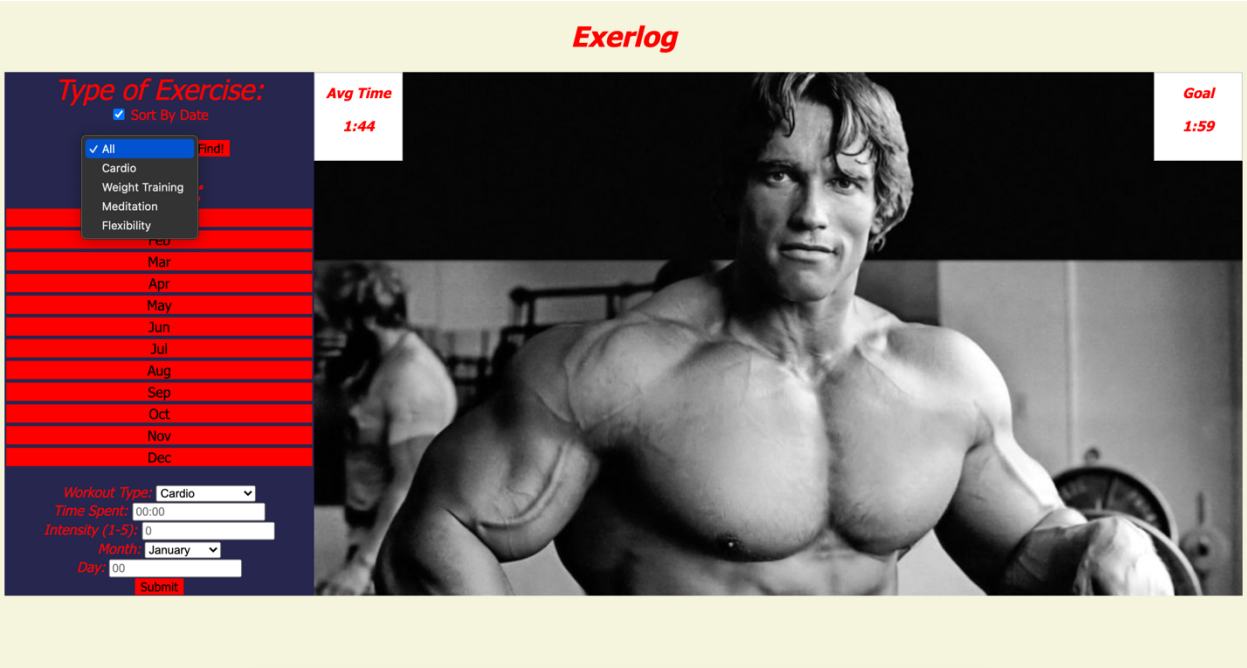
CPSC 314 Web Project Deliverable 4
Exerlog
Lucas Abeln
12/6/21

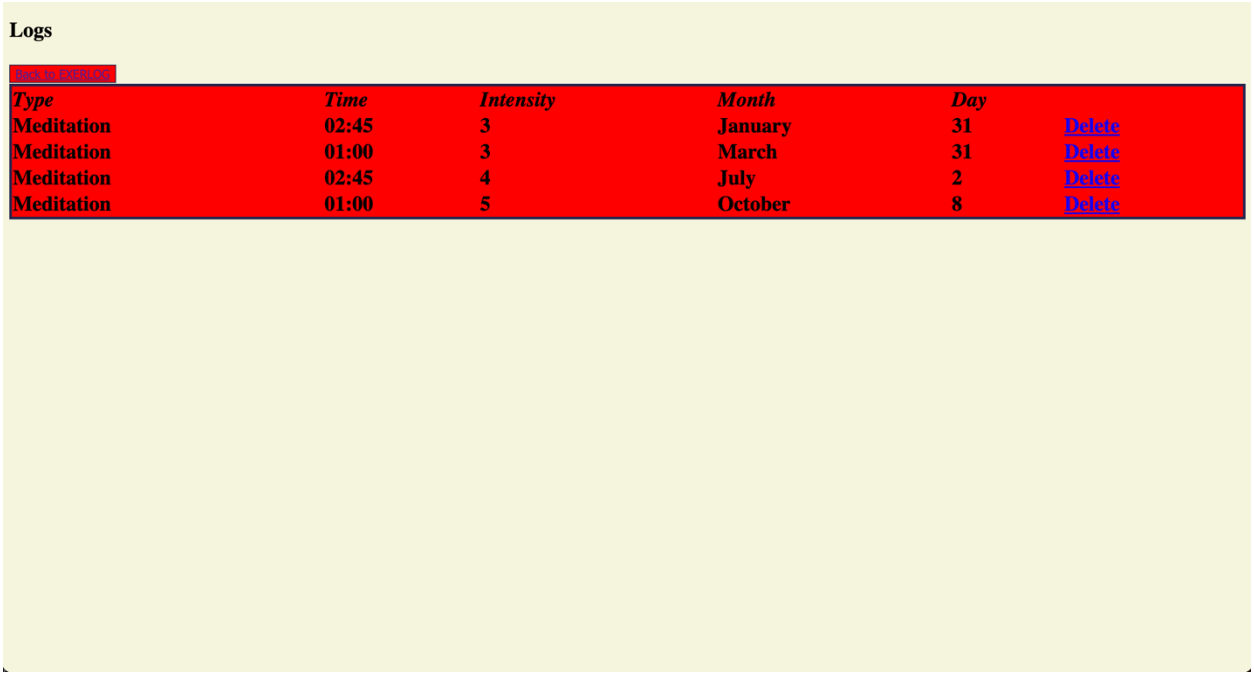
Github Link: https://github.com/Gonzaga-CPSC-Fall-2021-Olivares/cpsc-314-web-development-final-project-lucasabeln.git

Front End Functional Requirement 1

Must be able to use a dropdown to sort between workout categories

The user is able to sort logs by workout category using a dropdown menu on the main page of the site, the user can select any of the individual categories or select "All" to see every workout log that has been submitted. After choosing the category, you simply press the "Find!" button to show the results of the query.





Logs

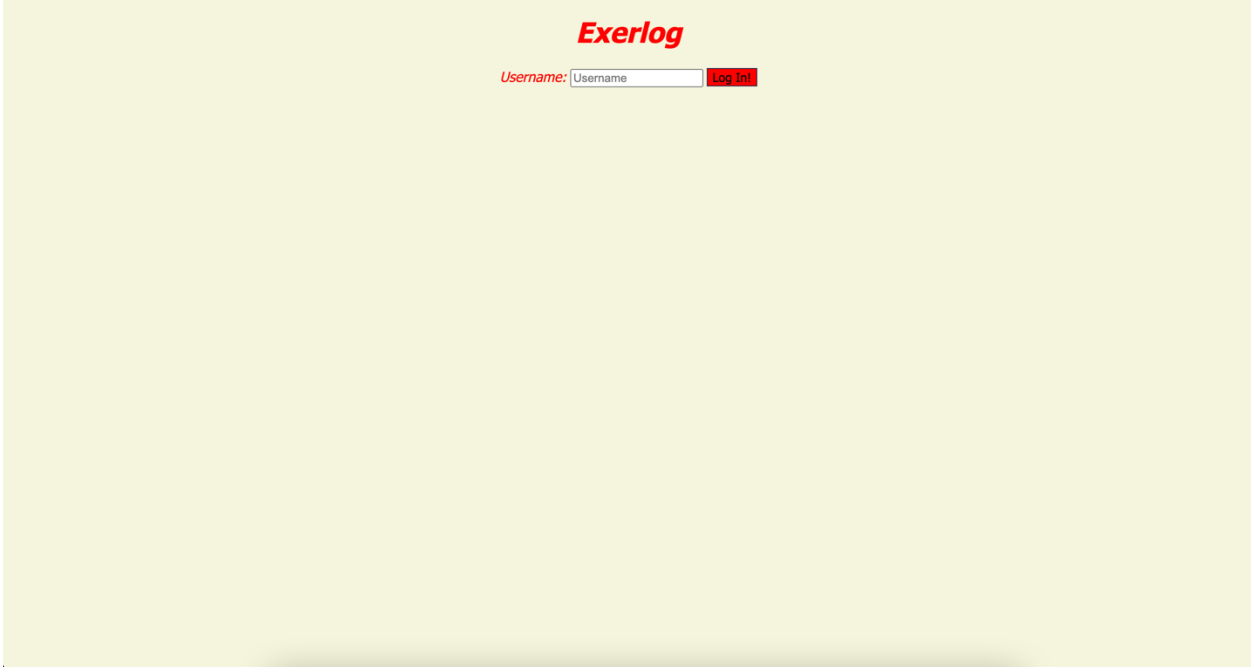
Back to EXERLOG

Type	*Time*	*Intensity*	*Month*	*Day*	
Meditation	**02:45**	**3**	**January**	**31**	**Delete**
Meditation	**01:00**	**3**	**March**	**31**	**Delete**
Meditation	**02:45**	**4**	**July**	**2**	**Delete**
Meditation	**01:00**	**5**	**October**	**8**	**Delete**

Front End Functional Requirement 2

Must be able to log in to a user profile or create a new profile

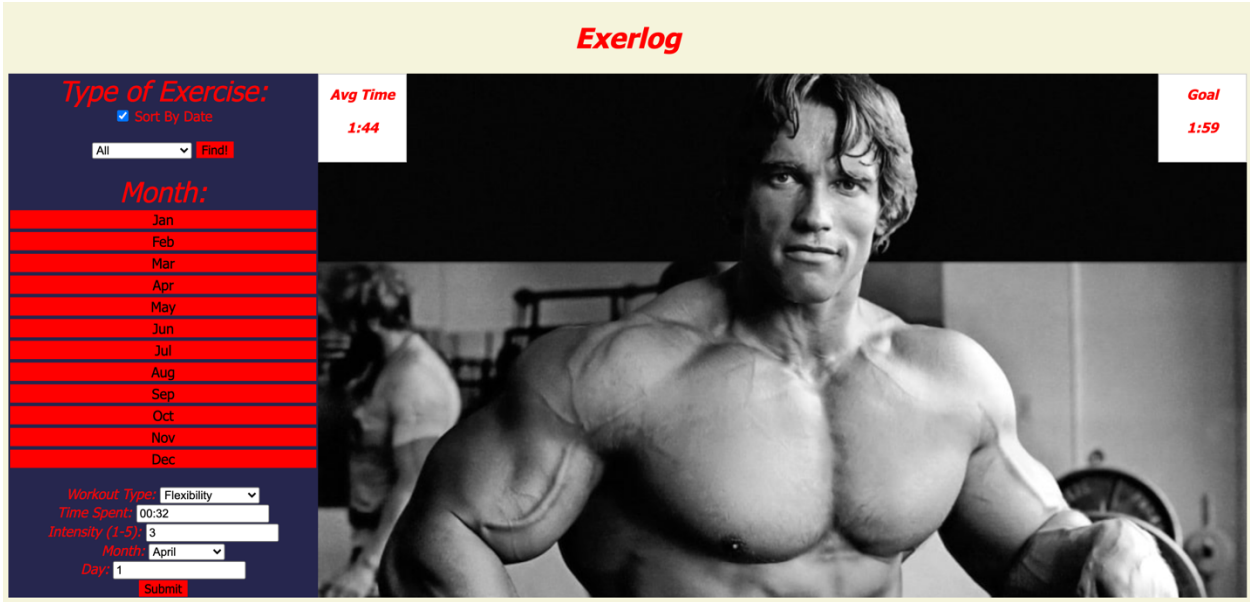
On the front end, the user logs in by entering their username into the first page which they encounter. After entering their login information, they are brought to the main exerlog page where their username is saved in the backend. The user must press the “Log In!” button to continue onto the main page.



Front End Functional Requirement 3

Must be able to create a new work out log

To create a new workout log on Exerlog you enter the needed information into the form on the left-hand side of the page. The user enters the workout type, time spent working out, intensity, month of entry and day of entry. The data from that form is then sent to the back end and the user is forwarded to the logs page.



Logs

Back to EXERLOG

Type	*Time*	*Intensity*	*Month*	*Day*	
Meditation	01:00	5	October	8	Delete
Meditation	02:45	3	January	31	Delete
Cardio	01:23	2	July	8	Delete
Meditation	01:00	3	March	31	Delete
Meditation	02:45	4	July	2	Delete
Flexibility	00:30	1	November	13	Delete
Flexibility	01:00	5	October	23	Delete
Cardio	01:23	2	September	8	Delete
Cardio	01:23	4	January	12	Delete
Weights	2:13	1	December	11	Delete
Cardio	02:45	2	April	2	Delete
Cardio	03:23	3	February	21	Delete
Cardio	01:23	4	December	12	Delete
Cardio	01:23	3	May	8	Delete
Flexibility	00:32	3	April	1	Delete

Front End Functional Requirement 4

Users must be able to toggle sorting logs by date

The user is able to toggle sorting the lists by category by hitting the checkbox beneath the “Exercise Types” label. If the checkbox is checked, the results of hitting the “Find” button regardless of type will be sorted by date on the logs page. If the button is not checked they will not be sorted on the logs page.

Logs

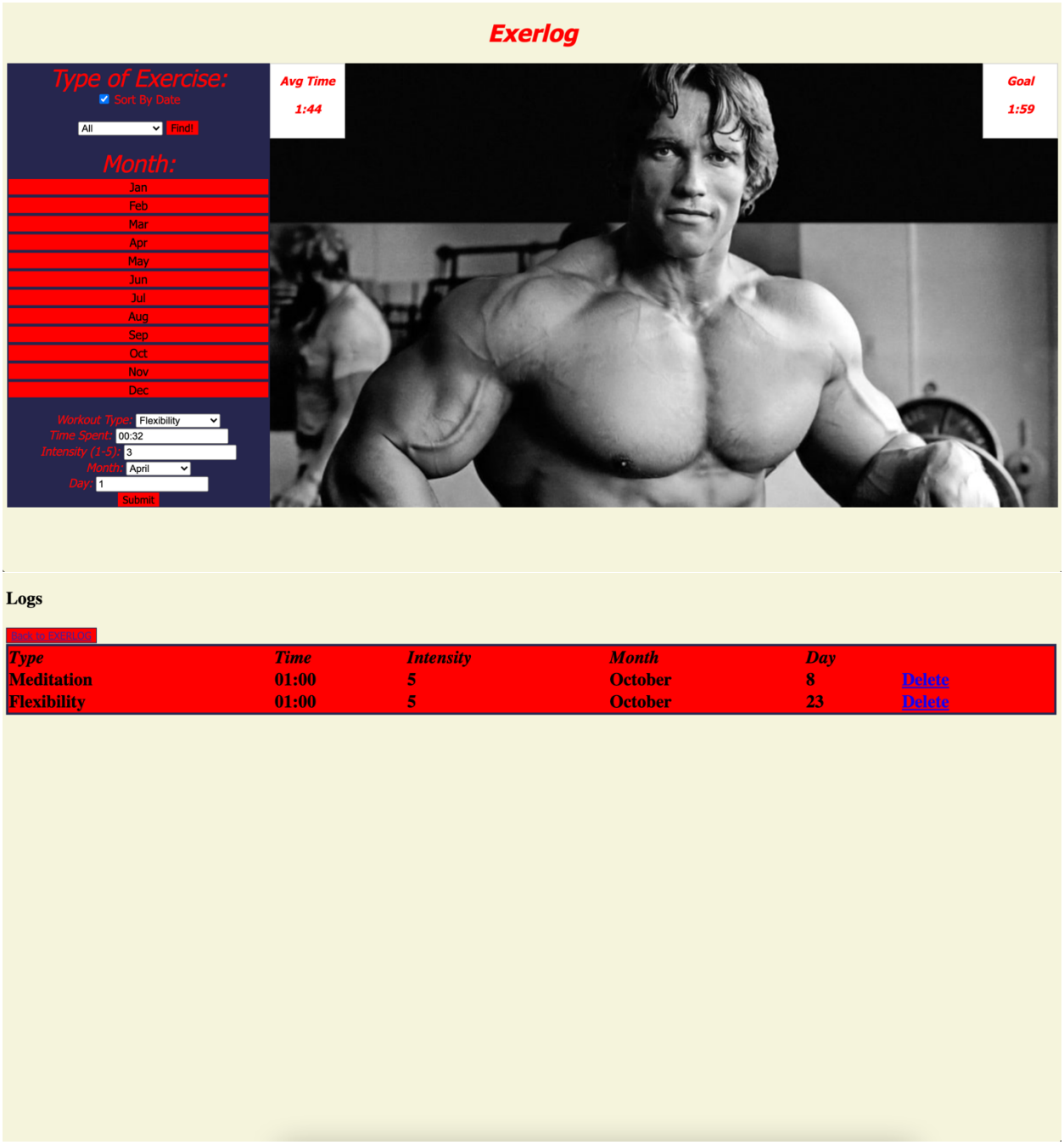
Back to EXERLOG

Type	*Time*	*Intensity*	*Month*	*Day*	
Cardio	01:23	4	January	12	Delete
Meditation	02:45	3	January	31	Delete
Cardio	03:23	3	February	21	Delete
Meditation	01:00	3	March	31	Delete
Flexibility	00:32	3	April	1	Delete
Cardio	02:45	2	April	2	Delete
Cardio	01:23	3	May	8	Delete
Meditation	02:45	4	July	2	Delete
Cardio	01:23	2	July	8	Delete
Cardio	01:23	2	September	8	Delete
Meditation	01:00	5	October	8	Delete
Flexibility	01:00	5	October	23	Delete
Flexibility	00:30	1	November	13	Delete
Weights	2:13	1	December	11	Delete
Cardio	01:23	4	December	12	Delete

Front End Functional Requirement 5

Users must be able to use calendar to access certain months logs

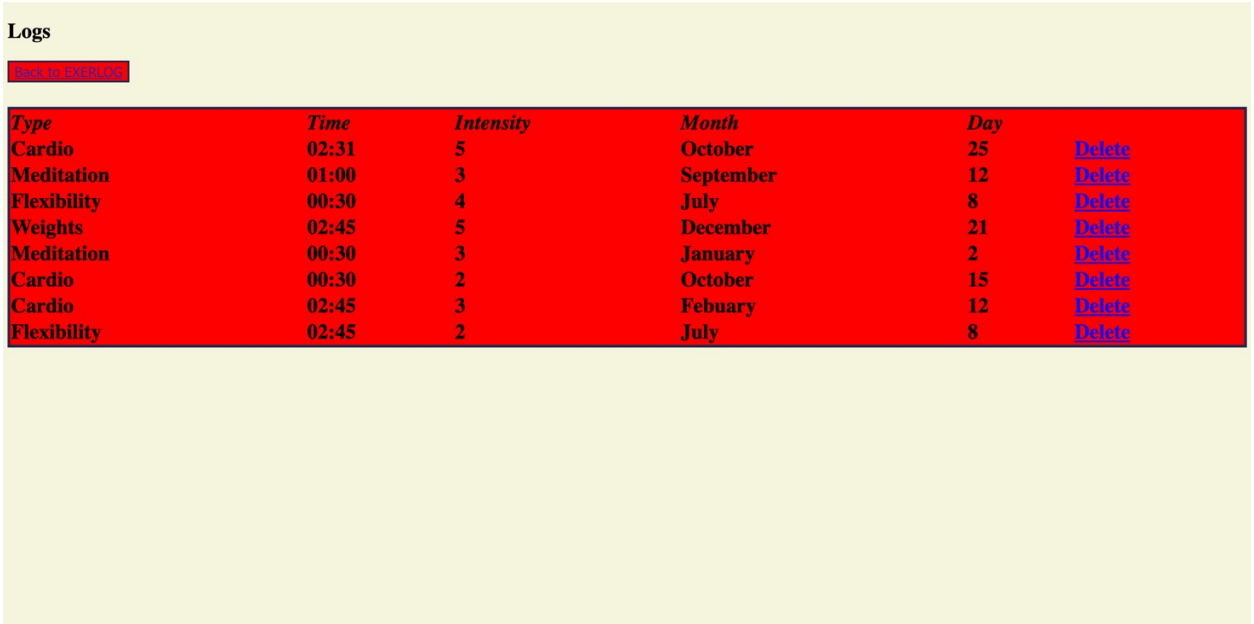
The user can press large buttons corresponding to months of the year on the left hand side of the main page to have the logs page only display logs from those respective months.



Back End Functional Requirement 1

Must log data from logs in a JSON

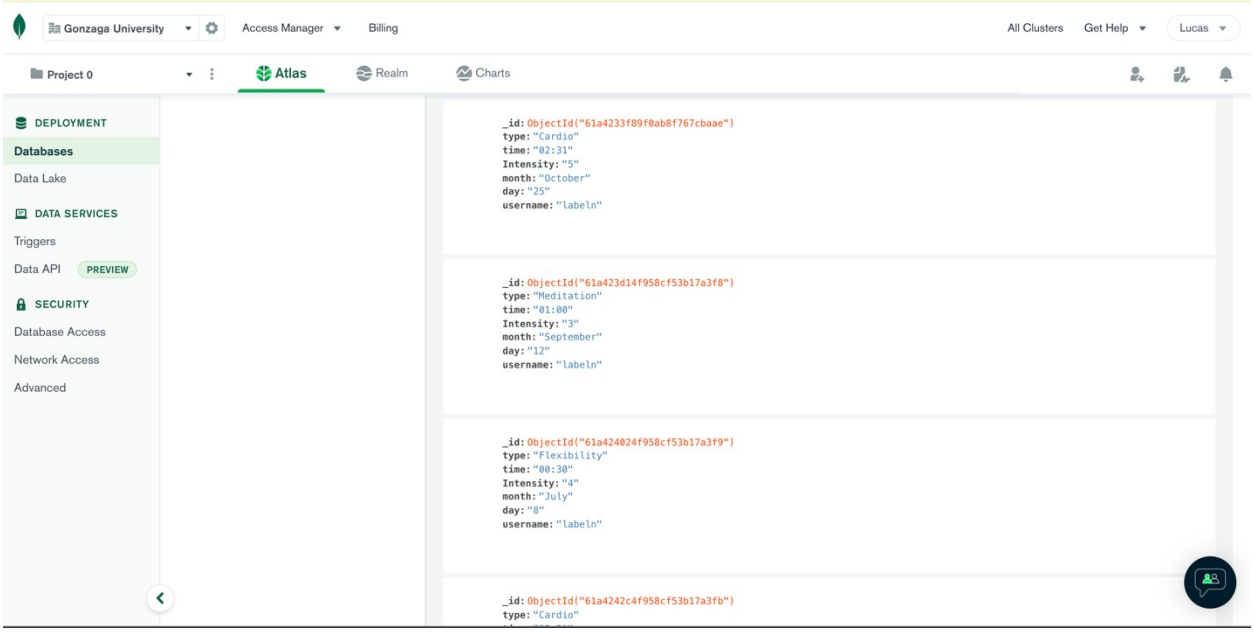
Each Log is submitted through a form on the main Exerlog page. After hitting the submit button, the users log is sent to MongoDB to be stored and the user is redirected to the “Logs” page where they can view all their logs. Inputs are taken through text fields as well as drop down menus.



Logs

Back to EXERLOG

Type	*Time*	*Intensity*	*Month*	*Day*	
Cardio	02:31	5	October	25	Delete
Meditation	01:00	3	September	12	Delete
Flexibility	00:30	4	July	8	Delete
Weights	02:45	5	December	21	Delete
Meditation	00:30	3	January	2	Delete
Cardio	00:30	2	October	15	Delete
Cardio	02:45	3	Febuary	12	Delete
Flexibility	02:45	2	July	8	Delete



Back End Functional Requirement 2

Must be able to delete a workout log.

On the logs page the user is redirected to either after hitting the submit button or choosing the month from which they'd like to view logs they will see a delete button on the far-right side of the screen next to each log. If the button is hit the log will be removed from the logs page and removed from the database itself, meaning it will no longer show up on the logs page even after logging in on a separate occasion.

Logs

Back to EXERLOG

Type	*Time*	*Intensity*	*Month*	*Day*	
Cardio	02:31	5	October	25	Delete
Meditation	01:00	3	September	12	Delete
Flexibility	00:30	4	July	8	Delete
Weights	02:45	5	December	21	Delete
Meditation	00:30	3	January	2	Delete
Cardio	00:30	2	October	15	Delete
Cardio	02:45	3	Febuary	12	Delete
Flexibility	02:45	2	July	8	Delete
Flexibility	02:45	2	July	8	Delete

localhost:3000/delete/61a5ac9b755d65dad678e873

Logs

Back to EXERLOG

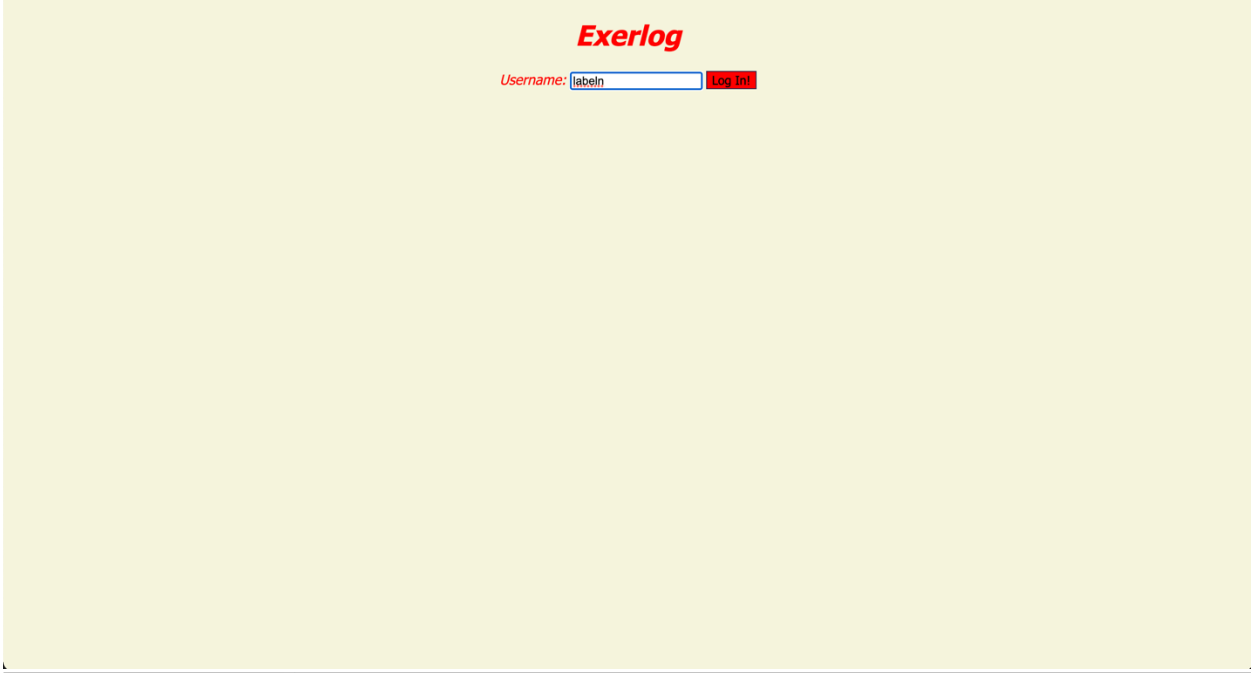
Type	*Time*	*Intensity*	*Month*	*Day*	
Cardio	02:31	5	October	25	Delete
Meditation	01:00	3	September	12	Delete
Flexibility	00:30	4	July	8	Delete
Weights	02:45	5	December	21	Delete
Meditation	00:30	3	January	2	Delete
Cardio	00:30	2	October	15	Delete
Cardio	02:45	3	Febuary	12	Delete
Flexibility	02:45	2	July	8	Delete

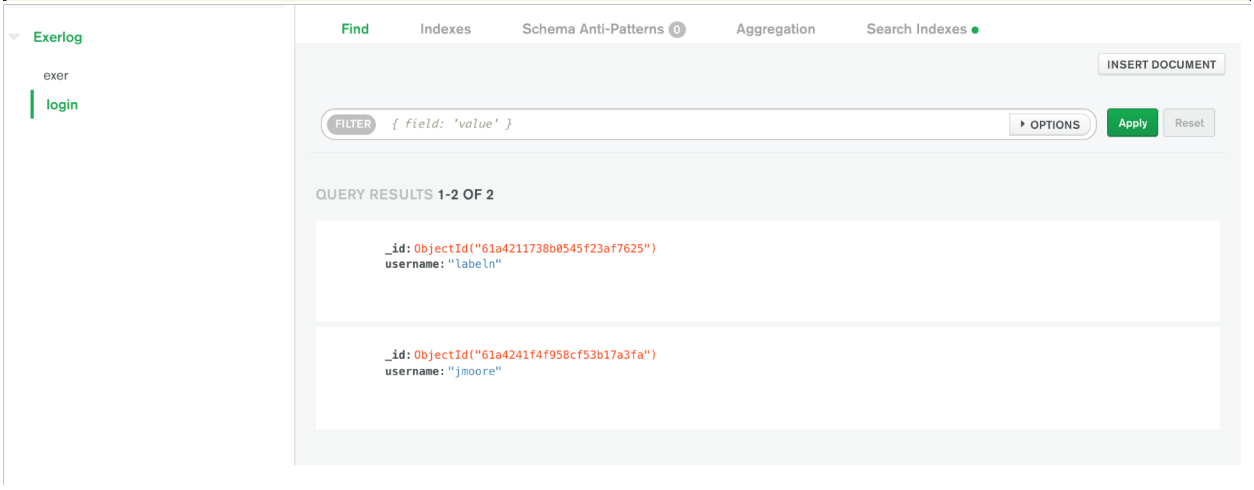
Log is also deleted in MongoDB database

Back End Functional Requirement 3

Must store unique data under each user login for retrieval upon login.

On Exerlog each user has unique logs that are specifically associated with their username. This is achieved by storing each username entered MongoDB and appending that username to each log the user makes. It then uses their username during get requests to retrieve only their logs. Usernames are only saved into the database if they have not been used before (each username only appears once in the database.) The user enters their username on the login screen before being redirected to the main Exerlog page.

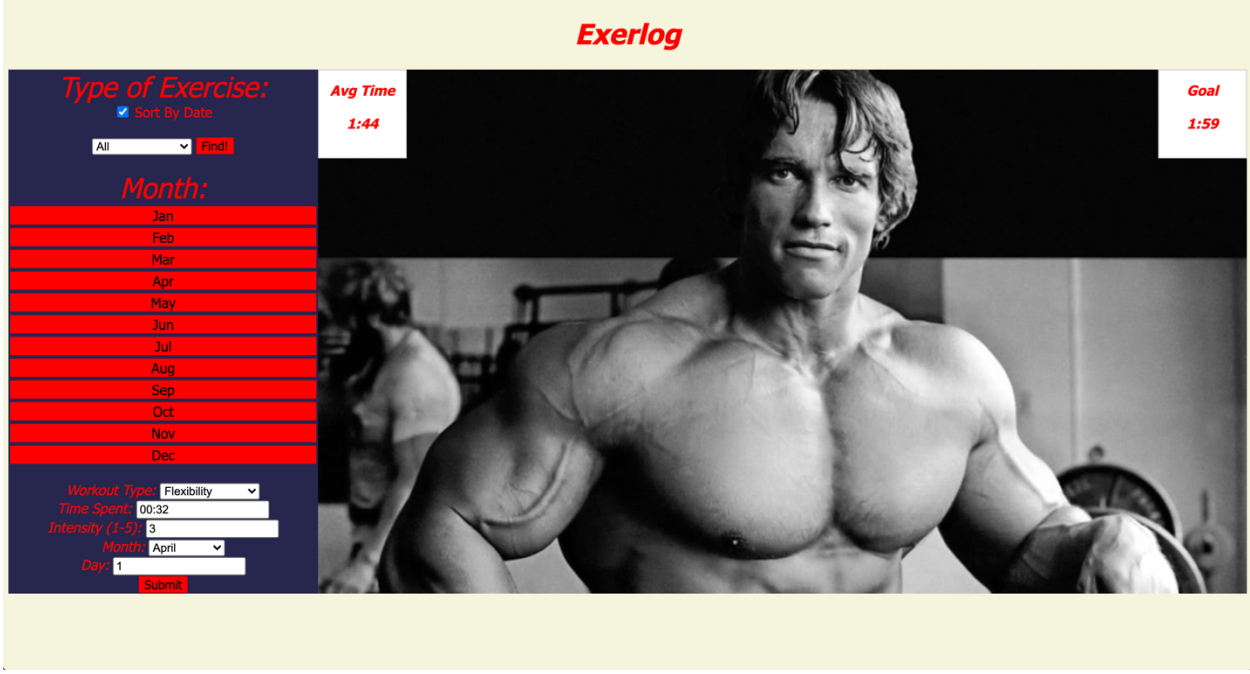




Back End Functional Requirement 4

Must compute average time spent exercising for display.

Exerlog displays the average amount of time you spent doing physical activities by averaging the amount of time you spent on each log. This is then displayed on the main Exerlog page for the user to see to the right of the form and filters of the main page. The app computes this average using embedded JavaScript and data taken from MongoDB.



Back End Functional Requirement 5

Must Set Goals for the Users Next Week

Exerlog displays ta goal for the user to achieve their next week of exercising based off the average from the previous week. This motivates the user to see that goal number steadily climb from week to week. This feature is also achieved using embedded java script and adds on to the current weeks' time based off past improvements the user has shown. (Screenshot shown above.) The average and goal are computed through embedded JavaScript that works with the data provided by the back end database after running a special query for the main page display through NodeJS.

```
</div>
<% var i = 0 %>
<% var g = 0 %>
<% exer.forEach( function (x) { %>
<% var time = x.time %>
<% var array = time.split(":")%>
<% var seconds = (parseInt(array[0], 10)* 60) + parseInt(array[1], 10)%>
<% i += seconds%>
<% g++%>
<%})%>
<% var avg = i / g %>
<% var hours = avg / 60 %>
<% hours = Math.floor(hours) %>
<% var minutes = avg - (hours * 60) %>
<% minutes = Math.floor(minutes) %>
<% var holder = ""   %>
<% if(minutes < 10){ %>
<%    holder = "0"   %>
<% }else holder = "" %>
<div id="log-canvas">
    <div id="avg" class="avgLeft">
        <h2 class="metric">Avg Time <br> <br> <%= hours %>:<%= holder %><%= minutes %></h2>
    </div>
    <% if(minutes + 15 >= 60){ %>
    <%      hours++           %>
    <%      var interval = 15  %>
    <%      while(minutes != 60){ %>
    <%          minutes++      %>
    <%          interval--     %>
    <%      }                  %>
    <%      minutes = interval %>
    <% } else minutes = minutes + 15 %>
    <% holder = ""   %>
    <% if(minutes < 10){ %>
    <%    holder = "0"   %>
    <% }else holder = "" %>
    <div id="avg" class="avgRight">
        <h2 class="metric">Goal <br> <br> <%= hours %>:<%= holder %><%= minutes %></h2>
```