Technology Feasibility

11/07/2017

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

Our client, Tailored Tutoring Co., is a start-up online tutoring company here in Flagstaff, AZ. It is kind of a blend of other online-tutoring platforms such as Chegg and Khan Academy. Like Chegg and Khan Academy, Tailored Tutoring offers in-person tutoring for clients, as well as online video tutoring. However, what makes Tailored Tutoring Co. unique is their feature "Submit My Assignment", where students can upload specific homework problems and receive repeatedly-viewable, personalized video-solutions from qualified tutors.

While the "Submit My Assignment" feature is what makes Tailored Tutoring Co. unique, it is also what is currently causing the biggest problem and headache for our client Robert Lokken, the founder and CEO of Tailored Tutoring Co. Robert currently has to manually handle every problem-submission via the "Submit My Assignment" feature, repeatedly wasting lots of time and energy during the process. For the user, the submission process is also a little cumbersome and frustrating.

As it currently goes: A student has to visit the Tailored Tutoring website, take a picture of the problem on their phone, most likely email it to themselves, download it to their computer, and then upload that picture to the Tailored Tutoring website. The current website, www.tailoredtutoringco.com, is not optimized for mobile use, making the user's problem-submission process more difficult.

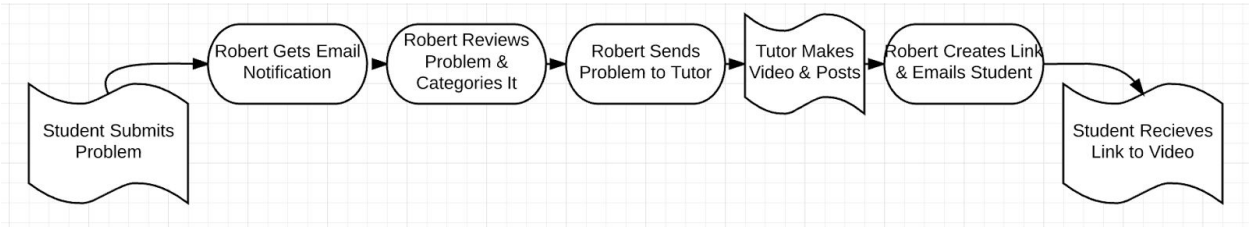

Diagram 1: Problem-Submission Process

From there: Robert gets an email notification that a problem has been submitted *(Diagram 1 above)*. He then has to review the problem and categorize it under the current subject, ex: Calculus 1, Chemistry, Algebra etc… Once categorized, he looks for qualified tutors in that subject area to see who is currently available, and then emails them the submitted-problem. The tutor creates a video solution, uploading it to a Google Drive, and emails Robert letting them know the video is complete. Robert then has to create a personalized link for the video solution, and email and send that link to the client notifying them that their solution is now available.

As you can see, the current process is just too cluttered, time-consuming, and unsustainable; both from the users end, and especially our client, Robert's end.

In order to refine their business process, and help Tailored Tutoring Co. scale, that is where we, team Business Web Solutions, come in. Business-Web Solutions is the NAU CS Capstone team comprised of Jesus Garcia, Alex Kahn, Tyler Mitchell, and Taylor Walker. Our goal here is to simplify our client's business-process by building an automated system for Tailored Tutoring Co.'s "Submit My Assignment" feature.
We will start with building a website that is optimized for mobile, that way the users/students can access the website via their phone, take a picture of their problem on their mobile-device and upload it directly to Tailored Tutoring Co. via their phone.

Diagram 2: Automated Submission Solution

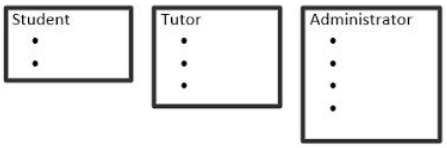

Diagram 3: GUI Profiles

Our other main goal is to automate *(Diagram 2 above)* the whole "Submit My Assignment" feature from Tailored Tutoring Co.'s end, making Robert's life much easier. This will involve building a scalable web-application with a GUI system and profiles *(Diagram 3 above)* for both tutors and students/users. The user profiles will detail the subjects and classes the students are currently taking, allowing them to submit the problem under a certain subject-category. From there, the system will match the subject-category with tutors in that area, and will automatically notify these currently available tutors about the problem. Thus, eliminating the need for Robert to be constantly on-call, and having to manually handle notifications and emailing both the clients and tutors.

Now that we have given a brief overview of who our client is, detailing their current business-problem and how we intend to solve it, the rest of this document will get more in-depth on our solution and focus on the technology-aspect of how we intend to build this solution. In this document, we will discuss the specific technologies we have chosen, why we chose those particular technologies versus other viable options, and prove the technological feasibility of how these technologies will function together enabling us to build out our solution of an automated system.

2) Technical Challenges

With any large scale project there are a number of considerations that must be made before the design process can start. A firm understanding of the key components of the project and the

technologies on offer that can complete those parts is crucial in any planning phase. This section goes over, at a high level, the major components of our project.

- **Lightweight Framework**- We will need a lightweight framework to integrate all of our technologies together, in order to complete the web app.
- **Storage**- We will need a way to store extremely large amounts of data in the form of videos and images. Ideally...
    - The client could add more storage capacity as needed.
    - The videos should be able to be streamed on the website.
    - The videos can be normalized.
    - The videos won't have to buffer under heavy load (i.e. exam season)
- **Database**- We will need to store customer and employee profile information in a database. Ideally:
    - This is easily scalable to other business sites for the future
    - This uses a flexible model to fit the client's evolving business needs
    - This software is open-source to support small business budgets
    - This is a reliable system and performs well under mixed workloads
- **Hosting**- We will need some service to host our website once it goes live
    - This service will need to fast and reliable, also accounting for future growth
    - This service must be able to host and incorporate all other pieces of technology we've chosen
- **User Interface/Mobile-Friendly**- We will need to design the web-application to be mobile-friendly.
    - All features must look great from mobile
    - All functions must work seamlessly from mobile

3) Technology Analysis

Now that we have established each of the major components of our project it is now time to dissect each one and find how best to tackle that section. In this section we look at each of the above stated problems, but much more in depth this time. We look at the potential challenges of each part, the technologies on offer that can help us, and at the end a decision on which of the technologies to use. This helps us form an overall understanding of each problems, how exactly to complete them, and inevitable, how each part joins to form the whole.

3.1 Frameworks

3.1a) Problem

Our client needs a stable application that can handle the many aspects of his business. To be more specific our client needs an application that is intuitively designed, so that both end users and his employees know exactly what to do and how to do it. He also needs full integration with

his file storage, file uploading, and payment systems. In addition to that our client needs his application to maintain personalized user accounts and pages for his clients and employees. These profiles can be accessed by site users as well as administrative accounts to oversee his application and its user. The problem of needing a lightweight framework that allows us the opportunity of the integration between our technology will be resolved with careful consideration and in-depth analysis of staple frameworks commonly used in applications today.

3.1b) Our Options

Our client maintains that the web portal for his site is one of crucial importance, making the decision on which web framework to use very important. Web frameworks are booming now as Web2.0 has increased in popularity alongside the development of web apps. As a result there are a plethora of frameworks to choose from. Such frameworks include Facebook's ReactJS, Google's Angular series, and MeteorJS. Web frameworks are extremely popular and are clearly backed by some big names in the industry, but no two are alike so they must be compared in order to find the best fit for our client.

3.1c) Comparisons

- **Meteor** - Meteor is a robust framework that can be used for just about any application you can think of. It has a massive package library to draw from that include things like components for logins, user interface features, and database connections. While it has a large library to draw from, Meteor can be limited in what it can actually do. Meteor is referred to as "opinionated," meaning that a lot of components are decided by Meteor and not the developer. For instance, Meteor exclusively works with MongoDB, and can't integrate with other databases. This is limiting for obvious reasons as it means we have only one option for our database. Lacking freedom can cause problems later on during development and can cause us to develop into a corner with no way out without a perfectly planned tech stack.

- **Angular** - Angular is another robust framework much like Meteor, and offers many of the quality of life bonuses that Meteor utilizes. Instead of a package library to install from, Angular seeks to solve all problems with their own implementations, so instead of constantly installing new packages made by independent open-source developers, it has it's own packages for every single thing. How this benefits the developer is through consistency. Since Angular is responsible for all packages, problems such as compatibility conflicts between packages is almost non-existent. Angular also suffers from being "opinionated" but to a lesser degree than Meteor. While it may try to dictate what components to use in the tech stack, it is still opened up for other implementations. This benefits the developer in helping develop the stack they want to use, but can still be limiting in the long run.

- **React** - React is different from the other in the one simple fact that it is not an actual web framework. Instead, React is just a javascript library for frontend development, meaning that React doesn't have a package library for every little detail that the developer may

need. It instead opens it up for the developer to design these components and features themselves. It embraces a do-it-yourself philosophy to development allowing the developer full control of their stack. The drawback to this is that a lot of the details on the back end are not abstracted out for the developer. Instead it is almost exclusively left up to the developer to establish the connections with their stack. This creates more work in the long run, however it is very difficult to be caught in a corner with no way out, and it helps to modularize the development process. It is ultimately the option with the most freedom, but at the cost of a learning curve and added development time.

3.1d) Conclusions

All three of the options offer their own strengths and weaknesses, and the decision ultimately lies in evaluating each of the options' strengths in relations to those weaknesses. If we are to look at our options in terms of the freedoms in our stack, and thus, the strengths they provide in their use we can compare the options thusly:

React>Angular>Meteor

We have seen that React is by far the least limiting option, and moving down we get more and more limited in what our options are in our stack. But this comparison is in direct opposition to the level of abstraction that each options gives. Comparing that abstraction we get a comparison that looks like this:

Meteor>Angular>React

Meteor is at the forefront of this comparison on account of its sheer level of abstraction, and as we move down the line we see less and less of that abstraction with React having almost none. This may look like a bad thing for React, however as we've shown above that lack of abstraction is what creates that abundance of freedom.

We believe that the lightweight, and limitless nature of React will allow us to deliver on our promise to our client with a tech stack that suits his needs.

3.2 Large Data Storage

3.2a) Problem

Our client is running a business that relies on online storage for videos and images, and as such, our solution requires a way to store massive amounts of data easily. On top of this, the service our client is providing is certainly one of convenience, so there is the added requirement of reliability where any given video has a very high rate of uptime. Of course, there is a desire to be as cost-efficient as possible as well.

3.2b) Our Options

Our client has shown great interest in Scalability with our solution, so with that we decided that to save our client time in the future, we would only consider the big names in the industry. The big names we had considered were Amazon Web Services, Google Cloud Platform, and Microsoft Azure. While these three cloud platforms are all similar, they have different methods of implementation and features, as well as different pricing models.

3.2c) Comparisons

For similarities, all services have 3 different levels of object storage types, a “Hot,” “Cool,” and “Cold” storage. These different storage types charge different rates for access, so for Hot, where access is determined to be common, there is a higher flat rate, but lower access request rates, and for Cold, there is a lower flat rate, but higher access rates, and Cool is a mix of both.
For prices, they all have a Per GB/per Month model of charging, as can be seen in diagram 4 and 5 below, which leaves no room for paying more than you need:

(Prices taken from the official websites, all for US West: Oregon server space)

Amazon S3	**Standard “Hot” Storage**	**Infrequent Access “Cool”**	**Archive “Cold” Storage**
First 50 TB / month	$0.023 per GB	$0.0125 per GB	$0.004 per GB
Next 450 TB / month	$0.022 per GB	$0.0125 per GB	$0.004 per GB
Over 500 TB / month	$0.021 per GB	$0.0125 per GB	$0.004 per GB
Google Cloud			
First 50 TB / month	$0.02 per GB	$0.01 per GB	$0.007 per GB
Next 450 TB / month	$0.02 per GB	$0.01 per GB	$0.007 per GB
Over 500 TB / month	$0.02 per GB	$0.01 per GB	$0.007 per GB
Azure Storage			
First 50 TB / month	$0.0208 per GB	$0.0152 per GB	N/A
Next 450 TB / month	$0.02 per GB	$0.0152 per GB	N/A
Over 500 TB / month	$0.0192 per GB	$0.0152 per GB	N/A

Diagram 4: Table comparing prices of services

They also charge for “per 10,000 operations,” like Read and Write operations:

Amazon S3	**Standard “Hot”**	**Infrequent Access “Cool”**	**Archive “Cold”**
PUT, COPY, POST, LIST	$0.005 per 1,000 requests	$0.01 per 1,000 requests	$0.004 per GB
GET and Other requests	$0.004 per 1,000 requests	$0.01 per 10,000 requests	$0.004 per GB
Delete	Free	Free	Free
Lifecycle Transition Requests into Standard – Infrequent Access *(or Cold)	N/A	$0.01 per 1,000 requests	*$0.05 per 1,000 requests
Data Retrievals	N/A	$0.01 per GB	(Special Pricing)
Google Cloud			
PUT, COPY, POST, LIST	$0.05 per 1,000 requests	$0.10 per 1,000 requests	$0.10 per GB
GET and Other requests	$0.004 per 1,000 requests	$0.01 per 10,000 requests	$0.05 per GB
Delete	Free	Free	Free
Azure Storage			
WRITE, LIST, CREATE	$0.055 per 1,000 requests	$0.10 per 1,000 requests	$0.10 per GB
READ and Other operations	$0.0044 per 1,000 requests	$0.01 per 10,000 requests	$0.05 per GB
Delete	Free	Free	Free
Data Retrieval per GB	Free	$0.01 per 10,000 requests	N/A
Data Write per GB	Free	$0.0025 per 10,000 requests	Free

Diagram 5: Table comparing prices of operations

3.2d) Conclusions

Among the various videos and articles we found, the general consensus seemed to be that AWS is for small businesses, Google Cloud is for personal use and research, and Azure is for Enterprises. With this knowledge in hand, it seems AWS and Azure are the two choices, and of those two, AWS would fit closer to what the client would be defined as. While our client would like this business to be huge, it isn’t a business model that would reach Enterprise level needs. Considering we will likely be using MongoDB, AWS and Google Cloud seem to have okay

documentation to using MongoDB with each service, however, Azure seems to only have a mere mention of the possibility of Azure+MongoDB. Our client has also shown interest in AWS and its pricing models so we will be using Amazon's S3 storage service.

3.3 Account Data Storage

3.3a) Problem

Our client has a business that deals with multiple employees and customers that communicate online every day. To expedite his business, our application will store user profiles for students, tutors, and administrators. Our application will need to use a database technology to store this personal data for users' profiles, as well as store transaction information. Our client is also planning to scale his business to other university campuses, and as such, we will need software to be scalable to accommodate this. In choosing database technology, we are researching options that are low- to no-cost to achieve as small a project budget as possible, scalable so that the client may expand their business technology easily in the future, and works well with a web application by using languages compatible with our front-end framework. Considering each of these requirements, NoSQL databases were our primary choice due to being easily scalable and using flexible data models that would fit the changing nature of a rapidly growing business. With relational databases, changing a data model to fit a change in the business would take a substantial amount of effort, so that alone was enough to rule them out for consideration for us.

3.3b) Our Options

We decided to choose NoSQL databases due to our client's growing business model, which would be constrained by a standard relational database. So, our options for NoSQL databases were some well-known choices, MongoDB, CouchDB, and Azure DocumentDB, each with ample documentation, as well as analyses done by professionals that have experience with the technology.

3.3c) Comparisons

- **MongoDB**
  MongoDB has many pros for us, each completely meeting our requirements. MongoDB is very scalable, especially horizontally, meaning it replicates easily to use at other business sites. This factor is huge for us, in that MongoDB is built to scale *by design*. Another pro for MongoDB is it's high consistency and performance, so that mixed workloads (reading and writing) are not an issue, and guarantees to return the most up-to-date value on a read. These capabilities guarantee a business-friendly system that will not report old data that could harm company operations, and will be able to handle the amount of traffic associated with a successful business. Finally, MongoDB is an open-source, document-oriented database, making it free software, and utilizes a flexible data model to meet our client's changing business. A summary of these benefits are shown below with diagram 6.

Some cons for MongoDB are it's searching capabilities. MongoDB supports basic text searching in the database, but nothing complicated, and is not designed to be a search engine, running large amount of searches all the time. But since our app will not need this type of feature, we can afford this weakness.

MongoDB

Property	Description
Horizontally scalable	Quick and easy sharing and replication
Highly consistent/performant	May read immediately after writing; handles mixed workloads
Flexible Data Model	Ideal for a growing business with growing needs
Document-Oriented	Stores JSON to work easily with web frameworks
Open-source	Affero General Public License v3.0 (AGPL v3.0)

Diagram 6: MongoDB at a glance

- **Azure DocumentDB**
  Azure DocumentDB was another strong consideration, since they offer a large suite of products that works easily together, so depending on choices for other technologies, Azure may be our project database. Some pros that DocumentDB has are similar to MongoDB, such as being document-oriented, consistent, and scalable (while not as scalable as MongoDB), as shown in diagram 7. So this fits many of our client's requirements.

  The main con for Azure products is that an Azure account is required, meaning our client will be purchasing the software by a usage rate. For DocumentDB, pricing is charged by capacity and request units together, request units being amount of reads and writes per second.

Azure DocumentDB

Property	Description
Horizontally scalable	Quick and easy sharing and replication
Highly consistent	May read immediately after writing
Flexible Data Model	Ideal for a growing business with growing needs
Document-Oriented	Stores JSON to work easily with web frameworks

Azure Account Required	Charged by Request Units (RUs) and capacity (GB)

Diagram 7: Azure at a glance

- **CouchDB**
  CouchDB was another popular alternative to MongoDB, and had many business cases for switching to CouchDB from other companies, so it is definitely worth consideration. There are many pros to Couch: it is scalable the same way as MongoDB, though some think Couch replicates easier, it is document-oriented and flexible, and it is open-source. These meet many needs from our client, but there is one primary difference with Couch from MongoDB.

  CouchDB's major problem is that it is highly available, rather than highly consistent. The difference between a highly available and a highly consistent database is that a highly consistent database will always guarantee returning the most up-to-date value, even right after writing the update. Highly available databases have eventual consistency, meaning a read immediately after updating may still return the old value, which could lead to some confusion or users not receiving the proper videos.

CouchDB

Horizontally scalable	Quick and easy sharing and replication
Highly available	Can always read for a document, eventually consistent
Flexible Data Model	Ideal for a growing business with growing needs
Document-Oriented	Stores JSON to work easily with web frameworks
Open-source	Apache License 2.0

Diagram 8: CouchDB at a glance

These choices meet the majority of our requirements in a database, so we had to compare them on a smaller level than just overall capabilities. Azure DocumentDB has been proven to be a less mature version of MongoDB, with less-sophisticated indexing, as well as less query language support. Azure is also a paid service, marking it lower than MongoDB again. We have considered Azure services for storage as well, and since DocumentDB works with other Azure products, DocumentDB would have complemented our storage solution, had we chosen Azure Cloud over Amazon's S3 storage service.

CouchDB is a popular alternative to MongoDB due to its' high availability, rather than consistency, which may fit some business cases for other clients. Our application will need to be consistent so that data is always up-to-date for customers and tutors using the portal. In other

ways, CouchDB and MongoDB are very similar, but we additionally considered that Alex and Tyler already have work experience with MongoDB, so it is our top choice for database technology.

3.3d) Conclusions

Pending input from our client, our top choice in database technology is MongoDB. In Diagram 9 below, you can see some of the considerations that led us to this decision. MongoDB is a NoSQL database that horizontally scales, which fits our client's scaling needs, as well as being very quick and simple to replicate, which assists maintenance and technology upgrades. MongoDB is a consistent database, so an update is guaranteed to be returned by a read operation for any given document. MongoDB is also open-source and uses JavaScript and JSON, and we have team members with experience using this technology, so it is easily our primary selection.

	NoSQL	Consistent	Scalable	Open-source	Web-Friendly	Experience
MongoDB						
DocumentDB						
CouchDB						

Diagram 9: Table comparing database features

To test our choice, we will be creating a simple web application that stores very basic data through user input. This will allow us to both test our front-end framework choice, as well as its compatibility with setting up a MongoDB collection for our data. This short application may serve as our demonstration, and will need to be discussed further.

3.4 Hosting

3.4a) Problem

The issue right now is that our client is using Wix, as an all in one hosting package. Not being very tech savvy, it is his current solution and what the site is built on. We feel that using a provider like this will limit us too much (will be discussed below), and have decided to build part of his current website, the "Submit My Assignment" portion, on a separate hosting of our choosing.

When clicking the link for "Submit My Assignment" it will take the user to the portion of the site we have built on separate hosting. This will enable us to build a better web-app for him, while also not having to rebuild the entire website. Also, it will leave our client the option to migrate away from Wix in the future, and hopefully migrate the rest of the site over to the better hosting

package that we chose. The technical challenge of integration of all our technologies also somewhat relies on this technology.

3.4b) Our Options

When looking at web-hosting, there are a few factors that are important to consider. First off is price, especially since Tailored Tutoring Co. is still a smaller company they do not have extra money to spend. Also important are speed and reliability. And maybe most importantly is compatibility.

3.4c) Comparisons

- Wix- Wix is a good starter web-hosting service. It is good for a novice web-builder because it has website builder tools and DIY options. Also, it automatically backs up your website. But the packaged plans, are definitely pricey if you're trying to build a bigger business with lots of web-traffic. It is also a little slower, and not as reliable as a cloud based service, uses shared hosting, and not as easily scalable. But overall, it is a good option and deal for smaller websites.

- **AWS EC2**- Amazon Web Services, AWS, is a cloud based web hosting service. Because it is built by Amazon, it now has a massive host-base, protection against DDOS attacks, unlimited scaling options, and would be compatible with our storage option of AWS S3, as well as being great for open-source development. However, the pricing is could get expensive if you use it a lot (using a pay as you-go, charge per hour model), and there is a bit of learning curve.

- **Azure**- Azure is also a cloud based web hosting service, which also incorporates a similar "pay as you-go" pricing plan. The only difference from AWS, is they charge per minute (which can be a bit more accurate). They also offer short-term commitments. Azure is another cloud-based service that offers great scalability, prioritizes security and privacy, and is easier to use out of the box, especially for Windows users. However, there could be compatibility problems with our AWS S3 storage option, and isn't as compatible with GitHub (which we are using for our project) and open-source development.

3.4d) Conclusions

Wix was a good starter web-site builder host for our client when he was getting Tailored Tutoring setup. However, when we look at important factors such as scalability and pricing, it just makes sense that it is time to migrate to a better platform.

We ended up deciding on AWS EC2, although it was very close between AWS and Azure. They both offer cloud-based hosting, which allows Tailored Tutoring to scale their business as it grows. And both also have a very reliable platform, while AWS has been around for longer, but

Microsoft for Azure is a trusted name. Even though it is a bit difficult to determine the pricing up-front, it seemed that AWS and Azure offered pretty similar pricing plans, and payment models, only making you pay for the services that you actually use.

It really came down to keeping things simple for us and our client. Since we are using AWS S3 for their storage, we know that AWS hosting will be extremely compatible with the storage system. And, it just keeps everything in one place and a bit easier to work with, for both us and the client.

Below, in Diagram 10, is a table with some of the key factors we considered when looking at each technology:

	Scalability	Pricing	Security	Compatible with AWS S3	Ease of Use
Wix					
AWS					
Azure					

Diagram 10: Table comparing hosting features (Light green represents partial support)

Pending our client's approval, we will try to set up both the AWS S3 storage, as well as the web-hosting using AWS EC2. We've discussed some of the pricing, and what it offers, and he does seem open to it. And also seems like he might want us to port over his whole website onto the AWS platform.
In the end, the documentation suggests that both of the AWS services are designed to work together. However, we will have to verify this once we get the go-ahead from our client, and actually build our working demo by the end of the semester.

3.5 User Interface Design

3.5a) Problem
The main problem for designing the user interface is having it mobile-friendly. We want to build a mobile-friendly web-app, that looks great on mobile devices. We foresee the end-user accessing Tailored Tutoring, and in particular the "Submit My Assignment" portal, via their phone. This will make it easier for them to take a picture of their homework problem with their phone, and then upload it directly via the web-app. For this, we really need something that is beyond "mobile-friendly", and rather a "mobile-first" design tool, as the technical challenges indicate.

3.5b) Our Options

The first thing to consider is whether we want just a “mobile-first” website, or an actual App. Our client does think he wants an App in the future, but at this current juncture just wants to get his web-site fully functioning and optimized for mobile. Once we accomplish that for him, he may choose to base an actual Native App off of our website solution. This lead us to consider a Hybrid mobile app, using something like HTML5.

3.5c) Comparisons

- **HTML5**- HTML5 is used to create hybrid mobile apps, which could be a mixture of what our client wants, but we don’t see it as being the ideal solution. Instead of giving our client a great product, we feel like we would be giving him another solution for the time being, but he would still need to recreate his website, and then create a faster app later. Hybrid apps simply are not as good as native apps. They are not as fast, reliable, or smooth as native apps. When it comes to taking pictures, and dealing with larger image and video files; we would want something fast and reliable. As an end user, it could really turn them off having to wait excessively long times to upload image files, or download and access the video solutions. For our client, he still would not be getting a fixed or upgraded website, which he and the company do need.

- **Bulma**- Bulma is built off of Flexbox, which is a framework designed for a responsive web design. It is easy to learn, very straightforward, and doesn’t have as steep of a learning curve. Being built off of Flexbox, and utilizing CSS Variables and CSS Grid, Bulma is extremely current with today’s browser technology. The grid system allows for easy layout, and there is no JavaScript included. Which means means designers can use their own to customize the SASS source files. While in our opinion we see no JavaScript being included as advantage, other programmers may see it as a disadvantage. Other disadvantages of Bulma might be a lack of customization options, as well as it only being 90% compatible with Microsoft’s web-browser, Microsoft Edge.

- **Bootstrap**- Bootstrap is a CSS framework that has been around for a while now, and has a heavy user base. We would be remiss not to consider this as an option. It has many clear advantages such as jQuery plugins, making it easy to add interaction to our website, and minimal cross-platform bugs. Also, the large community base lends itself to advantages such as more questions getting answered and more promptly, and more themes and plugins being available. Bootstrap also comes with its own disadvantages, such as JavaScript is tied directly to jQuery (which might be problematic when working with our chose JS library of React). Customization can also be tricky, requiring lots of overriding styles and/or rewriting files. Because of Bootstraps popularity, websites can start to look the same without significant customization.

3.5d) Conclusions

In the end, we wanted a tool that would allow us flexibility in designing an awesome web-based UI, but also allow us to rapidly prototype. We decided that Bulma was the best fit for us moving forward; and below, Diagram 11, is a table to summarize some of the elements that helped us reach this conclusion.

	Ease of Use	Mobile First Design	React Compatible	Large User-base	Customization
HTML5					
Bulma					
Bootstrap					

Diagram 11: Table comparing CSS Framework features

When deciding between the technologies, we seriously considered Bootstrap. HTML5 just didn't seem the route we needed to go for the project, or the client. However, between Bootstrap and Bulma, we just felt we could more rapidly prototype and use Agile Programming to show our client solutions, refactoring if necessary. Both Bootstrap and Bulma look good on and use a mobile-first design philosophy, however we also we felt that we had better customization available to us with Bulma. Also, it would be compatible with our JavaScript tool of React, to even further tailor our solution for the web-app.

Bulma is designed to be compatible with most current web browsers today, and should work in our development environment. We have already used Bulma to create our team website, which functioned perfectly, and looks really clean and professional. Using our team website as an example, we were able to see how easy it is to use this CSS Framework to create something the end-users of Tailored Tutoring would enjoy using.

To be extra sure though, we will build a simple web application that stores basic data through user input. Making sure that our front-end set up with Bulma, will in fact be compatible with our backend database selection of MongoDB. We will complete this test before fully designing and incorporating our User Interface for the Tailored Tutoring.

4) Technology Integration

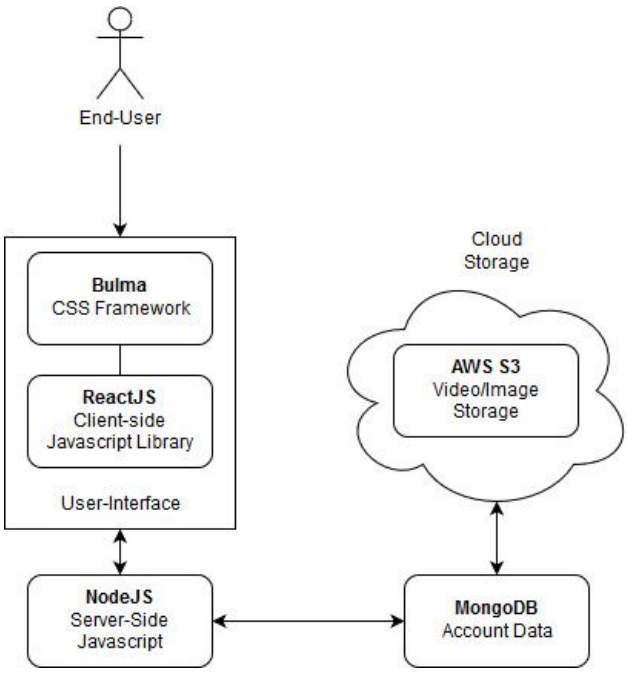

Diagram 12: Layout of the decided tech stack

The integration of our technology heavily relies on the framework picked for our project. This has been discussed in detail, but lacks more of the context surrounding the full integration of all of our technologies. While our client-side javascript library has been determined, the server-side will be the basic glue that will tie everything together. As a result the base of our application will be NodeJS. NodeJS works very well with React and so the two are often found, if not exclusively so, side-by-side. This is important in setting up a web server for our application. NodeJS will handle a lot of our server side javascript used to deliver and dynamically update our web app. This also gives us access to the Node Package Manager (NPM) which opens up an extensive and exhaustive database of open source libraries to help us with our project. Through NPM we can install libraries that can aide us in fully integrating all of our different technologies. The most useful of which will be when we are consuming the data from our database. We have decided to use a RESTful API when accessing data from our database. This means we will have to create an endpoint through NodeJS to consume and update our data from. NPM contains many useful tools for us to do this. There also exists an NPM package for the AWS-sdk, which will be useful when attempting to access the videos we have stored there. Lastly for things like our UI, we can use NPM to install our CSS framework for our UI. NodeJS will be how we tie everything together once our major technological components have been established. With its powerful package manager we can find a solution to all the little details that are too narrow or to specific to be handled by any other technology. Both utilities will allow us to connect all the pieces to complete a comprehensive whole. The above diagram shows a high

level concept of our current tech stack displaying how all the parts fit together to create our finalized system.

5) Conclusion

In summary, we, Business Web Solutions, have been tasked to build an automated solution to our client's, Tailored Tutoring Co., picture-submission and then video-solution process for their online tutoring platform. We plan to solve this problem by building an online GUI interface and create profiles for both sides of users, the clients/students and the tutors (also admin profiles). These profiles will incorporate pertinent details on the students and tutors, and automatically notify corresponding tutors when a problem is submitted, and then automatically notify the student once a solution has been posted, all via email notifications.

The purpose of this document includes going over the high-level business problem and solution; but it is to mainly focus on the technologies we plan to use to build our solution, and the feasibility of how these technologies will work together to accomplish this goal as well as any problems we might be able to foresee and avoid by doing our research. In the Technology Analysis section, we discussed the different components our tech needed to address, as well as different possible technologies we could use to accomplish those goals and the efficacy of each choice. Finally deciding on our particular technology choices and then how they all fit together being discussed in the Technology Integration section. As a refresher, Diagram 13 below is a table of the technologies that we chose and discussed:

Tech Challenge	Solution	Confidence Level
Storage of large amounts of data, in the form of videos and images	**AWS S3**	Very High
Store customer/student and tutor profile information, via a database	**MongoDB**	Very High
A service to host our website	**AWS EC2**	Very High
Mobile-friendly design of web-application and user interface	**Bulma**	Very High
A lightweight framework to integrate all of our technologies together, completing the web-app	**React**	High

Diagram 13: Overview of our solutions and our confidence levels

When designing any larger-scale technological solution, there will always be minor problems or complications that occur. However, to quote Ben Franklin 1736: “an ounce of prevention is worth a pound of cure.” While we hope, but maybe do not expect, that everything runs smoothly when building out our prototypes and automated solution for Tailored Tutoring; we know that we have thought about and thoroughly investigated the technical solutions we have decided on. Also, we are fully confident, that utilizing these technologies, we will build out a solution that all parties involved will be happy with: the end-users, our client, and us as the creators. In the end, providing a solution and service that enriches both our client’s and users’ lives.