Spotify PULSE

Personalized User-Learned Social Experience

Sprint 3 Retrospective

Team 18

Rheaa Sharma

Noah Stern

Dane Shaffer

Alex Polivka

Jason Zheng

Collin “Bodhi” Scott

What went well during sprint 3?

Our team experienced significant improvements during this sprint, marked by reaching what we feel is our peak level in collaboration, efficiency, and product quality. Having learned from the initial sprints, we seamlessly leveraged our collective experience and individual expertise, resulting in more refined implementations and modular code. Notably, our UI (fed by a seamless backend of course) was a high-point of the work done this sprint.

One key achievement was the smooth coordination between team members, with components being seamlessly passed between different specialists. This marked a stark contrast to previous sprints, where such transitions were often plagued by issues. The team's ability to work flawlessly together was especially evident in the reduced need for a last-minute, mad rush to meet deadlines. While a single day of crunching was required, this was a significant improvement compared to the previous four to seven days of frenzied activity we went through in other sprints.

The Thanksgiving break played a crucial role in this positive shift, as many team members voluntarily dedicated their time to work on the project during the holiday, reducing the pressure on the final days of the sprint.

Communication within the team saw substantial improvements both during and outside of meetings. Dedicated communication channels for specific components streamlined progress tracking, contributing to increased productivity. Despite some challenges in maintaining GitHub organization towards the sprint's end, the initial two weeks showcased a more organized approach with individual branches and a well-maintained main branch that was automatically deployed to the newly created heroku servers.

Meetings, a pivotal aspect of collaboration, continued to become more productive and less chaotic. Introducing virtual work meetings proved to be a game-changer once again, enhancing efficiency for all team members. Sub-meetings focused on individual components and user stories became a regular practice, contributing to the overall effectiveness of the team.

Overall, this sprint was characterized by a harmonious collaboration, efficient workflow, and a product that met or exceeded expectations. The focus on UI refinement, combined with improved communication and meeting structures, set a positive tone for our last sprint, with many of our members feeling a bit sad it's our last one.

Current Sprint Details:

User Story #1h:

Authentication and bad routing errors are handled gracefully and direct the user to an error page

#	Description	Time	Owner
1	Refactor entire backend to be able to handle errors and be consistent with routing	5h	Noah
2	Create exception handling class with custom messages and behaviors	2h	Noah
3	Implement error handling everywhere on the backend to account for all possible errors	3h	Noah
4	Create error page and redirect user to it if there was an error	1h	Rheaa
5	Add retries for authentication token before throwing error	2h	Noah
6	Manual testing	1h	Rheaa

Completed:

Errors are handled gracefully by the server. Many errors that can be avoided all together (like token expired errors) are automatically handled and fixed for the user, while other errors will direct the user to an error page displaying the unique error code and a link back to the homepage of the website.

User Story #1i:

As a user I would like to be able to actually visit the public website, and I would like the load times to be as quick as possible.

#	Description	Time	Owner
1	Research and implement server and set up everything short of updating previous code to be inline with server framework	8h	Bodhi
2	Build framework to ensure that server size can not grow exponentially due to any input, at great monetary costs	3h	Bodhi

3	Test every facet of the framework to ensure that server size can not grow exponentially at great monetary costs.	2h	Bodhi
4	Refactor database for use in tandem with the server. Make current database calls more modular, decreasing black box effect with regards to the backend and server.	4h	Bodhi
5	Refactor backend routes to be inline with server and updated database framework	4h	Noah

Completed:

The website is now hosted on a heroku server, that anyone is able to access (but Spotify limits our linked accounts to 20 until we apply for a dev package). The server is technically only running our backend, but it serves up frontend pages as needed, and the backend routes are not actually accessible by the user. Automatic checks have been put in place to ensure that we do not go over our server or database storage limit, and the accessible website runs at an acceptable speed.

User Story #1j:

As a user, I would like my data to be secure in all ways, at all times.

#	Description	Time	Owner
1	Research possible encryption methods and decide which to use for database/server	1h	Bodhi
2	Ensure Spotify information, analytics, and personal information in the database is securely encrypted	1.5h	Bodhi
3	Ensure Spotify information, analytics, and personal information on the server is securely encrypted	1.5h	Bodhi
4	Use privilege controls and sanitation to ensure the database is not susceptible to sql injections, refactoring where necessary.	3h	Bodhi
5	Protect against other common exploits such as cross-site-scripting and buffer overflow	1.5h	Bodhi

Completed:

All token's are encrypted in the database using an AES encryption method. Beyond that, no information in the database is sensitive, and therefore not encrypted.

The database is secured with necessary sanitation procedures, and is accessed using environmental variables stored server-side. Even if an attacker was to gain complete access to the database, they would only have permission to update columns 1 by 1, and couldn't get any sensitive information. The cookie regulations on the server are also strict, securing it against common exploits such as cross-site scripting.

User Story #1k:

As a user, I would like for songs, albums, and artists, shows, and episodes to be displayed consistently across the website.

#	Description	Time	Owner
1	Create UI for individual song, album, artist, show, and episode displays	2h	Alex
2	Create UI for lists	1h	Alex
3	Implement display onto previously existing pages	2h	Alex
4	Create backend for retrieving songs, albums, and artists needed for display	1h	Dane
6	Manual Testing	1h	Rheaa

Completed:

This requirement was fulfilled in a way that made implementation of many other features much easier. The ItemList module was passed parameters for data to be displayed, action to occur on mouse click of an item, selected item index, and a list of buttons to be displayed next to each item, including their respective actions. This was used in many features implemented this sprint, and extended to features from previous sprints. Songs, albums, artists, playlists, episodes, and shows all were displayed with their name, and respective information when applicable. These lists were intractable, usually through selection or starting playback.

User Story #2h:

As a user, I would like to be able to customize my public profile.

#	Description	Time	Owner
1	Add UI on profile page to edit profile page	1h	Alex
2	Create modular profile display component	1h	Alex
3	Add new information fields in database	0.5h	Bodhi

4	Create backend to store and retrieve new information fields	1h	Dane
5	Manual Testing	1h	Dane

Completed:

The status field, public display background color, and public display text color were added to the profile settings, and were set to be displayed on friend's information cards. The profile page was also reformatted to display a preview of the public friend card. A remaining bug is that in some instances, the status will display "Not implemented in backend", as a remaining debugging message.

User Story #2i:

As a user, I would like backgrounds to be saved along with themes.

#	Description	Time	Owner
1	Create UI to combine background and themes	2h	Alex
2	Edit database structure to store backgrounds with themes	1h	Bodhi
3	Edit backend functions to store backgrounds with themes	1h	Dane
4	Edit frontend to retrieve backgrounds and themes together	1h	Alex
5	Manual testing	1h	Rheaa

Completed:

This user story was completed, with themes saving backgrounds along with color palettes. The settings page UI was modified to reflect this change to the user. Additionally, the save changes button was moved to the top of the settings to be more obvious.

User Story #3k:

As a user, I would like to be able to specify what time period I want my stats to be from.

#	Description	Time	Owner
1	Allow advanced stats to be displayed in a time range	2h	Jason

2	Modify popup for above	1h	Jason
3	Allow followers to be displayed in a time range	1h	Jason
4	Ensure advanced statistics are stored correctly	1h	Bodhi

Completed:

This requirement was fulfilled as far as possible. Follower data were displayed properly but advanced stats could only be displayed in a month-by-month range due to how the server-side handling and formatting of the data worked. Ideally we could ask the server to regenerate advanced data based on a certain time range but due to how the algorithm works, this is unfortunately not feasible.

User Story #3I:

As a user, I would like to be able to view statistics in multiple different graphs or charts

#	Description	Time	Owner
1	Separate data formatting functions into a separate file that can be called for in a easy manner	3h	Jason
2	Create emotion radar graph for top songs	1.5h	Jason
3	Create pie graph formatting for data	1h	Jason
4	Format data for bar graphs	2h	Jason
5	Add option to resize grid layout	1h	Jason
6	Edit page layout to match rest of website	1h	Alex
7	Fix filtering of song data for their filter menu with sorted by top song metric & an option to display certain indexes (ie top 10-20 songs)	1.5h	Jason
8	Fix bug for graph state not being saved on “save graphs” and not being loaded	3h	Jason
9	Add/fix handling for exceptional cases such as no friends, no friend’s data, no advanced data, etc.	2h	Jason
10	Test graph creation for all variations of popup & debug remaining bugs	5h	Jason

11	Redesign UI to be more user-friendly and visually pleasing	3h	Rheaa

Completed:

All of these tasks were completed though data formatting functions weren't refactored enough in my opinion. They work and reduce the need for redundant functions for each graph type but as with everything, a lot more optimization could go into them. In addition, the UI was modified to be better but could still use a lot of work.

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	Fix race condition w/ friends data bug	1h	Jason
2	Allow for display of friends' data on graph	4h	Jason
3	Edit page layout to match rest of website	1h	Alex
4	Test display of friend data	1h	Jason

Completed:

Unfortunately due to limitations on time, multi-friend data display on one graph was removed. The documents were adjusted to reflect this change and we did get friend's data working. Just not displaying multiple friend data on one graph.

User Story #3x:

As a user, I would like to be able to preview graphs before creating them

#	Description	Time	Owner
1	Reformat popup to display in a better user-friendly format	2h	Jason
2	Allow for preview of graphs on the popup	1.5h	Jason
3	Stop display on popup for elements that cannot be modified	1h	Jason

Completed:

Popup is able to create a preview graph though only populated with sample data. It also stops display for elements that can not be modified so instead of graying out fields and looking somewhat odd, it has a better look now.

User Story #4j & #4i:

As a user, I would like to choose two playlists and have a new playlist made from the Spotify-recommended songs of both playlists or merge them.

#	Description	Time	Owner
1	Add frontend on playlist manager to search playlists and select 2 playlists to either merge or combine recommendations	3h	Alex
2	Add backend to search playlists	1h	Dane
3	Add backend to pull recommendations from two playlists and create a playlist from both	2h	Dane
4	Create backend to merge two playlists by inserting tracks in alternating order	2h	Dane
5	Manual testing	1h	Dane

Completed:

The playlist manager page was edited to have different tabs for different sections. This included a “synthesize” tab, which allowed two playlists to be selected and then be merged or fused, at which point the playlist list would be refreshed. Text was used to display the status of the attempted merge or fuse.

User Story #4n:

As a user, I would like my emotion presets to persist between logins

#	Description	Time	Owner
1	Edit emotion preset frontend to reflect storage changes	1h	Alex
2	Update database to store emotion presets and pull emotion presets	2h	Bodhi
3	Edit backend to allow for storage and retrieval	2h	Dane
4	Manual testing	1h	Dane

Completed:

This user story was completed, with the emotions stored in the database and retrieved upon the page loading. Emotions are automatically saved whenever a new emotion is created.

User Story #4o:

As a user, I would like my playlist manager to dynamically display my playlist data after it is changed

#	Description	Time	Owner
1	Edit frontend to call for updates upon actions	1h	Alex
2	Edit backend to support simple updates edit	1h	Dane
3	Manual testing	1h	Rheaa

Completed:

The ItemList modules associated with the playlist manager page automatically refreshed, displaying “Loading…” while the data was being retrieved. This refresh occurred anytime an action potentially affecting a list of items occurred.

User Story #4p:

As a user, I would like to choose at what portion of my playlist is reordered

#	Description	Time	Owner
1	Add frontend index selection	2h	Alex
2	Edit backend to allow index to be dynamic	2h	Dane
3	Manual testing	1h	Dane

Completed:

Three fields were added to the UI to select which indexes to use for reordering a playlist, along with the buttons to reorder a selected playlist. This was placed into the “Edit” category of the newly formatted playlist manager page.

User Story #6d:

As a user, I would like to see my game scores in a presentable UI

#	Description	Time	Owner
1	Verify game scores are still being stored correctly	1h	Rheaa
2	Show game scores in a coherent format	2h	Rheaa

3	Edit game page layout to match rest of website	1h	Alex
4	Manual Testing	1h	Rheaa

Completed:

The games' scores are now visible in a coherent format that makes it more intuitive for the user to know their past scores of every game.

User Story #9c & #9d & #9e:

As a user, I would like to transfer Spotify playback to a different Spotify device on PULSE, control the queue, and change my Spotify playback.

#	Description	Time	Owner
1	Create expanded playback page	2h	Alex
2	Display information about currently playing information	2h	Alex
3	Create backend function to pull spotify devices	1h	Dane
4	Create backend function to change spotify device	1h	Dane
5	Create backend function to pull queue	1h	Dane
6	Create backend function to update queue	1h	Dane
7	Create search bar for songs, artists, albums, playlists, episodes, or shows	1h	Alex
8	Create backend function to change spotify playback based off criteria	2h	Dane
9	Manual testing	1h	Rheaa

Completed:

On any page that has a playback bar, there is an up arrow button that will bring up the expanded player page. On this page, current playback is displayed (if any), along with a queue/search list. On the right, information about the current playback (if any) is also displayed, including a parameter list pulled from Spotify API. Many actions are set to additionally trigger resync of the playback, which unfortunately only successfully pulls song information if the player is currently playing. Another limitation is that some actions (change playback, pause) do not work if a podcast episode is playing. Another feature added is the ability for the playback to sync based on external triggers passed from

parent components. For example, if playback is started from an ItemList, then the playback bar will automatically sync.

User Story #12a:

As a user, I would like to get ‘matches’ on songs in a user-friend dating-like manner

#	Description	Time	Owner
1	Create navigation to song matchmaking page	0.25h	Rheaa
2	Create basic card component for matchmaking page	2h	Rheaa
3	Create left and right buttons for swiping, create a button to view all the songs you swiped right on	1h	Rheaa
4	Create animation for left and right swiping and additional animations	2h	Rheaa
5	Getting the songs from the backend functions and loading the image, name and artist onto the card	2h	Rheaa
6	Edit page layout to match rest of website	1h	Alex
7	Create backend functions to get image, name, artist, and genre of a song	1h	Noah
8	Algorithmically recommend user a song based on previous selections by dynamically updating and storing their preferences	4h	Noah
9	Write route to display all of the user’s liked songs and add them to playlists as well as other options	3h	Noah
10	Store and retrieve song match data and recommendation parameters from database	0.5h	Bodhi
11	Manual Testing	1h	Rheaa

Completed:

Users can match with other users based on their most listened to genres. They can either swipe right or swipe left on a given user, view their list of users they swiped right on, and easily send friend requests to those users.

User Story #12b:

As a user, I would like to get matched to other users based on their song preferences.

#	Description	Time	Owner
1	Build the toggle between user and songs with functionality	1h	Rheaa
2	Create new card component for user : with user image and favorite song (and maybe top artist and song)	1.5h	Rheaa
3	Animation for swiping	1h	Rheaa
4	Edit page layout to match rest of website	1h	Alex
5	Create a button to view all the people you swiped right on	0.25h	Rheaa
6	Connect with the backend functions	2h	Rheaa
7	Create algorithm to filter through users and display the ones with a closest ‘match’	3h	Noah
8	Create criteria for a user to match with another user with and without advanced stats	3h	Noah
9	Store and retrieve user match data from database	0.5h	Bodhi
10	Manual Testing	1h	Rheaa

Completed:

Users can match with tracks based on their listening preferences. They can either swipe right or swipe left on a given track, view their list of tracks they swiped right on, and easily add those tracks to one of their playlists. Additionally, their listening preferences are dynamically updated every time they swipe, biasing their preferences towards tracks they swiped right on and away from tracks they swiped left on.

User Story #13a-g:

As a user, I would like to search for new artists and follow them, unfollow artists, see an artist’s related artists, see an artist’s top songs by country, see what genres an artist is associated with, be able to inspect what artists I follow, see the audio features of a song.

#	Description	Time	Owner
1	Add information page for artists	2h	Alex
2	Add information page for songs	1h	Alex
3	Add search bar for songs and artists	1h	Alex
5	Edit page layout to match rest of website	1h	Alex
6	Create backend to follow and unfollow artists	2h	Dane
7	Create backend to see an artists related artists	1h	Dane
8	Create backend to get top songs for an artist	1h	Dane
9	Create backend to pull genres	1h	Dane
10	Create backend to see audio features of a song	1h	Dane
11	Create backend to pull followed artists	1h	Dane
12	Manual Testing	1h	Rheaa

Completed:

Users can navigate to the artist explorer page and search for artists. You can automatically see all of the data pertaining to an artist and follow and unfollow them from this page. The list of artists is auto populated with a user's followed artists. The song info for viewing audio features is under the expanded player and shows in the right info box when a song is playing.

What went poorly during sprint 3?

The third sprint posed significant challenges for our team (mostly caused by deploying to a server), and it's crucial to reflect on the shortcomings to learn and improve for our last couple of days of work before the final presentation.

The sprint 1 decision to have the frontend and backend running in tandem caused unforeseen problems during deployment to Heroku servers. We had to do a workaround of having a backend handling everything and serving frontend files when necessary. However, this approach proved to be cumbersome, leading to a major headache for the team.

Because of this fix, routing became a significant pain point. The added complexity resulted in difficulties ensuring smooth navigation and communication between different parts of the system. Addressing routing issues took up valuable time that could have been better utilized for feature development and UI work.

Another notable challenge was the lack of well-defined environmental variables for both production and development until the final week. This forced team members to constantly edit files locally, causing confusion and extra work. The delayed resolution of this issue impacted the team's efficiency and caused unnecessary friction leading up to the fix.

A crucial oversight was not focusing on making the code compatible with a server from the outset. This lack of foresight led to rewriting significant portions of the API to handle response codes and specific errors, not to mention everything described above. This avoidable rework had a substantial impact on the overall progress and quality of our deliverables.

Lastly, similar to the previous sprint, our branch management on GitHub experienced regression. In the last week, team members began working in the wrong branches for convenience, resulting in a surge of merge conflicts. The main branch was, at times, completely broken just before crucial milestones like the sprint review.

User Story #3k:

1	Allow advanced stats to be displayed in a time range	2h	Jason

Incomplete:

Due to how the function to generate advanced data works, it would take the equivalent amount of time to initially set up the data to generate advanced data based on a time frame. Thus, we only have month-to-month time range formatting instead unfortunately.

User Story #3x:

2	Allow for preview of graphs on the popup	1.5h	Jason

Incomplete:

Due to complexities with the current system, we decided to forego display of the user's data on the graph and instead opt for the easier solution of displaying sample data instead. This cuts down on the usefulness of the graph but does make viewing how an ideal graph would look like so while it is worse in some ways, it also has its benefits.

User Story #6d:

1	Verify game scores are still being stored correctly	1h	Rheaa

Incomplete:

The games' scores in our case are being scored correctly but during our demo one of the games "HeadsUp" did not function as required and therefore could not verify that those scores were being stored completely.

3	Edit game page layout to match rest of website	1h	Alex

Incomplete:

Even though the background was completed there were some minor changes that were incomplete, in terms of the styling being applied to each individual game.

What is our plan to improve?

Having experienced the complexities associated with the backend-frontend tandem deployment, our team has fixed them to the best of our abilities, meaning future work can be more focused on actual feature work and bug fixing. The decision to have a unified backend handling everything will be revisited, and adjustments will be made to simplify the deployment process. All new frontend components and API routes will be built with our server set up in mind, making development smoother over all.

Our focus before the final presentation is on bug fixes and UI improvement. Team members have kept a comprehensive list of bugs over the past 3 sprints, and now is the time we get to go back and fix them.