Spotify PULSE

Personalized User-Learned Social Experience

Sprint 1 Retrospective

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

What went well during sprint 1?

Everyone on our team became an expert in something new over the course of this sprint. A deep understanding of individual components was gained, from Flask, to React, to MySQL.

We set up a good framework for each new technology, one that will be easy to build off later. Our database connections and API calls are well organized, and easy to iterate.

We built in most of the basic functionality of individual components of the site. If functionality is not there, care was taken to ensure that functionality would be easy to integrate once completed locally.

We worked well as a team in all regards. Our workflow was more organic than expected, with team mates swapping tasks as our expertise dictated. This allowed for certain tasks to get completed quicker than if the original team mate kept their task.

Our UI looks beautiful and work's responsively. The ability to create a custom layout for the statistics page is really intuitive, and our games are mostly complete.

Current Sprint Details:

User Story #1c:

As a user, I would like to be able to register for a PULSE account using Spotify.

#	Description	Time	Owner
1	Create basic class structure with server-side functions and data storage	4h	Noah
2	Create user class and needed functions	3h	Noah
3	Create database to store data using Microsoft Azure and format user info schema	6h	Bodhi
4	Create component for side login and login button that allows for user to go to Spotify Login	3.5h	Rheaa
5	Create UI for welcome page	2h	Bodhi
6	Get access token from Spotify API	1.5h	Dane

7	Populate account fields	2h	Dane
8	Refresh access token when needed	1.5h	Dane

Completed:

When the user attempts to access PULSE, they see the initial welcome page and are able to login with their spotify account. If they have a valid spotify account, they are taken to the PULSE page and view their data. Additionally, user cookies are stored so if a user logs in, closes the tab, and accesses the page again in a short time span, then they are automatically logged in to their account.

User Story #1d:

As a user, I would like to be able to login and manage my PULSE account.

#	Description	Time	Owner
1	Create UI for profile page which includes a setting subsection	4h	Alex
2	Connect UI to database to update usernames and icons automatically.	2h	Dane
3	Store user info & settings in database	0.5h	Bodhi
4	Dynamically compress icons to target size for display	2h	Bodhi
5	Create icon database schema and store icons	1h	Bodhi
6	Create UI for user icon and username on login	2h	Alex

Completed:

When users log in for the first time they are greeted with a page where they can change their display name along with uploading an icon in either png or jpg format. Their display name defaults to their spotify display name, and they are able to choose to not edit it and not upload an icon. Later, if the user wants to upload/update their icon, they can do so in the setting page. Their updated icons and usernames persist between sessions, and icons are dynamically compressed to an appropriate size before being stored. The database schema for user info and settings is also finalized, but we decided to store icons server side instead of in the database.

User Story #1g:

As a user, I would like to access all other pages from a home page.

#	Description	Time	Owner
1	Create UI for top tab	2h	Rheaa
2	Create components for 4 sections	2h	Rheaa
3	Create component for the friends section	1h	Rheaa
4	Implement buttons to go to different pages	1h	Rheaa
5	Add icon and link the profile section	1h	Rheaa
6	Create buttons and link all the game pages	0.5h	Rheaa

Completed:

When a user logs in with spotify and has seen the username/icon set up page, then they see the homepage of the web app. The games pane is working, and takes the user to the game page. Likewise, the profile page works the same, and the friends page is visible on the right side of the screen. All other placeholder panes are visible, and the tabs page is accessible from the top bar.

User Story #2a:

As a user, I would like to have dark/light mode.

#	Description	Time	Owner
1	Set up data colors for light and dark mode in a file	4h	Rheaa
2	Set up a component that wraps all the components in the view that the modes apply to	1h	Rheaa
3	Insert UI toggle between light/dark mode	0.5h	Rheaa

Completed:

When the user creates their account, the webpage is automatically displayed in dark mode. They are able to toggle to light mode, and the mode switch is immediately applied to all relevant components of the page. The user is able to switch between the two modes, with all relevant components of the page updating as needed. If the user logs out or leaves the page, their preference is saved in the database and automatically loaded into the front end.

User Story #2b:

As a user, I would like to be able to change text size

#	Description	Time	Owner
1	Create UI for changing text size	2h	Alex
2	Update text size in the database	0.25h	Alex
3	Modify frontend code to utilize text size from database	2h	Alex

Completed:

The user is able to change their text size in the settings page. Changing the text size immediately applies the change to the settings page, as well as any other pages the user may then navigate to. The user's preferred text size is saved in the database so that it persists between sessions.

User Story #2f:

As a user, I would like to edit my username

#	Description	Time	Owner
1	Add a button to edit the username	0.25h	Dane
2	Add a button to finalize changes	0.25h	Dane
3	Edit the username on the backend and database	0.25h	Dane

Completed:

The user is able to change their username (the name displayed to them and friends). This change is stored in the database, and their updated username is used from that point on. The user must finalize the changes after editing, and they may change their username as many times as they would like.

User Story #2j:

As a user, I would like to modify my profile information such as gender, profile picture and location.

#	Description	Time	Owner
1	Modify profile UI to show gender and location	1h	Dane
2	Modify UI to allow editing of gender, profile picture and location	1h	Dane
3	Update database to store new data	0.25h	Dane

Completed:

The user is able to change their preferred gender, location, and icon. This change is stored in the database, and their updated info is used from that point on. The user must finalize the changes after editing, and they may change each field as many times as they would like.

User Story #3a:

As a user, I would like for my top songs and artists for the last 4 weeks, 6 months and all time to be tracked.

#	Description	Time	Owner
1	Pull top songs and artists for the last 4 weeks, 6 months and all time from Spotify API	2h	Noah
2	Store data in the database	0.25h	Noah
3	Pull song and artist images from Spotify for display on custom component	1h	Bodhi
5	Create custom component to display top song/artist images on the statistics dump page	4h	Bodhi
6	Allow songs or artist songs to be played after being clicked on	1h	Dane

Completed:

If a user is able to successfully log in to PULSE, then they have a valid spotify access token. Their top songs and artists for the last 4 weeks, 6 months and all time are pulled from Spotify API. The user is then able to add a component to the statistics dump page that shows these stats along with the images associated with each artist/song/album. Clicking on a song will play a snippet of it.

User Story #3h:

As a user, I would like to track my follower numbers.

#	Description	Time	Owner
1	Pull followers from Spotify API	0.5h	Dane
2	Store data in the database	0.25h	Dane

3	Display data on statistics dump page	0.25h	Dane
4	Store history in database	1.5h	Dane
5	Allow history to be accessed and pulled by time	0.5h	Dane

Completed:

Follower numbers are stored in a dictionary of datetime:followers every time the user accesses the statistic dump page. This dictionary is stored in the database, and updated as needed. It can be displayed on the statistics page as a graph of follower number over time.

User Story #3i:

As a user, I would like for my most recently listened to songs to be displayed.

#	Description	Time	Owner
1	Pull recently listened to songs from Spotify API	1h	Noah
2	Store data in the database	0.25h	Noah
3	Display data on statistics dump page	1h	Alex
4	Allow songs to be clicked on and play	0.25h	Alex

Completed:

Recently listened songs are pulled from the spotify API when the user logs in. The user can then choose to display a custom component that shows their recently listened songs, and can click on a song to play a snippet of it. We decided to not store the recently listened songs in the database, as it is just reloaded every time a user logs in so it doesn't make sense to store it every time. We did however, create recent song storage capabilities before realizing we weren't going to store it, so either way we view it as a completed task.

User Story #3j:

As a user, I would like for my saved songs, saved albums, and followed artists to be displayed.

#	Description	Time	Owner
1	Pull liked songs, playlists and albums from Spotify	1h	Noah

2	Store data in the database	0.25h	Dane
3	Display data on statistics dump page	1.5h	Dane
4	Allow items to be clicked on and played	0.25h	Dane

Completed:

Liked songs, playlists, and albums are pulled from Spotify using their API. They can then be tracked on the statistics dump page after inserting a component, and each can be played by clicking on them. Clicking on an artist takes the user to the spotify artist page. We decided to not store this data in the database, as it is just reloaded every time a user logs in so it doesn't make sense to store it every time. We did however, create storage capabilities before realizing we weren't going to store it, so either way we view it as a completed task.

User Story #3l:

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

#	Description	Time	Owner
1	Create UI frame for statistics graph dump	3h	Jason
2	Create graph graphics (pie chart, line chart, etc.)	3h	Jason
3	Make graphs responsive & test to ensure responsiveness	2h	Jason
4	Create preset themes for graphs	4h	Jason
5	Generate UI to allow user to select graph type and data to display	3h	Jason

Completed:

The user is able to navigate to the statistics page from the home page. A basic blank UI is displayed if the user has never customized their page. The user is able to add graph components then select the graph type and what data to display. They can customize each graph with preset themes if desired, and then choose to generate the graph and place the component in their statistics page.

User Story #3m:

As a user, I would like to be able to create custom layouts for the statistics page

#	Description	Time	Owner

1	Implement grid layout to place elements on	4h	Jason
2	Allow elements to be dynamically added and removed	3h	Jason
3	Allow elements to be dynamically resized	2h	Jason
4	Allow graph elements to be moved around the page by the user	4h	Jason
5	Make grid responsive & test to ensure responsiveness	2h	Jason
6	Format and store grid layout, element and element size in database	2h	Rheaa
7	Implement ability to save to and load multiple different graph layouts	4h	Rheaa

Completed:

Once the user is on the statistics page, they are able to add custom components such as favorite songs for different time periods and various graphs. The user is able to specify the parameters for each object in a pop up that occurs when attempting to add an element. The page is responsive, and these layouts and custom graphs can be saved in the database and then loaded back into the page later if the user wishes. Components snap to a grid, and can be resized as needed. A default layout is automatically loaded on a user's first time accessing the statistics page.

User Story #3s:

As a user, I would like to be able to view my playlists on the website

#	Description	Time	Owner
1	Pull playlists from Spotify API	1h	Noah
2	Display images on custom component for statistics dump page	1h	Bodhi
3	Play playlists when clicked on	0.25h	Bodhi

Completed:

A playlist component exists in the statistics page. It displays all of the user’s playlists.Our user story says the playlist plays on click, but logically that doesn’t make much sense so we made the choice to take the user to the playlist in Spotify when one is clicked on. This playlist component can be resized and moved, like every other

component on the stats page, and it is part of the default layout that is visible the first time a user enters the statistics page or if no layout is saved.

User Story #5a:

As a user, I would like to be able to find similar songs (based on Spotify API parameters such as loudness) to the current song

#	Description	Time	Owner
1	Develop UI for DJ Mixer page	4h	Alex
2	Develop UI for song recommender	2h	Alex
3	Get recommendation parameters from song	1h	Dane
4	Get recommended songs from Spotify API	1h	Dane
5	Display song titles and images	1h	Dane
6	Play recommendations in player when clicked	0.25h	Dane

Completed:

The user is able to navigate to the DJ mixer page which has the nav bar and friends tab as well as two buttons for the song and artist recommendation pages. The user is currently able to navigate to the song recommendation page using the button. In the song recommendation page, the user is able to search for a song and view similar songs and their associated art. Clicking on a song will play it in the spotify app if the spotify app is open.

User Story #6a:

As a user, I would like to play the game “Guess the Song/Artist” as a quiz format and heads-up format

#	Description	Time	Owner
1	Create classes and/or data structures to store the players’ scores and statistics for the game	3h	Noah
2	Create UI for title screen of game and settings	4h	Rheaa
3	Create & store settings options for game configuration	3h	Noah
4	Build skeleton of game backend	4h	Noah

5	Create UI for game states including the screens for each round and the final score page	4h	Rheaa
6	Randomly generate song from given category and other parameters in game settings and playback via Spotify music player	4h	Noah
7	Create prompt to display past scores	2h	Noah

Completed:

The user is able to navigate to the game pane, and view the available games. If they click on “guess the song/artist”, they are able to play the game. If they have played the game previously then up to 10 past scores are displayed. We decided to not save the user’s last settings, because in practice it became annoying if the options weren’t standardized to 1, and it's easy enough to click a couple of times to change the settings. Guess the song draws from a user's saved/liked songs, and the guess the artist game allows you to search for an artist to draw the songs from. The user can go through multiple rounds, indicating which players guessed correctly, and scores are saved accordingly. The user is able to quit the game by pressing the home button, and the game moves on in an intuitive and smooth fashion.

User Story #6c:

As a user, I would like to play the game “Guess the Next Lyric” as a quiz format and heads-up format

#	Description	Time	Owner
1	Create classes and/or data structures to store the players’ scores and statistics for the game	1h	Bodhi
2	Create UI for title screen of game and settings	2h	Alex
3	Create & store settings options for game configuration	2h	Bodhi
4	Build skeleton of game backend	2h	Bodhi
5	Create UI for game states including the screens for each round and the final score page	1.5h	Alex
6	Playback song sample	0.25h	Bodhi
7	Create prompt to display past scores	1h	Bodhi

Completed:

The user is able to navigate to the game pane, and view the available games. If they click on guess the next lyric, they are able to play the game. If they have played the game previously then up to 10 past scores are displayed. We decided to not save the user's last settings, because in practice it became annoying if the options weren't standardized to 1, and it's easy enough to click a couple of times to change the settings. The game allows the user to search for an artist to draw the songs from. The user can go through multiple rounds, indicating which players guessed correctly, and scores are saved accordingly. The user is able to quit the game by pressing the home button, and the game moves on in an intuitive and smooth fashion.

User Story #7a:

As a user, I would like to be able to search for songs, albums, artists, playlists.

#	Description	Time	Owner
1	Add functionality for song search bar and query API with searched for song and artist	2h	Noah
2	Add component for dropdown menu and filter by song, album, artist, playlist	1h	Bodhi
3	Display images on custom component for searched song dropdown	2h	Bodhi

Completed:

The song search bar and query functionality is housed in the song recommendation page. The user is able to search for songs, and see all results with their associated icons. They can click on a song to play it, and if the user changes their search then the results are automatically updated.

User Story #8b:

As a user, I would like to be able to control Spotify playback on the PULSE website

#	Description	Time	Owner
1	Get access to the user's song player	2h	Dane
2	Create detailed song player UI	4h	Alex
3	Place a compact song player on all pages	2h	Alex
4	Add functionality to pause/play songs	1h	Dane

5	Add functionality to skip to next/previous songs	1h	Dane
6	Add functionality to set repeat mode and playback shuffle	1h	Dane
7	Add functionality to adjust volume and skip to specific part of a track	2h	Dane
8	Add functionality to view and add items to queue	2h	Dane
9	Add functionalities to compact player	1.25h	Dane

Completed:

An omnipresent playback bar is visible on the PULSE site. The user is able to use it to pause/play, shuffle, repeat, skip or backup, scrub, and adjust the volume of music currently playing in their spotify app.

What went poorly during sprint 1?

The first week and a half went very smoothly. Everyone was on top of their tasks, and working on prototyping and setting up various aspects of our site such as the database and backend/frontend aspects. Everyone felt productive, and work was getting done at a rapid rate. However, near the end of week two, we realized that the majority of the work we had done didn't actually correspond directly to user stories or acceptance criteria such as setting up the database sanitization and permissions and creating the backend API framework. Furthemore, the concrete parts that did work were only working locally, and nothing was connected.

We began to work on connecting our portions together at the beginning of week three and ran into significant issues. Unfamiliarity with each other's code turned most issues into blackbox situations where an error would occur and only one person would have the know-how to fix it. We regrouped and took the time to understand all portions of the code, but the connection process was still slow, painful, and worst of all not reflected in our sprint one planning times or user stories. The time needed by the whole team to connect components took away from the time we had to actually work on said components, and by the end of week three it turned into a mad sprint in which we met every day for 8+ hours a day.

Such issues can partially be attributed to our initial timing estimates. Not only were some too small, but connection and set up times were not included at all. This led to each teammate working for more than 30 hours. Additionally we chose a wide swath of user stories to complete, meaning that the knowledge/expertise needed to complete

them was not focused and a new set of skills would need to be built up on each story. This was compounded by our assigning multiple teammates at a time to each user story, significantly increasing the communication needed which slowed the whole process down.

Another issue was our unproductive meetings. Up until our last couple of meetings in week three, our meetings were unproductive as a whole. Team mates would bounce from task to task, pulling each other away from work if a question came up. This rapid shifting of mindset was not productive towards our goals, and stopped us from getting done the work we set out to do. This was once again compounded by the fact our user stories had multiple people assigned to each. In order to be productive on a task, a single team mate would have to collaborate with other teammates who were most likely working on other tasks.

User Story #1d:

5	Create icon database schema and store icons	1h	Bodhi
6	Create UI for user icon and username on login	2h	Alex

Scrapped/changed:

Icons are not actually stored in the database. Instead we decided to store them on a server, and store file paths in the database. Additionally, as mentioned above, there is no popup for the user to edit their profile. The user simply goes to their settings and can edit them at any time. Also the intended goal was to be able to upload images for the images but instead, due to how the server was set up, images could not be easily uploaded without major changes. Thus, we abandoned this approach and instead got users to pass a link and then have the server download the image.

User Story #7a:

2	Add component for dropdown menu and filter by song, album, artist, playlist	1h	Bodhi

Not Completed:

Multiple backend functions to search for song/album/artist/playlist were created, but a filter tool was not completed. This was mostly an issue of never having an acceptance criteria for this task and thus we forgot to implement the functionality. We had the capability to do so but never remembered to do so.

User Story #8b:

2	Create detailed song player UI	4h	Alex
3	Place a compact song player on all pages	2h	Alex
4	Add functionality to pause/play songs	1h	Dane

8	Add functionality to view and add items to queue	2h	Dane

Changed/bugged/not completed:

The compact song player was not placed on every page. After thinking about our functionality and flow, we decided not to place it on pages such as the game page and DJ mixer. Additionally, there is a minor bug for the pause/play songs feature where if the user pauses a song in the spotify app while it is playing, the player pause icon does not update correctly. Additionally, the queue was not implemented. Along with this, we realized it didn't make sense to have a specific player page with the framework functionality so we scrapped the UI.

All user stories:

Needs polish:

Most of our UIs except for the homepage and login page are somewhat incomplete. They have complete functionality, and features are broadly where I want them, but a major focus in the upcoming sprints should be on polishing our UI and making it look as professional and act as responsive as possible.

Also the pulling of data from the server was quite slow and seemed to be dependent on what computer was running the server/client. This should be improved in future by the implementation of a server that can remove that reliance on hardware speed.

What is our plan to improve?

There are a couple of ways in which our group plans to improve, with the main way being how we handle preparation for meetings, and the actual meetings themselves. Team members need to be cognizant of what the purpose of meetings is, to ensure that they are adequately prepared. For example, many members came to the meeting in which we planned to connect components with code that was not actually ready to be connected. To combat this, everytime a meeting is scheduled, a reminder will be sent out to each member with the purpose of the meeting, and everything that is

planned to be completed. This purpose statement and agenda will help all group members not only be prepared for the meeting, but stay on task during the meeting.

Additionally, we will decrease the amount of people put on a single user story, and work to have defined sets of people that work on similar/the same user stories together so that sub groups can be created and everyone has a support net for questions. This will decrease the confusion that stemmed from having up to 4 people on the same user story. For example, if a user story needs different sets of expertise and calls for multiple members to work on it, then the members assigned to it will also be assigned to user stories with similar scopes, and we will try to pair up those two members across the sprint when teamwork is necessary.

We will also put more thoughts into the time estimates for sprint 2 and 3. This is especially important because test cases are a part of the next two sprints. Our time estimates were often way off the mark due to our unfamiliarity with many of the systems needed, and after a whole sprint we are better positioned to not only create more accurate time estimates, but better account for unforeseen issues that lead to inflation of these estimates. Connection database->backend->frontend was not even included in our estimate as we didn't fully comprehend the time needed to do so. Bigger architecture tasks such as connecting all parts need to be a part of our estimates in our future.