# Gratitude++ Design Specification

By: Joel Scott, Benjamin Seifried

## I. DESIGN DESCRIPTION

Gratitude++ is a website that will serve to aid people by giving them a safe space to keep their thoughts and feelings and secure them within our site's storage. Users will have freedom to make entries for their personal page as well as the community page. Users will only have the ability to delete (not edit) after confirming submission of a journal entry, leaving all the entries in an unedited format for them to be able to look at later.

The project has a mobile first approach to it, making it friendly for all users to access it regardless of the platform they are choosing to use. The website will be run through Replit using Node.js to establish a server, will utilize Google Firebase for data storage and retrieval, and will be designed to render on a Google Chrome web browser. [see **Appendix A**].

The following design flow will reference the UI Design Flow Document provided in the appendix.

**/Login page**

Users will begin by navigating to the [/Login] URL of the website. The page will contain a label with the application name and a sign-in button. Upon clicking the button, the user will be directed to authentication.

[see **Appendix C / Figure a**]

**/authentication**

The google authentication will redirect returning users to a separate page to complete the login process (see figure C, number 1). Users will be prompted to enter their username and password. (Firebase will handle the user authentication and verify their Google account.)

Once verified, users will be redirected back to the main page of Gratitude++, where they will be checked for an existing account on the site. The users will have a unique key that Firebase authentication generates that will be checked to find a matching userID within the database to determine if the user has an existing account or not.

If account information is found in conjunction with the userID, this will indicate a returning user and they will be directed to their home page.

If the user does not have an existing account or the userID cannot find a match, the user will be determined to be a new user and redirected to the /createNewUser page.

[see **Appendix C / Figure b**]

**/new user page**

The new user page will contain appropriate prompts in the form of labels, a text box for username input and a default image for a user avatar image and a submit button.

Users will be required to have a username and user avatar when their account is created. Usernames will have to meet a list of criteria outlined in the requirements document, and that name will be checked against the database to determine if that username is taken or not. If the username is taken, users will be prompted to try again via message dialogue, and the above steps will be repeated until the user has selected a unique username.

Users must also have a user avatar associated with their account. The avatars will be selected through a listing of 4 avatars, all selected by clicking on the avatar image to cycle through the choices. Once the user selects the avatar of their choice, it will be confirmed when they click the submit button and show on their account.

User account information will be stored inside the database along with all other relevant information relating to that user, including their username, user avatar, userID, and all information relating to journal entries. Once the user has successfully logged in or created an account, they will be brought to their personalized homepage.

[see **Appendix C / Figure b**]

**/navigation bar component**

The home page (and web page components EXCEPT for the /Login) will contain a navigation bar. The database will pull the user's username and user avatar and always display them at the top of the screen on the user's navigation bar (left-justified) and there will be a drop-down menu (right-justified). Clicking on the site's logo will redirect to the /home page. Opening the Drop-down will show options navigation to all pages in the web application to include: Home, Community, History, Resources, Settings, Log Out.

[see **Appendix C / Figure c**]

**/home page**

The homepage will contain a word cloud component prominently at the top of the page. Followed by a create article button and below that a listing of the 5 most recent posts by the user listed from most recent.

The word cloud will be comprised of keywords taken from all the user's headlines and content in previous posts. Clicking on one of the keywords will create an instance of the /filtered results component populated with a list of user created articles pertaining to that keyword.

Selecting the create article button will navigate the user to the /createJournalEntry page.

Clicking on a recent post from the list will create an instance of the journal entry view modal populated with the content from that article.

[see **Appendix C / Figure c**]

**/journal entry creation page**

The journal entry creation page will have a place for a headline, which will be limited to 100 characters, and a place for content, which will be limited to 1000 characters. The user will have a confirm button on the page to save the entry but can navigate away from the page without having any of the entry saved to the page. The site will not store any information from the previous visit to journal entry creation, so if the user navigates away from the page without clicking submit the information will be lost. Clicking the submit button will either create a new posts node if it is the user's first post, or add to the lists of posts the user has in the database. This new entry will contain a headline, content, the date it was created, and the visibility of the post. Users will also have the option before submitting their post to mark the post as public or private, which will be handled with a toggle switch for the user to select between the two. Private will be selected by default, which will keep user's entry hidden on their account and in their section of the database. If the user selects the option to make the entry public, when they hit the submit button the entry will be put onto the community page for others to view it. Once data has been marked as public it cannot but explicitly made private by the user again, only deleting the post will allow them to remove it from the community page.

[see **Appendix C / Figure d**]

**/word cloud**

The word cloud will be a component configured to display on the user's homepage when they navigate to it. The word cloud will render at the top of the homepage, populated with filtered headlines and content across the user's journal entries. If no entries are present it will display a message in that space telling users that no entries have been created yet. In the word cloud users will have the option to click on words to see all entries that correspond to that word. User's will be navigated to a separate page with the listing of all entries that contain that word, and by clicking on any of those entries they will be able to view them in the journalEntryView page.

[see **Appendix C / Figure c**]

**/history page**

Users will be given the option to view any previous journal entries they have created, and this is done by navigating to the journal history page. The journal history page will show all previous entries that the user has created, ranging from newest to oldest. The entries will be shown with the headline and the date and will be a clickable link for each entry. If the user clicks on a link, it will navigate the user to the view journal entry page where it will also show them the content of the entry. On the journal history page there will be a search bar for all previous entries to help users better find entries they have created. The search bar will allow users to search by headline by either typing in the entire entry or just a few words. If the user only types a couple words or letters, they will be shown all entries that start with that sequence of characters in the same format as on the journal history page. These entries will be clickable allowing the user to view them the same way as on the journal history page. Once the user has searched for articles there will be a go back button on the page to bring the user back to their journal history page.

[see **Appendix C / Figure e**]

**/resources page**

The resource page will include a page name label followed by a list of helpful URLs. The links will be directly related to mental health, and the content inside them will reflect that idea. Users who click on any of the links will be redirected away from the website to the URL of the link, with no direct return to the website without navigating back to the site's main page.

[see **Appendix C / Figure f**]

**/community page**

A major feature of the website will be the community page, which gives a place for users to share any entries from their account that they would like others to see. Users will navigate to the community page by clicking on the community page button within the navigation bar, which will redirect them to the community page. Once at the community page, users will be able to see all entries that others have marked as public within their personal entries. The page will be a general list of entries that are marked from newest to oldest, each with a headline and a date corresponding with it.

The entries for the community will be stored within a community section inside of the database, where each user's entries will be separated by a unique identifier for each separate posts. Inside each post node there will be a headline, the username of the original poster, the date that the entry was originally created, and a counter for the number of times a post has been flagged.

On the community page the entry will only display the headline and time of when it was created but will not show any content relating to the entry. To access the content, the user must click on the headline, and it will navigate to a page that will show the user's journal entry.

Due to the site not being constantly updated as changes occur, certain entries may be marked as private but have not been fully loaded into the community page. If a user selects a headline of an entry that has recently been marked as private, an alert will happen telling them that the entry could no longer be found. The page will not be correctly updated until the user reloads the page, at which point any entries that had been marked private during that time will disappear from the community listings.

A feature that will be present on all community entries will be a button to flag the entry. Flagging an entry will remove the entry from the community section of the database but the posts will still be viewable on that user's personal homepage. The flagging of entries will eliminate the need for strong moderation on the community page and allow it to feel freer for the users without a lengthy list of regulations stopping them from sharing their feelings. The flagging of posts exists to make a better community page for all users involved by removing entries that some might find offensive.

Examples of this include things like profanity, which some users may not want to see, and discussion of subjects that some feel is inappropriate or offensive. When a user flags a post, it will be removed from the community page section of the database and will follow the same rules as mentioned above about posts that had been marked as private. Users will be prompted when they attempt to flag a post if they want to proceed, and a box with a cancel and confirm button will appear. Clicking the cancel button will close the box and the post will not be flagged. Clicking confirm will successfully flag the post and remove it from the community page. When the user flags a post, they will get a confirmation message telling them that the post had been successfully flagged, and the posts will disappear when they reload the page. If a user attempts to flag a post that has been either flagged by another user or has been marked as private, they will receive a message telling them that the posts no longer exist on the community page. This ensures that users are aware whether their action has had any effect. No entries tied to the community page will allow users to post comments to avoid the need for moderation.

[see **Appendix C / Figure g]**

**/user settings**

On the user settings page the user will see their username and password, which will be displayed at the top of the screen. Within the settings users will have an option to change their existing user avatar. Avatars are stored within the sites code, and there will be a selection of 4 avatars for the user to be able to choose from. Users will have the option to choose one of the other 3 avatars that they don't have or to keep the same one. In the settings a cancel and confirm button will be present that allows the user to make their selection. If the user leaves the page their choice will not be saved, and they will keep the same avatar they have. If the user selects the confirm button the new avatar image will be updated inside of the database. The user will receive a confirmation message if their avatar has been successfully updated. The change will occur immediately on the user's page in their navigation bar at the top of the screen next to their username. Users will be given the option to delete their account if they choose to do so, effectively removing all their

information and posts from the database, along with removing any of their posts from the community page. Users can do this by clicking on the delete account button that will be located under the user settings page. When the button is clicked, a popup will appear asking the user if they wish to confirm the deletion or not. If the user chooses to delete their account, all their information will disappear, and they will be navigated back to the website's homepage. If the user clicks cancel, the popup will disappear, and the user will remain within their user settings page. Users will have an option to log out of their account on the navigation bar, returning them to the homepage of the website. If the user selects this option, they will be met with a popup asking them to confirm their selection to log out of their account. Clicking on confirm will log the user out and return them to the homepage of the site. Clicking the cancel button will close the popup and keep the user on the page they are currently on.
[see **Appendix C / Figure h]**

If the user is an admin, a special section in the user settings page will be created. Admins are currently only going to have the option to do one thing, and that is deleting user accounts. A list of all currently active users will be present sorted by username and each one will be clickable. Clicking on a username will prompt the admin if they wish to confirm the deletion of the account or not. If they choose to cancel, the dialog box will close, and nothing will change. If they choose to confirm, the user account associated with that username will be deleted from the database. The admin will also have a search bar at the top of the list of usernames that will allow them to search through the listing of users to find a specific one.

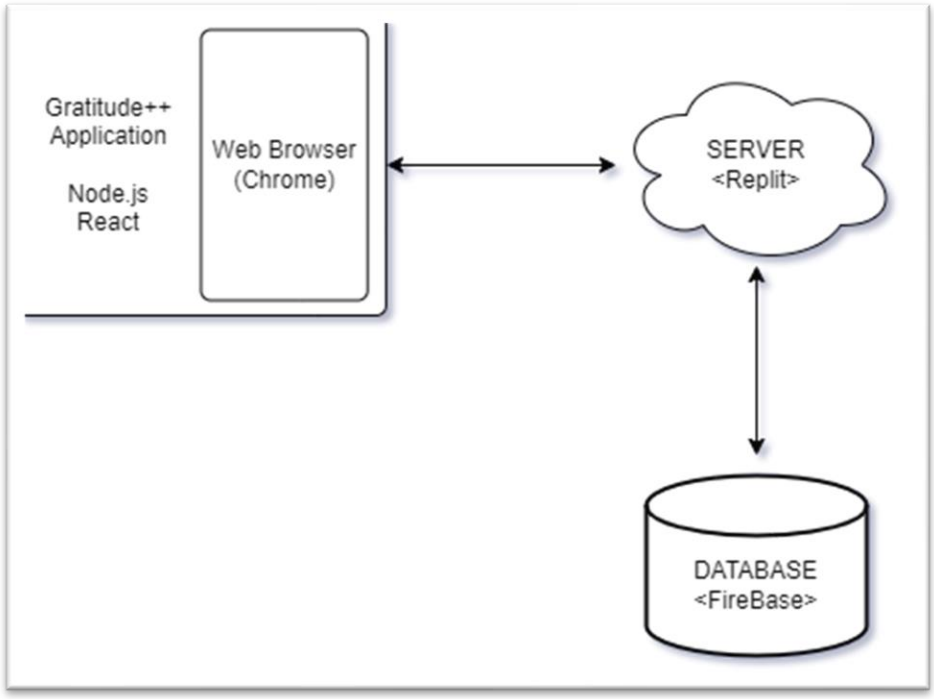
**Message dialogues**

Message dialogues will be created for all submit, delete, or relevant window closes if data has not been saved.

# II. APPENDIX

## A. APPENDIX – BLOCK DIAGRAM

**Figure a**




## B. APPENDIX – COMPONENT DIAGRAM

See attached document

## C. APPENDIX – USER INTERFACE STORYBOARD

See attached document for full view of User Interface Story Board

**Figure a**

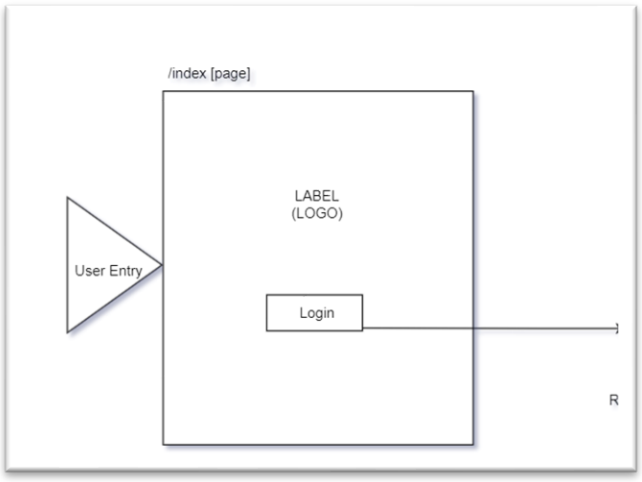



**Figure b**

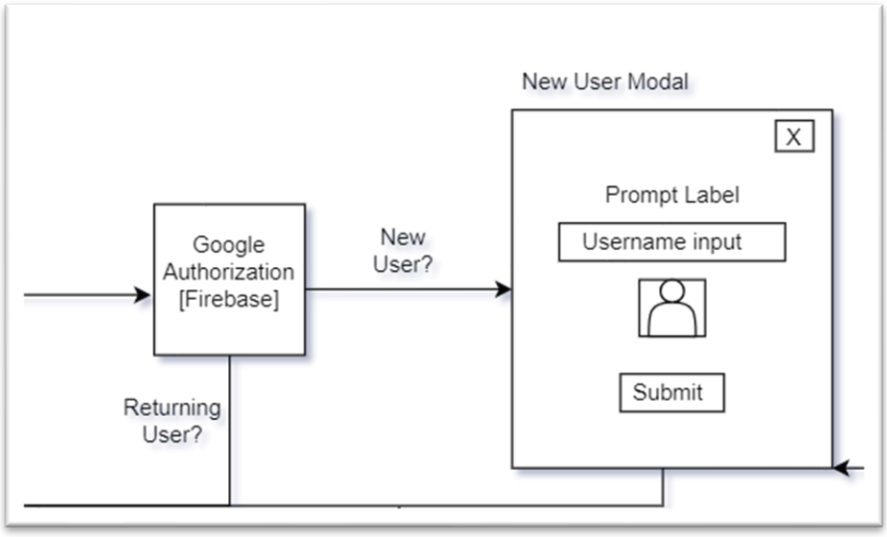


**Figure c**

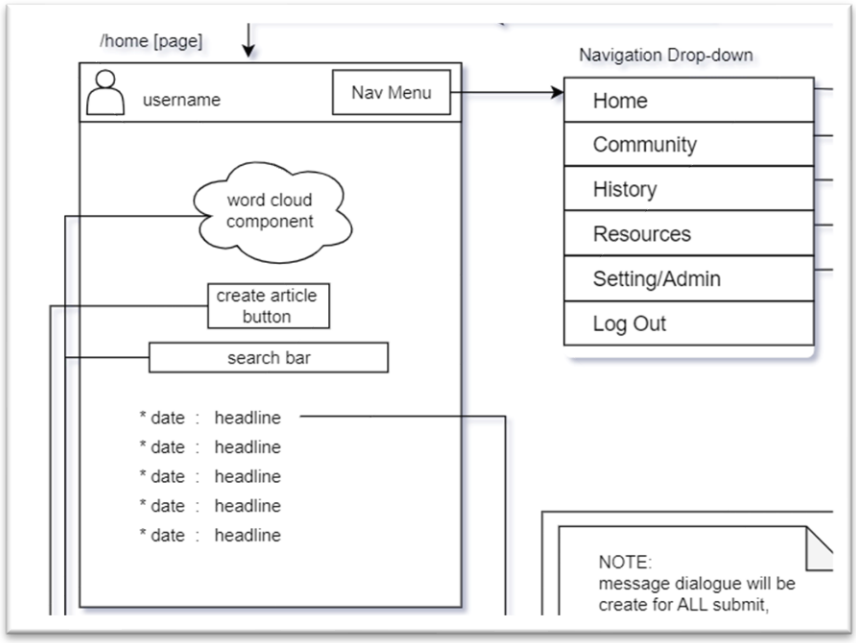



**Figure d**

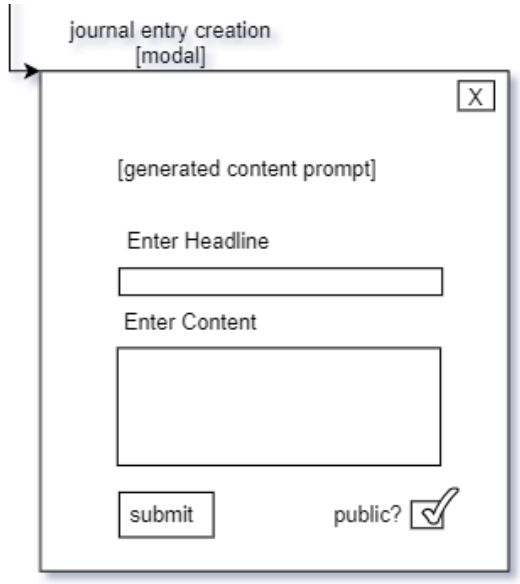


**Figure e**


**Figure f**

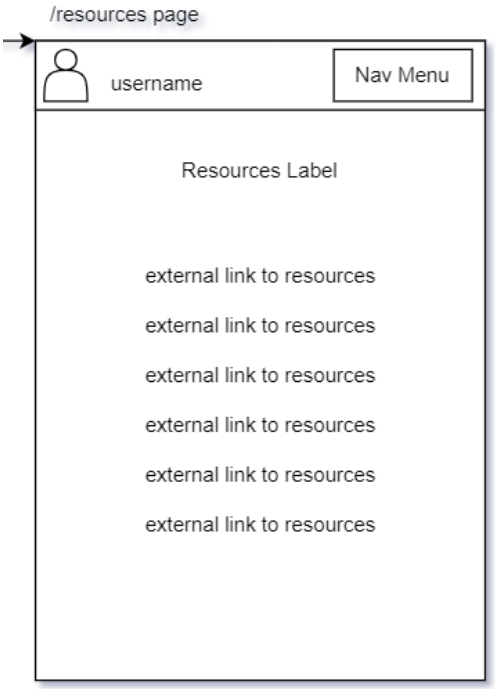
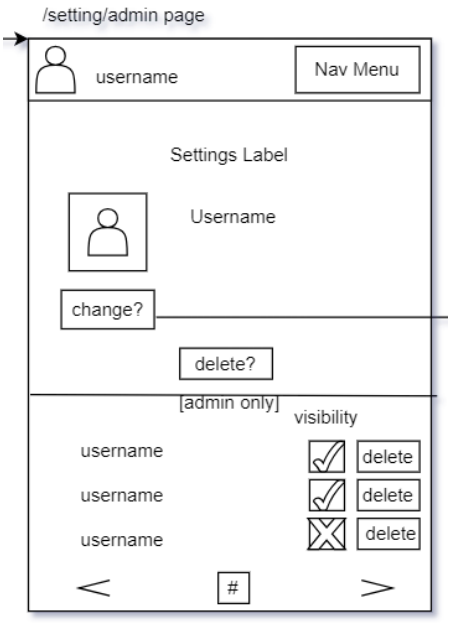


**Figure g**


**Figure h**



## D. APPENDIX – MESSAGE DOCUMENTATION

Messages being passed to/from the database

1. user authentication to firebase(text): Users will provide an email and password to firebase for google authentication, a message will display if the authentication was successful or not
2. create username(text): Creates a space in the database for that user to store their data and populates it with username, throws an exception if the username is already taken and tells the user to try again
3. select user avatar (image) Takes a selected user avatar and stores that information into the user's database section
4. Retrieve user profile info from database(text): verifies user to load their journal entries, username, and user avatar from the database, throws an exception if the user does not exist
5. New journal entries(text): users will send their journal entries to the database to store information including headline, date and time, content, and visibility of the entry when clicking submit
6. Word cloud(text): takes a list of headlines from the user in the database to create a word cloud of headline tags, throws an exception if no entries with headlines are found
7. Journal Search History(text): make a call to the database to find a matching headline, if matching headline is found send that data back to the user and present it on the screen, otherwise throws an exception
8. Community page(text): Call the database to get all entries under the database section and presents them on screen for the users, throws an exception if no community entries are found
9. Change user avatar (image): Calls a list of pictures from the database to choose from, and sets a new user avatar to the user's profile in the database if they choose to do so
10. Delete Journal entry(text): Calls the database to find a specific headline and id# to find and delete specified post from user's account
11. Delete user account(text): Send request to database to find and delete username, including all posts and child nodes associated with the account

## E. APPENDIX – STORAGE DOCUMENTATION


“users”: {

“UserID”: {

“username”

“profile_picture”

“Journal_Entries_#”: {

“Headline1”: {

“Date_and_time”,

“content”,

“visibility”

},

“Headline2”: {

“Date_and_time”,

“content”,

“visibility”

},

}

}

}

“usernames”: {

“user1”,

“user2”,

“user3”,

“user4”,

“….”

}

“Community_Posts”: {

```
“Username”: {
        “Post1”: {
                “Headline”,
                “date_and_time”,
                “content”
        },
        “Post2”: {
                “Headline”,
                “date_and_time”,
                “content”
        },
}
“Username”: {
        “Post1”: {
                “Headline”,
                “date_and_time”,
                “content”
        },
}
}
```

## F. APPENDIX – MISC DOCUMENTATION

### Developers/Architects

| Name | Role |
|---|---|
| Joel Scott | Developer, Architect |
| Benjamin Seifried | Developer, Architect |

### Tech-Stack

| Name | Purpose |
|---|---|
| Node.js | Framework/library to implement server |
| React | Framework/library to assist with front-end development |
| Firebase | Database to store information |
| Replit | Server Host |
| GitHub | Version Control |

### Tentative Schedule

| | |
|---|---|
| Week 1: August 28th – September 6th | **All Members:**<br>Project proposal, assessment of strengths and weaknesses, determination of technology and software, information gathering |
| Week 2: September 7th – September 13th | **All Members:**<br>Project requirements, refinement of project ideas, setup of technology for project development |
| Week 3: September 14th – September 20th | **Benjamin:**<br>Focus on front end development, Familiarizing with ReactJS, TypeScript, and JavaScript<br>**Joel:**<br>Focus on familiarizing with back-end technology NodeJS and Firebase |
| Week 4: September 21st – September 27th | **Benjamin:**<br>Begins to setup overall base of website, sets up front-end environment, and continues to work with ReactJS, TypeScript, and JavaScript |

| | |
|---|---|
| | **Joel:**<br>Begins to setup the database component, setting up back-end environment, and continues working with NodeJS and Firebase |
| Week 5: September 28th – October 4th | **Benjamin:**<br>Continue working on front-end environment, uses HTML (Hyper Text Markup Language) and CSS (Cascading Style Sheets) to create basic design of website<br>**Joel:**<br>Connects database to front-end environment, assists with front-end development, sets up connection for server component of the website |
| Week 6: October 5th – October 11th | **Benjamin:**<br>Continues working on front-end environment, works on creating UI and creating user journal entry page<br>**Joel:**<br>Assists with front-end, creates data structures to sort and manage user profiles within the server, tests, and debugs current back-end code to ensure quality |
| Week 7: October 12th – October 18th | **Benjamin:**<br>Begins with word cloud implementation, develops subject categorization<br>**Joel:**<br>Works on back-end development, works on user-authentication through Firebase |
| Week 8: October 19th – October 25th | **Benjamin:**<br>Continues front-end development, works with team to integrate front-end and back-end together<br>**Joel:**<br>Continues back-end development, works with team to integrate front-end and back-end together |
| Week 9: October 26th – November 1st | **All Members:**<br>Continue integration between front-end and back-end, begin working on project poster |
| Week 10: November 2nd – November 8th | **All Members:**<br>Continue working on project poster<br>**Benjamin:** |

| | |
|---|---|
| | Works on developing navigation bar, creating menu items to navigate around site<br>**Joel:**<br>Continues working on server component, tests and verifies that site has valid connection to server |
| Week 11: November 9th – November 15th | **All Members:**<br>Continue working on project poster<br>**Benjamin:**<br>Integrates front end and back end, works on UI and ensuring that navigation bar is functioning properly<br>**Joel:**<br>Works with database, ensures that information is being stored and brought to the website |
| Week 12: November 16th – November 22nd | **All Members:**<br>Finish project poster<br>**Benjamin:**<br>Continue testing features of the website ensure that all features function as intended<br>**Joel:**<br>Test back-end development, ensure that server and database components are working properly |
| Week 13: November 23rd – November 29th | **All Members:**<br>Continue testing of project, ensure that integration between the components is working |
| Week 14: November 30th – December 6th | **All Members:**<br>Continue testing and debugging, begin preparing for final interview and project presentation |
| Week 15: December 6th – December 12th | **All Members:**<br>Project finalization, continue working with group to ensure that they are prepared for the final interview and presentation |