IronDiggers Final Report

CSCI 370 Field Session

Summer 2018

Riley Miller

Madison Rogers

Emily Garnier

Satvik Saini

Advisor: Dr. Fisher

I. Introduction

The IronDiggers field session project was to create a web application for the Colorado School of Mines strength program. This was part of a field session project in summer of 2017, and some of the features developed during that time were intended to be carried forward to this year's project. The main goal of this application was paperless data tracking of weight lifting information for athletes at Mines. This web application was also to allow for paperless communication of workouts and the ability to track athlete progress over time. The way the system works is that the strength coach will enter specific workout information for groups of athletes, then that athlete will complete the workout and record the weights lifted. Eventually, the athlete and coaches can look at improvements made. This application will also ease the customization and consistency of the workouts because an athlete can easily access recent lifts so they know how much weight they should be lifting on a specific day. This will be an improvement over the current method of paper sheets to communicate workouts. The paper sheets make it difficult to see athlete progress over time. Once deployed, the online system will lessen the manual work of searching for sheets by the strength coach, as well as making workout history more accessible to athletes.

II. Requirements

The IronDiggers field session team focused on two main functional requirements which include: application rewrite and website improvements. Additional features being a schedule feature and mobile alert system, that we did not get to this summer.

Application rewrite

The code from last year's team did not meet the requirements of the client. The backend of the code application needed to be structured differently in order to meet the requirement of paperless data tracking set by the client. The two options being, either refactor the existing code base, which would be rewriting practically the whole backend of the application and any dependent parts of the front end, or start from scratch in a technology stack more familiar to the team. It was decided to start from scratch and write the application in the MEAN stack for better scalability and future uses.

Website improvements

One of the main goals for the website was to improve the user interface to be more intuitive. The project team replicated the paper sheets that used to be used to track data, but in a digital fashion. The athletes at Mines are accustomed to these weight lifting sheets, so having the web application be very similar to those will make the transition easier for the student athletes. The website would automatically generate the percentages of lifts to facilitate the weight room experience. This means that when the workout says to squat 60% of max eight times, each athlete will have their 60% of max automatically calculated for them, so they know

what they will be lifting before they go in the weight room. This not only customizes the workouts, but helps student athletes stay consistent with their weight training.

Another piece of functionality is the ability for the strength coaches to track individual or a team's improvement over time. The project team will set up an interface where the coaches can select an athlete and look at all of their lifts and progress over time. In addition, creating a way to look at a specific team and look at performance history.

Time permitting: Online Weight Room Schedule

The client also asked for a way to display the current weight room schedule. Having this schedule available online is a great improvement over the current method of just having a paper schedule posted outside the weight room. With the scheduling information online, a student athlete can easily look up hours of open lifting without having to make the trip to the weight room with the potential of it being in use.

Time permitting: Mobile Alert System

The project team would add a mobile alert system that can be used to send updates to a team about changes in timing for lifting or reminders to complete tasks associated with the weight room. This makes it easier for the strength coach to communicate about changed practice times or reminders. This is ideally done by having a section where the coach can enter a message they wish to send to a specific team or individual and then the message as a text to the athletes phone. Real time updates would useful in the case that the strength session needs to be moved from 3:30 to 4 for example.

Non-functional requirements

High Level Requirements:

-Well documented and prepared for the transition to the application maintenance phase.

-improvements based off of work completed by the Summer 2017 team.

Hosting Requirements:

-The improvements held in a GitHub repository and will be deployed to the irondiggers.mines.edu instance as the application gains functionality.

-Application configured with Shibboleth authentication through Multipass.

Development Requirements:

-Application rewritten in the MEAN stack to meet the clients requirements.

-Language standards and industry best practices followed while integrating new technologies.

III. System Architecture

The main technical design issue the project team faced was the decision to restart the web application using a different backend. The project team intended to build upon a project started during the 2017 field session. After talking to the client, the project team looked at how the backend would need to be restructured. The backend required significant redesign, and if the project team were to restructure the database, much of the front end would no longer work. The initial project framework (LAMP stack) worked from a short term perspective, but didn't align with the client's long term goals. The framework wasn't scalable in the way it was built, making it useful for a very specific use case. Also, most of the programming was PHP making it interconnected with the front end, which the team was planning to completely overhaul anyway. Thus, the project team talked with Professor Painter - Wakefield about the options of major modifications versus restarting and the decision was made to restart.

The old application used Laravel, Apache, and MySQL. The new application uses Mongo, Express, Angular, and Node. Figure 1 shows the attributes the project team intends to store in the new database. The project team set up a meeting with CPW and Dr. Romig to discuss the potential logistic and privacy issues of restructuring the original project according the the diagram below. The biggest issue identified by Dr. Romig was to make sure the ciphers for Node.js agree with the ciphers the school servers use. The consensus is that the project team should be able to develop the web app with the tools listed. Dr. Romig also gave the project team a couple of tasks. One was to renew the certificate used to host the website on the school servers. The other task is to run the code through vulnerability testing before it goes live.

Figure 1: Database Schemas - Each schema will be turned into a database collection in the MongoDB.

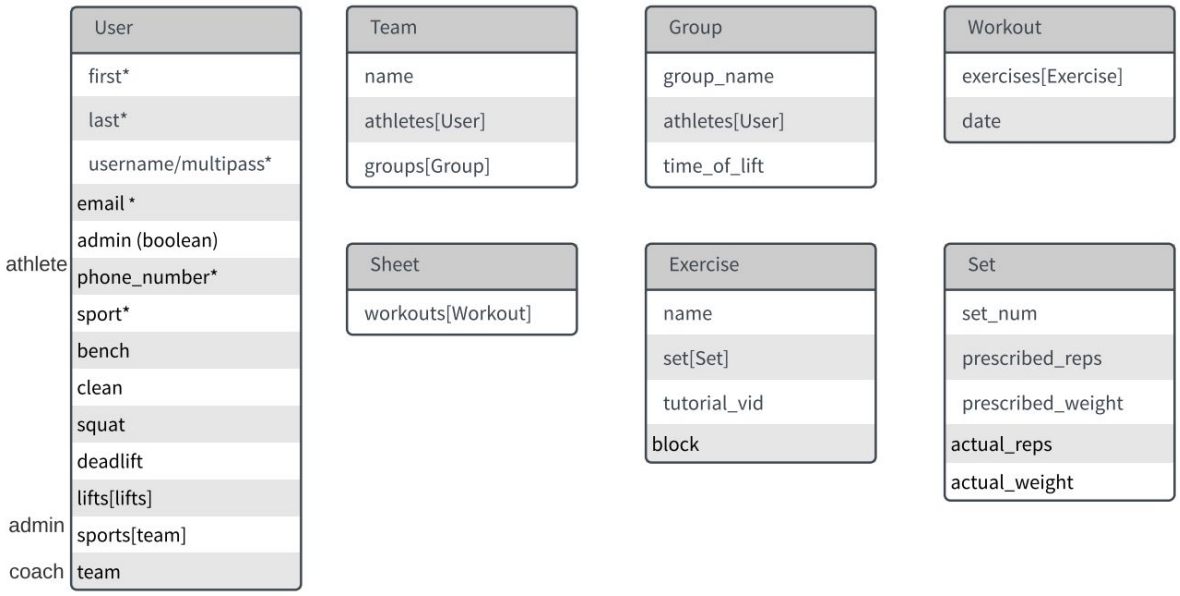


Figure 2: Data Flow - This graphic shows how the data will flow from the MongoDB, to the JavaScript controller files, to the API routes. The API routes are the endpoints that allow the front-end of the application to communicate with the database.

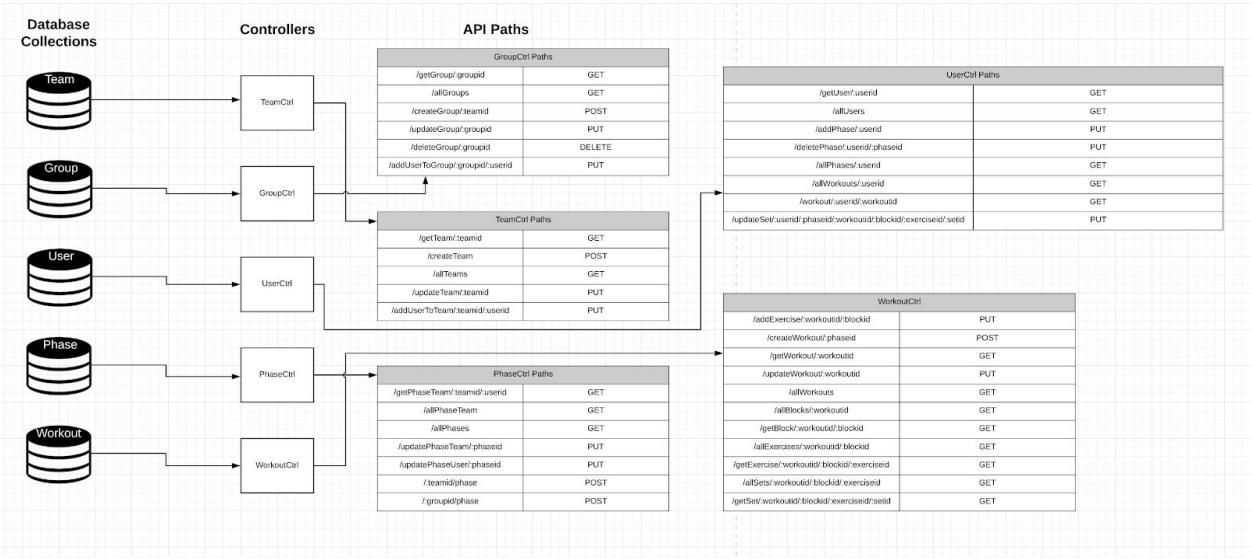
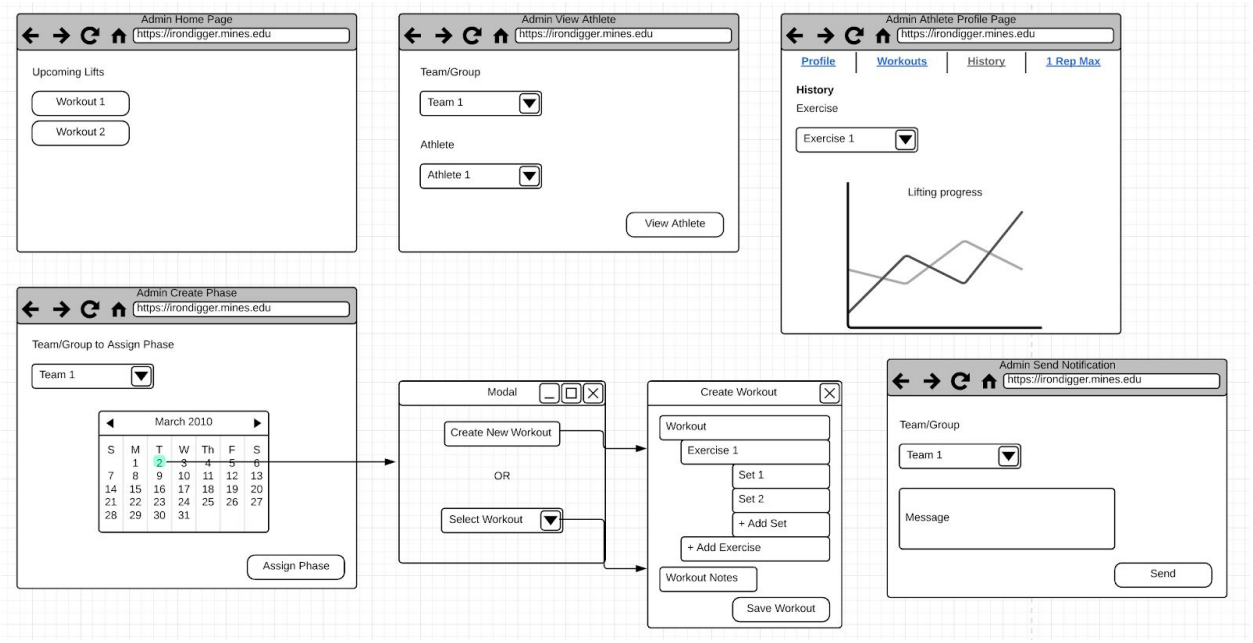
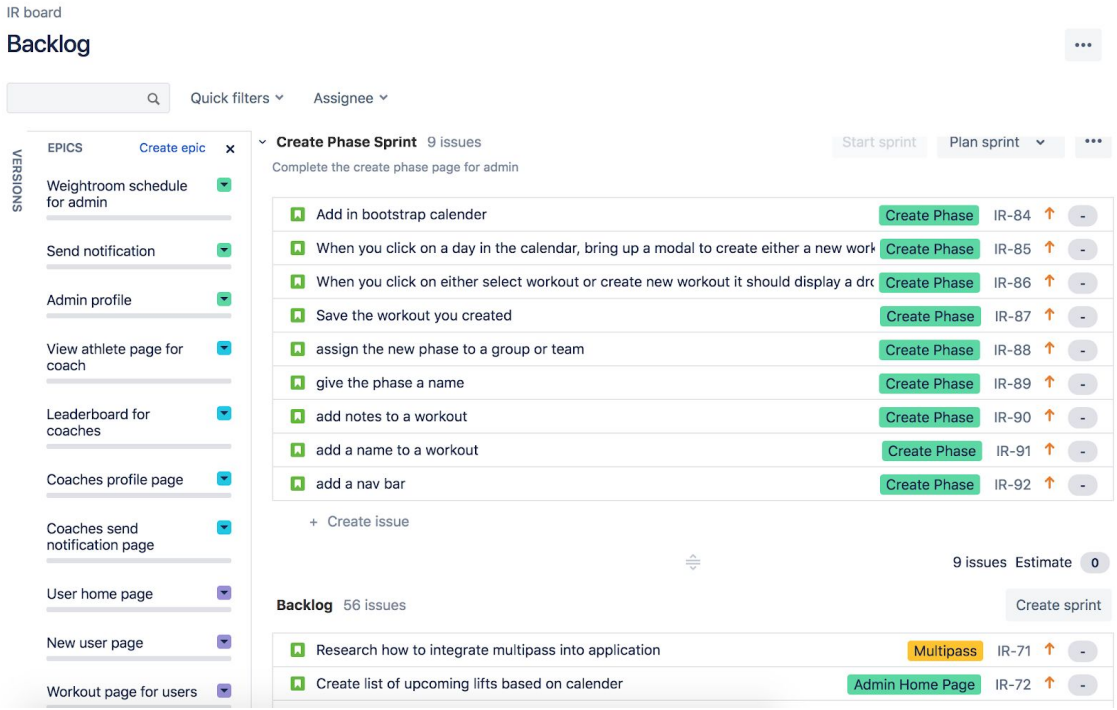


Figure 3: Admin views. These are mock ups of what we wanted the front-end of the application to look like. The User views and Coach views will be a simplified version of the Admin views



The project team is using Jira to map out the agile development process. Figure 4 shows some of the epics and tasks the team has outlined. The epics are based off of the views shown in figure 3 and the issues are the expected items to complete on each view.

Figure 4: Agile Development Process



IV. Technical Design

Create and Edit Workouts

One of the most important parts of this web application is the ability for an admin to dynamically create a workout for athletes. This posed a difficult front end programming challenge. The team decided to incorporate the FullCalendar javascript API with our code. This provided an interface for the admin to click on a date on the calendar and have a modal displayed for them to create a workout. The FullCalendar API took care of the scheduling aspect of the application but still left us to develop the interface to create workouts.

The goal of the Create and Edit Workout interface was for it to be dynamic and simple to use for the admin. Once a workout is made it will be displayed on the calendar and can be clicked on to be edited. The edit workout modal is the same as the create workout modal (see figure 6) but it is populated from the workout object connected to the event.

Figure 5: This figure is the admin's interface to plan phases (groups of workouts) for a team or group. The admin can select a team or group then click on a day on the calendar to bring up the modal to create a workout for that day. The create workout modal can be seen in Figure 6.

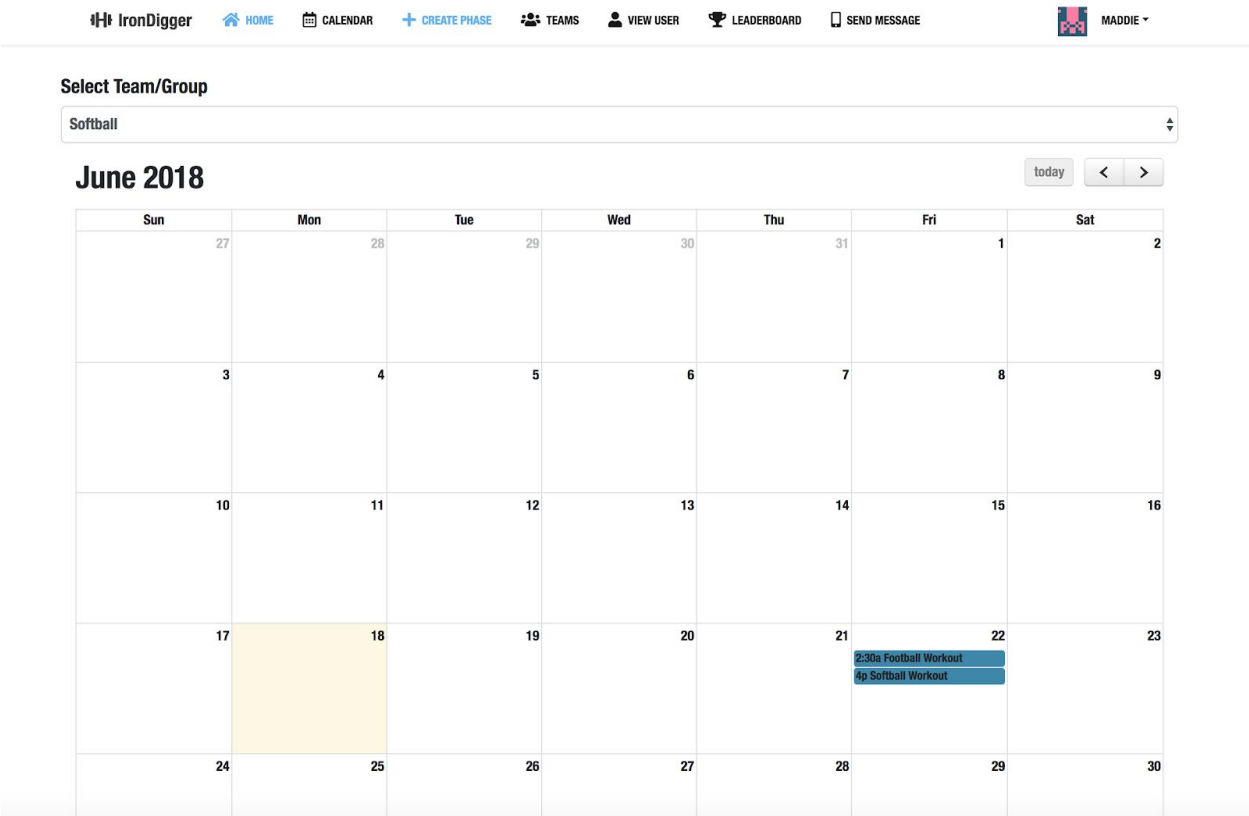
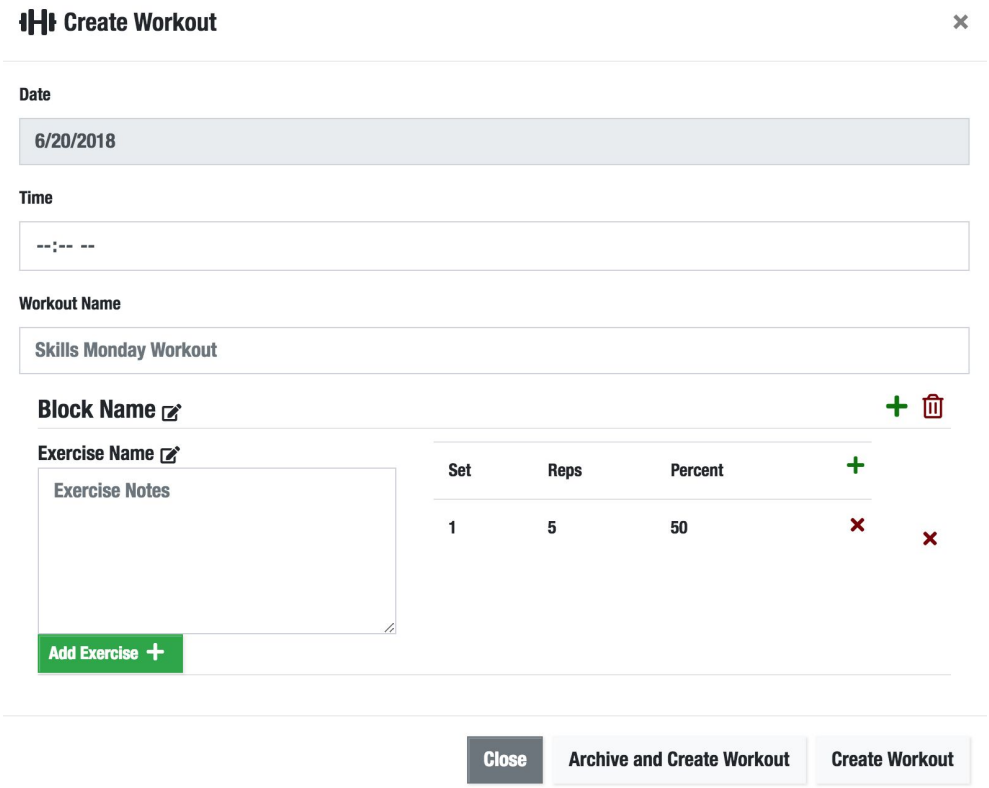


Figure 6: This figure is of the admin modal to create workouts. This is a dynamic form-like structure that allows the admin to add as many blocks, exercises, and sets as they desire. They can also set a time and name for the workout.



The project team ran into many problems while developing this part of the application. The dynamic nature of the UI was much more challenging than anticipated. In order to make the application dynamic the team had to create new DOM elements and generate unique IDs for the new elements. This created a big challenge when debugging the code. Looking back on our design decisions, it would have been smarter to use a front-end framework that keeps track of state, such as React, because our code became complicated and hard to follow because all of the elements and their IDs are interconnected.

Athlete Progress Graphics

A main requirement from our client was the ability for an athlete to be able to view their lifting progress as well as the adim or coach. The client said the athletes that use this application will be using their cellular device to view the web page, so it must be mobily responsive. The decision was made to utilize the d3.js graphics library due to its ability to create responsive graphics and visualizations.

The athletes are able to view their lifting history for any exercise. A graph and table of all their data for that exercise was created using d3.js. This allowed us to link the data to the different

elements on the graph, making the graph very interactive and dynamic for the user. The x-axis of the graph has the date the exercise was done and the y-axis has the weight. As seen in the figure, if multiple sets are done of the same exercise on the same day, they are shown on the same day. This was the athlete can see their progress for the day as well as over multiple days. Figure 7 shows a screenshot of an athletes lifting history for hang cleans.

It was a good design decision to use d3.js because it took care of all of the responsive design for the SVG elements. All the team had to do was set the width and height of the element to correspond to the device width and height.

Figure 7: This is the graphic generated using d3.js and the athletes individual lifting data. This is the view on a laptop. Figure 8 shows the view from a mobile device.

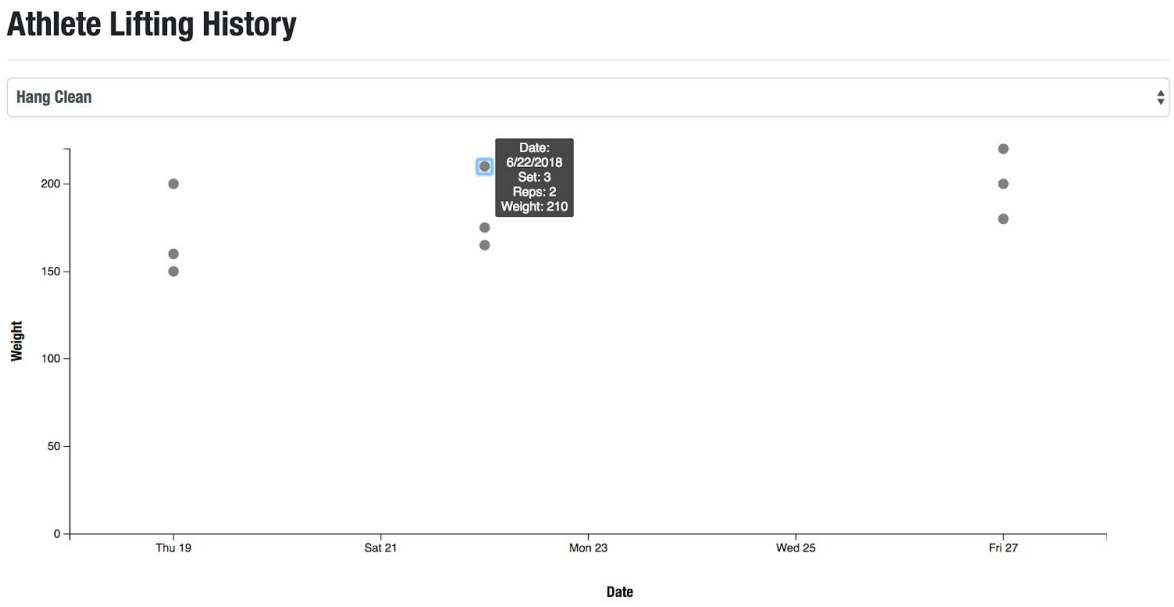
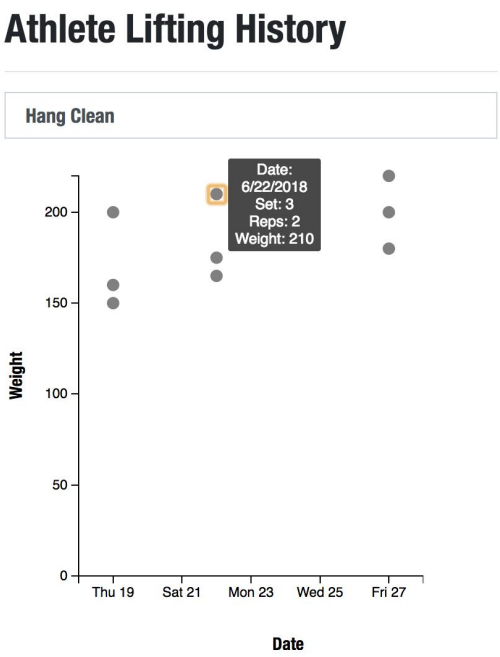


Figure 8: This is the same data as Figure 7 but viewed on a mobile device.



Viewing Workouts

Similar to the athlete lifting history pages, the view workouts page for athletes needed to be mobile friendly. Using bootstrap, it was relatively simple to create a structure that looks nice and is easy to use on both mobile devices and laptops. Figure 9 and 10 show the view workout page as a website viewed in a web browser and a mobile site.

Figure 9: This is the View Workouts page from a web browser on a laptop.

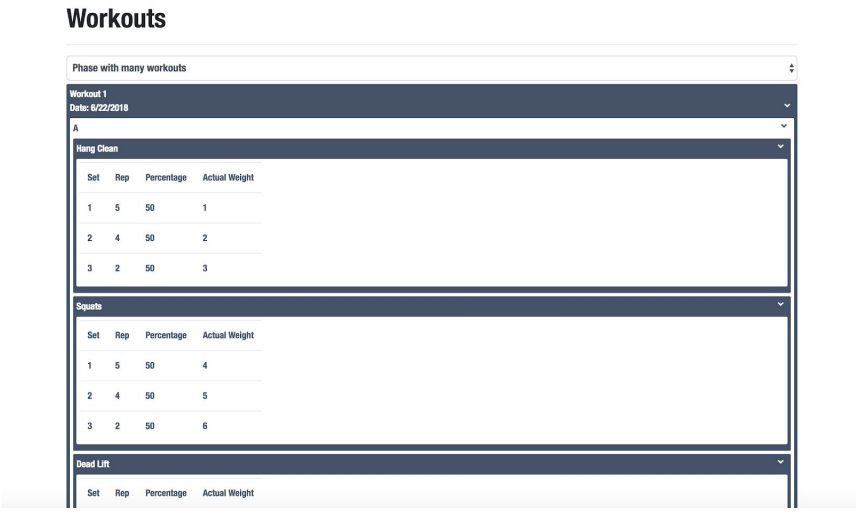
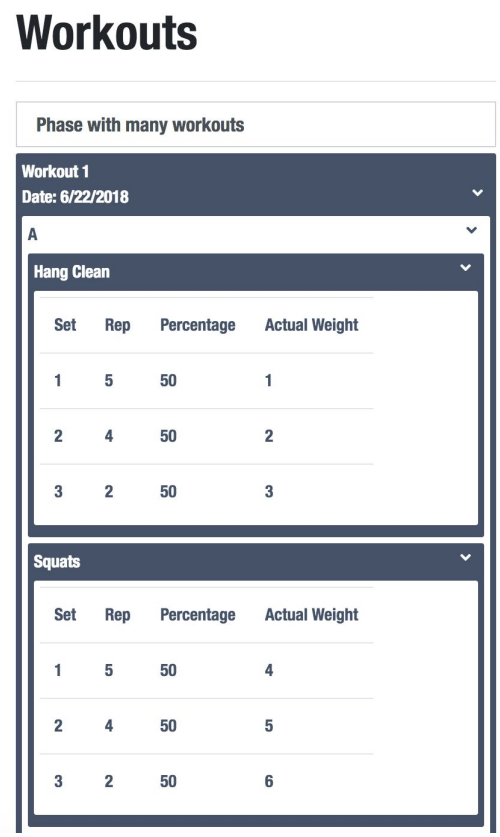


Figure 10: This is the View Workouts page viewed on a mobile device.



Deployment

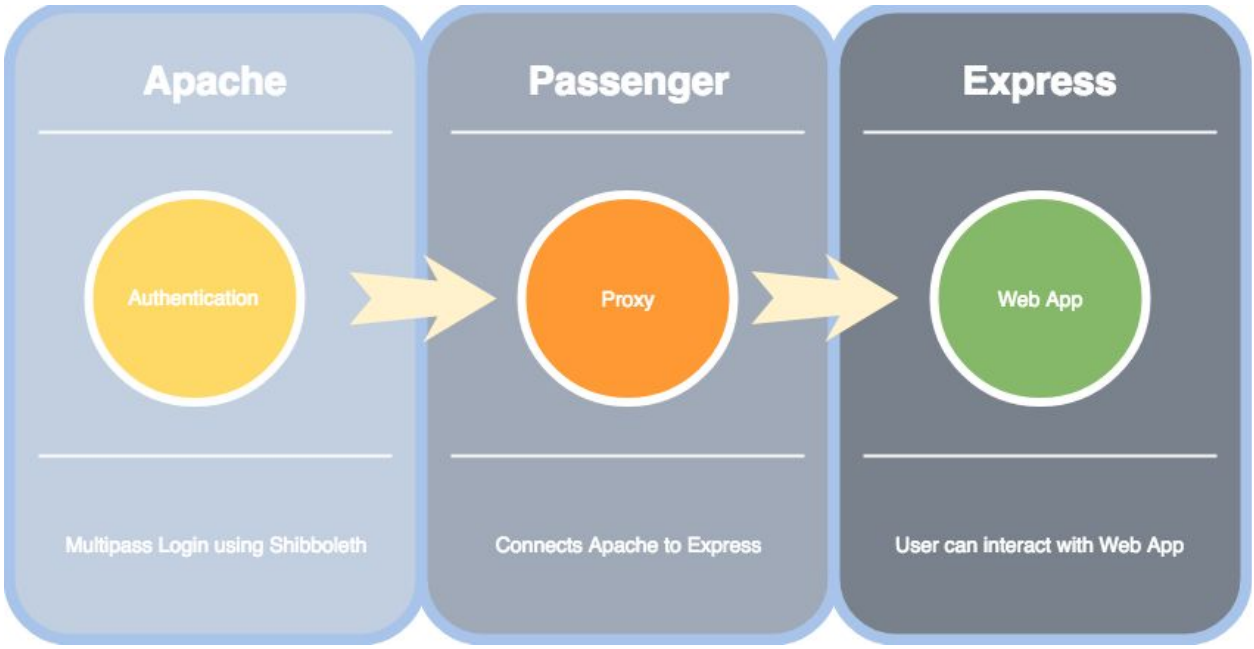
Another key piece of the web application was deployment. This is a key step to all software development, because otherwise no one can use the application created. Deployment involved three key steps: obtaining a certificate, setting up authentication through Mines Multipass, and hosting the application on the server.

The first step was obtaining a certificate. First, the project team had to collect metadata about the server irondigger.mines.edu. The metadata acts to identify the service provider to the identity provider. Mines uses Shibboleth as the identity provider to authenticate users by logging on with Multipass. The metadata collected contained a public key and a private key. Both keys were encrypted using RSA 2048. The private key is stored on the server and only root has access to read the file. The public key is sent to the service provider. The school has to send the metadata and public key to a web authentication service to get authenticated. Once this happened, the project team received a certificate for the server.

At this point the Shibboleth authentication system could be set up. This is an important step since the web app will be for Mines students only and their data should be secure. Shibboleth was configured with Apache so when someone visits irondigger.mines.edu behind the Mines firewall they are sent to a Multipass login page.

The project team used Apache as a proxy to handle the authentication piece, but then used Passenger to connect through to the Express instance that hosts the web app. The web app is stored locally on the server as a Github repository. This is convenient because any changes can easily be put on the deployment server. Figure 9 shows how the authentication connects through to the web app on the server.

Figure 9: Server set up to Authenticate a Mines user.



V. Decisions

Language Choice

The project team chose to use the MEAN stack to create the web application. MEAN stands for Mongo, Express, Angular, and Node. Most of the project is written in Javascript. Pug is the markup language used to display information of the front end.

Data Design

The information is stored in a Mongo database. This database is a good choice because having a relational database looked very messy with many foreign keys and cross reference table. The information is more easily accessed and understood in the Mongo schema.

Reuse/Library Choice

The project team is using Bootstrap to style much of the front end. This makes sense because Bootstrap has many useful tools that are already well styled. This way the project team avoids “reimplementing the wheel” as the team learned in the Tips talk.

The project utilized the github repository sahat/hackathon-starter which is a boilerplate for Node.js web applications. The term boilerplate refers to applications or programs that are intended to be used over and over again. This starter code is very generic, but acts as a good starting point for the framework of the website.

Implementation Decisions

The project team is using Express as the web server, however to implement Multipass login, the Express server has to be hidden behind an Apache authentication system. It is possible to use Shibboleth, which is used to implement the Multipass login, on Express, but the documentation at Mines is for Apache. After talking with Jack Rosenthal, who has used this authentication on a web application before, the project team decided the best way to implement Shibboleth is with Apache. Now the server will have Apache as the first point of contact with a user. The Apache server will act as a proxy and then connect an authenticated user to the web app locally hosted on the server.

Alternatives

The project team considered using React for some components of the web application. After wrestling with React and not having much success, the project team found that some of the functionality needed could be accomplished with Bootstrap. Also, the fact that none of the team members have very much exposure to react led us to use Bootstrap.

VI. Results

The main goal of this project is paperless data tracking of weight lifting information for athletes at Mines. This web application will also allow for digital communication of workouts and the ability to track athlete progress over time. This project was a continuation of last year’s field sesion project. They had considerable progress; however, the final product from last summer used a LAMP stack which didn’t align with the project’s ultimate vision. Thus, one of the functional requirements of our project was an application rewrite. A team decision was made to convert to a MEAN stack for better long term outlook & scalability. Starting from scratch, the backend was completely restructured. Along with the application rewrite, other functional requirements include various website improvements. The improvements include making an

intuitive user interface designed to replicate the paper sheets the athletes are accustomed to. Additionally, a view will be built out for coaches to track an individual's or a team's improvement over time. Our application meets all of our clients' functional requirements. However, due to time constraints the online weight room schedule & mobile alert system did not make it into the final product. These additional features can be built in future field sessions.

Things we learned

1. Installing the MEAN Stack was a difficult process for two of our group members. The steps are extensive and everything needs to be aligned perfectly. This proved to be slightly problematic on Windows & Macs.

2. After the somewhat troublesome installation of the MEAN stack, the benefits outweigh the installation issues. The flexibility of the MEAN stack allowed our group to easily develop, test, and introduce our app on the cloud. Also, the framework was intuitive and only needed proficiency in JS. Whereas, the LAMP stack (previous framework for this project) required expertise in MySQL, PHP & Javascript.

3. In order to host a website on the school server, the project team had to obtain a certificate. This certificate was encrypted using RSA 2048. The project team generated a public key and a private key. The public key was sent to CCIT and then they request a certificate from a web authentication service. Then the certificate was downloaded onto irondigger.mines.edu and the web page can be accessed at this address. This was a cool application of an encryption technique learned in class at Mines.

4. The d3.js library for creating visualizations of data was incredibly useful and powerful. Some of the many pros being this library is backwards compatible, thus easy to use with many modern browsers today. Most importantly the d3.js is a versatile library for manipulating data on the DOM. This was helpful in creating/editing workouts on the app.

5. Creating a dynamic form for the admin to create and edit workouts was much more difficult than expected. The hardest part being taking a workout object and creating the corresponding dom elements with the correct ids. Thus, in hindsight the project team might have considered a different approach.