**Music Matters Booking System**

**Cycle 1 Report**

Authors :

Chase Dumbacher

Sarah Pham

Austin Mongold

Austin Newkirk

Submitted in partial fulfillment of the requirements for COMP 4710 Senior Design to the Department of Computer Science and Software Engineering,Samuel Ginn College of Engineering, Auburn University

Auburn, Alabama

January 31,2021

**Table Of Contents**

*Completed By Chase Dumbacher*

[**Executive Summary (System Metaphor)**](#_4ic6os66jwrd) **3**

[**Project Introduction**](#_3wqw299o9uap) **3**

[Previous Development](#_vxu6jxgu6x1o) 3

[Cycle Intent](#_6il1a2dpc2p8) 3

[Future Work](#_va8gu6dfjlu1) 4

[**User Stories**](#_for9ln64fsyw) **5**

[Website User Stories](#_pk14f0smkgqe) 5

[Firebase User Stories](#_rs6noinna1c2) 9

[iOS App User Stories](#_c3ztbwczfh74) 10

[**Design Documentation**](#_lrimcv6zen0p) **13**

[High Level Architecture](#_e8h1wxm4djsz) 13

[iOS App Low Level Architecture](#_ezywznk36jlu) 14

[Website Low Level Architecture](#_g8f9ddvx6k8c) 16

[Website System Sequence Diagrams](#_iqnn8hblqyr2) 17

[Interface And UX Design](#_l24sxpku06v4) 18

[**Management Plan**](#_ubs9z3j8kzu4) **21**

[Task Assignments](#_t41zodw4wdxx) 21

[Development Schedule](#_6bfp3jmcr6yi) 24

[Planned Code / Feature Freeze](#_ui8l78nh861p) 25

[**Risk Mitigation**](#_fmxhf8wll1r0) **25**

[**Test Plan and Test Procedures**](#_b752929c2uhd) **27**

[**Lessons Learned**](#_ya2m2ph29fuz) **28**

[**Appendix A**](#_a9gdwuj02zdn) **28**

[Meeting Minutes](#_h3du364tvf5w) 29

[Correspondence/Digital Signature](#_634ds6vkzj1r) 30

[Correspondence Between Team And Customer](#_hzecfgx1c3uk) 31

[Correspondence Between Team And Instructor](#_twtore66pfj0) 33

[Correspondence Between Team And Team](#_v52th97r5bmw) 34

[Correspondence Between Team And Others](#_1rez8etvxhy4) 41

[Source Code](#_t8oido6flbsq) 42

# Executive Summary (System Metaphor)

*Completed By Chase Dumbacher*

To design and create an app and online calendar booking system that is capable of recording and organizing artist names, performance time and dates, and venue names as well as automatically creating and emailing invoices and confirmation documents.

# Project Introduction

## Previous Development

*Completed By Austin Mongold*

So far the mobile app team has fixed several bugs with the prior team’s code for the app, verified and tested the functionality of the PDF builder for the app, began work on the emailer functionality of documents, improved UI to be clearer and more bold, and added several quality of life changes to the app. The web team has created the basic framework for the website, added firebase authentication for log-in, gotten firebase data to display on most of the pages, added some basic calendars for future use, and worked to make the website look like Mike wants it to look.

## 

## Cycle Intent

*Completed By Austin Mongold*

Our team’s intent for this spike in regards to the mobile app is to get the app running more smoothly than it was when the last group was working on it, continue to verify previously implemented functionality, continue work on an email generator, and make the app more intuitive. Our team’s intent for the web app is to create and continue to improve the website framework, implement firebase login for authentication, have the website display firebase data in their respective locations, and have the calendar display relevant data on the pages where they are located.

## Future Work

*Completed By Austin Mongold*

As the project continues, the mobile app team plans to finish the full implementation of the emailer on the mobile app, verify and improve the PDF builder, and fix the errors that were left over by the previous group in the mobile app. The web team plans to finish the calendar implementation with the database, improve data readability, add the functionality to add, edit, or delete data from the firebase database in a manner that is consistent with the mobile app, and also have a functional pdf builder and emailer like the mobile app has.

# 

# User Stories

## Website User Stories

*Completed By Austin Newkirk*

Website Dummy

*Completed By Austin Newkirk*

**User Story :** As users, we want to have a basic website setup and accessible.

**Description :** A website will be accessible through a users local computer and will

be used as an example and starting point for the envisioned finished website.

**Task :** Create a simple running website using HTML, Javascript, and any other technologies.

**Planned Hours :** 4

**Actual Hours :** 8

**Coder(s) :** Austin Newkirk, Austin Mongold

**Tester(s) :** Austin Newkirk, Austin Mongold

**Status :** Completed

Cross-Platform Integration (Website and iOS App)

*Completed By Austin Newkirk, Chase Dumbacher*

**User Story :** As users, we want to see the system be usable from both iOS and PC devices.

**Description :** If a user wishes to view the system calendar on their PC they can open a website or application, but if the user wishes to view the system calendar they can open it on their iOS device.

**Task :** Create a web based interface that is capable of taking in the Firebase data and displaying it in a simple non-complete fashion.

**Planned Hours : 8**

**Actual Hours :** TBD (because of the nature of firebase, this user story turned out to be a non-issue)

**Coder(s) :** Austin Newkirk and Austin Mongold on the website, Chase Dumbacher and Sarah Pham on the iOS app

**Tester(s) :** All members

**Status :** Active development

Ease Of View

*Completed By Austin Newkirk*

**User Story :** As users, we want to see the calendar displayed by the system look similar to the vertex42 or google calendars displays.

**Description :** The calendar for the booking system on the PC application will be big enough to read clearly and will display artist names and performance times without having to click to expand.

Upon further research, developing a whole new calendar system would have been a very daunting task. However, we found out there is an open-source jQuery script that meets the criteria for what Mr. Moody wants, and more. We decided to implement this free, open-source software. [Link](https://fullcalendar.io/).

The aesthetics/look of the calendar will look nearly the same to google calendars. There might be minor changes like sizes, colors, etc., but the look as a whole has been finalized.

**Planned Hours :** 10

**Actual Hours :** 3 (so far)

**Coder(s) :** FullCalendar for Calendar creation, Austin Newkirk for calendar implementation, Austin Mongoldfor data entries.

**Tester(s) :** Austin Newkirk, Austin Mongold

**Status :** Near completion

Fast and Clean User Experience

*Completed By Austin Newkirk*

**User Story :** As users, we want to see the applications be as snappy, intuitive, smooth and responsive as possible.

**Description :** Work on things that make the apps feel intuitive and professional as well as making it all feel quick and responsive.

**Task :** Get the aforementioned areas of the app to perform as quick and

responsive as possible, but also make them look appealing and user

friendly.

Due to the nature of this user story, no code was developed specifically for speeding up the website, as the website's speed is determined by the internet speed of the user as well as the speed of firebase itself. There may be optimizations in code in the future, but the biggest contribution to swiftness is outside our control.

**Planned Hours :** 4

**Actual Hours :** 0 (so far)

**Coder(s) :** Austin Newkirk, Austin Mongold

**Tester(s) :** Austin Newkirk, Austin Mongold

**Status :** Active development (will remain active until project completion)

Displaying real information on the calendar

*Completed By Austin Newkirk*

**User Story :** As users, we want to see the calendar within the website to display useful information, including times of shows, who's playing, football games, and holidays.

**Description :** Using the information from firebase and integrating it with FullCalendar properly, displaying the same (or perhaps more detailed) information compared to the iOS app.

**Task :** Code the things needed to properly show the data from firebase (and google calendar for holidays) on the FullCalendar seamlessly.

**Planned Hours :** 12

**Actual Hours :** 2 (so far)

**Coder(s) :** Austin Newkirk, Austin Mongold

**Tester(s) :** Austin Newkirk, Austin Mongold

**Status :** Early active development and research

E-mail functionality

*Completed By Austin Newkirk*

**User Story :** As users, we want to be able to send the correct emails/forms to the correct people either automatically or with the push of a single button.

**Description :** Using the information from firebase, email the correct forms to performers about performances (or any other situation Mr. Moody might want to send an email regarding his company) automatically, or with a push of a button.

**Task :** Using an official MusicMatters email address and using the data from firebase, send an email to all of the performers that month with the proper forms they need to fill out. The email should likely need to be repliable.

The calendar display and the email functionality is ultimately what Mr. Moody needs, so these two User Stories will be the biggest focus during development.

**Planned Hours :** 15

**Actual Hours :** 0 (so far)

**Coder(s) :** N/A

**Tester(s) :** N/A

**Status :** Research, not started

## Firebase User Stories

*Completed By Chase Dumbacher and Sarah Pham*

Database Data Sync-ability

**User Story :** As users, we want to see a change made in the iOS app change data

Live on the Firebase database.

**Description :** When an artist or venue or any form of such data is changed, said data will be sent to the Firebase data and change the entry there.

**Task :** Connect or assure that the program connects the data run by the app

Is connected to the firebase database.

**Planned Hours :** 1

**Actual Hours :**

**Coder(s) :** N/A

**Tester(s) :** Chase Dumbacher and Sarah Pham

**Status :** Completed

Database Security

**User Story :** As a user, we want the database to be “read and write” safe.

**Description :** The database data will only be readable and writable if the user has the specific roles required

**Task :** Change the rules in the firebase database to require specific roles for

Read and write access

**Summary :** Specific user groups can be made to allow for admin level access for

Actual editing of calendar events and creating documents vs just being able to

View the calendar.

**Planned Hours : 5**

**Actual Hours :** 1

**Coder(s) :** Chase Dumbacher

**Tester(s) :** Chase Dumbacher, Sarah Pham, Mike Moody

**Status :** Early work in progress

## 

## iOS App User Stories

*Completed By Sarah Pham*

Error Checking/Recovery

**User Story :** As a user, we want the homepage bug to be fixed.

**Description :** After the second login, the program skips the homepage and goes straight to the list of venues, which should only be accessed via the “venues” button.

This bug could not be replicated on an iOS simulator, which was what we were using to test the live changes to the app. After messing around with the app stack and changing the initial views after the login, we were able to familiarize ourselves with the Appstore and upload new versions of the app. The original testflight given to us was built 2-3 months prior. It was discovered that the app stack and initial views were not the issue, but rather updating the dependencies and build files. A new testflight was created, and the bug was fixed and tested on an actual iOS device.

**Task :** Figure out why the homepage is being skipped.

**Planned Hours :** 5

**Actual Hours :** 1

**Coder(s) :** Chase Dumbacher and Sarah Pham

**Tester(s) :** Chase Dumbacher, Sarah Pham, and Mike Moody

**Status :** Complete

Functional Scrapping Shown on Calendar

**User Story :** As a user, we want to be able to view when the Auburn football games occur on the calendar.

**Description :** The calendar should show all events of Auburn University's football games.

The previous team was not able to get the football games to appear on the calendar. They seemed to have scrapped ESPN’s website on Auburn University’s football games by using a certain library and implementing it within the code. This function uses the Google Firebase cloud functions that we recently were able to get working. Now we just need to deploy this function alongside the emailer.

**Task :** Implemented the football games API within the calendar.

**Planned Hours : 4**

**Actual Hours : 4+**

**Coder(s) :** Chase Dumbacher and Sarah Pham

**Tester(s) :** Chase Dumbacher, Sarah Pham, and Mike Moody

**Status :** In progress

Functional Email System

**User Story :** As a user, we want to be able to choose client emails sent to the Music Matters Bookings email when a venue has been booked.

**Description :** The program should be able to automatically send an email to the Music Matters Bookings email with no issues.

This function uses Google Firebase cloud functions in order to call functions in a Firebase app. We initially ran into the issue of our Firebase billing account, we had to get Mike to re-verify the account (which took approx. a day for Google to approve of). Then we ran into the issue of some node modules or dependencies that were missing. After resolving that issue it was discovered that our deprecated Node version was impacting the deployment of our cloud functions. After updating Node and upgrading our Firebase account to the Blaze plan, we started running into issues with authentication (found by reading the Firebase logs within the project). We reauthorized Firebased on terminal, checked all client id’s, secrets, and API keys. We also checked any hidden config accounts and made sure the username and passwords were the correct. We are still receiving an authenticated response when deploying these cloud functions.

**Task :** Fix the emailer function

**Planned Hours :** 5

**Actual Hours :** 10+

**Coder(s) :** Chase Dumbacher and Sarah Pham

**Tester(s) :** Chase Dumbacher, Sarah Pham, and Mike Moody

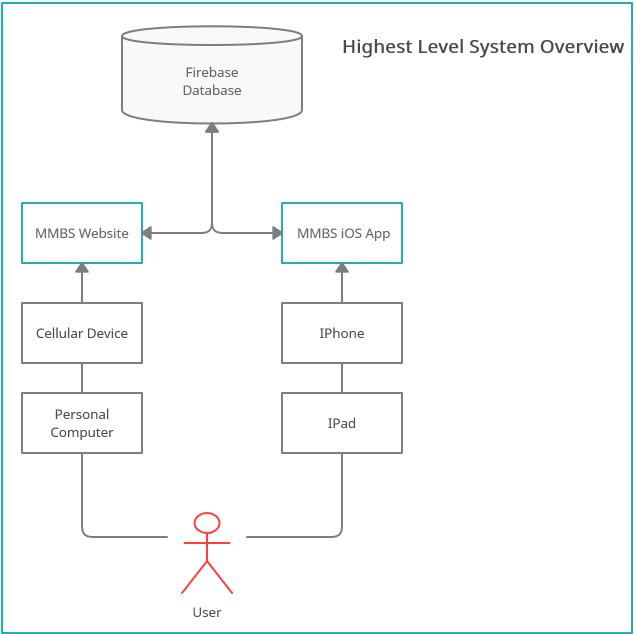
**Status :** In progress

# 

# Design Documentation

## High Level Architecture

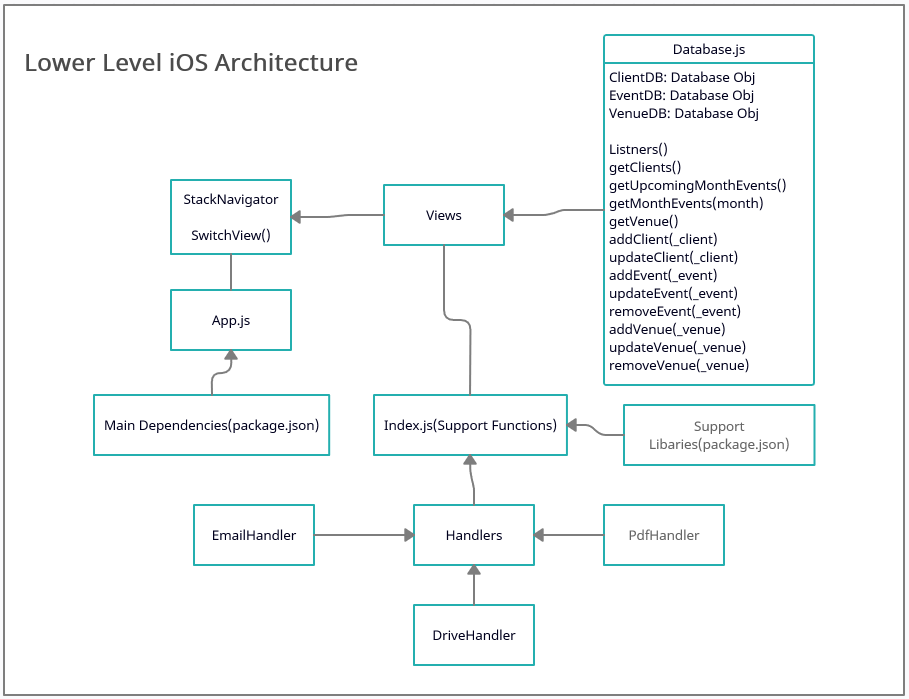
*Completed By Chase Dumbacher*



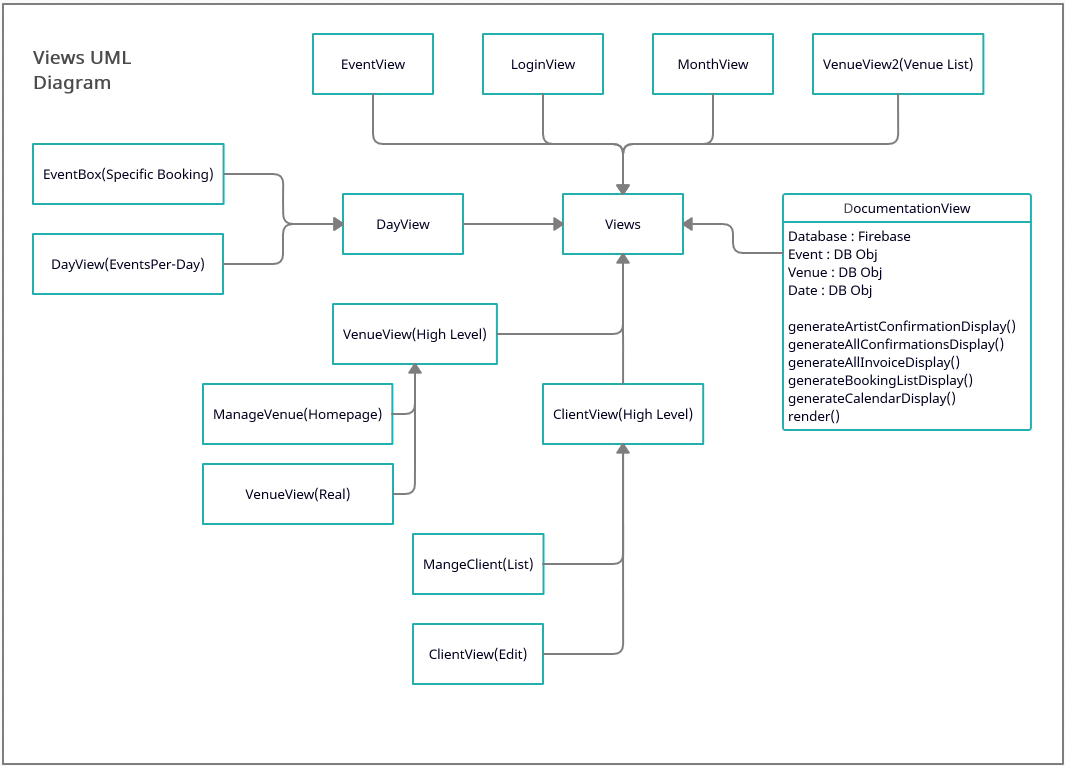
This figure describes the high level system interactions. A user is able to either use any device that can pull up a webpage to connect to the website or they can use an iOS device to pull up the iOS app. The respective application then communicates with the firebase database to pull needed data and record any changes to the data. These changes then trigger a change in the database which triggers both applications to pull the new changes from the database.

## iOS App Low Level Architecture

*Completed By Chase Dumbacher*



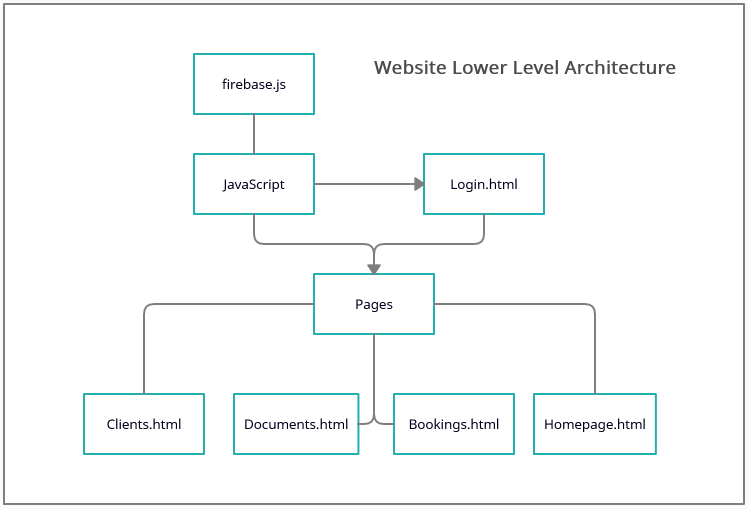
The iOS applications two main classes are the App.js and the folder of views that contains a number of views.js. The App.js functions as the entrypoint of the application and sets up the stack navigator. The navigator then jumps from page to page where each page is a specific view. Different functionalities of each page are accomplished through the support class of functions organized in the index.js file.



This diagram visualizes the specific files and structures that makeup the various views of the iOS app. Each page falls under a specific view class. Some files have multiple view classes in a single file. Most do not contain functions outside the standard render(), however some such as DocumentationView have specific functions coded into it that allow the correct type of DocumentationView page to be displayed.

## Website Low Level Architecture

*Completed By Chase Dumbacher*

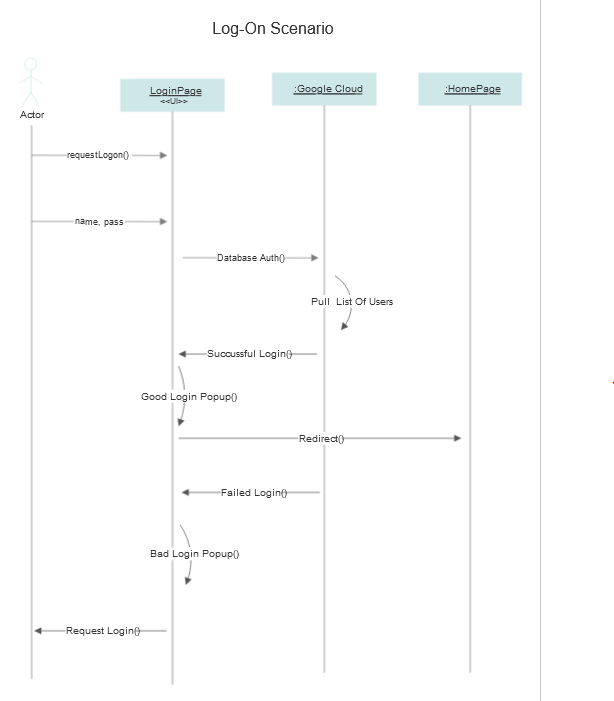


The Website uses the Login.html as the root page. This page uses functionality from the firebase.js to handle the user auth as well as database calls. All other pages currently also use this to reference database data such as venues and clients. Future javascript files will be made and be used to accomplish functionalities such as the document generator and the email sender.

## Website System Sequence Diagrams

*Completed By Chase Dumbacher*

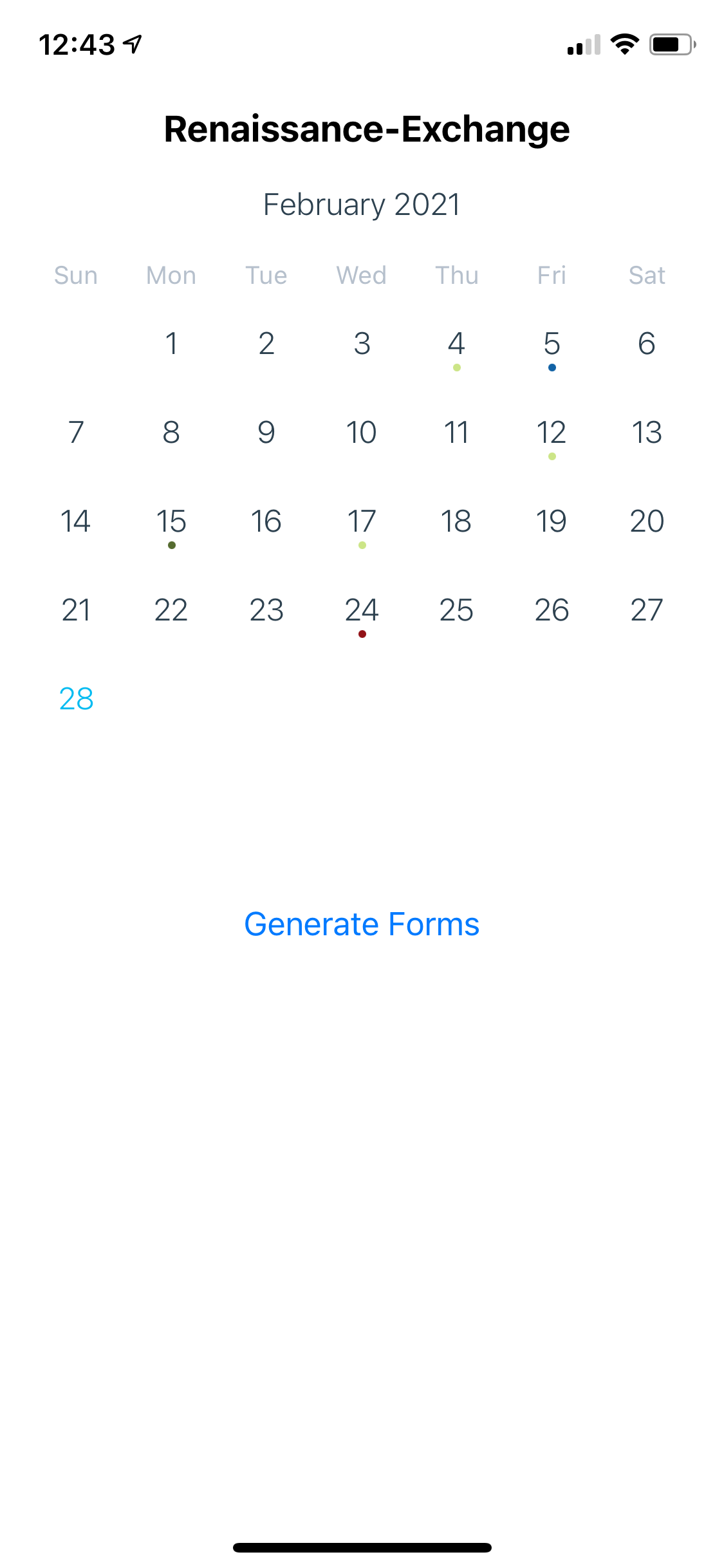
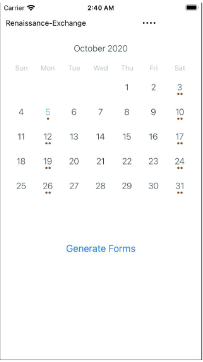
**Log-on Scenario**

****

## Interface And UX Design

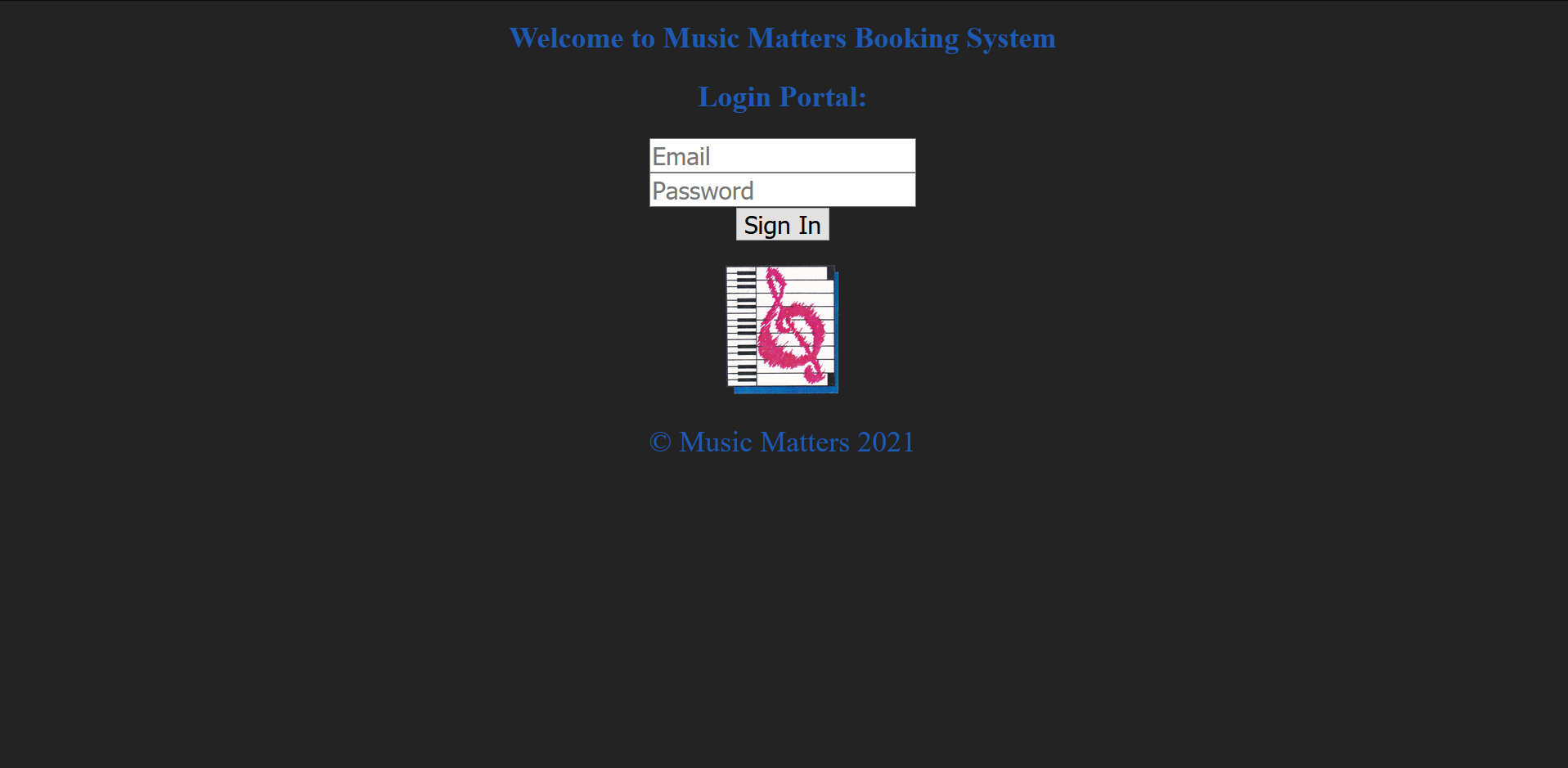
*Completed By Chase Dumbacher*

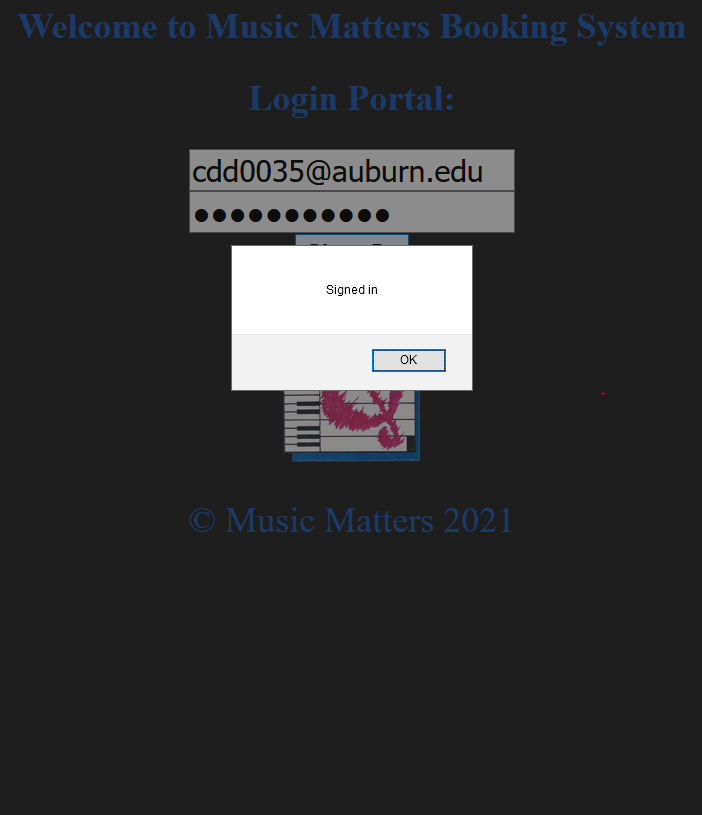
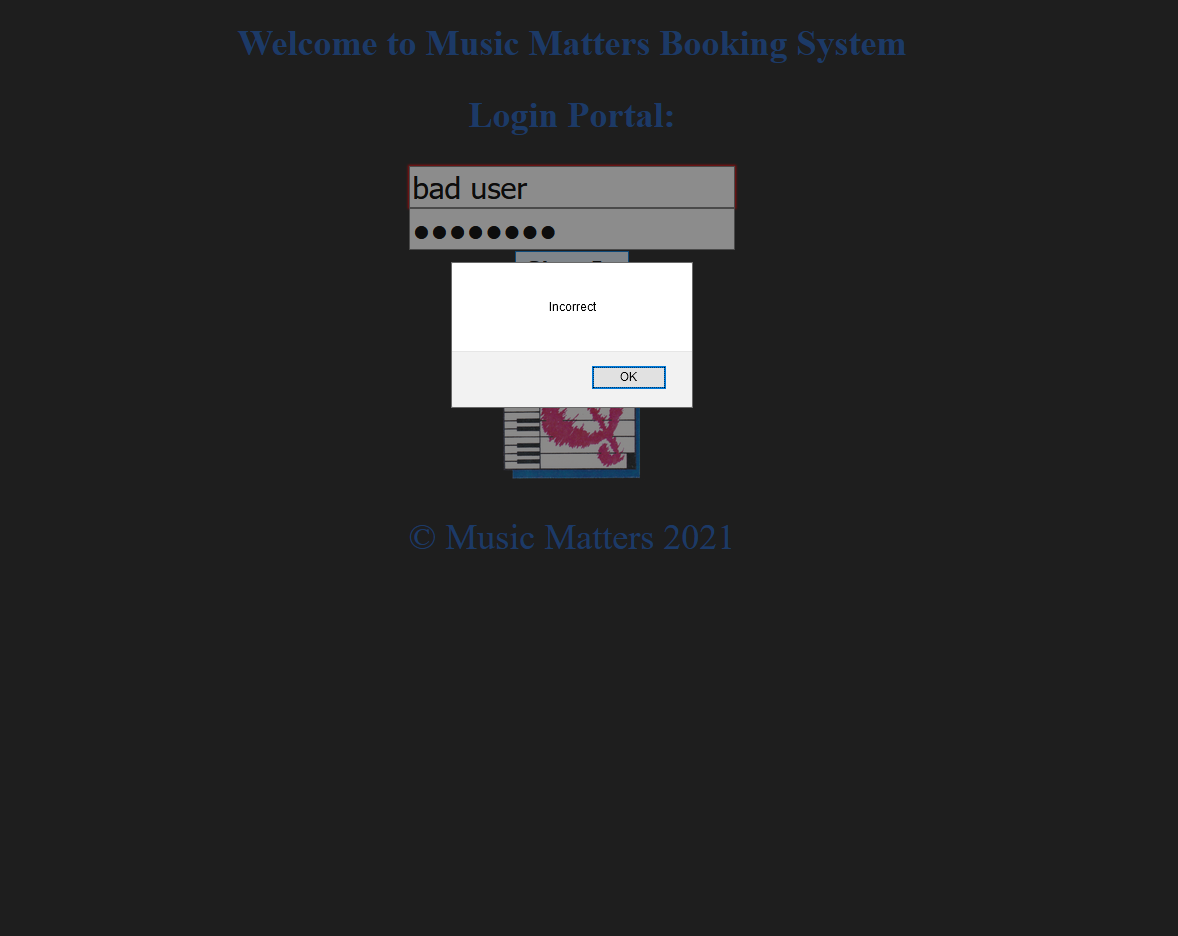
The majority of the iOS app has already been designed and approved by Mike. We have been adjusting minor details such as center text and making certain titles larger, for this reason the majority of the focus is on the website design. The focus of the website user interface design is to assure that all text has good contrast with the background so to allow easy readability and to design the calendar in such a way that all information is readily viewable and large enough for easy viewing. We are also developing the website with simplicity in mind. Currently the design has a bar on the left hand side that has a client, venue, documentation, and bookings. The main space of the website will constantly show the calendar and will change according to changes to the venue.



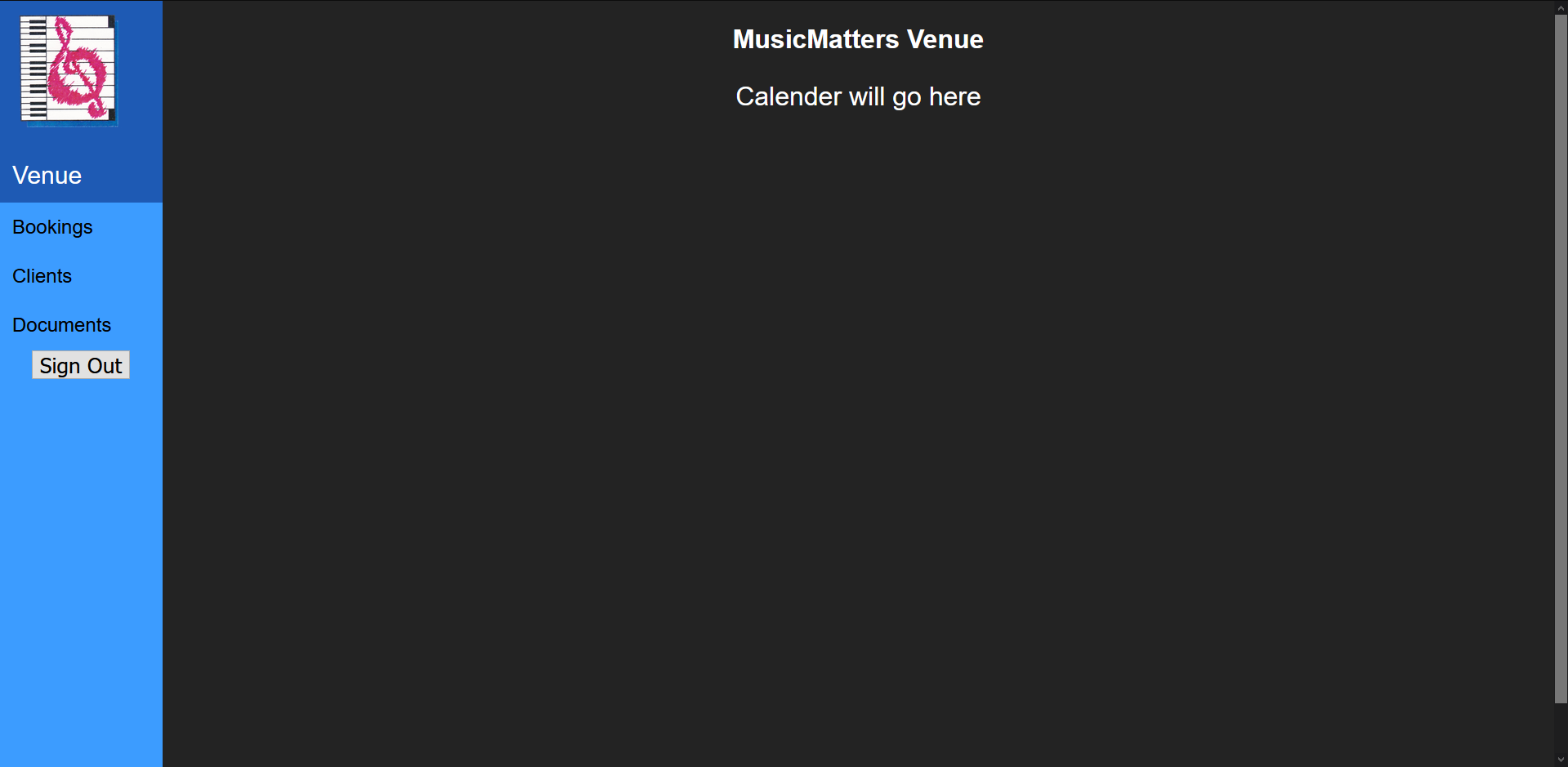
**→**

This image above shows one of the few UI changes that was made to the iOS app. The venue title was changed from left alignment to centered and the font was bolded and increased. We also took away the hamburger button shown in the top right due to the fact that its purpose was to go back to the last page, but this is already possible by swiping left to right.

****

This is the first page that users will interact with when trying to use the website version of the system. From here the user will log in and then be notified of successful or unsuccessful login.

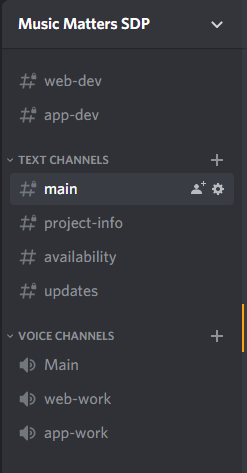
After a successful login the user is met with the homepage that is used to pull up different venue calendars and select whether or not they wish to send out booking documents.



The bars on the left are all currently selectable and will display a different page, however the contents of the page, in particular the calendar, is still being worked on at this current moment.

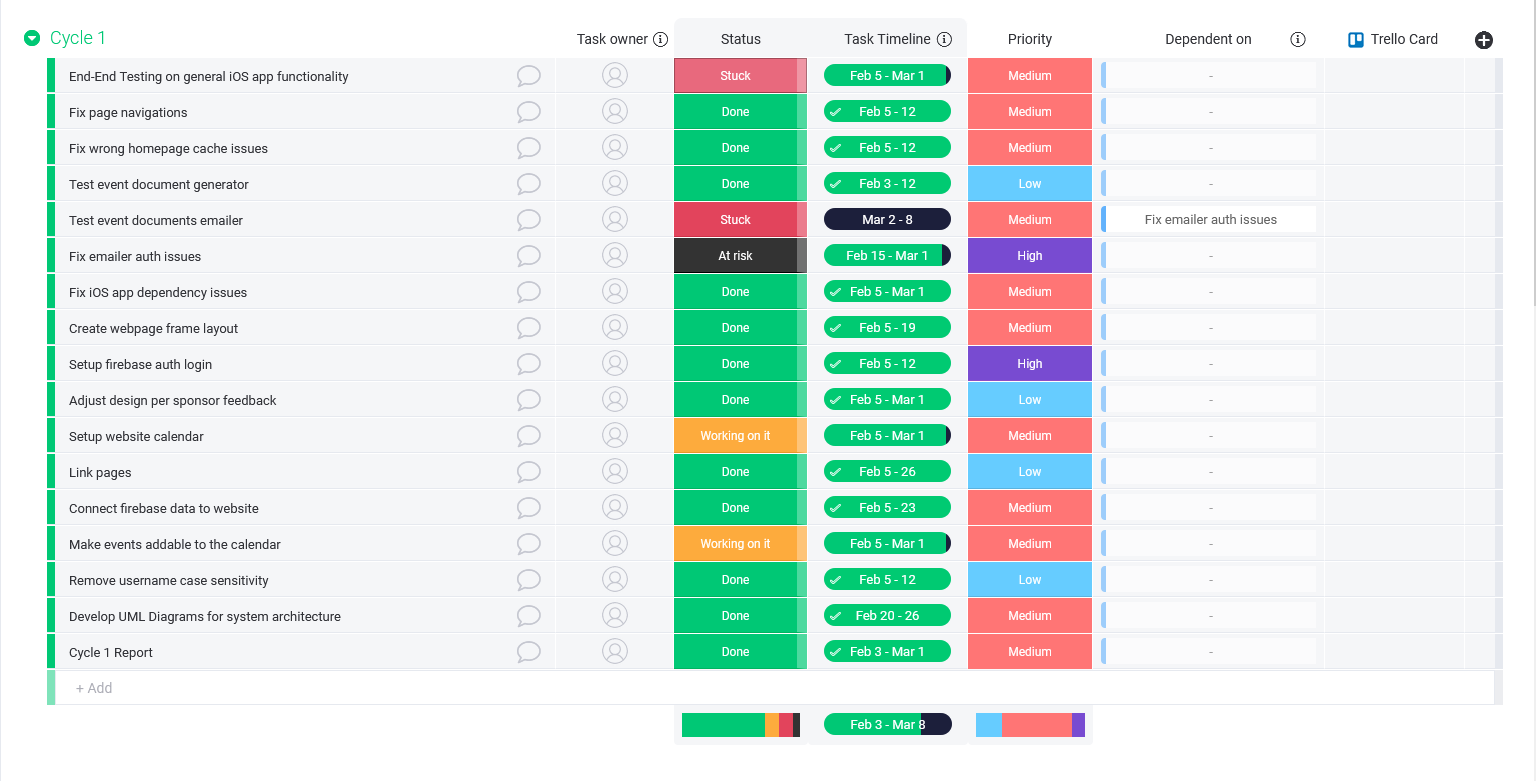
# Management Plan

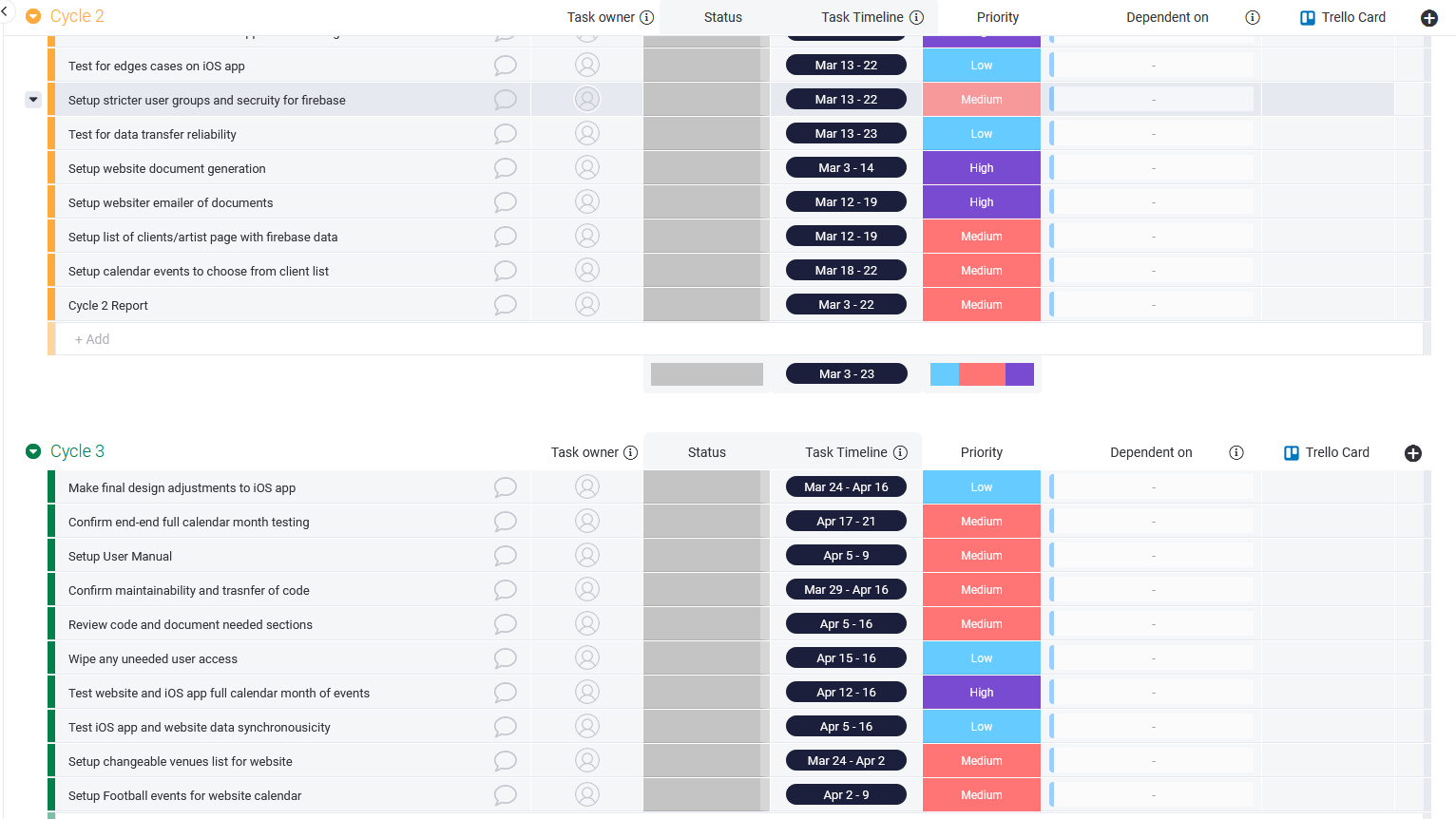
## Task Assignments

*Completed By Chase Dumbacher*

Tasks for the project have been primarily split between two smaller groups, the web team and the iOS app team. Chase and Sarah have been responsible for the iOS app and the Austins, Newkirk and Mongold, have been made responsible for development of the website. This split was made to help make it easier to give specific tasks to each group member and help create a more fair and balanced workload. Communication has been set up through discord because this is a common communication platform that all of the group members are familiar with. Discord allows us to have ping-able roles to notify specific teams or the entire team of important announcements or updates. Discord also has allowed us to create both text and voice channels that we can use to have chat meetings, voice meetings, as well as record progress and update teams of progress made daily. For ease of use we have also used zoom to facilitate our weekly meetings with Mike since this is easier and preferable to him. Specific delegation of tasks is discussed in the meetings and text channels of Discord, however we have also made use of a task organization system/website called Monday.com. This site allows us to create tasks, separate tasks by cycles, set deadlines and priority for tasks, as well as create gantt charts based on these tasks and timelines. To handle version control as well as file sharing we have set up a github for the project. Because the code is written entirely in javascript (node.js/react native) each member has been free to use any text editor of their choice. We have mostly used Visual Studio and Visual Studio Code.

This table shows all the planned tasks setup for cycle 1 as well as the deadlines and progress we made on these tasks. Currently the mailer authentication issues has posed to be a problem and is delaying full end-end testing of the iOS app, however the schedule for the iOS app development will be mostly done once the emailer works so it should not cause severe risk to the entire project as long as it is handled early on in cycle 2.

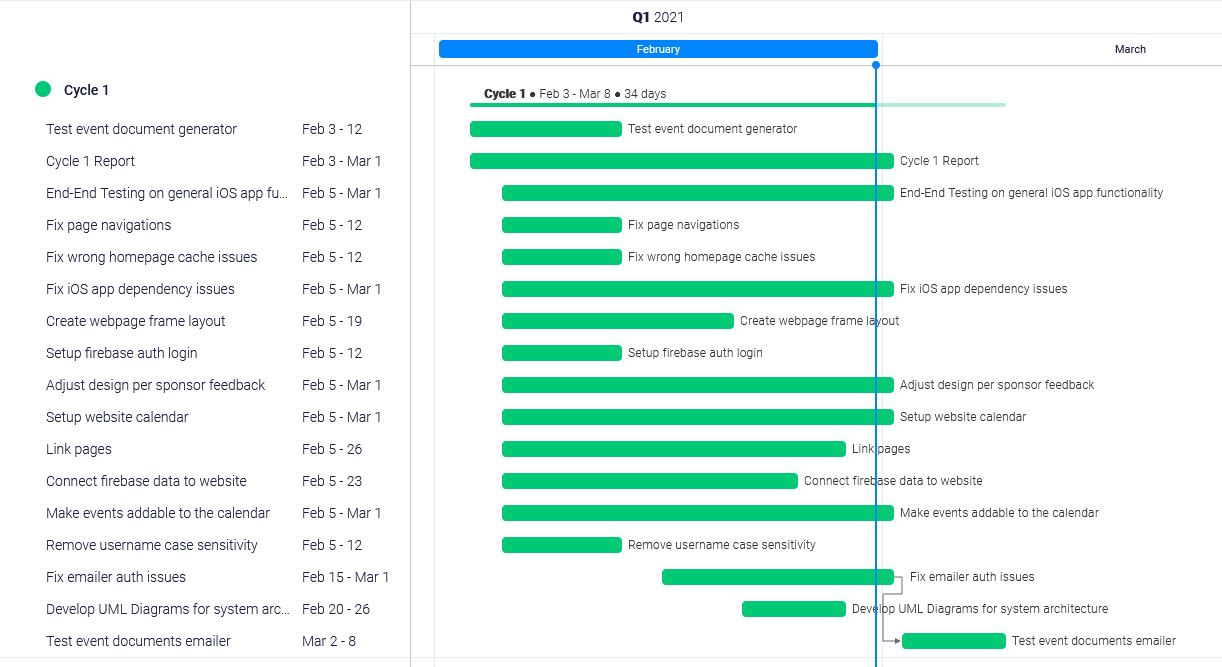


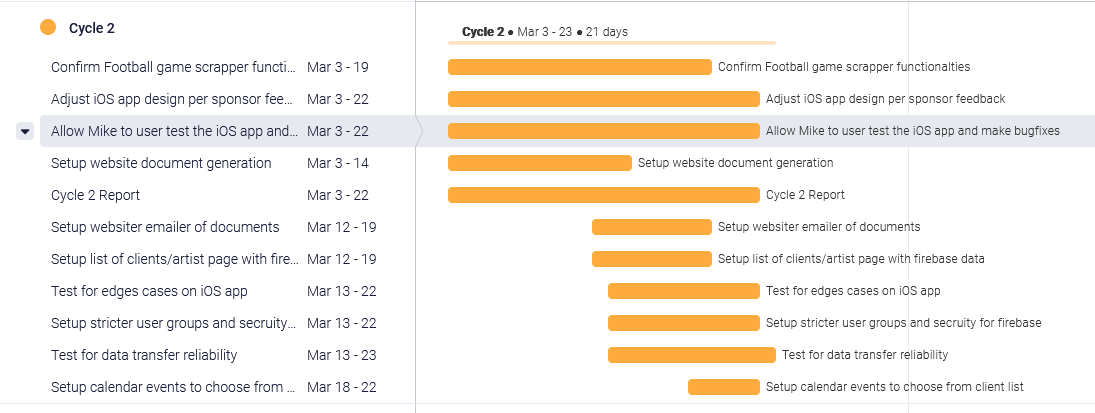
These tables show the planned tasks for the upcoming cycles and the planned deadlines for them. Although these tasks can be changed and added onto as the cycles are worked on these are an outline for how the project will ideally go.

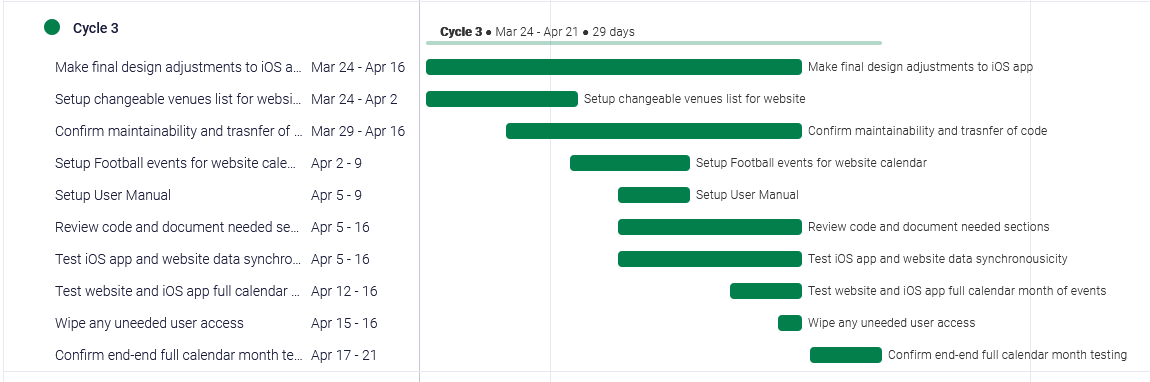
## Development Schedule

*Completed By Chase Dumbacher*

A detailed and more intractable preview of our development schedule gantt charts can be viewable [here](https://view.monday.com/1068854366-9fbc2a1db7bba73d7f3618c436ab6452?r=use1). Below includes screenshots of Cycle 1 - Cycle 3 of our planned gantt chart schedule.



******

******

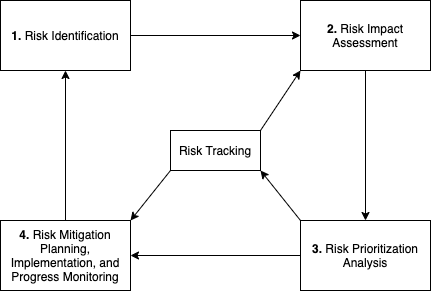
## Planned Code / Feature Freeze

*Completed By Chase Dumbacher*

The planned code freeze for this cycle is set for the end of the day on 3/3/21 since our group presents on Friday we thought this would give us time to assure the code is working how we want and work on the presentation without worry that the code will deviate from what we plan to present.

# Risk Mitigation

*Completed By Sarah Pham*

****

Risk mitigation is a process within Software Development that involves tracking any known risks, identifying any possible risks, and evaluating this process throughout the development.

Known/New risks within the app:

* Node.js was deprecated and was on version 8
* Undocumented code
* Repetitive code
* Bugs within the app

New risks with the website:

* Connecting to the same Firebase as the app and therefore the app may not update the same time as the Website (due to two separate projects)
* Website speed is dependent of user’s internet speed
* Setting to a domain name, currently is using a local host

Known/New risks with the Firebase:

* Billing account was unverified
* Account was on the free Spark plan
* There was found to be two billing accounts
* There was found to be two projects
* Easy access with authentication

Our risk mitigation plan consisted of tracking the known risks and analyzing the significance of their impact to the project. We then prioritized each risk, found solutions and implemented accordingly. An example would be a known risk within the app where Node was on a deprecated version (8). This risk was monitored throughout the software development process and was evaluated to be a low priority during the first stages of cycle one. However, after many bug fixes, we discovered that the latest version of Node was required in order to run Google Firebase cloud functions. We needed that functionality in order to have a proper emailer system. Therefore, the risk was reevaluated to be the highest priority due to its impact towards the development process. A solution was found to update Node to version 14 and we were able to continue working on the emailer. Lastly, our mitigation plan involved evaluating the effectiveness of the risk process. The team came to the conclusion that the risk process was indeed very effective in terms of preventing and managing risks. Moreover, the process has been implemented into our development cycle.

# Test Plan and Test Procedures

*Completed By Austin Newkirk*

On the website, testing has been simple yet effective. So far we have been testing our website by simply running through all the webpages and interacting with each new functionality as we develop. As an example, with the implementation of FullCalendar, we tested lots of functionality the calendar offers through different scenarios, and tested dummy data entries (not through firebase, but would be integrated similarly to how firebase data is pulled.) We plan on continuing this kind of testing, but also plan on adding new testing procedures as the website is built.

Some procedures we plan on adding are unit tests for all various data entries, including football games, dummy holidays/breaks, dummy performances, etc. Lastly, we want to add Mr. Moody himself as a tester, as we are ultimately building the website for him. We feel that his potential input/feedback in the testing phases will greatly improve our development in the future.

For the iOS app, we have a similar approach for testing. Our testing consists of running through numerous test runs in search for bugs, optimizations, and potential positive changes. Mr. Moody does have access to the iOS application because it is further along in development, as has been a great help along this process. As an example, we (along with teams in the past) noticed a home screen bug where after a second login, the homescreen landing page would not appear; rather the venues page would be the landing page. Because of our observations, we found the bug and patched it out in a later build.

Unit test procedures/implementation for the iOS app has been considered, though little has been currently done at the time of this document publication.

# 

# 

# Lessons Learned

*Completed By Austin Newkirk*

We, as a group, have learned numerous lessons from our time throughout cycle 1. It goes as followed:

1. Further solidified lessons already learned. The biggest takeaways we had during this cycle truly was learning how ridiculously important research and communication is when constructing code. Without our daily text conversations, our regular meetings with each other about our progress and plans, our weekly conversations with Mr. Moody and his input, we would not be able to build an iOS app and a website. That’s how truly important communication is. Additionally, a few hours of research before a coding section really goes a long way. As an example, the web team (consisting of Austin Newkirk and Austin Mongold) could have easily just started building an event calendar from pure scratch, creating a lot of development time for the project. However, after researching for about 2 hours, we were able to find an open-source software specifically designed for people in our situation, and has all the functionality we ever need and more to help complete this task. This research alone saved us tens of hours of development time.
2. There are advantages working in a small team. One takeaway we all had during this cycle was much work we accomplished with only 4 people. It is easy to communicate with a smaller team, it is easier to find times for meetings, and it is easier for all of us to be on the same page when developing software. With a larger team, it is significantly harder for everyone to communicate with everyone else; it’s a constant battle. This ease of communication allowed each individual person to be more effective compared to an individual on a large team project. Though a larger team should be more effective at delivering a completed application quicker, we learned that this relationship isn’t linear; rather it’s more-or-less logarithmic.

# Appendix A

*Completed By Chase Dumbacher*

**Status Reports**

[Status Sheet Week 1](https://docs.google.com/spreadsheets/d/1QGZSqCaEo36iFEgkhfgYDqMG5iHuQ1O6E5Ih__bPYUg/edit?usp=sharing)

[Status Sheet Week](https://docs.google.com/spreadsheets/d/1AlsY_vYUNZel98SOwmT8yPb6b22CRJyijM9SeshfO94/edit?usp=sharing)

## Meeting Minutes

2/04/2021

* Chase and Mike meet to discuss the desired design and button choices of the iOS app.

Time Spent : 30 minutes.

2/05/2021

* Chase, Austin N, Austin M, Sarah, and Mike held our weekly customer meeting to update Mike on our progress and receive any feedback on the product and work.

Time Spent : 30 minutes.

2/08/2021

* Chase, Austin N, Austin M, and Sarah held a team meeting to update the Web and App teams on the progress made so far in preparation for our meeting with Mike.

Time Spent : 30 minutes

2/12/2021

* Chase, Austin N, Austin M, Sarah, and Mike held our weekly customer meeting to update Mike on our progress and receive any feedback on the product and work.

Time Spent : 30 minutes.

2/20/2021

* Chase, Austin N, Austin M, Sarah, and Mike held our weekly customer meeting to update Mike on our progress and receive any feedback on the product and work.

Time Spent : 30 minutes.

2/20/2021

* Chase and Mike meet to discuss and handle the suspended bill account for firebase.

Time Spent : 30 minutes

2/24/2021

* Chase and Austin N held a team meeting to update everyone of any progress made, discussed status report attendance, and planned out work separation for the cycle report.

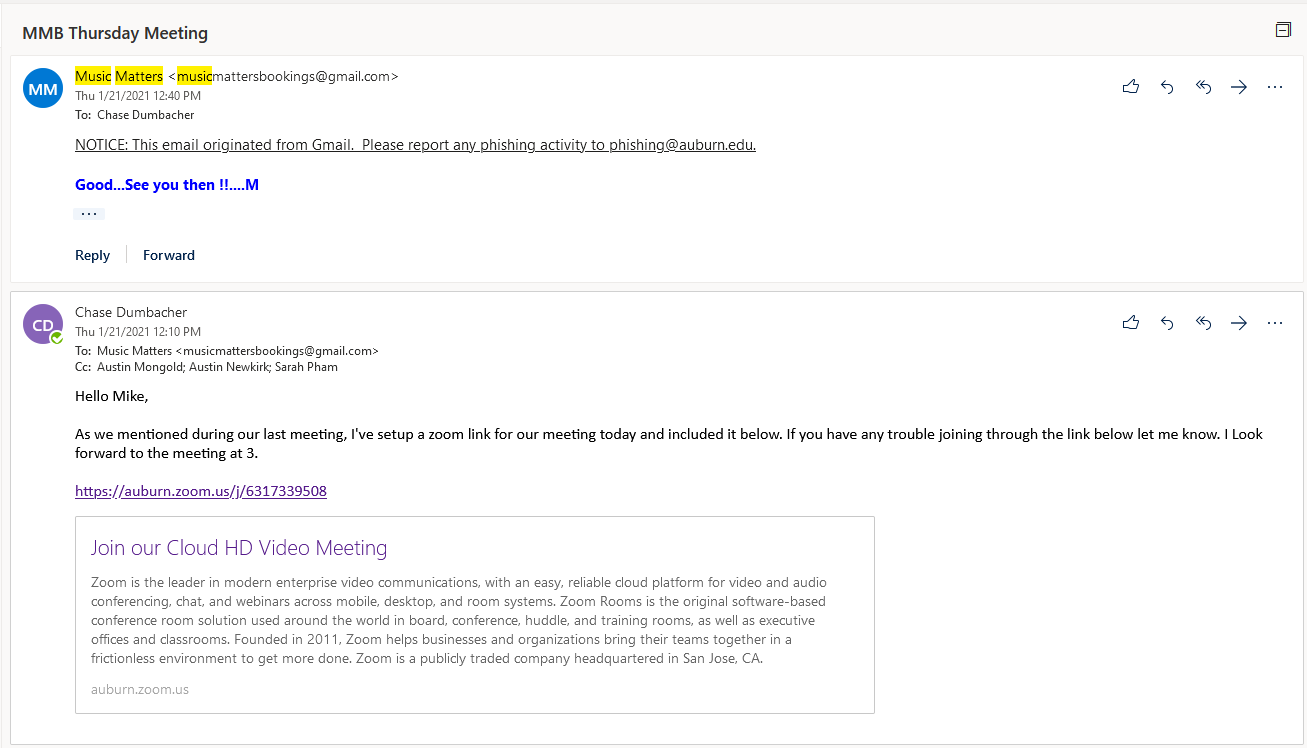
Time Spent : 30 minutes

## Correspondence/Digital Signature

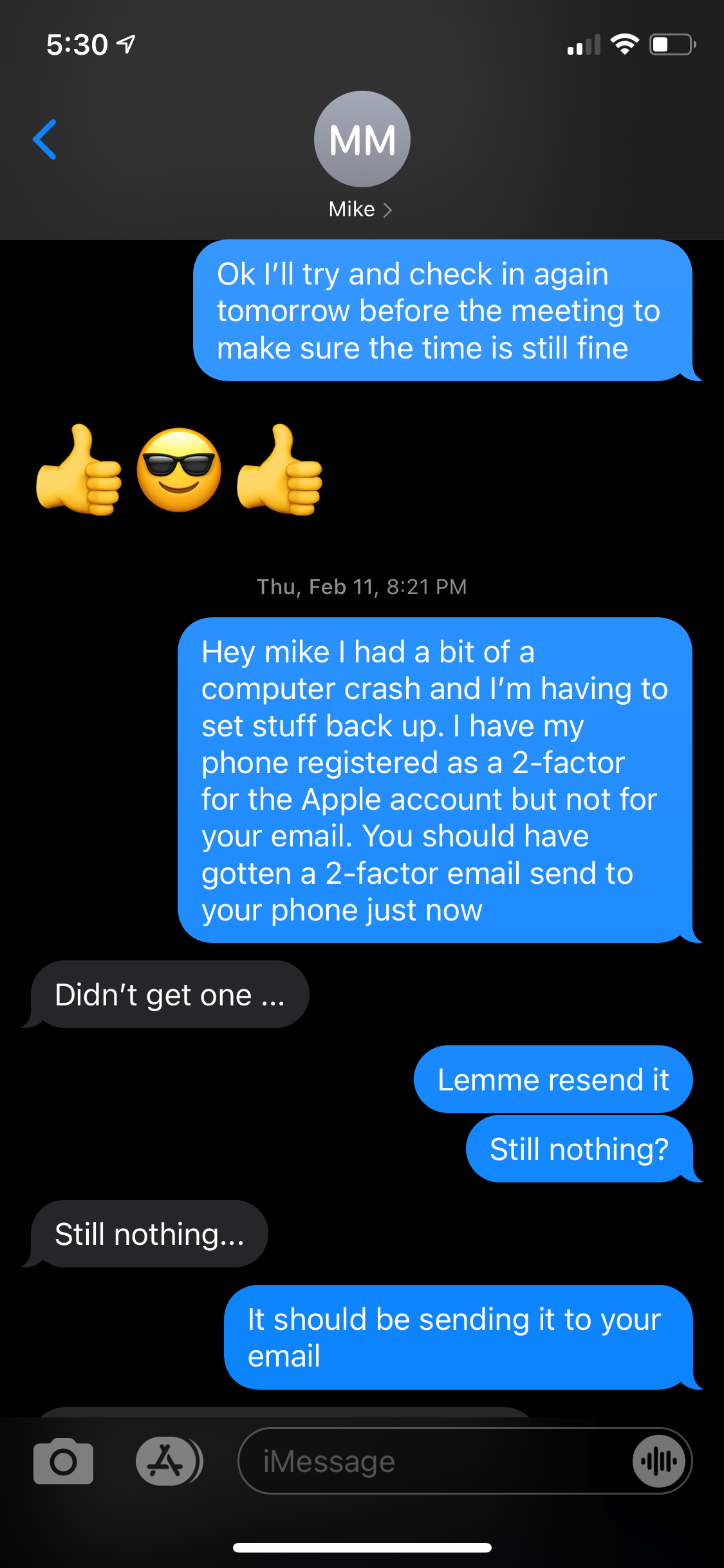
## 

****

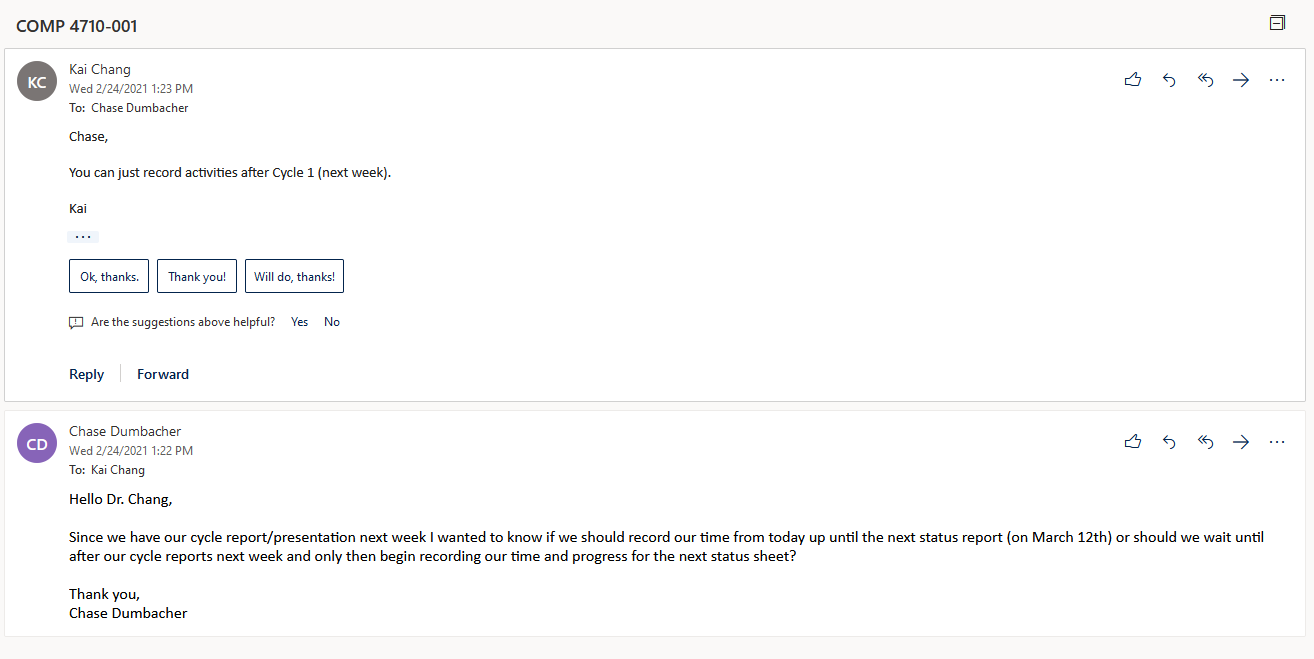
## Correspondence Between Team And Customer

****

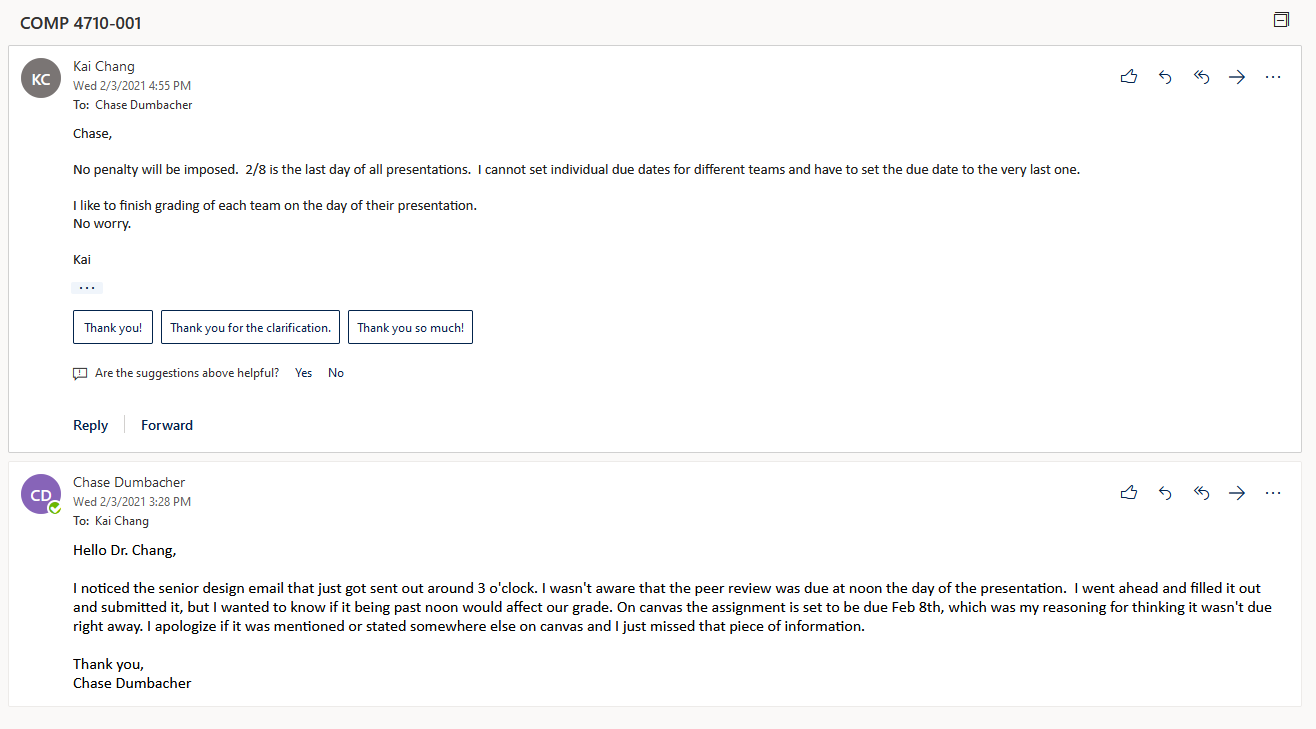
****

****

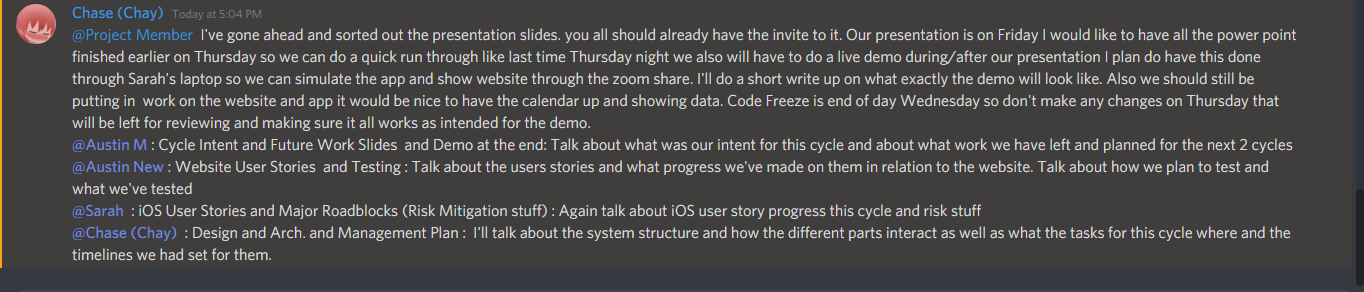
## Correspondence Between Team And Instructor

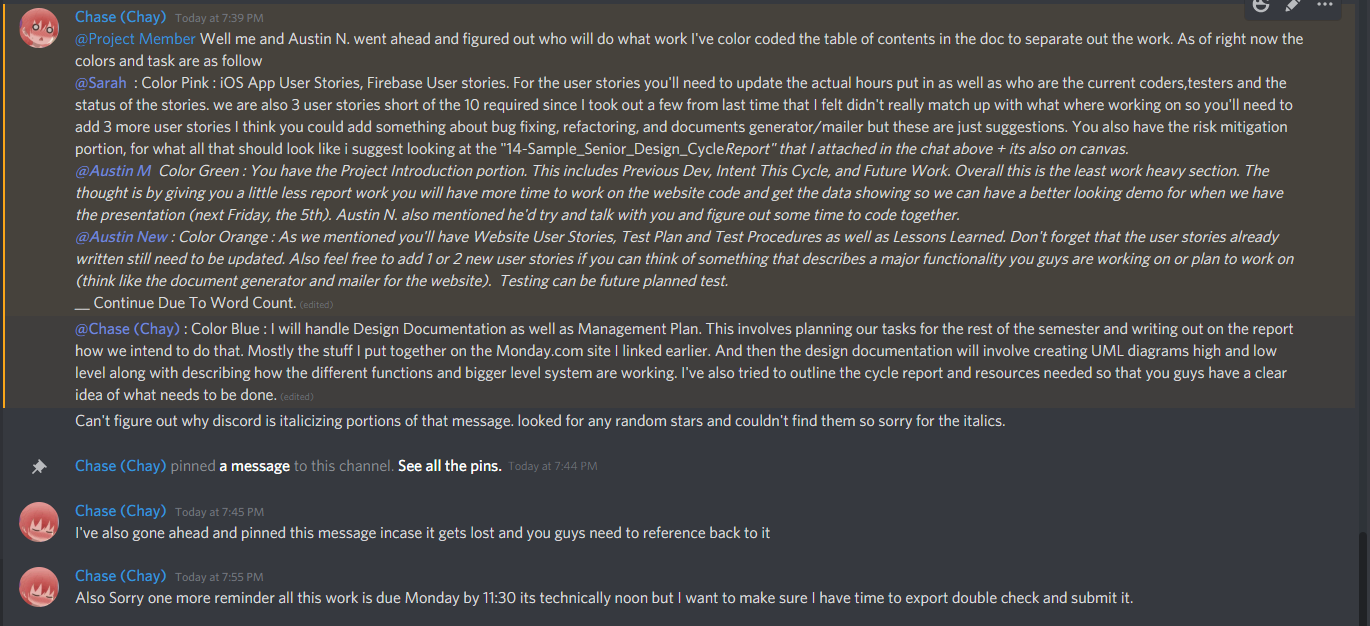
****

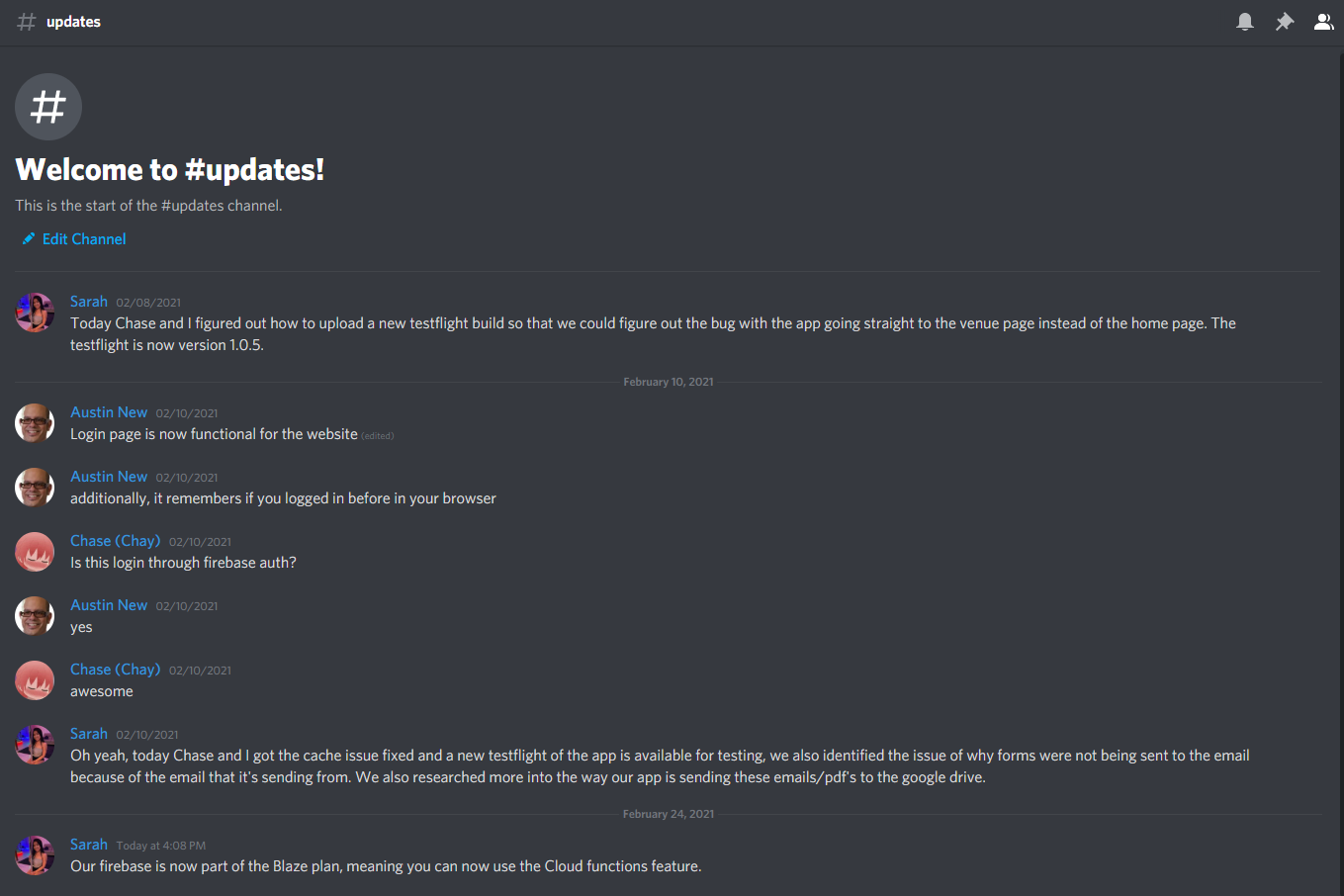
****

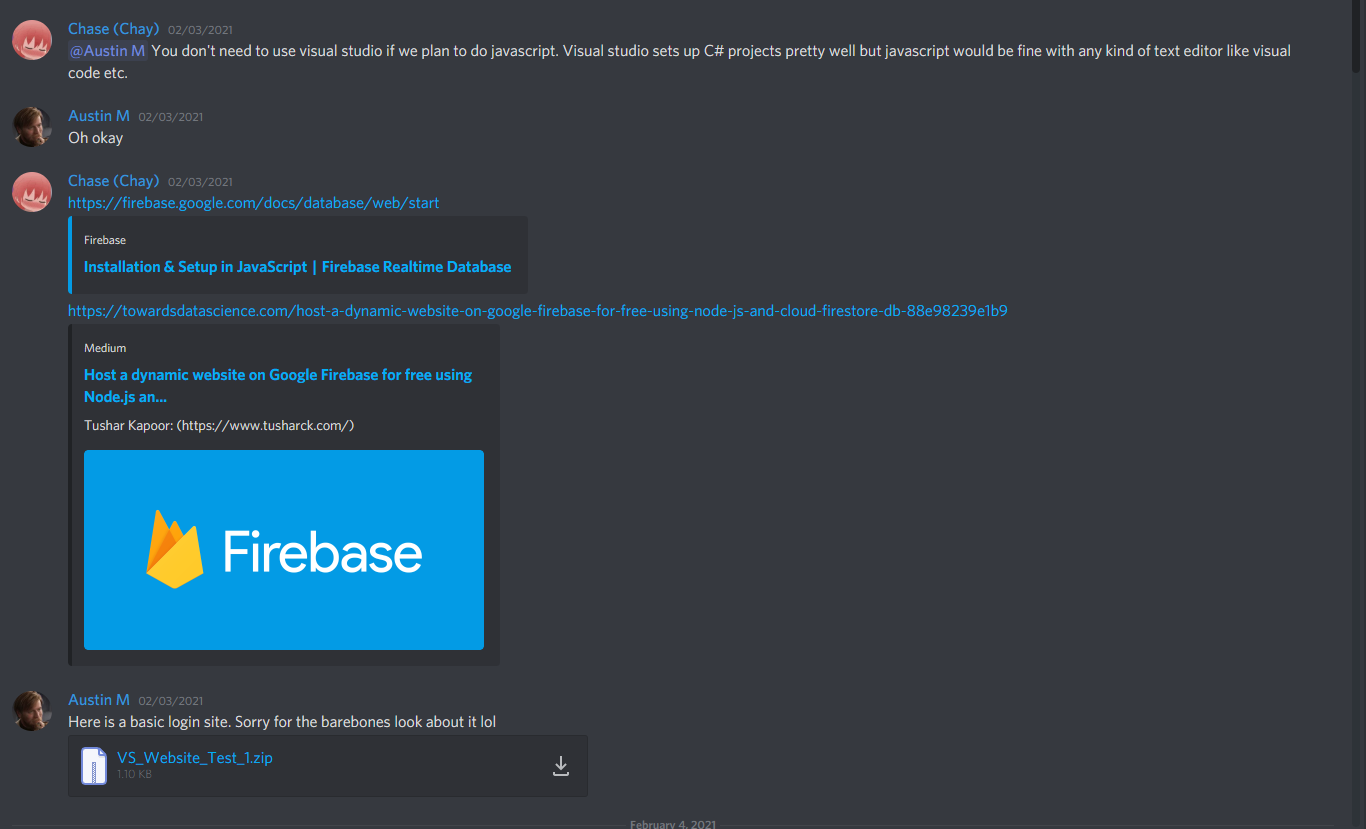
****

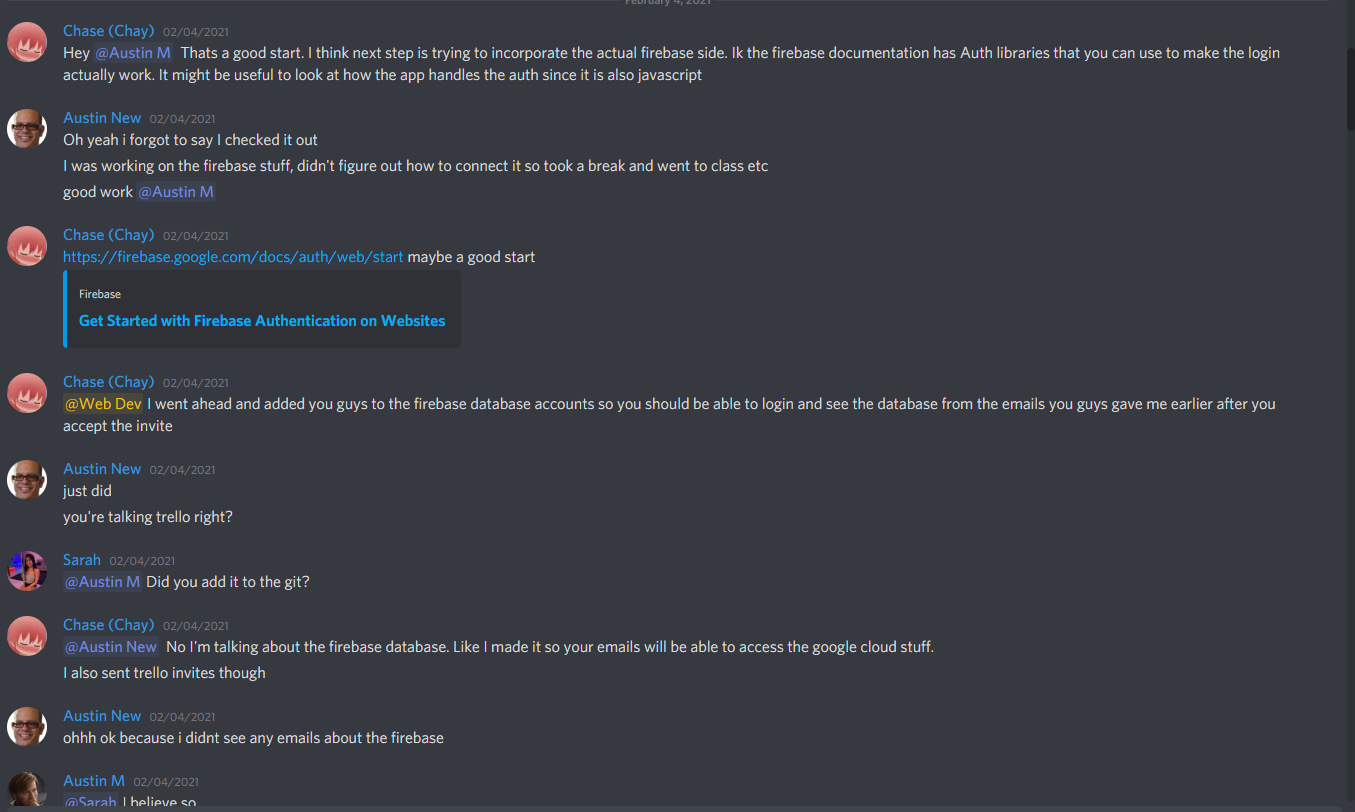
## Correspondence Between Team And Team

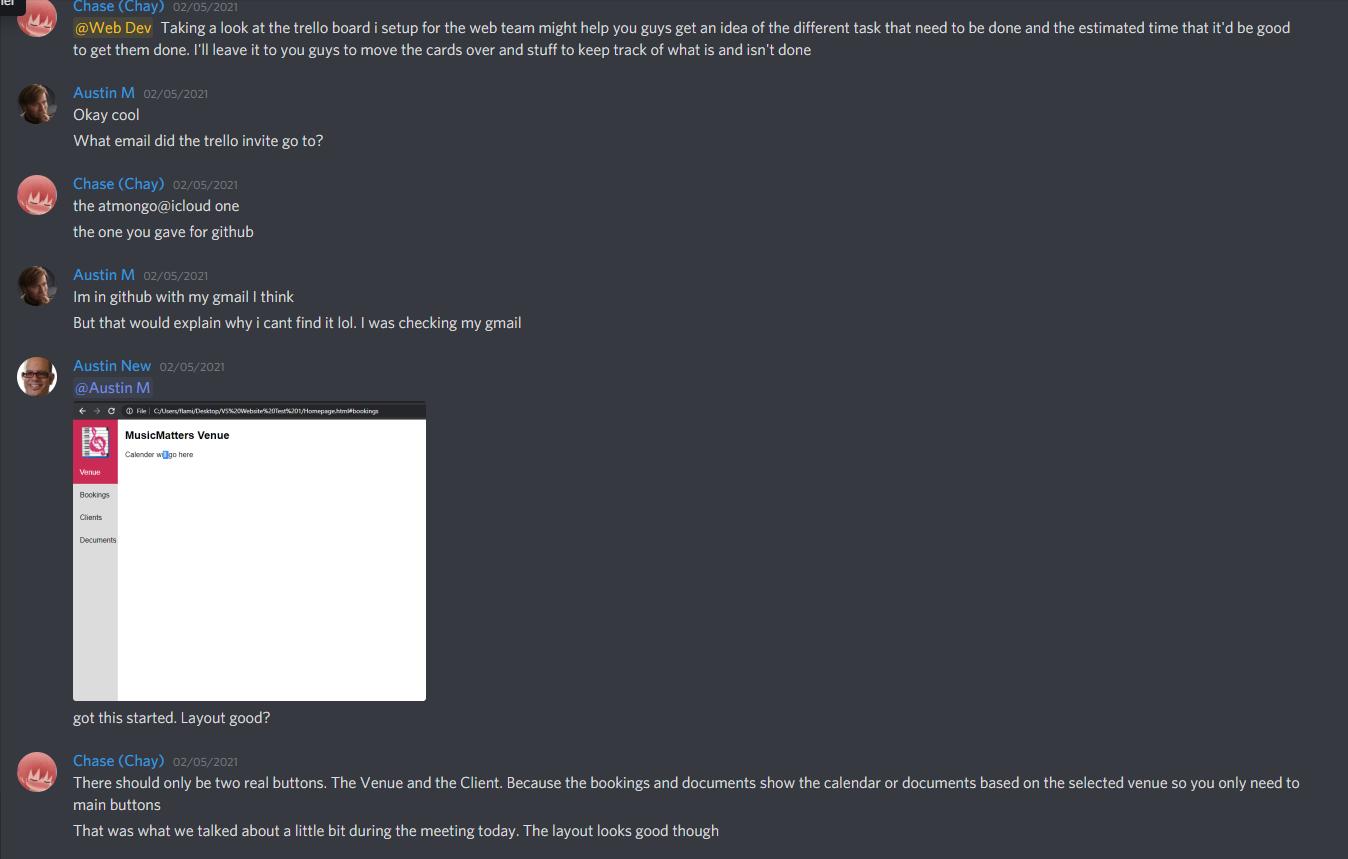


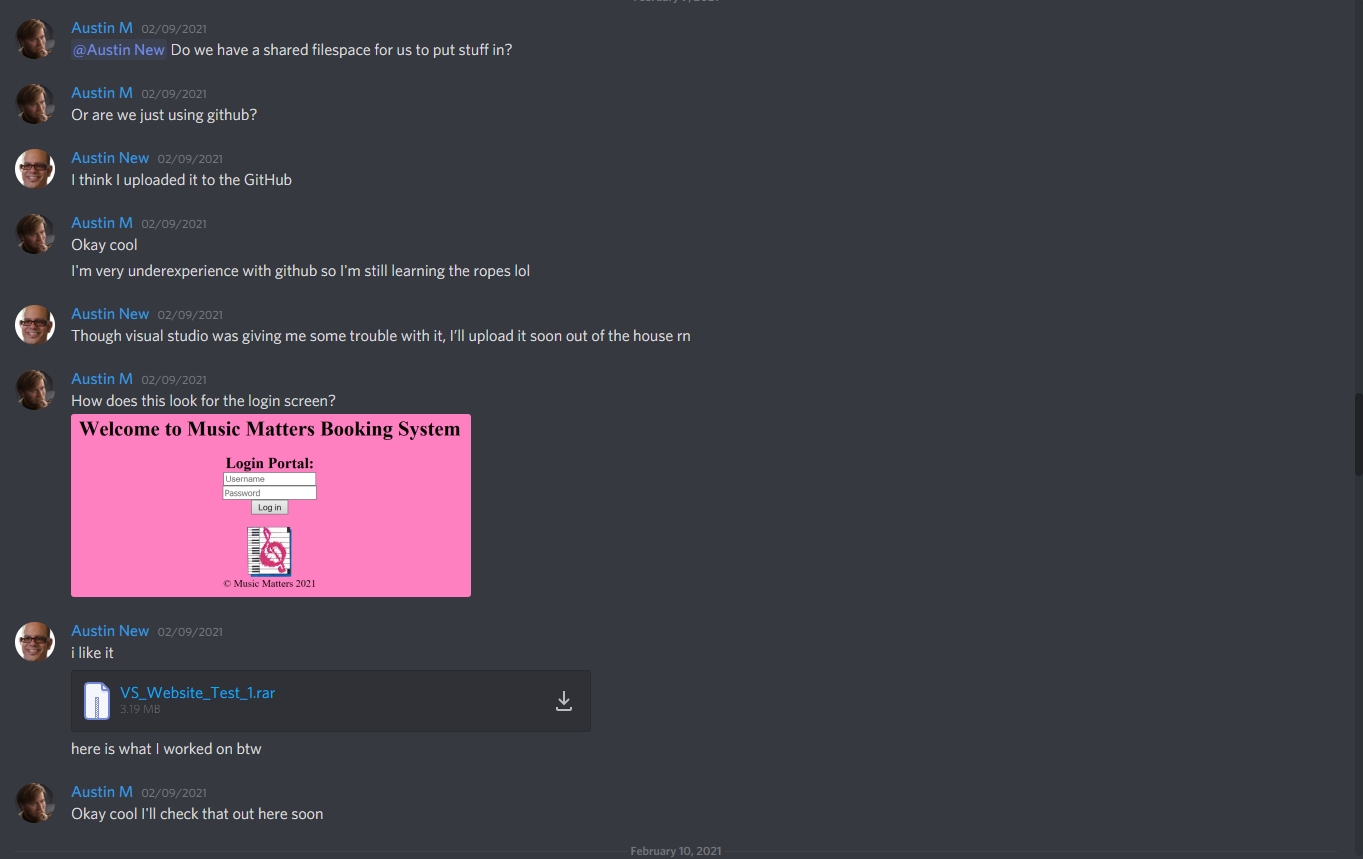
****

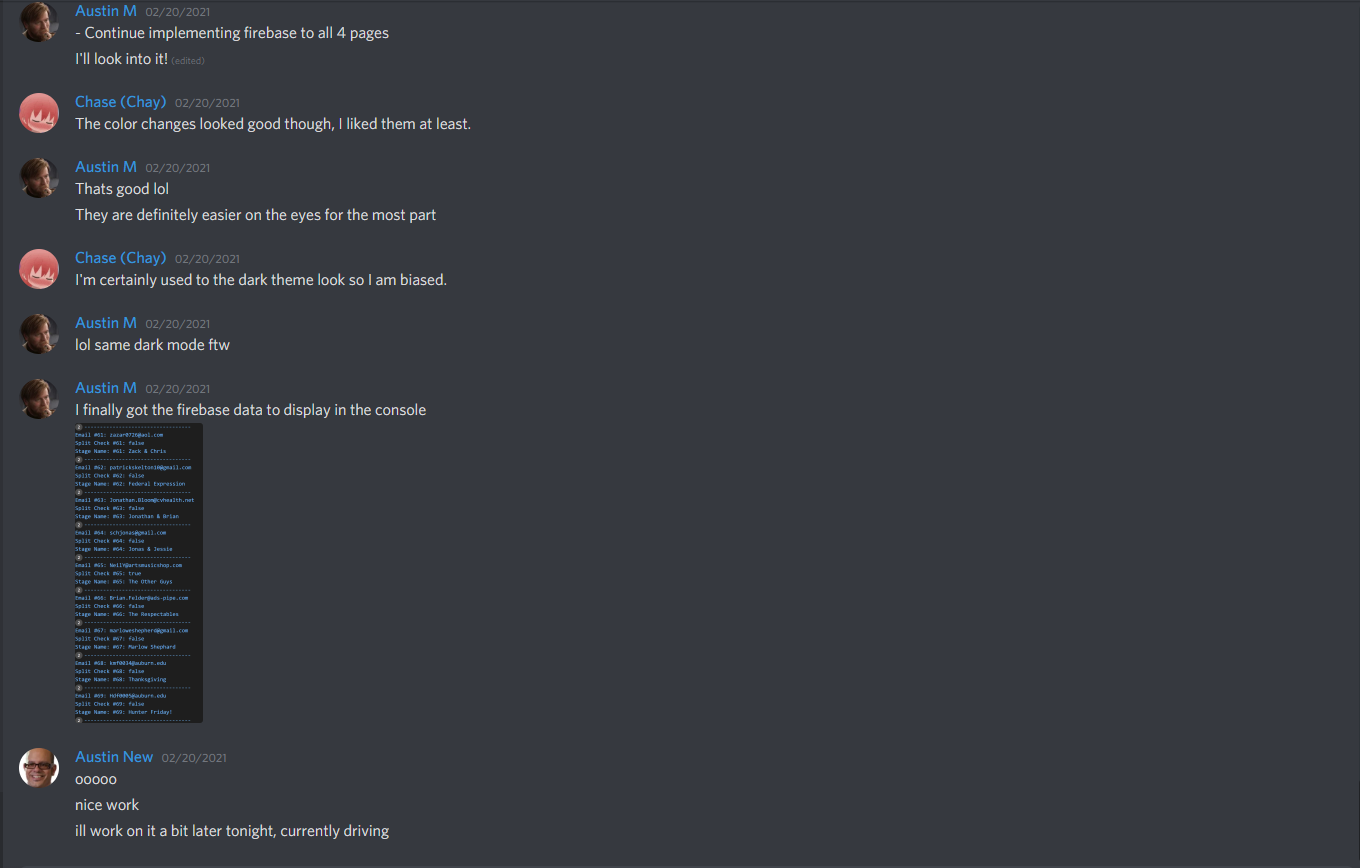
****

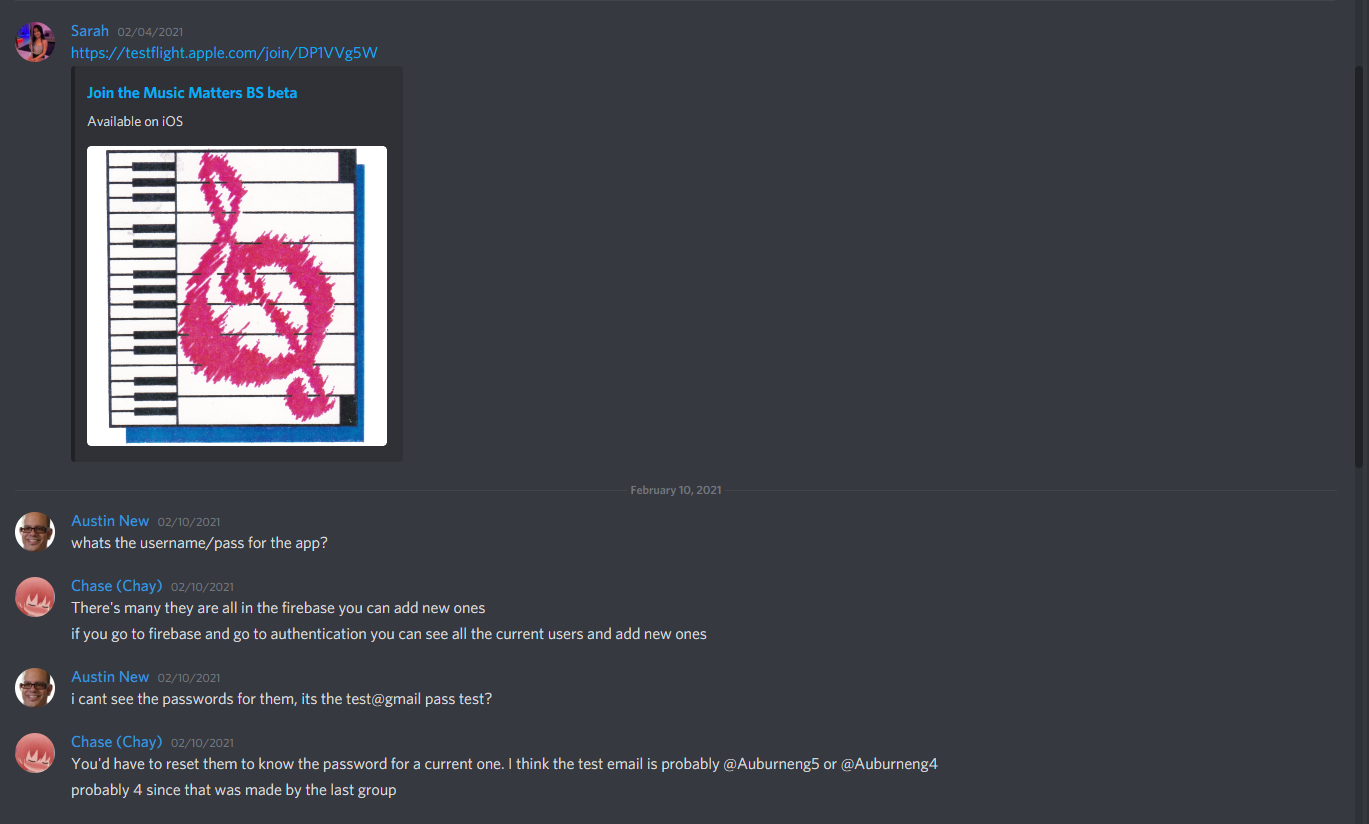
****

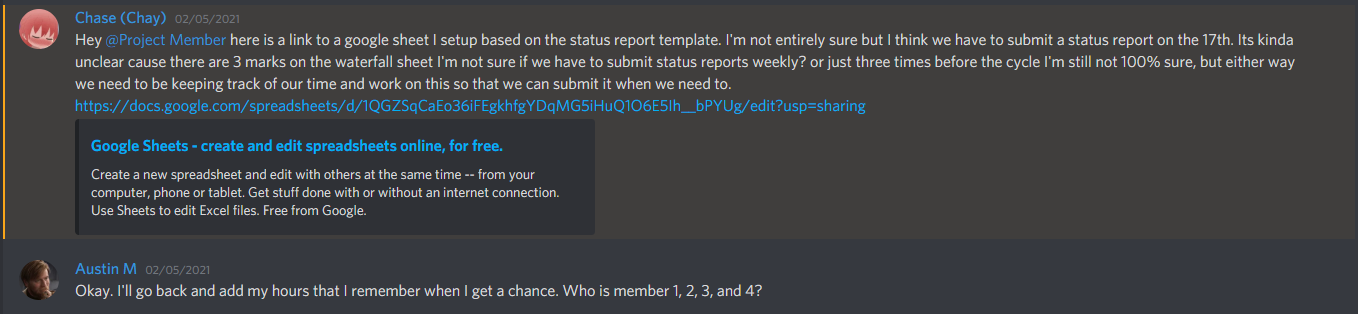
****

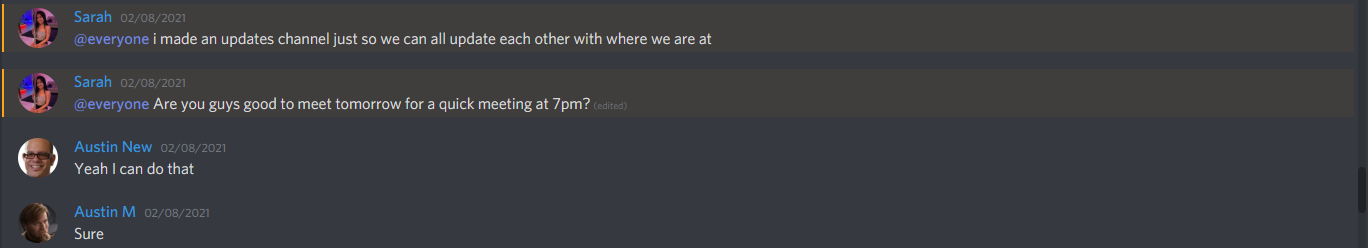
****

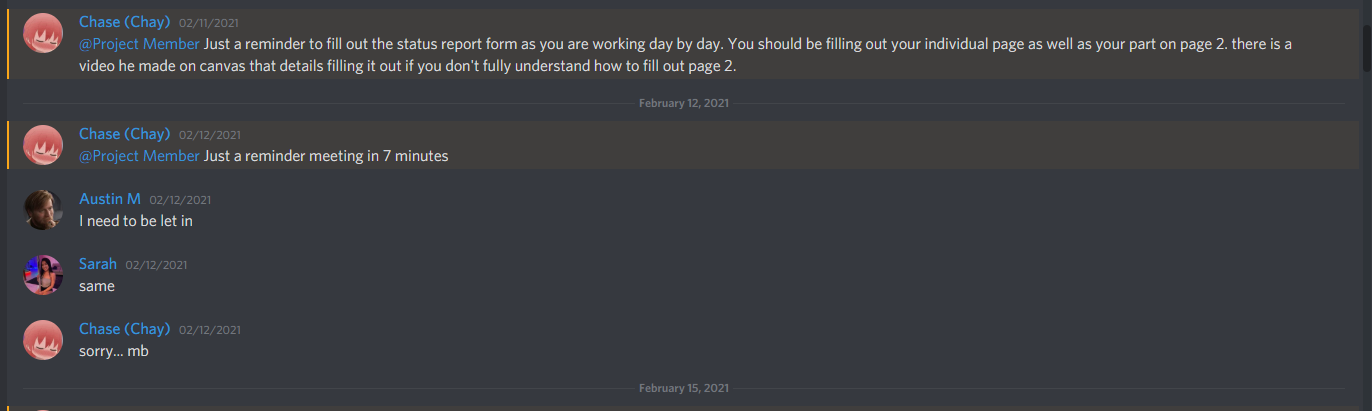
****

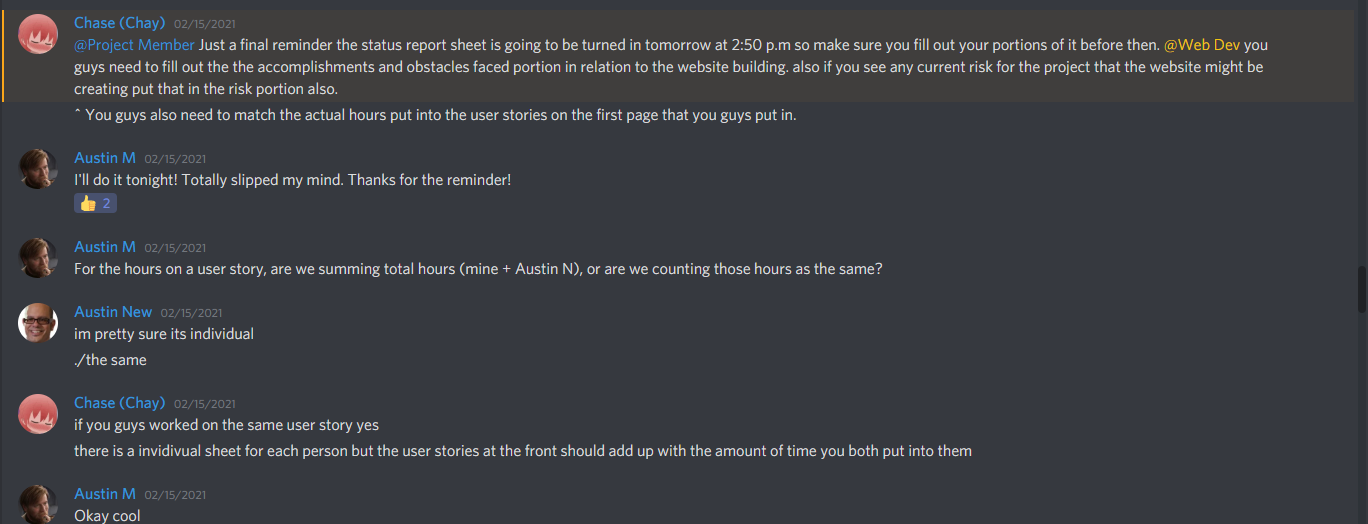
****

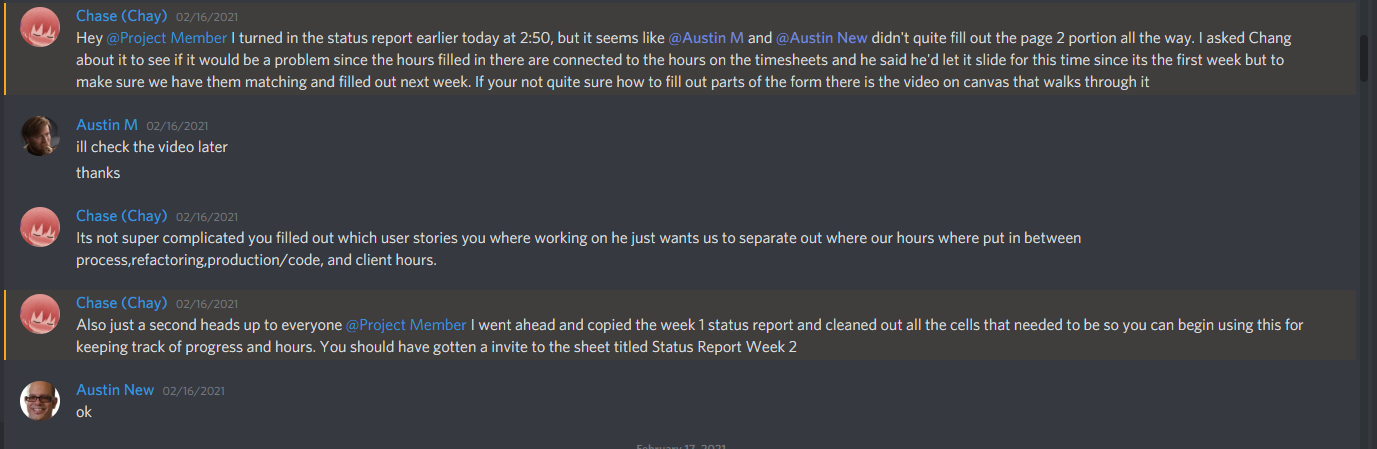
****

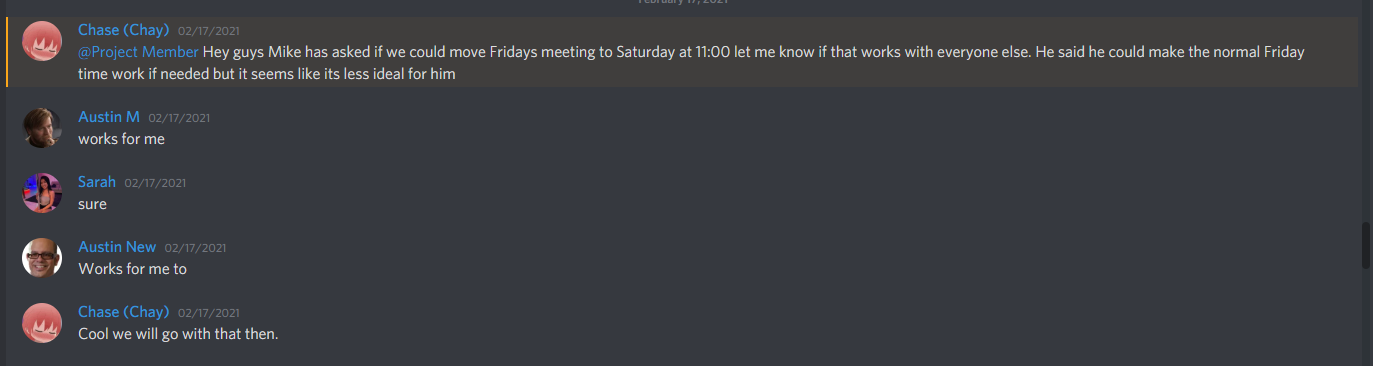
****

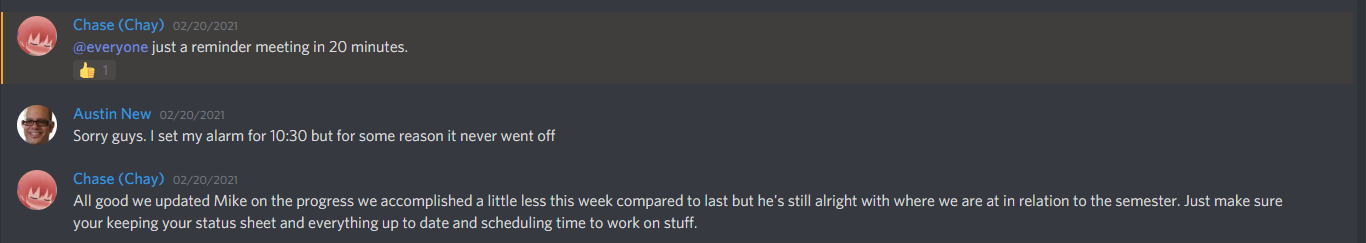
****

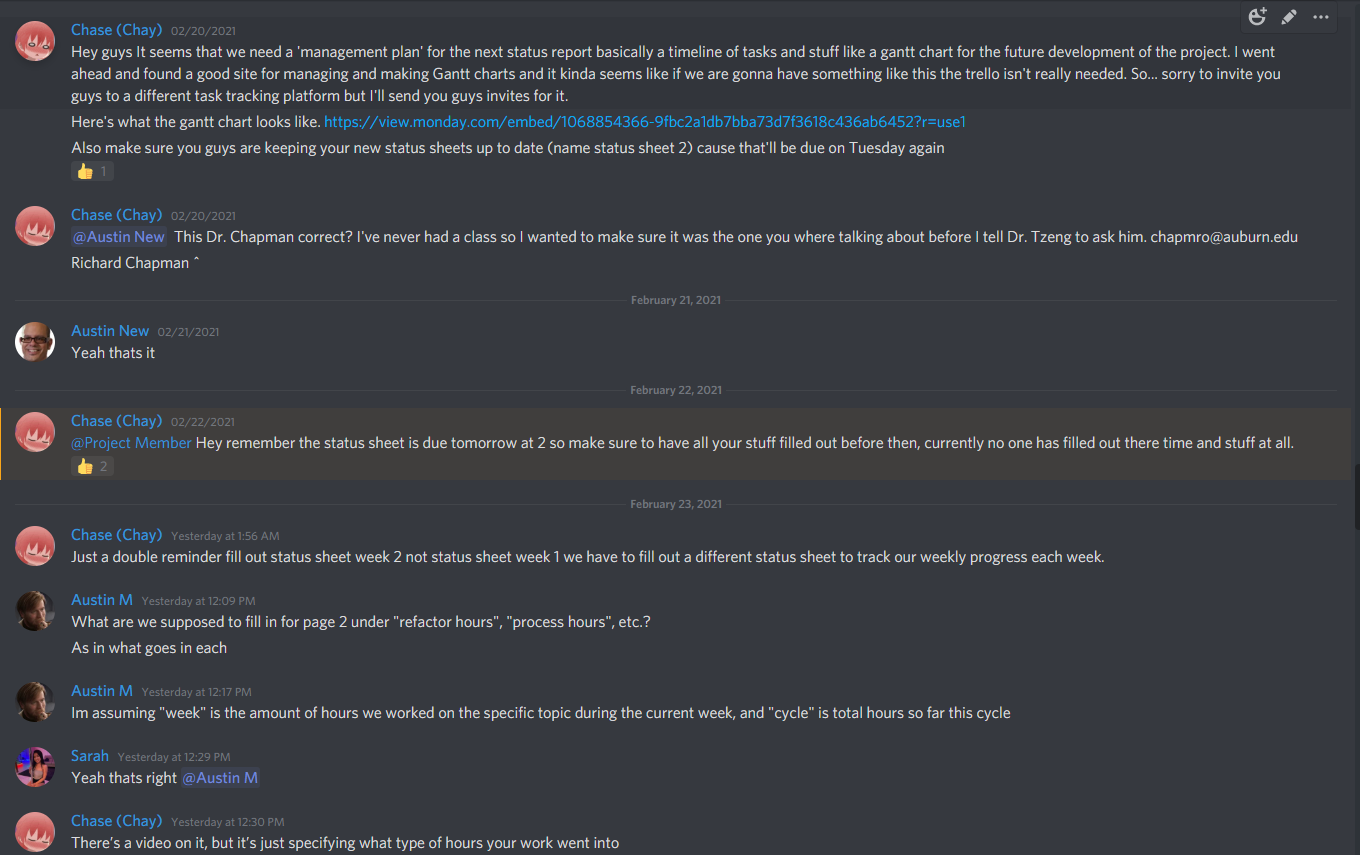
****

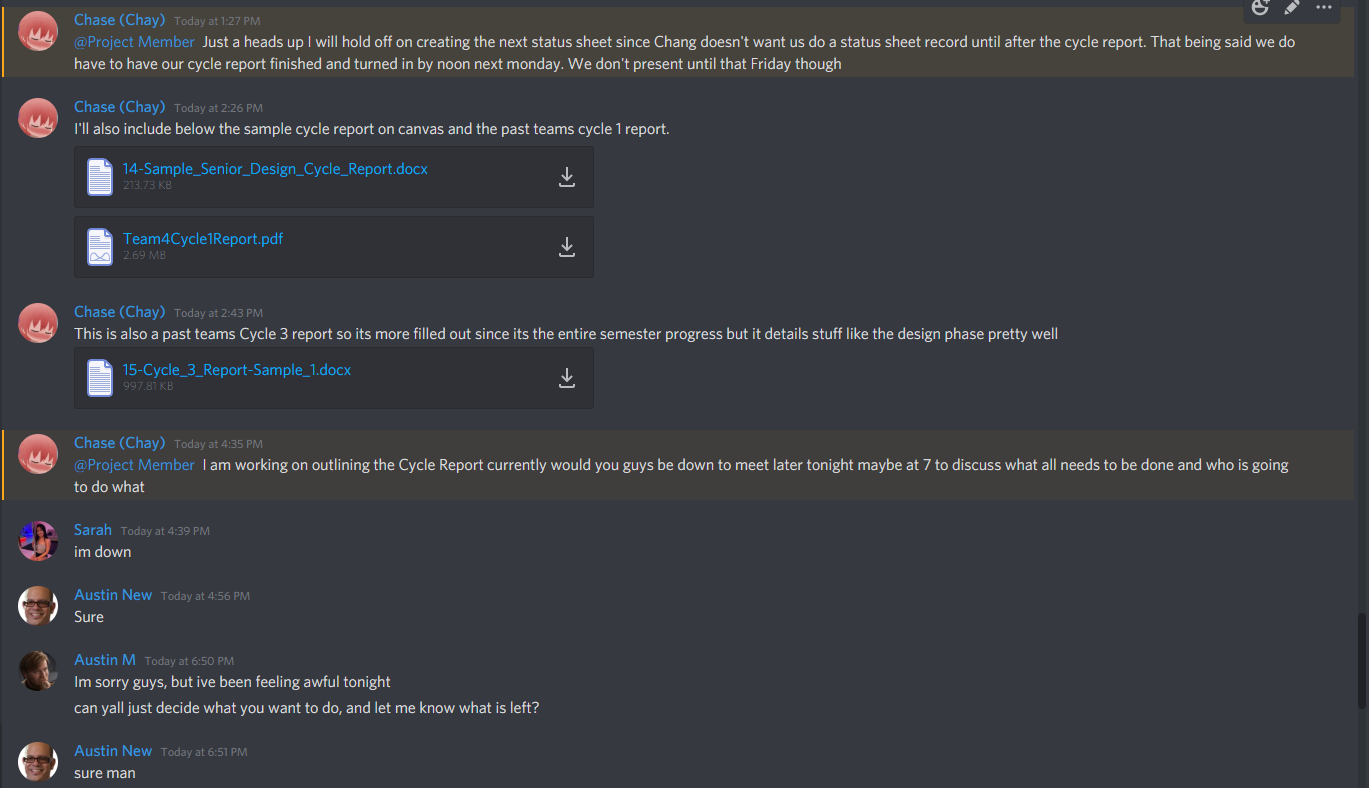
****

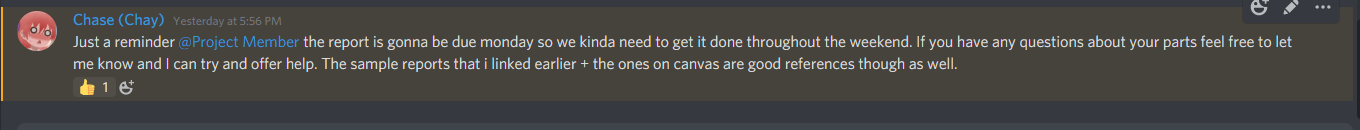
****

****

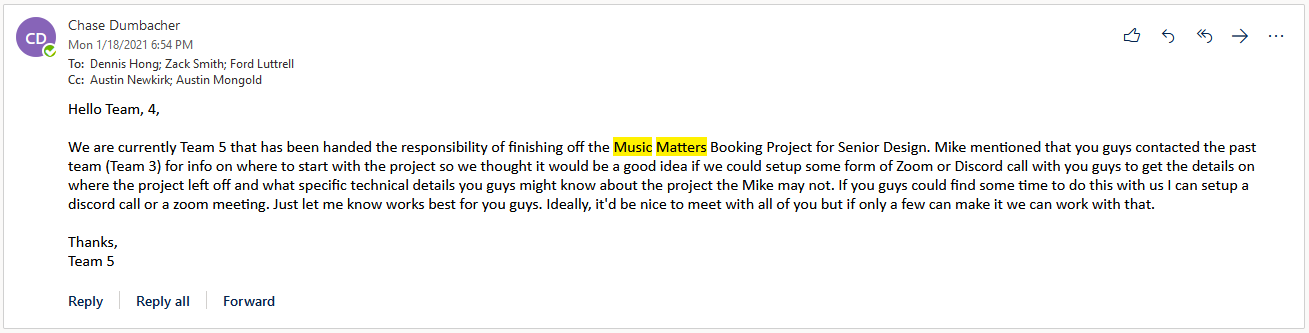
****

****

****

****

## Correspondence Between Team And Others

****

## Source Code

[MMBS Github](https://github.com/phamsarah/SeniorDesign)

