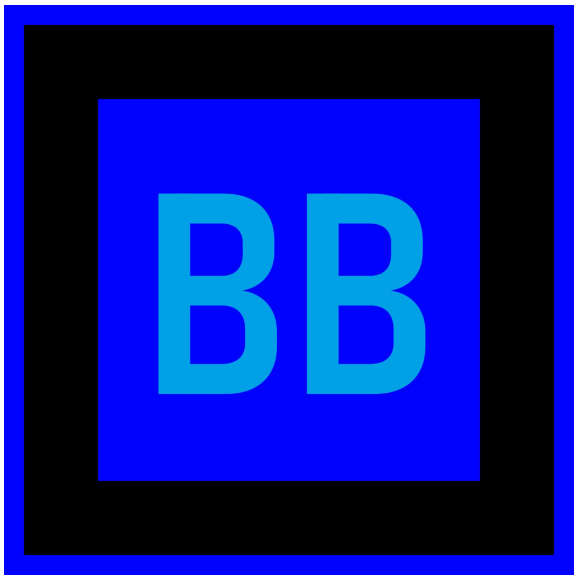
Technological Feasibility



Team Blue Box

Members

Bradley Barber

Melina Diamond-Sagias

Jonathan Hillman

Yunsong Wang

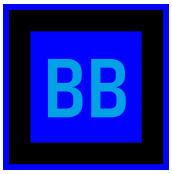
Zachary Wilson-Long

Mentor: *Tomos Prys-Jones*

Client: *Dr. Toby Dylan Hocking*

Contents

1.0-Introduction



TEAM BLUEBOX

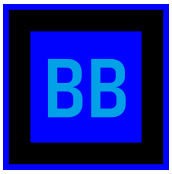
The incredible improvement modern medicine has made on human health can often be attributed to the advancement of technology. While technological advancement allows for more effective tools and medicines, it also provides medical researchers access to information that was previously unavailable. One notable example of this is gene science, which has become one of the most prominent medical advancements of recent years. As genomic sequencing technology has increased in efficiency, it has also significantly lowered in price. Its rise in accessibility and effectiveness has allowed 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. Due to the information density of genetic sequences, it is difficult to differentiate non-coding sequences from coding sequences. In addition to that, every persons' genome is slightly different, which translates to slightly differing peaks, meaning there is no single pattern for determining an active coding region.

It is for these reasons that our project Sponsor, Dr. Toby Hocking, developed a machine learning model for recognizing active coding regions. Dr. Hocking's peak prediction algorithm detects these active regions, or "peaks", and reports a visual model that predicts where the peaks will be in a continued sequence. Unfortunately, this prototype is currently only available as an R package which means that any genomic researchers that want to use the software need to have extensive knowledge in programming. Programming proficiency is not often a requisite for genomic researchers, so PeakLearner's current iteration increases the boundaries to entry for machine learning based genomic analysis. While this is not a problem for our project sponsor specifically, it will inhibit widespread use of the software.

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 adding and modifying labels, as well as HUD configuration. 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, it will also increase security.

After determining the web application as the solution to our client's problem, we begin a more careful analysis of the technological challenges regarding the development of PeakLearner. Broadly this document will provide a detailed description, quantitatively compare alternatives, and choose the one that scores most highly. The following section describes these sections with an explanation of how each individual solution will integrate into one overarching software solution. Thereafter the main sections will be a description of each technological challenge faced by our group.

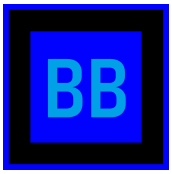
2.0-Technological Challenges



TEAM BLUEBOX

- Creating a secure user authentication feature on our web application
- Providing group collaboration across projects
    - Creating a permissions management functionality for groups
    - Allowing for real-time tracking and display of data changes for all group members
- Uploading R data files into our data storage system
    - Maintaining and manage the stored data
    - Updating and recording the additions to the data in the storage system
- Allowing users to interact with the data
    - Enabling users to modify labels
    - Enabling users to modify the hub configuration
- Providing the web application the ability to display graphical representations of the data
    - Integrating JBrowse with our web application to do this
- Providing a user interface to interact with the application

3.0-Technology Analysis



TEAM BLUEBOX

In this section we will discuss the various technology and methods that may be used to solve the challenges from section 2,0. Each section analyzes various desired characteristics, alternatives, and the benefits/disadvantages of each alternative. This is used to give a more specific explanation of each challenge and the solutions we have come up with.

3.1 Authentication and Security:

3.1.1 Introduction

Having a profile and login system provides a lot of flexibility in terms of user interaction with the system and other users. Having secure authentication is a vitally important non-functional requirement to protect a software's users and their data whether it is personal or for research. Currently, the system in use by the sponsor does not include authentication. From a security standpoint, this could lead to data leakage of research information. Additionally, once we start implementing user profiles, authentication will become requisite as we cannot risk the personal and research data of our sponsor or any future users .

3.1.2 Desired Characteristics

- *Security:* Protecting user information and credentials is a non-negotiable requirement for PeakLearner because of the importance of privacy and data security that could occur from information leaking. Failure to provide a secure application would counteract our team's progress of making good software.
- *Simplicity:* A simple user authentication is important because it will allow new users to quickly access the system and will reduce struggle for existing users when accessing their profile if issues do occur.
- *Usability:* Due to the large scale nature of gene science, it is common for scientists and developers to work in groups. We want PeakLearner to be group-friendly so we can impact the gene science community in the most powerful way possible. Additionally members in those groups should be able to authenticate as seamlessly as possible, preferably using accounts they already possess such as a Google account.
- *Client Preference:* For our project, client preference plays a large role in the process of selecting desired softwares. Unlike most groups, our client is maintaining part of the

server himself. His inclusive role in the development of our project makes his preference an important factor when designing our software.

3.1.3 Alternatives

1. **Google Sign-In and OAuth2.0:** Using Google's Sign-In functionality of their API we can implement user authentication by incorporating Google accounts to use as credentials in our systems user profiles. Google uses OAuth 2.0 to package credentials and user information from Google so that we can use Google accounts without compromising user security. [1]

2. **Using Passport.js to Implement Our Own Authentication System:** Using the Passport.js middleware to securely implement our own authentication system storing user credentials on our client's database. [2]

3.1.4 Analysis

1. **Google Sign-In and OAuth2.0**
   Google Sign-In is a feature of Google's API that manages OAuth 2.0 flow and token lifecycle. Essentially, it automates the authentication process. This means we don't have to deal with user authentication and private credentials so we are not responsible for storing personal user information and encoding encryption on that aspect of the application.

   To begin this process we need to set up an OAuth2.0 client for PeakLearner, load Google's Platform Library, and then connect Google's user profile to our own stored profiles using a simple function from the Google API. That is all that is needed to implement the authentication for our users.

   After that, our personal user profiles can be granted permissions and attributes that simply require a Google account to access.

   Pros:
   - It is very quick and easy to implement which would allow the majority of our time on this project to be allocated to more functional requirements while still achieving the client's desire for user authentication.
   - Most individuals already have a Google account of some sort, including our client's NAU account, which allows a great deal of accessibility for new users.

They will likely not need to create new login information, allowing them to access the application much more quickly.
- Additionally, having a Google account provide the login information will allow groups to invite users who have not yet established a profile on the account simply by inviting them through email.
- Because our client already regularly uses Google services, the client has specified their preference of this method.

Cons:

- Using this method means that we are subject to Google's platform, meaning that if their encryption is compromised our client's information would be at risk. However, it should not impact any information stored only on our system due to the use of OAuth.
- Reliance on Google's platform means that if for any reason their servers or API become unoperational then users will be unable to access our system until Google's services resume. However, this is not likely because Google is a trusted company with a multitude of servers and information backups, meaning that their ability to host user information is much more reliable than anything we could implement ourselves.

2. **Using Passport.js to Implement Our Own Authentication System**

Using Passport.js we can store user credentials using a number of different "strategies" (including Google Sign-In) with a local strategy being one of them.

Pros:

- Passport.js allows for better code modularity meaning we can switch strategies for authentication very simply. We could potentially use this in conjunction with Google Sign-In for future modularity.
- Passport.js was designed to be easily integrated into other frameworks. As authentication is an important, but non-functional requirement, this ease of implementation would be appreciated by everyone on our team.
- Passport.js is known to be very thorough. With over 300+ authentication preset strategies, Passport.js is among the safest authentication softwares available, provided the implementation is done correctly. It also offers custom authentication and encryption algorithms, making it very popular.

Cons:

- Implementing our own system has the potential to create vulnerabilities in security. Because none of our teammates are familiar with proper cybersecurity

protocols, implementing our own system could be extremely dangerous. After the first month of shutdown due to COVID-19, the rate of cyberattacks increased by 500%, and that number has only been increasing since. This is the best time to focus on cybersecurity, so taking it for granted could be disastrous. Additionally, the client specified the need for a secure software due to the nature of the genomic sequencing data it will manage.

- Passport.js is designed to be a bare-bones framework for user authentication. Because it is meant to be built upon by the developer, it will require significantly more time and effort to implement fully compared to some of the given technological alternatives. This may be considered ideal for projects with more specific and complicated requirements for user authentication, but for the simple requirements of PeakLearner's authentication functionality, it is more of a hindrance.
- Due to lack of familiarity, the client has specified their preference of an alternative method.

3.1.5 Chosen Approach

Using the four desired characteristics mentioned in section 3.1.2, we will provide each user authentication technology reviewed in sections 3.1.3 and 3.1.4 with a score between 1-5 as shown in *figure 1*.

Figure 1

Name	Security	Simplicity	Usability	Client Preference:
Google Sign-In	5	4	5	5
Passport.js	3	5	5	3

Taking into consideration our analysis and client preference we have decided to implement our User Authentication System using Google Sign-In as the method in which users can create profiles. Ideally we would want to use Passport.js to allow authentication from Google Sign-In and our own local accounts so that we are not reliant on Google's platform to have user access.

However, we have chosen to opt-out of creating our own authentication due to a few factors. Firstly our team has a limited amount of experience when it comes to cybersecurity. This means

that we should not have access to user credentials at this time because we are not able to foresee every detail required for maintaining our users' privacy. Additionally, due to the scope of this project, we have concluded that having local accounts is not a necessity for consumer use of this application. We have also determined the benefits to its addition would be miniscule and rather unnoticeable, making it a stretch goal that is of very low priority.

Our main goal is to implement Google Sign-In to ensure an easy and reliable authentication process. Ideally, we would use Passport.js as the middleware between our local profile system and Google Sign-In to combat any potential server issues with Google. This would be functionally identical to simply implementing Google Sign-In but would allow for future expansion should the client choose to do so. However, integrating the two will require a significant effort on the part of our team. As this is not necessary for the functional implementation of PeakLeaner's user authentication service, it is seen as a stretch goal.

3.1.6 Proving Feasibility

The method of authenticating with Google Sign-In will be proven in our Technical Demo at the end of the semester. In the demo we will implement a basic login system using Google Sign-In as the method of creating a user account. This will ensure that the use of personal data tracks in PeakLearner cannot be accessed outside of that profile. Before completion of the demo we will implement Unit Testing of user authentication and permissions to validate their implementation.

3.2 Server Management: Jon and Melina

3.2.1 Introduction:

PeakLearner deals with large amounts of differing data points by nature as a machine learning software. To prevent any misuse of the data, it is crucial to implement solid, concrete data management methods with a database that can quickly handle them. PeakLearner will need to access a diverse set of data inlcuding: label data in CSV format, model data, and user/permission data. As users are meant to interact with the data, it is necessary that we store it in a database that will allow for easy and fast access. Ideally, the database should be fast enough to reflect real-time changes of the data.

The progress from last year's capstone group provided a new constraint: we must use a non-relational database. Last year, the team working on PeakLearning used a relational database which was much too slow to provide quick accessibility to the database. From their experience, we now know that to create a satisfying product, we must incorporate a non-relational database.The two non-relational databases we analyze below share the characteristic of being a key-value database. Key-value databases hold data as values and are indexed/accessed with keys.

The benefit of this type of database is that data values with different keys do not need to share the same data type. As this is important functionality of PeakLearner, we decided on Mongo DB and Berkeley DB as they are both non-relational and key-value databases.

3.2.2 Desired Characteristics:

- *Speed:* With the increasing rate of data collection and computational capabilities, data driven applications have become extensively used. Data analysis can apply to a wide range of fields. On Glassdoor jobs, data science was ranked as the #1 field in the US in 2019. However, when the data required to perform an analysis is large, as it is naturally with genomics, runtime becomes an even more important factor in the quality of a product. Speed is one of the most important characteristics that we need in our database, and investing in a database with speed will help produce faster results and real time updates.
- *Simplicity:* As team BlueBox is relatively unfamiliar with database software, we must measure how difficult it is to manage and retrieve the data with any given database. Given the complexity of the project itself, we would like to minimize the time in which we must spend figuring out database operations.
- *Data Handling:* Our database must be able to handle different types of data. Most importantly, our choice of database must be able to store different attributes of genomic sequences. However, since the majority of the machine learning is done with R, it is necessary for it to be able to store RDS (R Data Structure) files as well. This data handling capability will provide swift and efficient integration with professional machine learning techniques provided by Dr. Hocking.
- *Client Preference:* For our project, client preference plays a large role in the process of selecting desired softwares. Unlike most groups, our client is maintaining part of the server himself. His inclusive role in the development of our project makes his preference an important factor when designing our software.

3.2.3 Alternatives:

1. **Berkeley DB:** Using Berkeley DB we can satisfy the high-speed data access requirements of PeakLearner, Berkeley DB also has support for many different data structures and has very little limits on what data types it can hold. This feature is important as it ensures that we can implement a database that holds all of the required data types[3].
2. **Mongo DB:** Mongo DB will perform data access at a speed faster than a non-relational database, which is necessary for the functionality of PeakLearner. We will be able to

choose from a variety of data management operations that will be useful when implementing the database[4].

3.2.4 Analysis:

1. **Berkeley DB**
   Berkeley DB is a non-relational database that allows for a lot of variation in data types and how they are structured. In terms of data access, it is known to be extremely fast due to its embedded nature. This is extremely important when considering what database to use as PeakLearner has high performance speed requirements.

   Pros:

   - Databases of up to 256 terabytes can be managed by Berkeley DB. It also supports a variety of data structures including hash tables, binary trees, persistent queues and record-number-based storage. Its large capacity combined with a diverse array of supported storage structures means that Berkeley DB will be able to effectively store and manage the multiple data types required by our client.
   - Berkeley DB works on multiple operating systems. Although our product is only required to run on a Linux OS, having access to Berkeley DB on a Windows OS allows the rest of our team to explore how this software works without needing to use a virtual machine.
   - Berkeley DB is an embedded database. This means that the database library is embedded inside of the web application. Embedded databases offer extremely fast data access as well as low latency writes. Low latency writes allow for quick read-write on applications, which would allow an almost real-time reflection of changes made to a data set across any group collaborators in PeakLearner (see section 3.4).
   - This database software is the first choice of recommendation from our client due to its potential for quick and diverse data management.

   Cons:

   - This software is considered difficult to use or implement. Since Berkeley DB isn't as popular as other database software, there are less documentations and tutorials made readily available for it. Additionally, Berkeley DB is known for being hard to debug, especially when there are improperly formed data constraints. The complexity of this software creates a high learning curve for our team.
   - Berkeley DB only provides simple data access services. Data accessed in the database is done so with key-value pairs, however Berkeley DB differs from other

key-value databases in that it never operates on the value part of the record. This means that the only four operations offered by the database manager are: inserting, deleting, finding, and updating records. While this does severely limit how the user can interact with the data in the database, this can be compensated for with data management functions provided directly in the web application.

2. **Mongo DB**

Mongo DB is a popular, easy to use, non-relational database software. It is also a document database, meaning it stored things as 'objects' which contain the key/value pair. In terms of functionality, this effectively makes it a key/value database. Unlike a regular key-value database, however, document objects can contain entire hierarchies which may make accessing that much faster.

Pros:

- Mongo DB is currently one of the most popular non-relational databases, meaning it has significant documentation. This contributes to it being notably easy to use, especially for beginners. As our team is largely unfamiliar with non-relational databases, the simplicity it offers would be extremely convenient.
- This database has the potential for very fast query performance. Because it stores most of the workable data in RAM, rather than the hard disk, it is able to serve data access extremely quickly. The potential speed of a Mongo DB database satisfies the speed requirement necessitated by our client.
- Mongo DB offers a flexible data model, which means that the data object, also known as documents, can hold any type of data as a value. This is important as PeakLearner will have to manage many different data types.
- While this software is not our client's first choice, he is still open to using this software with us.

Cons:

- While Mongo DB has a great query speed capacity, this potential can go unrealized if indexing is done incorrectly. If key indexes are in a wrong order or are poorly implemented, the speed data access decreases drastically. As our team has never worked with Mongo DB, or any non-relational database for that matter, indexing will require extreme caution
- If implemented incorrectly, Mongo DB may duplicate, and in many cases even corrupt data. PeakLearner will be storing and interacting with data that represents genomic sequences, meaning that any potential data corruption or duplication could have important consequences.

3.2.5 Chosen Approach:

Using the four desired characteristics mentioned in section 3.2.2, we will provide each user authentication technology reviewed in sections 3.2.3 and 3.2.4 with a score between 1-5 as shown in *figure 2*.

Figure 2.

Name	**Speed**	**Simplicity**	**Data Handling**	**Client Preference**
BerkeleyDB	**5**	**3**	**5**	**5**
MongoDB	**5**	**4**	**5**	**3**

Considering the analysis shown above, we plan to use Berkeley DB because of its fast and reliable nature. In regards to data handling, it is extremely beneficial for this project that Berkeley DB does not put restraints on data. This means that both incredibly large sequences of data, and user authentication information can be stored in the same database without issue. Additionally, Berkeley DB's architecture is notably simpler than that of a relational database, or even other key-value databases which will further increase data accessing speed.

While there are significantly less data access operations in Berkeley DB compared to Mongo DB, this does not pose an issue to this project. The highly specific nature of the data PeakLearner analyzes, as well as its machine learning components, make it unlikely to be supported by many of the additional operations of Mongo DB. This means that any necessary data access operations besides the addition, deletion, updating, and searching of key values will need to be implemented in PeakLearner itself. However, it is very likely that this is an unavoidable step in the creation of the web application regardless of chosen database.

In addition to our careful analysis of the qualities of both database alternatives, we must consider the client's preference of technologies. Our client has specified their preference for Berkeley DB because of its speed and capacity for holding large and diverse data sets.

3.2.6 Proving Feasibility:

Because Berkeley DB uses simple data access techniques in a system that will be embedded into the web application, we will be able to prove feasibility during the Tech Demo. During our Tech Demo, we will provide a prototype database implementation. It will have to be able to access test data quickly enough to allow interaction with the visual peak models.

3.3 User Interface:

3.3.1 Introduction

To be able to more efficiently browse and use PeakLearner, a clean and easy to use interface must be created. This will provide a way for users to interact with PeakLearner without the need for a command line. An interface for PeakLearner must have the ability to perform operations on peaks and give an easy to read graphical representation of the data all wrapped into a minimalistic website design.

3.3.2 Desired Characteristics:

In order to make PeakLearner as accessible as possible for gene scientists to be able to perform operations and analysis on data, we need an efficient user interface. We have come up with the following desired characteristics to achieve this:

- *Ease-of-use:* This is how straightforward it is to use the framework. If it is harder to implement and has a steeper learning curve, the framework will score low in this category. This is important because if we have an easy to use framework then we can spend less time learning it and more time making it look nicer and work more efficiently
- *System architecture:* The framework we choose must be able to work with the current PeakLearner code easily. If it does not work easily with the code then we will either have to completely rewrite PeakLearner's framework from the ground up, or find some sort of workaround.
- *Speed:* Speed is measured by how fast a change updates on the page**.** An ideal framework must update the page almost immediately when a user edits something. This is important because we do not want users to wait for a page update to go through.

3.3.3 Alternatives

1. Angular: Angular is a popular open source software widely used for website development. Using Angular for our framework would give us a powerful tool that allows us to create client-side web application user interfaces. This method would let us use current knowledge to implement it.

2. Express: Using ExpressJS for our framework would give us a fast and minimal web application framework. This is built off of NodeJS which is extremely popular among Javascript developers.

3. React: React is a newer technology but is rising up in the ranks to be a strong contender against the other frameworks.

3.3.4 Analysis

1. **Angular**
   The angular framework is extremely modular and simple. Created by Google, Angular provides efficient modularity and simplicity for general website structures. Angular offers a clean, efficient way to develop our user interface.

   Pros:
   - Angular is versatile, it is one of the bigger frameworks and has a lot of functionality. Using this would give us the ability to add almost whatever we want/need to the UI.
   - The framework would let us use current knowledge to develop. It is built off the idea that the user can use HTML to implement Angular so there is not a steep learning curve.
   - It allows for two-way data binding which synchronizes data from the UI and Javascript objects. This would make it much easier to translate changes that users make into objects that can be accessed by the system.

   Cons:
   - Angular is rather slow, this is because of its high functionality. We can do a lot with this framework at the expense of speed. This would make it so changes to the UI might take longer than desired.

- Angular has many major updates. The problem with this is that with each major update there is a chance to break something done previously. This is common amongst webpages using this framework. Because of these frequent updates, it would make PeakLearner require more consistent maintenance.
- Because Angular allows you to use simple HTML to work with it can cause issues when you have to implement bigger features such as getting the current PeakLearner code to work with a new framework.

2. **Express**

   The Express framework is a famous software based off of node.js. It's low-level design makes it fast, and the developers who made it intended it to be scalable and easy to use.

   Pros:

   - Express is extremely fast. When developers have a need for a fast front-end on a website, they use express. It is common among web games.
   - It is minimalistic, this framework is designed to be simple and fast. It is a barebones framework whose only purpose is to get the job done. This means it is a lot easier to learn than most frameworks and does not have a lot of useless tools.
   - Integration with other code is extremely easy. Express allows old code to be "wrapped" in a Javascript object and placed wherever needed. This makes implementing the framework with PeakLearner much simpler
   - Express is built off of NodeJS. Colleges often use NodeJS to teach their web development classes, so all of the team is somewhat familiar with it.

   Cons:

   - If there is an issue that Express cannot address, a middleware package must be installed and implemented which will sacrifice some of this framework's speed.
   - Because Express is event-driven, testing would be very difficult and could leave a large amount of unnecessary code

3. **React**

   The React framework is an open source software for web development. It is known to be extremely flexible and modular. The developers of React use modern and innovative techniques with the intent to make React the most popular web development interface on the market.

   Pros:

- Virtual DOM, shows a logical layout of the document in an easy to read DOM tree. This will allow us to understand the code and flow of the program much better, which could help quite a bit with testing and bug fixing
- React is a newer and rising framework which seems like it is going to be around for quite a while. Developing with this framework may get PeakLearner ahead of the curve if everyone starts switching over to React
- One-direction flow reduces the chance of hierarchical bugs. Since React only flows in one direction, a child process cannot affect a parent process. This makes for less likelihood of major bugs affecting the entire UI.

Cons:

- There is not a lot of existing documentation. Since React is newer technology, there are not many tutorials or documentation yet. This might make troubleshooting a lot more difficult
- React has an unstable API. Most newer technologies are unstable at first until they can get their ground. This is the same case with React, it changes constantly which could make for a maintenance headache

3.3.5 Chosen Approach

Using the three desired characteristics mentioned in section 3.3.2, we will provide each user interface framework reviewed in sections 3.3.3 and 3.3.4 with a score between 1-5 as shown in *figure 3*.

Figure 3.

Name	**Ease-of-use**	**System Architecture**	**Speed**
Angular	4	1	2
Express	4	3	5
React	2	1	4

Evaluating the table, our clear chosen approach is to use the ExpressJS framework to develop our user interface for this project. It is faster and will work better with PeakLearner's current code than the other two alternatives.

Another big reason we decided to go with ExpressJS is because of its NodeJS backbone. NodeJS is something we have all worked with and it is a technology that is a staple in the Javascript community, so it will likely be in use for a long time. This means that PeakLearner's UI will not become outdated anytime soon. Another note is that since NodeJS has been around for quite some time, it does not have many major updates anymore which makes continuous code maintenance unnecessary.

3.3.6 Proving Feasibility

We will be showcasing our use of ExpressJS in our tech demo at the end of the semester. In this demo we will create a basic user interface that will allow the user to interact with a few basic objects on the page and showcase what these objects will do. The purpose of showcasing the UI in our tech demo is to prove its speed and effectiveness to allow for a seamless user experience of PeakLearner.

3.4 Collaborative Groups Functionality:

3.4.1 Introduction

Having flexible interaction between users in group structures is important for allowing efficient collaboration between researchers. Including user profiles and groups will allow for a more streamlined use of the program. With the complexity of genomic analysis, collaboration on the same data is a necessity. However, JBrowse does not currently offer support for group collaboration, which makes collaboration inefficient.

3.4.2 Desired Characteristics

- *Simplicity:* A simplified approach will allow for a viable product with minimal errors. Additionally, since most of Team BlueBox is new to developing this type of system, simplicity has a significant weight on the selection of a collaboration style.
- *Usability:* Due to the large scale nature of gene science, it is common for scientists and developers to work in groups. We want PeakLearner to be group-friendly so we can impact the gene science community in the most powerful way possible.
- *Client Preference:* For our project, client preference plays a large role in the process of selecting desired softwares. Unlike most groups, our client is maintaining part of the server himself. His inclusive role in the development of our project makes his preference an important factor when designing our software.

3.4.3 Alternatives

1. **GitHub-Style Collaboration:** An open style of collaboration with groups of users, in which there will be one or more moderator(s). These moderators will have the ability to manage the permissions of others within the group. Some permissions include allowing users to read, create, or modify data about specific gene tracks in a given hub (assembly of gene tracks), or to change the structure of a hub entirely. Any changes made to a hub should be in real time, meaning that if one user changes something about a track, then all users with the requisite permissions should also be able to see the change shortly afterwards.

2. **BAC-Style Collaboration:** The Role-Based Access Control (RBAC) model is a much more hierarchical and pragmatic method of collaboration in which a lead of project or research has permission over the organization of a project in which they can assign roles which each have their own rules of permissions and authorizations. This allows for secure collaboration when certain levels of control need to be restricted. [5]

3.4.4 Analysis

1. **GitHub-Style collaboration**
   GitHub style collaboration excels at providing a way to create a software that is both easy to use and easy to implement. However, this simplicity comes with a cost of customization of functionality.

   Pros:
   - A strength of this approach is its ease of use. Since Github is the most popular platform for collaborating and sharing code, it will be very likely that a user will be able to adapt to this system quickly. The faster a user learns the collaboration style, the more satisfied they will be with the product and the more work they will get done.
   - The ease of implementation of this style offers a solid opportunity to have a viable product. The interdisciplinary and large scale nature of this project already provides a major challenge to us, so aiming to make a minimum viable product with a simple approach is more than reasonable.

Cons:

- This collaboration style's main weakness is one of its strengths: it's simplicity. This style may be easy to learn, implement, and use, but a user may want a style that's more customizable for a specific task. For example, A researcher at a university may hire several graduate research assistants and undergraduate research assistants to work with them. This researcher may want to allow more permissions to their graduate researchers than their undergraduate researchers. However, this implementation would not be possible with the GitHub-Style collaboration.

2. **RBAC-Style collaboration**

   The Role-Based Access Control (RBAC) model (put more stuff here)

   Pros:

   - The RBAC-style system provides the highest level of functionality for group collaboration efforts. Since it's designed to be customizable, it provides the most diverse collection of group dynamics.
   - This collaboration style is our client's first choice of implementation.

   Cons:

   - The main problem with an RBAC format is its difficult implementation. To create a functional, adaptive, and efficient product, we need to put the highest emphasis on modularity and encapsulation. This yields another obstacle for the complete development of PeakLearner.
   - The complexity, flexibility, customizability of this approach may hinder data scientists and developers from making a quick analysis on their sequences. It may be considered overkill to provide so much functionality, especially if it comes at the cost of analytic results.

3.4.5 Chosen Approach

Using the four desired characteristics mentioned in section 3.4.2, we will provide each user authentication technology reviewed in sections 3.4.3 and 3.4.4 with a score between 1-5 as shown in *figure 4*.

Figure 4.

Name	Simplicity	Usability	Client Preference
GitHub	5	4	3
RBAC	3	5	5

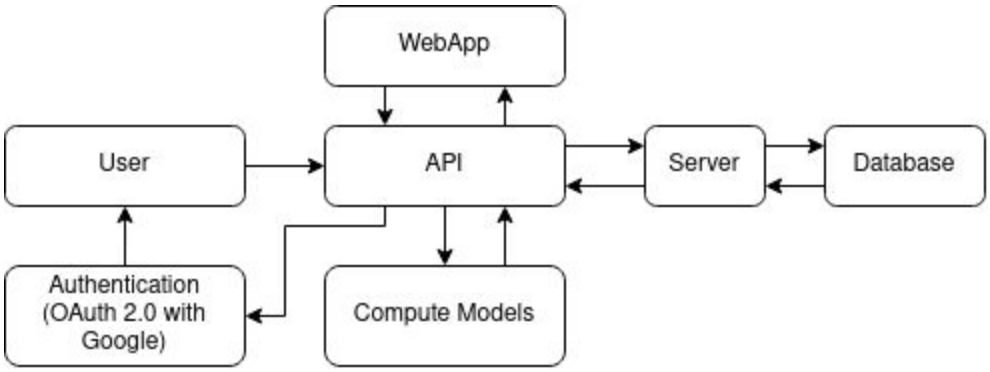
3.4.6 Proving Feasibility

The basis of the collaboration feature will be implemented as a prototype in the Technical Demo. It should allow users to have predefined roles and permissions in a group object that can be created by a user in which the creator can invite other users and assign permissions. Before completion of the demo we will implement Unit Testing of permissions to validate their implementation.

4.0-Technology Integration

TEAM BLUEBOX

Our team has selected solutions to the four major technological challenges of user authentication, database management, user interface, and group collaboration, based on the estimated feasibility in implementing them within the environmental restrictions of the project. Each solution will carefully integrate with one another to create a fully functional system where data is stored, retrieved, and then displayed and interacted with by the user. This is represented visually by the system diagram shown below.



To ensure proper security, a user will have to use authentication before accessing any sensitive information. After this authentication is complete, users can then use this API to access the contents on the webpage. The webpage will display specific data visualizations of the users choice. Since PeakLearner has built-in machine learning algorithms for changepoint detection, this API will also aid the user by providing a simplified way to detect peaks in the sequence. If the user is interested in accessing other datasets, and has the permission to do so, they can use the server to extract data input from other users.

5.0-Conclusion

TEAM BLUEBOX

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 the four major technological challenges of user authentication, database management, user interface, and group collaboration, and integrate their solutions into one functional system.

The application's user authentication, user interface, and group collaboration operations will rely on the ability to effectively manage and store the peak prediction data in a database. With Berkeley DB as our selected database, we have determined that this is more than likely very feasible. Once data management operations are in place, we will solve the user interface challenge by ensuring that users are able to interact with the data by modifying labels and hub configurations. We have determined a secure way to do so, as well as solve the user authentication challenge, by enabling user authentication with the Google API for Google sign in. After user authentication has been implemented, we will solve the challenge of group collaboration by using RBAC style collaboration because it allows the most flexibility.

Following the completion of this Technological Feasibility Document, we will begin work on determining the project's exact functional, environmental, and performance requirements. Once these are established, Team BlueBox will begin the first steps of development.

6.0-References

TEAM BLUEBOX

[1] *Integrating Google Sign-In into your web app. (n.d.). Retrieved from https://developers.google.com/identity/sign-in/web/sign-in*

[2] *Passportjs Authentication Documentation. (n.d.). Retrieved from http://www.passportjs.org/docs/*

[3] *What Is Berkeley DB?. (n.d). Retrieved from https://docs.oracle.com/cd/E17275_01/html/programmer_reference/intro_dbis.html#*

[4] *Mongo DB. (n.d.). Retrieved from https://searchdatamanagement.techtarget.com/definition/MongoDB*

[5] *Role Based Access Control (RBAC). (n.d.). Retrieved from https://searchsecurity.techtarget.com/definition/role-based-access-control-RBAC)*