

CS 4560/4561 - Software Design and Development I/II:
Spring 2017, Chang Liu, Yunyi Feng

Personal Performance Application:
Michael Clevidence



Team on a Cob:
Andrew Leach, Peter Essman, Roberto Whitmer, Brent
Gruber, Lucas Nation

Overview

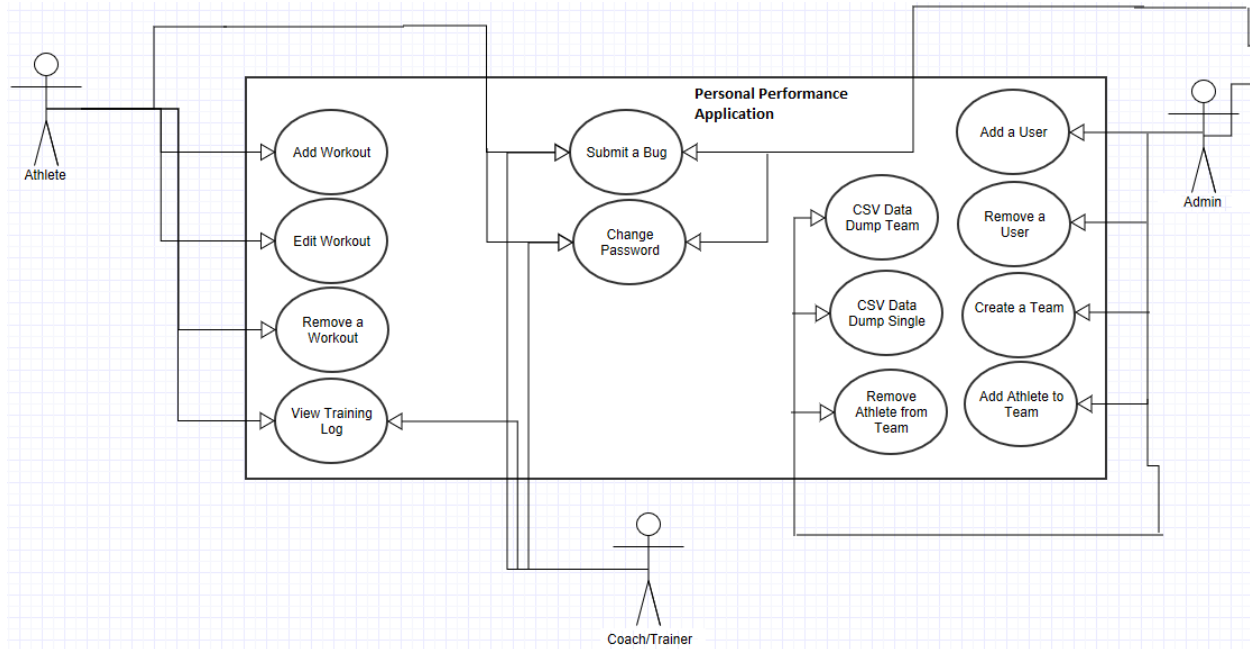
The overarching goal of this project is to create an interface that will allow athletes and coaches to easily assess their training regimen. Currently our client performs this task with a complicated excel spreadsheets that athletes must submit on a weekly basis via drobox.

This project will provide a friendlier user interface for allowing an athlete to enter their training data, and allowing a trainer to view that data. We will provide a user friendly website built on node.js and pug template files, which will store all the workout in a database. The site will allow users to enter workouts and view past workouts in chart format. The site also allows for coaches of a team to keep track of the statistics of all of their athletes, as well as for trainers to access data to perform local analysis. There are many other features that will be explained within this document that help solidify the self-managing nature of this application.

In conclusion, this application unifies all of the possible user roles and desired usage of each role into one uniform and convenient application. This documentation will help guide your understanding of how this performance program application was designed, built, and maintained.

Use Cases

Use Case Diagram



List of use cases:

- {Athlete - 1} Enters workout - The athlete will login to website and enter a workout.
- {Athlete - 2} Views training log – The athlete will login to website and view their previous workouts.
- {Athlete - 3} Edit a workout – The athlete will edit previously entered data and resend the data for one selected workout.
- {Trainer/Coach - 1} View an athlete's training log – The trainer/coach will login, can then view all of their teams, and will be able to select individual users to see data from.
- {Administrator - 1} Receive CSV individual data dump - Administrator logs on and requests a csv data dump for a single specific user.
- {Administrator - 2} Receive CSV team data dump - Administrator logs on and requests a csv data dump for a single specific team.
- {Administrator - 3} Add user - Administrator will add a new user to the application
- {Administrator - 4} Remove user- Administrator will remove a user from the application
- {Administrator - 5} Create a team – Administrator will be able to create a new collection of athletes and give the collection/team a name.
- {Administrator - 6} Add a user to a team – Administrator will be able to link/add a user to an already existing team.
- {Administrator - 7} Remove a user from a team – Administrator will be able to remove a user from a team they are on.
- {Any user - 1} changes password – A user will log in and change their password. They will need to enter their current password and a new password that is more than 5 characters.
- {Any user - 2} submit a bug – A user can submit information about something that has not gone as intended within the application.

List of actors

- Athlete - lowest level user, will enter and manipulate their own workouts and workout history, but cannot access any other user's information
- Trainer/Coach - mid level user, can view any athletes' data within their jurisdiction, does not have admin rights
- Admin - system admin, has access to all data from any athlete

Use case name: {Athlete - 1} Enters workout

Participating Actors: Athlete

Preconditions:

1. Athlete is authenticated by login procedure

Flow of Events:

1. Athlete chooses workout entry option and is redirected to workout page
2. Athlete fills in date for desired workout
3. Athlete selects full workout or interval workout mode
4. Athlete enters all data for their workout and presses enter button
5. Data is saved to database

Post conditions:

1. Athlete's data for the workout is stored in database and accessible to that athlete and his/her trainers.

The screenshot displays the 'My Workout' web application interface. At the top, a navigation bar includes links for 'New Workout', 'Coaches/Trainers', 'Administrators', 'Need Help?', and a 'LOG-OUT' button. Below the navigation bar, the main content area features a 'Date Information' section with a green 'AUTOFILL - TODAY' button and three dropdown menus for 'Month', 'Day of month', and 'Current Year' (set to 2017). Below this, the 'Workout Entry Style' section offers two radio button options: 'First Workout Mode' (selected) and 'Interval Mode'. The background of the page is decorated with a green and yellow hexagonal pattern. A URL is visible in the bottom left corner: <http://localhost:8080/elasticsearch/workoutentry>.

Use case name: {Athlete - 2} Views training log

Participating Actors: Athlete

Preconditions:

1. Athlete is authenticated by login procedure

Flow of Events:

1. Athlete chooses training log option and is redirected to training log page

Post conditions:

1. Athlete's data for the workout is displayed on this page and may be removed/edited.

Training Log:										
Date	Hours Slept	Illness	Injury	Percent Health	Cycle	RPE	Time	Distance	Notes	
Sun Feb 26 2017 00:00:00 GMT+0000 (UTC)	8	n/a	n/a	100	no	11	20	2		<div>EDIT</div> <div>REMOVE</div>
Sat Feb 25 2017 00:00:00 GMT+0000 (UTC)	10	n/a	n/a	90	no	9	30	2		<div>EDIT</div> <div>REMOVE</div>
Fri Feb 10 2017 00:00:00 GMT+0000 (UTC)	14	n/a	n/a	100	no	13	30	2		<div>EDIT</div> <div>REMOVE</div>
Mon Feb 27 2017 00:00:00 GMT+0000 (UTC)	13	n/a	n/a	100	no	11	25	2		<div>EDIT</div> <div>REMOVE</div>
Tue Feb 28 2017 00:00:00 GMT+0000 (UTC)	11	n/a	n/a	100	no	11	15	1		<div>EDIT</div> <div>REMOVE</div>
Thu Mar 02 2017 00:00:00 GMT+0000 (UTC)	12	n/a	n/a	100	no	13	30	3		<div>EDIT</div> <div>REMOVE</div>
Thu Mar 02 2017 00:00:00 GMT+0000 (UTC)	0	upper respiratory	Heel	50	no	6	50	50		<div>EDIT</div> <div>REMOVE</div>

Use case name: {Athlete - 3} Edit a workout

Participating Actors: Athlete

Preconditions:

1. Athlete is authenticated by login procedure

Flow of Events:

1. Athlete views previous workouts and finds a row that they wish to change
2. Athlete clicks on the edit button that is on the same row as the workout
3. Athlete may want to load the old workout information by hitting load workout
4. Athlete enters all of the changes they wish to make for that workout
5. Athlete hits submit
6. Data is saved to the database.

Post conditions:

1. The data that has been recently entered and submitted will be the new data for the workout selected to edit.

The screenshot displays a 'Training Log' application interface. A modal window titled 'Edit Workout' is open, allowing an athlete to modify a specific workout. The modal contains several input fields and a 'LOAD WORKOUT DATA' button. The background shows a table with workout logs, including columns for Date, Time, and various metrics like 'upper respiratory' and 'lower respiratory'. The modal form includes the following sections:

- Hours of Sleep:** A text input field.
- General Health:** A dropdown menu with 'N/A' selected.
- Irritation/Pain:** A dropdown menu with 'N/A' selected.
- Wellness Score:** A text input field with a note: '(100% = Full Health, 0% = Not able to Workout)'. Below it is a range slider.
- RPE Information:** A dropdown menu with '----' selected.
- RPE (6-20):** A text input field.

At the bottom right of the modal, there are 'EDIT' and 'REMOVE' buttons. The background table shows a list of workouts with columns for Date, Time, and various metrics.

Use case name: {Trainer/Coach - 1} View an athlete's training log

Participating Actors: Trainer/Coach

Preconditions:

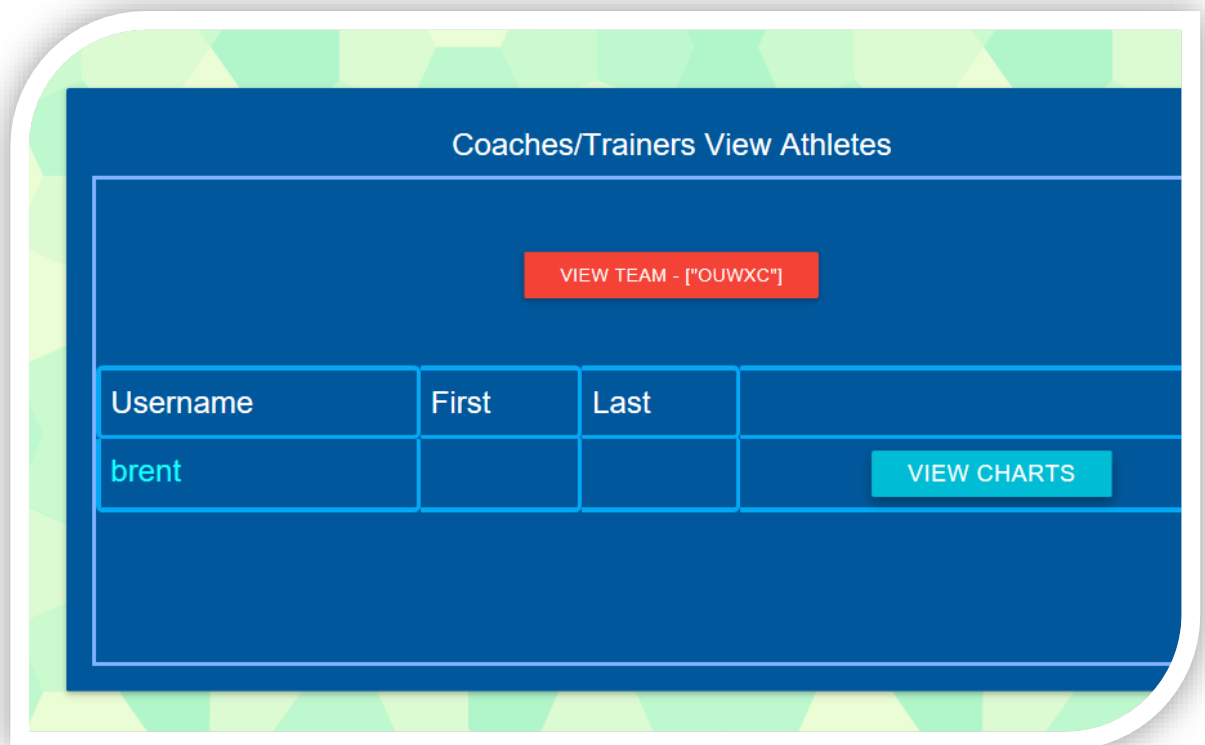
1. Trainer/Coach is authenticated by login procedure

Flow of Events:

1. Trainer/Coach selects the view athletes option
2. Trainer/Coach clicks on which team they want to further inspect (each team has their own button)
3. Trainer/Coach clicks on which user they want to see workout information for

Post conditions:

1. Trainer/Coach will be able to see charts for each user on every team that is Available for that coach.



Use case name: {Administrator - 1} Receive CSV individual data dump

Participating Actors: Trainer/Administrator

Precondition:

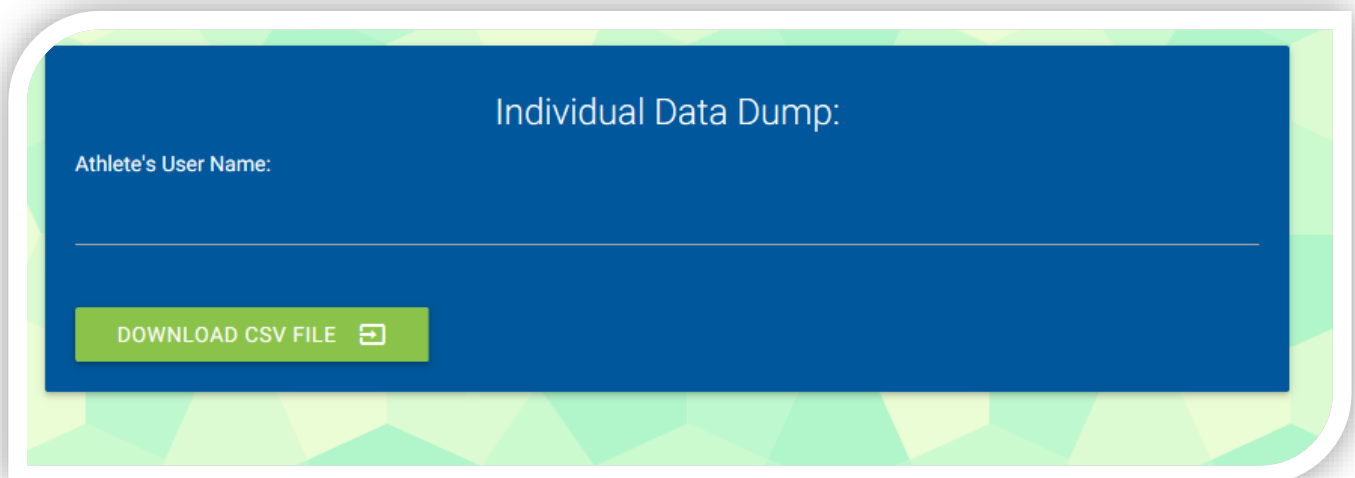
1. Data has been collected for a week/month by an athlete
2. User has been authenticated by login procedure
3. User has trainer or admin role

Flow of Events:

1. User chooses data dump for individual option in the application
2. User is redirected to page asking for which athlete we would like to get data for
3. User enters identification of athlete
4. Application downloads data dump (csv file) of specified athlete/group

Post condition:

1. Trainer has access to raw data in a csv file to perform physiological analysis on.



The screenshot shows a web form titled "Individual Data Dump:" on a dark blue background. Below the title, there is a label "Athlete's User Name:" followed by a horizontal input field. At the bottom left of the form, there is a green button with the text "DOWNLOAD CSV FILE" and a download icon (a square with a downward arrow).

Use case name: {Administrator - 2} Receive CSV team data dump

Participating Actors: Trainer/Administrator

Precondition:

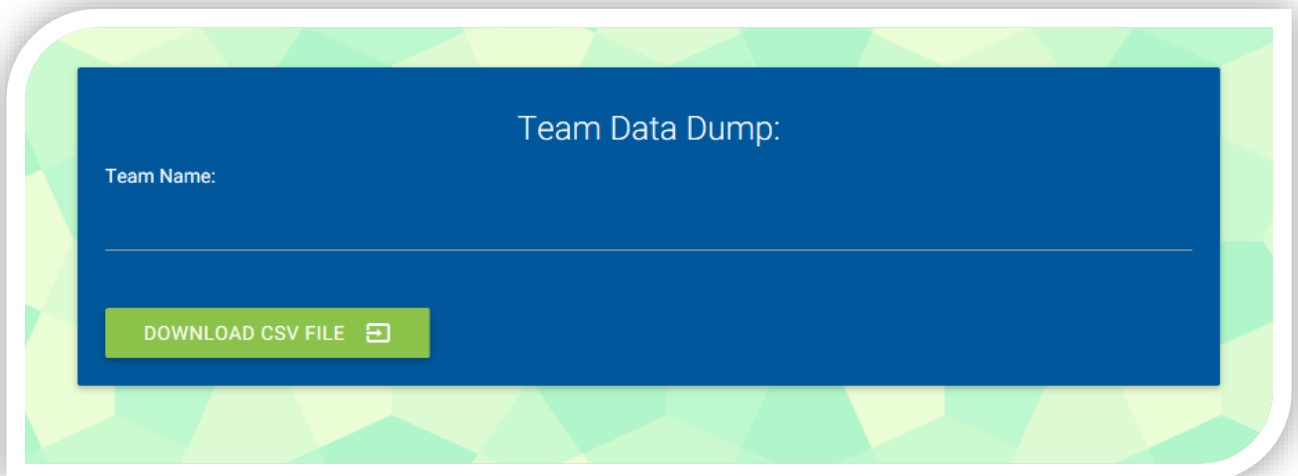
1. Data has been collected for a week/month by athletes on team
2. User has been authenticated by login procedure
3. User has trainer or admin role

Flow of Events:

1. User chooses data dump for Team option in the application
2. User is redirected to page asking for which team we would like to get data for
3. User enters identification of team
4. Application downloads data dump (csv file) of specified Team

Post condition:

1. Trainer has access to raw data in a csv file to perform physiological analysis on.



The screenshot shows a web form titled "Team Data Dump:" on a dark blue background. Below the title, there is a label "Team Name:" followed by a horizontal input field. At the bottom of the form, there is a green button with the text "DOWNLOAD CSV FILE" and a download icon (a square with a downward arrow).

Use case name: {Administrator - 3} Add user

Participating Actors: Admin, Athlete

Preconditions:

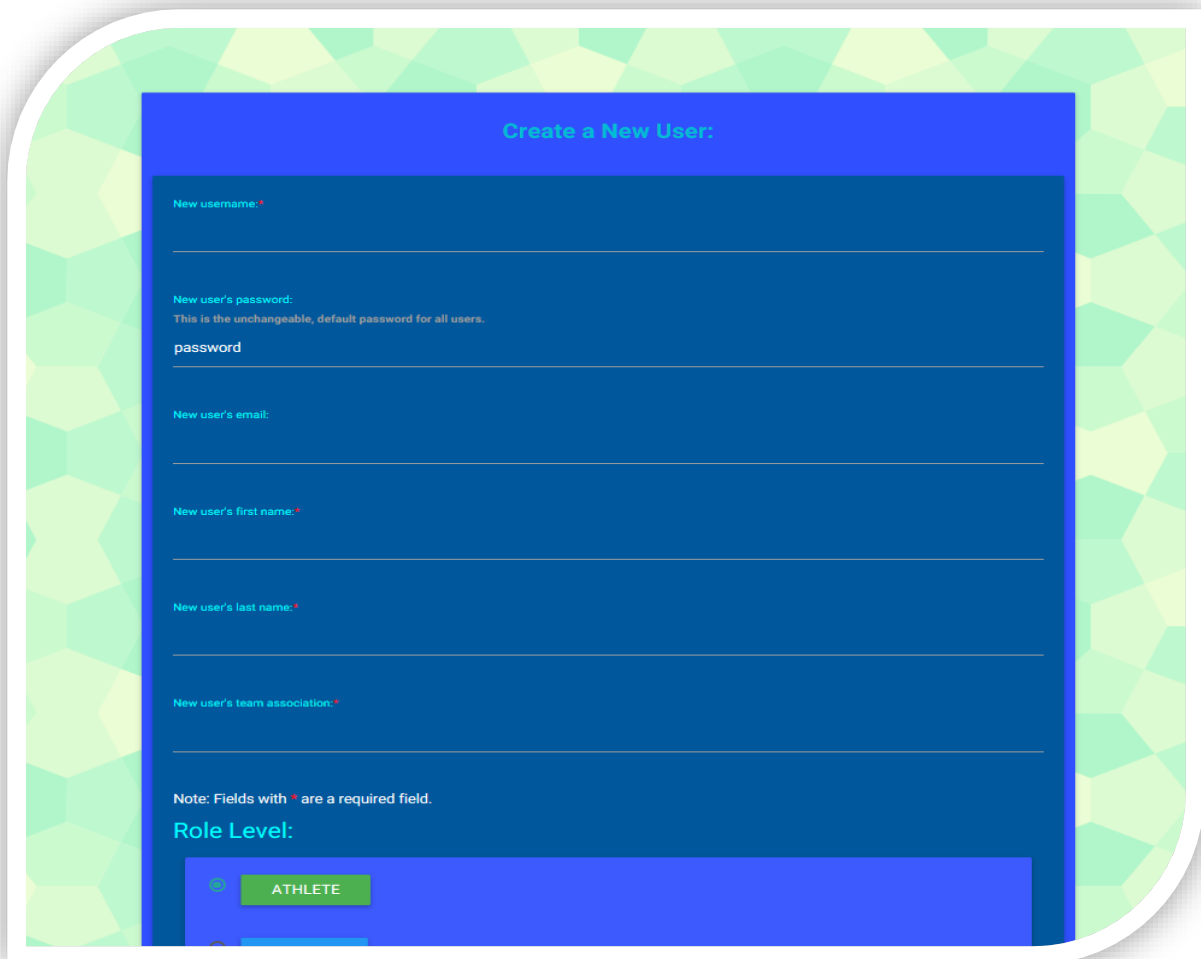
1. Athlete is not currently registered in application
2. User has been authenticated by login procedure
3. User has admin role

Flow of Events:

1. Admin chooses to add new athlete
2. Admin enters athlete's information
3. Information is saved to database
4. Athlete signs into application to create password
5. Password is saved to database

Post conditions:

1. Athlete can now use application to enter daily information



Create a New User:

New username: *

New user's password:
This is the unchangeable, default password for all users.
password

New user's email:

New user's first name: *

New user's last name: *

New user's team association: *

Note: Fields with * are a required field.

Role Level:

☒ ATHLETE

Use case name: {Administrator - 4} Remove user

Participating Actors: Admin, Athlete

Preconditions:

1. Athlete is currently registered in application
2. User has been authenticated by login procedure
3. User has admin role

Flow of Events:

1. Admin chooses to remove athlete
2. Admin enters athlete's information (username and last name)
3. Matching user entry is removed from
4. User receives confirmation of deletion

Post conditions:

1. Athlete can no longer login to application

Remove a User:

Username:

User's Last Name:

Note:
Please ensure you want to permanently remove this user, this process **cannot** be reversed.

REMOVE USER ➤

Use case name: {Administrator - 5} Create a team

Participating Actors: Admin

Preconditions:

1. Admin is currently authenticated

Flow of Events:

1. Admin chooses to create a new team
2. Admin enters new team name and other information
3. Admin submits new team name to database

Post conditions:

1. New team name is available to store athletes

The image shows a mobile application interface for creating a new team. The screen has a blue header with the title "Create a New Team" in white. Below the header, there is a dark blue form area with three input fields, each with a label in light blue: "New Team Name:", "New Team Sport:", and "New Team Gender:". Each input field has a white underline. At the bottom of the form area, there is a green button with the text "CREATE TEAM" and a white right-pointing arrow. The entire form area is framed by a light green border with a geometric pattern.

Use case name: {Administrator - 6} Add a user to a team

Participating Actors: Admin

Preconditions:

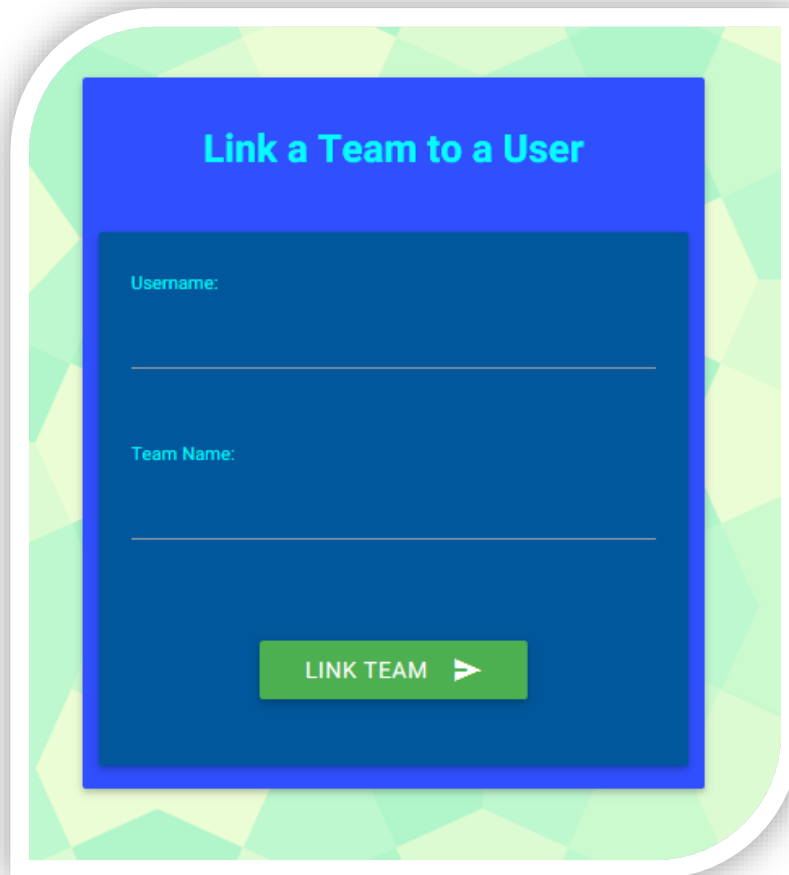
1. Admin is currently authenticated

Flow of Events:

1. Admin chooses to link a user to a team
2. Admin enters both the user and the team's name they want to linked to
3. Admin submits new team information

Post conditions:

1. Team now has that user in its collection of users



Link a Team to a User

Username:

Team Name:

LINK TEAM ➤

Use case name: {Administrator - 7} Remove a user from a team

Participating Actors: Admin

Preconditions:

1. Admin is currently authenticated

Flow of Events:

1. Admin chooses to remove user from a team
2. Admin enters both the user and the team's name they want to be removed from
3. Admin submits new team information

Post conditions:

1. Team now does not include that user in its collection of users

Remove a User From a Team

Username:

Team to be Removed From:

REMOVE USER FROM TEAM ➤

Use case name: User Views Charts

Participating Actors: Admin, Athlete, Coach

Preconditions:

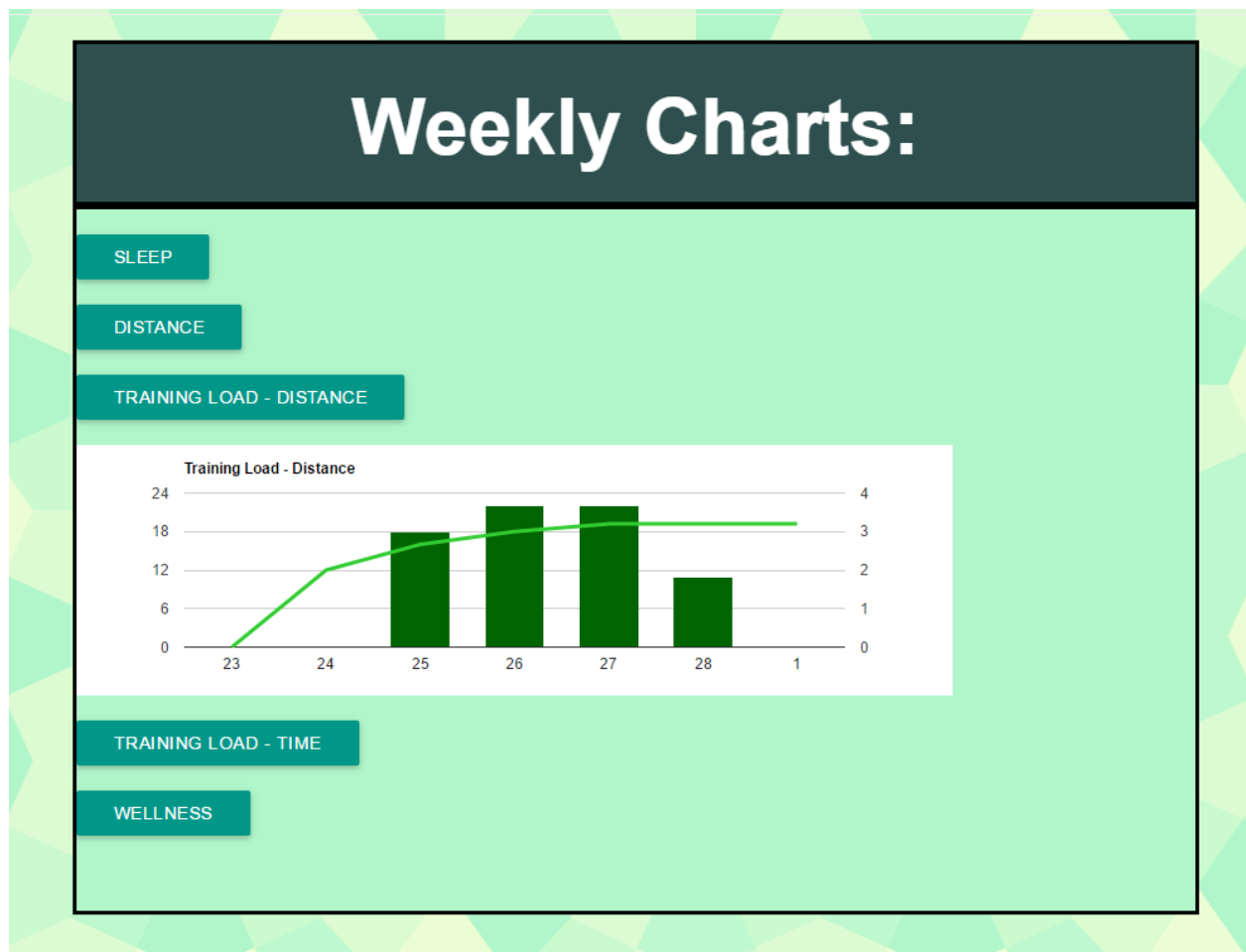
4. User is currently registered in application
5. User has been authenticated by login procedure
6. User has admin role

Flow of Events:

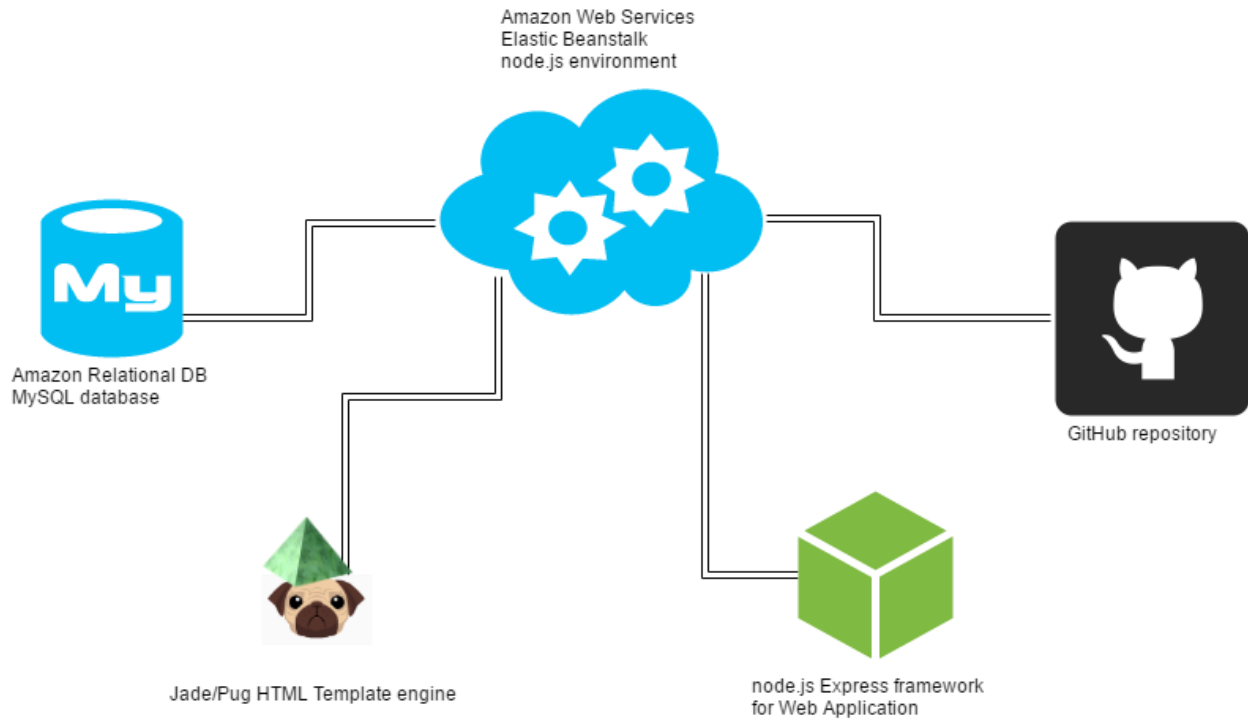
5. User navigates to training log page
6. User selects which chart to view
7. User views chart

Post conditions:

2. Charts are displayed on page



Technology Stack



- **Pug** - Graphical user interface, allows users to view and edit data in our application
 - **JavaScript**
 - **CSS Styling**
- **Materialize** - Allows pretty formatting for HTML/CSS
- **Google Charts** – Allows visual representation of data stored in database
- **Bootstrap** – Works well with Google Charts
- **Node.js** - Back end of application, handles the display of HTML pages along with interacting with Database
- **MySQL** - Database to store all user data
- **AWS EB** - used to host our website, free tier
- **AWS RDS** - used to host our database, free tier:

Narrative:

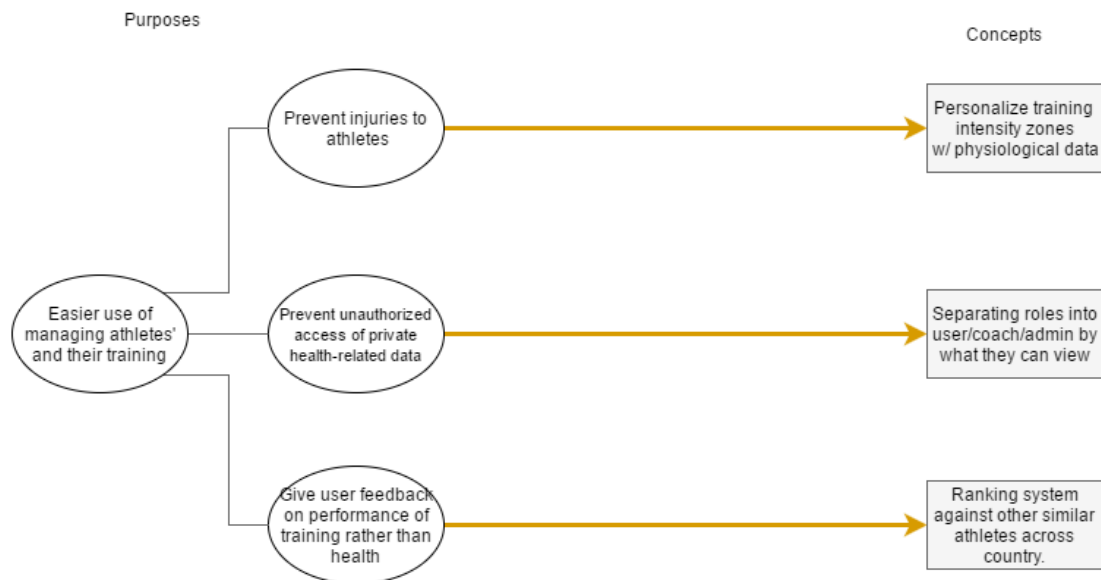
The application is hosted on an Amazon Web Services Elastic Beanstalk environment setup for node.js. With node.js, we are using the Express framework designed for web servers. We are using the Elastic Beanstalk Command Line Interface (eb on UNIX) to connect to our git repo. We can then deploy to AWS straight from the command line with our latest release on GitHub. This helps avoid issues of packaging the project and archiving, etc. We use the AWS Relational Database service to host our MySQL database server which stores athlete data. We also switched from using pure HTML to Pug (formerly known as Jade) for templating HTML files. This allows us to have more dynamic pages to display data/charts/etc.

Design

Design Overview

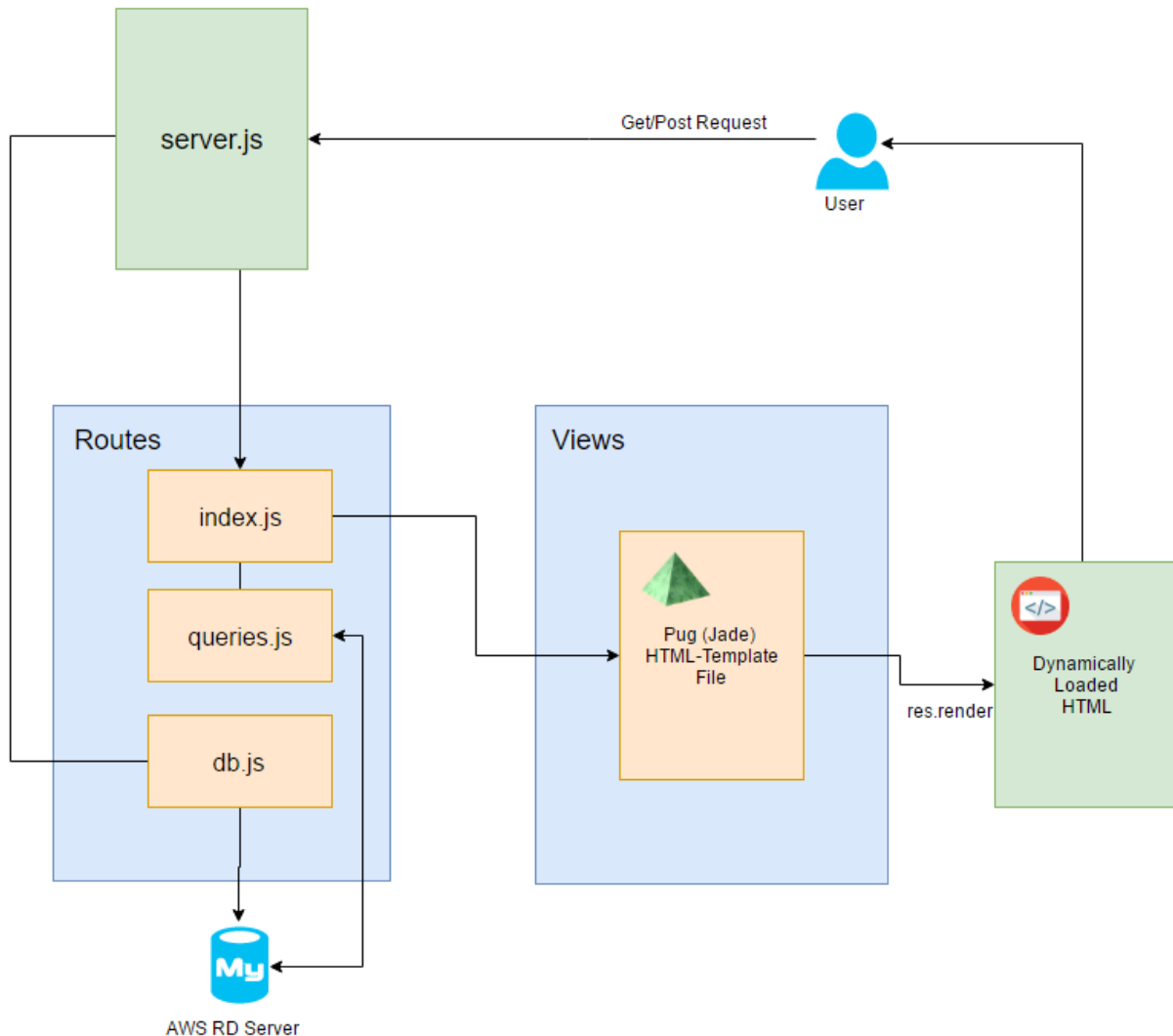
- The general purpose for this project is to provide an easy-to-use way for athletes to log data and in turn provide an easy to access method for trainers/researchers to be able to obtain data to make better predictions about athletic injuries. Some of the sub-ideas from that are how the athlete will be able to prevent injury by better managing their training load, allowing access to confidential data only to authorized users and only when necessary, and allowing coaches to provide feedback to athletes on their training.

Purpose-concept mapping diagram



- Purpose: Prevent Injuries to Athletes
 - The general goal of the OUWXC Performance Program is to allow trainers to better see the training regiment of the athletes in order to better predict/prevent injuries.
- Concept: Personalized Training Intensity Zones (pre-existing)
 - With personalized physiological testing, training intensity zones can be calculated to analyze an individual athlete's physical exertion in training.
- Purpose: Prevent Unauthorized Access to Health-Related Data
 - Due to the medical nature of this project, it is important to restrict access to the data of athletes to only those who need to access it.
- Concept: Roles (Athlete, Coach, Trainer, Admin) (pre-existing)
 - Each role represents another level of access to data. Athletes can only see their own data. Coaches can see some data of the athletes in their team. Trainers can see most data for all athletes. Admins can see all data (login info, etc).
- Purpose: Give User Feedback on Performance of Training rather than health
 - An athlete or coach may want to use this tool to keep track of their training performance rather than just preventing injury.
- Concept: Comparison system to show training stats relative to team (pre-existing)
 - While not necessarily trying to encourage competition, allowing a user to see how other similar athletes are training may encourage them to stay healthier.

Code Architecture



Sequence of events that occur when a user requests something:

1. The user sends a request to the server which ultimately goes to `index.js` to find the associated get/post method for that specific request to attempt to return the page contents
2. Some checks are made such as: querying the database or checking the users role status
3. Renders the pug file and the user sees the requested page

Implementation

- Documentation of the html/pug was handled directly in source for the more sophisticated parts that could be difficult to read such as multi layered card usage for aesthetics.
- Routes contains:
 - Index.js - a file which holds the get and post methods for different pages, as well as passport information
 - Queries.js - a file which holds functions that allow easy use of database queries
 - Db.js - a file which holds the connection information for the database server
- Views contains:
 - All of the .pug files. Pug files are HTML template files which are rendered on call and basically allow for dynamic HTML code with easy use of adding dynamic data and other resources. Pug was renamed from Jade recently, so any documentation on Jade will hold true for Pug.
- Node Modules contains:
 - All of the node modules from npm that either we, or AWS have loaded in. This is a large directory.
- Server.js is the main server file which runs the server and listens for requests.

Database Info

- Database
 - Tables:
 - Athlete_data (workout info)
 - Athlete VARCHAR(45)
 - Date DATE
 - Sleep INT(2)
 - health_status VARCHAR(45)
 - Illness VARCHAR(45)
 - Injury VARCHAR(45)
 - Percent_health VARCHAR(45)
 - RPE INT(2)
 - Time Float
 - Distance Float
 - Notes VARCHAR(140)
 - Surface VARCHAR(45)
 - Workout_id INT(11)
 - Users
 - Username VARCHAR(16)
 - Email VARCHAR(255)
 - Password VARCHAR(32)
 - First VARCHAR(45)
 - Last VARCHAR(45)
 - Team VARCHAR(45)
 - Role VARCHAR(45)
 - Create_time DATETIME
 - Pace_chart
 - Percent_v02_max_low INT(11)
 - Percent_v02_max_hi INT(11)
 - Percent_hearttrate_max_low INT(11)
 - Percent_hearttrate_max_hi INT(11)
 - Perceived_exertion_low INT(11)
 - Perceived_exertion_hi INT(11)
 - Pace_low INT(11)
 - Pace_high INT(11)
 - Zone VARCHAR(2)
 - Username VARCHAR(45)

Links

Application and Code Links:

Live Application:

<http://ouwxcpp.ik3pvw7c5h.us-west-2.elasticbeanstalk.com/>

GitHub Code Repository:

<https://github.com/RobertoWhitmerto/OhioUniversityWXCPeformanceProgram>

Technology Links and Other Useful Resources:

- Front End:
 - HTML Resources
 - <https://www.w3schools.com/html/>
 - JavaScript Resources
 - <https://www.w3schools.com/js/default.asp>
 - CSS Resources
 - <https://www.w3schools.com/css/default.asp>
 - Pug (Formerly known as Jade) Resources
 - <https://pugjs.org/api/getting-started.html>
 - Materialize
 - <http://materializecss.com/>
 - Bootstrap
 - <http://getbootstrap.com/>
 - Google Charts
 - <https://developers.google.com/chart/>
- Node-based Middleware
 - Node Documentation
 - <https://nodejs.org/en/>
 - Express Documentation
 - <http://expressjs.com/>
 - Passport Documentation
 - <http://passportjs.org/>
- Backend
 - SQL Resources
 - <https://www.w3schools.com/sql/>
 - <http://www.mysql.com>
 - Amazon Elastic Beanstalk (Hosts Application)
 - https://aws.amazon.com/?nc2=h_lg
 - Amazon Relational Database (Hosts Database)
 - https://aws.amazon.com/?nc2=h_lg
 - Amazon Command Line Interface (A CLI)
 - <https://aws.amazon.com/cli/>
 - Amazon Web Services (AWS)
 - <https://aws.amazon.com/>

Appendix

Ohio University Women's Cross Country Performance Program

"The injury rates associated with running are relatively high per hour of participation. Unfortunately, when injury does occur the sports medicine staff must retrospectively attempt to determine the cause of the injury. This is a difficult process as injury can occur because of multiple factors (training loads, biomechanical factors, clinical issues, nutrition, etc.), be very individualized, and more importantly most runners use rudimentary metrics such as mileage to track training loads. I am currently working with an interdisciplinary team consisting of faculty from Physical Therapy, Nutrition, Exercise Physiology, Athletic Training, Athletics, and local physicians to attempt to reduce injury rates in our Women's Cross Country Team. We conduct biomechanical, nutritional, clinical, and physiological testing to help develop a comprehensive look into the quantity and magnitude of training each runner experiences throughout a single run and across a whole season. Each athlete is individually modeled and tracked throughout the season to help us better understand injury when it occurs. We also track metrics like sleep patterns, dietary patterns, objective/subjective sensations of injury on a daily basis. Our overall goal is to individualize care for each athlete, and create more informed decisions for our sports medicine staff to help each runner prevent injury as opposed to coping with it once it occurs.

The portion I need assistance with is creating a user interface that is easier and more intuitive for each athlete to use and record the required information. An I-phone app, or online platform where they can go in and log their daily information and see and learn about their own trends in training. The platform also needs to allow staff to update formulas and data of each athlete that impacts the internal calculations. Currently I have built a model within Microsoft Excel that houses all of the calculations, formulas, and data. The plan is to implement an excel page for each runner this year using a cloud system where they can update items on a daily basis (Microsoft One Note or BOX), but the system does not allow staff to review the daily logs of each runner sufficiently nor does it elegantly output required data in an organized fashion. The ideal platform would export the data from all athletes into a concise excel file or similar format.

I have aspirations of providing this application to any athlete that is tested in the Ohio University Gait Lab, and would hope to expand the availability to rowers and teams sports as well. I have a colleague currently working with several major rowing programs across the US, and he has consistently expressed interest in a model for rowers. I am unfortunately not well versed in this area, and would have interest in your thoughts for any outlet that may help me bring a more functional model to fruition. I have attached two excel files. Both files represent my current design in Microsoft Excel with two months of data (September and October). One file demonstrates all of the information that training staff can view in regards to training loads. The second file only shows what the athlete sees as a daily training log. Hidden pages can be revealed by using the "unhide" feature at the bottom of the page. This will allow access to the formulas and underlying calculations that go into individualizing the program."

Michael Clevidence

Associate Lecturer | School of Applied Health Sciences and Wellness | Ohio University

Grover Center E190

E cleviden@ohio.edu | P 740.597.3356