Spotify PULSE

Personalized User-Learned Social Experience

Sprint 2 Retrospective

Team 18
Rheaa Sharma
Noah Stern
Dane Shaffer
Alex Polivka
Jason Zheng
Collin “Bodhi” Scott

What went well during sprint 2?

During this sprint, the members of our team were able to leverage the experience gained during sprint 1 along with each member's specialization which resulted in better implementations. Comparatively, more of our code written was modular, and our newer pages had much cleaner UI when compared to those from last sprint. Additionally, context was implemented to frontend components which greatly improved load times across the board.

Our communication outside of meetings also improved. The creation of communication channels for each specific component made it easier to follow progress, and increased our productivity. By the same token, our work on Github was much more organized for the first two weeks of the sprint, and devolved by the end when we were rushing to complete user stories. For a time, teammates had their own branches and main was kept up to date (more information about how we plan to keep it clean the whole sprint at the end of the document).

Meetings were also more productive. Our in person meetings were less chaotic for the most part, and we started holding some work meetings virtually which greatly boosted productivity for all members of the team. We had more sub meetings for individual components and user stories too, which worked out well and will be the norm going forward.

Current Sprint Details:

User Story #2c:

As a user, I would like to have custom backgrounds

#	Description	Time	Owner
1	Create UI for selecting background	1h	Alex
2	Implement custom background on pages	2h	Alex
3	Add preset backgrounds	1h	Alex
4	Create backend for background upload and background pulling	1h	Dane
5	Set up background storage system on the server	1h	Bodhi
6	Set up background file path storage in the database and ensure database and server are always synced	1h	Bodhi
7	Manual testing and testing documentation	1h	Alex

Completed:

The user is able to enter their profile/settings page, and scroll towards the bottom to change their background. There are 3 curated backgrounds they can select from, but they are able to choose their own image using a (valid image) URL. If the user chooses a curated background, or specifies a custom image, then the background is immediately reloaded. When the user decides to save their changes using the “Save Changes” button, the new background is saved to be displayed when the user reloads the page or logs out and logs back in. If no custom background has been chosen, the webpage has the default background that reflects the theme’s background color. In the future, a server should be set up to allow the user to upload images from their computer.

User Story #2d:

As a user, I would like to have preset themes to change up the interface’s look

#	Description	Time	Owner
1	Create UI for selecting theme colors	2h	Alex

2	Create UI for managing custom themes	1h	Alex
3	Implement custom colors on pages	3h	Alex
4	Add preset theme colors	1h	Alex
5	Create backend for custom theme upload	1h	Dane
6	Store preset themes in the database	0.25h	Bodhi
7	Manual testing and testing documentation	1h	Alex

Completed:

The user is able to enter their profile/settings page, and change their color theme. There are 3 curated themes they can select from, but they are able to create their own theme with a hexadecimal color picker. The user can pick a background, text, border, and accent color. These sets of custom colors can be saved as named custom themes, which can later be overwritten (to change colors and name), or deleted. When a custom or preset theme is retrieved, the theme colors are immediately loaded on the page. When the user decides to save their changes using the “Save Changes” button, the new background is saved to be displayed if the user reloads the page or logs out and logs back in. When the user creates their account, the colors default to the dark preset theme.

User Story #2e:

As a user, I would like to be able to view my Pulse friends

#	Description	Time	Owner
1	Create a storage method for friends list with packers/unpackers and getters.	1h	Bodhi
2	Create database functions for editing friends list through adding or removing friends.	0.25h	Bodhi
3	Create Database storage methods for friend requests with packers/unpackers and getters	0.25h	Bodhi
4	Create Database functions for editing friend requests	0.25h	Bodhi
5	Populate friends with their chosen icons	1h	Bodhi

6	Create UI for friends page that accommodates any amount of friends and has an add friend button, request button, and updates asynchronously between different accounts	2h	Bodhi
7	Create UI for friends tab on the right side of the screen that accommodates any amount of friends, shows any relevant information and lets you scroll	2h	Bodhi
8	Create an add friends tab on the friends page where the user can search for friends in a search bar and send friend requests	1h	Bodhi
9	Create a requests tab on the friends page where the user can see friend requests and accept or reject them	1h	Bodhi
10	Add functionality to add friends and pull list	1h	Dane
11	Add functionality to unadd friends	0.5h	Dane
12	Manual testing and testing documentation	1h	Bodhi

Completed:

A user is able to view their friends on the friends card and also the friends page. If a user has no friends, then a message saying so and directing them to add friends can be seen on the friends page.On the friends page, the user can remove friends, as well as navigate to the add friends page and requests page. On the add friends page, a user can search for other users, with complete error handling for searching for non-existent users and other search inputs. Adding a friend results in a request appearing in the added users inbox. Requests can either be denied or accepted on the requests page. User's can also now specify their favorite song on the profile page to be displayed along with their profile.

User Story #3b:

As a user, I would like to be able to export Spotify data and import it into PULSE to get the following statistics ("Advanced Statistics" using Spotify's "Extended Streaming History")

#	Description	Time	Owner

1	Parse through Spotify data text file and store it all locally	1h	Noah
2	Use this data to get the advanced statistics information needed and store it in a way that is both space-efficient and allows for easy accessing later	4.5h	Noah
3	Create and implement storage strategy + packer/unpacker for advanced statistics that will be stored in the database and delivered to the backend at later times.	3h	Bodhi
4	Create UI for uploading Spotify data	1h	Bodhi
5	Create page with information telling the user how they can get their extended streaming history from Spotify	1h	Bodhi
6	Add element to tell user if advanced stats has been downloaded & parsed by the server	0.5	Noah
7	Synchronize backend routes with frontend	0.5h	Noah
8	Call backend routes from frontend	0.5h	Bodhi
9	Automated unit testing for all advanced stats info with option to run it on a website route	2h	Noah

Completed:

The user is able to upload their advanced statistics data to PULSE in a user-friendly manner and with the option to upload multiple different files. If any file names are invalid the user is notified, and if the files all have valid data, the information needed to get their statistics is calculated and stored in the database. Any option to view their advanced data on the statistics page is then enabled properly.

User Story #3c & #3d:

As a user, I would like for number of minutes listened (#3c) and percentage of total music listened (#3d) to be tracked in my listening activity (not using API, but using Advanced Statistics from extended user streaming history)

#	Description	Time	Owner
1	Extract appropriate portions of data from advanced statistics and ensure its passed to backend in a parsable manner	0.5h	Bodhi
2	Create compatibility for graphs to display this information	0.75h	Noah
3	Synchronize frontend and backend routes	1h	Noah
4	Manual testing and documentation	1h	Noah

Completed:

Users who have uploaded their extended streaming history are able to see their number of minutes listened to and percentage of total music listened to in a user-friendly manner on the statistics page. Users who have not uploaded their extended streaming history do not have this option.

User Story #3e:

As a user, I would like for times when music was listened to (morning, afternoon, night, etc.) in my listening activity (not using API, but using Advanced Statistics from extended user streaming history)

#	Description	Time	Owner
1	Extract appropriate portions of data from advanced statistics and ensure its passed to backend in a parsable manner	0.5h	Bodhi
2	Create compatibility for graphs to display this information	0.75h	Noah
3	Synchronize frontend and backend routes	1h	Noah
4	Manual testing and documentation	1h	Noah

Completed:

Users who have uploaded their extended streaming history are able to see the times of day their music was listened to in a user-friendly manner on the statistics page. Users who have not uploaded their extended streaming history do not have this option.

User Story #3f:

As a user, I would like for times when songs were listened to fully or skipped or repeated in my listening activity (not using API, but using Advanced Statistics from extended user streaming history)

#	Description	Time	Owner
1	Extract appropriate portions of data from advanced statistics and ensure its passed to backend in a parsable manner	0.5h	Bodhi
2	Create compatibility for graphs to display this information	0.75h	Noah
3	Synchronize frontend and backend routes	1h	Noah
4	Manual testing and documentation	1h	Noah

Completed:

Users who have uploaded their extended streaming history are able to see how often their songs were skipped or repeated is visible in a user-friendly manner on the statistics page. Users who have not uploaded their extended streaming history do not have this option.

User Story #3g:

As a user, I would like to see songs, artists, genres and favorite era of music sorted by the metrics above (not using API, but using Advanced Statistics from extended user streaming history)

#	Description	Time	Owner
1	Extract appropriate portions of data from advanced statistics and ensure its passed to backend in a parsable manner	0.5h	Bodhi
2	Create compatibility for graphs to display this information	0.75h	Noah
3	Synchronize frontend and backend routes	1h	Noah
4	Create frontend user selection for favorites criteria	1h	Bodhi

5	Store user choice for what counts as a favorite in database	0.25h	Bodhi
6	Handle backend logic that determines what data gets passed to the frontend based on what qualifies as a user favorite for songs, artists, genres, era of music	1h	Noah
7	Manual testing and documentation	1h	Noah

Completed:

Users who have uploaded their extended streaming history are able to see their songs, artists, genres, and eras of music they listened to by percentage listened to, number of listens, and number of minutes in a user-friendly manner on the statistics page. Users who have not uploaded their extended streaming history do not have this option.

User Story #3p:

As a user, I would like to be able to create custom graphs with specified data and formats

#	Description	Time	Owner
1	Modify frontend/backend to allow for legend toggle when creating graph	2h	Jason
2	Modify 3 existing graphs with their respective settings for legend display	1.5h	Jason
3	Add toggle to allow users to select play music or pull up Spotify page	1.5h	Jason
4	Modify structure to allow for multiple data on graphs	2h	Jason
5	Add additional themes to display for graphs	1h	Jason
6	Add tooltip on title of graph to display graph settings	1h	Jason
7	Add option for custom axis titles w/ toggle & only	2h	Jason

	for valid graph types		
8	Modify popup validation to go from “graph” to “data” instead of the reverse	0.25h	Jason
9	Add validation for advanced stats	0.5h	Jason
10	Add option to display text data instead of graphs	1.5h	Jason
11	Add error-handling and all validation required to create a graph w/ testing	2h	Jason

Completed:

Users are able to toggle legends for compatible graphs, toggle “play music” or “go to spotify page” for image graphs, display multiple data points on one graph, use different themes and view their graph data in the graph tooltip (though that feature is bugged and incomplete). Additionally users can add their own axis titles and are unable to create invalid graphs. Also, if a user chooses to do so, they can create graphs with data displayed in text instead of in a graphical manner.

User Story #3q:

As a user, I would like to be able to have many choices of graphs to display data

#	Description	Time	Owner
1	Create Bump graphs for top songs/artists data	2h	Jason
2	Add calendar graph for data with timestamps	2h	Jason
3	Add radial bar graph for bar graph data	2h	Jason
4	Modify existing bar graph to have a horizontal bar graph equivalent	1h	Jason
5	Add scatter plot for line graph data types	2h	Jason
6	Test creation w/ all variations of inputs for graphs	1h	Jason

Completed:

Users can create bump, calendar, radial bar, horizontal bar and scatter plot graphs.

User Story #3r:

As a user, I would like to be able to view my friends' statistics as per previously mentioned user tracked statistics

#	Description	Time	Owner
1	Create backend functions to pull friend statistics from PULSE	2h	Dane
3	Modify frontend graphs to store whose data the graph belongs to	2h	Jason
4	Modify frontend graphs to allow multi-data graphs to also display multi-user data	2h	Jason
5	Test most combinations of graphs w/ friends data	1.5h	Jason

Completed:

Users can create graphs for their friend's data however testing is incomplete and should be completed.

User Story #3u & #3v:

As a user, I would like to see advanced data percentage-wise including what genres I've listened to (#3u) and what emotions I experience when listening to Spotify (#3v) (not using API, but using Advanced Statistics from extended user streaming history)

#	Description	Time	Owner
1	Extract appropriate portions of data from advanced statistics and ensure its passed to backend in a parsable manner	0.5h	Bodhi
2	Create backend to read emotions from Spotify list	2h	Dane
3	Create radar graph for displaying emotion data, and compatibility for statistics page to display emotion data	2h	Noah
4	Synchronize frontend and backend routes	1h	Noah
5	Manual testing and testing documentation	1h	Noah

Completed:

Users who have uploaded their extended streaming history are able to see what genres and emotions they felt (by track's parameters) while listening to Spotify in a

user-friendly manner on the statistics page. Users who have not uploaded their extended streaming history do not have this option.

User Story #3w:

As a user, I would like to see what emotions a song I select matches

#	Description	Time	Owner
1	Create backend function to pull current song & give emotions back	3h	Dane
2	Create compatibility for graphs to display this information	0.75h	Noah
3	Synchronize frontend and backend routes	1h	Noah
4	Manual testing and testing documentation	1h	Noah

Completed:

Users can search for their desired track on the statistics page emotion graph and get a radar display of the emotions the song matches.

User Story #4a & #4b & #4e:

As a user, I would like for recommendations for songs to add on the playlist based on genres (#4a) & artists (#4b) & albums (#4e)

#	Description	Time	Owner
1	Develop front end navigation to a playlist page and create playlist page UI	1.5h	Bodhi
2	Add search bar for playlists	0.5h	Bodhi
3	Add display for recommendations with method for adding to playlist	1.5h	Bodhi
4	Develop backend for search bar for playlists backend	0.5h	Dane
5	Develop backend for playlist iterating over songs inside a playlist	2h	Dane

6	Develop recommendation finder for playlists based of genre	1h	Dane
7	Add dropdown to choose what recommendation should be based off of	0.5h	Alex
8	Develop backend for playlist recommendations based off of artists	0.5h	Dane
9	Develop backend for playlist recommendations based off of albums	1h	Alex
10	Manual testing and testing documentation	2h	Alex

Completed:

A user is able to navigate to the playlist recommendation page when they want recommendations. The search bar automatically populates with all owned playlists and the user can click on the desired playlist to receive recommendations for. After clicking an automatically displayed playlist they receive recommendations based on genre and can switch the criteria using a drop down box. After clicking a new playlist the recommendations box automatically populates with songs using the new analytics. After clicking a recommended song the user is prompted to add the song to their playlist.

User Story #4f:

As a user, I would like for recommendations for songs or playlists based on User emotion presets (informed by Spotify API)

#	Description	Time	Owner
1	Create UI for choosing emotion	1h	Alex
2	Create emotion presets for parameters	2h	Dane
3	Update parameters to display emotion presets	0.5h	Alex
3	Create UI to display song recommendations	2h	Alex
4	Retrieve recommendations using parameters	1h	Dane
5	Manual testing and testing documentation	1h	Alex

Completed:

The user is able to navigate to the parameter recommendations page via the recommendations page, where they are shown a list of sliders each labeled with different parameters and their corresponding value. The values of these parameters can be saved along with a name as an emotion, which can later be overwritten (change values and name) and deleted. On the right of the page, there is a button which when pressed, will populate a list with songs that are retrieved from spotify based on the selected parameter values and selected genre. Requesting recommendations again will replace the old songs with new recommendations. On the bottom right of the page, a list of the user's playlists exists which the user can select and then press "Derive Emotion", which will set the parameter sliders to values based on the playlists contents.

User Story #5g:

As a user, I would like a song activity monitor that displays the amount of times a song was played in an user-friendly manner (not using API, but using Advanced Statistics from extended user streaming history)

#	Description	Time	Owner
1	Extract appropriate portions of data from advanced statistics and ensure its passed to backend in a parsable manner	0.5h	Bodhi
2	Create compatibility for graphs on statistics page to display song activity	0.75h	Noah
3	Create UI on the song page to display song activity	0.5h	Bodhi
4	Synchronize frontend and backend routes	1h	Noah
5	Manual testing and documentation	1h	Noah

Completed:

Users who have uploaded their extended streaming history are able to see their number of times their songs were played in a user-friendly manner on the statistics page. Users who have not uploaded their extended streaming history do not have this option.

User Story #4k:

As a user, I would like to be able to set my emotion parameters with user chosen playlist

#	Description	Time	Owner
1	Create UI to choose playlist and create parameter set	2h	Alex
2	Create UI to manage custom parameter sets	1h	Alex
3	Backend for custom parameter set storage and pulling parameters from playlist	2h	Alex
4	Store custom parameter set in database and create relevant retrieval and processing functions	1.5h	Bodhi
5	Manual testing and testing documentation	1h	Alex

Completed:

On the parameter recommendation page the emotion presets are in a dropdown and you can create new emotions or select other ones. You can select an owned playlist from a display box and derive an emotion from it. The parameter page automatically updates your selected emotion with the derived emotion.

User Story #4l:

As a User, I would like to manage playlists through Pulse.

#	Description	Time	Owner
1	Add navigation to playlist management page	0.5h	Alex
2	Add track adding and frontend	1h	Dane
3	Add image management and frontend	2h	Dane
4	Add playlist creation	1h	Dane
5	Add track remover/ and track replacer	1h	Dane
6	Add track reorder	1h	Dane
7	Add playlist unfollow and follow	1h	Dane

8	Manual testing and testing documentation	1h	Alex

Completed:

A user is able to navigate to the playlist management page when they want to manage their playlists. The display box automatically populates with all owned playlists and the user can click on the desired playlist to manage. After clicking an automatically displayed playlist they can then manage the playlist by removing tracks, reordering tracks, unfollowing the playlist, or changing the image. A simple search bar needs to be added to allow new playlists to be followed.

User Story #4m:

As a user, I would like to auto-generate playlists through Pulse based on genre

#	Description	Time	Owner
1	Add navigation to playlist generation page	0.5h	Alex
2	Create UI for playlist generation page	1h	Alex
3	Add playlist creation and auto-population	2h	Dane
4	Manual testing and testing documentation	1h	Alex

Completed:

A user is able to navigate to the playlist management page via the recommendations page. Here, they can find a name, genre, public, and collaborative inputs and a button to generate a playlist on the user's account with auto-populated songs.

User Story #5b:

As a user, I would like to be able to find similar songs by specifying Spotify API target parameters such as loudness to a chosen song.

#	Description	Time	Owner
1	Add sliders to specify parameters based on Spotify's major parameters to the recommendation page.	1h	Bodhi
2	Add backend functionality to parameter setting	1h	Dane

3	Add functionality to store saved preferences in the database to use on future logins	0.5h	Bodhi
4	Add functionality to grab recommendations using user chosen parameters	2h	Dane
5	Populate page with recommended songs according to the given parameters	0.5h	Bodhi
6	Manual testing and testing documentation	1h	Alex

Completed:

A user is able to navigate to the parameter recommendations page. All parameter sliders are functional and change the recommendations accordingly. Upon changing parameters the recommendations are repopulated accordingly.

User Story #6a:

As a user, I would like to play games such as Guess who listens to this song as a quiz format and heads-up format

#	Description	Time	Owner
1	Create UI for guess who listen to this song	0.5h	Rheaa
2	Find a way to merge playlists between friends (assuming friend tracking is done in the database)	2h	Rheaa
4	Editing the UI for the heads-up format	1h	Rheaa
3	Create scoring tracking	0.5h	Rheaa
4	Send scores to backend to be stored in database	1h	Rheaa
5	Add the scores of previous games to the scores history on the main games page	1h	Rheaa
6	Manual testing and testing documentation	1h	Noah

Completed:

The games guess who listens to the songs plays a song from the recently listened from one of the usernames inputted at the beginning of the game. The scores are then stored on the main games page.

User Story #11a:

As a user, I would like to interact with a PULSE chatbot through a textbox

#	Description	Time	Owner
1	Create the UI for the ChatBot	2h	Rheaa
2	Create functionality to have the send button to trigger the ChatGPT API	1.5h	Rheaa
3	On click of the Button call the ChatGPT API	0.2h	Rheaa
4	Time spent training the ChatGPT API to know specifics about our website	2h	Rheaa
5	Manual testing and documentation	1h	Noah

Completed:

Our Chatbot is accessible through the dashboard page. It uses the ChatGPT API and triggers it to respond to the user's prompts. The Chatbot was built to be user friendly and intuitive and provide valid responses to user's messages. The UI included a textbox for user input, a send button and a messages box above it that contains the messages previously send by the user and the ChatBot responses.The bot was trained to know specifics about our website such as the different sections, the different types of games, how to change specific settings and much more.

User Story #11b:

As a user, I would like to be able to get song recommendations per Genre, Artist, Location, Album and User emotion through PULSE chatbot

#	Description	Time	Owner
1	Setting up prompts for recommendations per Genre, Artist, Location,Album and User emotion through PULSE chatbot	2h	Rheaa
2	Testing and modifying prompts and setting them as the default prompts (for better user experience)	1h	Rheaa
3	Create functionality to send required data to the backend and database	1h	Rheaa

4	Manual testing and documentation	1h	Dane

Completed:

The Chatbot is able to provide recommendations based on Genre, Artist, Location, Album and User emotion.There are also some custom prompts that are built in the form of buttons over the input box including getting newly released music, getting workout music and much more.

User Story #11c:

As a user, I would like to be able to automatically populate playlists per Genre, Artist, Location, Album and User emotion through PULSE chatbot

#	Description	Time	Owner
1	Build backend functions and routes to get required data from frontend (Research required)	3h	Rheaa
2	Build backend function(s) to automatically populate playlists per Genre	1.5h	Rheaa
3	Build backend function(s) to automatically populate playlists per Artist	0.25h	Rheaa
4	Build backend function(s) to automatically populate playlists according to popular music in a specific location (country or region)	0.25h	Rheaa
5	Build backend function(s) to automatically populate playlists according to user emotion	0.25h	Rheaa
6	Manual testing and testing documentation	1h	Noah

Completed

As soon as the user asks the Chatbot for recommendations and there is more than one, a playlist is created on their spotify account with all the suggested songs. During the demo, this feature did not work completely as we switched accounts for the original testing account and discovered a bug that was fixed immediately after our sprint demo. The server restriction (as mentioned in user story #11b) applies to this user story too.

User Story #11d:

As a user, I would like to be able to get song information by inputting a short phrase from the song into PULSE chatbot

#	Description	Time	Owner
1	Implementing a ChatGPT prompt and testing to make sure it has the required capabilities	2h	Rheaa
2	Implementing the functionality to let the user play the song once found	1h	Rheaa
3	(Backup) embedding functionality similar to chosic.com into the PULSE chatbot (Research Required)	4h	Rheaa
4	Manual testing and testing documentation	1h	Bodhi

Completed:

When a user inputs partial lyrics of a song, the chatbot provides a link to the complete lyrics and immediately starts playing the song off the users spotify.

User Story #11e:

As a user, I would like to get help with finding/doing a certain action on the website by asking PULSE chatbot

#	Description	Time	Owner
1	Training the ChatGPT API to know specifics about our website features, structure, etc.	2.5h	Rheaa
2	Manually testing if the chatbot can provide all the needed assistance to the user	1h	Rheaa

Completed:

The Chatbot was trained to know specifics about our website such as the different sections, the different types of games, how to change specific settings and where each section that user may be looking for is located.

User Story #11f:

As a user, I would like to be able to give feedback to developers about requested features, bugs or general messages through PULSE chatbot

#	Description	Time	Owner
1	Training the Chatbot to identify when the user has something they want to communicate to the team, i.e, a feature, bug or general messages	1h	Rheaa
2	Creating functions to store these messages and have them sent to the project email (one of the team members personal email)	3h	Rheaa
3	Storing such messages/ interactions in the database	1h	Rheaa
4	Manual testing to ensure feedback is received by devs correctly	1h	Bodhi

Completed:

Whenever the user entered keywords that suggested they wanted to give feedback about the website the Chatbot would prompt the user to give the feedback and then send the feedback string to the database and send the user a confirmation message that the feedback was sent to the backend team.

What went poorly during sprint 2?

The biggest issue during sprint 2 was that team members got complacent coming out of sprint 1 on which we did great. This is somewhat irrational as the majority of our sprint 1 work was completed in 3 days before the review and extremely uncoordinated. Nonetheless, not enough agency was placed on getting work done early in the sprint, with many members adopting the mindset of “we did it once, so we can do it again.” This of course ended in a mad sprint to finish and connect components in the week leading up to the review, a strategy that did not work out in our favor this time as many acceptance criteria were incomplete. Tangentially related, some features were not researched enough and caught the team off guard when we realized it wasn’t possible on the second to last day of the sprint.

Not enough importance was placed on all aspects of testing. Members would routinely encounter issues when connecting components, as individual aspects were not tested in a vacuum with stubs and drivers. Similarly, manual test cases were not emphasized enough and for the most part were rushed out right before our sprint review.

Also, while solid for the first two weeks, our branch management on Github really devolved in the last week of work. Team members began to work in the wrong branches out of convenience, and the amount of merge conflicts skyrocketed. Our main branch was often not working, and it was actually completely broken an hour or two before our sprint review.

User Story #2e:

6	Create UI for friends page that accommodates any amount of friends and has an add friend button, request button, and updates asynchronously between different accounts	2h	Bodhi

Bug:

Sometimes when a user accepts an invitation on the requests tab, the change does not immediately take effect on the friends page, and the user must go back to the friends card or refresh the page.This seems to happen when a user removes a friend right before accepting the request, but functionality is not impacted.

User Story #3p:

6	Add tooltip on title of graph to display graph settings	1h	Jason

Bug:

Due to weird margins, sometimes, tooltips sometimes do not appear when hovering over the graph title. Also there could be more details added onto the tooltip for the graphs (such as their settings).

User Story #3q:

While this user story was technically completed, there are lots of limitations on what data can go on different graphs. There needs to be more data options for each of the graph options.

User Story #3r:

3	Modify frontend graphs to store whose data the graph belongs to	2h	Jason

Bug:

Friend's graphs are broken for specific use cases such as when a graph is created but the friend does not have advanced data uploaded. This breaks even if the user attempts to create data with the user's basic data.

4	Modify frontend graphs to allow multi-data graphs to also display multi-user data	2h	Jason

Bug:

Probably works as intended but needs to be tested with multiple user's data since only one user's advanced data was used and tested for this. (used for both self and friend data during testing). So may need more testing to be fully functional.

5	Test most combinations of graphs w/ friends data	1.5h	Jason

Not done:

We lacked time to properly test our data because of the lack of advanced data from friends to test with.

User Story #4l:

7	Add playlist unfollow and follow	1h	Dane

Not done:

For the following portion of this task to be applicable in a useful manner it requires a search box not originally planned for this sprint. The functionality works for both unfollow and follow, but to be implemented fully it requires a bit more work.

User Story #6a:

4	Editing the UI for the heads-up format	1h	Rheaa

Not completed:

We attempted to make the UI look different, but none of the changes to the UI made it feel completely like a head-up display, so it definitely needs some edits.

User Story #11b:

1	Setting up prompts for recommendations per Genre, Artist, Location,Album and User emotion through PULSE chatbot	2h	Rheaa

Bug:

As we do not have a server, the chatbot is not able to remember the history of

conversations which is a piece of functionality that would be ideal to have when asking the bot for recommendations.

User Story #11d:

1	Implementing a ChatGPT prompt and testing to make sure it has the required capabilities	2h	Rheaa

Bug:
The functionality isn't perfect as it uses the ChatGPT api and sometimes the bot guesses the song incorrectly. The server restriction (as mentioned in user story #11b) applies to this user story too.

User Story #11f:

2	Creating functions to store these messages and have them sent to the project email (one of the team members personal email)	3h	Rheaa

Not done:
We were not able to implement sending the emails as it requires me to implement a mail server and we attempted to explore other options but did not find an alternative.

What is our plan to improve?

One way in which our team plans to improve is to better distribute the time we work on the project. For the first two sprints we underestimated how long many of our tasks would take as well as the integration. We ended up doing way too much work the last week and didn't spend enough time the first week. We will all strive to spend enough time the first three weeks (though Thanksgiving will significantly cut into one of the weeks) to ensure that when we are integrating all of our individual tasks are done.

We also plan to decrease our underestimation of how much time each task takes and think each task out thoroughly and include all of the tasks that need to be done to complete each user story.

Lastly, we plan to thoroughly test things with stubs and drivers in order to ensure our individual tasks are done. A lot of times when we have been integrating we come across errors on both sides, whether this be frontend, backend, or database. We plan to be responsible for our own tasks and fully isolate each component while testing. We also plan to delegate specific branches to these components instead of being sloppy with what branch belongs to what feature.