**Technical Feasibility Draft**

**October 23, 2022**

**Team Teacher To-Do**

**Project Sponsor:** Chris Aungst

**Faculty Mentor:** Italo Santos

**Professor:** Michael Leverington

**Team Members:**

Sam Gerstner (Team Lead)

Alexander Frenette

Noah Nannen

Shlok Sheth

Bronwyn Wedig

**TABLE OF CONTENTS**

Introduction 3

The Problem

Technological Challenges 4

User Privacy & FERPA Compliance

Technological Analysis 5

MVP Tasks & Goals

Language and Frameworks

APIs & Data Access

Authentication

Database and Document Storage

User Interface & User Experience

Technology Integration 11

Conclusion 13

**Introduction**

The purpose of this technical feasibility document is to outline the high-level project requirements, technologies to be used, as well as the technical feasibility and any foreseeable problems with any project requirement or chosen product or technology. This document aims to provide an in-depth analysis of all MVP requirements and the technologies and platforms that will be used to help us accomplish our goals.

This document will be used as a reference point for the remainder of the project and will be one of the primary documents that guides our development and decision-making processes. One of the primary goals of this document is to outline potential problems, their solutions, and to help all stakeholders gain a better understanding of roadblocks that may be encountered during the development process.

**The Problem**

Arizona is currently facing a massive teacher shortage in our state. To help address this issue, the Arizona Department of Education has introduced a new program that will allow student teachers in their final semester of their undergraduate degree to fill some of these vacant teaching positions. As you can imagine, this process involves tracking many documents and requirements to ensure students meet the needed criteria. The College of Education currently spends hundreds of hours per semester managing these requirements, and they would like web-based applications that can help them manage these requirements. Our proposed solution is to create this web-based application that will not only help administrators manage these requirements, but also help students understand the process that they will be following and the requirements that they will be completing.

**Technological Challenges**

For our project, most of the potential problems stem from us being unable to gain access to the student database for resources. If we are unable to use real student data, we will have to try and spoof our data set and test from there, which may lead to inaccuracies in the program’s data storage system. We would also not be able to successfully implement the system, meaning that we would fully create the project that we are able, and then it would most likely sit on the sidelines for a few years until ITS gets the chance to modify it to work within the current system. We have done all we can to minimize this risk, even going as far as to become FERPA certified to be able to responsibly handle student data, however, it is ultimately a choice left up to ITS on whether they see it as to great a risk on their part. We do have an alternative option to directly importing `student data, which is to have students enter in via text and Boolean checks into the system, and then have an administrator verify the data, however, that comes dangerously close to the manual-input system the College of Education already has in place, and may not win over the administrators by having them learn an entirely new system that does essentially the same thing as the system that is already in place.

**User Privacy & FERPA Compliance**

Due to the requirements of FERPA, we will have to take great care to ensure the privacy of student records. As noted, before we will not be storing student permanent records. Still, there will be identifiable information about students that we need to take care to secure and anonymize. In order to uniquely identify students, we will have to record an identifying piece of information. In case of a data breach, we will be implementing secure hashing to safeguard against deanonymization. Hashing tool algorithms such as MD5 have been compromised, leaving only SHA and its variants. Due to the known format of NAU emails, a hash with only that will lead possible combinations, which would be cracked in moments. To increase entropy, we will need additional information specific to students such as id numbers. ID numbers being slightly more private.

**Technology Analysis**

**MVP Tasks & Goals**

Our minimum goal for the project would be to at least give the student their overall progress for fulfilling all the requirements and steps to achieve other requirements. In simpler terms, we could describe the website as, A professionally modeled website that imports the student information from their NAU login using CAS. The student can track and follow through their career path for their Student Teacher Intern Certificate (STIC).

STIC has a hefty list of requirements a student has to fulfill, hence the website would give the student a gist of that. The model will also represent the requirements that are completed by the student. It has a dashboard where the student can see their overall progress for the program. The website will also provide a feature to share, and upload signed documents on the server. The main objective of the website is to help the student be on the path for the STIC plan and give a user-friendly insight into their overall standing making it easier and less time-consuming for the student to track their progress. The website will create a path for the student and give the student a plan to follow through in order to fulfill the requirements. We also plan to accommodate the website by importing the NES exam scores from the Pearson website which plays a vital role in making the student eligible for the certificate program.

The website will also hold the facility to store the signed documents and send them over email for signature. Although this would be way down on our list of priorities for the website features. Instead, we would focus more on making the website very convenient and professional as we look forward to using it as one of the subsidiaries of the nau.edu domain. The website will be able to give permissions and access to its data to an official from the College of Education administrator, and our client Christopher August who is responsible for handling the STIC requirements.

For the backend part of the website, we would like to structure it in a way that it will import data from FileMaker Pro software that will use an application program interface, it will use the student user id from the database. As specified by Dr. Leverington, the minimum requirements for our data-sensitive website should be that it should be able to create a hashed value for the student user id in order to make the website secure and then search the student’s delicate information in the FileMaker Pro database. After finding the student’s information from the hashed value, it will perform the processing inside FileMaker Pro and return the information stored in the data structure with the hashed id of the student so that there is no sharing of sensitive data between the website and the FileMaker Pro application. In layman’s terms, it does all the processing inside the file maker application and does not return any FERPA-related information to the website hence one of our important goals would be to create a very strong hashing function.

**Languages and Frameworks**

One of the first challenges we faced with this project when coming up with our development plan, was what programming language we should use. The languages we considered using included Java, C#, AngularJS, & NodeJS. All these languages are capable of creating a web application, but the non-JavaScript languages had some clear advantages. Both Java and C# are popular choices for enterprise applications, but Java has a much larger market share and has more documentation available. Another advantage of Java & C# over the JavaScript based languages is that Java and C# both support the Model-View-Controller (MVC) design paradigm. By using MVC, we gain greater control over how we can assign development tasks to our team members.

After narrowing down our choices to either Java or C#, we started discussing within our groups about our previous programming experience and what languages we were familiar with. After some discussion, we found that all of our team members had previously used Java in either personal projects, or educational projects.

Ultimately, we decided to use Java for this project because of it’s wide use in enterprise applications, extensive documentation, long-term support, familiarity among our team members, and the wide variety of tools available to extend Java’s functionality. Java is an extremely popular choice for enterprise web applications because it is platform independent, has built-in memory management, is very cost-effective, and is easily scalable. By using Java to develop this application, we are also conforming to some industry best-practices by using a widely supported language that is unlikely to go away any time soon.

After deciding on Java for our programming language, we started looking for Java frameworks that could assist us in our development so we could abstract away the need to manage the underlying Java containers and more. For this framework we investigated using Spring Boot, Quarkus, and Micronaut. All these frameworks provide tools on top of the built-in Java functionality that could assist us in developing an enterprise application. Quarkus relies on Kubernetes to function, so it was not a viable long-term solution, because it would require the maintenance of a Kubernetes environment. Spring Boot is the most popular option amongst Java developers and provides extensive documentation and tutorials for new users. Because of these clear advantages, we chose to use Spring Boot for this project.

In order to streamline the development process, we chose to use the Java Spring Boot Framework which provides a wide array of tools for web application development, templating, security, and more. Spring Boot incorporates a style of web application development known as Model-View-Controller or MVC which we will be utilizing for this application. MVC allows for the separation of web and code components, allowing for easy division of development tasks among our team members based on individual’s strengths. By being able to divide tasks based on front-end/back-end development allows us to assign development tasks to the team member whose skills best match the specific task. In most web applications, the front-end and back-end development are very closely coupled, and don’t allow for the easy division of tasks based on team member strengths.

The Spring Boot framework will allow us to write our application in languages that are familiar to us (Java, HTML, CSS, etc.) and easily package our application into a JAR or WAR file that can easily be deployed to an Apache Tomcat web server. By using the Spring Boot tools, we are also ensuring coherency among components of our application, and greatly increasing the likelihood of all application components working together correctly to hopefully minimize the amount of work needed to implement basic components of the web application.

**APIs & Data Access**

Access to data is critical to our app. Due to privacy and security constraints, a great deal of our data will be held externally. The access mechanism provided is a rest API.

With the architecture of our system, we have decided to make the front end and back end loosely coupled. Access to the data should also be platform agnostic. This means that if we wish to swap our front end, the existing back end shall remain. This separation of logic and responsibility, will also allow for the addition of alternate front ends; an example being a mobile app.

Having our front-end logic separate from our backend logic will constitute the need for an access mechanism. Our options are a standardized remote procedure call such as gRPC, a REST API over http, or GraphQL implementation.

A primary concern of ours is the documentation of our data, and the ways in which to access, and modify it. All three options have services to generate such documentation as well as code stubs. OpenAPI for rest, Apollo for GraphQL, and gRPC-Doc for gRPC.

While looking at industry use, gRPC is more focused on communication between two servers, while REST and GraphQL are client service oriented. REST has been selected, as they both systems are equally capable, while our group has more familiarity with REST.

**Authentication**

As they have the need to uniquely identify students, and verify such identity, the need for an authentication system arises. As our system will need to interact with student permanent records, authentication is critical. If our system is to interact with such records, we believe it best that the university authentication system be used within our application as well. Otherwise, what guarantee do we have that a user is the student they claim to be. We would either need to send emails to the students, which would require human interaction or further automated systems. The secondary option would be to allow signups using only student emails which would then receive a confirmation. We would still have the issue of accessing school resources to fetch our data. Due to the necessity of having access to student data we have opted to use the CAS system along with NAU AuthZ for authorization on our servers.

We still have almost no idea how CAS works, and what it will provide, along with what the features of NAU AuthZ are. It looks like AuthZ has coarse grained blocking of pages on Tomcat. We need a more sophisticated system that will allow people to navigate to the same page and receive different results that are specific to their identity. We may have to implement a different authorization system using an identity token after the CAS system has provided user authentication.

**Database and Document Storage**

While we will not be storing student records, we will still have a need to store student information specific to our application. This includes status on tasks as well as specific documents. This leads us to two main options for databases. A simple relational database such as SQL or a document store such as MongoDB. SQL gives the benefit of familiarity but does not allow us to store user documents within the database. Instead, we would have to use an external system, and reference the external id's from within the SQL database or keep the systems semi-disjoint and only reference the database's primary key from within the blob store. MongoDB on the other hand offers the benefit of having a flexible schema, as well as the ability to store large documents from within the database using GridFS. The downside of which is that members of the development team are less familiar with MongoDB.

If a relational database is selected, there remains the need for document storage. While traditional cloud storage providers such as Google Drive and DropBox are an option, a more general blob storage option exists. A blob storage provider gives the ability to store arbitrary data using key value pairs, which would be beneficial. It would allow us to associate users from the database with documents more easily. This instead of using a less easily programmed system of naming documents with prefixes for identification or grouping within folders. As a result, if documents are to ever be stored from outside of the database, a blob storage system will be used. Functionality of said service appears uniform from all major cloud providers, so selection is best determined by our existing providers.

For the sake of familiarity, we have opted for a relational database, and will supplement the system with a blob data store.

**User Interface & User Experience**

For our user interface, we had to decide on what sort of style to go with to ensure that the websites generated for the project were aesthetically pleasing as well as useful to the users that would be accessing them. We already knew that we would have to use HTML to create the website, but the challenge arose when we needed to decide on a style to use. We ended up with 3 possible options for our styling. The first was Bootstrap, which allows for higher functionality and integration of complex tools into a site, as well as having complete access to whatever else needs to be added on in terms of tools and plugins. The second option is Bulma, a native HTML import that also allows for extended use and additional plugins to be added in the future, but less focus on integration of some of the higher end tools we were planning on using, like Springboot. The last option was Custom CSS styling, which is just using CSS stylesheets for each webpage. This is incredibly clunky and creating stylish webpages is very difficult to accomplish without large amounts of dedication, but it is still one of our options if one of the other two exclude something vital to the project.

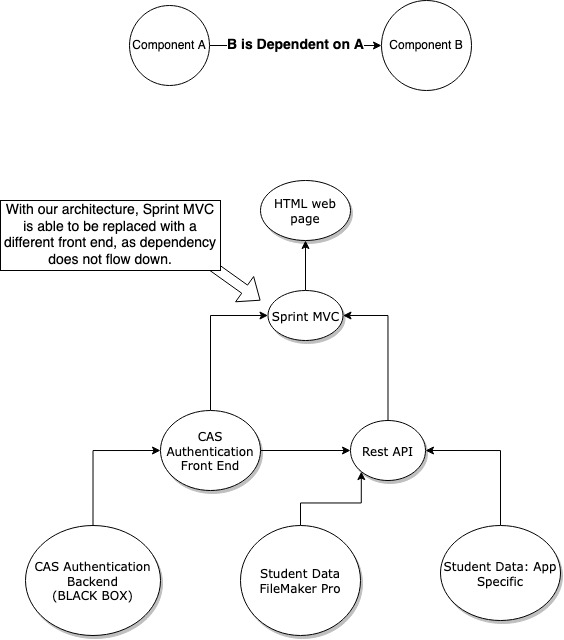
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Language Ratings** | Industry Adoption | Ease of Use | Components Included | Development Speed | Project Support | Total Score |
| Bootstrap | 4 | 5 | 5 | 4 | 5 | 23 |
| Bulma | 3 | 4 | 4 | 2 | 3 | 16 |
| Custom CSS | 5 | 1 | 1 | 2 | 2 | 11 |

We chose to use Bootstrap not only because of its high functionality or inclusion of other project types, but also because NAU already has a custom Bootstrap that we can use to ensure that the website can be recognized as an NAU webpage and has all of the functionality of a NAU website built in.

**Technology Integration**

During the analysis of our technical challenges we made sure to select technologies and design philosophies that would allow us to componentize our system. Each part of our system is able to be developed by separate teams, and have minimal dependency relations on other components. Broadly speaking our system can be broken up into two primary parts, front-end and back-end. The back-end is accessed through our REST API, which will be designed such that it is loosely coupled from any front-end architecture. This will allow future developers to replace the front-end system, or create additional views of our back-end system such as a mobile app.

Right now, most of our focus is into figuring out two main components of our application; CAS and the data structure of FileMaker Pro. As we are in the process of getting access to the FileMaker Pro application, we are unaware of the schema of the data, and the structure of the data and how it is being saved on the server. The closest we have been to that data is that FileMaker Pro can export all the data to an excel file in a form of CSV file, which is nothing but columns and rows of data, worst case scenario we could use this file to process our data, but we are planning on sticking to the strategy of using FileMaker Pro application in order to facilitate the idea of security and efficiency.

**Integration Diagram**

**Conclusion**

In these times of shortage of teachers in the education industry, there is a need for a change in the system, to enhance the quality and quantity of student teachers graduating. Our application would be able to make it easy and simple for the students and the department to track and follow through with the checklist of requirements for the STIC program. Our program is not only restricted to Northern Arizona University but can be instilled in other universities with similar program offerings across the states.

Our main aim right now is to build a better user interface and make it in such a way that it has the power to present complex information with ease. Considering not only this scenario, but our application also has other applications in document tracking and management framework. We are confident that we will be able to create the desired product, it has been an exciting journey so far and we are thrilled about how the problem is slowly folding out to be.