# Project Proposal

Spotifind

Nathan Wilke
Jordan Hassmann
Sehun Joo
Joshua Mo
Akash Gajendra

# Executive Summary

Currently, all music streaming services provide weak to no integration with social media. Considering this, and how easily people can bond over music, this creates a natural demand for a service that allows music listeners to find other people with similar tastes. Our project, Spotifind, is a web based application that seeks to solve this issue. It will connect to each user's preferred streaming services, and based upon their listening history in those platforms, connect them with other users in Spotifind.

## Design

The design of Spotifind is broken down into 3 main parts:

1. Front-end - consisting of a web application in which users can: create an account, connect to their music streaming services via dedicated API's, and connect with other Spotifind users
2. Back-end - will take data from various API's and utilize robust algorithms to recommend users artists and songs through various metrics like favorite artists or most listened to songs
3. Firebase Database - used to keep track of all user accounts and information.

The combination of clean frontend, robust backend, and organized database should provide for a seamless and engaging user experience while using Spotifind.

## Development Plan

Spotifind will be developed using the Agile methodology in order because it will allow us to have a constant tracker of how much more is to be done, in addition to having working releases at the end of each sprint.

Development Roles:

| | | | |
|---|---|---|---|
| 1. | Jordan Hassmann | - | Front-end Developer |
| 2. | Nathan Wilke | - | Technical Reporter |
| 3. | Joshua Mo | - | Back-end Developer |
| 4. | Sehun Joo | - | Usability Tester |
| 5. | Akash Gajendra | - | Project Manager |

## Goals

| | | | |
|---|---|---|---|
| 1. | Fallback | - | exact match searching, desktop compatibility |
| 2. | General | - | mobile compatibility, "Explore" page, complex search functionality |
| 3. | Stretch | - | in-site messaging service, user settings |

## Problem Statement

Currently, there are many different platforms in which users can stream music based upon search and via recommendation algorithms. Many of these platforms, such as Spotify and Apple Music, additionally attempt to provide a social service inside of their platform, in which users can view their friend's tastes and listening history. However, these attempts feel half-hearted, as they are simply sub features of a streaming service, as opposed to a social media platform based upon sharing music among friends and meeting people with similar musical tastes. Furthermore, with the rise of exclusive streaming services that refuse to link with each other, it leaves people who use certain services isolated from users of other services. A platform that would be able to connect users who listen to music via different streaming platforms together would greatly benefit anyone who wants to meet people with similar musical tastes.

## Considerations

In this project, we are assuming that the APIs provided by platforms like Spotify, Apple Music, etc. will be fairly simple (Not requiring authentication and misc.). Additionally, we are assuming that our chosen database will be fast enough to provide a smooth experience for the user as well as robust enough for the developers to create efficient queries. Moreover, we are assuming that our platform will temporarily only see a small flow of user traffic. One last assumption we are making is that the music streaming platforms we are connecting with will be okay with how we are manipulating their data for our platform. We are unconstrained by a budget, because all chosen technologies will be freely available. Ethically, a concern brought to our attention was age restrictions, and we have chosen to provide a means to restrict specific age ranges from your tailored data set (With an age cap if the user is a young child). In addition, we are expecting that last.fm will allow us to pull as many album covers as needed for the project, and will not block us from querying for albums/artist names.

## Personas and User Stories

**Create at least five user stories** of how you envision the product being used by your personas.

1. As a Group Creator, I want to create groups of my friends so that we can have a space to connect more and discuss our profiles. A Group Creator is a subset of the User, but also acts as the "Host" of the created group.
2. As an Admin, I want to manage site permissions and site states so that I can provide support when needed. An Admin role is basically meant for developers to perform real-time site updates.
3. As a User, I want to connect with others based on similar music based interests so that I can connect with new people and have my preferences in a single place. We assume that the user does have a need to connect multiple platforms as well as currently utilizes multiple platforms for music streaming. We also assume they have/want friends. A user might need a darker interface in case they're sensitive to light.
4. As a user, I want to be able to link different music platforms to Spotifind in a simple way so I can easily connect all my music streaming services.
5. As a Group Creator, I want to be able to invite other people to the group so that I can expand the group.

# Proposed Work

## Evaluation of Possible Solutions

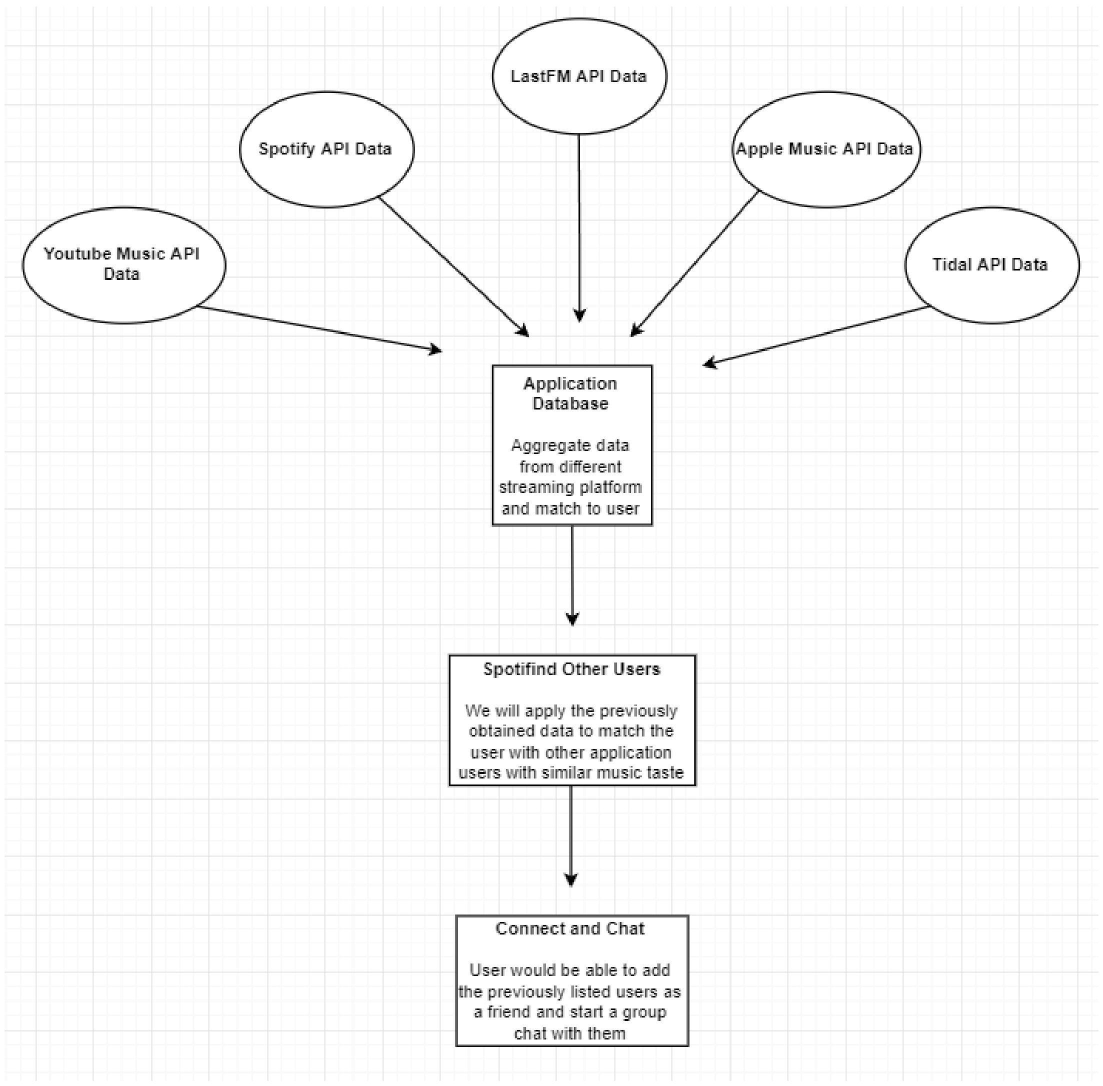
One other possible solution is providing users the ability to modify their data from each linked, respective platform through our platform. On one hand, this method would provide our users a much more interactive and versatile platform and experience, but on the other this would require much more secure authentication if we allow access to various other sites. Additionally, there would have to be a synchronization factor that we would be required to maintain between each individual linked site, which is extremely out of scope for this project.

A second solution is making the site *song based* where we run all our data and recommendations on the songs our users listen to. On one hand, this provides another medium for our users to connect based on as well as allows us to tailor their preferences and recommendations better. On the other hand, our initial technology might not be able to handle the much larger set of data in a timely manner as compared to using music artists. Additionally, there would theoretically be a large overlap in what songs people listen to between different platforms. I.e., if you listen to Song A & Song B on Platform A, then the odds are you also listen to Song A & Song B on Platform B as well. This overlap does not provide us as much insight as cross-platform music artists would.

Back to our solution, our site solves all the aforementioned issues and more. We provide an intuitive design while maintaining responsive speed. Your data is also very versatile, where we can reference the same form of data throughout various locations of the site (By leveraging Redux). Additionally, we ensure safety of our user data across *all* platforms because we choose to not expose more permissions than we are capable of protecting.

## Solution Design

### System Diagram


LastFM API Data
Spotify API Data
Apple Music API Data
Youtube Music API Data
Tidal API Data
Application Database
Aggregate data from different streaming platform and match to user
Spotifind Other Users
We will apply the previously obtained data to match the user with other application users with similar music taste
Connect and Chat
User would be able to add the previously listed users as a friend and start a group chat with them

In this section, we will be walking through the cycles of interaction users will be exposed to while engaging with our application. Upon successful signup, our backend will aggregate all the data received from the various APIs and integrate it into our application database. After which, our application will recommend users with similar music preferences. Additionally, any user will be able to connect with the recommended users and engage with them socially through our platform.

The user's music data will be taken from Spotify and Apple music, possibly from Youtube and Tidal if their unofficial API works properly. In addition, the album covers and artist profiles will be taken from LastFM, which provides that information in their API.

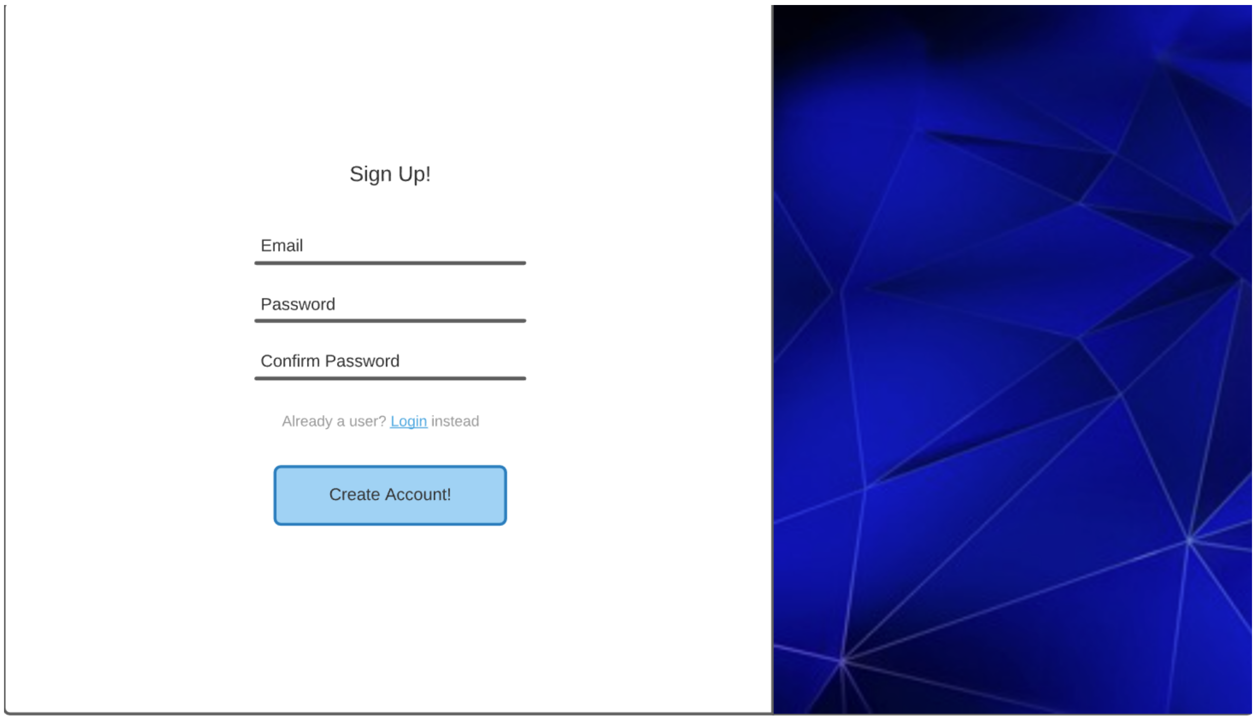
The application database will contain data for each user on their album and artist listen data, as well as miscellaneous data such as their oAuth token, join date, name, and birthday. We make this database using Google's firebase. It will also contain a list of that user's friends.

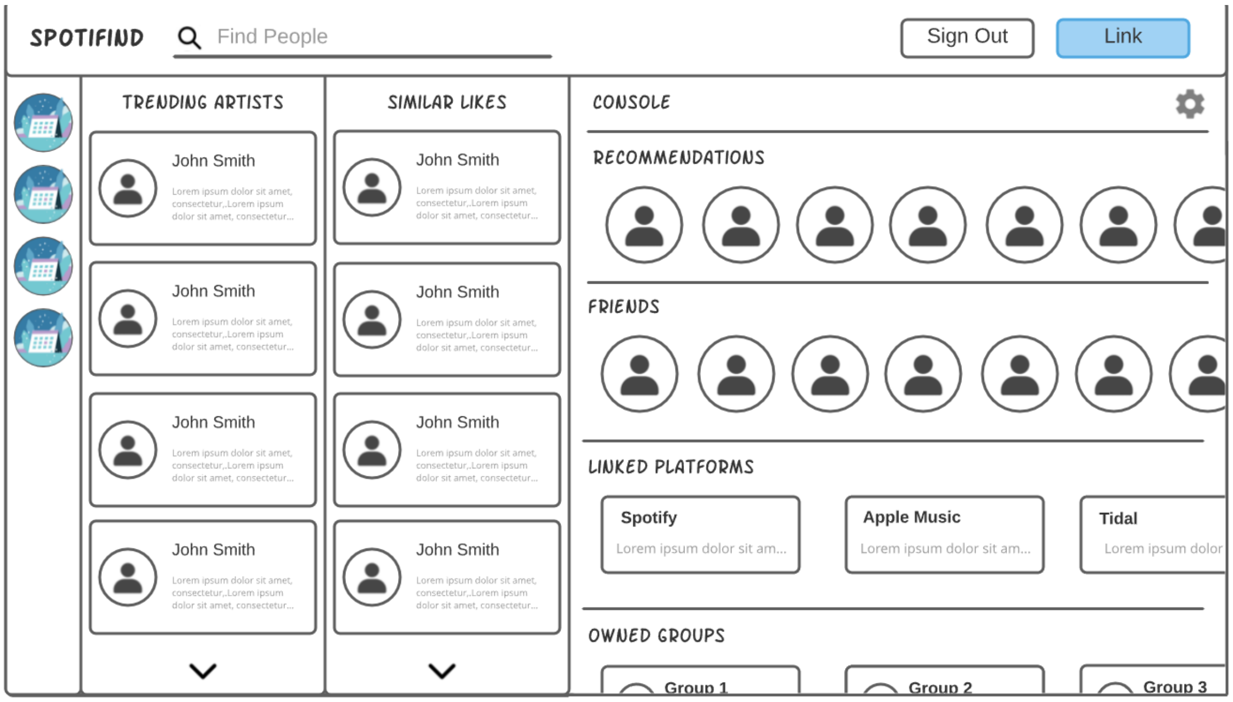
There will then be a backend written in either Javascript or Python which will connect to the database and dig through to find ways to connect users with each other, whether that be through randomization, mutual friends, location, mutual favorite artists, or similarity in names.

The frontend will have a way to add other users as friends either by recommendation or by search. In addition, we will allow some level of group chat or 1 on 1 chatting in the chat.

## Interface Diagram

One of our key considerations with this project has consistently been to create a satisfying user experience for our end product users by incorporating accessibility features, adapting pleasant color schemes and enabling easy navigation to mention a few.  We will be expanding on these following the designed digital wireframes.





Starting with the login page, we have prioritized making the UI easy to comprehend and hide any underlying complexity from the end users. Once the user is logged in, we display a whole host of support information ranging from trending artists to user-specific recommendations. Additionally, we also support the user by showing their groups, buttons meant for linking music platforms and SpotiFind signout. All of this information despite the size has been laid out in a minimalistic sleek fashion.

As mentioned earlier, one of our priorities in designing the UI has been accessibility. Our team will be making the application keyboard friendly i.e all buttons and fields will be accessible via the keyboard and not require a mouse. We will also be implementing a color contrast in the form of a light/dark mode for users sensitive to light or dark color schemes.

## Planned APIs

Over the course of our project, we intend to use and efficiently integrate a whole host of APIs into our final application. The web application our team has set out to build at its heart intends to match individuals with similar music tastes based on their usage patterns in external applications like Spotify, Youtube Music to mention a few.

Our preliminary designs for the project involve directly utilizing four APIs associated with Spotify, Apple Music, Tidal and Youtube Music. One of the problems we hope to address is that there does not appear to be a well-designed application that draws recommendations based on music taste across different platforms. We aim to use these to retrieve information on a user's music preferences despite their preferred music streaming service and make that the basis for connecting different users together.

Additionally, we will be using the lastFM API to gather artist specific details. This would serve as key to enhancing the user experience as we desire to display artist specific information across the various pages that any given user might navigate through.

## Project Management

### Team Roles and Qualifications

Jordan Hassmann: This member will be responsible primarily for the **front-end** development of UI elements and Redux stores.

Nathan Wilke: This member will take the role of **technical reporting**. He will be responsible for ensuring the integrity of technical documentation being submitted.

Joshua Mo: This member will take the role of **back-end** developer by connecting the Firebase database to the UI and providing helpful microservices where needed for additional UI functionality

Sehun Joo: This member will take the role of **usability testing** and will be responsible for testing features and finding bugs and fixing them

Akash Gajendra: This member will take the role of **project manager** and will be responsible for regular maintenance of the backlogs and burn-down charts.

### Development Methodology

Our team will be using the Agile development method to complete the project. In order to maintain a burndown chart, necessary tasks, and current development state, we will be using Jira software because it is specifically tailored for the Agile workflow. We plan to have regular backlog grooming meetings at the beginning of each week to discuss task assignments, story points, and overall sprint plans. Additionally, a daily standup will be scheduled for Wednesday and Saturday to check in and ensure we are all on the same page.

| Day | Time | Topic |
|---|---|---|
| Monday | 8:00am | Lab |
| Monday | 6:00pm | Backlog Grooming |
| Tuesday | 9:35am | Class |
| Wednesday | 8:00am | Lab |
| Wednesday | 6:00pm | Standup |
| Thursday | 9:35am | Class |
| Saturday | 6:00pm | Standup |

## Planned Scope

**Fallback Goals**
Ideally, we would like to provide the ability for our site to work well on mobile devices as well. However, if we do not have the time or resources to implement this functionality, we will just provide a page that will cover the site and disable all functionality if the device is under a certain size. Additionally, if an API proves to be too complex to work with, we will fall back on a smaller, easier to use API that provides a little less insight and functionality. Lastly, if we do not have the time for an "Explore" page or more complex search functionality, we will fall back on exact match, non-real time searches.

**Stretch Goals**
If presented with enough spare time, we would like to implement in-site messaging services so you can communicate with the other Viewers through our services. This however requires a larger understanding of Firebase as well as a decently large amount of time. Additionally, we would like to provide a means to alter your settings as a user. However, linking settings for given users can be very complex in terms of loading specific settings (Also site loading slowdown) as well as the database linkages.

## Task Breakdown and Scheduling

Our set of tasks revolve around the creation of a database to store user data in, the setup of React and associated Redux Stores, the configuration of our chosen APIs, design and implementation of various UI elements, and providing CRUD operations throughout the site.

Getting the user recommendation algorithms set up in the backend is dependent upon the completion of accessing the API's and getting the user database set up first.

The first three independent tasks that should be completed are the creation of the Firebase Database, figuring out how to connect to streaming API's for a given user, and creating a login screen. After these are done, these can be combined to create a login page in which accounts can be made and automatically connected to the database, and can also be connected to the streaming API's.

After this, the design of a backend that can recommend users based upon their listening history from the API must be created. At the same time, the UI of the main page can be generated. This UI will be broken into many subtasks that can be completed concurrently, with many of these being based around new ways of recommending new users or songs. After these are done, the entire main page should be functional. Ideally, this will be done by the end of the first sprint so we have a working release that we can get some user feedback from.

The rest of the development time should be spent on improving the product. This can be done using the user feedback, implementing our stretch features, and improving our previously made sections. This section should be the primary focus of sprint 2.

1. Create login/ signup page
    a. Create Firebase database
    b. Connect to API's given a user account
    c. Create login page UI
2. Create main page
    a. Create backend recommendation algorithm
    b. Create main page UI
3. Improving product
    a. Adding stretch features such as messaging, groups, ect.
    b. Improving efficiency of algorithms and connections

## Appendix 1: Product Backlog

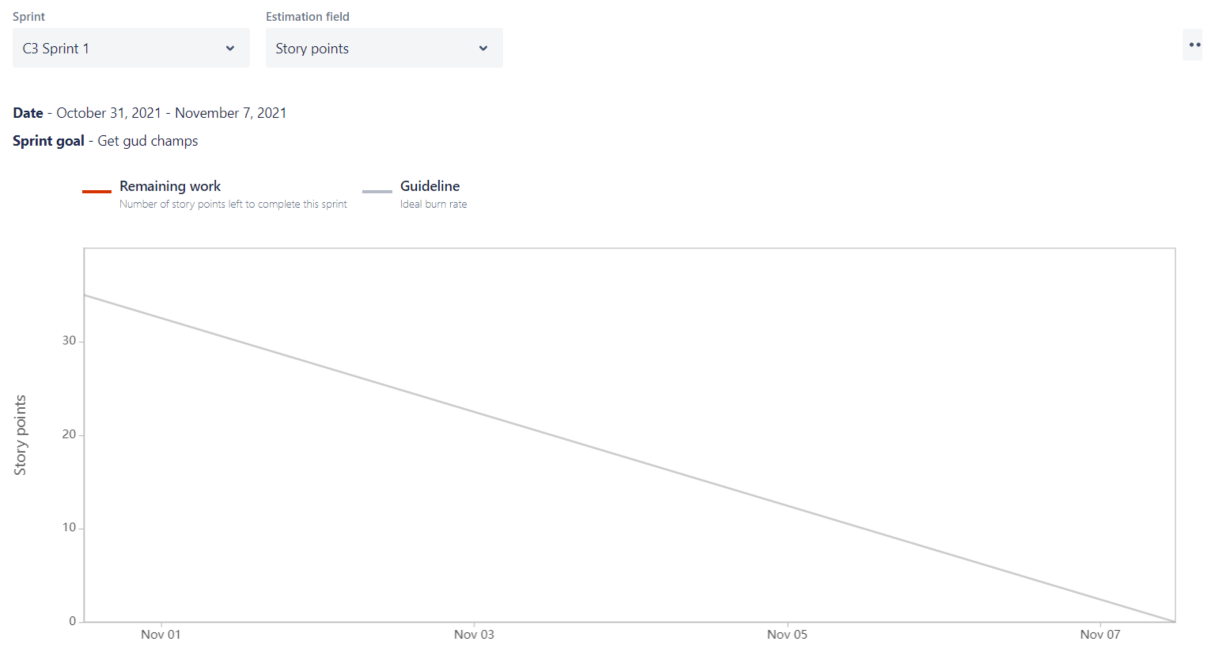
This link directs you to a public Google Sheets document containing all the tasks in the Product Backlog.

https://docs.google.com/spreadsheets/d/1mG8sHMU6gAmFNEbb-VBhbuv1kddLfz-fuZFbmytFxVM/edit?usp=sharing

## Appendix 2: Initial Product Burn-down Chart

Here is the initial Sprint 1 burndown chart (0 Complete), please see link below to go to the Jira site and look more in depth or let us know if you have any questions.
**Note:** For the purpose of task management, we are utilizing JIRA that has been designed for Agile development. Each story point on the burndown chart and spring catalog is associated with 30 minutes based on our estimations.

**Remaining Hours:** 17.5 hours (35 story points)



Sprint: C3 Sprint 1
Estimation field: Story points

**Date** - October 31, 2021 - November 7, 2021
**Sprint goal** - Get gud champs

## Appendix 3: Initial Sprint Backlog

To see all subtasks as well (Tasks), go to this link (https://jhassmann.atlassian.net/jira/software/projects/C3/boards/1/roadmap?shared=&atlOrigin=eyJpIjoiZDM1ZTQ2NzJiNDVmNDFjMjkyZDkwNzIxOWY1MjkwZDIiLCJwIjoiaiJ9) where I sent email invitations to view to Rob, Tommy, Dr. Thomas so they can see.

**Initial Time Estimate:** 17.5 hours
**Actual Time Estimate:** 0 hours (Sprint not yet begun)

https://docs.google.com/spreadsheets/d/19rKHhhlKfWxAKcr3ofRO7jG4NHQ8-ddcEIN6YSsluS8/edit?usp=sharing