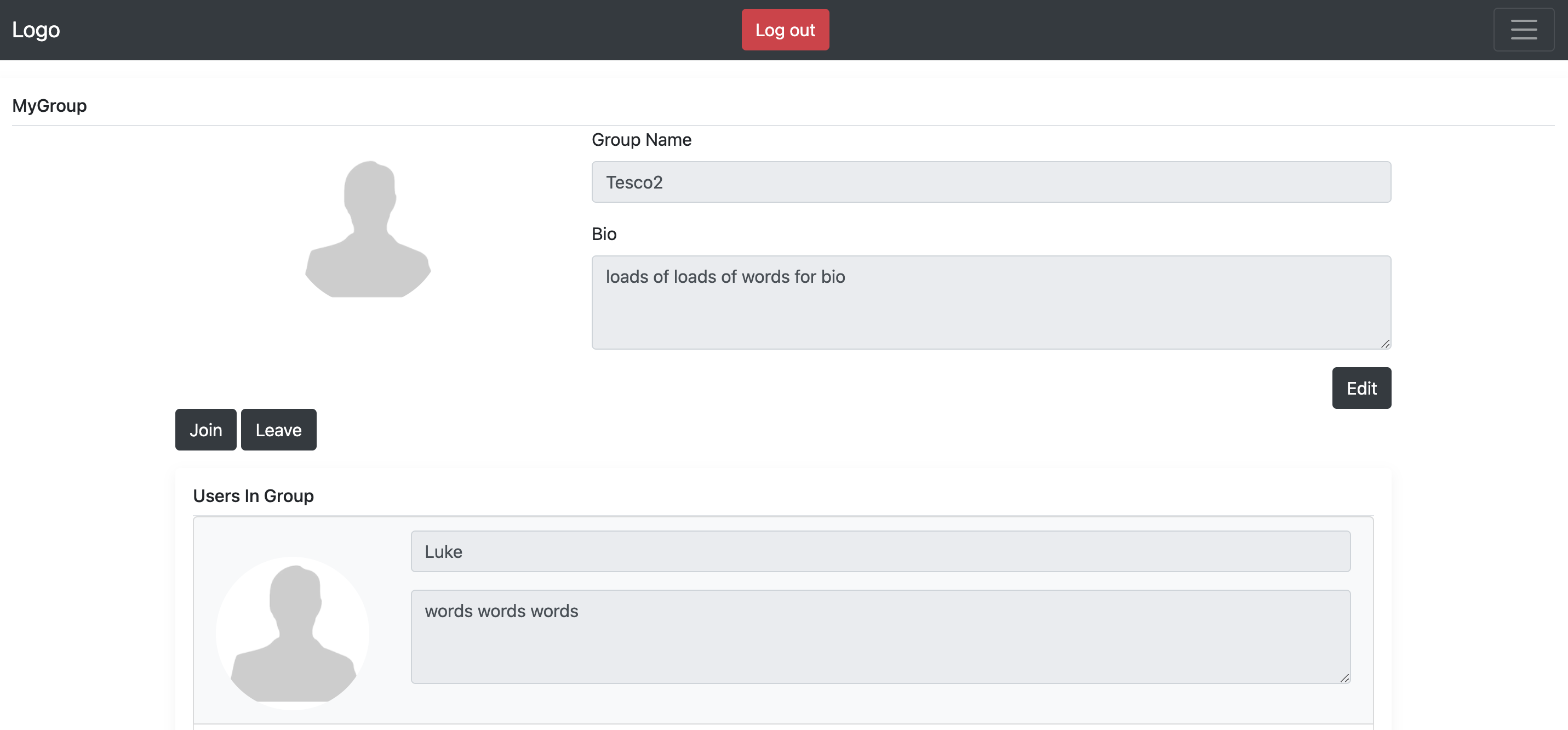


When the user searches “tes” it returns with the two groups that start with the user's input. When the user searches for the group's name exactly it will lead the user to that groups group page. The group they search for will have its GroupID saved to a cookie which will be used to get to its unique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

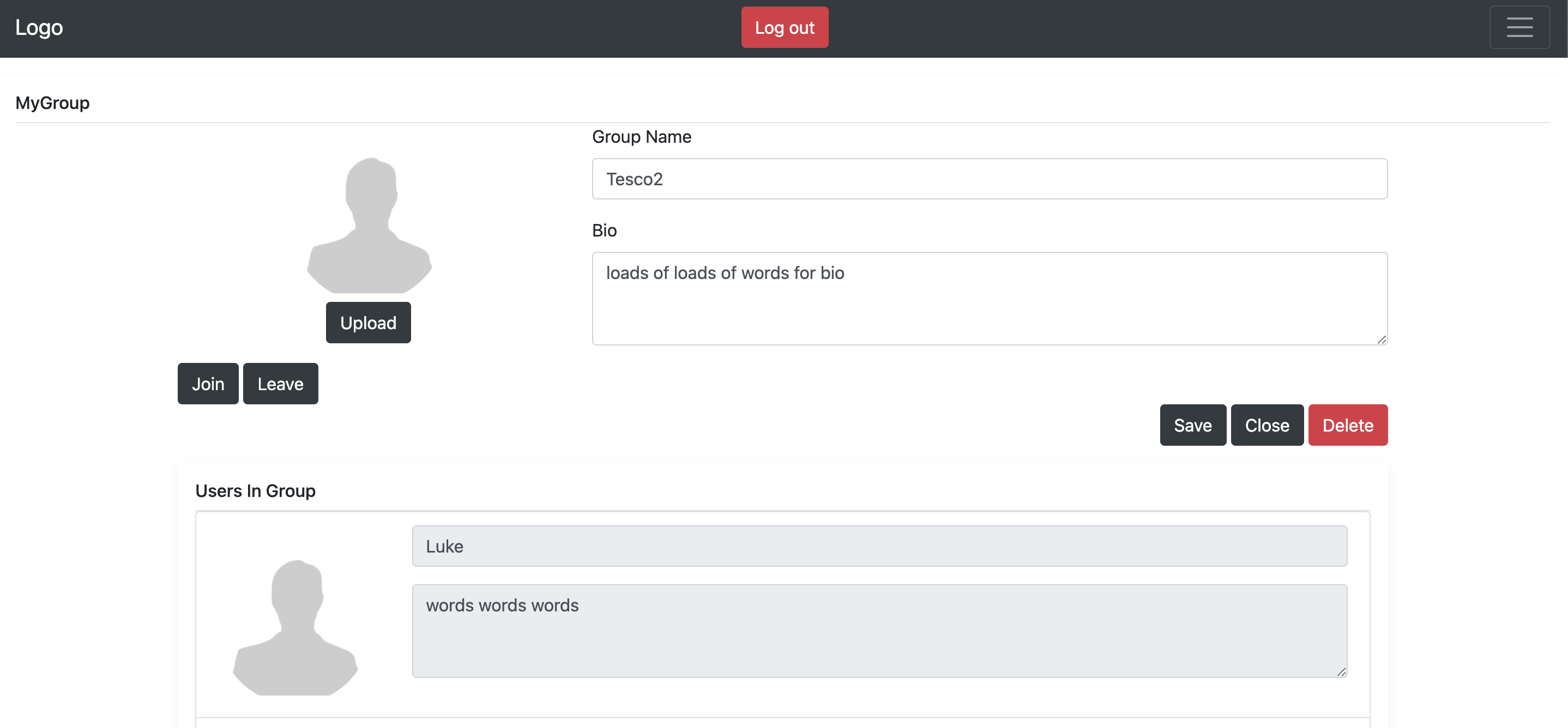
When the user clicks the “Create Group” page a new group is created and a cookie is set to the new GroupID of the group. The user is also instantly redirected to the new groups group page.

**Group Page**



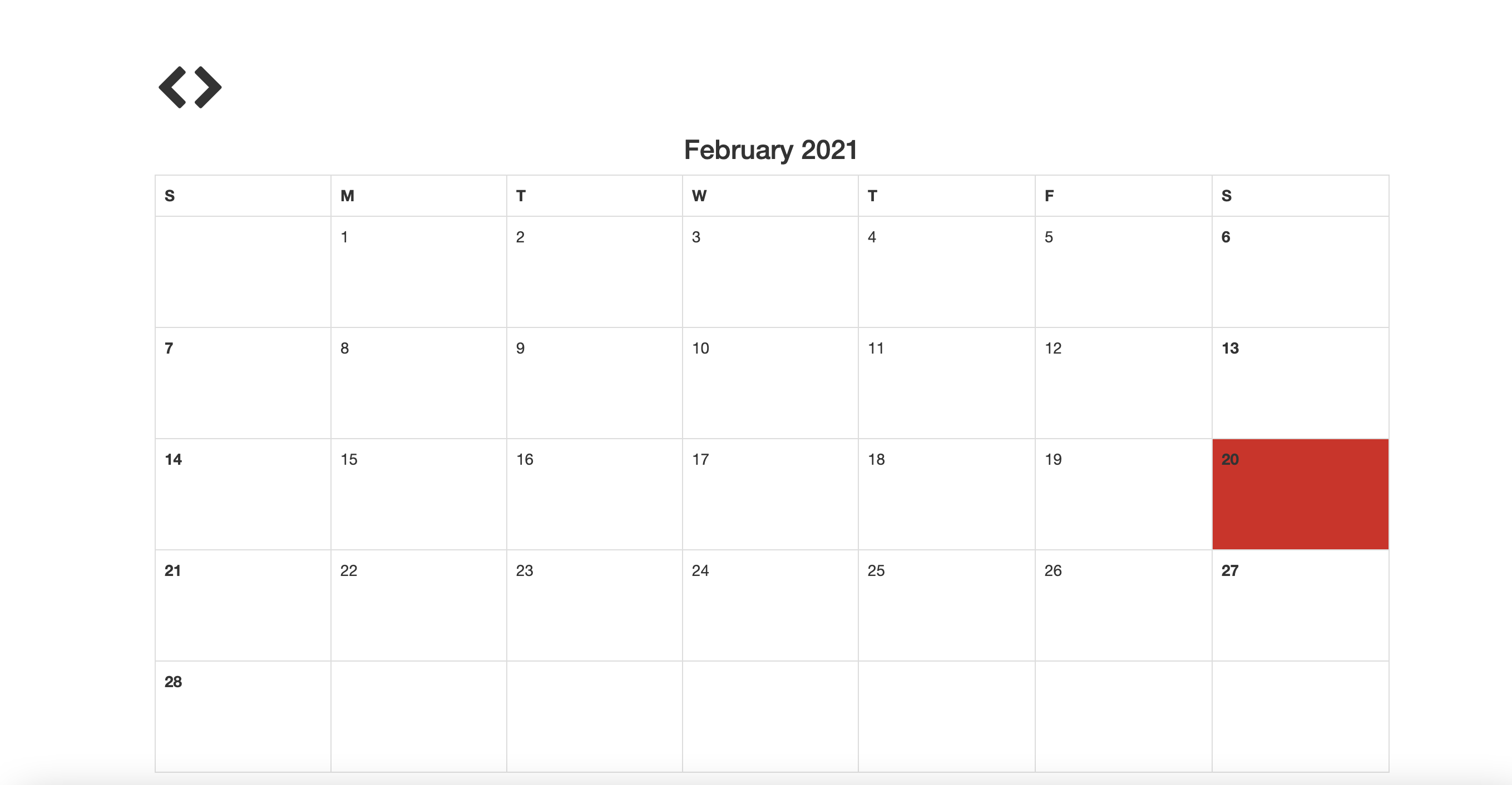
After the user has searched for an individual group or they have created a new group they will be redirected to this page. Similar to the profiles each group has their own separate page that displays their Group name, Bio and Users in the group. Just like the profile page you can click edit and then 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.



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.

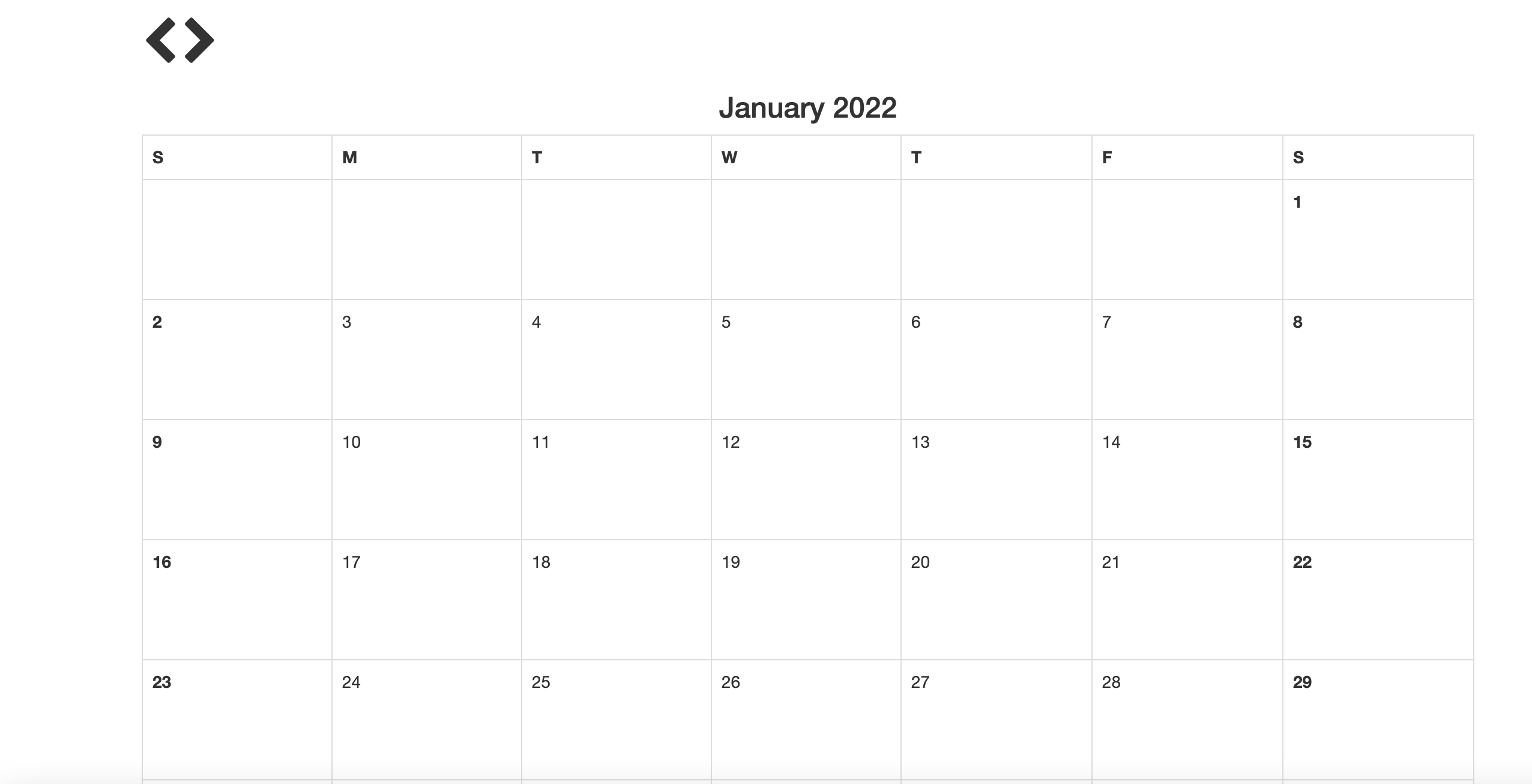
**Calendar**



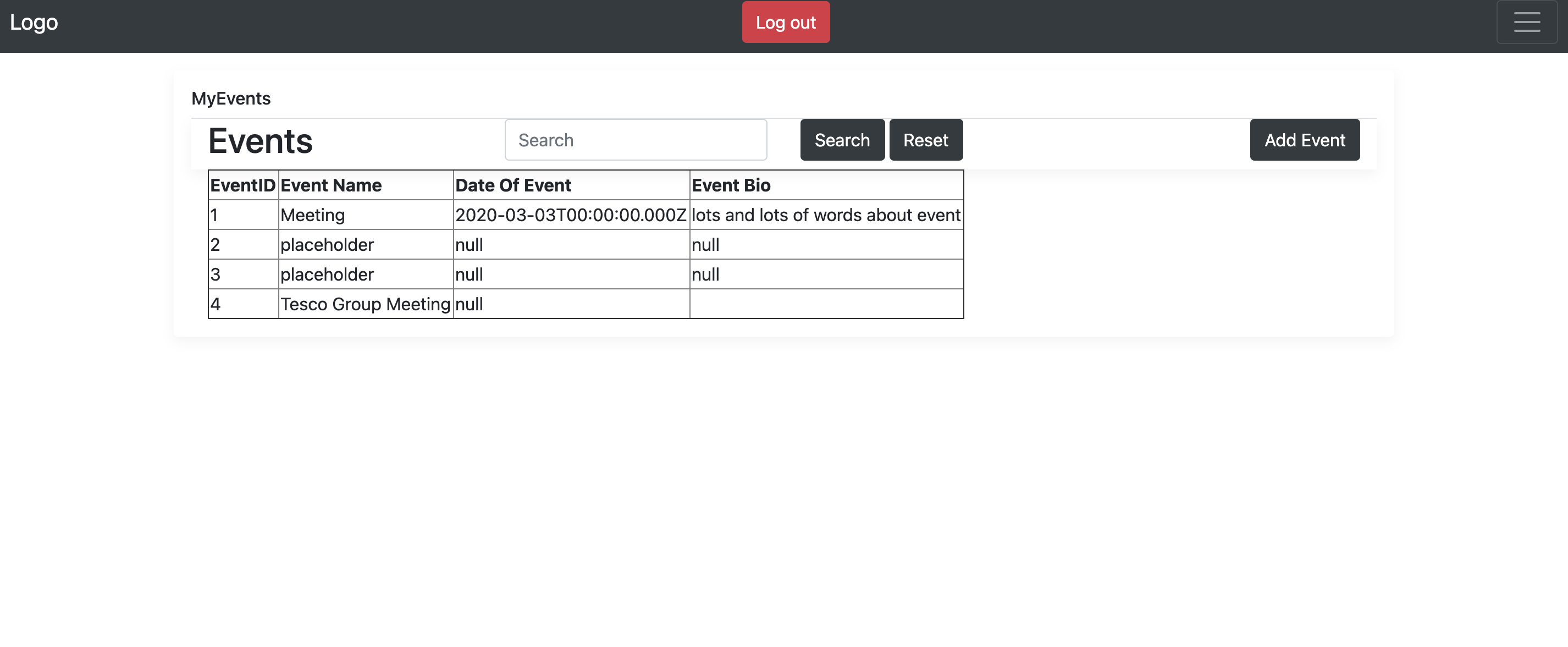
This is a simple calendar that displays all the months and the year. The red box highlights the current day.



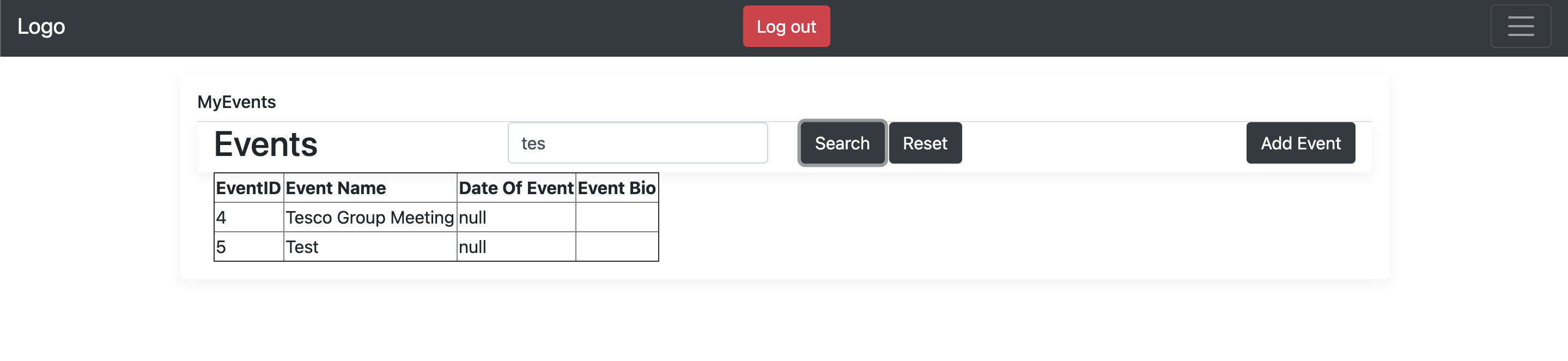
By pressing either the left or the right arrow you can go to the next month or the month before. The calendar keeps going and won’t stop just because you're at the next year as seen in this photo.



**Events Page**



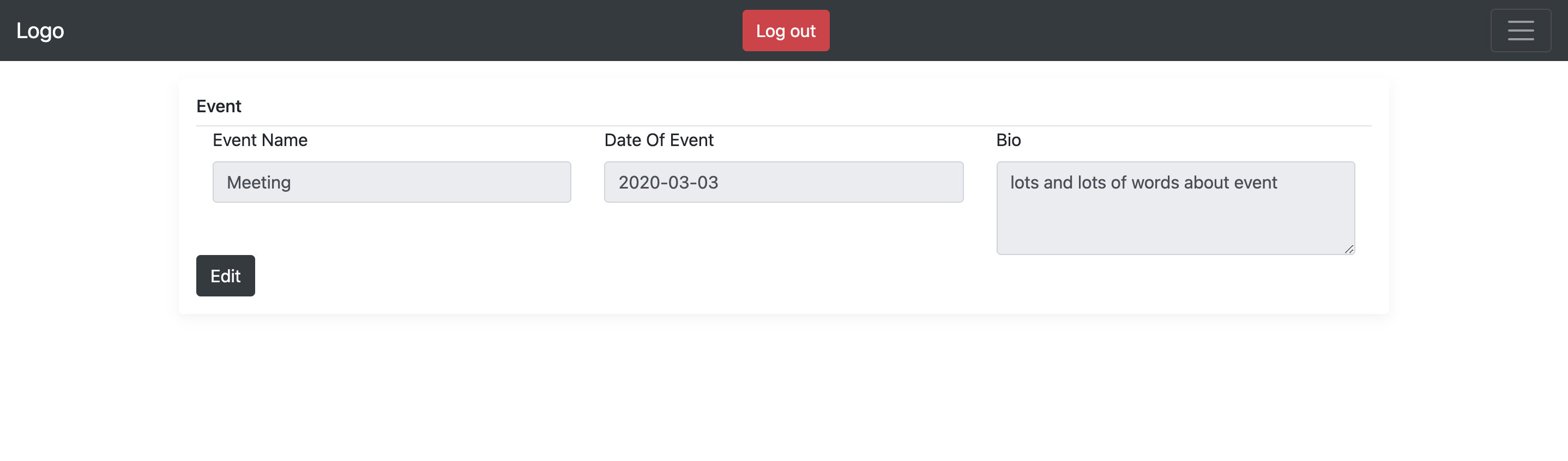
This is the Events page which you can get to by clicking the “Events” button on the drop-down menu. Similar to the Groups page it displays all of the events and each event has its own unique event page as each event is given an EventID. The search function works the same as how the Groups page does.



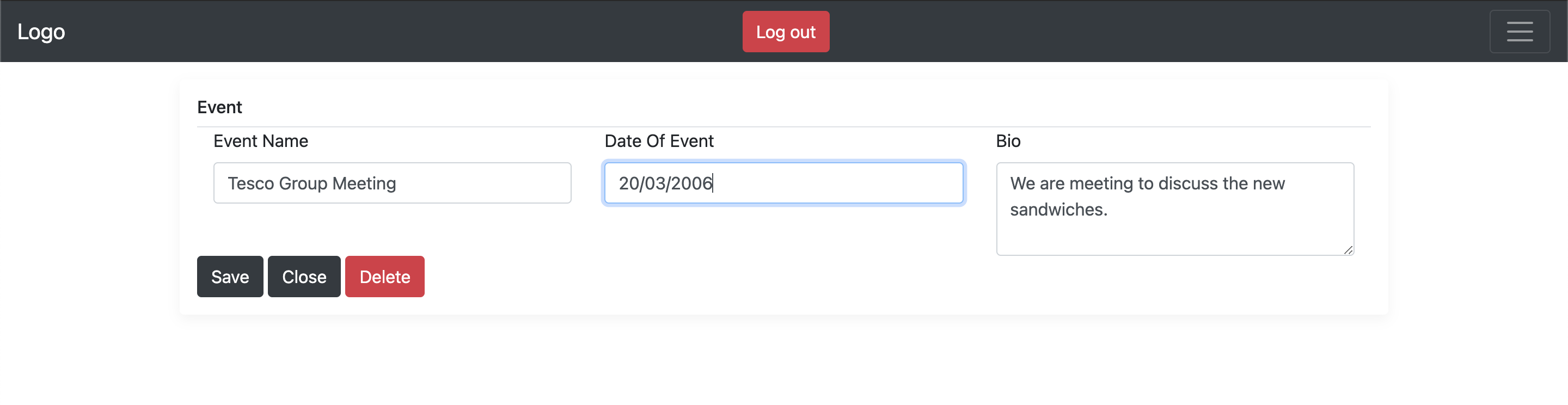
As we see in the photo once the user has typed something in the search bar and pressed the “search” button the table is updated to only display events that start with the letters that the user has inputted. If they want to go back and see all of the events then they can press the “Reset” button which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



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.



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 in my code as it’s quite an important part.