Computer Science Senior Capstone

CS476 - Requirements Engineering

ViralTech

Project: PIMpoint Collaborator Data Entry Point

Technology Feasibility Analysis

Overview:

The purpose of this document is to highlight the high-level requirements of our project by presenting technological and integration challenges, and give ourselves the chance to explore evaluations and solutions.

Team Members:

Jialei Chen

Carl Porter

Colton Spector

Weiheng Su

Scooter Nowak - Capstone Mentor

Client:

Jonathan Todd - Associate Bioinformatician

Northern Arizona University

TGen North

School of Informatics, Computing, and Cyber Systems Flagstaff, Arizona

October 22, 2019

Table of Contents

1.0 Intro

As humans, we can only ever see the world in abstractions. The definition of those abstractions changes depending on how versed we are in the specific domain we are abstracting. Non-medical professionals for example think of cancer as a single disease with a single treatment: chemo therapy. Medical professionals have a higher definition abstraction and see cancer as many separate diseases with separate treatments depending on the type of cancer. However even this abstraction is still an abstraction. No human can perfectly conceptualize the exact genetic structure of any single disease and it's exact interactions with any single patients DNA. While no human is capable of doing this, maybe us humans can create machines that can get pretty close to that.

TGEN, which stands for Translational Genomics Research Institute, is a non-profit organization dedicated to treating diseases not at an abstract level, but with razor sharp precision. They work tirelessly to break down and understand the genetic components of common and complex diseases. The amount of variables among diseases and people is so tremendous that one treatment can't possibly work for everyone. And so TGEN is working towards being able to genetically analyze the actual disease afflicting a person and prescribe the best treatment on a patient by patient basis. However the processes currently in place at TGEN aren't as efficient as they can be to allow the scientists to dedicate as much of their time as possible to actual science. This is where our team, Team Viraltech, comes into play.

Team ViralTech includes: Jialei Chen, Carl Porter, Colton Spector and Weiheng Su. Our project is PIMpoint Collaborator Data Entry Point. The sponsor for this project is Jonathon Todd, an associate bioinformatician at TGen North.

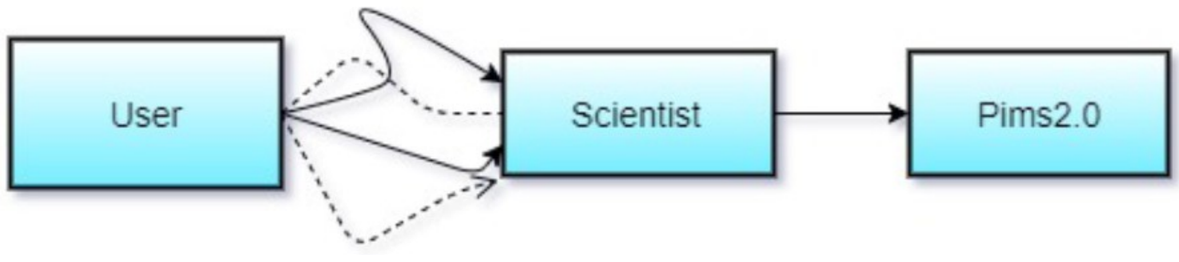


Figure 1: TGen North current data entry workflow

The process for collaborators submitting data involves them sending the data in various formats directly to the scientists at TGen north for them to input into the database (Figure 1). Currently this processes is problematic for two main reasons:

1. **Because data entry is non-standardized, scientists often have to track down or clarify necessary data that wasn't included properly in the submission**
2. **Scientists waste time every day manually entering data into their database.**

To combat this problem our team is designing a configurable web platform to collect data from contributors in a standardized format, then automatically upload that data to the database (Figure 2). The platform will also produce a printable bar code that the contributor can attach to their samples so that the scientists at TGen are able to quickly scan the sample and pull up all the information that they need. Problem 1 will be solved by creating a formatted spreadsheet that defines what fields of data are needed for the sample. Problem 2 will be solved by our website uploading the data to the database and by the attached bar codes allowing scientists to grab a sample and immediately start working on it. In addition to solving the two main problems our software will add a couple of new features that should make a positive impact on TGens workflow:

1. **Collaborators will be able to log into the website to track their samples progress through the various TGen processes**
2. **Our website will produce a shipping label that the collaborators can use to send their packages to TGen**
3. **Our software will be configurable with a text file, allowing our website to function for any amount of packages containing any amount of items going through any amount of different processes.**

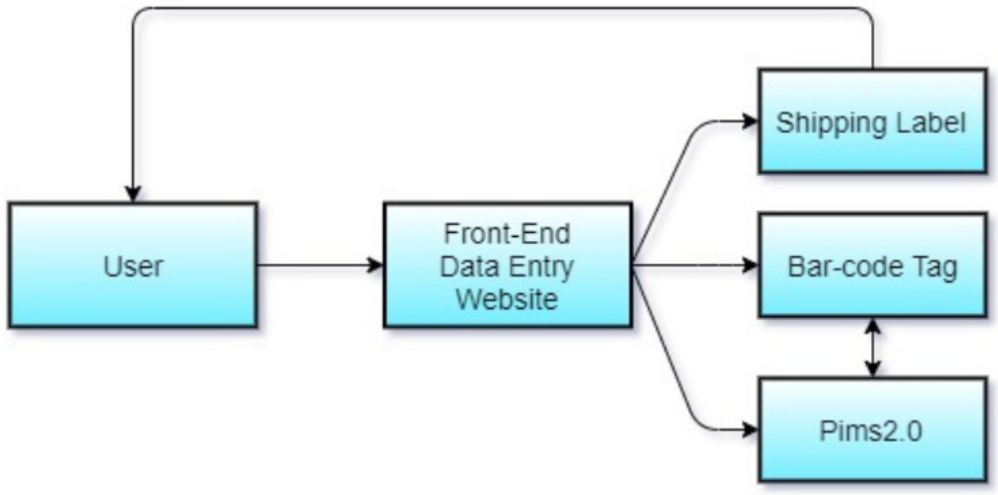


Figure 2: Our proposed solution

1.1 Document Purpose

This document is the Technological Feasibility document for our PIMpoint Collaborator Data Entry Point project. The purpose of this document is as follows:

1. **To outline the major technological challenges facing our project**
2. **To analyze the possible solutions to each one of our technological problems**
3. **To compare those solutions via objective measurements**
4. **To choose the best possible solution to each of our technological challenges**
5. **To determine how those solutions integrate together and into our clients system**

This document will be successful if it’s able to:

1. **Comprehensively identify all the technological challenges necessary to overcome for the success of the project**
2. **Choose the best solutions for those technological challenges**
3. **Verify that all of those solutions integrate smoothly with each other and with the clients system as it exists now**

If this documents process is thorough and comprehensive enough then the actual implementation of the project in the following semester will be significantly smoother.

The organization of this document is fairly straight forward. First, in our **Technological Challenges section**, we will outline and analyze all of the technological challenges facing our project. Then, in our **Technology Analysis** section, we will break down each and every one of our challenges and identify the possible solutions. The pros and cons of each solution will be discussed and weighed. After each solution has been weighed we will decide on which solution to implement in our project. To check our decision we will prove that that solution is implementable with small demos. Finally, in our **Technology Integration section**, we will go over the possible issues of integration. We will map out the basic conceptual structure of our system and identify every connection between technologies. Once we've identified those connections we can go through them one by one to determine if the connecting technologies integrate together smoothly.

2.0 Technological Challenges

In this section we will outline the technological challenges that we will overcome as we implement our solution. At it's core our project is a fairly common web data entry platform. Some challenges we face have been solved and thus should be straight forward. Having said that, there are also some challenges with our specific implementation that need to be addressed in some more detail. We will start by outlining the various customer constraints we've been given for this project. While these solutions are already determined, it's important to understand how they fit into our system. Finally we will outline the various technological challenges and define the metrics by which we will judge them.

2.1 Customer Constraints

Our client has some constraints that our system needs to be able to work with. Some of these constraints have to do with the preexisting system we are interfacing with while some are client preference. We will go over the customer constraints now, as they play a role in some of our technological challenges. The customer constraints we have are as follows:

- **Google Authentication**: Our client doesn't want to have to store collaborator passwords in their database. In addition to this, not having to make a new account for our website will be convenient for the collaborators
- **JSON Objects**: Our client stores dynamic data for their samples in their database in the form of JSON objects. Because of this our system needs to be able to create JSON objects and send them to the database.
- **PostgreSQL Queries**: Our client uses a PostgreSQL database and so our website needs to be able to perform PostgreSQL queries and insertions

2.2 Back-End Language

The back end language is essentially the brain of our web platform. Our back-end language will be responsible for doing all of the interaction with our clients database, configure the website, abstractly model all the data, and interact with our bar code and shipping label generation. Whatever language we choose needs to be able to:

- **Read a text configuration file to customize our websites structure**
- **Make PostgreSQL queries and insertions to update and read our clients database**
- **Send JSON objects to the PostgreSQL database**
- **Be object oriented to model our various packages, samples, processes and steps (Not strictly necessary but simplifies the coding process)**

We will select languages that fit this criteria and then judge them based on the following metrics:

- **Team Member Familiarity**: How many team members are familiar with this language already? (0-4)
- **Front-End Compatibility**: How compatible is the language with common front end solutions? Is this language used frequently for web development? (0-4)
- **Simplicity**: How simple is the structure of the language? Is there a lot of necessary plugins that don’t add any functionality necessary for our project? (0-4)
- **Cost**: Will it cost us anything to commercially use this language or is it open source? ($)

2.3 QR or UPC Generator

In order for our project to fully solve our clients current problems it must be able to produce some type of bar code. Searching the database for the sample data is quicker than inputting it yourself, but it still takes a significant amount of time away from a scientists daily productivity time. Because of this our back-end language needs to be able to use some type of bar code generating plugin. Our bar code generator needs to be able to:

- **Produce printable QR or UPC bar codes and ID numbers for those bar codes**

While the necessary requirements aren’t that stringent, there are a few metrics we can use to judge how well different generators will perform in our project:

- **Performance**: How fast does the generator produce the bar codes? (0-4)
- **Scan-ability**: The bar codes are often going to be wrapped around test tubes, how easy are they to scan when they aren’t flat but instead curved? (0-4)
- **Plugin Availability**: How many back-end languages does it support? Will it fit into any back-end choice we make? (0-4)
- **Cost**: Will it cost us anything to commercially use this generator or is it open source? ($)

2.4 Front-End Language

Certain aspects of the front end language, such as HTML, are essential to the front end development process. Despite the lack of choice for some components we will still be doing technical research on them to determine how best they fit into our system. There are still some areas where we have agency though and those areas must be able to:

- **Work well on most web browsers**
- **Dynamically display shipping/process stages**

After compiling languages that meet these criteria we will judge them with the following metrics:

- **Team Member Familiarity**: How many team members are familiar with this language already? (0-4)
- **Back-End Compatibility**: How many back-end languages work well with this front-end language? (0-4)
- **Screen Size Adaptability**: How well does the language support screen size adaptability? (0-4)
- **Mobile**: How well does the language work on mobile platforms? (0-4)
- **Cost**: Will it cost us anything to commercially use this language or is it open source? ($)

2.5 Shipping Label Interface

Our platform needs to be able to produce shipping labels for the collaborators to attach to their packages and send to TGen. The shipping label interface needs to be able to:

- **Generate printable FedEx labels**

While the requirements are straight forward we do have a couple metrics:

- **Tracking**: Does the interface also allow you to track shipments as well as create them? (Yes/No)
- **Back-End Compatibility**: How many back-end languages work well with this interface? (0-4)
- **Cost**: Will it cost us anything to commercially use this interface or is it open source? ($)

2.6 Web Data Entry Spreadsheet

Our client wants us to use spreadsheet style data entry because most collaborators already submit data in a spreadsheet format. We cannot use google sheets as several collaborators don't feel that it is secure enough. Spreadsheet data entry also allows the collaborator to input many samples at the same time in different columns. The spreadsheet applet we choose needs to be able to:

- **Accept copy and paste entry from common spreadsheet programs**
- **Set non-editable fields**
- **Pull data from spreadsheet and send it to back end language for database insertions**

Once we've compiled applets that can meet these requirements we will judge them based on the following metrics:

- **Expandable**: Is the spreadsheet dynamically expandable for further data entry? (Yes/No)
- **Front-End Compatibility**: How many front-end languages work well with this spreadsheet? (0-4)
- **Cost**: Will it cost us anything to commercially use this spreadsheet or is it open source? ($)

3.0 Technology Analysis

In this section we will go over the technological challenges we outlined in the previous section and analyze potential solutions to each. The general overlay of the different parts of the system and their place in the system can be seen in figure 3. After we've analyzed the options we will use our metrics to decide which option we will proceed with. Our technological issues are as follows:

- **Back-End Language**
- **QR or UPC Generator**
- **Front-End Languages**
- **Shipping Label Interface**
- **Web Data Entry Spreadsheet**

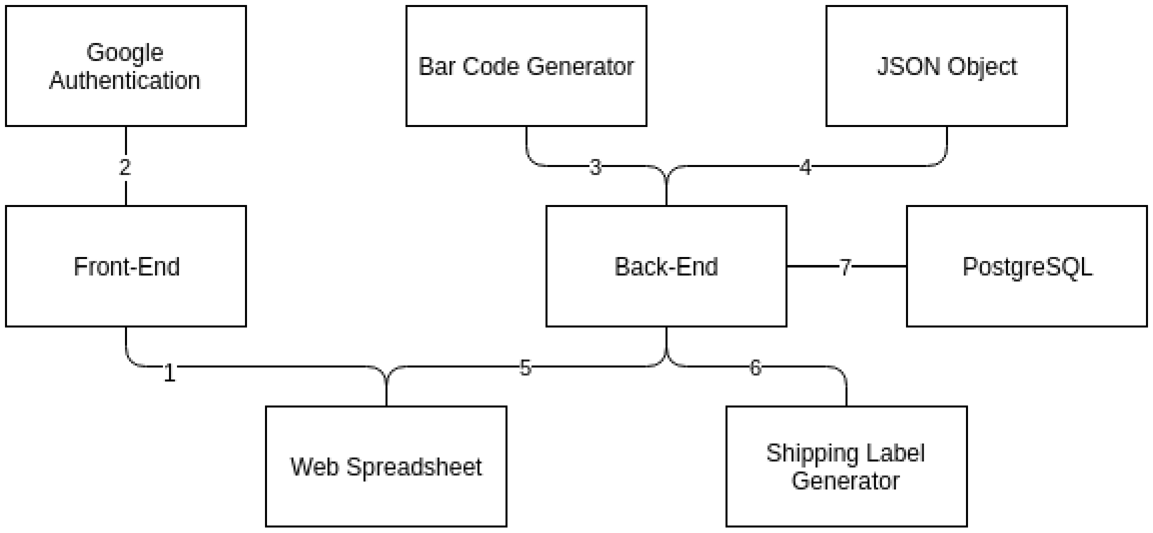


Figure 3: System Diagram with Connections

3.1 Back-End Language

The back end language is essentially the brain of our web platform. Our back-end language will be responsible for doing all of the interaction with our clients database, configure the website, abstractly model all the data, and interact with our bar code and shipping label generation. The back-end will be compiled and hosted on our clients server, coordinating operations between the front-end and database. Whatever language we choose needs to be able to:

- **Read a text configuration file to customize our websites structure**
- **Make PostgreSQL queries and insertions to update and read our clients database**

- **Send JSON objects to the PostgreSQL database**
- **Be object oriented to model our various packages, samples, processes and steps (Not strictly necessary but simplifies the coding process)**

There are three main alternatives we can choose for our back-end language: Java, Python or Node.js. **Java** is the most common runtime environment that enterprises use which means there is a large community of support behind it. **Python** is an increasingly popular language that offers a lot of new functionality not supported by Java. **Node.js** is a server side javascript platform, which is nice for uniformity of languages between front-end and back-end applications.

3.1.1 Java

Java is the most common runtime environment that enterprises use. It was released in 1995 and since then has morphed into one of the premier server side back-end development languages. Ironically it wasn't designed with this in mind, as at the time the internet looked much different than it does today. Java is object oriented and is used by around 9 million web applications.

Pros:

- **It's popularity means it has support for a lot of different plugins**
- **Everyone on our team can program in Java**
- **It's about twice as fast as optimized JavaScript**
- **Automatic Garbage Collection**
- **WORA Design, can port code to any platform with a JVM without having to re-write it**
- **Open-source**

Cons:

- **Different language from our Front-End**
- **Less built in functionality**

3.1.2 Python

Python has seen a surge of popularity in recent years. It's a higher level language than Java and offers a lot of built in plugins that support increased functionality. Because of it's popularity it still has support for a lot of platforms. Python is a dynamically typed language and offers a lot of support for mathematical and statistical operations.

Pros:

- **It's popularity means it has support for a lot of different plugins**
- **Everyone on our team can program in Python**
- **It has a lot of built in functionality**
- **Automatic Garbage Collection**
- **Open-source**

Cons:

- **Dynamically typed**
- **A lot of overhead for functionality not necessary for our project**
- **Different language from our Front-End**

3.1.3 Node.js

Node.js is a javascript language that can be run server side. It's a newer language released in 2009, and supports a "Javascript everywhere" methodology. The idea is to unify the coding language between the front-end and back end. Node.js is an asynchronous language meaning that the code isn't run in sequence but all at once.

Pros:

- **Same language as Front-End**
- **Open-source**

Cons:

- **Only one of our team members knows this language well for back-end applications**
- **Asynchronous**
- **Slower**

3.1.4 Back-End Chosen Approach

We chose Java for our back-end language. The main reason we chose java is because of team member familiarity with it and it's simplicity (Table 1). Java does not have as much overhead as Python and yet offers all of the functionality we need for our project. Java is also not run asynchronously like Node.js which simplifies our hierarchy of item building, taking into account that none of our team members have a lot of experience with asynchronous programming languages. Because of it's popularity, Java has the most support for different pieces of our system that we want to include. In this situation it doesn’t make sense to re-invent the wheel when our back-end tasks are fairly standard.

	Cost	Familiarity	Compatibility	Simplicity
Java	0$	4	4	4
Python	0$	4	3	2
Node.js	0$	1	3	3

Table 1: Back-End Language Metrics

3.1.5 Back-End Proving Feasibility

Java is an object oriented language, we will be able to abstractly model the objects that need to be displayed on our website (Figure 4). Processes will aggregate steps, which will aggregate Items. This allows us to load all of the samples in the system and place them at the correct step, which is placed in the correct process. Packages are also an aggregation of items, so that we can display the items inside of the package. This object model will be able to model any similar system, with the amount of each class, and it’s name being configurable via a text file. This will allow other companies that want to intake and track items going through processes to port our platform to their business with a simple edit of a text file.

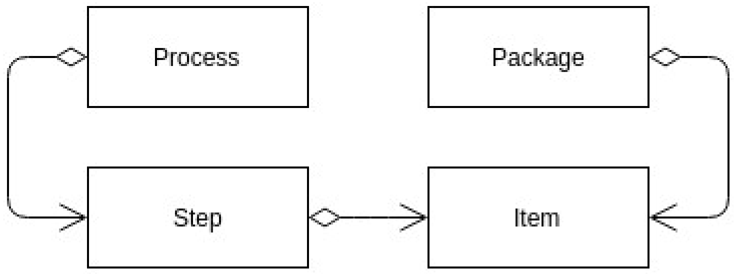


Figure 4: Class Diagram of our System

Our future plans for testing involve coordinating with our client to set up test environments on their server. We will create a java class that is able to query the server, and send that data to our front-end. We will start prototyping our class structure as well, so that we can start working on demos for the other parts of our website. The actual feasibility of our back end language is pretty sound, as we’ve all had years of experience designing systems in exactly this manner. The only new things to us in this process is running it on the server and database queries we will be performing.

3.2 QR or UPC Generator

TGen North has utilized printable barcodes to assist with collaborators getting an idea of what stage their sample is at inside of the TGen facility. These barcodes serve an additional purpose to help scientists scan and be able to have all sample data appear on the screen at their lab station. However, this process is not efficient as collaborators are submitting their samples using their own haphazard labeling systems that often times don't work with TGen North's current workflow. We believe that the best solution for this issue is standardizing the process for how collaborators are printing and labeling their samples through the use of generated QR or UPC barcodes that are small enough to be placed on the sample tubes. Also, these barcodes need to be big enough so that there isn't any issues when the scientist goes to scan it when at the workstation. A feature of these barcodes is that they need to be able to communicate with the PIMS 2.0 database when data is submitted and when a QR or UPC code is scanned by a scientists.

3.2.1 GOLang QR Code Library

The current back-end system of TGen North Utilizes the GO Language, a free programming language with attributes similar to python. The research we conducted into this back-end language as well as its libraries led to a discovery of a Github library dedicated to QR generation making the communication between front-end and back-end very seamless for us.

Pros:

- **Utilizes the back-end language to assist in creating QR codes for the clients**
- **We might be using GOLang for the back-end of our system and if so, this will make communication from front-end to back-end very easy for us.**

Cons:

- **This will require a lot of in lab testing by the scientists which might be difficult with time constraints**
- **We are not sure if this is the system they are currently using for their QR generation, so all the MetaData that they usually show below a QR code might not be viable.**

3.2.2 QR Code Generator (QR Code Monkey)

There are several free, open source QR code generating tools on the internet. Some offer more tools in the way you can integrate into your own website as well as sending data to the website and having the QR code then populate back onto your website. This would be incredibly convenient for the web application since we are attempting to standardize the current process collaborators are haphazardly doing.

Pros:

- **Fully customizable shapes that the barcode could be displayed and printed as. This is especially beneficial so we could test the shapes to see which ones would fit best on the sample tubes and are easiest for the scientists to scan**
- **We were able to identify a website that allows for fully customizable QR code generation and has an API that would allow us to use it directly on our website**

Cons:

- **This will require a lot of in lab testing by the scientists which might be difficult with time constraints**

3.2.3 UPC Barcode Generator (Barcode Tec-it)

Similar to the QR code generators there are countless free open source UPC barcode generator websites. This solution offers a different way of encoding data onto a scannable piece of paper and different possibilities to submit the data in the way it is stored.

Pros:

- **Integratable API that would allow UPC codes to be generated on our website**

Cons:

- **There are several versions of UPC barcodes that can be generated requiring research for which type would be best for database**
- **UPC barcodes are wider than QR codes when printed, which could be an issue when collaborators are labeling samples**

3.2.4 Barcode Generator Chosen Approach

As a team we concluded that the QR code generator that is a library will be best to integrate into our web application as it offers a wide variety of tools to help us generate QR Codes scanable from a test sample, and allows us to communicate the information inputted into the spreadsheet to a QR Code for the client (Table 2). We hope to create several demo versions to test data entry, QR code generation, and scanning to reduce the amount of errors that will occur throughout our development process.

	Cost	Familiarity	Compatibility	Simplicity
GOLang Library	0$	2	4	3
QR Code Monkey	0$	1	3	2
Barcode Tec-IT	0$	1	2	2

Table 2: Barcode Generator Metrics

3.3 Front-End Language

The front-end development language is to create Web pages or apps. These days, front-end development refers to the part of the web users interact with. In the past, web development consisted of people who worked with Photoshop and those who could code HTML and CSS. Now, developers need a handle of programs like Photoshop and coding in not only in HTML and CSS, JavaScript or jQuery (a compiled library of JavaScript). The language and development process we use to create our web application will set the entire foundation for our product so proper examination of all tools is crucial.

Below are some common front-end development languages the team has looked into to help assist in the creation of our web application

3.3.1 HTML / (CSS)

HTML is the markup that contains all the actual stuff that a web page has. All the text on this page you’re reading right now lives inside HTML tags that tell your browser how to order the content on the page. Go on, right click any element on the page and choose “Inspect Element” to open up your browser’s Developer Tools and it will show you the structure of the page.

CSS tells the browser if you want to display any of those tags a particular way, for instance, turning its background blue and pushing it a little to the left. In your Developer Tools, you can see the CSS styles in another panel, usually showing which specific properties were inherited from which lines of CSS.

Pros:

- **Our team is very comfortable using HTML/CSS to develop websites**

Cons:

- **Programming the web application without documentation of what each member is doing could halt the over development process. We will need to make sure we are communicating the changes we make.**

3.3.2 JavaScript

JavaScript is a high level, dynamic, untyped, and interpreted programming language. It has been standardized in the ECMAScript language specification. Alongside HTML and CSS, it is one of the three essential technologies of the World Wide Web content production; the majority of websites employ it and it is supported by all modern web browsers without plug-ins. JavaScript is prototype-based with first-class functions, making it a multi-paradigm language, supporting object-oriented, imperative, and functional programming styles. It has an API for working with text, arrays, dates and regular expressions, but does not include any I/O, such as networking, storage or graphics facilities, relying for these upon the host environment in which it is embedded. Javascript could help us to solve the problem of sending the inputted data from the front-end to the back-end for barcode generation, tracking label generation, and submitting the data to the database.

Pros:

- **Allows for data submission from front-end to back end**
- **Allows our HTML/CSS web pages to dynamically adjust depending on the data submitted**

Cons:

- **We have just one group member who is confident in JavaScript meaning we will need to learn this language if implemented.**

3.3.3 JQuery

JQuery is a simple and fast JavaScript library. JQuery learning is relatively easy compared to learning Java scripts, this is reduced. The difficulty of web front-end development, and jQuery is compatible with almost all browsers. JQuery is a JavaScript library designed to simplify HTML DOM tree traversal and manipulation, as well as event handling, and CSS animation. Web analysis indicates that it is the most widely deployed JavaScript library by a large margin, having 3 to 4 times more usage than any other JavaScript library.

Pros:

- **It is a free open source project**
- **Allows for HTML DOM manipulation, meaning webpages can be dynamically created**
- **It is a very fast extension of the JavaScript Library**

Cons:

- **No group members have experience with this technology meaning we will need to allocate time to learning and harnessing the utility of JQuery**

3.3.4 Mootools

MooTools is a collection of JavaScript utilities designed for the intermediate to advanced JavaScript developer. It allows you to write powerful and flexible code with its elegant, well documented, and coherent APIs. MooTools code is extensively documented and easy to read, enabling you to extend the functionality to match your requirements.

Pros:

- **Could allow us to do all data submission from the front-end to the back-end with little overhead**

Cons:

- **No group members have experience with this technology meaning we will need to allocate time to learning and harnessing the utility of Mootools.**

3.3.5 Ajax

Ajax is a set of web development techniques using many web technologies on the client side to create asynchronous web applications. With Ajax, web applications can send and retrieve data from a server asynchronously (in the background) without interfering with the display and behavior of the existing page. By decoupling the data interchange layer from the presentation layer, Ajax allows web pages and, by extension, web applications, to change content dynamically without the need to reload the entire page.

Pros:

- **It is a free open source project**
- **Allows for HTML DOM manipulation, meaning webpages can be dynamically created**

Cons:

- **No group members have experience with this technology meaning we will need to allocate time to learning and harnessing the utility of Ajax.**

In summary, there are a lot of tools on the internet we will be able to utilize for this project. As a team we concluded that the technologies that we will implement into the front-end of the web application will be HTML/CSS for the overall look of the web app and utilize JS, & JQuery to assist in dynamic page resizing and data manipulation.

3.3.6 Front End Chosen Approach

There aren't significant choices associated with front end languages, and thus its arguable that there doesn't need to be a front end section in an industry standard tech feasibility document. However, as our team is composed of students and we as of yet have no industry experience, it was an important exercise for us to prove (if only to ourselves) that the front end languages can do what we need them to do. So while our chosen approach is really our only approach we will still lay out the best analysis of the languages that we can in order to prepare ourselves for implementing our project. Our chosen front end languages are CSS/HTML for the display of the website, and Javascript to make it dynamic (Table 3). CSS/HTML does not interact with the back-end directly but instead users javascript as an intermediary. Our team if fairly familiar with it and it can adapt to various screen sizes with it's percentage settings on display size. Javascript can be integrated with most back-end languages and it's adaptability to screen sizes is also high as it can change it's logic depending on the screen.

	Team Member Familiarity	Back-End Compatability	Screen Size Adaptability	Mobile	Cost
CSS/HTML	3	0	3	3	0$
Javascript	2	4	4	3	0$

Table 3: Front-End Metrics

3.4 Shipping Label Interface

TGen North's collaborators ship their samples from all around the world, but most of the time collaborators don't know if and when the samples have been received by TGen North. Collaborators also use their own methods of printing and placing the labels onto their shipments of samples.

We believe the solution for this problem is having a printing label generated from the web application with instructions as to how to place the label as well as providing the user with and attempting to integrate Fed-Ex tracking on the web application since 80%-90% of packages are sent using Fed-Ex.

3.4.1 Integrated Fed-Ex Label & Package Tracking

Since 80%-90% of all packages received by TGen North come through Fed-Ex it makes sense to assist users in seeing where their packages reside in the shipping process. We hope to integrate the tracking box provided on Fed-Exs website to our web application, but if it can't be managed we can have a link that will take the collaborator to the Fed-Ex website using a hyperlink with the tracking number embedded in the hyperlink.

Pros:

- **Users can be aided in the tracking of their packages**
- **TGen North won't need to be asked if the package has been received or not. Saving time and money**

Cons:

- **No formal documentation on integrating Fed-Ex's tracking box**

Since Fed-Ex is used so often the team believes that integration of Fed-Ex tracking to assist collaborators would be best (Table 4).

	Tracking	Back-End Compatability	Cost
Integrated Fed-Ex Label & Package Tracking	Yes	4	0$

Table 4: Shipping Label Interface Metrics

3.5 Web Data Entry Spreadsheet

The spreadsheet is an important part of our project because it is used to organize and categorize data into a logical format to help scientists do their research. In our project, we first get data from the Front End, then create the spreadsheet in the Back-end, then use spreadsheet to organize and categorize the data from the front-end, and then finally pull data from spreadsheet and put it into back end language. Google sheet seems like a good choice because it is a free, web-based program for creating and editing spreadsheets. However, the contributors believe it be an unsafe method because Customer's believe their information is at risk when using Google Sheets.

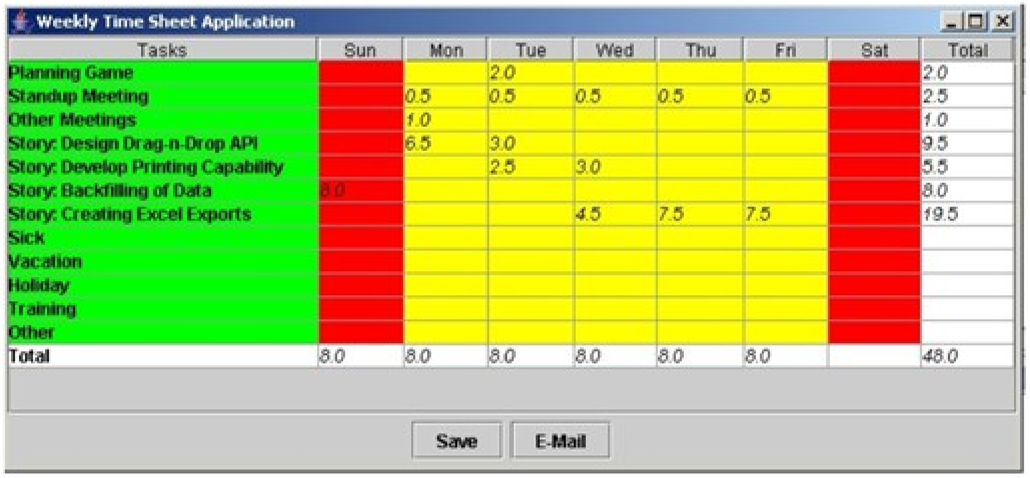
The solution to this issue is to find a free open source spreadsheet that has the ability to be integrated with our web app and TGens database.. The spreadsheet we created should have some essential quantities. First it has the ability to copy and paste into spreadsheet from other common spreadsheet project. Secondly, it has the ability to set non editable fields. Third, it has the ability to pull data from spreadsheets and put it into the back end language to be sent to the database.

3.5.1 Jakarta POI

Apache POI is the open source library of the Apache Software Foundation. The Jakarta POI project consists of APIs for manipulating various file formats based upon Microsoft's OLE 2 Compound Document format using pure Java. POI provides several APIs to read and write Microsoft Excel (HSSF), Microsoft Word (HWPF), and OLE Property Sets (HPSF). However, to read and write Excel (XLS) files using Java, the HSSF (Horrible SpreadSheet Format!) API.

Pros:

- **It is open source so you can learn the code from some website easily, it is also easy to learn how to create and read the spreadsheet.**
- **It is based on Java, and the team is familiar with the Java programming language**
- **The spreadsheet is dynamically expandable**
- **Has a clean interface within the front-end (Figure 5)**



Tasks	Sun	Mon	Tue	Wed	Thu	Fri	Sat	Total
Planning Game			2.0					2.0
Standup Meeting		0.5	0.5	0.5	0.5	0.5		2.5
Other Meetings		1.0						1.0
Story: Design Drag-n-Drop API		6.5	3.0					9.5
Story: Develop Printing Capability			2.5	3.0				5.5
Story: Backfilling of Data	8.0							8.0
Story: Creating Excel Exports				4.5	7.5	7.5		19.5
Sick								
Vacation								
Holiday								
Training								
Other								
Total	8.0	8.0	8.0	8.0	8.0	8.0		48.0

Figure 5: Jakatra POI

Cons:

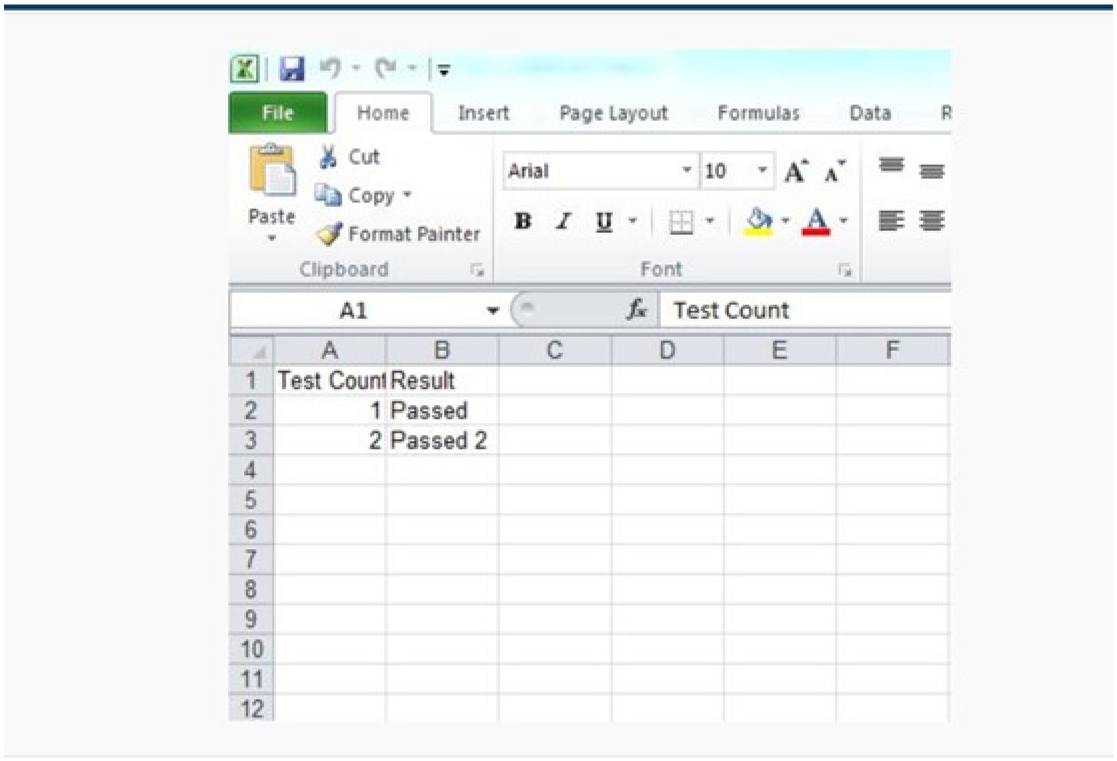
- **We are not sure if it is possible to copy a spreadsheet from say Excel and paste it into the spreadsheet of choice.**

3.5.2 Java Excel API

Java Excel API is a mature, open source java API enabling developers to read, write, and modify Excel spreadsheets dynamically.

Pros:

- **Supports font, number and date formatting**
- **Supports shading, bordering, and coloring of cells, so it will be able to display nicely on the front-end**
- **Below is a result from one of our test environments (Figure 6)**



	A	B	C	D	E	F
1	Test Count	Result				
2	1	Passed				
3	2	Passed 2				
4						
5						
6						
7						
8						
9						
10						
11						
12						

Figure 6: Java Excel API

- **Has the ability to modify existing worksheets, so you can copy and paste into a spreadsheet from other common spreadsheet program**
- **The spreadsheet is dynamically expandable**
- **It is open source so you can learn the code from some website easily, i mean it is easy to learn how to create and read the spreadsheet.**

Cons:

- **The only downside is it just generates spreadsheets in Excel 2000 format. It can not work on other Excel format.**

3.5.3 Spreadsheet Chosen Approach

We decided on using the second option. Compared with Apache POI and Java Excel API, they both have has the ability to set non editable fields, the ability to pull data from spreadsheets and put it into back end language as well as display nicely on the front-end (Table 5). However, Apache POI does not support the ability to copy and paste information from one .csv file to another which could be annoying when we implement into the web application. While in the Java Excel API, it allows for such copy and pasting to and from .csv files.

	Cost	Language	Dynamically Expandable	Display	Copy and Paste Data
Jakarta POI	0$	Java	Yes	3	Unknown
Java Excel API	0$	Java	Yes	3	Yes

Table 5: Web Data Entry Spreadsheet Metrics

4.0 Technology Integration

Great pieces of technology are useless if they don't fit into the larger solution. Even if a bar code generator can do everything we want it to do perfectly doesn't mean it will work with the rest of our system. Because of this we need to map out our system and determine which pieces need to work with each other. Once we've established the connections we need to test each one so that we can be reasonably confidant in our choices.

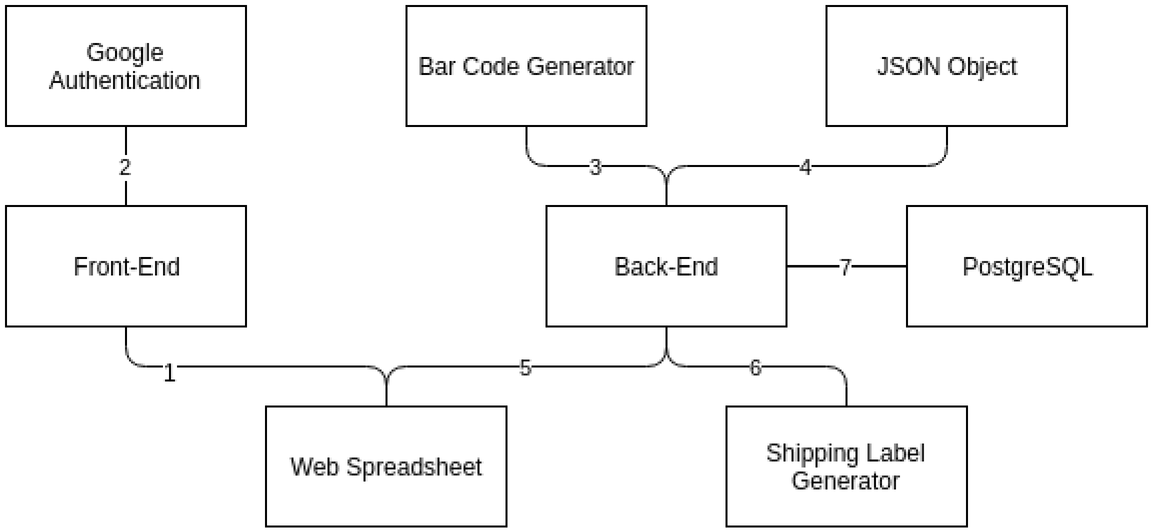


Figure 7: Components of system and their connections

4.1 Connections

4.1.1 Front-End and Web Spreadsheet (1)

We can embed Java Excel API into our HTML page and reference it with javascript. This will allow us to add on click functions to buttons that pull the data out of the applet and send it to the back-end application for processing. Javscript will be our main connection here, as we will use it to communicate the data entered by the user into the spreadsheet applet to the back-end.

4.1.2 Front-End and Google Authentication (2)

We can embed the Google Authentication widget into our HTML and reference it with javascript. The on click function will take the user to google authentication page where they will sign into their google account. From there google will redirect the user back to our page and give us an identifying key that will also serve to verify that the user authenticated successfully.

4.1.3 Bar Code Generator and Back-End (3)

Our QR generator has a plugin that will work with our back-end language. We will import the plugin and use the functions to generate printable QR codes that we can send to the collaborator directly via email. We can use different methods to create different QR code printing formats depending on the collaborators needs.

4.1.4 Back End and JSON Objects (4)

JSON Objects has a Java api that we can include in our back end solution. This will allow us to create and read JSON objects from the database.

4.1.5 Back-End and Web Spreadsheet (5)

Java Excel API, like the name suggests, has an API for Java. We will be able to include this in our java back-end and access any spreadsheets the client submits. If a client opts to use the website generated spreadsheet for their data entry then our back-end will receive all the info via parameters in javascript method calls to the back-end.

4.1.6 Back-End and Shipping Label Generator (6)

Fed-ex has it’s own APIs to support various platforms that will allow us to include it in our Java application. With the plugin we can take the tracking number and call the methods to give us the tracking information to display on our front-end.

4.1.7 Back-End and PostgreSQL (7)

PostgreSQL has a Java API that will allow us to access the server directly with method calls from Java. This will allow us to configure the queries for different servers depending on the config text file. Using the plugin we can insert data into the server as well as query data from the server.

5.0 Conclusion

Our problem we are solving involves issues with the current workflow process and data submission with TGen North and their collaborators. Both TGen North and ViralTech envision a dynamic web application with the capability to instruct collaborators on the proper way to ship their packages, label their sample tubes with generated QR codes, and see the overall progress of both their package and their sample tubes when received by TGen North.

This document outlined in detail the technological and integration challenges of creating this web application. The primary technologies were a QR Code generation, a shipping label interface, google authentication, a web data entry spreadsheet, and possible front-end and back-end languages. Below, are the technologies and our confidence level with each technology (Table 6).

Challenge	**Potential Solution(s)**	**Confidence Score (1 to 5)** **1 = low confidence, 5 = high confidence**
QR or UPC Barcode Generator	QR Codes	5
Shipping Label Interface	Fed-Ex Tracking	3
Web Data Entry Spreadsheet	Java Excel API	4
Front-End Languages	HTML/CSS/JavaScript/JQurey	4
Back-End Languages	Java	5

Table 6: Confidence Levels

ViralTech believes that our technological and integration challenges have been addressed, which assures us that our solution is definitely feasible. We have high confidence in our Back-End language as well as in our choice of Barcodes (5/5). While we have less confidence in our Shipping Label Interface (3/5), we are confident in our Front-End language and Web Data Entry Spreadsheet (4/5). We will continue to address technological and integrational issues as they arise through the development process.

With our selected technologies, we are beginning the creation of a prototype application to test and verify our selections. ViralTech believes that working on the prototype sooner rather than later will be beneficial in being able to diagnose problems early on so it can be quickly addressed.

While there are always unforeseen issues, we have done as comprehensive of an analysis as can be efficiently done at this time. We have turned our project from an abstract concept of what we need to do, into a concrete understanding of the various parts and technologies we need to develop and integrate for it to work. We are happy with our analysis, excited to begin development and confident we can get the job done.