Department of Computer Science and Engineering
University of Notre Dame

CSE 40746 - Advanced Database Projects
Spring 2023

Final Project

Group 4 - Report

Benjamin Egloff, Thomas Deiser, Jason Brammer

Table Of Contents:

Abstract:

Gainzmaster is a workout tracking and research application that users can utilize for their own personal fitness goals. Aside from having the ability to suggest workouts based on their popularity and number of searches on the internet, Gainzmaster serves as an educational tool, using pictures and detailed instructions on how to execute specific exercises. As an exercise tool, Gainzmaster offers four main services for fitness enthusiasts. First, the Muscle Catalog displays forty-six different specified muscles within a general muscle category. Users can also select a specific entry from the muscle catalog to learn more about the anatomy of a specific muscle. Second, the exercise catalog contains over 2,600 exercises that contain information about the specific muscle the exercise targets and what equipment is necessary to perform said exercise among other information. Third, the workout creator leverages the exercises in the exercise catalog, giving the user an option to create a workout based on a general category, or a workout of their own choice. Fourth, the workout log displays all previous workouts a user created in the workout creator for reference purposes. To tie all of these features together, we developed a web application based on a Vue.js front end and a python REST API based backend. As a result, Gainzmaster offers a quick single page application that is comfortable and easy to use. Leveraging these technologies granted our group a great number of possibilities in terms of database querying, and page design with relative ease. In all, as a fitness application, Gainzmaster condenses a large amount of exercise and muscle data into a format that is easy to digest and query for workout creation purposes.

Project Details:

The development of Gainzmaster included several hiccups and decisions that affected the overall outcome of the project and its success. Most of the project aspects described below are important notes that describe the internals of the application and other insights into its development.

Database Creation & Data Procurement:

Before we started any sort of web development work for our project, we had to make a decision on how to first procure our data, and appropriately store it. To secure our data we referenced the website https://exrx.net/, using the extensive exercise libraries and muscle information to serve as the starting point for securing our data. The exrx website contained a plethora of the exact information that we were looking for, but with limited querying ability. So, we were left with few choices on how to secure said information, but we ultimately settled on web scraping to secure the necessary data. To scrape the data from the several pages located on the source website, we utilized regex searching based on html tags to extract titles, headings, and other appropriate paragraphs to store in our database. Once we had written a python script to extract data from each of the thousands of exercise pages and pack it into a csv file, we focused on securing an image link for each exercise and muscle. To solve this goal, we simply wrote a short python script that queries google images and puts the first image link into the appropriate spot within a python script.

After completing both of these goals we had successfully procured a csv file for general muscle category, detailed muscles, and exercises. To create the tables referenced in our final design the only remaining task was to create a .ctl file and use sqlldr to import our csv files into our sqlplus database.

Automated Workout Creation:

One of the most complex parts of this project was the automated workout creation. Given little input, our application is able to give users a full workout pulled from our workouts database. Overall, we have three different instances of automated workout creation on our site. The first version is the most controlled, where the user inputs what type of workout they would like to have, and we construct a workout that fits the criteria. This version is present in the workout creator page, and happens when the user presses the “Push”, “Pull”, “Upper”, or “Legs” button. In these creations randomization is present only in the sql queries, as rows that match the search are order by the dbms_random.value algorithm.

The second version is slightly less controlled, where our site also assumed the type of workout the user would receive. This version is present on the home page, in the bottom left corner of the page. This type of automation seemed fit for this context, as it takes no input from the user, yet still returns a workout that would make sense. Although this version, similar to the first, randomizes exercise that match a criteria in a query via dbms_random.value, it is different

in the sense the the criteria itself is random as well. This criteria is derived in JavaScript, as it pulls a random value from the lift of value workout types and searches based on that.

The final automated workout creation is by far the least controlled, the most random. What makes this version so different form the last two is the fact that there is no criteria. This means that while the last two versions of automated workout creation would return exercises that fit the same criteria, this version pulls completely random exercises. This is present in the "surprise me" feature of the workout creator, and probably has the least practicality of the three versions. Because the exercises are completely random, users are much less likely to receive an optimal workout. This randomization is also based on ordering by dbms_random.value.

Technology Stack:

To decide the internal workings of our web application, we put significant thought into different technologies to use for our frontend and connection to backend. For our frontend technology, as mentioned earlier, we decided to use Vue.js as two out of our three group members had extensive experience with it. In addition, we figured that using a technology that most of the group was familiar with would help us complete our deliverable within the set timeline. As for our connection between front and backend, our decision was a lot less simple. None of our group had experience with connecting an oracle database to a web front end, other than through the php demo shown in class. However, we decided to do some research in order to stick with Vue.js for our front end rather than accepting some of the limitations that come with using php and barebones html. In the end, we found that the best decision was to host a python script on our apache server that when invoked executed sql queries based on the url query string.

Team Organization:

At the beginning of our development process we detailed specific roles and responsibilities that each team member would be responsible for given their strengths and weaknesses. Ben opted to work on scraping the data from the source website, linking the backend and frontend, and other general backend duties. To work to Tommy's strengths with Vue.js, he assumed part of the frontend duties, particularly those pertaining to the home page, the workout creator, and the workout log. Similarly, Jason held primarily front-end responsibilities, developing the exercise catalog, muscle catalog, and their respective detail pages for individual muscles and exercises. With these general roles in mind, we proceeded with the project sticking mainly to our own assignments, but also working collaboratively whenever we ran into roadblocks.

Coding Issues, Debugging & Testing Strategy:

As with any coding project, our team experienced several bugs in both our front end code, scraping code, and backend REST API.

For our REST API, many of our bugs arose from general unfamiliarity with the apache server and running a script that can be invoked on it. Using the python module cx_Oracle, we

figured that creating a working script would be relatively easy, however, the main issues that we encountered were quite time consuming. First, we found that the httpd.conf file needed to be modified to accommodate python scripts, a small detail that caused a great deal of headache. Second, we found that running a script from an apache server references the globally installed python modules on the machine. To solve this bug, we had to find a way to install pip3 (globally) and install the appropriate module globally, which doesn't seem like a lot of work, however our process included a lot of trial and error. Finally, we found that upon script execution that the environmental variables of our group vm were not loaded into the running script instance. After testing around with modifying environment variables within a python script and using exec to re-run the script in the appropriate context we achieved success. To test our python script for success, we simply ran a node.js script using axios get and post requests to verify responses were what we expected.

Many of the bugs associated with our scraping code came from incorrect regular expressions resulting in improper data selection. In fact, our first csv files were horribly inconsistent with what we had expected, requiring a lot of regex testing and debugging print statements. Another bug we found in our scraping code had to do more with the representation of html data on the internet. In several spots in our data we found fragments like "#34;" and "#39;" which represent the unicode for " and ' respectively. Reformatting our csv file to address this wasn't really an issue, but finding all such instances of oddly formatted strings was an issue, but we are nearly certain that there are no more fragmentation issues within the database.

The main source of the bugs in our project came from our front-end code. However, since our backend connection was available from the instant we started our front-end development, we were able to focus all of our debugging and testing efforts on our web application. The main source of bugs that we struggled with was formatting data from our database in a clean tabular format. However, the streamlined nature of the Vue-CLI made testing and resolving our numerous bugs quite easy, as simply saving a file would update the html document. Another source of error in our project was simply aligning the html elements on each page properly. Our struggles with css continued throughout the whole project, but we have gotten to a point where our application (for the most part) is styled appropriately, aside from some issues like mobile screen formatting.

Interface Selection & Implementation:

With the main goals of our project being to create pages that display exercise and muscle information, our main goal for each interface was to maximize digestibility and visual appeal. Accordingly, each of our pages has a different design, putting different pieces of data into focus according to the page's purpose.

The muscle catalog page design (see Appendix A) is quite simple, placing emphasis on the name and category search fields to show the simplicity of querying the table located directly below the search box. We believe this design works quite well for the muscle catalog as it pulls

four simple pieces of information from each muscle, ensuring users don't get lost in technical information.

The exercise catalog page design (see Appendix B) borrows the table and search engine components from the muscle catalog page, using an increased degree of complexity however. The search box gives several more options to the user, which is necessary to filter over 2,500 exercises. In addition, the page features a preferability scale in bright colors to attract the attention of the user and emphasize that some exercises are less desirable than others.

The workout creator page (see Appendix C) also builds on the exercise catalog table, using the table component to effectively display large quantities of information, while also providing the ability to search said information. Also, the workout creator page gives users the ability to add and remove exercises to a workout queue, stating the preferred number of sets and reps. We chose to display the exercises in the workout queue using a table clearly showing the exercise, sets, reps only, as more information would make the table too cluttered. The current workout table on the page is intentionally simple to contrast the detail of the exercise table below it.

Finally, the workout log page (see Appendix D) displays workouts that a user has previously created. Our design of the workout log page was intentionally very simple to promote a "glance and go" feel, as what would typically occur in the gym. Our user base wouldn't want to look through 10 lines of content to find what exercise they want to do, and sometimes an exercise name isn't descriptive enough for the movement. What we came up with to solve both of these problems is a three column table displaying the exercise name (with a hyperlink to the exercise detail page), an image of the exercise, and the number of sets and reps for the given exercise. We believe this concise view is perfect for use in the gym, as it isn't too distracting or detailed.

Fault Tolerance:

In the frontend, there are several areas where a user can input certain fields to create their account and naming their workout upon creation. The method used to protect against SQL injection attacks is forcibly changing user input to a string where single quotes and semicolons are redundant and cannot modify the database in any way. These are the only places where a SQL injection attack could occur because there are no other places of custom input. The rest of the site only allows users to query through existing tables which would not affect the database in any way.

Using a framework like Vue, a nice benefit is that a lot of the errors that come with html and javascript are extracted away from us. Queries that return an empty set are handled through a series of if statements so that an empty query would not cause any malfunctions within the site. Also, when a user fills out a form such as a new workout, we have default values set in place in case the user forgets to enter data into a specific field. This ensures that a workout will still be generated for the user in the case that they do not completely fill out the form.

Conclusion:

Gainzmaster is an online platform that gives users the ability to track their workouts and goals. The site provides a number of features such as a muscle and exercise catalog, workout creator, and a workout log. The site was created using the Vue.js framework for the frontend and a python REST API based backend. Vue was crucial in terms of providing an easy-to-use template framework and built-in functions that make navigating the site relatively smooth. The database was created by scraping the data on the https://exrx.net/ website and populating csv files via a python script. Then, the data is inserted into the database through a control file which targets the csv's and a bash script that calls the sqlldr function. This allows our database to be relatively portable aside from the user, session, and workout log tables which would need to be exported to another csv.

The team worked very well together in terms of dividing up the work evenly and coming together whenever issues arose. We ran into a few issues when setting up the apache server, but were able to move past them after installing a few packages on the group virtual machine. The script to insert the data into the database also caused a couple setbacks due to certain data not being in the format we were expecting it to be, however, we were able to fix all of the fragmentation bugs that we spotted. Other issues occurred from the css and styling of the pages, but through perseverance and dedication we were able to come up with a design that was satisfiable to all of us. The website is secure and protected from any injection attacks that could modify our database. All in all, the group learned a great deal about web development and connecting to an oracle database and we are all very satisfied with the way our site turned out.

Appendix A:

Muscle Search

NAME

CATEGORY All

Muscle Name	Muscle Category	Other Names	Related Muscles
Sternocleidomastoid	Neck 	Sterno-mastoid,Neck (Front / Sides),Neck Flexors	Splenius
Splenius	Neck 	Neck (Rear),Neck Extensors	Trapezius,Upper Fibers,Sternocleidomastoid,Erector Spinae

Muscle Catalog Page Design

Appendix B:

Preferability Scale:

Each exercise is color coded according to how gym experts would rank them!

1 2 3 4 5 6 7 8 9 10

Exercise Search

NAME MUSCLE GROUP All DETAILED MUSCLE All SORT BY PREFERABILITY

EQUIPMENT/METHOD All UTILITY All MECHANICS All MUSCLE FORCE All

Reset Fields Search

Name	Muscle Group	Detailed Muscle	Equipment/ Method	Utility	Mechanics	Muscle Force	Preferability
Wall Straight Leg Calf Stretch	Calves 	General Calves	Stretch	Other	Other	Other	5
Lever Seated Iron Cross Fly	Chest 	General Chest	Lever (selectorized)	Auxiliary	Isolated	Push	3

Exercise Catalog Page Design

Appendix C:

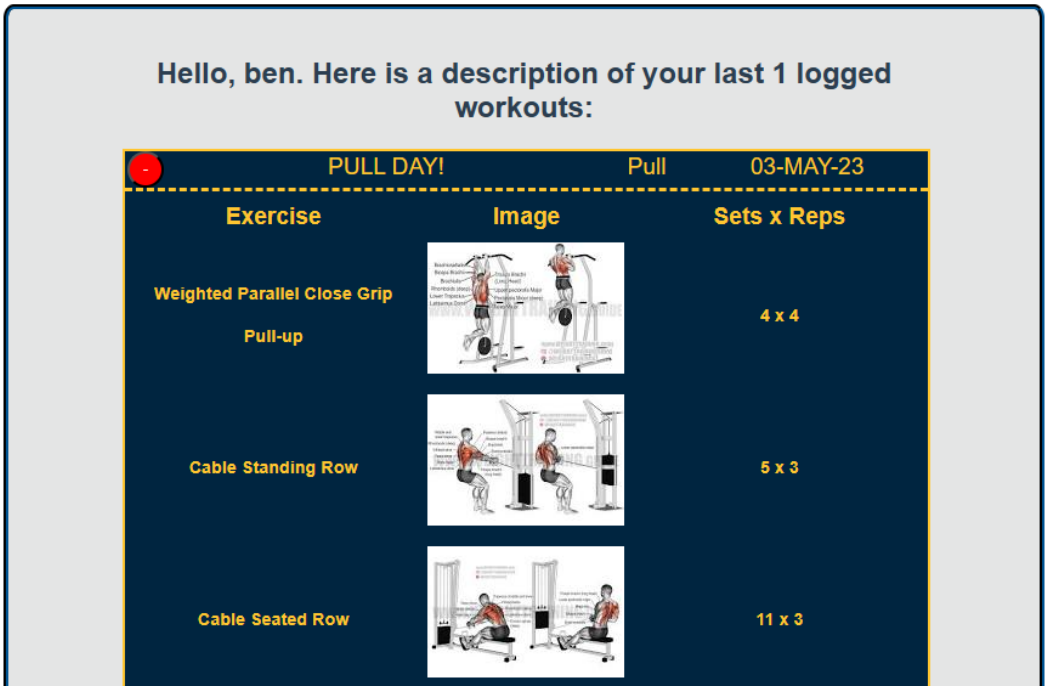
Current Workout:

Remove Exercise	Exercise	Sets	Reps
-	Cable Standing Fly	3	9
-	One Arm Push-up	3	13
-	Push-up (on knees)	3	13
-	Cable Bar Standing Incline Chest Press	5	3
-	Cable Bar Shoulder Press	4	8
-	Cable Bar Standing Chest Press	3	5
-	Weighted Push-up (on handles)	3	5

Clear workout Complete workout

Workout Creator Design

Appendix D:



Workout Log Design