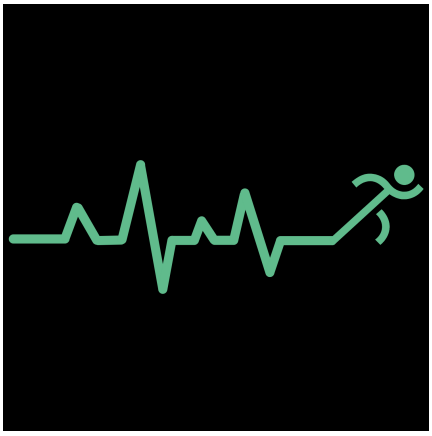
Team Agone

Technological Feasibility Analysis

October 20, 2021



Team Members:

Rylee Mitchell , Kaila Iglesias , Samantha Rodriguez , Jonathan White Velasco

Sponsor:

Adam Stepanovic

Mentor:

Volodymyr Saruta

Table of Contents

1 Introduction

A study from Yale found that 50% of regular runners get injured every year, primarily from overuse. That is one in every two people who run regularly that will suffer from an injury just this year. Insane! For an activity that is so common among people of every age around the world, it is quite a shame that this statistic is so high.

For almost any athlete or any individual that participates in sports, whether it be competitively or as a hobby, injuries are among the worst culprits for impeding progress. While some preventative measures can be taken, such as using proper form, stretching, and taking time off from activity, it is virtually impossible to prevent injury let alone predict when it will happen. As a result, injury is something that can only be responded to after the injury has already occurred.

However, PWR Labs has a better solution - to predict and prevent injury rather than treating injury after the fact. Reading through various statistics on athlete performance and seeking out the minor indicative details of change in an athlete's body is a process that can be time consuming, but incredibly revealing for coaches. These details can influence the development of dynamic training regimens for each athlete in an effort to protect their bodies, thereby reducing injuries. PWR Lab's vision to help facilitate this process, is to take the biometric data that can be tracked from wearable devices (such as Apple Watches and Fitbits) and make it easily accessible to be viewed and analyzed by coaches.

Problem

Currently, PWR Lab has a system to take in continuous streams of data such as heart rate, elevation gain and loss, and other categories and process it into a textual format. Although they have developed a way to collect these data streams, reading through text alone and searching for changes across large amounts of data can be a time consuming and inefficient task. That is why PWR Lab wants to find a way of displaying data and making it easy to interpret for both athletes and coaches. The objective of Team Agone's capstone project is to use the API developed by PWR Lab for data collection and then further process that data into a visualized format, such as graphs and charts.

Solution

Our goal is to build a wellness platform that serves as a one-stop-shop for coaches and athletes alike. This wellness platform will include:

- A customizable environment that allows coaches to select specific data to view and analyze accompanied with a visually appealing GUI.
- A secure and scalable database system to create and store user information.
- A messaging system that allows athletes and coaches to communicate within the portal, sending notifications when a new message is received.

Being able to predict and prevent running injuries is extremely valuable. However, this can go far beyond only runners; this wellness portal could be used in several different professional fields. For example, physical therapists play an integral role in helping athletes

and others recover from varying types and severities of injury. Consider a therapist that is trying to determine whether their suggested exercises and activities have been successful in treating an individual. Our wellness portal, used alongside devices that track the necessary biometric information related to that individual's injury, could provide the physical therapist with visual information that would allow them to more easily see whether progress is being made. They could even fine tune their treatment plan and make adjustments that could rapidly improve the healing process, allowing people to recover from injuries faster.

This is why our wellness platform is so vital to not only runners, but to anyone that could benefit from visualizing data relating to and produced by the human body. Now that we have established the problem and suggested a general solution, we can dive deeper into a careful analysis of the requirements for building such a platform. At this early stage of development, we are in the process of analyzing the key components of the platform including the technological challenges, the potential solutions and various technological routes along with their alternatives, as well as the integration aspect of it all.

2 Technological Challenges

The main task of the web application is to collect the data from the athletes in a more efficient fashion. This will be efficient for the athlete to get a geared training plan for them as well as the coach to make that training plan. Having a web application will also allow for the athlete and coach to remain in contact about this training plan and how well it works for them.

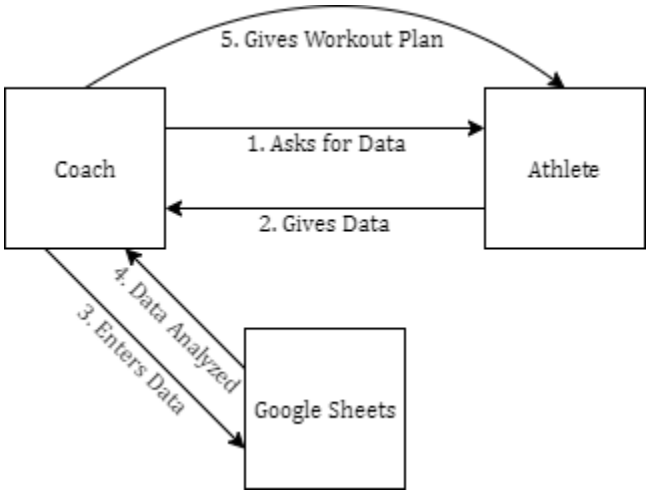


Figure 1. The current system that the coach is working with. Depending on the number of athletes, analyzing each one of their data and giving a specified workout plan can take 8-9 hours.

Considering these key goals and the downfalls of the current system (Figure 2), we have identified the following challenges that we will be facing during development:

- **Flexibility in data analysis -** Ideally, the web application will have access to different forms of data personalized to what the coach specifically wants.
- **Secure coach and athlete portal -** In theory, the only person allowed to see all the athlete's data is the coach, therefore the athletes data needs to be secured.
- **Mobile-friendly web application -** Nowadays mobile devices act as a main avenue of communication. Providing easy access to web portals from such devices will add convenience for the athletes and the coach. We want to have our web application easily accessible through a mobile device.

These are the most crucial technological challenges for the development process. The core technologies of our system must address these issues in an effective manner. In the next section, we will first analyze each challenge and propose a solution as it pertains to that problem.

3 Technological Analysis

On the basis of our technological challenges, we have identified the major design decisions that are critical to this project's success. Our goal is to develop a product that best fits our client's needs. The primary criterion of analysis is how well each tool will help contribute to the value and functionality of the final product. In each of the following subsections, we will discuss the issues, introduce key characteristics of an ideal solution, and analyze various alternatives before coming to a decision on a technology.

3.1 Database Management System

3.1.1 The need for a Database System

The database serves as the backbone for our module. It stores the data that will help the coach and athletes. Without the data, none of the other components of our system will be effective in assisting our client. Therefore, choosing an appropriate database is vital to our project.

3.1.2 Desired Characteristics

- **Performance-** A database that can handle the large amount of data that is being entered every day.
- **Scalability-** Scalable to extend to other coaches and athletes of other institutions other than NAU.
- **Security-** A secure database that handles athletes' sensitive information as well as the team.
- **Cost-** A database that is not only effective in storing data but effective in price as well.

3.1.3 Alternatives

MySQL

MySQL, being one of the more popular database systems and therefore the first database we are going to investigate. Currently owned by Oracle Corporation, MySQL is an open-source relational database management system (RDMS). It is well known for its robustness and performance. There are built-in security features that the system has to offer. For example, there are secure connections, authorization, authentication, privileges, and data encryption.

MariaDB

MariaDB is a branch of MySQL built under different visions of performance, reliability, as well as openness. Similar to MySQL, MariaDB is an RDBMS also, but it has better speed and scalability. It is also under the GNU General Public License because it is open-source software.

PostgreSQL

After MySQL, PostgreSQL is the second most used database management system. It is available for free and expandable. Also, supporting JSON and working well with different analysis programs, such as MATLAB, R, and Python. Python being a programming language that our client is using for his database analysis. PostgreSQL works well with different frameworks, Django, Ruby on Rails and others that we could also use for this project.

Redis

Redis is an open-source software. It has a BSD license; unlike the other systems we are investigating. Giving no limitations to what features we can use. Also, unlike our other options, Redis is an in-memory data structure store, making it one of the quickest database systems on the list. A downside of Redis is that there is no direct querying. While, sometimes Redis can be used as an add-on for other databases, it can also be used as its own. Instead of being known as a database server, it is known as a data structure server.

MongoDB

To widen our search, we looked at a database that is non-relational. MongoDB is a NoSQL document-oriented database. It is mostly used for data storage of high amounts. A downfall is that MongoDB doesn't work well with complicated table structures and instead is on a single document. While the system does have some security features it does not have the major ones, such as automatic encryption or authentication.

3.1.4 Analysis

On the basis of the criteria above (3.1.2), we will analyze five options: MySQL, MariaDB, PostgreSQL, Redis, and MongoDB. To investigate each system, we will be running a dataset (of 500KB) with arbitrary data and test it against the above criteria. Also, reading through

the documentation to get a better understanding of how each database management system works.

MySQL

MySQL is not recommended for a database that is going to grow because it was not built with scalability in mind. For our project, the database will grow. Furthermore, some features and plugins are only available on exclusive editions. The system is under a dual license, which means you can use the GNU General Public License or a standard commercial license purchased from Oracle. The price for the editions start at $2,000 annually and rise from there up to $10,000 annually. The price would lead us to default to the standard edition. While none of the features on the commercial version would be essential, having them unavailable would limit any optimization in the future. In testing of MySQL, it performed poorly in speed. In query functionality, it returned the results in 2.2 seconds. While this is under a different edition than the one we may use in our project, we feel that it is a good indicator of what MySQL is capable of. In conducting a load/stress test on the database, for security, MySQL was able to upkeep its services and stay operational.

MariaDB

In testing the same data, MariaDB did not perform highly in query parallelism. It took 1.3 seconds to output the results. While right now that is not a high number, in order to output results along with visuals of the results, we need a much quicker system. For the load/stress test of the data, MariaDB was able to stay operational for most of the test. The test was conducted a total of five separate times and did not crash for three of those five tests. From the lack of features such as query parallelism and partial indexing, it takes away from the overall potential of the database.

PostgreSQL

In testing, PostgreSQL performed as one of the quicker databases we will be investigating. For query functionality, it returned the results in 0.8 seconds. PostgreSQL also offers high scalability and high performance. For security testing, it performed equally with MySQL. It offers features such as authentication management and data encryption for security.

Redis

The query functionality test was not able to be done, but instead comparing the read/write speeds to the others, Redis completed the task the quickest. Redis does have a built-in encryption for the data but would need add-ons to have authentication and authorization security.

MongoDB

In query functionality, MongoDB tested faster than MySQL but slower than PostgreSQL at 1.4 seconds. For security, it did not test highly compared to the other tested systems. The load/stress test showed it was not as reliable with a two of five-success rate. However, MongoDB was able to recover faster (0.5 seconds faster) than MariaDB was.

3.1.5 Chosen Approach

In general, all database systems have their strengths and weaknesses of their own. In our analysis, MongoDB and Redis did not have very well implemented security fixtures. While MongoDB did recover from crashes, it suffered in being non-relational. MariaDB was quicker than MySQL was but since our client is currently using MySQL, we have MySQL and PostgreSQL as our top two.

	Performance	Scalability	Security	Cost	Total
MySQL	2	2	4	4	**12**
MariaDB	3	4	3	5	**15**
PostgreSQL	4	4	4	5	**17**
Redis	5	5	3	2	**15**
MongoDB	4	5	2	3	**14**

Table 1. Evaluation of criteria for database management systems. The scale is 1-5, where 5 represents the best. The total score is out of 20 points.

As shown from Figure 1, PostgreSQL has the highest score overall. MySQL has the second lowest score. PostgreSQL does score higher in every criterion we are looking at, for our client it would be an easy transition to stay with MySQL. Looking further into other features, PostgreSQL will be able to work better with other data visualization languages. For that reason, we will be using PostgreSQL for our database management system.

3.1.6 Proving Feasibility

Moving forward, we will want to test our database system more. Getting data from our client will assist us in testing against data that will be more similar to what the system will be receiving. Using the demo data, we will be able to see the storing system, the security of specific information, and the speed of the query functionality.

3.2 Hosting Services

3.2.1 The Need for a Hosting Service

As discussed previously, we need a database system to hold all the data. We also need a service that holds such databases to be accessed on any computer. For this reason, we need a database hosting service that can manage the database environment. Some of the criteria from the previous section will be similar to that of this section.

3.2.2 Desired Characteristics

- **Scalability-** A highly scalable database that can adapt for future growth of our product.
- **Reliability-** Reliable storage of the data over a long period of time, as it will be used year-round by athletes and coaches.
- **Cost-** An inexpensive system that won't become a burden financially for our client.
- **Security-** Holds specific information about the athletes securely.
- **Compatibility-** Compatible with our technologies chosen, specifically the database management system.

3.2.3 Alternatives

AWS

AWS is the most popular cloud computing service provider. It offers a wide range of options for different types of projects, which is why it has popularity in the field. It was developed by Amazon as a cloud computing software. Our client is currently using AWS to store their other data.

VMware

VMware is a software company that provides virtual software as well as cloud computing. VMware has similar security features as AWS does. VMware can also act as an addon for AWS. While this feature may go beyond the scope of the project it can be helpful for future iterations.

Digital Ocean

Digital Ocean allows for applications to run on multiple computers on a cloud framework. The company worked to make reliability their number one priority. To help with security, each server is being run on their own private network. Similar to AWS and VMware, the data is encrypted in transit and at rest.

3.2.4 Analysis

Based on the desired criteria above we will investigate three options: Amazon Web Services (AWS), VMware, and Digital Ocean. We will be looking over the documentation and the descriptions of each product. We will also be testing them each with the arbitrary dataset from the previous section.

AWS

Since AWS has such a large base of customers, they have proven to be very reliable. This is being upheld by an abundance of servers that hold the backup data to keep a constant live connection. With AWS, how much space you use is how much the service is going to cost. It is unlike other services that have a monthly or annual subscription. As we are collecting sensitive individual data, we not only need a reliable platform but a secure one as well. Though the more expensive services that AWS have more security features to offer, those

provided are able to provide basic security encryption. AWS is also compatible with PostgreSQL. During the test, the upload of the dataset was almost instantaneous at 0.3 seconds. Because of the compatibility with PostgreSQL, it was an easy system to use as well as provide information.

VMware

VMware is subscription-based software with either a one or three year licensing purchase. Since there is subscription based licensing, you only get the space you pay for and it is not scalable. You would need to increase your subscription. Even though you would have paid for a year and needed to upgrade, you would need to pay an additional year or three. There are backups of the given data to keep a reliable connection. Unlike AWS, this software doesn't have a specific service for PostgreSQL databases, but it is still compatible. The upload of the dataset was also quick with a similar time to AWS (0.4 seconds).

Digital Ocean

Having backups of data does not come at an extra charge, unlike AWS. For cost, there is no scaling compared to the data size but there also isn't a yearly subscription. It is more of a manual operation on a monthly basis to determine what amount of space we would need. Digital Ocean is also compatible with PostgreSQL, while this is an upside to the software it is not compatible with the other database systems we investigated. In the case of needing to switch systems this would become a very large hurdle to overcome. Digital Ocean was also very quick in uploading the testing dataset.

3.2.5 Chosen Approach

All of these technologies can be very helpful, our primary concern is reliability. There needs to be a constant input or output from our product, or it will be of no use to our client. It was a very even score for reliability with AWS and Digital Ocean scoring the same. AWS ended up scoring the highest overall, as shown below.

	Scalability	Reliability	Cost	Security	Compatibility	Total
AWS	5	5	4	4	5	**23**
VMware	1	4	3	4	4	**16**
Digital Ocean	3	5	5	4	3	**20**

Table 2. Evaluation of criteria for database management systems. The scale is 1-5, where 5 represents the best. The total score is out of 25 points.

Based on what was discussed and investigated, we will be using AWS as our database hosting service. It fits our criteria the best of the three technologies. A great advantage is our client's company is currently using AWS to hold their data from their other projects.

3.2.6 Proving Feasibility

In the future when we have our database system up and running, we will be testing how AWS holds the abundant amount of data. This data will be the same as mentioned in 3.1.7. Once they are conjoined, our team can test the security and reliability of the hosting service.

3.3 Framework for a Web Portal Interface

3.3.1 The need for a Framework

The framework serves as the glue that will hold all other technologies together. Choosing the best framework will be vital to the success of our project. Our framework will be the base that receives data from the API and database while also allowing coaching staff to access the visualized data, communicate with student athletes and secure users and their data.

3.3.2 Desired Characteristics

- **Development Process-** While some of us on our team might have experience building web portals and manipulating data, not all of us do. We need a framework that is easy to pick up and learn. We do not have the time to learn one of the more complex frameworks in depth so we will need something with a low skill level while also giving us the proper tools to accomplish our goals.

- **Performance-** A Framework that is fast is always ideal. We will be dealing with large amounts of student data. The API will constantly be updating data to the site and we will be serving over 120 athletes plus their coaches. Not only do we need speed but also a framework that can handle such a large amount of users reliably.
- **Maintenance-** While we may never work on this web portal again after we graduate, that does not mean that PWR Labs will stop supporting the portal after it has been developed. As a result, we need a framework that allows for long term maintenance. We do not want to end up with the PWR Lab team scrapping this product. Future planning is essential for our project.
- **Scalability-**We need a framework that allows for flexibility in terms of scale. If other teams decide to use our portal

3.3.3 Alternatives

React

React is “A JavaScript library for building user interfaces”. React is component-based and declarative making it the favorite of many website developers and website builders including Facebook, Instagram, Netflix, uber, etc. In fact, React was created by Facebook and released in 2013.

Angular

Angular is a framework created in 2009. It is now maintained by a team working for Google. Angular is a TypeScript based framework and prides itself on performance and speed. Just like React it has declarative templates. Additionally, Angular has been built to be viable for mobile web browsers as well as regular desktop web browsers.

Vue

Vue is the last framework we will be looking at. Vue is an open-source front end framework that was created by Evan You in 2014. Evan and his team still maintain and update the site to this day. View is very similar to React and Angular in terms of features it offers.

3.3.4 Analysis

React

Advantages

While React may be known for its steep learning curve, Reactjs.org has a plethora of documentation, tutorials and other training resources that will make learning hopefully not too difficult. Additionally, React uses virtual DOM which allows for the DOM to exist entirely in memory. This speeds up performance immensely compared to other frameworks using DOM. Lastly, the JS library provided by React is enormous. On top of that it's very flexible and has been the main choice for many developers still today.

Disadvantages

React is developed constantly and at a very quick pace. This causes issues in terms of new documentation not being readily available until later and less predictability of the framework's direction. Additionally, React is not object oriented, which can be very confusing for some developers. React is also known to have a bit of a steep learning curve.

Angular

Advantages

Most notable of Angular is its two-way data binding which allows for view changes to be displayed instantaneously in the model. This also works the other way around. Angular has found a way to reduce CPU strain by exclusively serving static files which is great for performance. Angular also has a form of DOM manipulation which involves its two-way binding. The binding method removes the need to manually translate and update the DOM elements.

Disadvantages

Angular has tools that can be very difficult to pick up and use. The learning curve for angular requires a lot of time and effort to learn. Additionally, this framework can be slow and require a lot of time for browsers to load the site. On top of its learning curve it requires developers to be very familiar with Model-View-Controller patterns.

Vue

Advantages

Vue is a lot like React in terms of its use of Virtual DOM, ease of use and large amount of documentation that contributes to its easy learning curve. Vue can be used for many more challenging web interfaces as well as single page applications. The many tools offered by Vue make it almost trivial to start sites and up the scale. Like Angular, Vue also has two way data binding making it almost the best of both worlds involving the previous two frameworks. Vue is extremely flexible in terms of transferring from React or Angular to view. However, this will not be necessary as our client is already using Vue.

Disadvantages

One of the biggest disadvantages to Vue would be its scalability. While not terrible it can be hard to adopt it for large scale projects. Vue is a more recent framework, so it hasn't been adopted into the market like React or Angular.

3.3.5 Chosen Approach

To better show our choice the graph below shows a side-by-side comparison of the few frameworks we chose to review. It includes all the desirable characteristics we require and gives each framework a score.

	Performance	Maintenance	Scalability	Development Process	Total
React	5	4	4	3	**16**
Angular	3	4	5	2	**14**
Vue	5	3	3	5	**16**

Table 3. Evaluation of criteria for site framework. The scale is 1-5, where 5 represents the best. The total score is out of 20 points.

As you can see Vue and React are both on equal footing. The deciding factor for us is that our client already uses Vue as stated before. We believe with Vue's ease of use, and excellent performance it will be the most fitting software for our wellness portal. Additionally, the fact that our client uses Vue will make integration a seamless process.

3.3.6 Proving Feasibility

Testing the feasibility of our framework will require us to build a skeleton framework for our site. This may prove to be a difficult task as we will need to read up on documentation of our said framework before we can even begin. Additionally, we will need to test our back end to make sure we will be able to communicate with the database. Feasibility will be crucial for this aspect of our project since the front-end framework will be communicating with every other technology in the project.

3.4 Data Visualization

3.4.1 The Need for Data Visualization Capabilities

Data visualization is one of the most important aspects of our web portal. It is the solution to both the client's and the user's problem of not being able to visualize the data they are interested in and already have access to. Thus, our goal is to implement a data visualization aspect that is not only presented in a clean, user-friendly format, but one that will also be customizable to a user's needs.

3.4.2 Desired Characteristics

- **Customizable -** The data visualization portion of this web portal is one of the most important and will be one of the most dynamic parts to program. The client has noted that their customers require a portal that can include a wide range of data visualization details and columnal information. Since users of the portal vary highly in what information they want to see and the orders in which it appears, the best approach to the user dashboard would be to allow the user to edit a primary dashboard template to include all of the information they are consistently interested in. Afterward the user can save the template and have that set of information available in their dashboard along with the ability to reorder it when needed. Columns of data would best be represented in a drag and drop format to further increase the customizability of a user's data dashboard, but if this is not possible, a selection / checkbox format would be equally applicable.

- **Clearly Depicted -** Regardless of the biometric data types that are being collected, they should be easy to differentiate. Since the data changes across time by either increasing or decreasing in value (i.e., VO_2 max, elevation gain, heart rate, etc.), it seems most obvious to represent this data in line charts. Therefore, these charts should be represented in such a way that the data selection and viewing process is easy to navigate. The data charting characteristic should be clear and adjustable, in order to significantly facilitate the work of interpreting data by the individuals using the portal.
- **Performance -** Since there is an abundance of data being processed and we intend to make the web portal accessible on mobile devices, the data visualization should be consistent and fast. That means that graphs and their respective figure descriptions should look the same regardless of the device they are being viewed on. The same functionalities of being able to manually select which data streams to view should also be available in web and mobile formats. In a mobile scenario, it is especially vital that graphing remains consistent. If a user wants to quickly pull up biometric information and provide analysis or insight to an athlete, the graphs should be as functional as the web format.
- **Compatibility -** The data visualization software should be compatible with the rest of the chosen technical solutions and or the materials that our client has already implemented. Considering that our client has already developed an API for fetching data from biometric tracking devices and the data is delivered in a JSON blob to the

database, it seems most viable to use a data visualization package that works with JavaScript. This will ensure that if the client decides to continue using the web portal solution, the data visualization aspect can be maintained alongside the original API for data collection.

3.4.3 Alternatives

Chart.js

Chart.js is meant to be a lightweight API that is simple to understand. It is a free, open-source JavaScript library that has the capability of supporting 8 different types of data visualization. Chart.js can be used with React, Angular, and Vue. Chart.js was developed in 2013 and is one of the most popular charting libraries used in GitHub repositories. Chart rendering in this software is carried out in conjunction with the HTML5 canvas element.

Plotly.js

It is a standalone Javascript data visualization library that can be used to produce dozens of chart types and visualizations. Plotly was founded in 2012 and is considered to be one of the fastest growing chart libraries. It allows for out-of-the-box features as well as sophisticated visualizations in 2D and 3D. There are open-source interfaces of Plotly.js available for Python, R, MATLAB, and React.

ApexCharts

ApexCharts is an open-source JavaScript library that allows for developing charts and data visualizations with a simple API. This library was released in 2018 and is relatively new, but it has high functionality and is a simpler API than some of the more mature visualization API's available in JavaScript (Plotly.js, Highcharts, etc.). ApexCharts' API is intended to be used with desktop, mobile, and web applications and can be integrated with Angular, Vue, and React.

Highcharts

A pure JavaScript data visualization library that is meant to be used with web applications, mobile apps, and IoT devices. It provides interactive charting capabilities, a wide array of chart types and designs and is compatible with Angular, React, PHP, .Net, and Vue. This software library was released in 2009 and is free for personal use but not for commercial use.

3.4.4 Analysis

Chart.js

Advantages

Some of the most obvious advantages to implementing the chart.js library is that it is free, interactive, and offers a variety of charts. Another benefit is that this library is easy to use because it has lower functionality than some of the other charting libraries. Furthermore,

even though chart.js offers only 8 chart types, they are all responsive, which means they will render appropriately on different types of devices. Using Chart.js would be a good route if we are looking for a very lightweight graph rendering library.

Disadvantages

One of the main disadvantages to Chart.js is that it has slow rendering. Rendering abilities will also be impacted depending on the browser that users try to use the mobile application in. Since Chart.js relies on the HTML5 canvas element for depicting charts, which means that we would also need to develop Polyfill code to allow for Chart.js charts to be rendered in web browsers that do not support HTML5 features. Lastly, this library has a low-quality zoom element and exporting capabilities. This might be problematic for users if they want to view smaller sections of data and or export the data for viewing later.

Plotly.js

Advantages

The primary advantage of Plotly.js is that it is highly customizable and interactive. Plotly.js has integrated zoom capabilities as well as data filtering tools. Plotly can also be used with highly complex graphs including 3D graph rendering and graphs with complex axes or combinations of information. Data visualization also looks simple, yet sophisticated thanks to rendering through WebGL. Charts can also be edited online, they are responsive, and can be easily exported in high quality images. This library is also free and open source.

Disadvantages

The main disadvantage of Plotly.js is that it is not explicitly integratable with Vue. However, David Desmaisons made a wrapper for plotly.js that can be used with Vue (Vue-plotly) and it is endorsed by Vue. Furthermore, Plotly.js has been noted for its poor documentation. If we are to move forward using a Vue framework, we may find it difficult to not only integrate the wrapper for plotly.js in our application, which might take time away from ensuring customizable data functionality.

ApexCharts

Advantages

ApexCharts is a free and open-source library for beautiful and simple graph visualization. It is compatible with React and Vue which are two frameworks we are considering using. Charts developed using this API are responsive and the API is relatively simple. Other functionalities include zoom and SVG rendering.

Disadvantages

The main disadvantage of ApexCharts is that graphs and charts are slow rendering. Furthermore, the capabilities that it does offer, such as zoom and interactivity, are not as sophisticated as the other libraries. Lastly, it does not appear that ApexCharts allows for exporting of data in the form of images or XML files.

Highcharts

Advantages

There are many advantages to Highcharts. It is responsive, well documented, handles a variety of data types and amounts, and is extremely easy to customize. Highcharts is reported to have a relatively low learning curve, which is good since we aim to add a lot of customization functionalities for the users to interact with. There is also built-in functionality for exporting charts to image files.

Disadvantages

The main disadvantage of this data visualization library is that it is the only paid service. Although Highcharts is free for personal use in website or school projects, if we intend to hand this product over to the client Highcharts will then be used in a commercial fashion and will need to be paid for.

3.4.5 Chosen Approach

The table below shows a breakdown of the details on the characteristics of a data visualization library that will be useful to our web portal and solving the client's problem.

	Cost	Functionality	Development Process	Aesthetic	Total
Chart.js	5	2	5	3	**15**
Plotly.js	5	4	2	4	**16**
ApexCharts	5	3	4	4	**16**
Highcharts	3	5	5	5	**18**

Table 4. Evaluation of criteria for database management systems. The scale is 1-5, where 5 represents the best. The total score is out of 20 points.

3.4.6 Proving Feasibility

Overall, Highcharts appears to be the most useful data visualization tool as it provides the most functionality with the easiest implementation. Among other charting libraries, it is consistently rated as one of the easiest to use due to its low learning curve. Considering the sheer number of advantages and ease of usability that this library provides, it may be worth implementing this over other libraries using the free version and then considering the paid version with the client if they determine that they like this tool. However, if we do decide to use this software we need to get it approved with PWR Labs due to it being a paid service. Otherwise, the next best option we would be using is ApexCharts because of its high functionality and the fact that it is free even for commercial use that can accomplish what Highcharts can while at a different degree. Implementing Highcharts into Vue is simple as there is already a wrapper for Highcharts in Vue, which allows for generating interactive charts.

3.5 Secure Authentication

3.5.1 The Need for Secure Authentication

An essential part of this portal is making it secure. Each athlete should only be able to access their individual data while coaches should be able to see their entire team's data. To do so, we need a way of creating some sort of log-in or unique identifier, such as an ID, to authenticate users and keep data confidential. We want to create a cohesive and consistent wellness platform, so the security aspect must contain all of the desired characteristics listed below.

3.5.2 Desired Characteristics

- **Implementation-** Recognize and securely store existing users' information while also being able to create new users and add them to the system.
- **Confidentiality-** Restrict athletes to only their own data entries and allow coaches access to all of their athletes' data, but not their log-in credentials.
- **Security-** Keep data private and therefore protect against potential vulnerabilities and attacks.
- **Scalability-** Scalable to beyond the NAU Cross Country team and easily maintainable for our client and future potential users

There are a couple of ways this can be done. In any case, we must have a way to store all of the log-in credentials securely, and this is where we bring in a database for support. The question is not if we will use a database- because without a doubt we are. The question is how we can best implement the authentication system and how we can best ensure its security. A couple of our options are mentioned below, along with a description/analysis of their respective pros and cons.

3.5.3 Alternatives

Django

Django is a high-level Python web framework that essentially takes care of all the details of web development for you. It assists in creating software that is complete, versatile, secure, scalable, maintainable, and portable- all attributes that we want for our system. It is free, open sourced, and is well documented in an effort to make implementation as easy and cohesive as possible.

Create Our Own Database

While a heavy task, it is an option for us to create our own database and tailor it to our specific needs. Databases are good because multiple users can read and modify the data at the same time. Creating our own database allows for customizability; it would be extremely scalable and easily modified as requirements change.

Firebase Authentication

Firebase Authentication is a backend service that securely saves and stores user data in the cloud. It supports authentication via passwords, phone numbers, and even popular federated identity providers like Google and Facebook. It stands on its own as a complete user interface or can be manually integrated into our own mobile application.

3.5.4 Analysis

Django

Mainly, Django provides a secure way to manage user accounts and passwords by default. Instead of creating our wellness platform from scratch, we could use the existing source Django and properly configure these components into our own website/application. It uses a database-based scheme to store authentication information or is customizable to where we could add in our own existing system (say, if we wanted to partner with NAU to use the Cross-Country team's NAU log-in information for our platform). Not only that, but it uses a hash function to encrypt passwords and protect them from attacks; it also prevents other common vulnerabilities including cross-site scripting and forgery. Django is Python-based, and our client did mention that they would prefer to use Python for the data science aspect of the platform; this is an extra benefit and allows for easy implementation.

Create Our Own Database

The benefit of creating our own database is that we've all had experience doing so through MySQL. Of all the options, it is the most familiar, but also the hardest to implement; we leave a lot of room for error and potential security bugs. We could create a database that contains a single table with each user having their own row that would include their user ID, password, name, etc. However, creating our own database leaves the question regarding versatility, maintainability, and portability. We would have to be sure to provide extreme documentation, which provides a completely additional workload. Another major drawback of creating our own database is that users would directly enter their information, and therefore pose a potentially significant security risk. We are not familiarized enough with the ways in which to protect a system from errant users, and although an option, I do not believe it is our best option.

Firebase Authentication

Firebase Authentication is a great option for our portal. The Firebase Authentication SDK automatically provides methods to create and manage users while also handling the task of resetting forgotten passwords (which will inevitably occur). It also, like Django, is already pre-built for us and we simply would only need to integrate it into our system. This allows for easy implementation as well as saves us and our client a ton of time and resources. Firebase is compatible with Python which is a crucial necessity as our client did mention they would prefer to use Python for the data-science aspect of the portal. However, one drawback is that by default, authenticated users can read and write data to the real time Database cloud storage. This makes the system slightly less secure and allows much easier access for potential attackers. This setting can be toggled, but overall leads to more work and more room for error.

3.5.5 Chosen Approach

The table below rates the two options above on four key components of a secure authentication method: the ability to easily implement it, the ability to maintain the system, the portability and finally, the scalability.

	Implementation	Confidentiality	Security	Scalability	Total
Django	5	4	5	4	**18**
Database	2	3	1	4	**9**
Firebase	5	4	4	4	**17**

Table 5. Evaluation of various authentication methods and security systems. The scale is 1-5, where 5 represents the best. The total score is out of 20 points.

Based on the analysis above, it seems that the best option for our client and our project would be to use Django, however, Firebase is an extremely viable alternative and we would be able to use either if need be. Referencing the table, the scores clearly indicate that Django is much more advantageous and practical for our wellness platform than creating our own database. It is a major benefit to have a framework provided to us that does everything we want it to; this saves time, resources, and energy by not starting from scratch.

3.5.6 Proving Feasibility

Because Django is specifically designed with the thought of easy implementation, it is exciting to know that we will be able to use this framework. The components are already built for us, and we essentially just need to choose and manipulate these modules to fit our specific needs. It is open-source and free to use, and we can create a development environment further down the line in which to test it out before implementing it into our final product. Because it is built to be implemented into other systems, it allows us the most customization and personalization as well.

4 Technology Integration

Now that we have discussed the details of our technology plans, we must find a way to merge all of these solutions into a single, functioning wellness platform. We want to make the interface as easy to navigate as possible for the user, as well as keep our platform secure, organized, and efficient. Our ideal solution incorporates a secure authentication system, a powerful API, an easily managed database system, and a visually appealing and easy to navigate GUI to most effectively accomplish data analysis and visualization.

We believe that the technologies we have chosen will provide us with the tools necessary to feasibly accomplish our goals for this project. Below we have created a diagram of how our technologies will work together and solve our clients' problems.

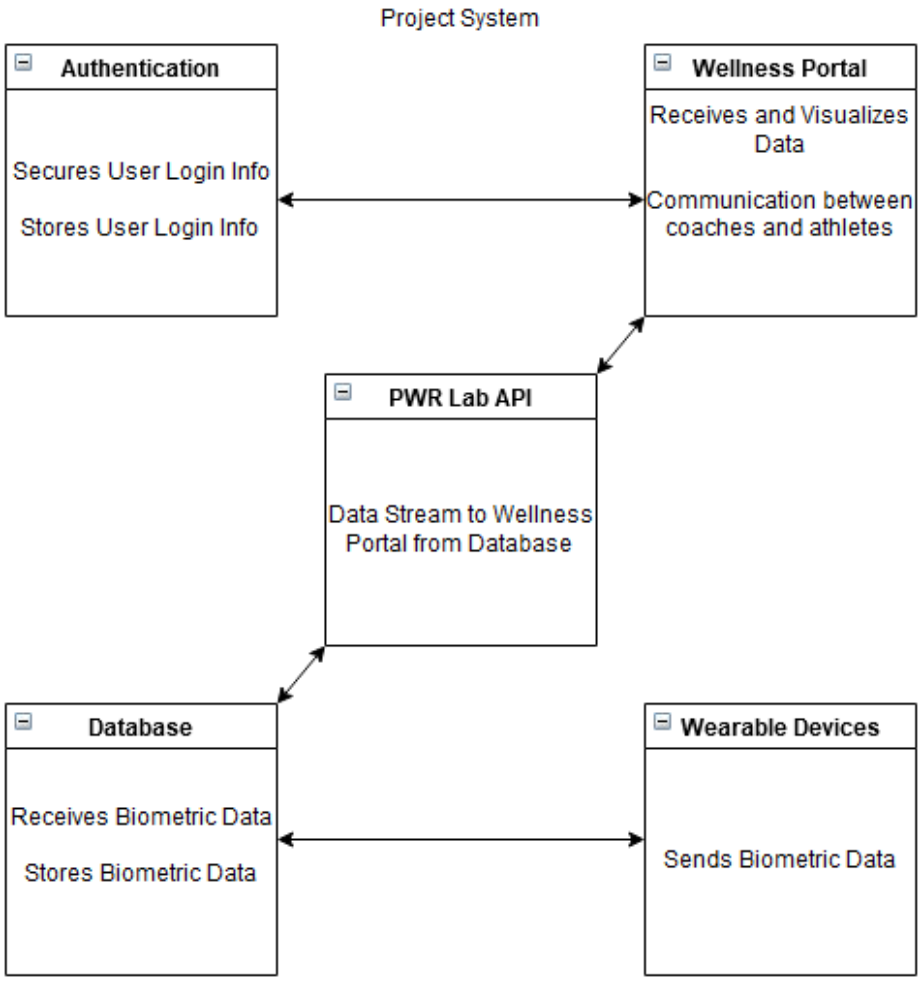


Figure 2. The Project System connects technologies and displays how they'll interact with each other.

As shown from the visual above our plan consists of creating a wellness portal run on AWS that uses our proposed framework . Data will be collected through wearable devices sent to the AWS database and then relocated to our wellness portal through PWR Labs API. The wellness portal will then use Highcharts to visualize the data and display it on the website for the coaches and serve as a communication platform between the coaching staff and their respective athletes. Additionally coaching staff will be able to organize and prioritize data that they feel will be most relevant to a specific athlete. On top of that coaches will be able to comment on certain data or diet updates from the athletes. Site Users will be required to create accounts to view their specific information using Django. This will create a secure environment for staff and athletes alike. We will be using the likes of PostgreSQL to hold and distribute biometrics data in an organized manner. Overall, we believe the framework and data visualization will provide the best flexibility in terms of data representation, our authentication and database system choices will keep our users information and biometrics secure, and our framework will also provide us with a web portal that is mobile browser friendly.

5 Conclusion

Biometric data offers a lot of potential in both an athletic sense and for professional use in a number of different fields. For athletes and coaches, biometric data can give insight into performance and health in a much more detailed sense. Compared to simply observing and reacting to changes in health, PWR Lab helps coaches develop dynamic training plans to help prevent athlete injury by consistently collecting data that can be used to assist in making decisions regarding changes in workout routines. Through an API they developed, PWR Lab successfully captures anything from changes in heart rate, V02 max, pace differences, or any other measurements that are tracked using biometric devices. Implemented alongside the web portal outlined in this feasibility document, our client's API would be capable of further facilitating the interpretation of biometric data and the promotion of wellness for users. Overall, the primary purpose for developing this web portal is to provide ease of access in visualizing data that can be tracked by wearable technology. Specifically, this portal is to be developed with the intention that it can be used by anyone that requires, or is interested in, visualizing biometric measurements in an easily understandable format.

This technical feasibility document lays out a framework for how we intend to develop such a web portal. Through creating and implementing the aforementioned functionalities, it is the team's goal to produce a final product that overall can: easily display biometric data, provide a wellness communication platform, and be highly customizable to suit user's needs. In summary, we believe that implementing an AWS with a Vue framework will allow for easy and highly functional development. For security purposes, we intend to implement a Django to configure user access and viewing of the information in the database. Lastly, we believe that the most functional data visualization API would be highcharts.js. The details on our decisions for each of the major components of the web portal are outlined above and were determined through a careful analysis of the key characteristics in the functionality of each component (i.e. speed of rendering for data, security aspect for user information, etc.). Overall, we have considered a number of potential tools to use in each portion of the development process and believe that the solution provided in this document is both viable and will be functional alongside the technologies already used by the client.

6 References

[1] Yale Medicine - Running Injuries
https://www.yalemedicine.org/conditions/running-injury

[2] MySql
https://www.mysql.com/

[3] MariaDB
https://mariadb.org/

[4] PostgreSQL
https://www.postgresql.org/

[5 Redis
]https://redis.io/

[6] MongoDB
https://www.mongodb.com/

[7] AWS
https://aws.amazon.com/

[8] VMware
https://www.vmware.com/

[9] DigitalOcean
https://www.digitalocean.com/

[10] React.js
https://reactjs.org/

[11] Angular.js
https://angular.io/

[12] Vue.js
https://vuejs.org/

[13] Python - Create a Dashboard
https://python.plainenglish.io/create-a-simple-covid-19-dashboard-with-flask-plotly-altair-chart-js-and-adminlte-a92ef86a3ca8

[14] Chart.js
https://www.chartjs.org/

[15] Plotly.js
https://plotly.com/javascript/

[16] ApexCharts.js
https://apexcharts.com/

[17] HighCharts.js
https://www.highcharts.com/

[18]Django- Django Authentication
https://docs.djangoproject.com/en/3.2/topics/auth/

[19]Firebase - Firebase Authentication
https://firebase.google.com/docs/auth