Groupify System and Unit Test Report

The Groupify Development Team

Updated as of 5/31/2021

System Test Scenarios

Sprint 1:

User Story 1 from Sprint 1: “As a Spotify user, I want to be able to login using my spotify account.”
User Story 2 from Sprint 1: User Story 2: “As a Spotify user, I want to be able to have access to see my profile.”

Scenario:

1. Navigate to the Groupify index page;
   Select either “sign in” or “Continue with Groupify” ;
2. User should see a Spotify page which prompts Spotify login
   Either click “agree” or Log into Spotify via type;
   a. name = <testemail@ucsc.edu>
   b. password = <12345>
   c. password confirmation = <12345>
   d. press “Log In” button
   e. User should see a Spotify page which prompts Spotify login
   f. Click “agree”
3. User should see a blank Groupify profile page displaying their Spotify profile photo and display name, confirming they have logged in.

Sprint 2:

User Story 1 from Sprint 2: "As a Groupify user, I want to be able to see my top listened to tracks so that I can gain insight on my own music taste.”

Scenario for user story 1 and user story 2:

1. Log into Groupify
2. Navigate to the Groupify home page
3. The user scrolls down to see tile labeled “Top 10 Songs”
4. Directly adjacent to the “Top 10 Songs” label, there is a dropdown menu that says “last 4 weeks”.
5. Below both the “Top 10 Songs” and dropdown menu, the user will see a vertical list of the top 10 songs they’ve listened to in the last 4 weeks.
   i. Hover over the dropdown menu and other options listed are “last 6 months” and “of all time”
   ii. Click on “last 6 months”

        1. The user will see the list change and display a vertical list of the top 10 songs they've listened to in the last 6 months.
    iii. Click on "of all time"
        1. The user will see the list change and display a vertical list of the top 10 songs they've listened to of all time.
6. Each song will have a song image and then directly to it's right, the name of the song and the artist it's by is displayed.
7. Click on the song image, song title, or artist name
8. The spotify web browser will open in another tab and start playing the song that was clicked.

User Story 2 from Sprint 2: "As a Groupify user, I want to be able to create a bio and other unique profile features so that other people can see it"

Scenario 1 for user story 2:

1. Log into Groupify
2. Navigate to the Groupify home page
3. Scroll to see the tile that has the user's profile picture, the user's display name, and a text box. This tile is in the center of the page.
4. Click inside the white text box
    a. The user should see a green button with a checkmark and a red button with an 'x' will appear on the right of the tile next to each other directly below the text box.
5. In the text box type:
    a. bio = <This is my Groupify Bio>
    b. Click the green button with a check mark to save your bio
    c. The user should see that the green and red buttons will no longer display and "This is my Groupify Bio" will be displayed inside the text box.

User Story 3 from Sprint 2: "As a Groupify user, I want to be able to visually change my profile so that it matches the aesthetic of my music tastes, and remain that way when I log out and back in."

Scenario for user story 3:

1. Log into Groupify
2. Navigate to the Groupify home page
3. Hover "theme" in the page's top navigation bar
4. User should see a list of theme names drop down from the "theme" button
    a. Displaying "Default", "Country", "Rap", "Pop", "R&B", "Lo-Fi", "Metal"
5. Click "Pop"
6. User should see background gradients, tile colors, button colors, and text color all change to reflect the theme change. For Pop, the colors will change to pink, blue, and white.

User Story 4 from Sprint 2: "As a Groupify user, I want to be able to add friends so that I can see their profiles"

Scenario 1 for user story 4:

1. Log onto Groupify
2. On the left panel labeled "Following" scroll down
3. Click the box that has a "plus" sign icon.
4. The user should be redirected to another page that is titled "Follow a User"
5. Below the title, there is a white search bar that has the words "Search User"
6. Click inside the search bar
7. Type:
    a. friend name = <my friend>
    b. Click the blue "Submit" button
    c. User should see a bar with their friend's profile picture, display name, and a green button with an icon of a person and the plus sign "+"
    d. Click the green button
    e. The user's input from the search bar will be cleared.
    f. Click the blue "Back" button
    g. In the left panel labeled "Following" the user should see "my friend's" profile picture, their display name, and their active status in a tile.

Scenario 2 for user story 4:

1. Log onto Groupify
2. Visit the link that "your-friend" sent you:
    a. Got to = <http://shams.pythonanywhere.com/groupify/user/your-friend>
    b. User should see at the center of your-friend's profile page, there is a green "Follow" button
    c. Click the green button
3. In the left panel labeled "Following" the user should see "your-friend's" profile picture, their display name, and their active status in a tile.

User Story 5 from Sprint 2: "As a Groupify user, I want users to be able to search for albums using Spotify so I can add it to my profile."

Scenario for user story 5:

1. Log onto Groupify
2. Click the blue "edit banner" button in the center of the screen
    a. The users should be taken to a page titled "Select Album to Edit"
3. Click on a square
    a. A red button and text saying "search to edit album" will appear
4. Go to the right panel titled "Search for Cover Art"
5. Click in the search bar labeled "Find an Album"
6. Type:
    a. Album_input = <Ariana grande>

b. User should see results for albums show up as they type in the album_input
c. Click on an album
    i. Selected_album = <Positions>
    ii. The user should see the selected_album art appear in the squared that was clicked
d. Click the green “Save” button
    i. The “Save” button should turn white and a small black bar will pop up at the bottom of the screen that confirms “Albums Saved!”
e. Click the red “Exit” button
    i. The user should be redirected to the home page and the Album art will be displayed on the album banner

User Story 6 from Sprint 2: "As a Groupify user, I want a settings page to determine the information about myself others can see, and I want my settings to be saved so I don’t have to set them every time I log in."

We didn’t complete this user story because we reprioritized and realized this user story was not needed, therefore there are no Systems Tests for this user story.

Sprint 3:

User Story 1 from Sprint 3: "As a Groupify user, I want to be able to access the app online so I can use all the cool features.”

Scenario for user story 1:

1. Type into browser search bar;
    a. < http://shams.pythonanywhere.com/groupify/index >

User Story 2 from Sprint 3: "As a Groupify user, I want to be able to recommend albums to people who view my profile, and be able to easily customize that list.”

Scenario 1 for user story 2:

1. Log onto Groupify
2. Click the “Edit Banner” button in the center of the screen
    a. The users should be taken to a page titled “Select Album to Edit”
3. Click on a square
    a. Text saying “search to edit album” will appear
4. Go to the right panel titled “Search for Cover Art”
5. Click in the search bar labeled “Find an Album”
6. Type:
    a. Album_input = <Post Malone>
    b. User should see results for albums show up as they type in the album_input

    c. Click on an album
        i. Selected_album = <Hollywood's Bleeding>
        ii. The user should see the selected_album art appear in the squared that was clicked
        iii. Click and drag the added album to the first spot of the grid
            1. The user should see the album take the place of the image that was previously there and it should reorder the banner so that the images shift to the left by one

7. Repeat step 6 until all the squares are filled, ordered, and customized with albums chosen by the user.
8. Click the green “Save” button
    a. The “Save” button should turn white and a small black bar will pop up at the bottom of the screen that confirms “Albums Saved!”
9. Click the red “Exit” button
    a. The user should be redirected to the home page and the albums will be displayed on the banner exactly the way the user saved it

Scenario 2 for user story 2:

1. Start with a full banner (using the steps from Scenario 1)
2. Click the “Edit Banner” button
    a. “Search to edit album” text should not be showing on any albums
3. Click in the search bar labeled “Find an Album”
4. Type:
    a. Album_input = <Talking Heads>
    b. User should see results
    c. Click on an album
        i. Selected_album = <Remain in Light>
        ii. A black pop up that reads, “Select a square for this album!” should appear
    d. Click on an album already in the banner
        i. The “search to edit album text” as well as a red button with an “x” should appear
        ii. Click on the selected album from the search results again
        iii. This album should replace the highlighted entry on the banner, still displaying the text and button overlay
5. Click the green “Save” button
    a. The “Albums saved!” pop up should appear
    b. The “Save” button should now be gray and unclickable
6. Drag an album from the banner and drop it back into the square it is already in
    a. The list should not change
    b. The “Save” button should still be gray
7. Click on an album from the banner, see the overlay covering it, and drag it to a different square

8. The list should be reordered with the dragged album in the square it was dragged to
    a. The overlay should still appear over the album that was dragged
    b. The “Save” button should be green (do not click it for this test)
9. Click the red “x” button on the highlighted album
    a. The cover art should disappear and be replaced by a square with a pencil icon
    b. The “x” button should disappear, but the overlay text should remain
10. Drag the empty square to a different space on the banner
    a. It should relocate exactly like the filled spaces
11. Click the “Exit” button
    a. An alert that reads, “Your albums are not saved. Are you sure you want to exit?” should appear at the top of the page
        i. Click “OK”
        ii. The user should be redirected to the home page, and the albums should be displayed in the same order as in Step 1

Scenario 3 for user story 2:

1. Start with a full banner (using the steps from Scenario 1)
2. Click the “Edit Banner” button
    a. Drag any album on the banner to a different square
    b. Click “Exit”
    c. The confirmation alert should appear
        i. Click “Cancel”
        ii. The user should not be redirected and remain on the same page

Scenario 4 for user story 2:

3. Start with a full banner (using the steps from Scenario 1)
    a. coverList=<[“Plastic Beach”, “Weezer”, “Remain in Light”, “Paul’s Boutique”, “Pet Sounds”, “Hello Nasty”, “Demon Days”, “Is This It”, “Revolver”, “Room on Fire”, “Animals”, “Low”]> (Note: these albums would actually be stored as images, with the file names provided by Spotify)
4. Click each album on the banner
    a. Each square should open the corresponding album on Spotify in a new tab or window
5. Click the “Edit Banner” button
    a. Drag the albums on the banner into a different order
        i. For example: coverList=<[“Animals”, “Demon Days”, “Hello Nasty”, “Is This It”, “Low”, “Paul’s Boutique”, “Pet Sounds”, “Plastic Beach”, “Remain in Light”, “Revolver, “Room on Fire”, “Weezer”]>
    b. Click “Save”
    c. Click “Exit”
        i. The user should be redirected to the home page

        ii. The albums should be shown in the new arrangement
6. Click each album on the banner
    a. Each square should open the corresponding album on Spotify as in Step 4a, with the link of any given square matching the displayed album

User Story 3 from Sprint 3: "As a Groupify user, I want my profile to display my Spotify profile's public information, such as my playlists, my top artists, and what genres I enjoy"

Scenario 1 for user story 3:

1. Log into Groupify
2. Navigate to the Groupify home page
3. The user scrolls down to see tile labeled "Top Artists"
4. Directly adjacent to the "Top Artists" label, there is a dropdown menu that says "last 4 weeks".
5. Below both the "Top Artists" and dropdown menu, the user will see a vertical list of the top 10 artists they've listened to in the last 4 weeks.
    i. Hover over the dropdown menu and other options listed are "last 6 months" and "of all time"
    ii. Click on "last 6 months"
        1. The user will see the list change and display a vertical list of the top 10 artists they've listened to in the last 6 months.
    iii. Click on "of all time"
        1. The user will see the list change and display a vertical list of the top 10 artists they've listened to of all time.
6. Each artist will have an image of the artist's spotify profile picture, and then directly to it's right, the artist's name, the top 2 genres their music falls under, and the number of followers they have on Spotify.
7. Click on any of the Top artist components
8. The spotify web browser will open in another tab and take you to the artist's profile on Spotify.

Scenario 2 for user story 3:

1. Log into Groupify
2. Navigate to the Groupify home page
3. The user scrolls down to see tile labeled "Top Artists"
4. At the bottom of that tile click on the words "See All"
5. The user should be taken to another page titled "Your Artists"
6. Directly under the search bar, there is a dropdown menu that says "last 4 weeks".
7. Below both the "Top Artists" and dropdown menu, the user will see a vertical list of the top 50 artists they've listened to in the last 4 weeks.
    i. Hover over the dropdown menu and other options listed are "last 6 months" and "of all time"
    ii. Click on "last 6 months"

            1. The user will see the list change and display a vertical list of the top 50 artists they've listened to in the last 6 months.
        iii. Click on "of all time"
            1. The user will see the list change and display a vertical list of the top 50 artists they've listened to of all time.
8. Each artist card will have the artist's name, an image of the artist's spotify profile picture, and then directly to it's right, the top 2 genres their music falls under, and the number of followers they have on Spotify.
9. Click on any of the Top artist components
10. The spotify web browser will open in another tab and take you to the artist's profile on Spotify.

Scenario 3 for user story 3:

1. Log into Groupify
2. Navigate to the Groupify home page
3. The user sees a tile that says "Public Playlists". This tile is on the right side of the page.
4. Below the "Public Playlists" label the user should be able to see a vertical list of their public playlists.
    a. Each playlist will display the playlist image and then directly to it's right, the name of the playlist, and the playlist description (if it has a description).
5. Click on the song image, song title, or artist name
6. The spotify web browser will open in another tab and take you to the playlist on spotify.

User Story 4 from Sprint 3: "As a Groupify user, I want to be able to edit my profile easily and intuitively"

Scenario 1 for user story 4:

1. Log into Groupify
2. Navigate to the Groupify home page
3. User will see that the blue button in the center says "Edit Banner"
4. Click on the blue button
5. It should take you to another page where you can edit the look of your album banner

Scenario 2 for user story 4:

1. Log into Groupify
2. Navigate to the Groupify home page
3. On the left panel labeled "Following" scroll down
4. Hover the box that has a "plus" sign icon.
    a. The user should see an animation of the words "Add Friend" appearing
    b. Click on the "Add Friend" button

5. It should take you to another page reserved for just searching, adding, and removing friends

User Story 5 from Sprint 3: "As a Groupify user, I want to be able to listen to music through Groupify's music player with my added friends, and join in listening rooms with my friends"

Scenario 1 for user story 5:

1. This user wants to host a group listening session
2. Log into Groupify
3. Log in and Open Spotify
4. Navigate to the Groupify Group Listen Page
   a. Click on the "group" button on the top navigation bar
5. Start playing music on Spotify
   a. Song = <test drive by ariana grande>
6. Click the clipboard button at the bottom of the page
   a. The user should get a pop up confirming that you have successfully copied the "share your session" link
   b. Link = <http://shams.pythonanywhere.com/groupify/groupSession/userID>
7. Send that copied link to your friends
8. Tell your friends to log into Groupify and open Spotify and visit the session link the user sent
9. The user should be able to see their friends cards soon after they have visited the link.
   a. The user should see their friends profile picture and display name in the card
   b. If the friend leaves, the user should be able to see their card disappear.

Scenario for user story 5:

1. This user wants to join a group listening session
2. Log into Groupify
3. Log in and Open Spotify
4. Navigate to the Groupify Group Listen Page sent by "your-friend"
   a. Link = <http://shams.pythonanywhere.com/groupify/groupSession/your-friend>
5. When the user reaches the group listen page:
   a. The user should see who else is in the group listening session. Every card will have the other listener's profile picture and display name.
   b. If the friend leaves, the user should be able to see their card disappear.
   c. The current song that the host is playing will start playing on the User's spotify

User Story 6 from Sprint 3: "As a Groupify user, I want my friends to see a custom active status when I am online"

Scenario 1 for user story 6:
1. Navigate to user home page
2. Navigate to Add Friend's panel on the left
3. Click on active status box
    a. Type "Active on Groupify!" in text box
    b. Click save button
4. Tell your friend to go to their home page
5. User should be able to see your newly updated active status

Scenario 2 for user story 6:
1. Navigate to user home page
2. Navigate to Add Friend's panel on the left
3. Click on active status box
    a. Type "Active on Groupify!" in text box
    b. Click delete button
4. Tell your friend to go to their home page
5. User should be able to see your previously saved active status

Sprint 4:

User Story 1 from Sprint 4: "As a Groupify user, I want to see who is in my currently joined group session"

Scenario:
1. Go to group listen page
2. Play a song via Spotify
3. Send your session link to your friends
    a. Refresh/Wait 10-15 seconds and user should appear

User Story 2 from Sprint 4: "As a premium Spotify user, I want groupify's group session to be restricted for premium users only"

Scenario 1:
1. Go to group listen page as non-premium user
    a. User should see page that says group listen feature is only for premium users only

Scenario 2:
1. Go to group listen page as premium user
    a. User should be able to access the page with no restrictions

User Story 3 from Sprint 4: "As a Groupify user, I want my friends to only join my group session if I am on the page as well"

Scenario 1:

1. You as the host, is not on your session page
    a. Users enter your group session page
    b. Users should see a page notifying that the host is not in session

Scenario 2:

1. You as the host is on your session page
    a. Users enter your group session page
    b. Users should be able to see your name and icon along with their name and icon on the group panel
    c. Users should be able to see what song the host is listening to and listen along

User Story 4 from Sprint 4: "As a Groupify user, I want to be able to sync with the host, pause a song, or skip to the next song.”

Scenario:

1. Go to someone else’s hosted session
2. Listen to the current song playing
3. Click pause button
    a. User should be out of sync with host
4. Click play button
5. Click sync button
    a. User should now be caught up with host’s song & current time

User Story 5 from Sprint 4: "As a Groupify user, I want to be able to queue songs and see what songs are currently in a queue for the group session, despite if I am host or not”

Due to time constraints, we did not complete this user story and do not have any scenarios to implement.

User Story 6 from Sprint 4: "As a Groupify user, I want to see all my playlists, not just the initial few shown on my profile page”

Scenario:

1. Go to home user page
2. Navigate to Public Playlist column
3. Scroll to the bottom and click See All
4. User should be directed to a page with all public playlists shown
    a. Scroll through the scrollable box if there are more than 3 playlists
    b. Users should see all their public playlists along with its description

General Overview of Our Testing:

Most of our testing was done manually. Each team member tested anything they worked on by locally hosting their application and seeing if the changes produced expected results. In general, one person is not able to cover all equivalence classes on their own, and therefore required other people to test the changes they made. People who did not work on a feature would black box test new features. Integration tests were also performed manually. They became especially important when merging two separate features located in different branches.

When dealing with the Spotify API, one of the main concerns during testing is the many possibilities user's may have for their top artists, songs, and playlists. A key concern was how to deal with a user who lacks pieces of information, such as an image for their playlist. Testing these scenarios manually required us to modify aspects of our own Spotify information for accurate replication.

Why So Few Unit Tests?

Looking at our unit tests in groupify/test_controllers.py, viewers of the file will note that the number of unit tests included does not match the number of functions written. This is due to three major factors: a serious lack of resources and documentation for unit testing in py4web (our web framework), difficulties in mocking Spotify login information, and a failure to create functions well suited for constructor and setter injections. These reasons are listed in order, with the lack of information on unit testing in py4web being the most important reason.

Py4web is a web framework with very little information written about it. In fact, most of the information about it is from a different web framework that it builds on top of, web2py. Popular alternatives to py4web like Flask and Django have much more information and guides written about them. An outdated piece of documentation for web2py claims that unit testing in this framework is much different than other frameworks. Unfortunately, the examples given on how to unit test in web2py are incompatible with py4web. This substantially hindered our ability to make unit tests without intermingling test code with production code. For this reason, manual testing was the preferred means of testing.

Modules Tested

Logging In to Groupify:

Expected Output: Users should be greeted to a welcoming page asking them to click on a button and be redirected to a Spotify login page. Then after the user has put in their login information, they should be redirected to their user profile in Groupify.

How It Was Tested: As this involved getting user information from the Spotify API, the team had to use their own Spotify profiles to test the login process.

Errors Encountered: Spotify could return an error during the login process, causing the user to be stuck in a page displaying the error code Spotify returns.

Solutions: Any error Spotify returns had to be caught, and then the user had to be redirected to the login page to commence the process again.

Functions Involved: getIndex(), userLogin(), getCallback(), getUserInfo()

Displaying The Correct Name and Profile Picture:

Expected Output: Upon login, users should see the same name and profile picture on their Spotify Profile to be on their Groupify user page.

How It Was Tested: This was manually tested by logging in to the website and verifying that all the information is equal to that on Spotify. In addition, names and profiles were changed to verify that information could change on Groupify as well.

Errors Encountered: Some profile pictures had a size that made our website extend in length.

Solutions: The size of each user profile had to be statically set.

Unit-Test: Entering an existing profile is tested by checking the status code it returns. If the status code returns (200 OK), then the test is passed.

Functions Involved: getUserInfo(), getUserProfile()

Browsing Other Peoples Profiles:

Expected Output: Navigating to other people's profiles should return all the information they profile owner is able to see. However, the visitor must not be allowed to edit the profile's theme, bio, and album wall. Traveling to a user that does not exist should redirect users to a user not found page.

How It Was Tested: This was manually tested by making sure certain buttons were not visible to the visitor. The team would look in the navbar to see if the theme could be changed. In addition, the team would make sure the edit banner button did not display on other people's profiles. Finally, the bio was tested by making sure the text area could not be edited, and the submit buttons would not display. The team would edit the URL to travel to a page where a user does not exist. Then we would make sure we would be redirected to a user not found page.

Errors Encountered: N/A

Solutions: N/A

Unit-Test: A request is made to enter a page where the user does not exist. This is done by setting the userID in the URL to 1, which is impossible to be given to Spotify users. Once the request is made, the status code is asserted to be a redirect.

Functions Involved: editableProfile(), getUserProfile()

User's Top 10 Songs:

Expected Output: After logging in, the user's most listened to songs are displayed and stored in the database. They can click through different periods of time. When looking at another person's profile, they can see that person's top songs and click through different periods of time as well.

How It Was Tested: Verifying the top songs by using other applications which also display top songs. Making sure every period was able to be displayed. Checking the database entries for the song information.

Errors Encountered: N/A

Solutions: N/A

Functions: getUserProfile(), getTopSongsPost(), getTopSongs(), getTopTracksFunction(), getTopTracksLen()

User's Top Artists:

Expected Output: After logging in, the user's most played artists are displayed and stored in the database. They can click through different periods of time. When looking at another person's profile, they can see that person's top artists and click through different periods of time as well. The genre of artist is displayed.

How It Was Tested: Verifying the top songs by using other applications which also display top artists. Making sure every period was able to be displayed. Checking the database entries for the artist information.

Errors Encountered: Artists can have multiple genres. These genres need to be stored in the database, but with a field type of a list of strings, one artist held the genres of every single artist.

Solutions: Store the genres inside a list, which results in a string of "[genre, genre]" being inserted into the list of strings. Then split this list by the brackets and assign it to its corresponding artist.

Functions: getUserProfile(), getTopArtistsPost(), getTopArtists(), getTopArtistsFunction(), getTopArtistsLen()

User's Playlists:

Expected Output: After logging in, the user's public playlists are displayed and stored in the database. When looking at another person's profile, they can see that person's public playlists as well.

How It Was Tested: Verifying the public playlists manually by making sure the team's public playlists are correctly displayed on our profiles. Checking the database entries for the playlist information.

Errors Encountered: Any playlist that did not have an image would return a 505 error upon access to the user's profile.

Solutions: Catch any time a playlist does not have an image and insert a placeholder image in the database instead.

Unit-Test: Create a mock of a Spotify API call asking for a user's playlists. Assert that the parsePlaylistResults() function correctly parses through the JSON and returns relevant playlist information. Another unit test to check if the function correctly handles playlists with no descriptions or images. Another unit test to check if the storePlaylists() correctly stores the playlist information in the database.

Functions: getUserProfile(), parsePlaylistResults(), getPlaylistsFromAPI(), storePlaylists()

The Settings Page:

Expected Output: Users click on the settings page and see their profile picture and a link to their profile. Users can click on the link to have it copied to their clipboard. In addition, a user guide is embedded. Below that, a delete profile button that removes all your information from the database.

How It Was Tested: The team clicked on their settings page to make sure all their information was correct, and the resource guide was correctly embedded. Then we made sure deleting their profile deleted the information from the database by checking the py4web dashboard. In addition, we would use separate accounts to check that the profile we deleted is no longer available to see.

Errors Encountered: N/A

Solutions: N/A

Functions: getSettings(), deleteProfile(), userNotFound()

The Edit Albums Page:

Expected Output: Users can search for albums in the right-hand panel. They should be able to change the order of their albums. Their information will be stored in the database upon hitting save or canceled upon hitting exit.

How It Was Tested: Inserting album titles or artists names into the search bar and verify its output. Check for boundary cases such as titles with numbers or multiple words. Move album tiles around and check if saving maintains the order. Make changes to the albums and hit exit without saving to see if the changes are correctly discarded.

Errors Encountered: Any user editing their banners simultaneously would result in those users overwriting each other.

Solutions: Make sure that the function uses session.

Functions: editUserSquare(), getSquares(), saveAlbumsCoversToSquares(), doSearch(), parseAlbumResults()

Editing The Bio:

Expected Output: A user on their own profile can edit a textarea. Once they are done editing, they see a button to save their changes or discard them.

How It Was Tested: Writing random sentences. Checking what would occur with very long text. Checking if the character limit was maintained. Making sure non alphanumeric characters would not return any errors when stored in the database. Refreshing to see if bio was saved. Logging in to different accounts to see if the bio is displayed for other users.

Errors Encountered: N/A

Solutions: N/A

Functions: getUserBio(), postUserBio(), getUserProfile()

Editing The Theme:

Expected Output: A user can edit their theme by clicking a desired theme on the top navbar. Changing theme is not possible anywhere except on their own profile page.

How It Was Tested: Changing the theme and making sure it remains upon refresh and logout. Making sure no theme is possible to be changed anywhere besides your own profile. Making sure the theme is maintained in all other pages the user owns, such as the “See All” pages and the group session.

Errors Encountered: N/A

Solutions: N/A

Functions: updateThemeInDB(), returnTheme(), getUserProfile()

Seeing Followed Users:

Expected Output: When visiting your own profile, or someone else’s profile, the left navbar should display the users you follow along with their active status. A search bar allows users to filter to find a friend. Users cannot add a friend twice.

How It Was Tested: After following a user through the “add friend” page or through the follow button, the team verified that the friend was displayed on the navbar. The team also made sure to check that the person is placed in alphabetical order. The search bar was tested to find specific people.

Errors Encountered: Users could add the same person twice.

Solutions: Check for a duplicate friend entry before following a person.

Functions: addFriend(), addFriendFromProfile(), getUserProfile()

Unfollowing Users:

Expected Output: Clicking an unfollow button in the “add friend” page or on a user’s profile will immediately display the option to follow back. The navbar in the left will be updated with the removal of the person. In the dbFriend table, the entry designated the following status will be removed.

How It Was Tested: The team made sure that unfollowing a user would mean one could also immediately follow back. The navbar was checked to see if the person was correctly removed. The number of entries in the dbFriends table was observed to decrease by one.

Errors Encountered: N/A

Solutions: N/A

Functions: deleteFriend(), unfollowFriendFromProfile(), getUserProfile(),

Seeing Group Session Page:

Expected Output: The only people who are allowed to enter the group session page are those with a premium Spotify account. In addition, people cannot join another person’s group session if the host is not in the session.

How It Was Tested: A non premium account was used to test whether a user was correctly blocked from the session. Premium accounts were used to find out if they were correctly allowed in. Attempts to join group sessions without the host were attempted by going to profiles we knew were not in the group session.

Errors Encountered: It was possible to join a group session of a user that did not exist.

Solutions: Prevent someone who is not the host from creating a group session that is not their own.

Unit-Test: Entering an existing group session, or one the host will create, is tested by checking the status code it returns. If the status code returns (200 OK), then the test is passed. In addition, joining a group session that does not have a host or does not exist is asserted to be a redirection.

Functions: groupSession()

Keeping Track of Host’s Playing Song:

Expected Output: Upon entering the group session, a call is made to get the song the host is listening to. Every 5 seconds the call is made again. During this time, the playback time of the song is incremented every second.

How It Was Tested: This was manually tested by joining the group session and seeing if the song and time displayed on the group session page was accurate to what was playing on our own Spotify window. In addition, we would skip around and change songs to see if the 5 second calls afterwards were accurate.

Errors Encountered: Once a song ended, there was a chance that the user had to wait 5 seconds for the new song to be displayed.

Solutions: Check if the song has reached its end, then update the song the host is playing.

Functions: groupSession(), isGroupSessionHost(), getCurrentPlaying(), getDevice(), pauseOrPlayTrack()

Synchronizing Visitor's In Group Session:

Expected Output: Upon entering the group session, a call is made to get the last updated position of the song the host is listening to. If the user becomes out of synch with the host, they can hit the synchronize button to get the last updated position of the song the host is listening to.

How It Was Tested: One user would host a session and play a song while the other user would be the visitor and test the visitor functionality. They would make sure the synchronize button would play the correct song.

Errors Encountered: Once a song ended, there was a chance that the user had to wait 5 seconds for the new song to be displayed.

Solutions: Check if the song has reached its end, then update the song the host is playing

Functions: groupSession(), isGroupSessionHost(), shouldSynchronizeVisitor(), getDevice(), synchronizeVisitor(), pauseOrPlayTrack()

Keeping Track of Who is In Group Session:

Expected Output: When a user enters or refreshes the session, they can see the people who are on the same group session. When a user leaves the session, it should be updated within 15 seconds to each user that they have left. When the host leaves, the rest of the group session members are kicked out.

How It Was Tested: Two users would join a group session. They would test if the people could see each other immediately upon refresh. They would also test if they would eventually see each other within 15 seconds. They then tested if a user leaving would be reflected upon refresh or waiting 15 seconds. The team tested if a host leaving would kick everyone remaining from the session.

Errors Encountered: People who left the session by closing a tab or leaving the browser would not be kicked out of the session.

Solutions: Set up a last active time field in the database. If the user has not updated this field in a certain amount of time, the user will be kicked out of the session.

Functions: groupSession(), getPeopleInSession(), removePeopleInSession(), checkActivePeopleInGroupSession()