Requirements Document

Team Blue Box

Date: November 6th, 2020

Members

Bradley Barber

Melina Diamond-Sagias

Jonathan Hillman

Yunsong Wang

Zachary Wilson-Long

Mentor: *Tomos Prys-Jones*

Client: *Dr. Toby Dylan Hocking*

Version: 2.0

Accepted as baseline requirements for the project:

For the client:___________________________

For the team:___________________________

Contents

1.0-Introduction

TEAM BLUEBOX

The incredible improvement modern medicine has made on human health can often be attributed to the advancement of technology. While technological advancement allows for more effective tools and medicines, it also provides medical researchers access to information that was previously unavailable. One notable example of this is gene science, which has become one of the most prominent medical advancements of recent years.

While the efficacy of these technologies has increased, the cost of sequencing has reduced, making it possible for medical researchers to understand the genetic foundations of many diseases and identify over 6,000 genetic disorders. In particular, medical researchers use genomic data to analyze the genetic causes behind diseases in order to develop treatments that will eventually prevent those conditions. However, while promising, genomic analysis can be challenging to utilize.

The density of a human genome (containing 3.2 billion bp) makes them difficult to parse. These challenges can significantly impede the identification of genetic issues and our understanding of genomes and further developments have the potential to improve countless lives. To reduce processing time, researchers look for ways to gather as much information from the sequences as possible while excluding redundancies and unnecessary processing steps. However, further tools are needed to facilitate research further by increasing speed and efficiency.

A common issue with new technologies in gene science is obsolescence of new innovations, as the field rapidly progresses [2]. This means that lots of resources are put into creating new tools and technologies that do not live to justify their investment. So, we need new tools to be scalable, adaptive and provide functionality that will be beneficial sooner rather than later.

We are working with our client, Dr. Toby Hocking, a biologist and machine learning expert here at NAU, to create a software tool that will greatly improve genomic research for the foreseeable future.

2.0-Problem Statement

TEAM BLUEBOX

The most popular choice of software for genomic analysis, JBrowse, doesn't provide enough functionality to be effective for modern gene science. Originally created to view genomic sequences, JBrowse incorporates a dynamic scoping algorithm to provide detailed visualizations of genomic sequences. However, it lacks good ways to upload data, provides no collaborative features, and requires manual inspection to identify specific characteristics of the data. Our client, Dr. Toby Hocking, a biologist and machine learning expert at NAU, requests a software that offers more functionality than JBrowse. Since JBrowse is open-source it can provide a useful foundation to create a plugin that will provide major improvements to collaboration and the ability to use machine learning will dramatically increase the rate and effectiveness of analysis.

Currently, JBrowse limits users by forcing them to use it on their local machine. Since it offers no functionality across the internet, using JBrowse collaboratively is slow, difficult, and frustrating. Due to the complexity of genomic sequences, genomic-related projects demand collaborative efforts which JBrowse fails to provide. This inefficiency compounds overtime making collaboration a detriment rather than an advantage.

To use JBrowse, a user must provide data from their local machine instead of directly accessing it from the internet. This required storage space makes JBrowse very inconvenient to use, especially for gene scientists who want to analyze large or several datasets. If a group wanted to use JBrowse collaboratively, they would have to store the data on each machine they are using. Updating the data or sharing results with others provides an unnecessary obstacle that gene scientists already face a lot of.

JBrowse's biggest drawback is its lack of machine learning application. Currently, to identify non-coding regions, or "peaks", of non-coding regions, gene scientists must manually label them throughout the sequence. An example of this can be seen in *Figure 1* which shows our clients current implementation of PeakLearner thus far. Given that a single human genome may contain 3.2 billion nucleotides[1], attempting to label all of these peaks is practically impossible. Being able to identify peaks automatically using machine learning principles, would further improve the analyses of genomic sequences.

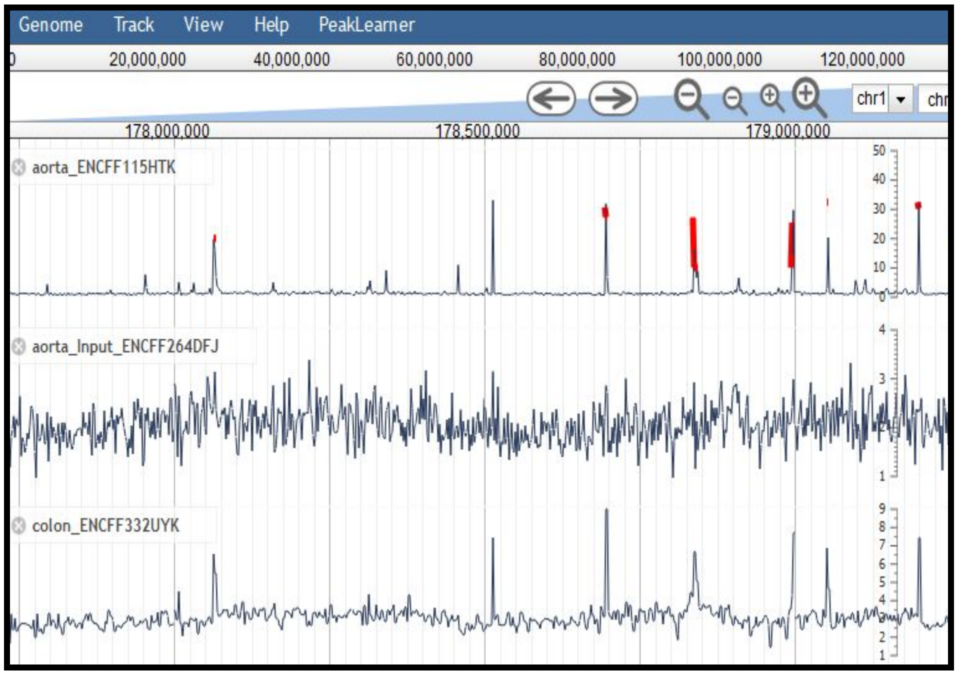


Figure 1

Our client, Dr. Toby Hocking, has requested us to work on a software that extends JBrowse to address the weaknesses it currently has. Creating this software successfully will not only provide Dr. Hocking with a new tool for his own analysis, but will help the entire gene science community extend the domain of human knowledge and contribute to future medical breakthroughs.

3.0-Solution Vision

TEAM BLUEBOX

Our primary goal as Team BlueBox is to create a web application called PeakLearner to interact with the software and the data it provides. It is of primary importance to ensure the application allows interaction with the sequencing data in the form of creating and deleting hubs, where tracks are uploaded and managed. In addition, to enable use across multiple project groups, the application will contain an authentication feature, which will allow collaborators to manage group permission and access to data. This will not only increase the usability of PeakLearner by allowing multiple researchers to modify the same data, but it will also increase security.

With the implementation of the JBrowse plugin for PeakLearner developed by the project's previous capstone group, PeakLearner is able to visualize genomic sequencing data. However, our client's machine learning algorithm has only just been integrated into the existing software. Our primary goal is to allow for the interaction of the visualized peak data and predicted peak models in a single web application. This web application will allow for users to interact with the data in ways previously unavailable, like hub configuration.

As stated above, our plan to address the lack of security in the current iteration of PeakLearner is to develop and implement a user authentication system. By doing so, we will ensure that only specific users have access to certain track data files by way of username and password. The creation of user profiles will then contribute to our implementation of a group collaboration feature which will allow users in the same group to have access to the same track data and label information. The scope of a user's ability to access and modify these things will be dependent on their permissions in the group.We believe that the incorporation of both user authentication and group collaboration will dramatically increase the security of the application.

In addition, the integration of the group collaboration feature will increase the usability of PeakLearner. This is because researchers often work in groups, so it will be extremely beneficial if research groups are able to view the same track data information at the same time. A group collaboration feature with a shared hub will also allow research groups to keep track of the progress each team member has made. In order to implement our proposed solutions so that PeakLearner may be a fully functional web application, we first need to determine the key functional, performance, and environmental requirements of these solutions. The following section will examine the project's key requirements.

4.0-Project Requirements

TEAM BLUEBOX

Our project will be divided into major requirements under three different subsections. First we will address our functional requirements relating to actual features that must be developed. Secondly we will discuss the performance requirements relating to certain attributes and standards that the product must maintain. Lastly we must list our environment requirements which will determine the tools, workflow, and limitations that we must adhere to during development.

4.1 Functional Requirements

To develop a web application that will implement our solution vision, our application will have specific functional requirements. These features have been split up into four key requirements sections including *User Interface*, *User Authentication*, *Database API*, and *Collaborative Groups*. These will be described below.

4.1.1 User Interface

To prevent the need for users to learn technical skills, an efficient and easy to use web interface will be developed. This will contain: an easy to use user interface for PeakLearner operations and analysis, a user profile page, and an about page.

4.1.1.1 Webapp Design

The design of the webpage will make it easy to use and follow. PeakLearner's web app will be intuitive and will not require the user to research how to use the webpage. However, we will still create an about page with a basic (and in depth) tutorial for any common questions.

- **Main Page:**
  The main page will have a button to go to the sign in, an about page, and the main GUI for PeakLearner. The main page of the web application will take people straight to an empty peak graph with options to upload data using links to data as opposed to local files. Once the user picks their data file from their computer it will fill in the graph and give the ability to perform operations on the data. If the user is not signed in yet, it will prompt them to sign into their Google account before uploading any data.

- **About Page:**
  The about page will give a tutorial on how to use PeakLearner. The idea is to make the webapp as easy to use as possible, but an about page will be useful for even the most basic of questions. This page will talk about how to upload data, how to perform operations on the data, how to join a collaborative group, and any other major information that we come across during testing. The information on this page will be basic on the surface but each section will have the ability to be hovered over to display a gif showing how to do this specific task.

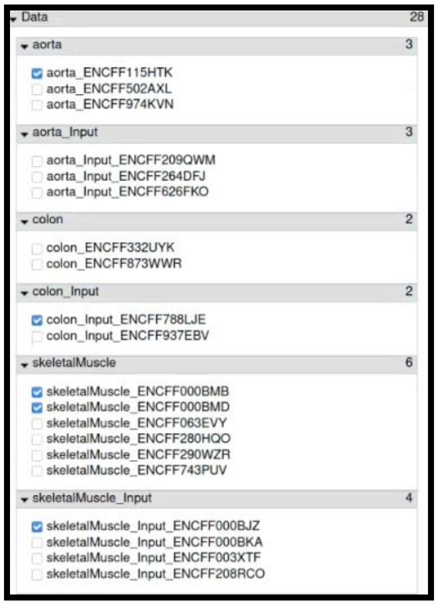
4.1.1.2 Data Operations GUI

To make PeakLearner a competitive genome browser, its data operations need to be effective for the gene scientists using them. To maximize the effectiveness of PeakLearner, we will focus on improving the hub management and label interface that JBrowse currently provides. Additionally, the look of the data operations GUI is important for ease-of-use. We will create a simplistic looking GUI that is intuitive and powerful.

- **Obvious design:**
  The design of the GUI must be intuitive, a user should be able to navigate the interface with relative ease. A tutorial will be supplied but the design will be easy enough to go through without the need to use the about page. Our metric for measuring this is discussed in section 4.2.4.

- **User Hub Management**
  Each user will have access to a collection of hubs. If the user created the hub, they will have special permissions that no other member in the group can have (as seen in section 4.1.5.1). The hub will contain all of the shared data and data analysis made by a corresponding group of users.

- **Hub Configuration**
  The central component of data analysis, the hub, is composed of a collection of tracks, or sequences. PeakLearner will provide a way for users to add, remove, or organize tracks within a hub. The addition of tracks will be through the use of a filelink with a BigWig format. The hub will also be able to visualize multiple tracks at once, and it will contain them with a nested dropdown menu that is intuitive to any user familiar with a standard file explorer. An example of a hub configuration is shown below:



Data 28

aorta 3

- ✓ aorta_ENCFF115HTK
- ☐ aorta_ENCFF502AXL
- ☐ aorta_ENCFF974KVN

aorta_Input 3

- ☐ aorta_Input_ENCFF209QWM
- ☐ aorta_Input_ENCFF264DFJ
- ☐ aorta_Input_ENCFF626FKO

colon 2

- ☐ colon_ENCFF332UYK
- ☐ colon_ENCFF873WWR

colon_Input 2

- ✓ colon_Input_ENCFF788LJE
- ☐ colon_Input_ENCFF937EBV

skeletalMuscle 6

- ✓ skeletalMuscle_ENCFF000BMB
- ✓ skeletalMuscle_ENCFF000BMD
- ☐ skeletalMuscle_ENCFF063EVY
- ☐ skeletalMuscle_ENCFF280HQO
- ☐ skeletalMuscle_ENCFF290WZR
- ☐ skeletalMuscle_ENCFF743PUV

skeletalMuscle_Input 4

- ✓ skeletalMuscle_Input_ENCFF000BJZ
- ☐ skeletalMuscle_Input_ENCFF000BKA
- ☐ skeletalMuscle_Input_ENCFF003XTF
- ☐ skeletalMuscle_Input_ENCFF208RCO

Figure 2

The hub will also provide ways for users to add or remove labels from individual tracks. A stretch goal for us would be to implement updates to the interface in real time.

4.1.1.3 Domain

Peaklearner will have to be hosted on a website so users can access it whenever they need. At the request of our client, this domain will be: *http://peaklearner.rc.nau.edu/*. By hosting it through NAU, no additional domain will need to be set up, only a branch from NAU's domain. Even though it is hosted through NAU, anyone with the link will be able to access Peaklearner and sign in.

4.1.2 User Authentication

PeakLearner will implement user account profiles that can store their imported data and their modifications to said data in our online database. We will then authenticate those profiles using Google's API.

4.1.2.1 User login

Our user accounts will be authenticated using Google sign-in. Google has a publicly available API that once implemented will handle all the login information of our users, store and encrypt their credentials, and allow us to avoid the security risk of implementing our own encryption system therefore maximizing user privacy.

- **Google Sign-In Implementation:**
  Google’s API implementation is a straightforward process that involves creating credentials on our database for Google’s API and then associating credentials given by Google accounts to our own local profiles. This would allow us to have user profiles without implementing our own encryption or login systems, therefore making our system much more secure.

- **Modularization:**
  To keep our project open to adjustment regarding our chosen login method we will implement the Node.js middleware *Passport.js* to implement Google sign-in. We will do this because it would provide improved modularity by allowing for quick implementation of potential future authentication services.

4.1.2.2 User Data Storage

In order to provide user authentication, group collaboration, and hub editing we will have to use a non-relational database that can store many different datatypes in the same, quickly accessed database. Our client will have a primary implementation of the database that stores links to bigWig files and other various data types. This means that we must integrate our authentication data into the database our client sets up.

- **User profiles**
  After user credentials are implemented and associated with user profiles, we can use the profiles to create a consistent set of data and interaction across multiple platforms. This would include links to track data files in a user profile so that they need only upload it once to their profile for continued ease of use. Additionally, a user's data operations with those tracks should be associated and stored with the user’s created profile so that they can access those changes remotely and share them with their fellow collaborators.
  The storage of these profiles is associated through Google Sign-In credentials and the profile information would be dependent on a database. Stored user profile data will also include a user’s affiliated metadata, groups, and their corresponding permissions all in our local database.

- **Cross-Platform Data Handling:**
  The advantage of the data consistency given by a user profile would allow researchers to practice their analysis over multiple devices seamlessly and reduce the need for constant storage and backup solutions. Such an improvement would allow for a more efficient analysis process and reduce the risk of data loss.

4.1.4 Database API

The main form of communication between a user and our database will be through our API. This API will allow users to be able to indirectly communicate with the database, letting them add and remove data from their account as needed. This will be done with both a graphic user interface and a terminal tool.

4.1.4.1 Data Storage

In order to provide user authentication, group collaboration, hub management, and label interaction, we will have to integrate a non-relational database that can store many different datatypes in the same, quickly accessed database.

- **User Profiles:**
  An authentication service that requires a username and password would be dependent on a database to store this information. Stored user profile data will also include a user's affiliated groups and their corresponding permissions. Depending on a user's permissions, certain track data files stored as links in the database will become available to the user.

4.1.4.2 Terminal Tool

Our client has requested that we implement a terminal tool to interact with the API that we develop, with included documentation for how it is used. This is so that our client and future developers can use our API without needing to understand the source code's workings. Essentially we are to abstract the processes done in the API into easy to use commands that will accomplish a specific task.

4.1.5 Collaborative Groups

Implementing collaborative groups in PeakLearner will primarily involve the sharing of certain data such as hub configurations, labels, etc. We will also have a simple permission system in which the user profiles will have access to the permissions of access and editing hubs, as well as the ability to modify permissions themselves.

4.1.5.1 Permissions

Implementing permissions will be done similarly to how it is done in a lot of group-based software. Using our authenticated profiles a user has the ability to upload data and save it to their profile; they can also create groups and invite other users (using their Google account). All of these structures will give the creator supreme privileges over them automatically.

- **Access to Data:**
  The ability to access a hub from a user will at first be hidden to anyone without expressed access by the profile which that data hub is stored on. That user will invite other users to the hub by selecting a button prompting the entrance of a Google account (in email form).

  The creator can also define the specificity of those permissions in a checklist format where a user only has certain permissions granted to them by a superior user. The system will contain these kinds of privileges:
  - The ability to change the publicity of a data hub to all users (ie public, private, unlisted).
  - Viewing of a data hub and generated models
  - Modification of labels (addition or removal)
  - Modification of data tracks.
  - The right to grant or revoke privileges from lower level users.

4.1.5.2 Collaboration Method
Our goal is to create seamless collaboration between users that allows them to share data and their modifications to those pieces of data. To achieve this we must implement the infrastructure for the sharing of data in a simple fashion that the user can access simply by using the UI. Additionally, we need to make sure that data sharing is consistent across each user's system.

Allowing collaborators to access the same piece of data is a complicated process because of the range of possible issues encountered with real time updating of shared data. The current system implemented is asynchronous meaning each collaborator's personal use of the system is mainly independent and so is their data. Problems may arise when two processes attempt to access the same resource simultaneously, so we will simply prevent any overlap of recourse changes.

- **Checkout System:**
  When a user starts to change a piece of data, for instance a gene track, the system will "check out" that data for the user until they have completed their changes. During this checkout period any other access to the resources under modification will be locked until the user has finished their changes and checked their data back into the shared system.

4.2 Performance Requirements

To ensure the effectiveness of our software we must consider four attributes and see that they adhere to a certain set of standards. These attributes include: *speed*, *security*, *simplicity*, and *maintainability*. Below we will analyze and then consider how to address these requirements.

4.2.1 Speed

In order to get PeakLearner fully functional, the system must be able to access information stored in the database extremely quickly for authentication. Enabling features such as hub configuration, authentication, and group collaboration requires fast data access speeds. It is requested by our client that the web interface itself authenticates the user and loads the web page in 1-2 seconds, which is the average load time for most websites. Fast data access speeds from the database are dependent on two things that we will discuss below: input lag, and latency.

- **Input Lag/Latency:**
  Input lag, or the speed at which the interface reacts to user input is dependent on how quickly the inputted user information interacts with the database. For user authentication, this means that our system must be able to access user information quickly enough to load the proper webpage in 1-2 seconds as stated above.

4.2.2 Security

Since we are planning to use Google's login system, we will also be using Google's security for accounts. Data is encrypted through Google using AES256. This form of encryption is based on the substitution-permutation network principle which uses a series of linked operations. This form of encryption is used by the United States government to keep data safe. Users should still do whatever they can to protect their Google account, but using AES256 keeps a user's information as safe as possible in the event of a cyber attack.

4.2.4 Simplicity

The importance of simplicity is mainly pertinent to our user interface. The makings of a good user interface involve the user being able to find what they need through visual affirmations making navigation from wherever in the application they are. Additionally, we want to implement a system where a user can reach any option in the interface from the main page in around one to three clicks. We can accomplish this by asking unfamiliar users of different backgrounds on the software to complete acceptance testing. If the user cannot locate the UI feature they want we will determine where they would like to have access to it. If the user cannot figure out what they need to use to accomplish the task in the system we will adjust our UI and documentation to fit their feedback.

4.2.5 Modularity

Modularity is important to our code because it allows the client to further expand our software to their liking at a much lower cost in terms of time and resources. This is a responsible software engineering practice and one which we must implement to provide the best product possible for our client. To accomplish this we will make sure our software is modular in terms of user authentication and in its extension of JBrowse.

- **User Authentication:** This will be implemented using the *JavaScript* framework *Passport.js* which allows for use of multiple user authentication methods simply. It will cover what would be several lines of code, and several processes in a single function in which we specify our method of authentication. For example, we will input the *"Google"* as a parameter and implement our servers credentials with Google's API and our work is done. Then to simply add another or change the authentication process, we only need to change, or add another parameter without needing to change any fundamental aspects of the login system itself.

- **JBrowse:** Currently PeakLearner is built on top of JBrowse so we can use its core functionalities. It is implemented as a submodule of PeakLearner, meaning we can update as updates are released for it or we can retain whatever version works best for us. This makes our software more independent than a simple plugin usually would be because our software does not automatically break with each update. Instead, we retain the previous version as a submodule in our system. We can then update our submodule and update PeakLearner to adjust if necessary or simply just keep the older JBrowse submodule. Essentially this gives us multiple options and makes our software reliable.

4.3 Environmental Requirements

Our team must establish our environment requirements before we begin development. This will help us prepare a clear solution by knowing and understanding the necessary tools for development.

4.3.1 JBrowse

PeakLearner is fundamentally built on JBrowse which is currently implemented as a submodule in which all of PeakLearner's functionality extends. Our software, being an extension of JBrowse, means we need to understand JBrowse to implement it effectively. We should also keep it updated through our modular updating process described above (see 4.2.5).

4.3.2 Linux Environment

To continue our client's work on the PeakLearner project, we must set up Linux development environments. This is because the systems in which the server and database run are reliant on certain software found on the Linux platform. On our computers, if we want to run a local server with full functionality for testing and development, we need to run it on the Linux operating system.

4.3.3 Google Authentication

Our client requested the use of Google Sign-In because most individuals already have a Google account and implementation of this process would be secure. This means that PeakLearner is reliant on the functionality and stability of the Google API so that our login and user profile systems continue in functionality.

4.3.4 Berkeley DB

Our client, Dr. Hocking, has requested we use Berkeley DB as PeakLearner's database. This is required for multiple reasons, the first being that it is a non-relational database which will allow for the storage of multiple data types in the same database. It is imperative that we can do this because PeakLearner's database will need to store user data, permissions data, track data links, label data, and hub configuration data, all of which are different data types. In addition, it is imperative that the implemented database has extremely fast data accessing speeds as discussed in section 4.2.1. According to Dr. Hocking, Berkeley DB is the most suitable database for the time and data type constraints in PeakLearner.

4.3.5 Testing

With each iteration of the project, we are expected to complete headless browser testing in a Travis-CI/Selenium/PhantomJS module. This will allow us to test the functionality of the user interface. During implementation, we will conduct unit testing on individual modules to ensure the full functionality of every feature we implement.

4.3.6 Documentation

One way to increase our software's maintainability and simplicity is to include good documentation. This documentation will contain instructions about the different ways to use our GUI and terminal tool. The documentation for our GUI will describe every possible way to add or remove labels from a track, add or remove tracks from a hub, add or remove hubs from an account, create groups, or adjust permissions. It will also provide ways to do all of these things with the terminal tool as well. The documentation will have its own tab on the main page and will be included in our GitHub repository.

5.0-Potential Risks

TEAM BLUEBOX

Every project faces potential risks and ours is no difference. Common risks may include function upgrades, troubleshooting, or shifting requirements of the clients, etc., which leads to the diversity and uncertainty of projects. In this section we: a) highlight risks, b) determine whether they are internal or external, c) assess their severity, d) likelihood, and e) find solutions.

5.1 Non-technical Risks

The current climate is one of stress, fear, illness, and uncertainty. These factors cause anxiety in countless lives and unfortunately due to some of these conditions and several personal factors as well it is a possibility that our team experiences some drawbacks in productivity. This would occur because a member cannot perform to their optimal level and would thereby affect the project schedule and delivery commitment. In addition, it is possible that we may lose a team member throughout the development process. However, with better team communication and commitment the consequences of these things can be relieved. The risks of lowered team productivity and loss of a team member are explored in further detail below.

5.1.1 Internal and External Factors

There can be multiple factors that cause an individual's productivity to decrease, some internal and some external. Internal factors include not considering personal physical and mental health, as well as physical illness. External factors include work obligations, living conditions such as moving, and global restrictions to travel such as COVID-19. Similar internal causes, like personal crisis, may cause the loss of a team member. External causes of this might include COVID-19 or global restrictions.

5.1.2 Likelihood and Severity of Risks

If an individual's productivity is lowered because they are faced with such conditions, the pace of the project development will decrease causing the team to operate less effectively and increasing pressure on the rest of the team. For this reason we consider this to be of low to moderate risk. The possibility of losing a team member would be quite serious for our team, but not catastrophic. As our team currently has five members, we would be able to re-distribute the workload amongst the four remaining members making this a risk of moderate severity.
On our team we have a member who is an international student. Due to this factor combined with the uncertainty of COVID-19, lowered productivity and/or loss of a team member is more likely

than usual. Additionally a few members of our team have family members that are especially vulnerable to the pandemic. There is always a medium risk of one of these individuals being unable to continue working with us. However, it is very unlikely that more than one member will be unable to continue their work.

5.1.3 How to Mitigate

In order to mitigate these risks, we will need to assess the impact of solutions provided by project team members, update the project management plan and keep relevant parties informed, require team participation to manage the situation, and update project measurement indicators to determine the possibility of adding new resources. We will also need to contact the missing team members in time to ensure that the project can return to its original workflow as soon as possible.

5.2 Technical Risks

Because PeakLearner will be integrated with software that is not maintained by our team, like Monsoon and Google's authentication API, we face the potential risk of not being able to access the integrated PeakLearner functions during a server shutdown. These risks will be examined below.

5.2.1 Internal and External Factors

There are few factors that can cause a server shutdown for the Monsoon server, most of them are external. For example, if the system needs maintenance the server will be temporarily offline, meaning that PeakLearner will be unable to generate models which is a part of its main functionality. As our authentication system will use Google's authentication API, if Google's servers go down, PeakLearner's users will be unable to login and access track data.

5.2.2 Likelihood and Severity of Risks

We have determined that although a Monsoon server error has the potential to be a severe issue, it usually only happens during pre-planned maintenance times. This means that our team will be able to plan around these days which lowers the overall severity of the risk making it a moderate risk. On the other hand, because Google's servers rarely go down, the severity of that risk is considered low. While still a possibility that Google's servers shut down, it is extremely unlikely given the size and scale of the company.

5.2.3 How to Mitigate

Unfortunately, the Monsoon server is maintained outside of the scope of this project so our best mitigation strategy is to plan around the scheduled maintenance periods. We can, however, mitigate the consequences of a Google server shutdown by implementing passport.js into our authentication system.

6.0-Project Plan

TEAM BLUEBOX

Fall Schedule:

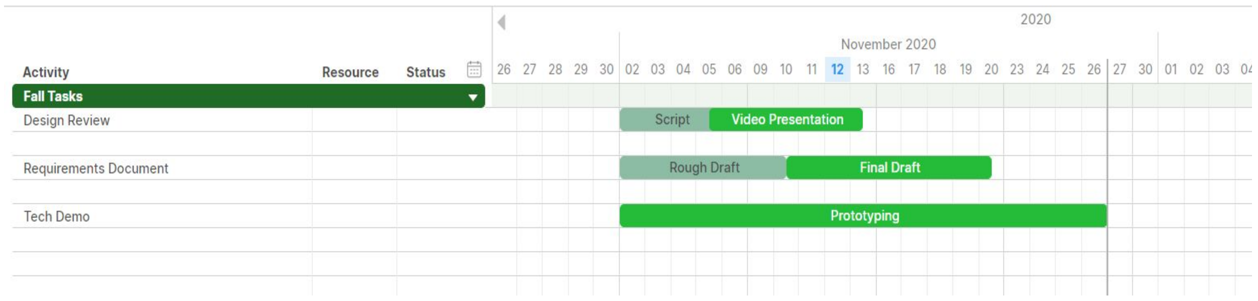


Figure 3: Completed Fall Schedule

Spring Schedule:

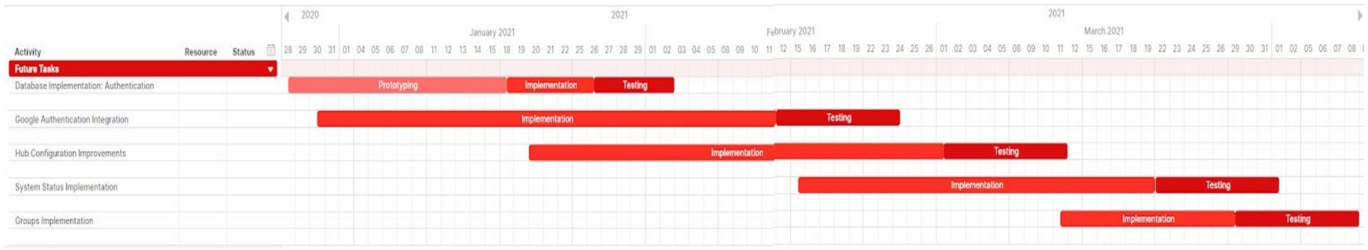


Figure 4: Future Spring Schedule

- **Database integration (December - February) -** As database integration is the first step to implementing our other key requirements, we will begin the process before the start of the semester. This way we will be able to test its functionality by the end of January as seen in figure 4. Throughout the implementation of the database API, we will likely include a command line interface.

- **Authentication (January - February) -** We estimate that the implementation of user authentication in PeakLearner will not take a very long time. It is for this reason that authentication will be implemented during the database integration process, as well as simultaneously with the hub configuration seen in figure 4. During the creation of this feature we will also create a log-in page and about page tutorial.

- **Hub Configuration (January - March) -** Hub configuration development will include the implementation of new features like the deletion of hubs for a user and the improvement of pre-existing ones. We estimate that this process may be lengthy so we will likely not begin testing these features until the beginning of March.

- **System Status Implementation (February - April)** - With the proper implementation of a well indexed database, implementing the system status update feature will not be incredibly time consuming. For this reason, as seen in figure 4, we have designated a two month window with testing to be completed in the beginning of April.

- **Groups Implementation (March - April) -** This is likely the last thing our team will implement for PeakLearner as it will least affect the software's ability to function. This will be completed with testing by the end of April so that final, post-development tests can be done in May.

- **User Interface (January - May)** - The development of the features listed above will ultimately contribute to the overall user interface of PeakLearner. Throughout the process of development from January to May, we will be implementing these features to build a fully functional GUI. Post-development testing for the functionality of the interface as a whole will be completed in May.

7.0-Conclusion

The study of genomics has an incredible potential for improving human health and medicine. Gene science, like most other medical advancements, can be greatly improved by furthering the technology used to understand it. Our sponsor's machine learning prototype does just that. However, in its current iteration, it is only available for use as an R package which means whoever uses it must be proficient in programming. As a result of this, our goal is to create a web application that allows for interaction with the peak prediction data in a comprehensive and effective way. In order to do so we need to implement four key functional requirements: user interface, user authentication, database API and collaborative groups.

These key functional requirements need to be implemented in such a way that they adhere to the examined performance and environmental requirements. The database must allow for data accessing speeds that reflect real time updating as well as being functionally simple to use. One the database has been implemented and is able to store user and permissions data, as well as label and hub configuration data, we will be able to implement the other key requirements. User authentication and group collaboration will likely be implemented at the same time as the latter is heavily dependent on the former. The primary requirement of PeakLearner according to our client, data operations, will rely on a successful implementation of the database.

Throughout the course of this document, we have identified the key requirements we need to implement for the successful production of PeakLearner as a web application that would make our client's machine learning prototype available for regular use. We have determined the speed, simplicity, maintainability, and runtime requirements the key requirements need to adhere to. As we have already determined the most feasible technologies to implement these requirements with, our team is ready to begin prototyping the PeakLearner web application.

As we begin our implementation process, it will be important for our team to mind the risks we determined in our risk assessment. These risks range from non-technical risks like losing a team member, to technical risks like the Monsoon server being down. Should these risks occur, we can overcome them by implementing the risk mitigation strategies described in our risk assessment section. In doing so, we can ensure a smooth development process for PeakLearner moving forward.

8.0-References

[1] *Genetic Genealogy: Understanding Ancestry DNA. (2016). Retrieved from https://whatisdna.net/wiki/genetic-genealogy-understanding-ancestry-dna/*

[2] *Computational Strategies for Scalable Genomics Analysis. (2019). Retrieved from https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6947637/*