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

Software Design Document

February 17, 2023

Team Teacher To-Do

Project Sponsor: Chris Aungst

Graduate Mentor: Italo Santos

Faculty Mentor: Michael Leverington

Version: 2.0.1

Last Revised: February 16, 2023

Team Members:

Sam Gerstner (Team Lead)

Alexander Frenette

Noah Nannen

Shlok Sheth

Introduction

Arizona and states across the nation are facing an unprecedented teacher shortage; Teachers are a critical part of the development of any society and states are taking swift action to address this shortage. To help fill vacant teaching positions in the state, the Arizona Department of Education is now allowing education students in their final semester to fill these vacant teaching positions in lieu of the traditional student teaching experience. This benefits the students of Arizona, as well as the teacher who gains valuable classroom experience even before earning their degree.

While this program has many benefits, it carries a heavy administrative burden for NAU and the College of Education officials who must track all of the requirements associated with the program. The College of Education is currently using a manual process in an Excel spreadsheet to track student requirements. This process is extremely cumbersome and consumes an estimated 200 man-hours each semester that could be used to provide other support services to students. Because this process is so time consuming, the College of Education is looking for a web application to help them track and maintain documentation related to student success in this program.

There are approximately 75 students each semester participating in this program, and each degree program has different requirements that must be met. In addition to the general requirements to become a teacher, there are subject certifications exams that teachers must pass to be able to teach a subject. Documentation of exam scores, fingerprinting requirements, and more are all needed for a student to participate in this program and these documents must be maintained by the College of Education in the event of a state audit.

College of Education staff are looking for:

- A streamlined way to store and track documentation.
- The ability to generate reports for individuals or groups of students.
- View what requirements have/have not been met by each student.
- Filter students base on requirements that haven’t been met, or are past due.
- Create new requirements to account for possible changes in the program.

By freeing up the time that is currently used to maintain this documentation, COE staff can provide more support services to their students and reduce the chance of human error which can be costly during an audit.

Our main requirement currently involves ensuring that our users' data is secured and accurate, which we branch off into three branches: Student users, Administrator users, and Data Verification. We will be accomplishing this by using the NAU Google authentication system and ensuring both student accounts and administrator accounts must become verified by an overarching administrator, represented as our client. We plan to fulfill these requirements with our project, a streamlined dashboard style web

application that can be used to track student progress and provide a simple interface for administrators to use.

Implementation Overview

As mentioned previously, the College of Education is looking for a way to streamline and modernize this STIC process. So, we are creating a web-based application using Spring Boot Java that is going to have a user-friendly interface. The website works as a to-do list that contains all the requirements to enroll in the Student Teacher Intern Certificate and helps the student to keep track of and follow through with the list of requirements. STIC requires a hefty list of requirements in order to make a student eligible to teach in a school, obviously, you don't let anyone with a degree teach students and become a teacher.

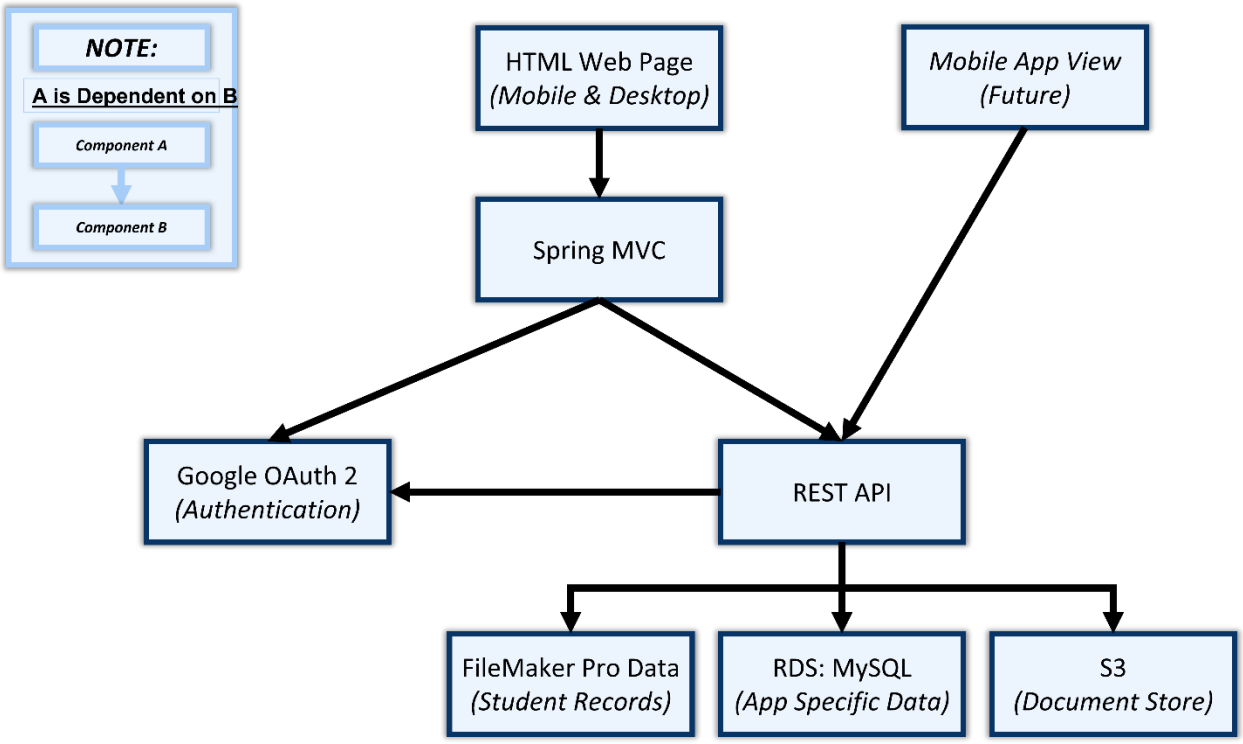
The primary goal of the application is to track completed and missing requirements for students participating in the STIC program. By using a systematic approach to pull student data, COE staff can avoid hours of manual work to load student data into an excel spreadsheet and manually track requirements. Because the application keeps track of all requirements, COE staff can easily log in and generate reports to see which students have missing requirements, etc.

We are planning to keep the student information secure by introducing a hashing algorithm into the website backend, this will keep the student information secure by encrypting the FERPA related information. Currently the department is handling the condition by manually checking for the completed requirements for each student and laying out the plan. Whereas the website we are building is going to create that in a few seconds for the students on demand, with no room for human error.

The website consists of two types of users, one being a student who is our user and the other being an administrator who is going to manage the schema of the database if there are any changes in the structure in the future. Our website is going to give the benefit of doubt to the data stored in the database rather than checking its credibility. One of the stretch goals described by the client is that they want the user to get notified about the pending and upcoming tasks on their phone via SMS, this can be integrated with our system although it requires us to set up an SMS server. We look forward to working on this feature once the goals specified by the client are achieved.

We don't plan on storing or caching any personal information in the website backend, it is going to use CAS to let the NAU student log in into the portal which will directly pass the information to the place holders and the website is going to use these placeholders to search through the database and encrypt the information by not storing but just displaying it directly to calculate the plan for individual student.

Architectural Overview



The Architecture of the capstone project comprises of 3 different phase, first, one bring the input of the FERPA-related information about each student enrolled in the program into the system, second is the phase where the website displays the student's actual data stored in the database correctly and provide different facilities to the user on the webpage like, contact page, which shows the information of the department and responsible parties the third and the last phase would be the phase where the system with a list of Boolean flags calculates the path for the student that has logged into the website using the Google's OAuth facility.

As expected, using this architecture built behind the system we should be able to generate all the objectives out of the website as expected by our client, the amazon S3 server helps us store the documents that are shared and revolved around the secure chain of hierarchy for legal purposes. We plan to not store the FERPA-related information of the students in our system to support the security and anonymity features of the project.

The HTML webpage is provided by the spring framework which supports both mobile and computer view, the webpage as a standalone entity is based on HTML, JavaScript, and CSS and it is attached to the spring framework in order to supply the features to the website, the framework s able to provide facilities like drop-down lists, buttons, and tabs. Firstly, the procedure to operate the website as planned is that there will be two types of users inside the system, one is the student and the other the administrator. le OAuth handles the user type and authenticates the user inside the system. It uses the google account linked to Nau id and creates a token to verify the user's identity. Once that is passed onto the spring MVC, it fetches the data of the student from the database using mySQL. The database we are using is FileMakerPro, this application is used by our client as the main source of storage and a source of verified details about the users. FilemakerPro specifically uses REST API to operate around the data. We are going to use it to fetch the current information of the user inside the database. The Rest API has helped us control the huge data and manage it in a structure of Boolean flags that carry information in secure way as possible.

The returned data is then displayed in the form of completed, uncompleted and pending tasks. As the website is linked to spring MVC, the controller is able to display the returned values in their appropriate places. The task that involve the user of pdf filled forms with accountable parties with restricted access are stored in the S3 document server which will play a very important role in time efficiency as we are planning to support ensign so that the users of the system don't have to scan and print the individual documents manually.

The design of our system is meant to be as loosely coupled as possible. This allows us to think of the entire system as being a collection of modules. These modules are then able to be interchangeable if at future times the client desires to update, extend, or entirely replace a system. The cornerstone of our system is the Rest API. All changes to data will be made through the Rest API, and all resulting changes are visible through the API. All front-end systems will be dependent on the Rest API. That means that if the API changes, then the front-end systems will need to change; but not vice versa. If a front end system wishes to modify their representation of the data, they are free to do so. It is even possible for future front-end systems to be implemented along side existing ones. A great example being a possible native mobile application. As with any front end view, the Rest API will be able to provide all necessary information, as well as a uniform interface to modify the underlying data.

The Spring MVC system is the front-end system we will be implementing. The goal of the Spring front-end system is to effectively act as a view template of our underlying API system, as well as provide the forms necessary to collect the user's input for the Rest

API. This input will not be processed and modify the underlying system, but instead transformed into the required scheme that is accepted by the Rest API itself.

While it is possible to eliminate the need for our Spring system to modify the data, and instead have our web view directly interact with the Rest API, the reason for this decision is simply due to our team's inexperience with reactive web applications. Our team feels motivated that this architecture should be able to satisfy the needs and requirements of the capstone project, we plan to stick to this and build the project. We have tried to keep the system architecture as simple as we could and involve as few variables as possible. Another constraint for us to choose these technologies is the economy, the cost involved in the technologies planned had to be as low as possible. Our team also faces security concerns related to student data, as this data is FERPA-restricted our website had to be secure and side by side independent so that there are no leaks in the data stream.

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

Module & Interface Descriptions

Rest API

The Rest API will be a collection of endpoints, or routes. Each route will have a collection of verbs available that represent the CRUD model. Crud being create, read, update, and delete. These verbs are mapped to the available HTTP methods. These methods are Post, Get, Put, and Delete. Each route will be a hierarchical structure of objects. The base URL of each is representative of the collection of objects. You are then able to select an object using the URL, with the object type's ID as a parameter, and receive the object itself. Each object will have its own schema used to define the fields and types of the request object, and response object.

The Schemas that will be used by the request and response objects are the same schemas that will be within the Spring MVC system, as the spring system will have to generate and receive these same objects.

Routes:

/users

- Post: Create a new user and add it to the collection
- Get: returns a partial collection of users

/users/:user_ID

- Post: Not allowed, due to security concerns, we will not allow a user to select their user_ID
- Get: returns a user that matched the user_ID

/students

- Post: Create a new student and add it to the collection
- Get: returns a partial collection of students

/students/:user_ID

- Post: Not allowed, due to security concerns, we will not allow a user to select their user_ID
- Get: returns a student that matched the user_ID

/students/:user_ID/requirements

- Post: Not allowed
- Get: returns a partial collection of requirements for a particular student

/students/:user_ID/requirements/:requirements_ID

- Post: Not allowed
- Get: returns a requirement object matching the requirement_ID

/students/:user_ID/documents

- Post: add a document to the collection of documents for a matching student_ID

- Get: returns the list of documents with their document ID that matched the user_ID

/students/user_ID/documents/:documents_ID

- Post: Not allowed
- Get: returns the document matching the document ID for the user_ID

/requirements

- Post: creates a requirement object
- Get: returns a partial collection of requirement objects

/requirements/:requirements_ID

- Post: Not allowed
- Get: Return a specific requirement matching the requirements_ID

/documents

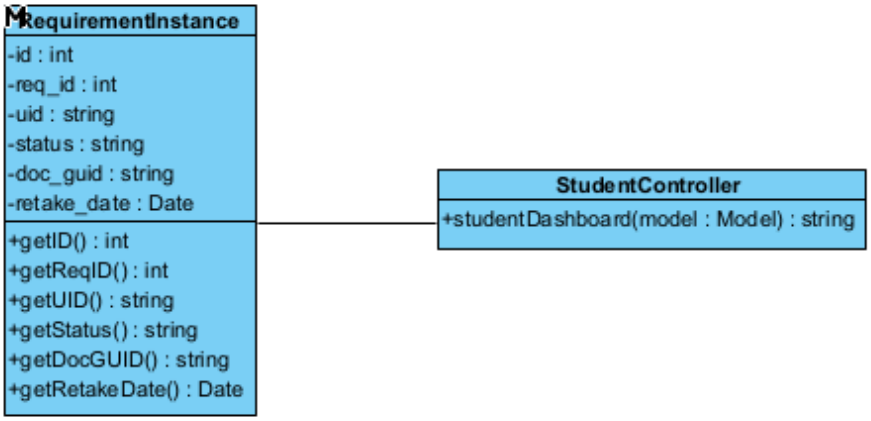
- Post: creates a document object
- Get: returns a partial collection of document objects

/documents/:document_ID

- Post: Not allowed
- Get: Return a specific document matching the document_ID

Student Dashboard

The student dashboard will be the primary module that the students interact with. The student dashboard will allow students to easily view all the requirements associated with their program, and the status of each requirement. From within this dashboard, students will be able to complete all necessary tasks including uploading supporting documentation, adding a retake date for a specific test, marking a requirement as complete, or even contacting the College of Education with questions.



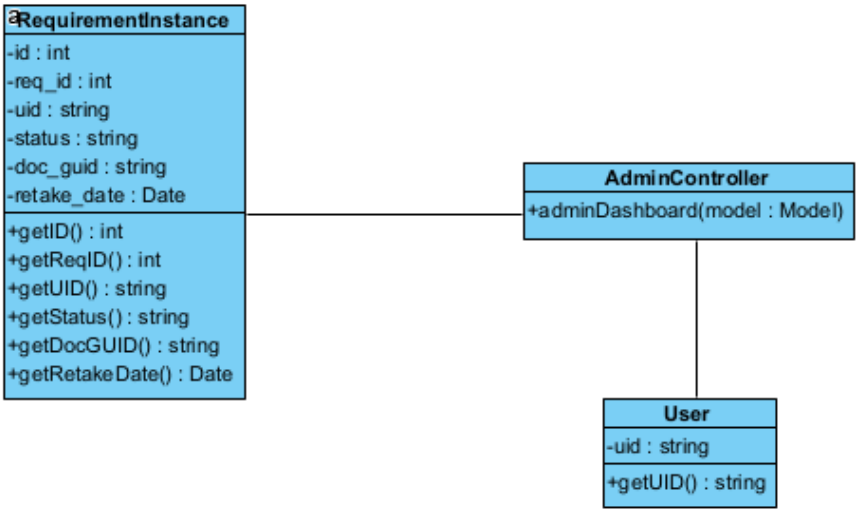
The Java classes and methods associated with the student dashboard are fairly simple. Because we are using the MVC design paradigm, a lot of the "magic" of the data linking and display is done in the Java/HTML Thymeleaf template that allows us to programmatically create the HTML page based off the data it receives from the model. All of the Java classes and methods that are needed to obtain data from the database will use helper methods from a class called MySQL_Helper. The MySQL_Helper class has several static functions that perform various database operations including creating a connection to the database, getting a list of approved users, etc.

The studentDashboard() method is actually very simple and consists of only a few lines of code. The function accepts a single parameter of a Model object. The Model object is part of Spring Boot and acts as a "container" where we can store data or variables that need to be accessed later by the view.

Any HTTP GET request that are sent to the student dashboard will be forwarded to the studentDashboard() method in the StudentController class to be handled. The studentDashboard() method uses the static method getRequirementInstances() from the MySQL_Helper class to obtain a list of requirement instances for the signed in user, then passes the array of requirement instances to the view to be displayed to the user.

Admin Dashboard

The admin dashboard will be the primary module that administrators interact with. The admin dashboard will allow administrators to search or filter students based on a wide variety of criteria including major, specific requirement, completion status, etc. The dashboard will also allow administrators to view details about a specific student including their requirements, completion status, graduation date, and more. The admin dashboard serves as the information center of the application.

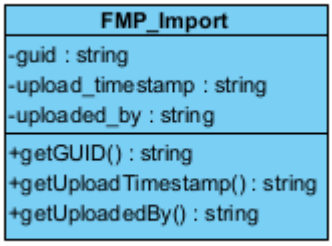


As with the student dashboard, the Java classes and methods associated with the admin dashboard are also fairly simple. We use the same MVC “magic” here as we do with the student dashboard, so most of the work of displaying the information is done programmatically in the HTML.

All of the HTTP GET requests that are sent to the admin dashboard will be forwarded to the adminDashboard() method in the AdminDashboard controller class to be handled. Once again, the adminDashboard() methos accepts a single parameter of a Model type. This object stores all of the data that will be displayed by the HTML page later on. The adminDashboard() method first begins by obtaining a list of all users approved to use the application. This is accomplished using the getAllApprovedUsers() function in the MySQL_Helper class. After getting a list of all approved users, the adminDashboard() method then iterates over the list of approved users, and obtains a list of the requirement instances associated with that user by using the getRequirementInstances() function from the MySQL_Helper class. These UID and requirement instance list combinations are then placed into a HashMap that is passed via the model to the view.

FileMakerPro Import

The FileMakerPro import module will allow administrators to upload the export files from FileMakerPro. These export files are used to determine student eligibility and are a crucial part of the application. This module will also allow administrators to see a history of the uploaded export files including the upload date, and user that uploaded the file.



Document Storage API

The document storage API will leverage Amazon Web Services (AWS) S3 storage service to store uploaded documentation related to specific requirements. These documents may include things like the STIC supervision plan, score reports from Pearson, and more. This API will be used to both add and retrieve documents from an AWS S3 bucket allowing the application to handle all tasks associated with storing and maintaining this documentation.

Document Approval

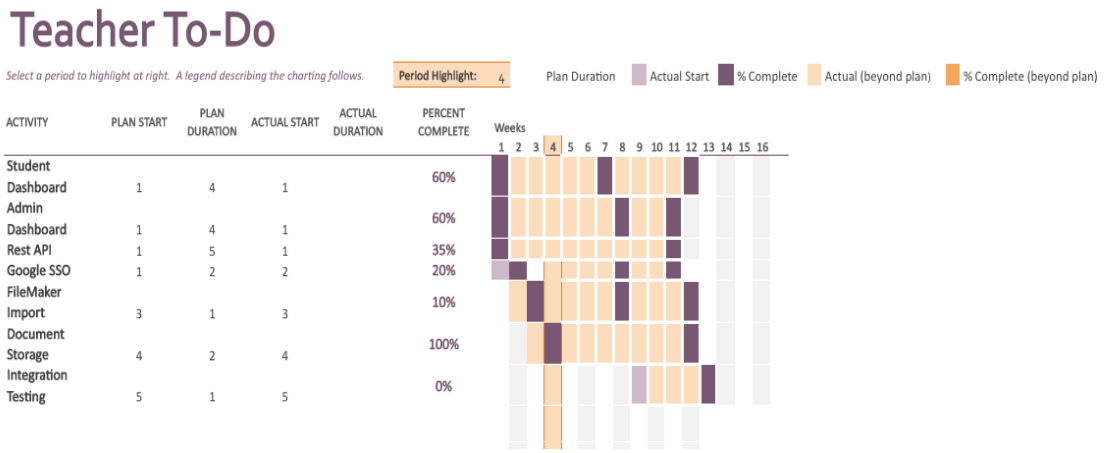
The document approval module will allow administrators to approve supporting documentation that has been uploaded by students. Many requirements require some kind of supporting documentation to prove that the requirement has been met, but this documentation needs to be verified by COE staff. This module will provide administrators with an easy way to view uploaded documentation and approve or deny documentation uploaded by students. This module will also send automated email notifications to students when their document is approved or denied.

Eligibility Verification/Access Request

When a student logs in for the first time, they will be presented with the eligibility verification and access request module. This module is responsible for verifying the student's eligibility for the STIC program using the exported data from FileMakerPro. This module will verify the student's eligibility and attempt to extract their major from the FileMakerPro export file. If the student's major is not specified in the export file, this module will prompt the student to select their major from a drop-down menu.

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

Implementation Plan



The current implementation plan is broken up into two major teams. The front-end and the back-end. The front end is responsible for the Spring MVC section of our design, while the back-end is responsible for the Rest API. With these systems being loosely coupled, the first task at hand is defining the necessary view data as well as the model for our database. These two are the only coupling that we currently have.

For the first few weeks we will have two teams working in parallel. The first team is our front-end team. They are working on creating the pages that students and administrators will see. This includes forms along with the data views showing each student's progress. Our objective is to create the HTML pages of the website compatible with the Spring MVC so that the transition of data from searching the database to displaying it accordingly is smooth.

Our second team, focusing on back-end is working on defining the Rest API and the associated schema. The schema definition aspect is being done in tandem with the front-end team, as both may think of different pieces of data that might better suit our system. The Rest API is also responsible for the storage of student documents. This is a task that is isolated from others, and is a good starting point to get the system up and running.

Another task is creating a Google Single Sign On account, and configuring our system to interact with google OAuth. Currently we are able to build the API and front-end without implementing Authentication and Authorization. However, both front-end and back-end will need to have the authentication and authorization systems in place in order to prevent the leaking of sensitive data. Currently either team could take this responsibility depending on workload, and available time, however we have not made it a current priority as we are working on other tasks.

While it is clear that other tasks remain, we are currently working in increments, and taking time to confirm integration at the end of each section of our work. Our current plan can be thought of as the foundation of our application, and will provide us with the necessary building blocks to further add functionality and expand upon each module.

Conclusion

We believe that as our project takes shape, we will come closer to finding a solution to the teacher shortage in Arizona. It is imperative that the next generation of teachers are properly trained and better equipped to deal with the challenges of the next generation of students. We believe that our system of tracking software will not only assist the College of Education, and specifically our sponsor, in finding new teachers, but also allow them to be able to handle more important issues without a cumbersome system of management for these students. We aim to create a software capable of organizing and planning the requirements of a student and be able to not only express to the student where they are in their academic plan, but where they are going and to do so in such a way that it can be easily understood, but also express to an administrator of the College of Education where their students are in this process to be able to better help them see where aid is needed and to be able to free up their time elsewhere. With the help of this document, through analyzing risks, adopting requirements, adding functionality, and following our project plan, we are well on our way to accomplishing the goal of improving the College of Education and aiding students that may have a hard time along their academic journey.