Software Design Document

2/16/2018
Version 2.0

Project
Tailored Tutoring Business Portal
Robert Lokken

Mentor
Ana Paula C. Steinmacher

Team
Alex Kahn
Jesus Garcia
Taylor Walker
Tyler Mitchell

Table of Contents

1) Introduction

With the growth of the internet and technology, and the hassle of in-person tutoring, there has been a steady rise in online tutoring platforms. It just seems "easier", and "more-efficient" to access tutoring online. While learning styles may be a personal preference, the data and statistics on the industry supports that claim for the majority of people. In 2016, the total revenue of online tutoring services was $411 million dollars, with an annual growth rate from 2011 to 2016 of 4.2% [1]. For 2017 to 2021 the projected annual growth rate is 12.75% [2], meaning that the market is expected to double during this time. But why are we interested in this?

Meet our client, Tailored Tutoring Co., and their founder and CEO, Robert Lokken. A smaller, Flagstaff-local start-up in the online tutoring industry. While they are up against some of the industry-giants like Chegg, and people's choice of the most complete online tutoring platform: WebWiseTutors [3], Tailored Tutoring is finding their niche market that really makes companies successful.

This is where Tailored Tutoring Co. shines. Not only are they a local Flagstaff start-up, tailoring specifically to NAU (Flagstaff AZ) students and courses. But, they also have plans to grow their business down to ASU in Phoenix, and then University of Arizona in Tucson. However, what truly sets Tailored Tutoring Co. apart from their competitors and potentiates their capability for growth, is their unique "Problem-Submission" feature.

The "Problem-Submission" feature allows students to upload a picture of their homework problem to the site, and receive a personalized video-solution that they can then view and watch as many times as they want. When you look at some of the industry-giants, they do not have this option. As a user/student, sometimes you just need help with one or two specific problems that you don't understand. Do you really need to schedule and set up a whole 30-minute or 1-hour long tutoring session for help on one problem? Or, isn't it easier to submit a picture of your problem, and then have access to a detailed video-solution that you can rewatch?

While this "Problem-Submission" feature is what makes Tailored Tutoring Co. unique, it is also a laborious and completely manual process for founder and CEO, Robert Lokken. Currently, it is a big pain for Robert because it is such a manual, tedious, and time-wasting problem for him. To briefly summarize, Robert receives an email notification that a problem has been submitted, then manually go find and notify a qualified tutor in that subject matter, wait on the tutor to notify him that a video-solution has been created, and then Robert has to upload the video and create a personalized

link to send to the student/user. Another user submits a problem?...repeat the tedious process over and over. As the Tailored Tutor Co. begins to grow and scale, Robert needs to be eliminated from the equation, and automation must be implemented.

That is where we, Business Web Solutions, come in. We are building an online portal for both users and the tutors; where students can submit and keep track of their problems, and tutors can receive notifications and post video solutions. The online portal will handle all forms of notification, so that the whole process is automated and enjoyable; thus eliminating Robert from the process. The top part of Figure 1 below shows the process as it is currently is, with Robert manually having to take care of every problem-submission. However, the bottom part of Figure 1 shows our automation process and the web-application handling everything.

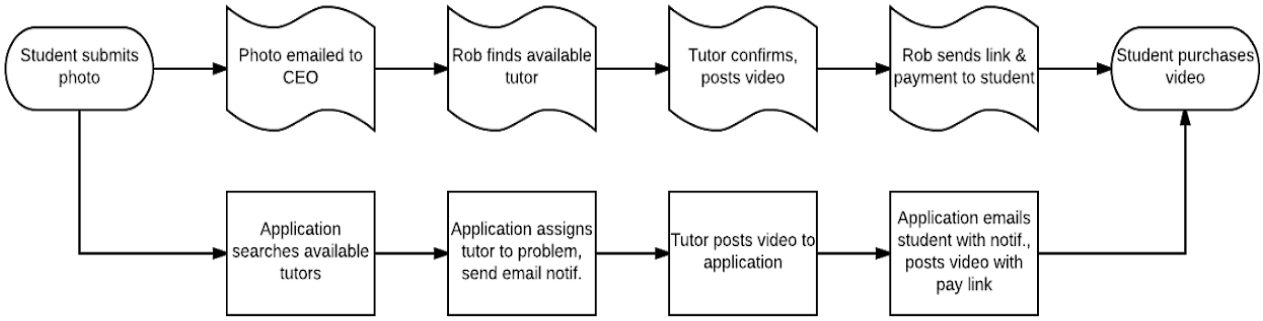

Figure 1: Automation Process

This document will focus on this online portal and the actual implementation of it. A brief overview of what we will be building is an online web-application where students can log-in, view their profile, submit and manage their problems/questions, as well as receive notifications, pay for, and view their video-solutions once complete. From the tutor side it will be much the same, except they will receive notifications when problems in their subject area have been submitted, and will be able to post video-solutions through the online portal. This web-application will handle all of the automation, notifying both Students and Tutors, and serving as a central place for both users to access everything they need.

Also, the web-application will completely eliminate the need for Robert's constant attention and involvement, thus allowing him to focus on running the business and helping Tailored Tutoring Co. to continue to grow and scale. For Robert and Tailored Tutoring Co., we see this online portal as solving a huge area of pain and fixing a bottleneck in their business workflow. With the implementation of the the web-application, we hope to provide a much needed solution of automation for our

client, as well as create a user-friendly online portal and experience for the users of Tailored Tutoring Co.

2) Implementation Overview

In order to remove Robert from the day-to-day manual handling of the business, we are building an online portal to implement automation and take care of the User's needs. To be more specific, this online portal will actually be a web-application capable of handling user log-in, with a database to store user's information. This database will also need to store pictures of problems submitted and videos for solutions posted. The web-application will require many different technologies working together cohesively to implement and create the whole system.

For example, are full tech stack will include: React JS, MongoDB, FireBase, and Amazon Web Services. Excluding FireBase, in previous documents, we have already explained why we chose each technology that we did. Here, we will describe how each technology will be utilized in our web-application.

In order to understand how each piece of technology is being used, we have to understand a basic working process of our web-application. A new user will be welcomed to a "Sign-Up" page, and that user will enter their credentials to create an account. These credentials will be stored in our MongoDB, database, and each user will be assigned a unique user_id in our database. The user will then continue onto their Profile Page (essentially their Home page), and utilizing ReactJS, React will render certain elements or "divs" on the page to present the users information. For the users Profile Page, this include the users name, their school, and even a profile picture. React will also handle presenting these elements on the page in a visually and orderly fashion.

Lastly, AWS (Amazon Web Services) is our online hosting and storage space. It will be used to store and handle all of our image uploads and downloads, as well as all of the videos, or video solutions from the tutors. For example, on the users Profile Page, they may choose to include an image for their profile picture. AWS will hold all of our images, including homework submission images as well as users profile pictures. So, ReactJS will render this image element to the page, calling on MongoDB holding the user's unique user_id and a field with a link to that pictures hosted url on AWS. AWS will return this image, which will be rendered via React JS and visually displayed to the page and the user. The Figure 2 below should help to visually follow this process.

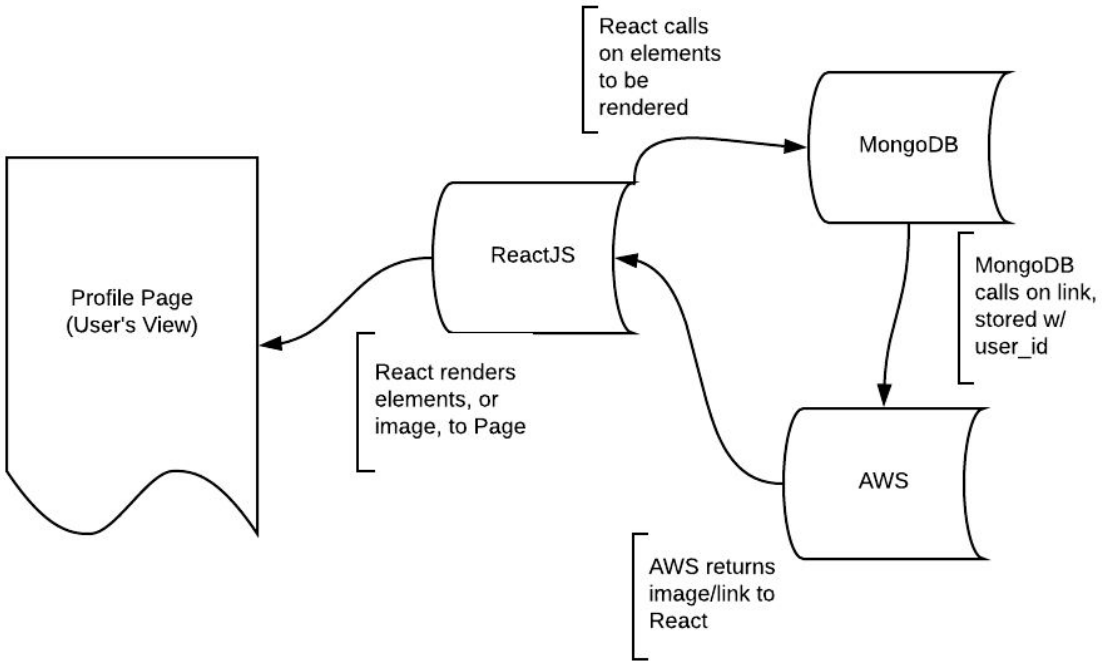

Figure 2, Tech Flow

As you can see from Figure 2 above, our complete web-application involves a few pieces of technology, all integrated together to create the full application. One more tool we should note here, but won't be discussing too much in later sections is our use of FireBase. We used this tool to authenticate our users during login and creating accounts during sign up. It enables us to generate a unique id to tie to each account and to access the profile data in our MongoDB. But, since that is its only role, and for the sake of simplicity, we will solely refer to MongoDB for this whole process (even though FireBase is also technically used).

With an overview of all of the technologies and how they will be utilized in our web-application, we will now continue onto the architectural overview.

3) Architectural Overview

From a high-level, we see our online web-application as being broken up into two major parts: Front end and back end, represented by the User Profiles and The System,

respectively. We use these two higher-level pieces to then break down the actual functionality and control-flow of our web-application.

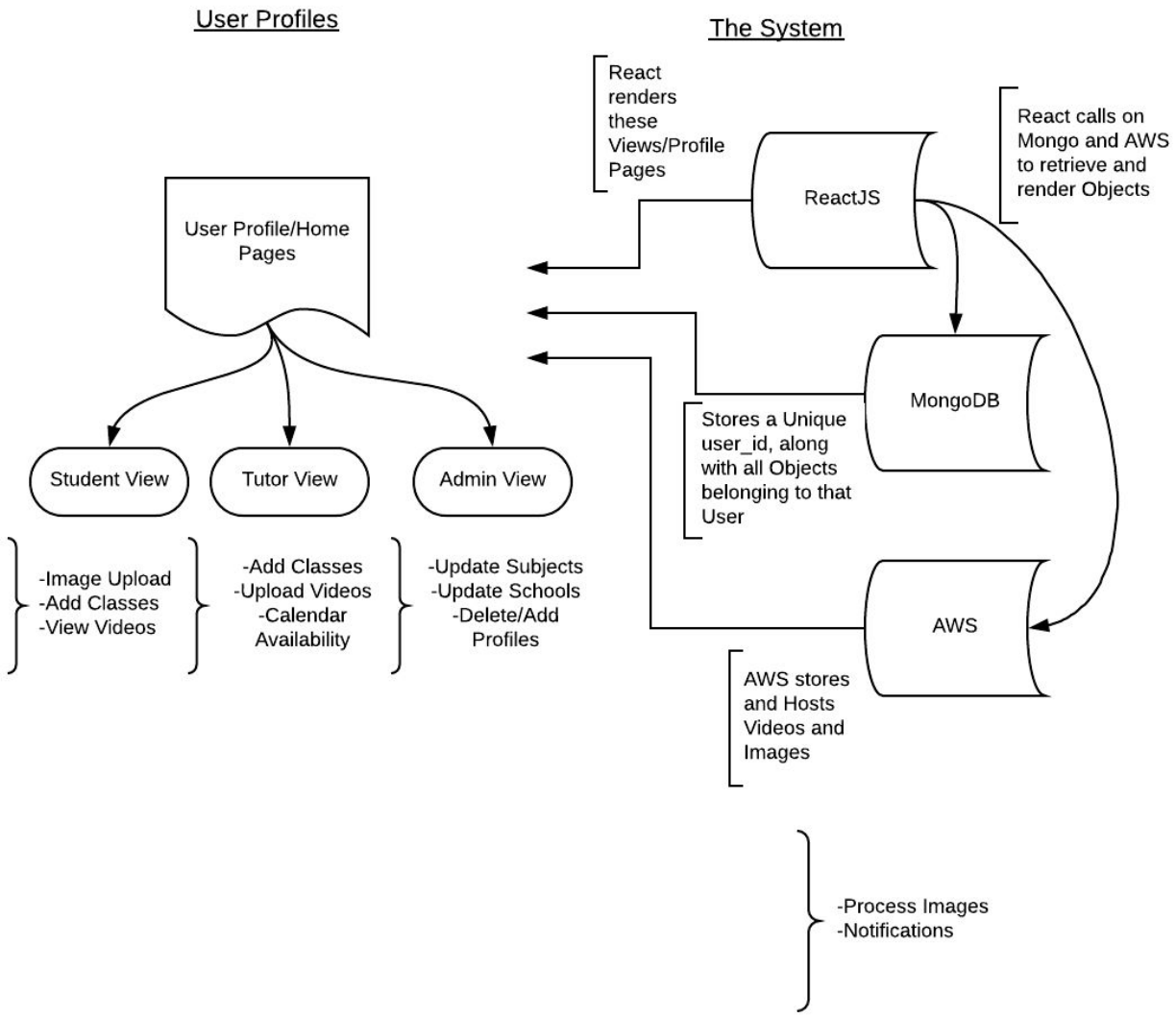

Figure 3: Project Architectural Overview

In Figure 3 above, we can see a breakdown of the User Profiles being the Student Profiles (or View), the Tutor View, and the Admin View. Along with the unique key functionalities of each one of these users (ex: Image Upload, Add Classes..etc). The System, above, is an expansion on Figure 2, and how our pieces of tech work in order to produce the views and functionality that the users experience. Along with the key unique functionalities of The System, which will be processing images that are uploaded, as well as sending out notifications to users.

While this is a higher-level overview of our whole web-application, we also want to provide a good example of how all of these pieces work together to complete a process or particular function. We will go into further detail on each functionality of our web-application in the section, Module and Interface Descriptions. But for now, we will use one example from above to provide a general understanding of all of the pieces.

The Student Profile Page needs to be able to Add Classes that the user is currently taking. From the Student homepage (profile page), there will be a button to add classes. Once clicked and there, a student will choose from available classes. ReactJS will grab these elements chosen on the page, and pass it to mongoDB to be stored under the user_id of that Student user (currently logged-in), in the corresponding "classes" field. These classes will then

While that is just one example of a certain functionality of our web-application, in general, this is the roles that each piece of our web-application will be playing throughout each process. With the User Profile Pages being broken into either Student, Tutor, or Admin, and having their unique functionalities. And with ReactJS rendering the display, and passing elements to be stored in MongoDB, while AWS will be hosting all of our storage. In the next section, we will go more into the detail for each functionality and process of our web-application.

4) Module and Interface Descriptions

As explained in the overview section above, we have broken our web-application into two major parts or modules, consisting of User Profiles, and The System. User Profiles will encompass Student Profiles, Tutor Profiles, and Admin Profiles, where we will discuss the key functionality and differences unique to each user base. All modules and pieces, especially The System, will have to work together in order to provide the overall goals and functionality of our web-application. We will provide graphics, where needed, in order to help visualize the workflow and decision trees of our modules and web-application.

4.0 All User Profiles

Key functionalities:

- *Sign-Up*
- *Log-In*

Sign-up/Log-in

For all of the User Profiles (Students, Tutors, and Admin), the basic workflow will look like the diagram, Figure 4.11, below:

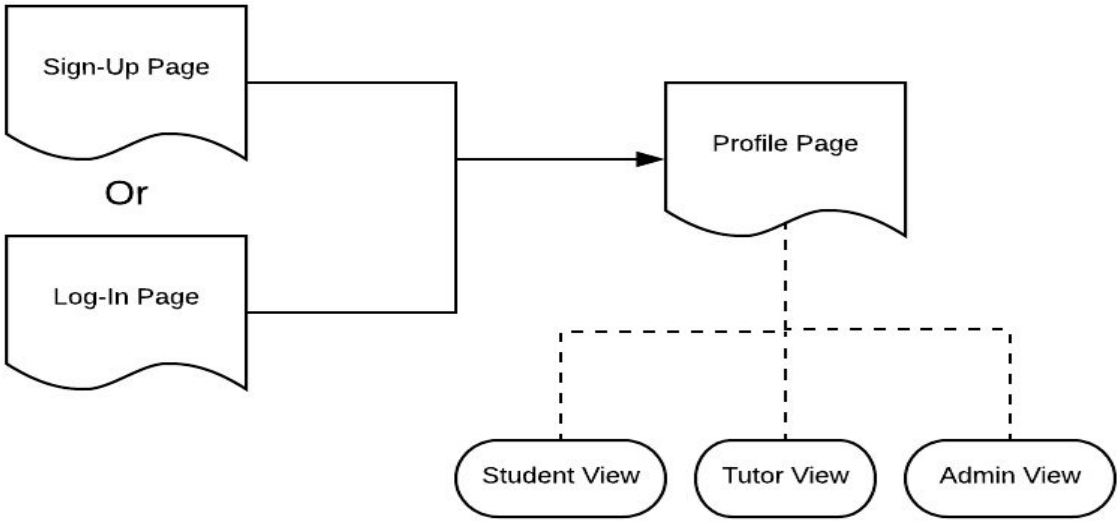

Figure 4.11: User Log in/Sign up

Where they will then differ slightly, is when they get to their Profile Page. The theme, feel, and general layout will be similar across all profile pages, but each User type will have a unique set of functions and capabilities available to them. We will first take a look at the Student Profiles view and capabilities.

4.1 Student Profiles

Key functionalities:

- *Add classes enrolled in*
- *Upload images*
- *View Video-Solutions (Includes Payment Processing)*

Profile Page

The Student Profile will include the Student User's bio information, such as name, location, and school. However, they will also need access to:

- Add classes they are enrolled in to their profile
- Upload images, or homework submission problems
- View Video Solutions, posted by Tutors. Also, pay for these video solutions

Adding enrolled classes to their profile, homework submission, and viewing solutions will be the key functionalities of the Student Profile, depicted in Figure 4.11 below.

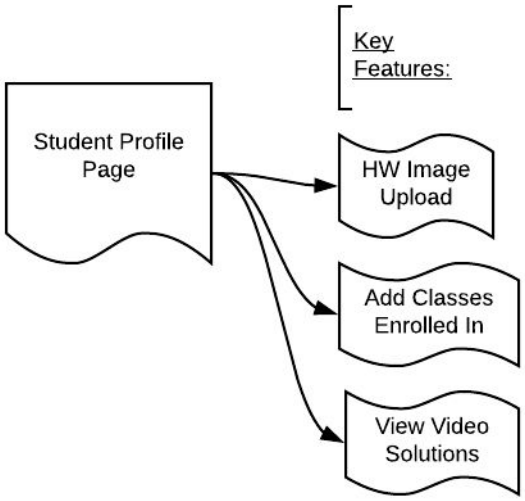

Figure 4.11, Student Key-Features

On the Student Profile Page, there will be a link enabling them to access each one of these key features.

HW Image Uploading

The student will click the link “Upload Homework Problem”, which will taken them to a page where they can upload an image. The student can there choose a local image file, and click to submit it (Note: the Student will also be asked to add a “Subject” tag with the image, see next section “Updating Class List” to better understand subjects). Below, in Figure 4.12, is an image of what happens to make this process work.

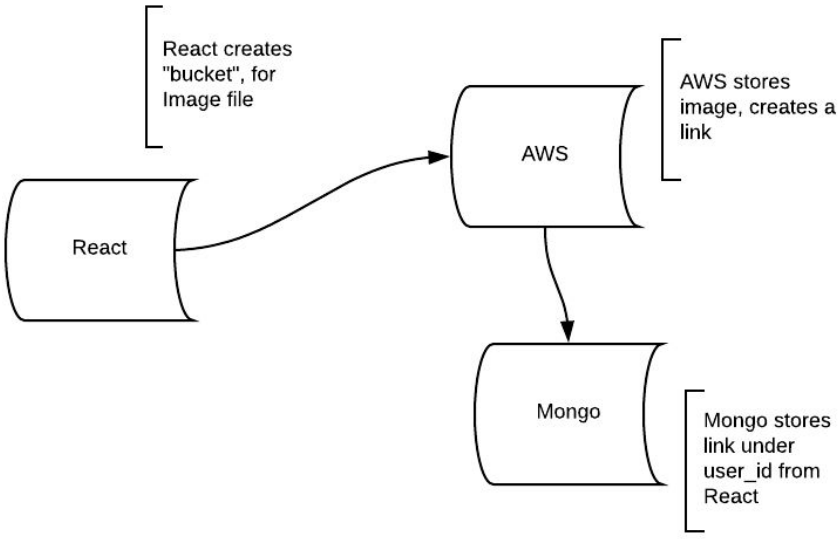

Figure 4.12, HW Image Upload

React creates a bucket object, that holds an image file. This file is then sent to AWS to be stored, where a link is created, and the link is then stored in our database under the users user_id. This enables the image to be stored on our AWS server, while the information linking it is now saved and accessible for retrieval by being stored in our Mongo database.

Updating Class List

The Student User will click a button “Add Classes”, which will take them to a page where they can choose from available class categories, or subjects. React will render these available subjects from an array in our database, Mongo. Below, in Figure 4.13 is a visual of this process.

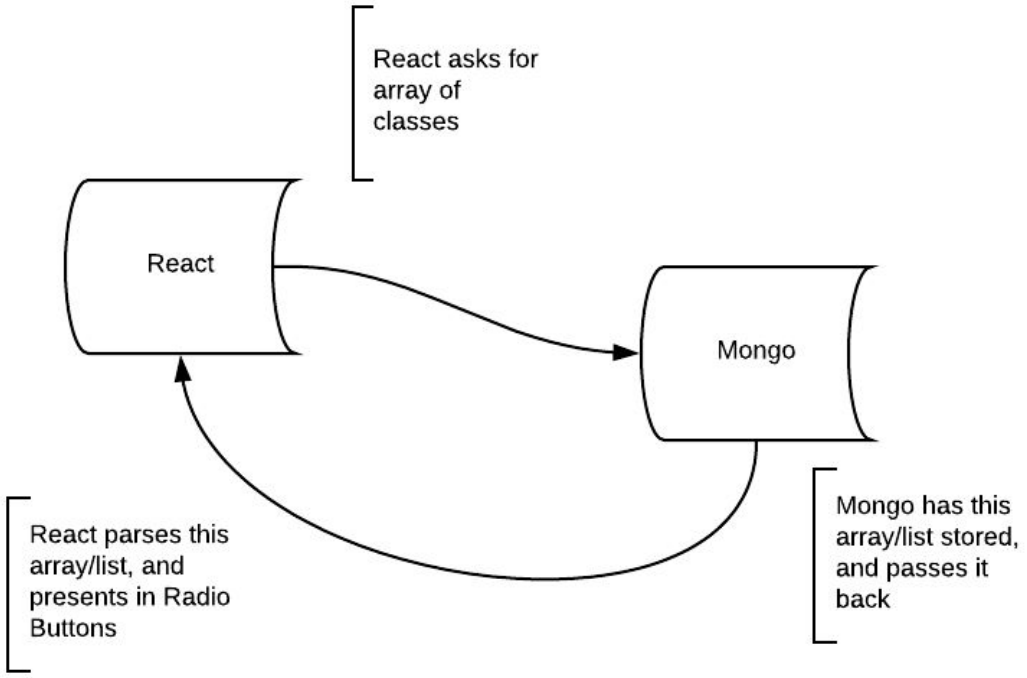

Figure 4.13, Adding Classes/Subjects

As you can see from the figure above, React will ask Mongo for the array variable of Classes, which Mongo will pass back to React. React will then parse this array, presenting each item (or class subject), with an Radio button. The Student User will then be able to click these Radio buttons (or subjects), and add them to their profile.

Adding them to their profile will work much in the same way, whereby React saves these Radio components that have been set to True, or On. And then React will pass these back to Mongo to be stored under the user’s user_id. This information will then be displayed on their Student Profile/Home Page (when refreshed).

Video Viewing & Payment Processing

The System will send the Student a notification that a video solution has been posted for their problem. Within this notification will be a link to view it, however, it will first route them to a payment option in order to be able to view the video. This process is described below in Figure 4.14.

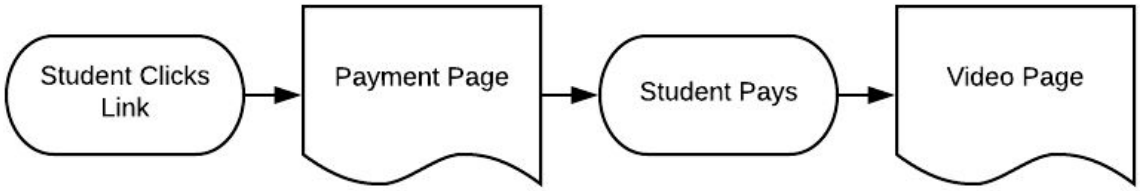

Figure 4.14, Student View and Payment Process

Once the Student User completes payment, they will be redirected to a new page where they can view their video solution. For payment, we are working on implementing a Paypal API. Simply, the link will link to the Paypal API, and once completed, will be rerouted to our video viewing page. Once we correctly implement the Paypal API, it will be very easy to simply route to the Paypal API Page, and then back to the Video Page using React.

In order to retrieve the video, it will work much in the same way as the “HW Image Uploading”, except in reverse (See Figure 4.15 below).

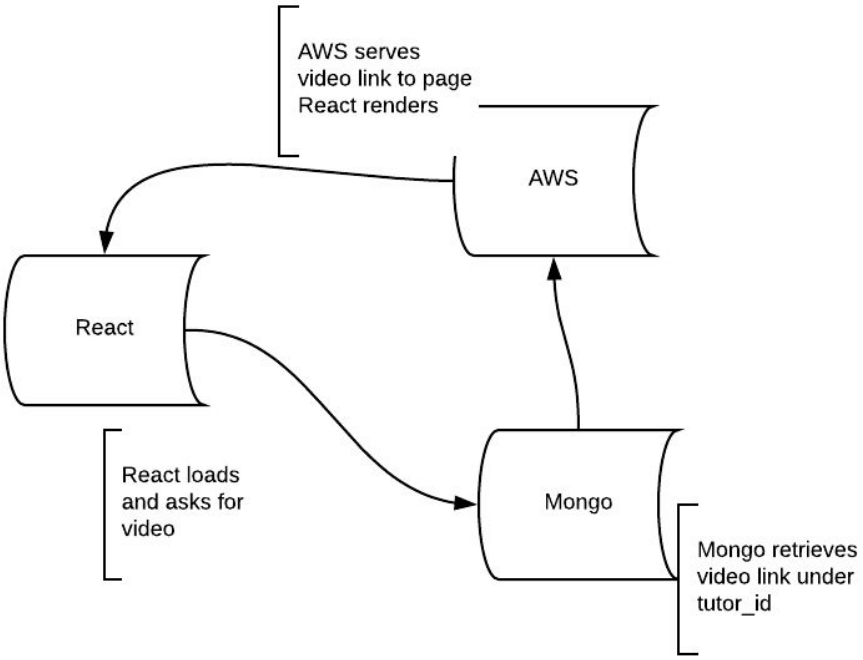

Figure 4.15, Video View Process

React will load the page, asking for the video element from Mongo. Mongo will retrieve the link stored under the user_id (tutor's), and AWS will serve that link back to the page as React renders it.

4.2 Tutor Profiles

Key Functionalities:

- *Add "Classes" that they are qualified to tutor in*
- *Add Calendar Availability*
- *Upload Video Solutions*

Profile Page

The Tutor Profile will include the Tutor User's bio information, such as name and location. However, they will also need access to:

- Add classes they are qualified to tutor in
- Add time that they are available to tutor
- Post Video Solutions, to students submitted problems

Adding classes they are qualified to tutor in, adding calendar availability, and posting video-solutions will be the key functionalities of the Tutor Profile, depicted in Figure 4.21 below.

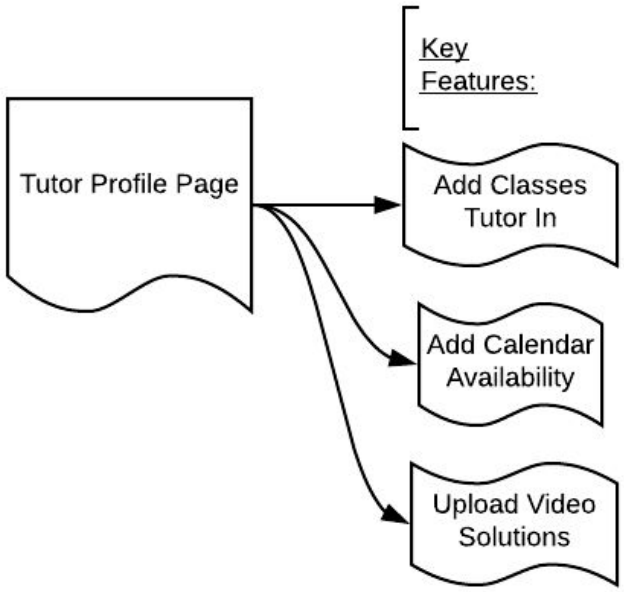

Figure 4.21, Tutor Key-Features

Add Class

This process is also the same as the student profile, in figure 4.13, where the tutor clicks the ‘add classes’ button, then enters their qualified classes to tutor, and this data is persisted in the database to be shown on their profile page. Please see section, “Updating Class List”, under “4.1 Student Profiles”, if a further explanation is needed.

Calendar Syncing

Another important service for the tutor profiles is to sync their calendars with the system. Currently, they can set their working availability and sync their calendars to the website, a service which we will preserve so that they can be scheduled easily. This process is described in Figure 4.22 below.

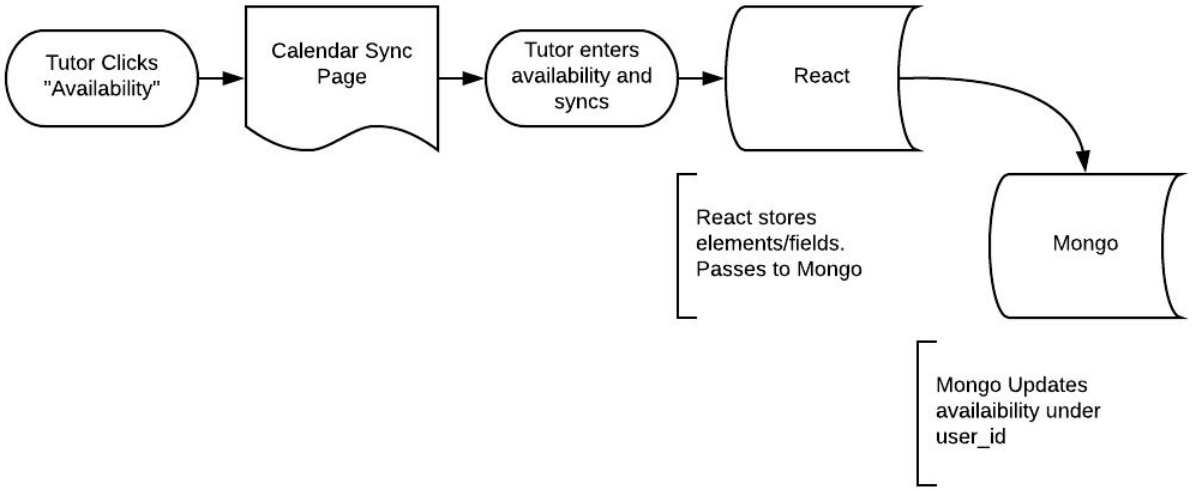

Figure 4.22, Calendar Sync

We will be keeping the current calendar sync page from the original site. However, when tutors update their availability, we will use React to retrieve these elements from the page and store it in our Mongo database under that Tutor User's user_id.

Video Upload

This process will work much like the "HW Image Upload" found in section "4.1 Student Profiles", except with tutors and videos instead of student users and images. However, a quick look at Figure 4.23 should help remind and explain it.

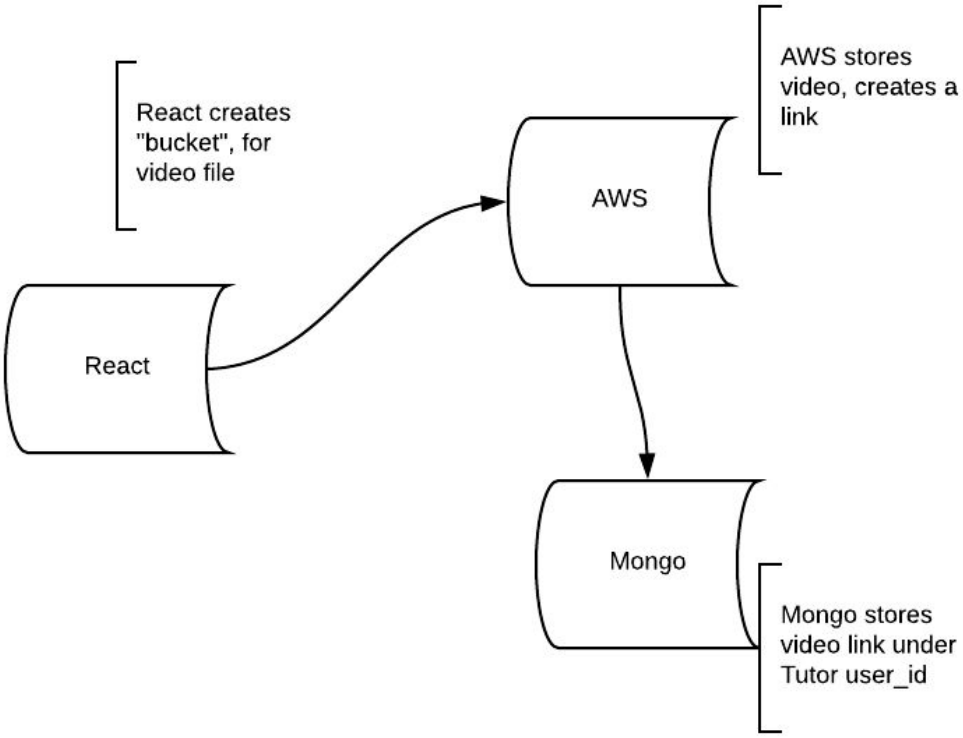

Figure 4.23, Video Upload

4.3 Admin Profiles

Key Functionalities:

- *Create/view/delete Tutor Profiles*
- *Add/Delete Subjects*

The main function of the administrator profiles is to manage tutor profiles, which can be done through the FireBase interface. This interface is easy to use, and already developed with FireBase, and allows administrators to create and delete tutor profiles at will. This access, password and controls to the database, will be given to Mr. Lokken once we complete the web-application system. It is a back-end profile, not a forward facing one.

Also, Mr. Lokken will be shown how to edit a particular variable in the database, “Subjects”. That way he can add and delete class subjects that they currently offer tutoring in. “Subjects” will be the array/list that is available to both Student Users and Tutor Users when they click the “Add Classes” button.

4.4 The System

Key Functionalities:

- *Process Images -> Notifying Available Tutors in Class/Subject area*
- *Send Notifications*

Remember, The System here refers to all of the backend functionality of our web-application including ReactJS, MongoDB, AWS, and FireBase. Much of the previous sections on User Profiles describe The System, doing quite a bit already. You should be familiar with how The System works. For that reason, in this section, we will treat The System as a whole, and focus more on the decision making and logical processes involved in two more key functionalities that don't fall under any of the User functionalities: processing images and sending notifications.

Send Notifications

We will first explain notifications, which will help us to better understand an essential element in the "Process Images" functionality. Our system will send notifications upon two events occurring, either a Student User uploads a HW Image, or a Tutor User uploads a video solution. Both of these events will trigger a notification, depicted in Figure 4.41 below.

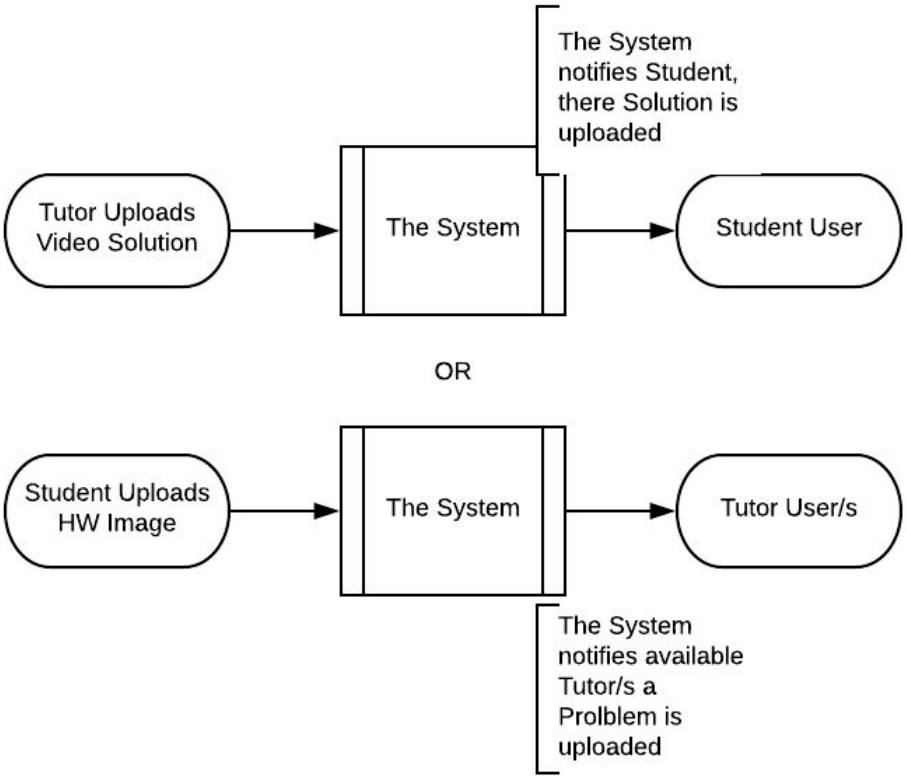

Figure 4.41, Notification Events

If a Tutor uploads a video solution, The System will notify the Student User that the video solution belongs to that their hw problem has been answered. Vise versa, if a Student uploads a hw image problem, The System will process the image, described in the section below, and then notify available tutors that a hw problem has been submitted that they may now answer. As of right now, we are still unsure whether we will be implementing notifications through the User Profile Pages, or with just email using AWS. For that reason, we will not go into detail here on The System process.

Process Images

The System needs to recognize when a Student uploads an image, store and read the Class/Subject associated with the image submission, and then read and see a list of all currently available Tutors that are able to teach in that subject.

This would involve saving the “Subject Tag”, and then parsing all tutors that have that “Subject Tag” attached to their profile, and finally checking Calendar Availability, and parsing currently available tutors, and then sending those tutors a profile notification. Also, possibly sending notifications to currently-unavailable tutors if None are currently-available. A visual can be seen in Figure 4.42 below.

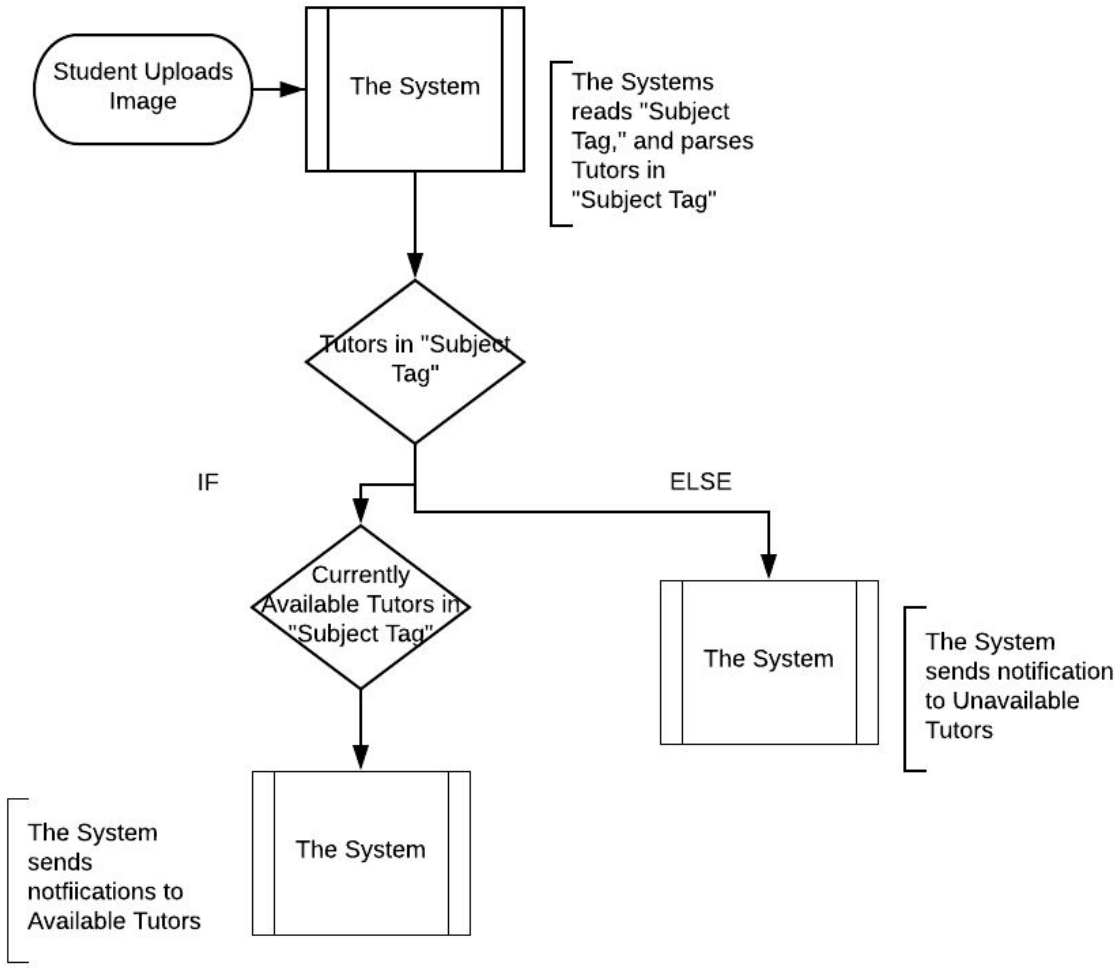

Figure 4.42, Process Images Flow

Here, The System is essentially parsing information and implementing a little bit of logic. To quickly describe the back-end of The System here, it is very similar to other processes previously discussed. The System, or React will take in the element associated with “Subject Tag”, utilizing a function and passing it back to Mongo, where we will perform a search on all Tutor Users with “Subject-Tag” in their subjects array/list. From there, The System will have a leftover of all the Tutors with that “Subject Tag”.

From there, The System will parse that results of tutor users further, by checking the availability of those Tutor Users. Utilizing a time function, if there are currently available Tutor Users with that “Subject Tag”, a notification will be sent out to all of those Tutor Users. If not, it will send it out to all of the Tutor Users with the “Subject Tag” (this will include unavailable tutor users, obviously).

5) Implementation Plans

As discussed before, this software solution requires the implementation of many different pieces working together. That being said, we will try to break down production of the software into many different actionable pieces and steps. However, there will be much overlap and constantly checking for synchronization so that we don't waste time building a piece that won't work or function with the rest of the software as a whole.

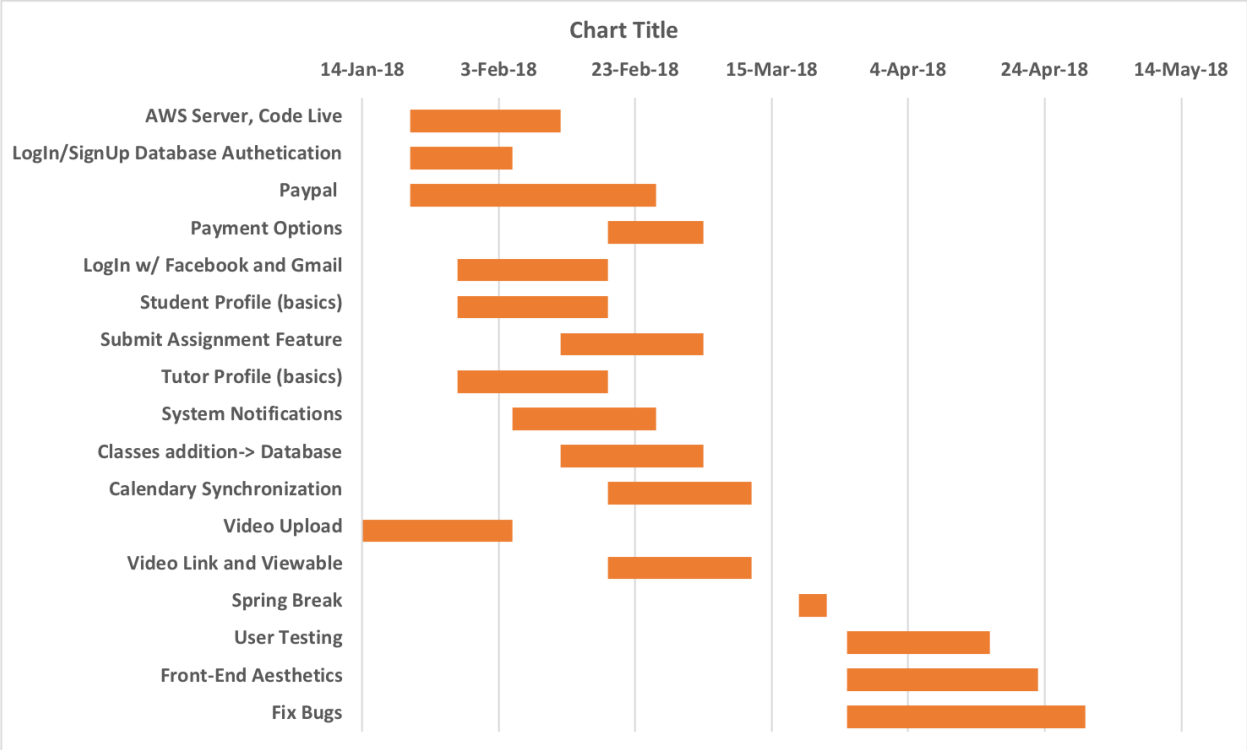

Chart 5.1, Gantt Chart

Above is a Gantt Chart visualizing the breakdown and timeline of the development of our web-application. Two important segments of our production, to note, are the time before spring break (March 16th), and the segment after spring break. Our main goal before spring break is to have a completely functioning version 1.0 of our software up and running. After spring break is when we plan to polish this version of our web-application, making it pretty, ironing out the kinks, debugging, and implementing user-testing. As with any software, the development process is never truly complete, and the software gets updated and improved with continued use and feedback. However, our goal is to have a working infrastructure by March 16th, and an improved

and polished version 1.0 ready for our client by the end of the Ugrads Presentation, or April 27th.

The Gantt Chart above shows specific tasks and functionality, along with a timeline of when it is being built. However, it does not show which individual in our group, Business Web Solutions, will actually be building each component. This is because our group functions in a way where each individual offers to build or complete certain tasks that align with their strengths or interests at the time. It has worked for us in the past, and will work for us in the future. But, to provide a general understanding of the main focus for each team member, below we have provided a table showing the team member's name and their focus and strengths. Note, this is still flexible, but what we generally notice and adhere to. An example of our strengths and some of our task are listed in Table 5.1 below.

Team Member	**Strengths/Interests**	**Examples of Task**
Taylor Walker	Front-End, User Interaction	*Ex: Student Profile, Front-End Aesthetics*
Alex Kahn	Front-End, User Interaction	*Ex: Tutor Profile, Calendar Synchronization*
Tyler Mitchell	Back-End, Overall System Cohesion, Authentication	*Ex: LogIn/SignUp Database Authentication*
Jesus Garcia	Back-End, Overall System Cohesion, Uploads/Downloads	*Ex: AWS Servers-Code Live, Video Upload*

Table 5.1, Team Member's Strengths

This section's goal was to provide the reader with an understanding of how the software, web-application, is incrementally being built out over the semester. As well as provide the tentative timelines of those tasks, and who will be focusing on building them. We are currently on-time in accordance with our Gantt Chart (Chart 5.1), with tasks such as Video Upload, and having a live connection to our AWS server already being completed. However, this our plan, to the best of our knowledge, but we will also be implementing an Agile development process and handle breakthroughs and issues as they arise. Timelines may be affected as development occurs, but we will use our Gantt Chart as a guideline to keep on pace.

6) Conclusions

In conclusion, Tailored Tutoring Co. has a really unique feature, “Problem Submission”, that sets them apart from the rest of the online-tutoring industry. However, in its current state, the process is completely manual and extremely cumbersome and time-consuming for founder and CEO, Robert Lokken. Our solution is to build an online web-application where users of Tailored Tutoring Co. can interact with this web-application and The System, which will completely automate the “Problem Submission” process and more.

We have discussed the tools we will be using to build this web application, which consist of MongoDB, AWS, and React. Also, we have abstractly broken down our solution into 2 major pieces: User Profiles and The System, where User Profiles entails Student, Tutor, and even Admin Profiles. Discussing the key differences and functionalities of each, as well as how they all interact with each other to complete the whole make up and functionality of our web-application.

We are currently on pace to have our web-application built by March 16th (Spring Break), already having completed several tasks on our Gantt Chart and having key functionalities such as user sign-up and log-in already implemented. Also, we have the beginnings of our user profiles pages set up and viewable. Once we implement all of the functionality of our application by March 16th, we will be able to continue refining it for the remainder of the semester.

But most importantly, this automation and online web-application we are developing will enable Tailored Tutoring Co. to be able to grow and scale as a company. Completely removing Mr. Lokken from the equation and solving his current problem. Now, not only can Mr. Lokken focus on growing the business, instead of running the day-to-day operations. But, the business, or web-application, will be able to handle the and grow right along with them.

References

1. “Online Tutoring Services: Market Research Report.” Online Tutoring Services Market Research | IBISWorld, www.ibisworld.com/industry-trends/specialized-market-research-reports/online-retail/lifestyle-services/online-tutoring-services.html.
2. Ravipati, Sri. “Online Tutoring Market to Grow 12.75% Between 2017-2021.” THE Journal, thejournal.com/articles/2017/01/12/online-tutoring-market-to-grow-12-percent-between-2017-2021.aspx.
3. “The Best Online Tutoring Services of 2017.” Reviews.com, www.reviews.com/online-tutoring/.