Technological Feasibility Analysis

9 October 2020

Team DigiFolio
Dr. Andy Wang
Fabio Santos

Burbank, Logan(lead)
Braudaway, Jackson
Chen, Kailin
Marschel, Parker
Yang, Zhenyu

Table of Contents

1. Introduction

Every year, millions of college students graduate with a degree, with their next step being to find a job. Surprisingly, there is a large majority of these graduates who have no prior work experience in their field, such as an internship. This lack of experience often leads to trouble with finding a job, as many organizations are looking for employees who understand the work environment already. Ultimately, this leaves hundreds of thousands of students desperate for a position in their field, possibly not even finding a job for months.

Currently, universities are making an effort to provide resources for their students that allow them to find internships. This is often done through online programs, such as Handshake, or through on-campus events, like career fairs. The two biggest problems with these programs, however, is that they are voluntary, and the university is not able to see how many students are successfully getting internships. The current method of tracking how many students have achieved an internship or work experience is typically through exit surveys. These are given to students right before they graduate, and while it does give the university an idea of how many students have experience, it is usually found out too late to help the graduating class. As of right now, there is no simple way being used to track the students' career paths throughout their college careers. This is where our solution comes in.

We have come up with a solution to this problem, with the end product being a web application. This web application will be capable of pulling career information off of student LinkedIn accounts, and compiling the data into a simple-to-use site. This will allow for students to pull up a profile page highlighting their career milestones. This information can then be used to help students create resumes, or to motivate them to gain more experience when they see an empty page. The application will also allow faculty to pull up statistics on how many students have specific experiences, such as an internship, so the faculty can track student success throughout their college careers, and not just as the students are leaving. Once this tracking is achieved, the university can see what goals of theirs they are hitting, and which goals they need to push more.

To summarize, the problem our client's business is running into involves:

- A lack of student career tracking
- A lack of academic goal tracking
- A lack of passive career assistance

As a solution, we plan to create a web application with the ability to:

- Track student career paths through a semi-automated system
- Track percentages of students achieving specific academic goals
- Allow for students to use a profile as a resume of sorts

At this early stage of the project, we are in the process of analyzing the key technological challenges we expect to face, as well as the possible choices for solutions to these challenges. In this Technological Feasibility Analysis document, we begin by analyzing the major technological challenges we expect to encounter, by first breaking down what challenges we envision, and then following this up with an in-depth look at possible alternatives as a solution to each challenge mentioned. Following these deeper dives into the challenges, we will bring all of the ideas and solutions together and envision how they will work as one system. With that, we once again begin by looking at the technological challenges we expect to face.

2. Technological Challenges

Following the details surrounding the problem and solution, we are figuring out the many different challenges and requirements we will have to overcome. These challenges can be broken into multiple parts that will ultimately come together to create the system as a whole. Among these high level requirements are:

- We need secure user authorization and accounts for user specific access.
- We need a login page that will securely allow students to sign into their own account.
- We need role based permissions, to allow for students, administrators, and alumni.
- We need to implement a strategic goal manager, as this will allow the website to keep track of key performance indicators.
- We need to create a digital dashboard that will display graphics based on input statistics, which will come from our database.

Among these challenges, we have broken them up into the technology decisions and obstacles we are going to have to make in order to find solutions. In doing so, we have come up with the following technology decisions:

- Database software: Which database management system we will use
- Server location: Where our web application will be implemented and run
- Programming language: Which language we will use for the core of our application
- Dashboard generation tools: Which libraries we will use to generate a graphical dashboard
- Web framework: Which framework we will implement for our application

In the following section, we will analyze the possible alternatives to each of these decisions, and come up with a strategy to decide which alternative will be the best solution for our product.

3. Technological Analysis

In order to gain a better understanding of the project, we will now go into analyzing the different technological obstacles that we expect to encounter. We will go over each problem, and decide possible alternative solutions to help us get past the obstacle. We will end each subsection with ranking the different alternatives using a scoring system, allowing us to decide on the best solution out of all of the possibilities for each obstacle.

3.1 Database Software

One of the challenges we expect to come across is the issue of choosing which database software to use for our web application. There are many different types of databases, as well as variations on the specific types, that all serve the same purpose with different characteristics. Ultimately, the database we do choose will be the backbone of the whole application, as the purpose of the application is to display different pieces of data to the users.

Desired Characteristics
Read speed is one of the most important characteristics, as our web application will be heavily focused on users reading data from the database, while not necessarily writing to the database or editing the data often. This is especially true when a page is grabbing multiple pieces of information connected to one unique entity.

Write speed is another important characteristic as the web application will need to write data from the different sources it will be pulling information from. While this is not as crucial as read speed, it is still important since users will expect their data to be consistent throughout the site, as well as up to date. Write speed can cause problems with consistency and out of date information, as it may take time to transfer data.

Ease of maintainability is another important characteristic to look at, as the database is expected to be running for years. This means there will likely be maintenance on the database consistently, meaning a database using less data duplication could be more useful as it will not create issues when modifying data to correct errors in the data.

Reliability is another key characteristic to look for, as the database is expected to be up constantly. This is not as important as the other characteristics, necessarily, as the information on the server will typically not be needed in an urgent manner. However, with students accessing the site from many locations and timeframes, it will be helpful to them for the server to be up as often as possible.

Scalability is yet another characteristic to consider, as the database will need to be able to consistently grow as new groups of students are added in. Typically, as alumni will still be in the system and able to access it, this will mean more capacity being needed on a yearly basis without any previous data being overwritten.

Finally, language integration compatibility is something to look for. While this does not matter as much, as most database models will have some form of integration in most languages, it is helpful to know what is exactly possible with specific software that implements these models.

Alternatives

MySQL is a relational database management system that allows for a database to be set up using tables. We have come across this system through other classes of ours, and the relational model allows the SQL language to be used through multiple languages. [1]

MongoDB is a document store database management system that we have found through online forums. Looking into multiple programs in use today using MongoDB, it is clear that it can allow for a flexible database due to its lack of enforced schema. [2]

Cassandra is a wide column store database management system. We found this while looking into MongoDB's reliability, as both types are NoSQL and Cassandra is a different implementation with better reliability. It also helps that Cassandra is one of the most widely used systems for wide column store databases. [2] [3]

Analysis

Characteristic	**Weight**	**MySQL**	**MongoDB**	**Cassandra**
Read speed	.40	1	3	2
Write speed	.20	3	1	2
Ease of maintainability	.10	3	1	2
Reliability	.10	2	2	3
Scalability	.15	2	2	3
Language compatibility	.05	2	3	1
	TOTALS	**1.90**	**2.15**	**2.20**

Chosen Approach
The table shown is a summary of the final evaluation of each software compared to each criteria. The rankings have been decided based on how well the software handles the criteria compared to the other options, having a best = 3, average = 2, and worst = 1. This does not necessarily mean a 1 is terrible at the criteria, just that it is considerably worse than the other options. The weights were also noted, and the total for each software involves multiplying each score by the criteria weight, and adding these products for the software. Using this table, we are deciding to work with Cassandra, as that has the best overall performance for our needs.

Proving Feasibility
In choosing Cassandra as the database software to use, we will look more into testing it to ensure it will work given our program. This testing for the time will include creating a limited (< 10 student records) version of the envisioned database, including information for student pages, as well as the ability for the data to be queried through our code.

3.2 Server Location

When our project gets closer to being complete, a concern we will have to face is where to get it running and keep it running. This gives us the challenge of deciding what cloud services are going to be used. This will play a huge role in what could make or break our project. The concerns with what cloud service our client wants to use comes with include price, servicing and maintenance of the service, storage size and more. [4]

Desired Characteristics
Cost is important for our project because if we have to buy a service, then we need the appropriate budget in order to pay for it. One of these problems we need to think about is that our client might only have a limited budget for paying for services, which includes keeping the project we are creating up and running. Without the ability to pay for a cloud service, in the long term, there would be no way to apply the applications of our project, which is why knowing a budget is important.

Another problem that we will potentially encounter is ease of maintenance. Some cloud service providers have blackout dates. These blackout dates are days that their services are not usable. This means that the website we intend to create could be down for an unknown number of days, depending on the service provider [6]. Our ideal cloud service would have a limited number of days of maintenance. Potentially the service provider would have maintenance days on days NAU could be on break, whether winter break, spring break, or summer break.

Additionally, security of cloud services is very important. The ability to know how safe the data we store is important. Many of these service providers are well known and speak through their reputation. If one of these providers are as popular as people say, then their security should back that up.

Lastly, a problem we wish to assess is the ability to have a backup. Many service providers offer a backup service but the question is how quickly the backup takes and to what extent the backup option is given. Ideally, the service we choose to go with would backup our data everyday and have a fairly quick process in doing so.

Overall, these desired characteristics are what our team is going to look for in a cloud service provider.

Alternatives

One of the options in choosing a cloud service provider is Microsoft Azure. Microsoft Azure was released nearly a decade ago, in 2010 [4]. Azure means that there are no physical servers on site which reduces costs for an onsite support team. Azure also offers 12 months free of their service platform and many options of payment, even a pay-as-you-go plan.

Another option of a cloud service provider is Amazon Web services. Amazon web services are highly customizable with a lot of choices of servers. Amazon also includes free trials as well as free tiers for their servers as well as multiple payment options for their services. [5]

A final option that we are looking into are the NAU's servers. We do not have much information on what NAU could provide for us. However, since NAU already runs websites and has servers for storage, our team believes we might be able to run this site through NAU servers. We are working with our client, Dr. Andy Wang, to see if this is a possible solution.

Overall, there are tons of options that we have to weigh in order to make a decision.

Analysis

Characteristic	**Weight**	**Microsoft Azure**	**AWS**	**NAU Servers**
Cost	.50	1	2	3
Ease of Maintenance	.20	2	2	3
Backup	.15	2	3	N/A
Security	.15	3	3	N/A
	TOTALS	**2.15**	**2.80**	**~3.00**

Chosen Approach
The table shown is a summary of the final evaluation of each cloud service provider compared to each criteria. The rankings have been decided based on how well the software handles the criteria compared to the other options, having a best = 3, average = 2, and worst = 1. This does not mean that a 1 is terrible at the criteria, just that it is considerably worse than the other options. The weights were also noted, and the total for each software involves multiplying each score by the criteria weight, and adding these products for the software. Using this table, we are deciding to work with the services that NAU can provide for capstone projects. This will be the overall cheapest and easiest option to work with. Since NAU has this option for capstone services we are going to research more on what options NAU can provide in giving us access to a server for our project.

Proving Feasibility
In further testing the NAU servers, we will first be contacting ITS services to obtain a better understanding of how to set up and use these. We will test the services based on the desired requirements that gave us this choice in the first place. Testing will further include on how to set up our servers, backing up our servers, and explore our long-term upkeep options. Overall, testing will take time to make sure this is our best option.

3.3 Programming Language

A fundamental challenge is determining the language we will use to develop our project. Our client has expressed his desire to use whatever language we would like to use. This greatly expands the range of our potential candidate languages.

Desired Characteristics
One important characteristic is the familiarity our team has with the language. The more familiar we are already with a specific language, the easier it can be to integrate our system using it.

Another important characteristic is the package availability for each language. As each language has different provided tools through the packages, it is important to know which ones each language provides.

Server compatibility is another characteristic we are looking for, as we need the language to work for front and back end. When looking into this, it is also important to consider how flexible the compatibility is.

Alternatives

Python is one of the most popular languages among developers. Python has seen a surge of popularity in recent years. Due to its popularity, it still has support for a lot of platforms. Python is a dynamically typed language and offers a good amount of support for mathematical and statistical operations. It means that design restrictions could possibly occur which could require more testing time. [7]

Java is the most common runtime environment that enterprises use. It was released in 1995 and since then has morphed into one of the premier server side back-end development languages. Ironically, it was not designed with this in mind, as at the time the internet looked much different than it does today. Java is object oriented and is used by around 9 million web applications. [8]

JavaScript is a programming language which can implement complex functions on web pages. What web pages show you is no longer simple static information, but real-time content updates, interactive maps, 2D/3D animations, scrolling video and so on. [9]

Analysis

Characteristic	**Weight**	**Python**	**Java**	**JavaScript**
Familiarity	.30	1	1	3
Package Availability	.35	3	3	3
Server Compatibility	.35	1	1	3
	TOTALS	**1.70**	**1.70**	**3.00**

Chosen Approach

We have chosen to work with JavaScript for our project. There are few reasons for our group to choose JavaScript as the primary programming language. Firstly, it is simple, versatile, client-side based, and most importantly, it is compatible with a wide variety of programming. Most of us have the experience in using JavaScript. Because of the compatibility, we don't need to spend a lot of time learning some new programming. Secondly, it is a single thread, which means that we can avoid various problems caused by managing multiple threads inside the program.

Proving Feasibility

We plan to use JavaScript as our programming language. In the process of programming, we will test code frequently in small pieces, so that we can find errors as soon as possible. JavaScript is a scripting language depending on the programmer, and we will do our best to keep good practices in use.

3.4 Dashboard Generation Tools

When we need an information management tool that can visually track, analyze, and display key performance indicators (KPI), metrics and data points to monitor the health of a business, department, or specific process, we should use a library tool to generate a dashboard. The challenge is which tool we are going to choose. [10]

Desired Characteristics
Access is one of the important characteristics for the tool. The tool should provide high-speed access and real-time updates. Users need full functionality and mobile-friendly design to take full advantage of the tool. It is important to make sure that our tool is portable and accessible on all necessary devices.

Data management is the most important characteristic of all the work we want to accomplish with the tool. We need to make sure that the tool we choose supports access to multiple data sources, including data warehouses, internal databases, clouds, and data marts. Depending on the different sources we used, our tool may also need to cleanse and transform data for proper use within its system.

Data visualization makes our analysis animated. We need to make sure that the dashboard is configurable and contains dynamic real-time information. Some tools provide many graphical options and interactivity, such as drill-down capabilities. We will need to be able to visually demonstrate the features of influence points, and we also seek flexible graphical visualization features (including heat maps, geographic maps, etc., and a variety of chart styles).

Usability is a key success factor despite it often not getting the attention it deserves. Albeit the ease of use, visual appeal, and intuitiveness seem great, for users, these nuances can make a difference between successfully adopting users refusing to use the tool. [11]

Alternatives
FusionCharts is one of the most widely used charting libraries written in JavaScript, it can be installed via CDN, npm, or locally. This tool is very easy to learn how to use, it has comprehensive documentation for each library with lots of live examples. It also has great configurability, which includes over 100 charts and 2000 maps.

Mozaik is a Node.js-based dashboard tool used to create dashboards that can be defined and built using relatively simple JavaScript configuration settings. It is designed to be extensible and scalable, its layouts work well on different size devices due to its responsive HTML design. Mozaïk provides a simple way to define the dashboard layout using a grid system. It also has optimized backed communication, most extensions need to communicate with APIs, this tool

eases this by providing a backend, which handles API calls and pushes data to widgets through WebSockets.

FreeBoard is a dashboard tool designed for simplicity and ease of use. Its JavaScript system has drag-and-drop functionality, and new data sources can be added without any programming knowledge. It is designed for IoT use and can be used to create attractive dashboards for any purpose. If the user is not familiar with creating dashboards and wants to get started quickly, it is one of the best choices. It offers both free and paid plans, allowing users to create dashboards using open source tools without any installation.

TipBoard is a dashboard creation tool written in JavaScript and Python. It has various widgets that are completely separated from the data source, which provides great flexibility and relatively high customization possibilities. It is open-source, so it is possible to contribute (on Github) and share your own widgets with the community.

Analysis

Characteristic	**Weight**	**FusionCharts**	**Mozaïk**	**FreeBoard**	**TipBoard**
Access	.15	3	3	3	2
Data Management	.40	3	3	3	2
Data Visualization	.20	3	2	2	3
Usability	.25	3	2	3	2
	TOTALS	**3.00**	**2.55**	**2.8**	**2.2**

Chosen Approach

The table shown is a summary of the final evaluation of each library tool compared to each criterion. The rankings have been decided based on how well the software handles the criteria compared to the other options, having a best = 3, average = 2, and worst = 1. This does not necessarily mean a 1 is terrible at the criteria, just that it is considerably worse than the other options. The weights were also noted, and the total for each software involves multiplying each score by the criteria weight, and adding these products for the software. Using this table, we are deciding to work with FusionCharts, as that has the best overall performance for our needs.

Proving Feasibility

In choosing FusionCharts as the library tool to generate dashboards, we will look more into testing it to ensure it will work given our program. This testing for the time will include creating a limited version of the envisioned dashboard.

3.5 Web Framework

One of the technological challenges that our team must overcome in developing our product is finding and using a web framework that best suits the product we wish to deliver. This framework will assist in the creation/development and implementation of our web application by reducing the overhead associated with web development by perhaps providing APIs, URL routing operations, user interaction, authentication, etc.

Desired Characteristics
Database integration and support by the framework could play a very crucial role if we decide to implement one later. A framework that focuses on this slightly more than others will help to reduce the overhead that comes along with database management.

The framework should also have API support. Another technological challenge facing our group is utilizing the LinkedIn API. Having a framework that works well with API's will allow us to implement one much easier into our application.

User-friendliness is another characteristic that will reduce a lot of overhead in the development and creation of our product. Being able to understand and use the framework will indeed be a learning process but a framework that is easier to use will allow us to spend less time learning and more time developing.

A programming language that the whole team is familiar with is also desired and equally as important. With all of us being able to use the framework things will be completed at a much quicker rate and as stated previously, the less time we have to spend with something foregin will reduce the overall overhead we endure.

Alternatives
Django follows a model-view-template style and is one of the most popular open source web frameworks available and is compatible with Python, a language we are all familiar with. Another important feature of Django is that one of it's main focusses is making database driven websites easier to develop in comparison to other available frameworks. It also contains several packages to extend it's original behavior, including API provision and consumption. Once again, it is written in python which adds to the user-friendliness of the framework. [12]

Vaadin is another available open-source framework which comes with a set of web components, a Java web framework, and a set of tools and application starters. Vaadin also allows for the implementation of HTML 5 web user interfaces using Java, another language that we should all be comfortable with by now. Also, the product being a very popular tool suggests that it is most

likely user friendly and it is compatible with several IDE's such as visual studio, etc. As previously stated Vaadin comes with several components to assist with things such as UI generation, API services, security, validation, etc. [13]

Codeigniter is based on the model-view-controller development pattern that PHP uses to create dynamic web applications. It comes with tools that can manage and work with databases and make API response traits simpler. PHP is not a language that all of us have too much experience using compared to Java or Python but is not overly complex in it's syntax. [14]

Drupal is a free open-source web framework used all over the world from blogs, government, corporate sites, etc. The framework is written in PHP and provides a sophisticated API for developers. At the same time, web site installation and administration requires no programming skills whatsoever adding to the user-friendliness of the software. Drupal supports both a web server capable of running PHP and a database to store content and configuration. Unfortunately, our team's overall background knowledge with PHP is not as strong as Python or Java but should not be too much overhead to become familiar with if needed. [15]

Analysis

Characteristic	**Weight**	**Django**	**Vaadin**	**Codeigniter**	**Drupal**
Database Integration	.40	3	2	2	3
API Support	.30	2	3	2	3
User-Friendliness	.20	3	3	1	2
Language Familiarity	.10	2	3	1	1
	TOTALS	**2.60**	**2.60**	**1.70**	**2.60**

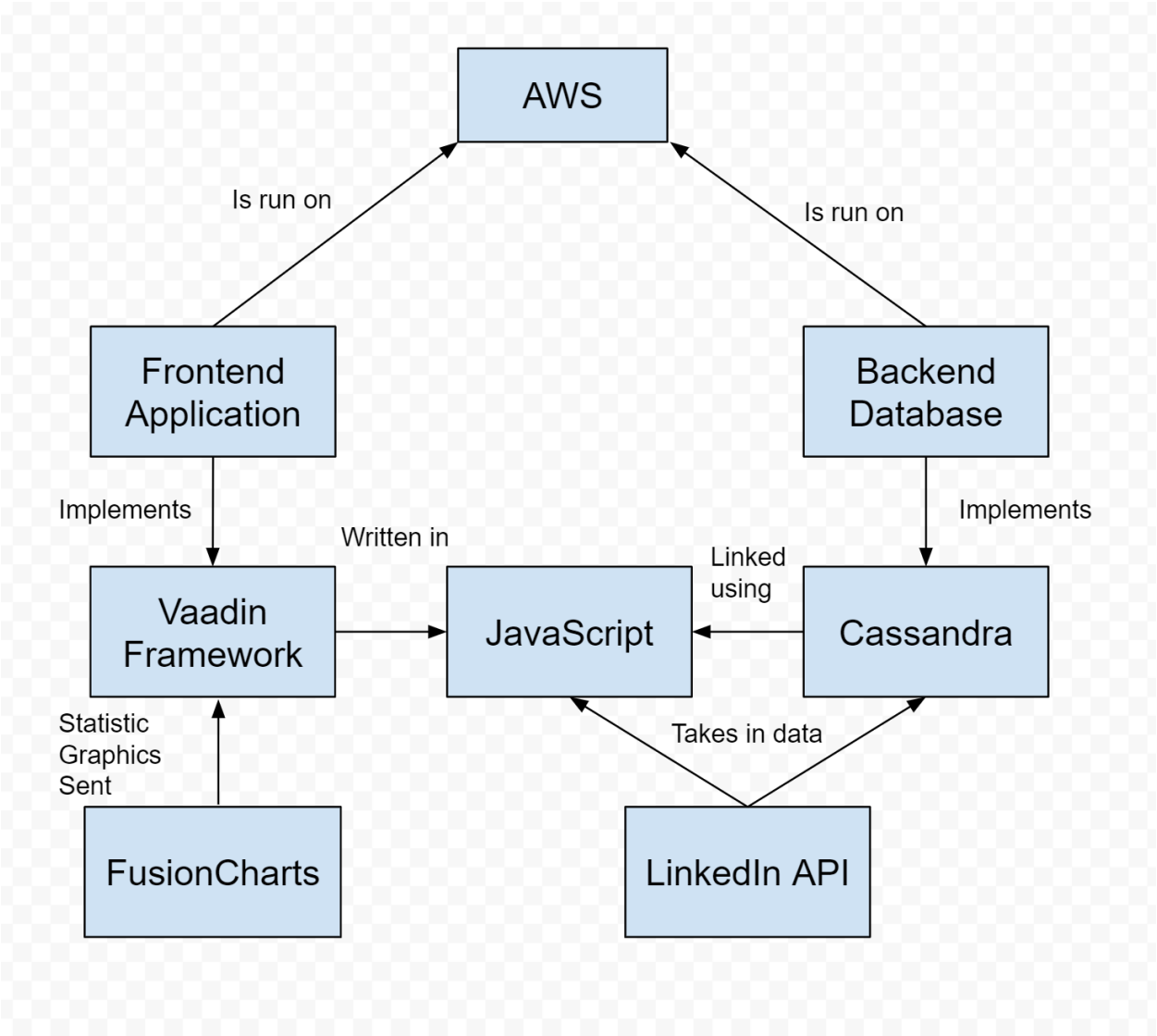
Chosen Approach

Looking into open-source frameworks there are multiple that would be suitable for our product. However, with utilizing one of these frameworks certain characteristics are sought more than others for assistance with the development of our product. Most importantly, having tools for database integration, API use and support, being fairly user friendly, and being written in a programming language we all have prior experience using. With that understanding, and our main language choice being JavaScript, Vaadin seems to be most favorable and advantageous to our product, with the JavaScript integration being the tiebreaker.

Proving Feasibility
In our choosing of Vaadin as our web framework we will create a simple sandbox application to become more familiar with it before diving right into the creation of our product. The sandbox application will help us learn the tools for things such as database integration, etc.

4. Technology Integration

Now that we have come to solutions for our different technological obstacles we intend to encounter, we must look into how the pieces can come together. We need to see if all of the best solutions can be implemented to work as one system, or if we will need to possibly use a second-best alternative in some spots.



In the figure, we look at how each part will work with one another. The different pieces will all have different interactions, and it is important to outline and go over each connection.

To begin, the core of the whole system is AWS, or Amazon Web Services. This is because when the project is complete, everything will be implemented onto AWS. This includes both our frontend web application, as well as the backend database supporting the application.

The frontend application will be created by implementing the Vaadin framework. This framework will provide the architecture for our frontend interface that users will interact with directly.

The Vaadin framework will process information and provide graphical dashboards for the users through the use of FusionCharts. These tools, as well as Vaadin, are able to be implemented

through JavaScript. Vaadin is mainly through Java, but there are ways to allow it to interact with JavaScript.

The backend database will be implemented through the Cassandra database management system. This system will need to be able to work on Amazon Web Services, and there is a system called Amazon Keyspaces that is made specifically to allow this integration.

Cassandra is also able to be implemented through JavaScript using tools through Node.js. This will allow us to query the Cassandra database using Javascript.

Finally, the LinkedIn API does have an integration tool for JavaScript. This will allow us to take in data from LinkedIn to populate our database with, as both the database and the API are able to use JavaScript.

5. Conclusion

In researching these different options we hope to obtain the best possible technologies in order to create the best possible solution for our Engineering Career Builder (ECB). This ECB will overall create a digital portfolio for students at NAU and will be able to show the progression of students' knowledge throughout their years here at NAU. Throughout our early stages of our project, we processed and analyzed the key technological challenges we are expected to face. In addition to these challenges, we compared and came up with the best solutions to these challenges. In this Technological Feasibility Analysis document, we began by analyzing the major technological challenges we expect to encounter, by first breaking down what challenges we envision, and then following this up with an in depth look at possible alternatives as a solution to each challenge mentioned. Following these deeper dives into the challenges, we brought all of these ideas and solutions together and gave our best envisionment of how they will work as one system. Going forward in our project, we will be able to make the best possible solution with the best technologies through our research.

6. References

1. *MySQL Database* https://www.w3schools.com/php/php_mysql_intro.asp
2. Simic, Sofia. *MongoDB vs Cassandra* https://phoenixnap.com/kb/cassandra-vs-mongodb
3. *What is Cassandra?* https://www.datastax.com/cassandra
4. *9 Things to Look Out for While Choosing a Cloud Service Provider* https://www.acecloudhosting.com/blog/choosing-cloud-service-provider/
5. *Amazon S3* https://aws.amazon.com/s3/?did=ft_card&trk=ft_card
6. *Best Cloud Computing Services of 2020* https://www.techradar.com/best/best-cloud-computing-services
7. Solutions, Mindfire. *Advantages and Disadvantages of Python Programming Language* https://medium.com/@mindfiresolutions.usa/advantages-and-disadvantages-of-python-programming-language-fd0b394f2121

8. *History of Java Programming Language* https://www.freejavaguide.com/history.html
9. *Why Developers like JavaScript* stackshare.io/javascript
10. *What is a data dashboard?* https://www.klipfolio.com/resources/articles/what-is-data-dashboard#:~:text=A%20data%20dashboard%20is%20an,of%20a%20department%20and%20company. (10/19/2020 19:54)
11. Silva, T.. *5 Expert tips for selecting the right business intelligence dashboard tool* https://www.clicdata.com/blog/5-expert-tips-for-selecting-the-right-business-intelligence-dashboard-tool/ (10/19/2020 20:05)
12. *Django, The Web framework for perfectionists with deadlines* https://www.djangoproject.com/
13. *Vaadin, Full stack framework for building web apps in Java* https://vaadin.com/
14. *CodeIgniter* https://codeigniter.com/
15. *Drupal* https://pantheon.io/resources/drupal-8-technical-sheet?campaignid=1335635982&adgroupid=85951675374&adid=438751986903&https://pantheon.io/resources/drupal-8-technical-sheet&_bt=438751986903&_bk=%2Bdrupal%20%2Bwebsite&_bm=b&_bn=g&_bg=85951675374&gclid=Cj0KCQjw28T8BRDbARIsAEOMBcwtTWKfwtH7tcY1Ba3g7PjhIjgCKc8C94pqxDmsgQF2zSHA01mvXX4aAqVAEALw_wcB