SW Engineering CSC648/848 Spring 2021

SYNC

Team 06

Team Lead	Rebecca Zumaeta	rzumaeta@mail.sfsu.edu
Front End Lead	Bryan Fetner	bfetner@mail.sfsu.edu
Back End Lead	Ashwini Managuli	amanaguli@mail.sfsu.edu
Front End Member	Malcolm Angelo De Villar	mdevillar@mail.sfsu.edu
Front End Member	Hirva Patel	hpatel11@mail.sfsu.edu
Github Master and back end member	Vishakha Tyagi	vtyagi@mail.sfsu.edu
Back End Member	Luong Dang	ldang2@mail.sfsu.edu

Milestone 4

05/13/2021

History Table

Version	Date	Notes
M4V1	05/13/2021	
M3V2	04/25/2021	
M3V1	04/22/2021	
M2V2	04/08/2021	
M2V1	04/01/2021	
M1V2	03/09/2021	
M1V1	03/05/2021	

Table of Contents

1. Product Summary

SYNC

1. Unregistered Users

1.1. Unregistered Users shall be able to log into their Spotify Premium.
1.2. Unregistered Users shall be able to access the landing page of the website.
1.4. Unregistered Users shall be able to access the FAQ of the website.
1.5. Unregistered Users shall be able to access the Contact page of the website.

2. Registered Users

2.7. Registered Users shall have a premium Spotify account.
2.8. Registered Users shall be able to login into their Spotify Premium.
2.9. Registered Users shall be able to listen to music in real time.
2.10. Registered Users shall be able to access the Homepage of the website.
2.12. Registered Users shall be able to access the FAQ of the website.
2.13. Registered Users shall be able to access the Contact page of the website.
2.26. Registered Users shall be able to logout.
2.28. Registered Users that create a room shall be able to name the room.
2.29. Registered Users that create a room shall be able to add songs to queue
2.31. Registered Users shall be able to pause audio out put of currently playing songs.
2.35. Registered Users shall be able to create a “room” public
2.36. Registered Users shall be able to create a “room” private.
2.37. Registered Users shall be able to search a public room.
2.39. Registered Users shall be able to join a public room.
2.40. Registered Users shall be able to join a private room.
2.42. Registered Users shall be able to share room link to invite people to their room.
2.44. Registered Users shall be able to search for songs.
2.45. Registered Users shall be able to add a song to the queue that can be played
2.48. Registered Users shall be able to chat in all room types.
2.49. Registered Users shall be able to chat with people who joined in their created room.
2.57 Registered Users shall be able to vote on songs to be played next in queue
2.58. Registered Users shall be able to leave rooms
2.59. Registered Users shall be able to vote

3. Administrators

3.61. Administrators shall be able to join all rooms
3.63. Administrators shall be able to ban users.

3.67. Administrators shall be able to delete rooms.
3.64. Administrators shall be able to leave messages regarding reasons for user bans.

4. Rooms

4.69. Rooms shall display the room name.
4.70. Rooms shall display if they are public or private.=
4.76. Rooms shall display the current song.
4.77. Rooms shall display the song queue.
4.78. Rooms shall display genre.
4.79. Rooms shall display chat.
3.80. Rooms shall display who commented in the chat.
4.81. Rooms shall have the voting system.
4.85. Rooms shall play music

5. Website

5.82. Website shall display username.
5.83. Website shall display the user's profile picture.
5.84. Website shall let users resume activity using cookies
5.89 Website shall keep logged in users' info by cookies.
5.91 Website shall let users login.
5.93. Website shall display the website's contact info.
5.95. Website shall allow user to copy invitation link from rooms
5.96. Website shall allow users to search for a room.
5.97. Website shall allow users to join a room.
5.98. Website shall allow users to see public rooms' info.
5. 99.Website shall display available public rooms.
5.100. Website shall play music from Spotify.
5.101. Website shall support popular browsers

Unique to SYNC:

SYNC is a site that brings people together to listen concurrently with others through their music interests. The unique features of other platforms that sync music with others are that our rooms cater to the users. Our site provides the user's the capability to create their personalized listening rooms and search and browse through rooms created by other users. Not everyone likes country music, but there are quite a few country music lovers; room creation and room search allows users a fun and easy way to listen to music synced with others. Once in a room the user has many ways to interact with the site and with other users. Searching for a song to place into the room queue, voting on songs within the queue to place priority on the next queued song and of course chat with others within a room. These unique features allow for users to express their interests and create a sense of community. A common pass time with friends is sharing music; our site makes it easier for friends near or far to listen to music together in SYNC.

URL to Product: SYNC

http://18.219.141.181:3000/

2. Usability Test Plan

Test Objectives

We plan to test the usability of our project by analyzing how easy it is for the user to familiarize the functionality and reasoning behind the site, how easy it is for the user to navigate within the site, and the user satisfaction through their experience moving through the site. To properly test the site, we'd like for trial users to test a few of the unique and basic features SYNC has to offer. The goal of this test is to get an understanding on how non technical people view and use the site so that we may improve in any place that may not be clear or understandable. Some of the features to be tested are the creation of a room or joining an existing room to listen to music with others. We'd like to see if they enjoy the customization of creating a room and if joining a room is straightforward and fairly quick. Another set of features we'd like for the trial user's to test would be the features within a room such as voting, chatting and more. We'd like to test for smooth and painless handling of these applications for the user. User experience is our top priority for our application, so testing it's features and its entirety is key to a great functioning site.

Test Description

System setup

As a user, what is needed to then properly set up for using our application are a few hardware components and access to networking. Any given user wanting to use our site would need a computer (a tower personal computer or laptop will suffice) with power, running Windows or MacOS operating system and connectected to a internet (though WIFI or directly plugged into a router) are needed. Users will also need a browser such as Chrome or FireFox installed on their computer and a Spotify Premium account.

Starting point

Once a computer is set up (with above specifications) navigate to one of the supported browsers. The browser application will prompt a search engine such as Google. Typing in our site link http://18.219.141.181:3000/ and hitting enter will then prompt the search engine to navigate to the site. The starting point of our site is the landing page. The first thing the user is prompted to do on the landing page is to sign in with their Spotify account. It is important for the user to sign into our site with their Spotify's premium account. The user will be redirected to Spotify related site to fully sign in but then will be redirected to our home page. From here the user can access SYNC site and play around with the many features the site provides.

Who are the intended users

The intended Users are primarily user's with access to the correct technology, Premium Spotify accounts, music lovers and social seekers. SYNC is a site that accepts all who love music and sharing their interests in music taste with others. Community is key for our site to be successful.

URL of the system to be tested and what is to be measured

SYNC: http://18.219.141.181:3000/. Developers and those involved with building SYNC would like to measure the amount of knowledge the user understands of the site after a short period of time roaming around within it, ease of access and movement within the site, user friendliness of the site, and satisfaction of it's social capabilities.

Usability Task Description

The tasks the testers need to execute before completing the questionnaire is to correctly set up their system using the above section 'System Setup'and get to the landing page using steps from the above section 'Starting Point'.

TASK	**Description**
Task	Create a room
Machine state	Create page or home page
Sucessful completion	Room is created, and they are taken to newly created room.
Benchmark	30 seconds

TASK	**Description**
Task	Search for a room
Machine state	Join Page
Sucessful completion	Search result displayed based on search query.
Benchmark	5 seconds

TASK	**Description**
Task	Search for a song in a room

Machine state	In a room with the song search component open.
Sucessful completion	Song results based on song search query.
Benchmark	5 seconds

TASK	**Description**
Task	Vote on a song in the queue in a room
Machine state	In a room with the queue component open.
Sucessful completion	Checkbox is checked, and vote count increments.
Benchmark	1 second

TASK	**Description**
Task	Chat in a room
Machine state	In a room
Sucessful completion	Inputed test will appear in shared chat log.
Benchmark	10 seconds

TASK	**Description**
Task	Share link to a room
Machine state	In a room
Sucessful completion	Gaining access to link to copy and share.
Benchmark	5 seconds

TASK	Description
Task	Signing into Spotify
Machine state	Landing page
Sucessful completion	Redirected to home page, with username depicted at top right of page, signifying being logged in.
Benchmark	1 minute

Test/ Use cases	% Completed	Errors	Comments
Create a room	100%	none	Room creation complete
Search for a room	100%	none	Search for a room is complete
Search for a song in a room	60%	Some songs are not discoverable	Spotify API does not provide all discoverable songs that Spotify application offers.
Vote on a song in the queue in a room	90%	Vote count did not increment in relation to users input	Does not reflect my vote when the vote checkbox is clicked.
Chat in a room	80%	Some text will not appear in chat history	Does not take into special characters or emoticons
Share link to a room	100%	none	Copying on link within a room is implemented to share on other platforms
Signing into Spotify	70%	Some users are not allowed to proceed	Only premium spotify users are allowed to proceed to our site

Questionnaire

Question	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
The site was easy to navigate	1	2	3	4	5
Creating a room was a straightforward process.	1	2	3	4	5
I was confused when interacting in the rooms	1	2	3	5	5

Results

Question	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
The site was easy to navigate	0	0	0	0	3
Creating a room was a straightforward process.	0	0	0	0	3
I was confused when interacting in the rooms	2	1	0	0	0

3. QA Test Plan

Test Objectives

Our objective with these tests is to verify that information is being handled correctly through out the site. Due to the fact that most of the sites focus is on organizing "listening rooms" and using data from Spotify as well as sharing that to various user, we need to know that this will all be handled reliably. We will be targeting specific tests such as retrieving authorization from Spotify on the users behalf, and also retrieving song track information from Spotify that will provide a large part of our content on the app. Additionally, we will test our ability to store and retrieve information on the database, such as user information and room information, which will be used in various instances on the site and is required for most of it's functionality.

HW and SW setup

For all features the hardware need is a computer (a tower personal computer or laptop will suffice) with power, running Windows or MacOS operating system and connectected to a internet (though WIFI or directly plugged into a router) are needed. Users will also need a browser such as Chrome or FireFox installed on their computer and a Spotify Premium account. Last set up requirement is to navigate a browser and paste **http://18.219.141.181:3000/** to navigate to SYNC.

Feature to be Tested

1. Login to Spotify:
   Test if logging into Spotify grants the user the required authentication token used to access various Spotify features in the app.
2. Search room by genre:
   Based on a selected genre during search, the resulting rooms would contain the genre specified in search query.
3. Getting song track information from Spotify API:
   This tests the retrieval of specific song information needed for app functionality fron the Spotify API.
4. Storing room information on database:
   This tests the ability to store room information that will be shared with other users who will potentially joining the same room.
5. Retrieve information from all public rooms:
   This is to test the backend query to the database that requests only the specific rooms based on their public or private status.

QA Test plan

Tested on Chrome, FireFox and MS Edge: produced same results

#	Test Title	Description	Test Input	Expected Correct Output	Pass/Fail
1	Login to Spotify	Test if logging into Spotify though SYNC will give users Spotify Authentication.	User Spotify Premium account credentials	User receives Spotify Authentication Token in their Cookies.	Pass
2	Search room by genre	Test the tag ‘pop’ in search bar when it is set to look at only genres	‘pop’	Get 10 room results of gener descriptions that are ‘pop’	Pass

3	Getting song track information from Spotify API	Test retrieval of song track information from the Spotify API using Axios.	Axios.get(
			"https://api.spotify.com/v1/tracks/" +
			"3n3Ppam7vgaVa1iaRUc9Lp",
			{
			headers: {
			Accept: "application/json",
			"Content-Type": "application/json",
			Authorization: "Bearer " +
			userInfo.spotifyToken,
			},
			}
			)
				songName = "Mr. Brightside"
				
				songArtist = "The Killers"
				
				songTrackUrl = "spotify:track:3n3Ppam7vgaVa1iaRUc9Lp"
				
				smallSongImageUrl = "https://i.scdn.co/image/ab67616d0000485119c284a6855f4945dc5a3cd73"
				
				largeSongImageUrl = "https://i.scdn.co/image/ab67616d0000b2739c284a6855f4945dc5a3cd73"
				
				songDuration =
					Pass

				222200	
4	Storing room information on database.	Test acquisition of data onto database in regards to newly created room information such as room name and room genre.	Room Name: "Bill's Room" Room Genre: "Rock"	In Room Table a new column with the room_name as "Bill's Room", and genre as "Rock"	Pass
5	Retrieve information from all public rooms	Test a query that requests all public room information	SELECT room_id, room_name FROM Rooms WHERE roomType =True;	Array of all room data excluding any private rooms	Fail

4. Code Review

The coding style that we have chosen for our code is vertical, organized and easy on the eyes to view. There are some rules we are following to write our code.

1. Clarity and simplicity of Expression: We are clear of our objective throughout our code.

2. Naming: We are naming the methods, functions and variables in our code as clear as possible.

3. Comments: We are adding comments to the code for ease in understanding.

4. Information hiding: We are securing information from the rest of the system where possible to decrease the coupling between modules and make the system more maintainable.

5. Nesting: We are avoiding deep nesting of loops and conditions so that understanding the program logic is easy and we do not harm the static and dynamic behavior of a program.

7. Module size: We are maintaining uniform module size and avoided using too big or too small modules.

8. Module Interface: We are examining modules with complex interface carefully.

9. Side-effects: We are trying to avoid side effects when possible because when a module is invoked, it sometimes has a side effect of modifying the program state.

Peer review from another team

From: Malcolm Angelo Santos De Villar <mdevillar@mail.sfsu.edu>

Sent: Tuesday, May 11, 2021 2:23 PM

To: Christian James Achacoso <cachacoso@mail.sfsu.edu>

Subject: Code review (CSC648 Team 6)

Hi Christian,

I would like to request a code review for the snippet of code that we implemented for searching a room and joining a room. Your input is greatly appreciated. Attached is the file for review with comments on each line.

Thank you,

Malcolm

From: Christian James Achacoso <cachacoso@mail.sfsu.edu>

Sent: Wednesday, May 12, 2021 8:44 PM

To: Malcolm Angelo Santos De Villar <mdevillar@mail.sfsu.edu>

Subject: Re: Code review (CSC648 Team 6)

Hi Malcolm,

Sorry for the late response.

After taking look at the code, I thought it was very well organized.

While the comments helped me understand the context of the code, I was still able to understand the implementation of the searching and joining of a room for your application.

I was especially interested in the dropdown menu for both rooms and the genre.

Overall, really great work and implementation of these functionalities. Hope to use this app later on in the future!

Best regards,

Christian Achacoso

From: Malcolm Angelo Santos De Villar <mdevillar@mail.sfsu.edu>

Sent: Wednesday, May 12, 2021 9:12 PM

To: Rebecca Maria Zumaeta <rzumaeta@mail.sfsu.edu>

Subject: Fw: Code review (CSC648 Team 6)

Hi Team Lead,

I am forwarding the email of Mr. Achacoso's code review of a snippet of our code. Please review and let me know of any concerns or questions.

Thank you,

Malcolm

Peer review from within 06 team

From: Malcolm Angelo Santos De Villar <mdevillar@mail.sfsu.edu>

Sent: Monday, May 10, 2021 11:23 PM

To: Rebecca Maria Zumaeta <rzumaeta@mail.sfsu.edu>

Subject: Code Review Request (Join.jsx)

Hi Team Lead,

I would like to request a code review for the snippet of code that we implemented for searching a room. Your input is greatly appreciated. Attached is the file with comments on each line.

Thank you,

Malcolm

----------start of code----------

```
const onClickfunction = ({ key }) => { //if else for search bar
   if (`${key}` == 2) { //search by room name
     searchByName(); //call searchByName function to search by name
     setSearchBarText({
       textField: "e.g. Tom's room", //placeholder
       dropDown: "RoomName", //what shows up in the dropdown menu in the website
     });
   } else if (`${key}` == 3) { //search by room genre based on key specified under menu
     searchByGenre(); //call searchByGenre function to search by name
     setSearchBarText({
       textField: "e.g. Electronic",//placeholder
       dropDown: "Genre",//what shows up in the dropdown menu in the website
     });
   }
 };
```

// note from Rebecca: the indentation is perfect, the code is understanable and is effiecent

```
 const menu = (
   <Menu onClick={onClickfunction}>
     <Menu.Item key="2">RoomName</Menu.Item> //identify the key of what category to
search (RoomName)
```

```
    <Menu.Item key="3">Genre</Menu.Item> //identify the key of what category to search
(Genre)
   </Menu>
  );
```
//note from Rebecca: menu as an object name may to broad, since there are many mini menues throughout the site,unless this is a necissary naming convention

```
  useEffect(() => {
    Axios.get("http://localhost:8000/api/adds/") //use backend api to grab all currently existing
rooms in db
      .then((res) => {
        setViewData(res.data); //show response data
      })
      .catch((er) => console.log(er)); //show error in console if there is an error
  }, []);
```

//note from Rebecca: Naming convention of res and er are very efficient but not a dead giveaway, tehcnically res can mean results or response. For a first time viewer this can be a bit confusing but 'catch(e)' is a standard so understandable! greatly formatted

```
  useEffect(() => {
    if (searchValue === "") { //if nothing in search bar, load all rooms that are existing.
      setSearchedData([]);
    }
  }, [searchValue]);

  const searchAll = () => {};
  const searchByName = () => {
    Axios.get("http://localhost:8000/api/adds/") //api call for searching rooms by room name
      .then((res) => {
        let tempOptions = [];
        res.data.forEach((d) => { //traverse to all matching rooms
          tempOptions.push({ value: d.room_name }); //shows all rooms that have the matching
name in search input
        });
        setOptions(tempOptions); //set search option based on search input
        setViewData(res.data); //render all matching rooms
      })
      .catch((er) => console.log(er)); //show error in console if there is an error
  };
```

//note from Rebecca: I am assuming 'd' in linke 48 and 49 represents data. Love all the comentation, really shows understanding of functions. using 'reassigning variable this to acomplish this' is another way of doing the same thing. options!

```
  const searchByGenre = () => {
```

```
    Axios.get("http://localhost:8000/api/adds/") //api call for searching rooms by room genre
      .then((res) => {
        let tempOptions = [];
        res.data.forEach((d) => { //traverse to all matching rooms
          tempOptions.push({ value: d.genre }); //shows all rooms that have the matching genre in
search input
        });
        setOptions(tempOptions); //set search option based on search input
        setViewData(res.data); //render all matching rooms
      })
      .catch((er) => console.log(er)); //show error in console if there is an error
  };
```

//note from Rebecca: great to put colsole printing for errors at critical points for easy and qiock fixing if need be. Great Job!

```
  {
    /* For joining rooms */
  }

  const joinRoom = (getFromid) => {
    console.log(getFromid)
    const resultRoomId = getFromid; //assign resultRoomId as getFromid to be passed later for url
generation
    console.log(resultRoomId)
    props.history.push("/Room/"+ resultRoomId + "/") //generate the url
  };
```

//note from Rebecca: love comment on 80, i truely understand what is going on here, very cool function!

```
  const searchRoom = () => {
    if (searchValue === "") return;
    let result = viewData.filter((d) => //filter results
      d.room_name.toLowerCase().includes(searchValue.toLowerCase())|| //convert to case
insensitive for ease of use room name
      d.genre.toLowerCase().includes(searchValue.toLowerCase()) //convert to case insensitive for
ease of use genre
    );
    setSearchedData(result); //get all results
  };
```

//Ending notes from Rebecca: I'd hit enter a couple time inbetween blocks or functions, but other then a cople naming conventions, its clear, its consisise, its efficient, it takes into account errors, great all around. Great Work!!!

----------end of code with----------

Good morning Malcolm,

I was impressed with the organization and comments left on this section of code. I have reviewed the code and left some comments but overall happy with the results.

Great job,

Rebecca Zumaeta

rzumaeta@mail.sfsu.edu

5. Self-check on best practices for security

List major assets you are protecting

SYNC's database does not have many major assets since we have now implemented the Spotify API. We only hold displayname and profile pictures from the user's account from Spotify. Other personalized and unique to the site assets we keep in the database are rooms and those are protected through our .pem file as our key to the database.

Confirm that you encrypt PW in the DB

We allow users to login with a Spotify account, therefore, the encryption is done by Spotify's endpoint and no sensitive information from users is stored on our local database. When users login through a pop-up window, spotify returns to us a token which also has expiration that ends when the session ends. Token is the key for a scoped profile information, which is only sufficient for users' activities on our website. In other words, our site has an extra encryption step compared to Spotify.

Confirm Input data validation

Database has all the rooms with tags of genres and room names

The search bar has three options to search. The first one is **All** which takes input of the room names or genre and gives results for the same.When user selects **RoomName** option it populates the search bar based on names with available rooms.When user inputs roomname the option allows searching only based on that. The other option **Genre** allows users to search only based on the genre of the room which is input by the user.

The user input can either be lowercase or uppercase, the search bar lets the user to search through it.

The user can also input partial names and genre, the search bar still gives all the relevant results.

If the user inputs roomname or genre that doesnt exist, wrong input then it shows room with such roomname or genre was not found.

6. Self-check: Adherence to original Non-functional specs

Functionality

1. The website shall be conforming to the tools and frameworks that were approved by the CTO. - **DONE**
2. The website shall be using Amazon Web Services for deployment. - **DONE**
3. The website shall be user friendly. - **DONE**
4. The website shall be simple.- **DONE**
5. The website shall be usable to those who are slightly knowledgeable of navigating websites. - **DONE**

Security

6. Spotify username/ email address and password shall be required to use the web app. - **DONE**
7. Login prompt shall appear when the user first visits the website. - **DONE**
8. Private rooms shall only appear to invited Registered Users. - **ISSUE -** All private rooms only appear for admin to view to keep hidden from register users. Only some registered users can view some private rooms, as long as the creator or participants of a private room share the corresponding link

Privacy

9. Let users accept policies before creating an account. - **ISSUE** - Spotify API handles creation of accounts, SYNC only authenticates the token of completion of logging into Spotify . SYNC does require users to accept SYNC's TOS.
10. Password and other personal information shall be kept hidden. -**ISSUE -** Spotify API handles safekeeping of personal information such as password, due to the account being done through Spotify.
11. Collect Spotify data through API - **ON TRACK**
12. We use API to dictate recommended rooms - **DONE**
13. We use API to group people together. - **ON TRACK**
14. Authenticate users by checking with username and password - **ISSUE** - The Spotify API handles this function, keeping user's account secure

Performance

15. The website shall be up all the time - **ISSUE** - SYNC is hosted in an AWS instance that would need a constant running computer to be up constantly
16. The website shall make a search for a room easily - **DONE**
17. All users shall have their music synced with others having same preferences - **ISSUE** - music will be synced but ‘same preferences’ are not going to continue with our current design
18. Empty rooms should be destroyed automatically - **ISSUE** - SYNC does not have many users quiet yet for this to be a great feature. Rooms will all be closed due to lack of users on site constantly. Does not work with our current design
19. The invite links to private rooms shall stay active till the room is inactive - **ON TRACK**
20. The website shall not add more than 2 seconds to the time it takes to login to Spotify - **DONE**
21. Unused rooms shall be destroyed with 5 minutes of non attendance - **ISSUE** - similarly to 18; SYNC does not have many users quiet yet for this to be a great feature. Rooms will all be closed due to lack of users on site constantly. Does not work with our current design

System Requirements

22. The website shall work up to Version 88.0.4324.150 of Google Chrome. - **DONE**
23. The website shall work up to Version 85.0 of Mozilla Firefox. - **DONE**
24. The website shall work up to Version 81.0.416.64 of Microsoft Edge. - **DONE**
25. The website shall work up to Version 14.0 of Safari. - **ISSUE** - Spotify SDK is not supported on this browser
26. The website shall support Spotify music streaming. - **ON TRACK**

Marketing

27. Each webpage shall have the company logo in the upper left corner. - **DONE**
28. Each webpage shall be clear and easy to understand for first time visitors. - **DONE**
29. Each webpage shall be able to link to social media platforms. - **ON TRACK**

Content

30. The website shall have a navigation bar - **DONE**
31. The website shall have a search bar to search public rooms available. - **DONE**
32. The website shall present a general community room when users first sign in. - **ISSUE** - Due to change old design and lack of users this feature had to be cut for best user experience
33. The website shall present recommended rooms for the specific users. - **ISSUE** - it is displayed under the recommended by search rather than recommended room specific to the user. This was changed due to the Spotify API complexity
34. A navigation bar shall be present to direct users to other parts of the website - **DONE**
35. The website shall present an option to join private rooms - **DONE**

Scalability

36. The rooms shall be able to handle a large number of users. - **DONE**
37. The website shall have sufficient rooms to accommodate a growing number of users. - **DONE**
38. The webapp shall be developed using microservices architecture, contributing to the maintenance of the application. - **ISSUE** - broad statement, we have checks for database entries, tos, confirmation pages, etc to keep simple maintenance of the application

Capability

39. The website shall be capable to provide the same data as requested by the user - **DONE**
40. The website shall be capable to be updated in a timely manner - **DONE**
41. The website shall be capable to resolve problems - **ON TRACK**
42. The website shall be capable to communicate efficiently with the users - **ON TRACK**
43. The website shall be capable to recover from failures - **ON TRACK**

Look and Feel

44. The website shall have subtle colors - **DONE**
45. The website shall have readable fonts - **DONE**
46. The website shall have simple layout - **DONE**
47. The website shall have a balanced design - **DONE**
48. The website shall be easy to navigate - **DONE**
49. The website shall load pages in less than 5 seconds - **DONE**
50. The website shall load images in less than 2 seconds - **DONE**
51. Public rooms shall be easily identifiable **DONE**
52. Private rooms shall be easily identifiable - **ON TRACK**
53. Room content shall be immediately recognizable. - **DONE**

Coding Standards

54. The code shall be understandable - **DONE**
55. The code shall be organized - **DONE**
56. The code shall have proper working functions - **DONE**
57. The code shall have in-line comments - **ON TRACK**
58. The code shall be pushed or pulled from branches in git properly - **DONE**
59. The code shall be well maintained in the relevant branches - **DONE**
60. The code shall have proper documentation - **ISSUE** - Did not get enough time to do proper documentation of the code
61. The code shall be maintained with one coding style - **ON TRACK**
62. The code shall have proper formatting - **ON TRACK**
63. The code shall have indentation - **DONE**

Availability

64. The website shall be active even if no users are active on the website - **ON TRACK**
65. The website shall resync when loss of connection occurs - **ON TRACK**
66. The website shall refresh when the website does not load. - **ISSUE** - This feature had to be cut as we did not have much time to into refreshing pages automatically
67. The website will make general rooms unavailable within a set time or minimum user tolerance. - **ISSUE** - Due to lack of users visiting the site, this function would not work correctly so we changed the design
68. The website shall generate error messages when an error occurs
    1. Connection loss - **ISSUE** - There is many ways to loose connection, SYNC does have some generated error messages due to pages no longer existing but not for connection losing on our end
    2. Incorrect Credentials - **ISSUE** - Spotify API handles this error

Fault tolerance

69. Website shall be able to refresh the Chat area when disconnected. - **ISSUE** - We did not get around to looking to how to do automatic refreshing
70. Room shall be able to be refreshed when disconnected. - **DONE**
71. Website shall reattempt accessing database if a database query fails. - **DONE**

Storage

72. Store users' listening history on the database.- **ISSUE** - Removed due to lack of time, is in lower priority
73. Store users friends list on database.- **ISSUE** - Removed due to lack of time, is in lower priority
74. Store administrator notices about users. - **ISSUE** - Administrators will be able to ban users and will hold notices on strictly that issue
75. Remove friends from the users friend list on the database if user removes - **ISSUE** - Removed due to lack of time, is in lower priority
76. Store users favorited music on the database. - **ISSUE** - Removed due to lack of time, is in lower priority
77. Remove the favorite list from database if user removes - **ISSUE** - Removed due to lack of time, is in lower priority
78. Store chat while the room is open in the database.- **ISSUE** -Removed due to lack of time, is in lower priority
79. Store users voting record.- **ISSUE** - Removed due to lack of time, is in lower priority
80. Store usernames on the database. **ON TRACK**
81. Store passwords on the database. - **ISSUE** - Spotify API handles this function
82. Store users ignore list on the database. - **ISSUE** - Removed due to lack of time, is in lower priority
83. Store a banned word list on the database.- **ISSUE** - Removed this feature due to lack of time, will be implemented during maintenance period.
84. Store a banned user list on the database. - **ISSUE** - The implementation of a list of banned users will not be held in the database, but users will be marked banned once Admin bans them
85. Store room name on database - **DONE**
86. Remove room name from database when room is destroyed - **ON TRACK**
87. Store room description on database. - **ISSUE** - This is removed because our genre tag to the room is what is stored instead of description
88. Remove room description from database when room is destroyed.- **ISSUE** - This is removed because description no longer exists, refer to 87
89. Store room song history on database.- **ISSUE** - This is removed because due to lack of time and research
90. Remove room song history from the database when the room is destroyed.- **ISSUE** - This is removed because due to lack of time and research
91. Store room voting history on database.- **ISSUE** - This is removed because due to not needing to keep history of votes
92. Remove room voting history from the database when the room is destroyed.- **ISSUE** - This is removed because due to not needing to keep history of votes; refer to 91

Expected Load

93. The website shall support as many rooms as Amazon Web Services can support. - **DONE**
94. The website shall support as many users as Amazon Web Services can support. - **DONE**

Legal

95. The website shall have Terms and Conditions that the user will be required to accept before they log in. - **DONE**
96. The website shall have privacy policies in the settings section. - **ISSUE** - This is done through the user accepting SYNC's TOS
97. The website shall have copyright notice. - **DONE**

7. List of Contributions to the document

Team Lead, Document Master	Rebecca Zumaeta	Wrote up most of documentation, conducted Usability and QA tests, coordinated code review with member outside of team and conducted code review as a team member. Added branch QA for testing purposes
Front End Lead, mediator	Bryan Fetner	Continued implementing Spotify API to existing horizontal prototype. Added branch QA2 for further testing purposes. Integrating functions such as spotiy login, song search with spotify API, etc. Assisted in confirming the the committed functions and to original non-functional specs.
Back End Lead, Document contributor, Site Deployer	Ashwini Managuli	Built chat room function and maintained instance, assited in building out all tables within DB diagram. Assisted in confirming the the committed functions and to original non-functional specs.
Front End Member Document contributor	Malcolm Angelo De Villar	Integrated functions needed for smooth user experience such a button mapping and error pages and to maintain the UI of the site. Assisted in confirming the the committed functions and to original non-functional specs.
Front End Member	Hirva Patel	Created administer panel and updated tables for private vs public rooms, added function of banned user in DB, assisted in sharing link function, modified functions of search bar. Assisted in confirming the the committed functions and to original non-functional specs.
Github Master and back end member Document contributor	Vishakha Tyagi	Assited in building out all tables within DB diagram like table for voting, assisted with document in section 4 and 3. Assisted in confirming the the committed functions and to original non-functional specs.

Back End Member	Luong Dang	Assisted in the building functionality of the spotify login, spotify player and spotify api integration into queue and syncing the music. Assisted in confirming the the committed functions and to original non-functional specs.