Technological Feasibility Analysis

9 November 2018

Team Jasper
Jabulani School Simulation Portal

Sponsor: Dr. Gretchen McAllister

Mentor: Ana Paula Chaves Steinmacher

Team: Karsten Nguyen
Carli Martinez
Ruben Rincon
Jasmine Mitchell

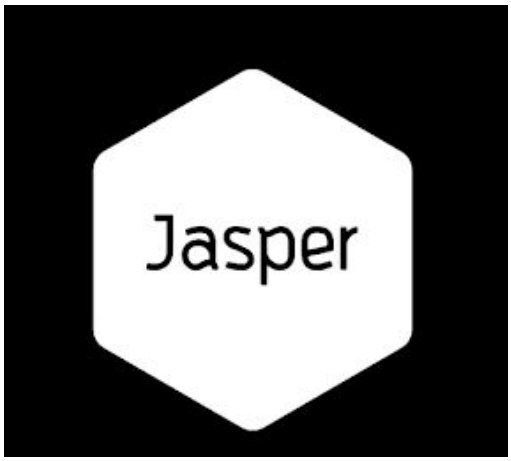


Table of Contents

Technological Feasibility: Content

Introduction

With today's ever increasing diverse population, there is now a new inquiry into how instructors can embrace the diversity of their student body — whether it is between a traditional face-to-face class, or one that is taught online. Current research suggests that diversity in a traditional classroom is a powerful asset, providing that the instructor is sensitive to individual students' backgrounds. However, it can prove difficult to deal with the diversity gap between students and teachers. To allow these teachers to engage with their students respectfully, teachers must know their students and their academic abilities individually in order to be able to respond in a culturally, socially, and linguistically appropriate manner. The best approach for teachers to obtain knowledge for handling specific diversity-related circumstances is by connecting to the experience on a personal and professional level of students with various backgrounds.

Our sponsor, Gretchen McAllister, is a Professor of Education at NAU. While teaching abroad in South America, she contemplated the idea of developing a school simulation portal that could amplify and fully encompass diversity sensitivity in an academic setting. This portal, appropriately named "Jabulani" — the siSwati (Kingdom of Swaziland) word for happiness, is to made to lay the foundation for the future of diversity training in academia. McAllister's idea of the portal is to highlight a few of the key challenges and concerns regarding diversity, and illustrate ways to gain an understanding of diversity in the classroom and beyond. So far, McAllister has accumulated 600 virtual student profiles in an Excel sheet, expanding many different diverse backgrounds. Along with these students profiles, she has also developed scenarios where these "teachers in training" will be able to interact with multiple diversity situations and address them accordingly. The concept of the portal is to allow both faculty and students to log in to the portal; faculty members can drag and drop from the online virtual database to create their own virtual classroom and assign scenarios and exercises to the students accordingly.

To address these challenges, we have been working on developing a virtual training space called "The Jabulani School Simulation Portal". This portal will allow teachers in training to access potential classroom scenarios and allow them to address the scenarios in the most appropriate manner. The solutions to such scenarios will be reviewed by the admin or instructor, since these responses are never as simple as "right" or "wrong". To enable this portal we are going to create a web application where faculty members and students can login to access this virtual classroom. Faculty

members will be able to deploy a virtual classroom to further expand the student's academic experience. One of our tasks is to convert the 600 virtual student profiles excel spreadsheet into a online database where the faculty members can filter out or drag and drop virtual students into their own customizable virtual classroom. Along with their customized classroom, faculty members will be able to assign exercises and scenarios to them. From there, students will be able to login and interact with their assigned virtual classroom. As of now, Dr. McAllister resorts to using Google Classroom with the collected data to create simplified diversity scenarios/assignments. Our goal is to design a system that allows educational faculty to create and deploy individually customized classroom simulations, which are then used as the basis for a series of training exercises that allow education majors to gain hands-on experience with diversity issues they will typically face in a diverse classroom.

Our portal will be similar to that of Bblearn when it comes to authentication and system initialization. Students and faculty will be able to log in to the portal with their credentials and will be granted separate privileges. Faculty will have admin controls, where they can set up their virtual classrooms and teachers-in-practice will be able to self-enroll in one. From there, faculty will have the privilege of grading their exercises and enable them to advance.

This document will assess the details of how we intend to deliver our product as well as analyze what is technologically feasible given our scope. Our technological feasibility will be useful to organize the information properly and highlight proposed technology.

Technological Feasibility: Scope

Some of the technological feasible major design aspects that we envision include:

- **Programming and scripting languages**
- **Database technologies**
- **Hosting**
- **User Authentication**

For this project, given the many tools and resources we plan to use, we will effectively start ‘from scratch’ but we will not reinvent the wheel. We will make the most out of our languages and tools so we will only spend time developing what is custom to our requirements. For example, for styling, we may use a CSS framework, such as a Bootswatch, that will automatically style our web pages, but we will also use custom CSS stylesheets in cases where we need specific items styled per request of the client. We believe that with this approach, it will be easiest for our team to grasp a solid understanding of our project code since its inception began with us.

Technological Challenges

The technological challenges of our project fall under three categories, front-end development, back-end development, and hosting. This web application will need an attractive and well-designed interface, a robust and functional back-end complete with a modular database, as well as a web host that will handle the application's needs as well as the client's.

Front-End Development

The goal of our front-end technology is to provide a user-interface that is both user friendly and accessible for users with disabilities. In addition, our front-end must display a virtual classroom along with its functionality. Administrators must be able to drag and drop student profiles into a virtual classroom from a given database. When the classroom has been created, admins will be able to assign exercises and scenarios related to the virtual classrooms. The teachers in practice will be able enroll in these virtual classrooms, and work on the assigned scenarios and exercise. After they have worked through a given amount of exercises, they will have to wait for their input to be reviewed by the admin. Once they have been approved by the administrator, they can continue to progress through the virtual classroom. All of the profile data used to create the scenarios should be able to be extracted from the database by the administrator. This issue also presents UX and UI challenges, and must be easy for non-technical users to quickly grasp and understand.

Back-End Development

To develop our web application, we must select a back-end language that is capable of meeting the current and future needs of the client. Ideally, it would be easy to learn and work with for our team to develop in, have a lot of resources and a large community around it, and be flexible.

Faculty must be able to access a database that will be used to handpick a classroom for the teachers in practice. Using database technology, teachers in practice must also be able to enroll in one and only one classroom given by faculty. CRUD (Create, read, update, and delete) functionality will need to be performed on database values. The admin will also be able to have a search filter option where that helps the user select a subset from the database. Faculty must also be able to add exercises to the webapp.

The webapp will need user authentication and a secure interaction. Faculty must be able to restrict/allow access to profile data based on scenario completion. Teachers in practice must be able to interact with each virtual student as a clickable link. The scenarios will need to have sharing capability among a community.

The provided excel sheet containing student profiles incorporated in the previous iteration of Jabulani will need to be extracted into the online database. This is so the data can be utilized by the database and manipulated by the user. This will be a one-time process and profiles included later will need to be added to the system internally.

Hosting

Our final challenge is to find suitable hosting for our web portal. We we will need to find somewhere to host the web application that will be compatible with our chosen web framework as well as our chosen database. The host must also provide quality performance with little to no deterioration in speed when many people are using the webapp simultaneously and performing many tasks. This will require a host with a notably decent bandwidth. In addition, the host must be free yet provide the aforementioned features.

Technology Analysis

A back-end language/framework must be one that is easy to work with and learn, and be capable of meeting project requirements. For this analysis, we chose three of some of the most popular technologies for web applications.

Back-End Functionality and Language

Alternatives

Ruby on Rails:

Ruby on Rails is a popular framework for building web applications that one of our team members has had success using in the past. Ruby on Rails, or Rails, is a server-side web application framework written in Ruby under the MIT License. Rails is a model–view–controller framework, providing default structures for a database, a web service, and web pages

PHP:

PHP is a general-purpose scripting language that is especially suited to server-side web development, in which case PHP generally runs on a web server. Any PHP code in a requested file is executed by the PHP runtime, usually to create dynamic web page content or dynamic images used on websites or elsewhere.

ASP.NET:

ASP.NET is an open-source server-side web application framework designed for web development to produce dynamic web pages. It was developed by Microsoft to allow programmers to build dynamic web sites, web applications and web services

Chosen Approach

The chart below will establish a fundamental comparison between the three back-end frameworks/languages, Ruby on Rails, PHP, and ASP.NET.

1 = Poor **3 = Best**	**Ruby on Rails**	**PHP/**	**ASP.NET**
Overview	A web-application framework that “includes everything needed to create database -backed web applications” (3)	A popular scripting language designed for web development with a long history and large community around it. (3)	A web application framework developed by Microsoft to allow users to build dynamic sites. (2)
Learning	A team member is experienced and comfortable in development with this language. (3)	A team member is experienced and comfortable in development with this language.(3)	A team member is experienced in development with this language.(2)
Speed	Medium (2)	Medium (2)	Fast (3)
Availability of Resources and Information	Medium (2)	High (3)	Medium (2)

Set up and development time	Fast (3)	Medium (2)	Slow (1)
Availability of testing frameworks	Many (3)	Some (2)	Few (1)
TOTALS:	16/18	15/18	11/18

For a back-end language, we have chosen Ruby on Rails due to our team's comfortability with it, its growing popularity, and ease of use. It is also compatible with a large amount of frameworks and libraries to use alongside it, including many testing framework options for test driven and behavior driven development.

Testing

We installed the latest version of Ruby on Rails and scaffolded a starter project. We ran it on a local server following the completion of tutorials we found online.

Back-End Functionality and Modular Database

Alternatives

Azure SQL:

Microsoft Azure SQL Database is a managed cloud database provided as part of Microsoft Azure. A cloud database is a database that runs on a cloud computing platform, and access to it is provided as a service. Managed database services take care of scalability, backup, and high availability of the database

PostgreSQL:

PostgreSQL is a general purpose and object-relational database management system, the most advanced open source database system. PostgreSQL was designed to run on UNIX-like platforms. However, PostgreSQL was then also designed to be portable so that it could run on various platforms such as Mac OS X, Solaris, and Windows.PostgreSQL is free and open source software. PostgreSQL requires very minimum maintained efforts because of its stability. Therefore, if you develop

applications based on PostgreSQL, the total cost of ownership is low in comparison with other database management systems.

Chosen Approach

The chart below will establish a fundamental comparison between the two database technologies, Azure SQL and PostgreSQL.

1 = Poor **3 = Best**	**Azure SQL**	**PostgreSQL**
Overview	Microsoft Azure SQL Database is Microsoft's cloud computing database platform. Rather than creating a whole database, access to it is provided as a paid service from Microsoft. (1)	PostgreSQL is a general purpose and object-relational database management system, and like Ruby on Rails, it is open source. It is portable, and can run on various platforms. It is is free and has been used with Ruby on Rails in that past by one of our team members. (3)
Perceived Learning Curve	Potentially large and time-consuming learning curve. (1)	Easy to implement with Ruby on Rails with tutorials available. Successfully used in other projects. (3)
Cost	Is not free. (1)	Free. (3)
Database Compatibility	It is compatible with our language, but not our hosting platform. (2)	It is compatible with our language, and our hosting platform. (3)
Performance	Depending on the network environment, we may not be able to connect or we may lose the connection if the SQL Database server doesn't allow traffic from our client IP address. (2)	Writing in test-data must be done via the command line until the forms are set up in the application. (2)

TOTALS:	7/15	14/15

For our database, we chose PostgreSQL due to its compatibility with both Ruby on Rails and its modularity. We will import the Excel dataset in through the command line into a PostgreSQL database. The database will then be controlled through Ruby on Rails' Active Record, which generates SQL commands from written 'plain english' directives, stores development history, and has a database schema that is viewable right in the project files for easy access. It is quicker than writing SQL commands, it has been used in the past successfully by one of our team members, and is easy to get started with.

Testing

We installed PostgreSQL and consequently followed tutorials describing how to connect the two Previous similar projects that use these two together were used as a reference. To test it is working, we used Active Record to interact with the PostgreSQL database by running initial migrations and commands to begin with a simple 'user' table.

Front-End Functionality and Virtual Classroom

The front-end functionality of the web application must support accessibility for various forms of disability. It must be suit our needs by having helpful documentation, being compatible with common web browsers, being familiar among team members, and providing compatibility with our chosen server-side language, Ruby on Rails. Since the project's UI/UX will be improved upon and maintained in the future by others, it is also a central goal for the team to produce clean and attractive code to read. Fulfilling these requirements will allow us to best support our clients goals and vision of the virtual classroom.

Alternatives

HTML, CSS, & JavaScript:

Often referred to as a "triad of cornerstone technologies for the World Wide Web", HTML, CSS, and JavaScript work together to produce the bare-bones of many websites

and applications used today. HTML is able to provide structure to a website while CSS enhances this stylistically. JavaScript operates on the client side to enhance user experience (UX).

Everything else:

Haml (HTML Abstraction Markup Language) is a templating system that is built upon HTML and lets the programmer write more human-readable code. This yields cleaner and easier to read code since it has been abstracted for that purpose. SASS is a superset of CSS that is based on JavaScript meant for creating a clean and appealing user-interface.

Chosen Approach

The chart below will establish a fundamental comparison between the two front-end development languages, HTML/CSS/JavaScript and Everything else.

1 = Poor **3 = Best**	**HTML, CSS, & JavaScript**	**Everything else**
Overview	The most popular and fundamental technologies of the web (3)	Very unpopular and potentially outdated web technologies, difficult to find as a reputable competitor to HTML, CSS & JS - almost nonexistent (1)
Has thousands of tutorials and resources	Yes (3)	Few (1)
Compatibility with web browsers	Compatible with many web browsers, and many versions (3)	Not compatible (1)
Team familiarity and experience using before	Very familiar and experienced with (3)	Unfamiliar with (1)

Compatibility with our chosen server-side language, Ruby on Rails	Compatible (3)	The CSS alternative, SASS is compatible with Ruby on Rails, however, it is build upon CSS. Other alternatives provide unknown compatibility (1)
TOTALS:	15/15	5/15

Based on this ranking system, will use HTML for defining the logical structure of a web page; CSS for specifying the appearance of each element on a web page; JavaScript, the programming language that brings web pages to life; The HTML5 canvas element, which provides a space on a page where your code can draw using JavaScript instructions; and HTML input controls such as buttons and sliders for user interactivity.

We will also utilize JavaScript for the fact that it is currently being used by more than 94 percent of all the websites. JavaScript is useful because it is able to be a front-end programming language which helps web developers in Web Application Development and makes dynamic and interactive web pages by implementing custom client-side scripts.

Due to our client's plans to introduce a team to the project to take over the styling of it next year, we believe HTML, CSS and JavaScript will fit perfectly to meet our requirement to produce a project with simple and basic styling and appearance that can be built upon in the future.

Testing

These front-end languages have built-in compatibility with our Ruby on Rails project. To test them, we wrote a simple view in HTML with a starter stylesheet and JavaScript and ran it on a local server to view and make sure they are operating correctly. Ruby on Rails also has views that we can generate from the command line during scaffolding; we decided to try that as well to test proper functionality of our chosen front-end languages.

User Authentication

This project requires the ability for users and administrators to securely create an account on the JASS system, log in, be able to set and reset their password, and have their password safely stored. They will also be able to manage their account, updating their username, password, and any other information related to their account. This presents a large security challenge, especially on a public software that is hosted online. This also must mirror previous user authentication systems that many of our users have seen before, to minimize any confusion.

Alternatives

“Devise” Ruby Gem:

Compared to other ruby gems like authlogic, Devise is the most-popular and well-established, and also has the most support for it. A team member also has experience using it successfully and is knowledgeable in how it works, and that it can meet the requirements.

Hard Coding:

An alternative to using a gem like Devise, if we find the gem to be too constricting to build out the complex requirements we must implement, is to create our own user authentication, carefully coding the password hash algorithms and all related functions. This may prove to be difficult and time consuming compared to Devise or other gems, and it may also cause security vulnerabilities if we do not successfully encode and decode the password hash properly. There is, however, plenty of support and resources we can utilize if we end up needing to go this route.

Chosen Approach

The chart below will establish a fundamental comparison between the two user authentication possibility, Hard Coding and Devise.

1 = Poor **3 = Best**	**Hard Coding**	**Devise**
Strong adherence to	For both front-end to back-end connections and	Contains a lot of different configuration options.

standards	default account settings, alternate decisions must be made at design time. On many systems, a default administration account exists which is set to a simple default password which is hard-coded into the program or device. (3)	Devise views directly into your application folder. A new folder called *devise* will be created inside the *views* directory (3)
Large and active community	Most companies use third party application to create authentication protocols. There is still a large community who hard code SQL and PHP authentication (3)	53 companies on StackShare use Devise. Devise has a large and active community. (3)
Great amount of helpful tools and libraries	There aren't many tools to use when coding login and authentication. You are also limited in libraries (1)	There are hundreds of different helpful community-created "gems" and libraries that you can use as a part of your own software.(3)
Performance time	The boot time of the framework is quite long, especially when you work with a massive project (2)	Devise is designed to handle large amounts of data from the ground up. Compiled mapping reduces time/space at run time (3)
Security	If hard-coded password encryptions are used, it is almost certain that malicious users will gain access through the account in question. (1)	Third-party plugins and gems are great for extending a website in a quick and efficient manner. Sometimes security requirements require a

		more robust solution than what Devise provides out-of-the-box. (2)
TOTALS:	8/15	14/15

Since we are using Ruby on Rails, we will explore the many ruby gems available to us that are popular, with a proven and tested codeset, and is up-to-date with fewest bugs. One gem that a team member is familiar with is Devise, and it has met all of these requirements. Upon identification of a gem we will use, we will add it to our Gemfile and follow instructions and documentation to implement all of our project requirements with it.

Testing

After familiarizing ourselves with the tutorials of Devise, we will install the gem into our starter Ruby on Rails project and follow steps to get started. We will test it by going into our portal and writing in sample accounts and signing in and out of them, as well as forgetting passwords and following steps to get a new password. If necessary, we will also use other similar projects that use Devise as a reference to help us understand what successful implementations of it should look like.

Web Application Hosting

Hosting our application on the web must be done by a service that is capable of meeting our needs, is easy to work with, and has agreeable prices. It should preferably have decent support for our chosen frameworks and databases along with reliable bandwidth speeds. Most of these issues can be solved with a reliable host.

Alternatives

Heroku:

Heroku is a cloud “platform-as-a-service” that allows for web hosting. A free account provides a small 512MB of RAM for performance needs. Heroku mainly supports PostgreSQL databases and it is complicated to overcome that limitation. Workarounds

for PostgreSQL on Heroku do exist but we'd be unlikely to use them as one of our team members is experienced using PostgreSQL databases. Heroku has good support for a variety of frameworks such as Node.js and Ruby.

NAU-provided Server Hosting:

NAU is already able to provide some of its server resources and web-hosting capabilities to students for free. Since our client is NAU faculty, it may be possible to request even more resources than usual. Bandwidth and performance would generally be fast and stable as it is mostly supported by NAU's IT department. NAU offers Unix and Windows web-hosting. Unix hosting has support for a MySQL client and database with Apache 2 framework support. Windows hosting uses ASP.NET for framework support but is limited to Oracle and Microsoft Access database support. Microsoft Access and Oracle would require some time to learn as our team does not have much experience with those databases.

Firebase:

Firebase is both a mobile and web-application hosting/development platform owned by Google. Even a free account provides generous benefits: 1GB each for hosting data, cloud storage data, and realtime database storage along with 5GB of regular storage. Authentication and cloud services are also provided but Firebase bandwidth is limited to 10GB a month handled by Google Cloud. In addition, there isn't much in the way of framework support for Firebase; you must use Firebase web-hosting tools and the Firebase Realtime Database. Both of these would require some team investment into learning time especially since the Realtime database is not a SQL-like relational database.

Chosen Approach

The chart below will establish a fundamental comparison between the three host servers: Heroku, NAU servers, and Firebase.

1 = Poor **3 = Best**	**Heroku**	**NAU Servers**	**Firebase**
Feature Availability/Cost	Offers a variety of supported features (3)	Offers some, but limited features (2)	Offers good features and storage (3)

Database Compatibility	Mostly PostgreSQL which one of our team members knows well (3)	Unix: MySQL Windows: Oracle + Microsoft Access (2)	Firebase Realtime Database (non-relational) (1)
Learning Curve/Difficulty	Our team is experienced in some of its supported features (3)	Requires learning most of its frameworks/DBs besides MySQL (2)	Requires learning its unique database and front-end tools (2)
Framework Compatibility	Good support for some frameworks but mostly Ruby (2)	Forces usage of Apache2 or ASP.NET (2)	Must learn to use its custom framework tools (1)
Bandwidth/ Performance	Provides fast but small 512 MB RAM (2)	Good amount of bandwidth/reliability from NAU (3)	Very generous amounts of allotted traffic/bandwidth (3)
TOTALS:	13/15	11/15	10/15

Heroku appears to have the best balance of features and is what we've chosen as our main hosting solution. It supports our chosen framework of Ruby, since Heroku has many features that were built to work with Ruby. It also has seamless PostgreSQL integration which won't be too difficult to learn as our team is already familiar with MySQL databases. This will be a problem if we need to use a database that isn't PostgreSQL. Heroku gives enough resources for us to test a small web portal as necessary, however 512MB of RAM might require our project to be ported to a larger hosting service. Should our hosting requirements change and we need more resources/bandwidth, NAU's servers will be the next best hosting service but will require us to spend time fully learning its databases and frameworks.

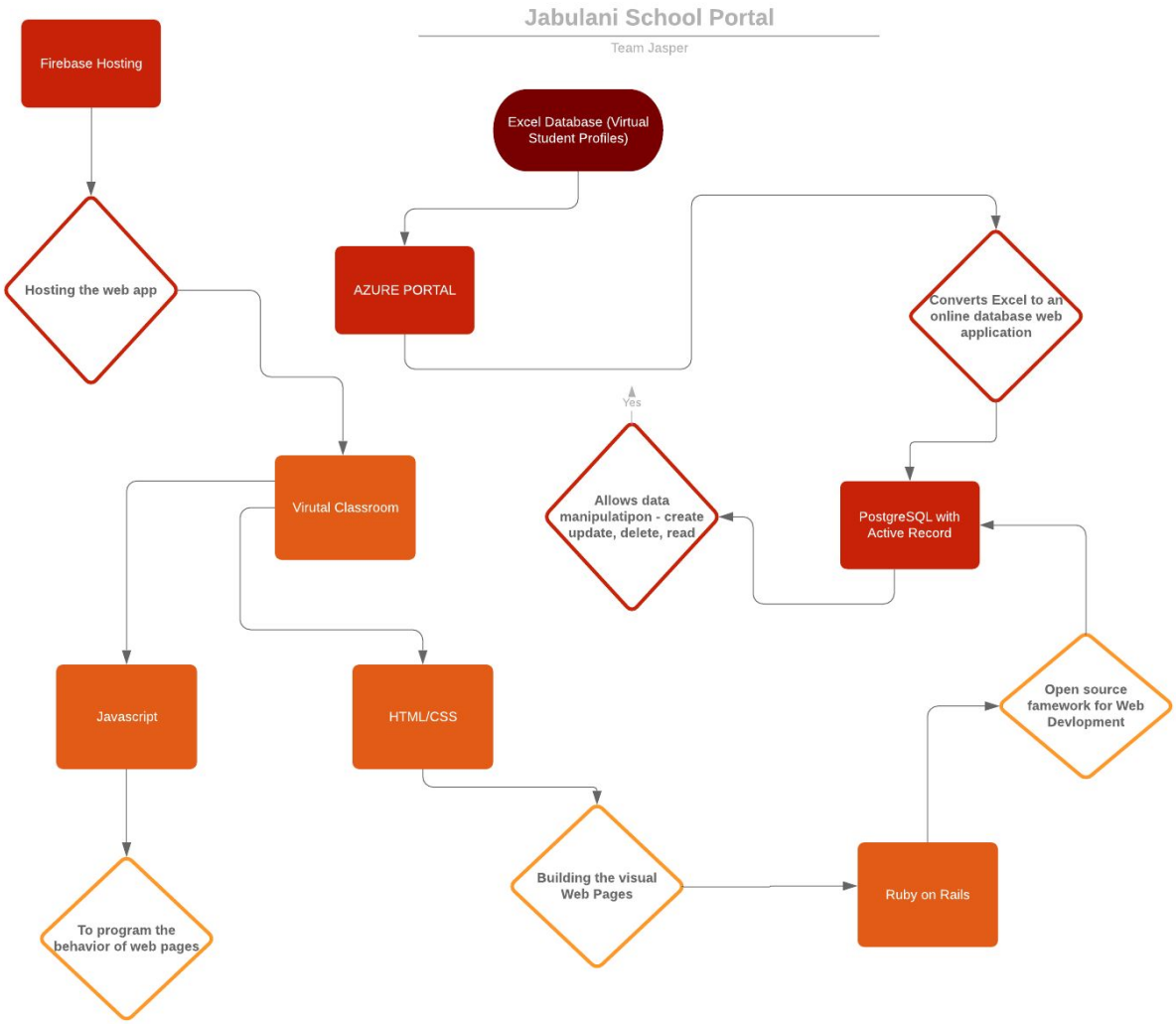
Testing

One of our team members has experience with and has tested Heroku's hosting service with small sample sites and PostgreSQL data entries, so using this knowledge to host a sample portal of our project on Heroku shouldn't be too much of a step up. If we decide

we need to port our site to NAU's servers, some of our team members already have experience working with NAU's framework tools to upload and manage sites, and will likely be able to transfer them over after testing database entries and small web pages.

Technology Integration:

- We are planning on using Heroku to host the project, since it is free, has good framework support, and is scalable.
- We are using Ruby On Rails to work with the backend development and maintenance, and HTML/CSS/Javascript for frontend development.
- We may also use various frameworks, libraries, and Ruby gems along the way where necessary.



Conclusion

This has been a Technological Feasibility Analysis of the JASS portal project to be built by Team Jasper. We have analyzed the feasibility of our project requirements and have determined them all to be fit for implementation without any major issues. We also compared technologies and established the pros and cons of each, ultimately arriving at a decision about what we plan to use. With Ruby on Rails for the back end for commands to our PostgreSQL database, HTML, CSS and Javascript for the front-end, and Heroku as the host, we are confident that this platform will work successfully to help us carry out each of our project requirements. We have summarized our technological challenges into a table with their description and how we plan to solve them with our chosen technologies:

Challenge	**Solution**	**Confidence Level**
Constructing a modular database capable of holding hundreds of student profiles and other data that can be easily accessed in our project.	With PostgreSQL as our database, and Ruby on Rails' Active Record acting as an easy way to interact with it, we will use this combination due to it's proven functionality and easy set up, as well as it's helpful online support and documentation.	High
Creating a virtual classroom that matches our client's stylistic vision and appears correctly across multiple web browsers. It also must minimize confusion and include a very friendly user interface. Much thought must be placed in the UX as well.	We will use the most common front-end technologies used by web browsers and developers alike, HTML, CSS and JavaScript. With these powerful languages, there are extensive tutorials and information available online that we can use.	Moderate; Additional frameworks or libraries may need to be used, i.e. SASS
This project requires the ability for a user or administrator to safely and	Instead of writing our own code for this, we will use a dependable ruby gem that is safe to use and proven to work. This	High

securely create an account, log on, update their profile, and log out. This part of the project can result in a significant security vulnerability if not implemented correctly.	solution also saves a considerable amount of time in this area of our project since we are not manually encoding and decoding a password hash.	

Overall, we are confident in our solutions, but we remain ready and adaptable to whatever new challenges may arise along the way. Our team has tremendous coding and problem-solving skills that we will utilize to the best of our abilities to ensure a successful outcome in the development of a high-quality and impactful software.