NAU NORTHERN ARIZONA UNIVERSITY
School of Informatics, Computing, and Cyber Systems

Technical Feasibility Draft

October 23, 2022

Team Teacher To-Do

Project Sponsor: Chris Aungst

Faculty Mentor: Michael Leverington

Team Members:

Sam Gerstner (Team Lead)

Alexander Frenette

Noah Nannen

Shlok Sheth

Bronwyn Wedig

NAU NORTHERN ARIZONA UNIVERSITY
School of Informatics, Computing, and Cyber Systems

Overview

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 road blocks that may be encountered during the development process.

Scope of Project

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.

Plan for Development

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. 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.

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 still remain. This seperation 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 have the need to uniquely identify student, 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 fine 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 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 options 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 Privacy & FERPA Compliance

Due to the requirements of FERPA we will have to take great care to ensure privacy of student records. As noted before we will not be storing student permanent records. Still, there will be identifiable information of 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 data breach, we will be implementing secure hashing to safeguard against deanonymization. Hashing tools 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 26^3*10^2 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.

User Interface & User Experience

For our user interface, we are planning on using Spring Boot, a Java addon that allows us to generate HTML5-based webpages from our backend and use them to import any data that the user needs to import into their profile. Spring Boot will act as a sort of observer, seeing what webpage would best suit the needs of the importer and generating the page from a series of templates that our UI developer will create and implement into the Spring Boot language.

Potential Problems

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 create the project to the fullest extent 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 in 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.

Conclusion