Final SW Engineering Class CSC 648-848 Section 04 Spring 2021

Team 06

SYNC

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

http://18.219.141.181:3000/

05/20/2021

Table of Contents

1. Product Summary..... 3
2. Milestone documents..... 6
3. Screenshots of Actual Final Project.....7
4. Screenshots of key DB tables……..... 11
5. Screenshots of Task Management (Trello)..……..……..... 12
6. Team member contributions ……..……...16
7. Post Analysis 29

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 output 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' information by cookies.
5.91 Website shall let users login.
5.93. Website shall display the website's contact information.
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' information.
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/

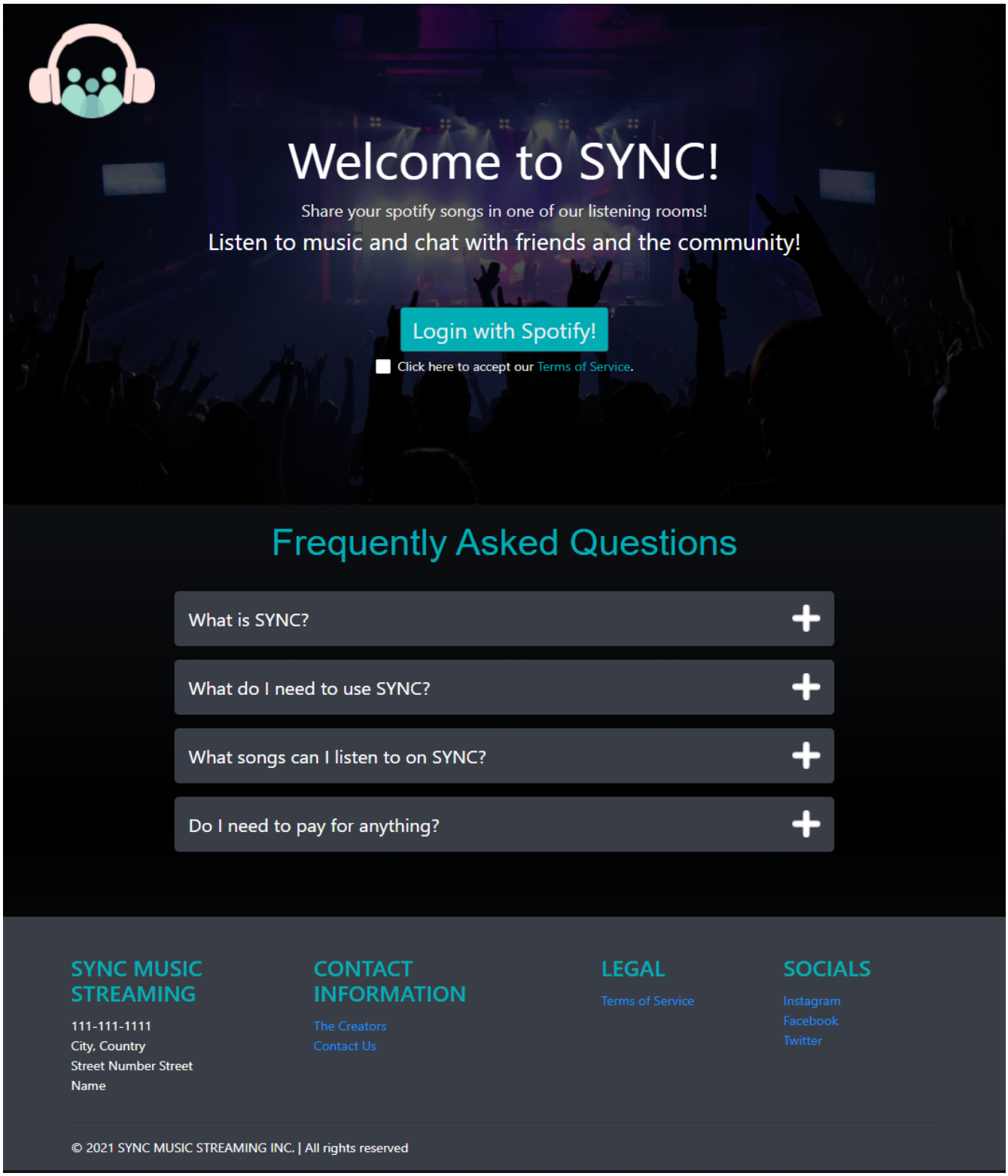
Provided Spotify Premium account with Admin access:

Email : sickgatorz06@gmail.com
password : letzSYNC

2. Milestone Document

Milestone 1 Version 2	**30**
Milestone 2 Version 2	**60**
Milestone 3 Version 2	**91**
Milestone 4 Version 2	**144**

3. Screenshots of actual final product



Spotify

sync

You agree that sync will be able to:

View your Spotify account data
Your email
The type of Spotify subscription you have, your account country and your settings for explicit content filtering
Your name and username, your profile picture, how many followers you have on Spotify and your public playlists

View your activity on Spotify
Content you have recently played
The content you are playing and Spotify Connect devices information
Your top artists and content

Take actions in Spotify on your behalf
Control Spotify on your devices
Stream and control Spotify on your other devices

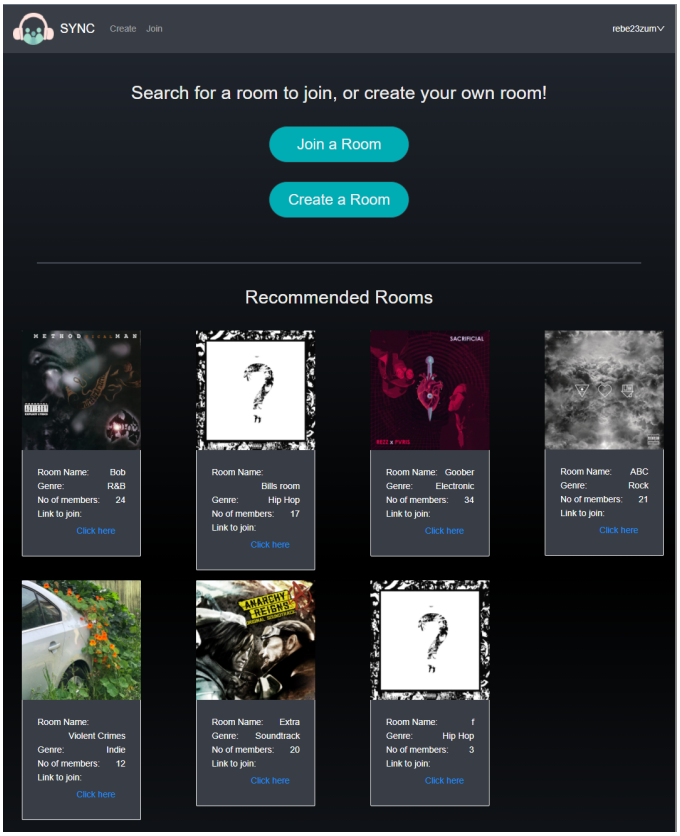
You can remove this access at any time at spotify.com/account.

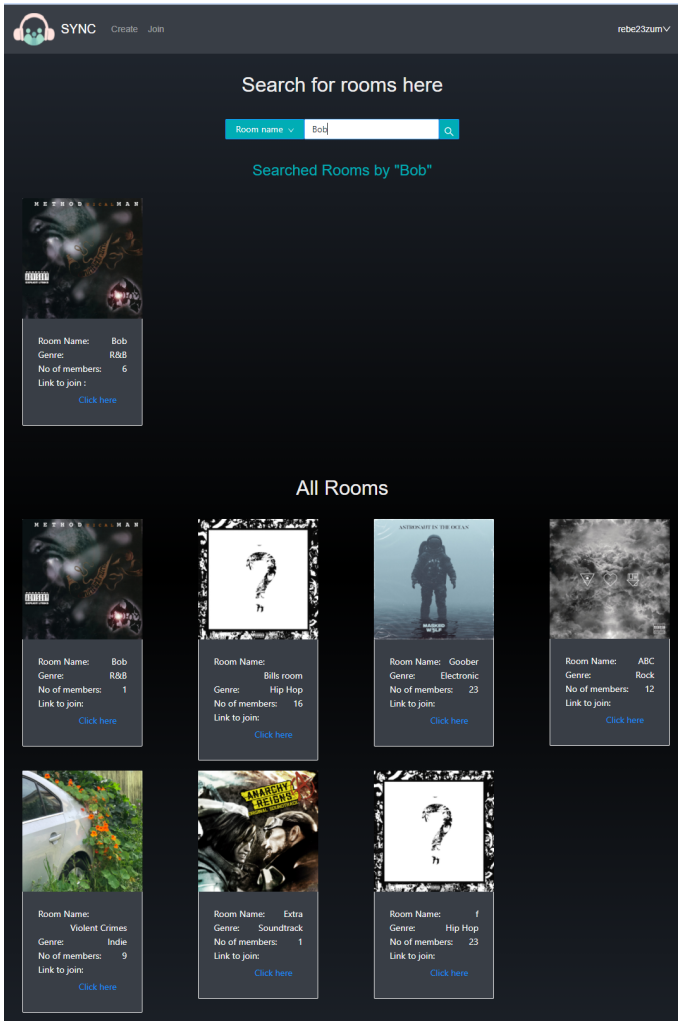
For more information about how sync can use your personal data, please see sync's privacy policy.

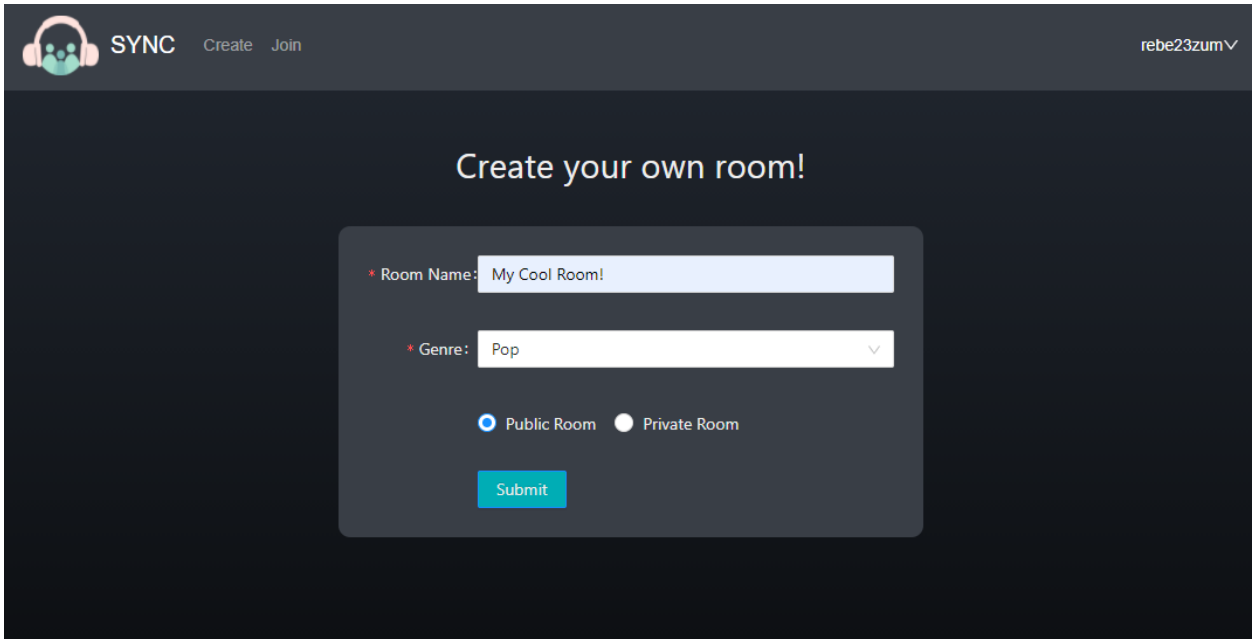
Logged in as rebe23zum.
(Not you?)

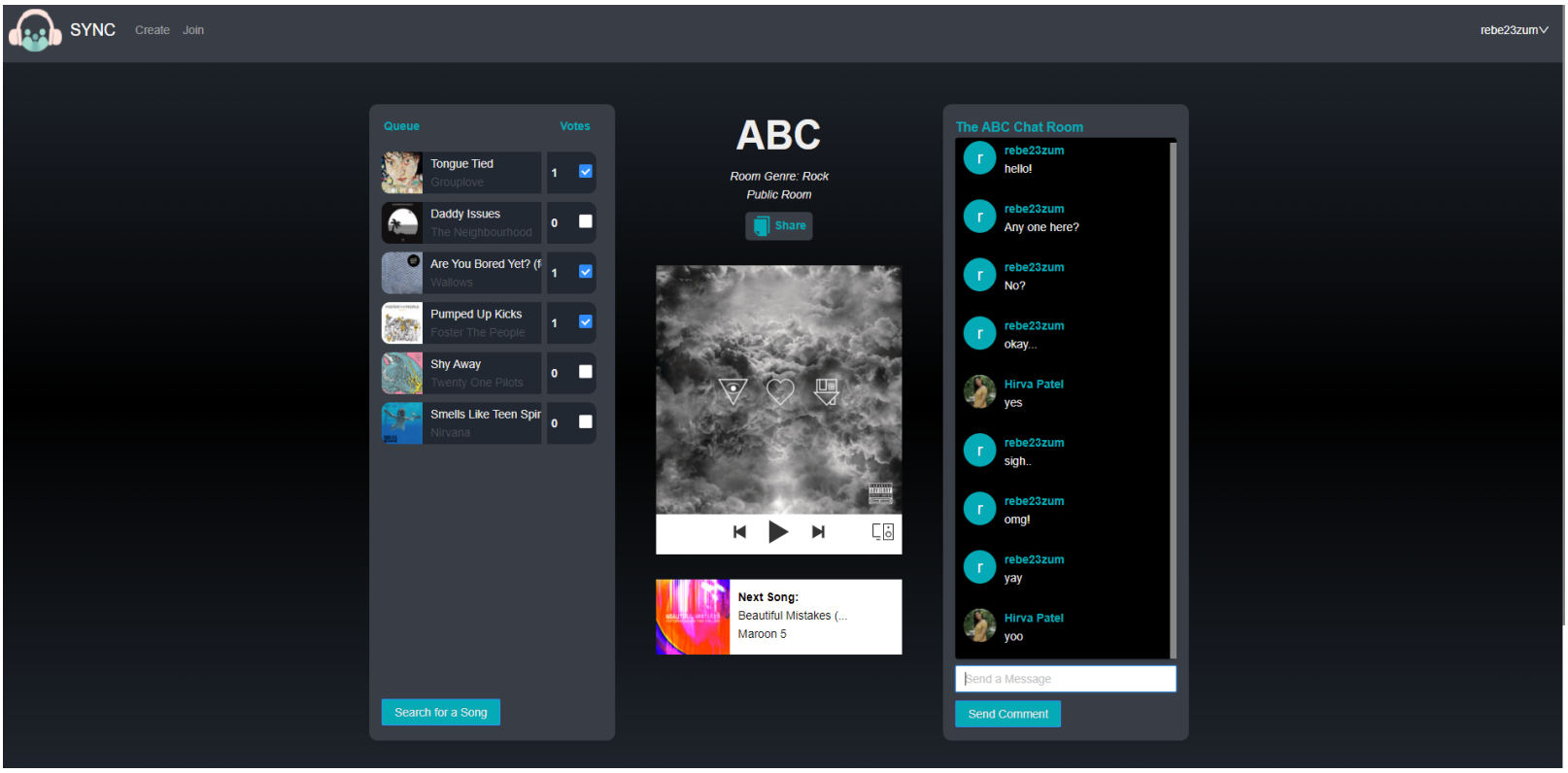
AGREE

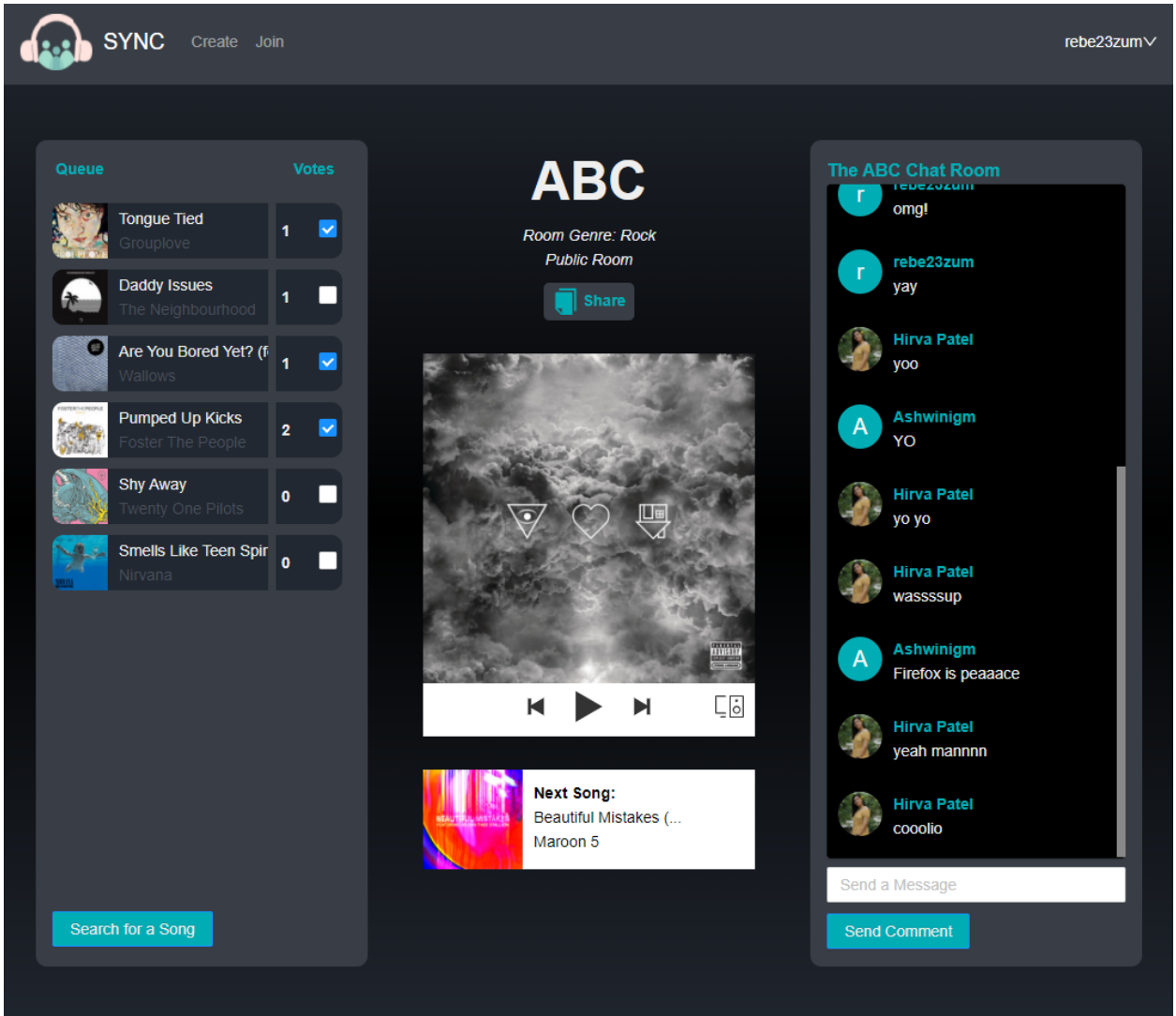
CANCEL




SYNC
Create
Join
rebe23zum
Search for rooms here
Room name
Bob
Searched Rooms by "Bob"
Room Name: Bob
Genre: R&B
No of members: 6
Link to join :
Click here
All Rooms
Room Name: Bob
Genre: R&B
No of members: 1
Link to join:
Click here
Room Name: Bills room
Genre: Hip Hop
No of members: 16
Link to join:
Click here
Room Name: Goober
Genre: Electronic
No of members: 23
Link to join:
Click here
Room Name: ABC
Genre: Rock
No of members: 12
Link to join:
Click here
Room Name: Violent Crimes
Genre: Indie
No of members: 9
Link to join:
Click here
Room Name: Extra
Genre: Soundtrack
No of members: 1
Link to join:
Click here
Room Name: f
Genre: Hip Hop
No of members: 23
Link to join:
Click here


SYNC
Create
Join
rebe23zum
Create your own room!
* Room Name:
My Cool Room!
* Genre:
Pop
Public Room
Private Room
Submit


SYNC
Create
Join
rebe23zum
Queue
Votes
Tongue Tied
Grouplove
1
Daddy Issues
The Neighbourhood
0
Are You Bored Yet? (f
Wallows
1
Pumped Up Kicks
Foster The People
1
Shy Away
Twenty One Pilots
0
Smells Like Teen Spir
Nirvana
0
Search for a Song
ABC
Room Genre: Rock
Public Room
Share
Next Song:
Beautiful Mistakes (...
Maroon 5
The ABC Chat Room
rebe23zum
hello!
rebe23zum
Any one here?
rebe23zum
No?
rebe23zum
okay...
Hirva Patel
yes
rebe23zum
sigh..
rebe23zum
omg!
rebe23zum
yay
Hirva Patel
yoo
Send a Message
Send Comment


SYNC
Create
Join
rebe23zum
Queue
Votes
Tongue Tied
Grouplove
1
Daddy Issues
The Neighbourhood
1
Are You Bored Yet? (f
Wallows
1
Pumped Up Kicks
Foster The People
2
Shy Away
Twenty One Pilots
0
Smells Like Teen Spir
Nirvana
0
Search for a Song
ABC
Room Genre: Rock
Public Room
Share
Next Song:
Beautiful Mistakes (...
Maroon 5
The ABC Chat Room
omg!
rebe23zum
yay
Hirva Patel
yoo
Ashwinigm
YO
Hirva Patel
yo yo
Hirva Patel
wassssup
Ashwinigm
Firefox is peaaace
Hirva Patel
yeah mannnn
Hirva Patel
cooolio
Send a Message
Send Comment

4. Screenshots of key DB tables

Room's table

room_name	genre	roomImageUrl	population	roomType	room_id
Bob	R&B	https://i.scdn.co/image/ab67616d0000b273309...	NULL	0	1267630720
Bills room	Hip Hop	https://i.scdn.co/image/ab67616d0000b2732d7...	NULL	0	136110601
Goober	Electronic	https://i.scdn.co/image/ab67616d0000b2733de...	NULL	0	1531741818
ABC	Rock	https://i.scdn.co/image/ab67616d0000b273c03...	NULL	0	173952732
Violent Crimes	Indie	./assets/9.jpg	NULL	0	320956661
Extra	Soundtrack	https://i.scdn.co/image/ab67616d0000b273e0d...	NULL	0	407310638
f	Hip Hop	https://i.scdn.co/image/ab67616d0000b273806...	NULL	0	741396979
NULL	NULL	NULL	NULL	NULL	NULL

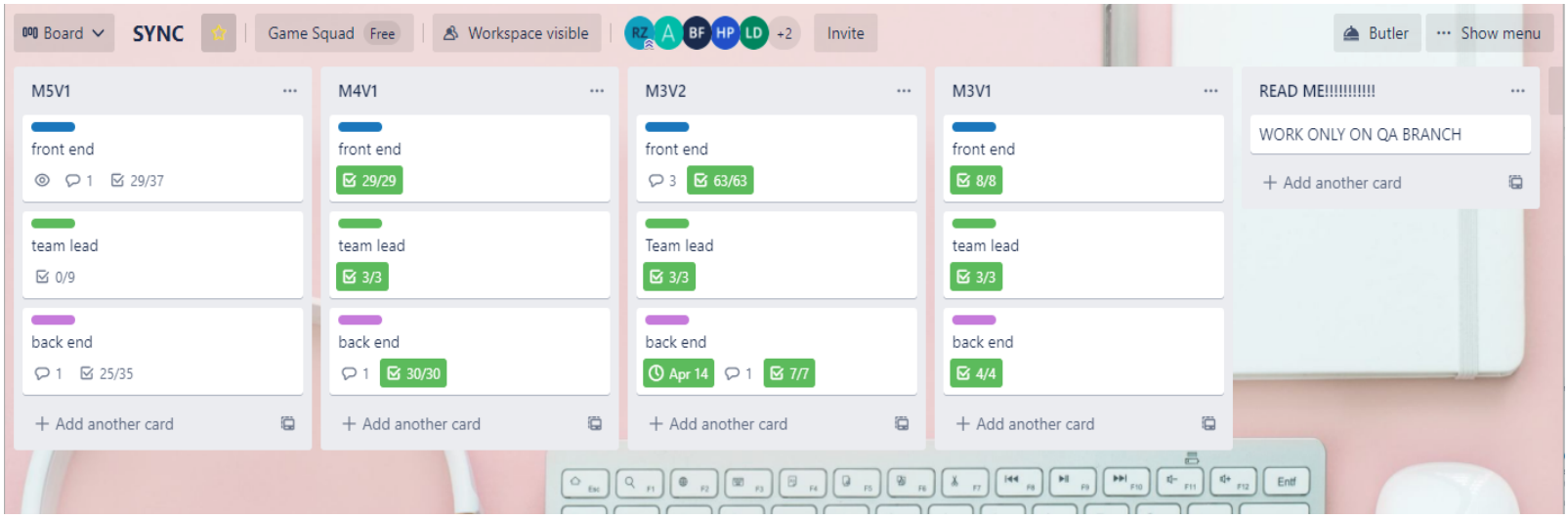
User's Table

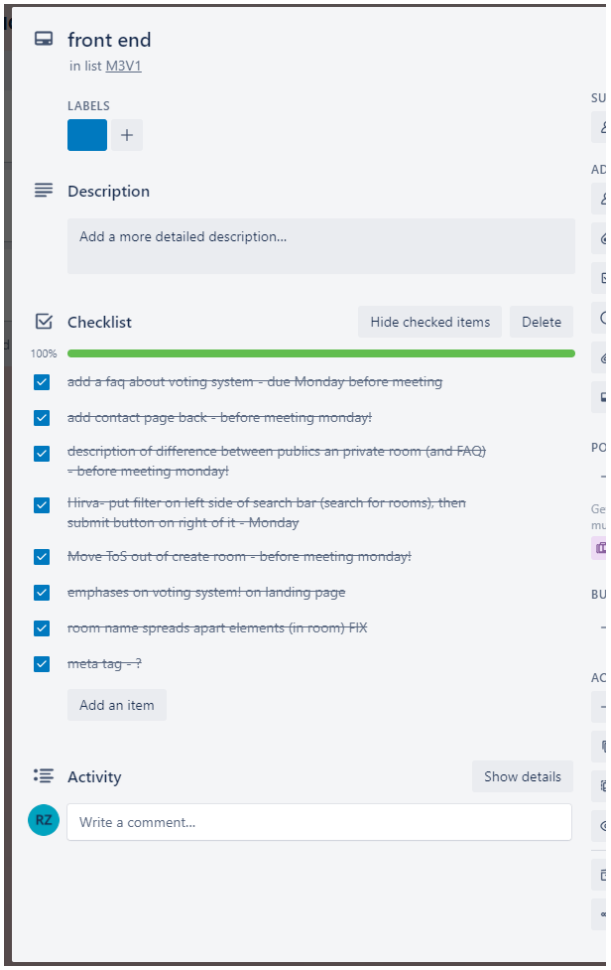
user_id	display_name	profile_pic	admin_status	ban_status	ban_comments
12129725114	LDEEZY	https://i.scdn.co/image/ab6775700000ee851a8...	0	0	NULL
6vyb0iibl7cyb558iht55lsa8	Mafaldine	none	0	0	NULL
7dzljxfz0z0uvjxiiplzj55lb	TheSickestOfGatorz06	none	0	0	NULL
ffmfsqqfvm3671xf42f45whyq	BryanCCSF	https://i.scdn.co/image/ab6775700000ee85dd0...	0	0	NULL
kc86ozhjb1ykoxqiflulrgwaf	Hirva Patel	https://i.scdn.co/image/ab6775700000ee852ba...	0	0	NULL
n7bbi3pwo17602ganuguspfjk	Vishakha Tyagi	none	0	0	NULL
rebe23zum	rebe23zum	none	0	0	NULL
t7hisvylwi1sw7ldqr69pxepu	Ashwinigm	none	0	0	NULL
v8z1yhdn581hzr2l97gpkneoc	BryanSFSU	none	0	0	NULL
NULL	NULL	NULL	NULL	NULL	NULL

Vote's Table

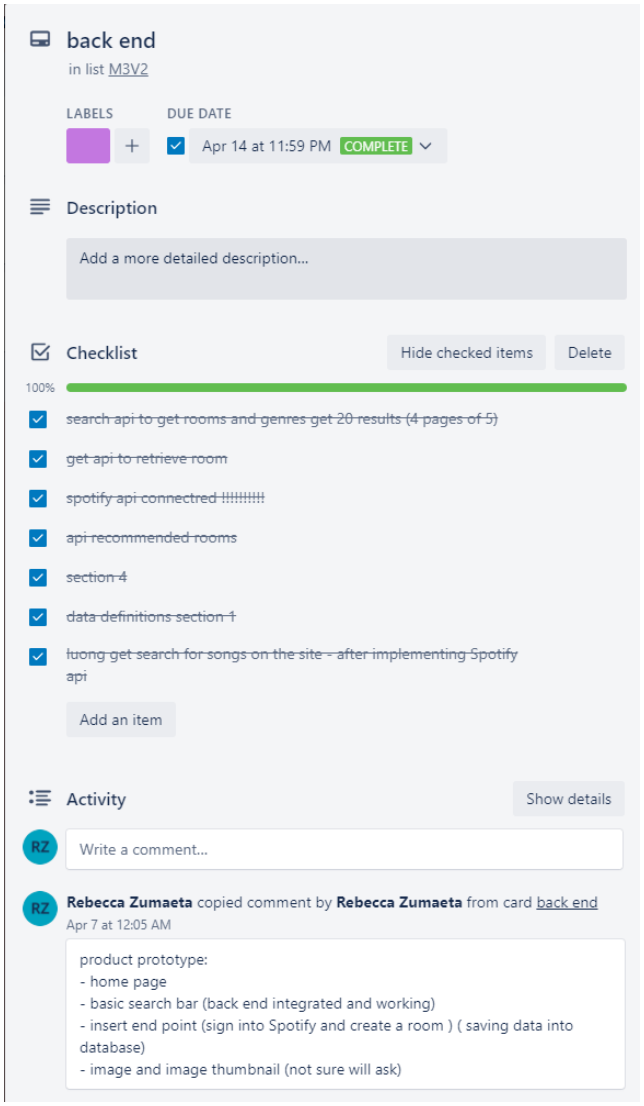
vote_id	room_id	user_id	song_id
10074618296vyb0iibl7cyb558iht55lsa8	1531741818	6vyb0iibl7cyb558iht55lsa8	1007461829
1007461829ffmfsqqfvm3671xf42f45whyq	1531741818	ffmfsqqfvm3671xf42f45whyq	1007461829
1007461829rebe23zum	1531741818	rebe23zum	1007461829
1021636757ffmfsqqfvm3671xf42f45whyq	1531741818	ffmfsqqfvm3671xf42f45whyq	1021636757
1032080577kc86ozhjb1ykoxqiflulrgwaf	1531741818	kc86ozhjb1ykoxqiflulrgwaf	1032080577
111685874112129725114	136110601	12129725114	1116858741
1154051393n7bbi3pwo17602ganuguspfjk	1267630720	n7bbi3pwo17602ganuguspfjk	1154051393
119449801312129725114	136110601	12129725114	1194498013
12663113ffmfsqqfvm3671xf42f45whyq	407310638	ffmfsqqfvm3671xf42f45whyq	12663113
12663113kc86ozhjb1ykoxqiflulrgwaf	407310638	kc86ozhjb1ykoxqiflulrgwaf	12663113
1295585029ffmfsqqfvm3671xf42f45whyq	1531741818	ffmfsqqfvm3671xf42f45whyq	1295585029
13077001016vyb0iibl7cyb558iht55lsa8	1531741818	6vyb0iibl7cyb558iht55lsa8	1307700101
1307700101ffmfsqqfvm3671xf42f45whyq	1531741818	ffmfsqqfvm3671xf42f45whyq	1307700101
130770010 1rebe23zum	1531741818	rebe23zum	1307700101
13103063456vyb0iibl7cyb558iht55lsa8	1531741818	6vyb0iibl7cyb558iht55lsa8	1310306345

5. Screenshots of Task Management System (Trello)

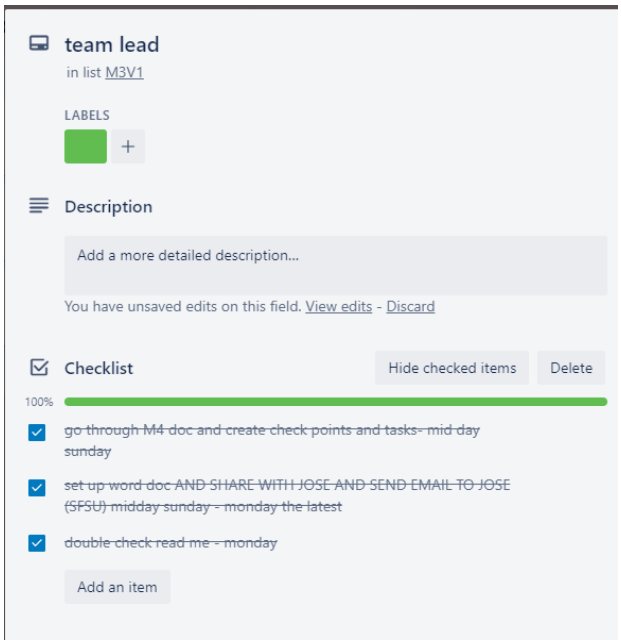


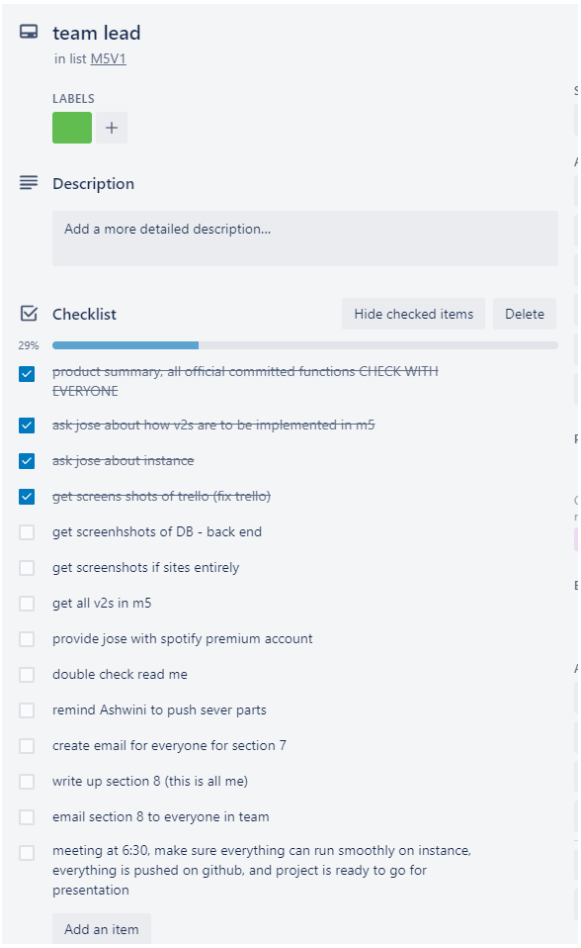

front end
in list M3V1
LABELS
Description
Add a more detailed description...
Checklist
Hide checked items
Delete
100%
add a faq about voting system - due Monday before meeting
add contact page back - before meeting monday!
description of difference between publics an private room (and FAQ) - before meeting monday!
Hirva- put filter on left side of search bar (search for rooms), then submit button on right of it - Monday
Move ToS out of create room - before meeting monday!
emphases on voting system! on landing page
room name spreads apart elements (in room) FIX
meta tag - ?
Add an item
Activity
Show details
RZ
Write a comment...


front end
in list M5V1
LABELS
Description
Add a more detailed description...
Checklist
Hide checked items
Delete
78%
Hirva-save toggle private public room to back end similarly to how to did with room name and genre. Done.
Hirva - Keep in touch with Luong about song search.
Bryan - Implement Spotify login.
Hirva - Check if there is a queue system through spotify that we can use.
add back contact us
Song Search - move the sync song search the is on the testing branch to the QA version of the site to replace what we have there.
Song Search - take the value from the select song, and store them, we are going to eventually need to send these to the queue on the backend database
Song Search - Find out how to get song information from the song that is selected so we can display it correctly.
Song Search - Find out how to build a recommended list of songs pulled from spotify.
Administrator panel - create a dashboard for administrators, this can be included in the logout popup in the navbar. Something simple, doesn't need to be fancy. Hirva -done
Administrator panel - Add an option to submit a user name for banning. This should tell the backend to update the users ban status, but for now, add it without any backend support. Also needs a comment box to leave a message about the ban.
Administrator panel - Add an option to submit a room id to delete that room. Hirva-done.
Malcolm - Rooms - make an error page/component for rooms that don't exist. Luong says "move on"
Footer - add link to Facebook/Twitter/Instrgram.
ERROR CHECKING
Try to specify this for coding style - The issues usually considered as part of programming style include the layout of the source code, including indentation; the use of white space around operators and keywords; the capitalization or otherwise of keywords and variable names; the style and spelling of user-defined identifiers, such as function, procedure and variable names; and the use and style of comments.
Malcolm - Review Horizontal Prototype for Jose comments and fix stuff.
Add Faqs for new features.
Rooms - Base rooms on their ID, do a get when entering room to get roomname and genre. Hirva-done
Users - add ban status
Landing - add scopes for stream ect.
Login - fix issue with renewing tokens.
Room - define public and private on backend and shown in rooms. Hirva - done
Search - Restrict Users from seeing private rooms in search, but allow administrators to see private rooms. Hirva -done
Join - Enable enter button in the search. Hirva -done
Join - omit private rooms from search. Hirva-done


back end
in list M3V2
LABELS
DUE DATE
Apr 14 at 11:59 PM COMPLETE
Description
Add a more detailed description...
Checklist
Hide checked items
Delete
100%
search api to get rooms and genres get 20 results (4 pages of 5)
get api to retrieve room
spotify api connectred !!!!!!!!!
api recommended rooms
section 4
data definitions section 1
tuong get search for songs on the site - after implementing Spotify api
Add an item
Activity
Show details
Write a comment...
Rebecca Zumaeta copied comment by Rebecca Zumaeta from card back end
Apr 7 at 12:05 AM
product prototype:
- home page
- basic search bar (back end integrated and working)
- insert end point (sign into Spotify and create a room) (saving data into database)
- image and image thumbnail (not sure will ask)


back end
in list M5V1
LABELS
Description
Add a more detailed description...
Checklist
Hide checked items
Delete
77%
building structure of all available tables referring to MDB diagram in M3 to then be implemented !!!!!!!!!!!! - Vishakha and Ashwini
table for voting - Vishakha
Create a participants table (userID, roomID) (Reference from Room and User tables) - Vishakha
add column in room table for private and public so that search doesn't search all private rooms - Vishakha
search for songs (Spotify API) - Luong
Integrate Spotify API stuff into the room - Ashwini
implement chat function (Ashwini) - done once after we have a few a functions are built
administrator - table so that we can - ban users - for later ----------Check landing page, look for function "retrieveCurrentUser" and store constant userInfo.displayname onto database.
Try to specify this for coding style - The issues usually considered as part of programming style include the layout of the source code, including indentation; the use of white space around operators and keywords; the capitalization or otherwise of keywords and variable names; the style and spelling of user-defined identifiers, such as function, procedure and variable names; and the use and style of comments.
delete rooms
Figure out implementation of queue
Figure out implementation of voting system.
Figure out implementation of sync.
Get, Post, delete API for the queues table, users table and vote table
Users table - need admin status, ban status, ban comment columes.
(Done, just need the finalized version of Room before pushing) SYNC/ Music player (Spotify API) - Luong
api/adds - it uses the id value when grabbing room information; ${id}, can you change it to the newly added room_id variable instead, since we are using that? -Bryan
api/vote - need song id, room id(instead of queue), user id, and vote_id.
api/votes - delete based on vote_id
api/votes - delete all votes based on room_id
api/votes - delete all votes based on song_id
api/queue - need queue_item_id, room id, and song id(need to be string)
api/queue - delete based on queue_item_id.
api/queue - delete all based on room_id
api/users - prevent duplicate user_ids.
api/adds - need population column.
api/adds - need to post updates for room, so maybe just a post that checks if that room_id already exists, and if it does, just update the other data?
API to retrieve all public rooms


team lead
in list M3V1
LABELS
Description
Add a more detailed description...
You have unsaved edits on this field. View edits - Discard
Checklist
Hide checked items
Delete
100%
go through M4 doc and create check points and tasks- mid day sunday
set up word doc AND SHARE WITH JOSE AND SEND EMAIL TO JOSE (SFSU) midday sunday - monday the latest
double check read me - monday
Add an item


team lead
in list M5V1
LABELS
Description
Add a more detailed description...
Checklist
Hide checked items
Delete
29%
product summary, all official committed functions CHECK WITH EVERYONE
ask jose about how v2s are to be implemented in m5
ask jose about instance
get screens shots of trello (fix trello)
get screenhshots of DB - back end
get screenshots if sites entirely
get all v2s in m5
provide jose with spotify premium account
double check read me
remind Ashwini to push sever parts
create email for everyone for section 7
write up section 8 (this is all me)
email section 8 to everyone in team
meeting at 6:30, make sure everything can run smoothly on instance, everything is pushed on github, and project is ready to go for presentation
Add an item

6. Team member's contributions

From: Rebecca Maria Zumaeta

Sent: Wednesday, May 19, 2021 11:34 PM

To: Luong Khai Dang <ldang2@mail.sfsu.edu>; Bryan Kenneth Fetner <bfetner@mail.sfsu.edu>; Ashwini Gururaj Managuli <amanaguli@mail.sfsu.edu>; Vishakha Tyagi <vtyagi@mail.sfsu.edu>; Hirva Maneshkumar Patel <hpatel11@mail.sfsu.edu>; Malcolm Angelo Santos De Villar <mdevillar@mail.sfsu.edu>

Subject: RE: CSC648-04 Spring 2021 Milestone 5 Team 06 - Team member contribution

Approved by Team Lead

From: Luong Khai Dang <ldang2@mail.sfsu.edu>

Sent: Wednesday, May 19, 2021 11:34 PM

To: Rebecca Maria Zumaeta <rzumaeta@mail.sfsu.edu>; Bryan Kenneth Fetner <bfetner@mail.sfsu.edu>; Ashwini Gururaj Managuli <amanaguli@mail.sfsu.edu>; Vishakha Tyagi <vtyagi@mail.sfsu.edu>; Hirva Maneshkumar Patel <hpatel11@mail.sfsu.edu>; Malcolm Angelo Santos De Villar <mdevillar@mail.sfsu.edu>

Subject: Re: CSC648-04 Spring 2021 Milestone 5 Team 06 - Team member contribution

Name: Luong Dang

ID: 916 116 778

Group 6

Role: Backend Engineer

Milestone 1

- Setup server and deployment for M0.
- Wrote sections 7 and 3
- Discussed concepts, features, and coding implementations into our project.
- Gave great concepts and ideas for other members to bounce from.

Milestone 2

- Build working versions of sites using Spotify API and other functions we would use later in development.
- Wrote section 5.
- Draw EER Diagram

Milestone 3

- Assisted in the document when notes were given.
- Built functions that will be implemented when back end is implemented such as Spotify API. Contributed to api functions that have and will be implemented, such as the function of album covers to be shown for rooms and within rooms (player). Attended all scheduled meetings and contributed into conversation when wanting to share thoughts and experiences. Was open to feedback and reported back to the team lead. Built mini sights to test functions that can be implemented into ours.

Milestone 4

- Refactor landing page and room page to implement Spotify API.
- Assisted in building out all tables within the DB diagram like table for voting, assisted with document in section 4 and 3.
- Assisted in confirming the committed functions and to original non-functional specs.

The number of commits I have made on Github development branch thus far are 3. During Milestone 1, I committed everything in one commit. During Milestone 3, I committed the completed refactor version of the website in one push. During Milestone 4, I committed Implementation of Spotify Web Playback in one push. Even though I have the least number of commits in the team, that was because I mostly work with a team member in the team.

LUONG DANG

Backend Engineer of Team 06

ldang2@mail.sfsu.edu

From: Rebecca Maria Zumaeta

Sent: Wednesday, May 19, 2021 11:14 PM

To: Bryan Kenneth Fetner <bfetner@mail.sfsu.edu>; Ashwini Gururaj Managuli <amanaguli@mail.sfsu.edu>; Vishakha Tyagi <vtyagi@mail.sfsu.edu>; Hirva Maneshkumar Patel <hpatel11@mail.sfsu.edu>; Malcolm Angelo Santos De Villar <mdevillar@mail.sfsu.edu>; Luong Khai Dang <ldang2@mail.sfsu.edu>

Subject: RE: CSC648-04 Spring 2021 Milestone 5 Team 06 - Team member contribution

Approved by Team Lead

From: Bryan Kenneth Fetner <bfetner@mail.sfsu.edu>

Sent: Wednesday, May 19, 2021 11:13 PM

To: Rebecca Maria Zumaeta <rzumaeta@mail.sfsu.edu>; Ashwini Gururaj Managuli <amanaguli@mail.sfsu.edu>; Vishakha Tyagi <vtyagi@mail.sfsu.edu>; Hirva Maneshkumar Patel <hpatel11@mail.sfsu.edu>; Malcolm Angelo Santos De Villar <mdevillar@mail.sfsu.edu>; Luong Khai Dang <ldang2@mail.sfsu.edu>

Subject: Re: CSC648-04 Spring 2021 Milestone 5 Team 06 - Team member contribution

Hello Team 6,

The following is my list of contributions to the Sync project regarding specific sections:

- Milestone 1
    - Executive Summary, half of the Use Cases.
    - Contributed to functional requirements and non-functional requirements.

- Milestone 2
    - Reviewed data definitions and functional requirements.
    - Did initial UI mock-ups, ultimately redone by Rebecca.
    - Did UML diagrams with assistance from Malcolm.
- Vertical prototype
    - Built out the Create room component and modified other components such as the navbar.
    - General layout placement and enhancing the second version of the prototype, excluding the search.
- Milestone 3
    - Reviewed and gave critics on overall document.
- Horizontal prototype
    - Managed front end team through Trello and communication through discord chat and voice chat.
    - Built out the room and related components for front end presentation.
    - Helped with layout on landing and join pages.
- Milestone 4
    - Helped with building the Usability Task Tests, and the QA Tests
    - Did general reviews and critiques on the remaining document.
- Milestone 5
    - Did a general review of the document.
- Final Web App
    - Continued to manage front end team through Trello.
    - Implemented the login system utilizing Spotify login, based off Luong's original design.
    - Implemented Spotify search api, based off Luong's original design, to match horizontal prototype design.
    - Implemented queue system to work with Sync api.
    - Implemented vote system to work with Sync api.
    - Implemented the Music Player from Luong into the rooms and adding updating songs for it.
    - Implemented real-time chat that Ashiwini provided into the room.
    - Helped with other minor aspects of the site.

My interaction with github has grown to 137 commits.

If there are any questions or commits, feel free to ask.

Bryan Fetner

Front End Team Lead

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

Sent: Wednesday, May 19, 2021 11:03 PM

To: Ashwini Gururaj Managuli <amanaguli@mail.sfsu.edu>; Vishakha Tyagi <vtyagi@mail.sfsu.edu>; Hirva Maneshkumar Patel <hpatel11@mail.sfsu.edu>; Malcolm Angelo Santos De Villar <mdevillar@mail.sfsu.edu>; Bryan Kenneth Fetner <bfetner@mail.sfsu.edu>; Luong Khai Dang <ldang2@mail.sfsu.edu>

Subject: RE: CSC648-04 Spring 2021 Milestone 5 Team 06 - Team member contribution

Approved by Team Lead

From: Ashwini Gururaj Managuli <amanaguli@mail.sfsu.edu>

Sent: Wednesday, May 19, 2021 11:02 PM

To: Rebecca Maria Zumaeta <rzumaeta@mail.sfsu.edu>; Vishakha Tyagi <vtyagi@mail.sfsu.edu>; Hirva Maneshkumar Patel <hpatel11@mail.sfsu.edu>; Malcolm Angelo Santos De Villar <mdevillar@mail.sfsu.edu>; Bryan Kenneth Fetner <bfetner@mail.sfsu.edu>; Luong Khai Dang <ldang2@mail.sfsu.edu>

Subject: Re: CSC648-04 Spring 2021 Milestone 5 Team 06 - Team member contribution

Hello Team 06!

My role in this project was to be the back-end lead.

I believe I fulfilled the role of a back-end lead by:

- Communicating with the front-end lead often to discuss what they require from the back-end.
- Conducting meetings and discussing what needs to be done with the back-end members on a weekly basis.
- Communicating with the team lead with timely updates and progress.

For the document, I contributed to writing out most of the functional requirements and placing them in particular sections, writing some points for the non-functional requirements and the main data items and entities for M1 and M2.

Here are my contributions to the development of the site:

- Connected Django to our MySQL database.
- Built out the APIs to retrieve information from the database for the vertical prototype.
- Assisted with the design of the NavBar.
- Deployed the site to a virtual machine on AWS and managed it with Amazon EC2.
- Connected Django to React using Axios with Hirva (Front-end member).
- Built out the corresponding REST APIs to GET, POST, DELETE, PATCH information from/to the database on Django.
- Built out the same for the rooms, queues, votes, users functionalities.
- Implemented the real-time group chat in the rooms using WebSocket communications.
- Constantly updated the instance to work with new installations and new developments and deployed each development.

The number of commits I made to our repository is 16 commits (including ubuntu commits every time I pushed from the instance)

Thank You.

Ashwini Managuli

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

Sent: Thursday, May 20, 2021 11:28

To: Vishakha Tyagi <vtyagi@mail.sfsu.edu>; Hirva Maneshkumar Patel <hpatel11@mail.sfsu.edu>; Malcolm Angelo Santos De Villar <mdevillar@mail.sfsu.edu>; Ashwini Gururaj Managuli <amanaguli@mail.sfsu.edu>; Bryan Kenneth Fetner <bfetner@mail.sfsu.edu>; Luong Khai Dang <ldang2@mail.sfsu.edu>

Subject: RE: CSC648-04 Spring 2021 Milestone 5 Team 06 - Team member contribution

Approved by Team Lead

From: Vishakha Tyagi <vtyagi@mail.sfsu.edu>

Sent: Wednesday, May 19, 2021 10:56 PM

To: Rebecca Maria Zumaeta <rzumaeta@mail.sfsu.edu>; Hirva Maneshkumar Patel <hpatel11@mail.sfsu.edu>; Malcolm Angelo Santos De Villar <mdevillar@mail.sfsu.edu>; Ashwini Gururaj Managuli <amanaguli@mail.sfsu.edu>; Bryan Kenneth Fetner <bfetner@mail.sfsu.edu>; Luong Khai Dang <ldang2@mail.sfsu.edu>

Subject: Re: CSC648-04 Spring 2021 Milestone 5 Team 06 - Team member contribution

Hello Sickgatorz (Team 06)

Here are the contributions I made to the project.

As a backend member,

- Build most of the database tables
- Made the ERD diagram for the application
- Helped team members fix the bugs
- Build the backend part of the login registration
- Wrote a document to Setup VSCode for Python App Development using Github private repo

For the document,

- Drawn out all the use case diagrams
- Made deployment and network diagram

- Came up with user interface design for the application
- Written most of the data definitions for M2
- Written whole coding style section

In addition to these, I have always completed the work assigned to me either by team lead or backend lead. Also, I have 8 commits to the Github repository.

Please reach out if you have any questions.

Thank you

Vishakha Tyagi

(Backend Member)

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

Sent: 19 May 2021 22:42

To: Hirva Maneshkumar Patel <hpatel11@mail.sfsu.edu>; Malcolm Angelo Santos De Villar <mdevillar@mail.sfsu.edu>; Ashwini Gururaj Managuli <amanaguli@mail.sfsu.edu>; Bryan Kenneth Fetner <bfetner@mail.sfsu.edu>; Luong Khai Dang <ldang2@mail.sfsu.edu>; Vishakha Tyagi <vtyagi@mail.sfsu.edu>

Subject: RE: CSC648-04 Spring 2021 Milestone 5 Team 06 - Team member contribution

Approved by team lead

From: Hirva Maneshkumar Patel <hpatel11@mail.sfsu.edu>

Sent: Wednesday, May 19, 2021 10:41 PM

To: Rebecca Maria Zumaeta <rzumaeta@mail.sfsu.edu>; Malcolm Angelo Santos De Villar <mdevillar@mail.sfsu.edu>; Ashwini Gururaj Managuli <amanaguli@mail.sfsu.edu>; Bryan Kenneth Fetner <bfetner@mail.sfsu.edu>; Luong Khai Dang <ldang2@mail.sfsu.edu>; Vishakha Tyagi <vtyagi@mail.sfsu.edu>

Subject: Re: CSC648-04 Spring 2021 Milestone 5 Team 06 - Team member contribution

Hello, Team 06 aka Sick Gatorz,

As you all know I joined this team from Milestone 1. My role was to be a frontend member.

a) Here is the list of contributions I believe I made towards the team and the project:

- Code Part

1. Connected Axios with Django, basically frontend with backend.
2. Helped with deploying the application using AWS
3. Made most API calls using Axios
4. Implemented the search bar with options to search based on room name and genre
5. Displayed recommended rooms on the home page
6. Added images to the search results of the rooms and recommended rooms.
7. Added links to join the rooms
8. Added dropdown to navbar to logout.
9. Added the modal to accept TOS while logging in.
10. Implemented admin panel
11. Added options to the navbar for the admins
12. Made a functionality to let users not see or search private rooms
13. Implemented functions to make admin control both public and private rooms
14. Added delete rooms functionality for admins
15. Added ban a user functionality for admins
16. Connected backend to the contact us form
17. Modified database schema as and when needed.

- Documentation Part

1. Helped in writing requirements initially
2. Helped in writing data entities and definitions
3. Also helped in making database architecture
4. Researched various other similar sites and made the competitive analysis
5. Helped in making storyboards as it was a part of the frontend work.
6. Helped in making various diagrams in the documentations
7. Also, helped in testing out the application

b) The number of submissions I made to the Github development branch is 55 commits.

Please let me know if you have any questions or concerns.

Thank you,

Hirva Patel.

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

Sent: 20 May 2021 11:08

To: Malcolm Angelo Santos De Villar <mdevillar@mail.sfsu.edu>; Ashwini Gururaj Managuli <amanaguli@mail.sfsu.edu>; Bryan Kenneth Fetner <bfetner@mail.sfsu.edu>; Luong Khai Dang <ldang2@mail.sfsu.edu>; Hirva Maneshkumar Patel <hpatel11@mail.sfsu.edu>; Vishakha Tyagi <vtyagi@mail.sfsu.edu>

Subject: RE: CSC648-04 Spring 2021 Milestone 5 Team 06 - Team member contribution

Approved by Team Lead

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

Sent: Wednesday, May 19, 2021 10:38 PM

To: Rebecca Maria Zumaeta <rzumaeta@mail.sfsu.edu>; Ashwini Gururaj Managuli <amanaguli@mail.sfsu.edu>; Bryan Kenneth Fetner <bfetner@mail.sfsu.edu>; Luong Khai Dang <ldang2@mail.sfsu.edu>; Hirva Maneshkumar Patel <hpatel11@mail.sfsu.edu>; Vishakha Tyagi <vtyagi@mail.sfsu.edu>

Subject: Re: CSC648-04 Spring 2021 Milestone 5 Team 06 - Team member contribution

Hello Team06 aka Sick Gatorz,

I am part of the Frontend team and is one of the frontend developers. As we are responsible for the User experience, I am mainly tasked on working and helping with the user experience and user interface, along with helping in the functionalities and testing the frontend work. Below is the bulleted format for the details of my contribution:

App Development:

- Created Landing page.
- Created layout of room page.
- Created Footer.
- Created Creators page.
- Created Contact Us page.
- Created the FAQ component.
- Handled redirect of pages if it does not exist.
- Created the TOS component to be used for authentication check in Landing page.
- Created .css files for each page and components I specifically started and handled.
- Helped implementing functionalities for each page.
- Testing functionalities (both local and in running instance) to make sure it still works after every change

Document:

- Mainly worked on Functional requirements, non-functional requirements, and List of main data items and entities.
- Contributed to features and concepts.
- Helped on deciding about small and big changes to be written in the document.
- Helped on grammatical errors to improve readability.
- Improved Data definitions
- Identified key risks to be able to identify what needs to be prioritized.

As of writing this email, I have 42 commits with the most active branch.

Feel free to leave a feedback or ask questions.

Thank you,

Malcolm Angelo De Villar

Frontend Developer of Team06

mdevillar@mail.sfsu.edu

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

Sent: Wednesday, May 19, 2021 10:34 PM

To: Ashwini Gururaj Managuli <amanaguli@mail.sfsu.edu>; Bryan Kenneth Fetner <bfetner@mail.sfsu.edu>; Luong Khai Dang <ldang2@mail.sfsu.edu>; Hirva Maneshkumar Patel <hpatel11@mail.sfsu.edu>; Vishakha Tyagi <vtyagi@mail.sfsu.edu>; Malcolm Angelo Santos De Villar <mdevillar@mail.sfsu.edu>

Subject: CSC648-04 Spring 2021 Milestone 5 Team 06 - Team member contribution

Hello Team 06 aka Sick Gatorz,

As you all know, for this team project, my role was to be team lead. In my opinion this role entails, the obvious: leading the team, but more specifically: provided guidance, provided instruction, provided direction, a middle man between team and CTO, overseer, time manager and make sure execution is done correctly and accordingly.

Here are the list on contributions I believe I made towards the team and project:

- Lead team though out the semester
    - Lead conversation regarding project
    - Scheduled 2 weekly meetings for updates and task assigning
    - Assigned tasks to be done by each meeting.
    - Set tasks as a check list via Trello and managed Trello
    - Created and hosted meeting with CTO
    - Communicated with CTO with any concerns or questions regarding the project
    - Submitted each assignment via email to CTO
- Documentation
    - Wrote out most of sections in documentation
    - Oversaw sections I did not write,
    - Pushed Documentation to GitHub
    - Communicated with CTO questions or concerns about documentation
    - Email CTO document link each millstone

The number of submissions I had made to our GitHub development branch thus far are 29 commits.

Please let me know if you have questions or concerns. Please respond with your list of contributions as well.

Thank you,

Rebecca Zumaeta

Team Lead of Team 06

rzumaeta@mail.sfsu.edu

7. Post Analysis

Creating SYNC was an experience we are all very grateful to have; a huge learning experience. We all learned many new things, such as learning new technologies and languages. SYNC made us push ourselves to go farther than we usually do and we are so proud of the outcome. Although the project is marvelous, there are noteworthy challenges made during the creation of SYNC and better ways for it to have been dealt with to have an even better outcome. The three major challenges we as a team dealt with was being too ambitious with not having enough time, lack of preparation, and communication. Each challenge only pushed us harder to create great outcomes, but there are ways we could have solved the issue, possibly in a more efficient manner.

When we teamed up, we were excited to create a project of our own and design it to be whatever we'd like. Ambitious, we wrote out a million ideas that we really thought were possible in the time frame. If we had really honed down a few special features and perfected those features then we could have a better outcome. Having a better understanding of the time frame in the class and what is needed in each milestone would have then produced a, possibly a simpler, project that would not have been so complicated and over our heads.

Team 06 worked with some technologies we have never worked with before. Most if not all of us had never worked with React, AWS, Axios and others that were implemented into our project. It was great that the team was able to learn it and work with these technologies, but it was a huge learning curve and took much time to learn, taking time away from developing the project. To avoid losing time, we could have just chosen technologies that we were most familiar with rather than pushing ourselves thin trying to learn many things at once.

Lastly, although our team dynamic is very communicative and uplifting, it still can be improved. As a team lead, communication is something I knew would make or break a team. I know I could have been pushy at times and it may have been interpreted as something else, but intentions were to keep the team moving along and improving each milestone. I believe communication between the front and back end could have been slightly more transparent. As team lead I would have made front and back end meetings mandatory once a week and would have attended them to oversee the process and possibly maneuver the conversion when need be. I'd even ask for a meeting note, for the meetings I could not attend. The meeting notes could even be for the opposite team to review to then discuss and grow off of.

Having this experience is a great one as we learned the do's and don'ts of developing a web based project and how it really is like to work as a team. I am forever grateful to my teammates for an awesome project, and wonderful semester.

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	Luong Dang	ldang2@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	Ashwini Managuli	amanaguli@mail.sfsu.edu

Milestone 1

03/05/2021

History Table

Version	Date	Notes
M1V1	03/05/2021	
M1V2	3/8/21	

Table of Contents

1. Executive Summary..... 3
2. Main Use Cases.....4
3. List of main data items and entities..... 12
4. Initial list of functional requirements.....14
5. List of non-functional requirements.....18
6. Competitive Analysis.....22
7. High-level System Architecture and Technologies Used..... 25
8. Team Contributions..... 27
9. Checklist..... 30

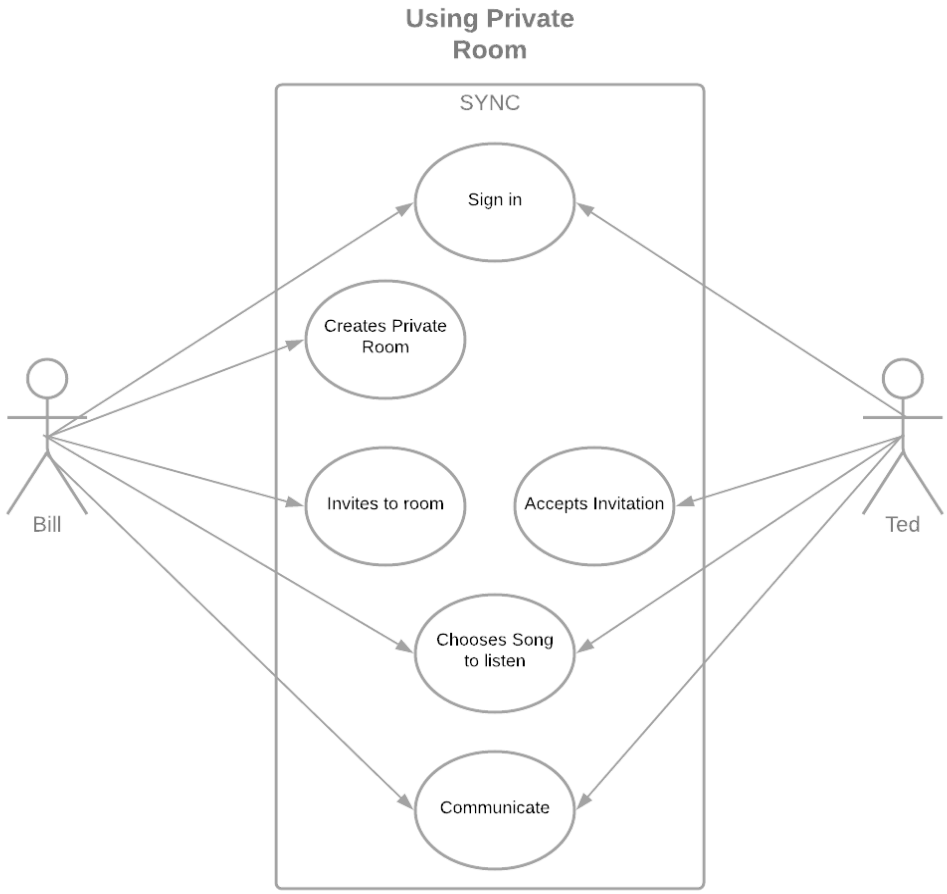
1. Executive Summary

The inception of the internet was brought with the idea of bringing people together. The effort towards such means is ever pursued, and we intend to continue this endeavor. A device for bringing together individuals has often been music; be it concerts and shows alike. We took a look at the current music offerings on the internet, and there are plenty, but they don't offer quite the focus on connecting people and communities. More often than not they just slap together a playlist and a chatroom and leave it at that. Our web application, aptly named Sync, seeks to give users a more interconnected experience where they can control together what is being played and make their experience a more evolving one, not just limiting them to a specified category and allowing them to easily group together or splinter off to their own desired areas.

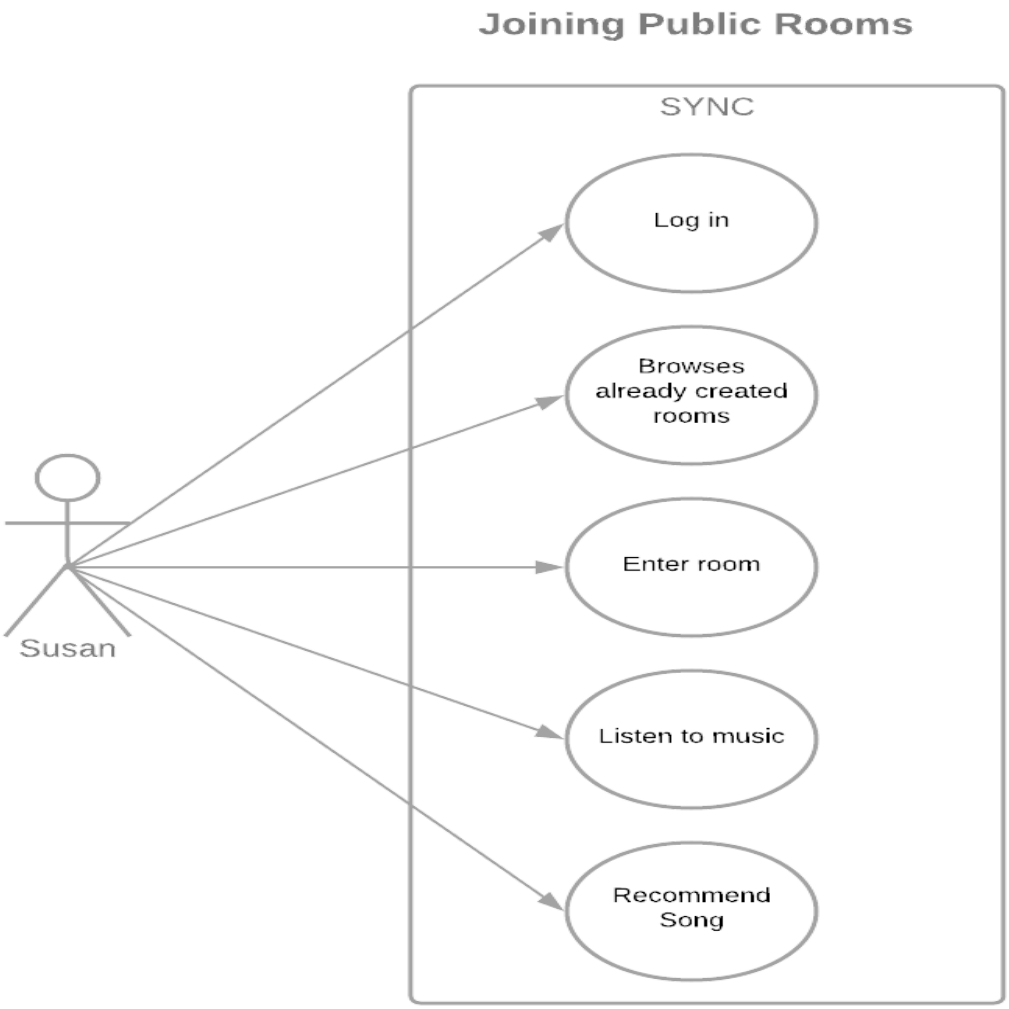
Now, you might be wondering, what does this mean exactly? Simply put, we put the control in the hands of the users. Most often the case with these types of web applications, is the control is often put in the hands of a single "DJ" while others in the room must simply idly sit as a single person chooses what is played. What we propose with Sync is that instead there is a more communal effort in song selection. This we believe will create a more accurate depiction of the tastes of those within the listening room, as well as create a more active participative experience due to the allowance of suggesting and voting for music that will be played. No more will they only sit around, but now they can be in control. This provides a n avenue for those to also share music that may be considered more "niche", or past songs that don't get the attention deserved anymore. Along with this we intend to give users the ability to connect with each other through means of commenting with the music room as well as personal messaging. Though not a unique feature, but an important one to keep people connected. The creation of rooms will be easily accessible and dynamic to follow with the evolving nature of the rooms themselves. With these features combined we believe that we can bring people together through their music preferences.

2. Main Use Cases

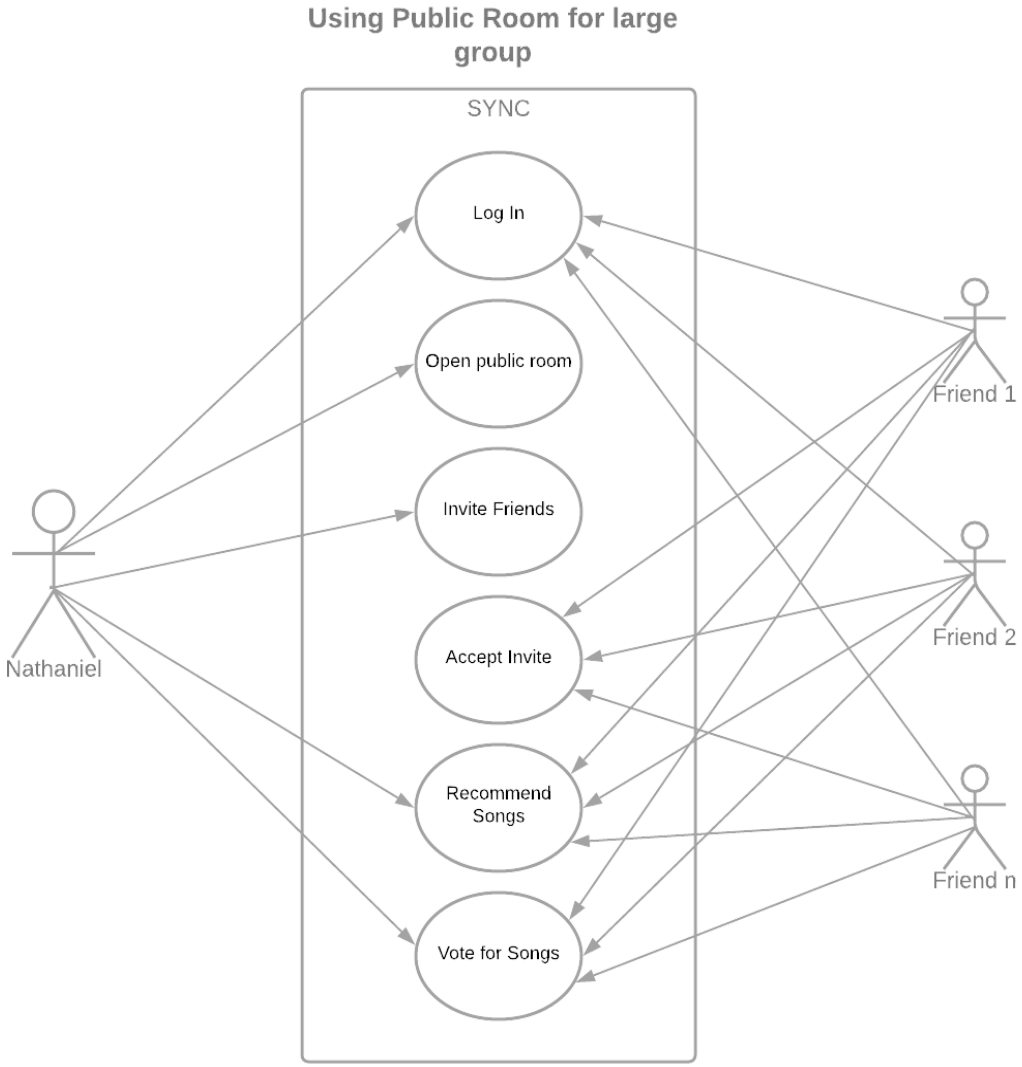
Title:	Using Private Rooms
Actors:	Bill and Ted (Friends who share similar music preferences)
Description:	Bill and Ted both want to listen to music together and are unable to spend time together in person. To remedy this, they both sign into SYNC using their Spotify accounts. Bill sees an option to create either a public room or a private room. Since he wants a room for Ted and himself, he chooses a private room to restrict any other users from entering. After creating the room he sends an invite to Ted. Ted receives the link to the room through his direct message and clicks on it to join Bill's room. Here they choose various songs to listen to while using the included commenting system to communicate with each other and talk about the music they are listening to.



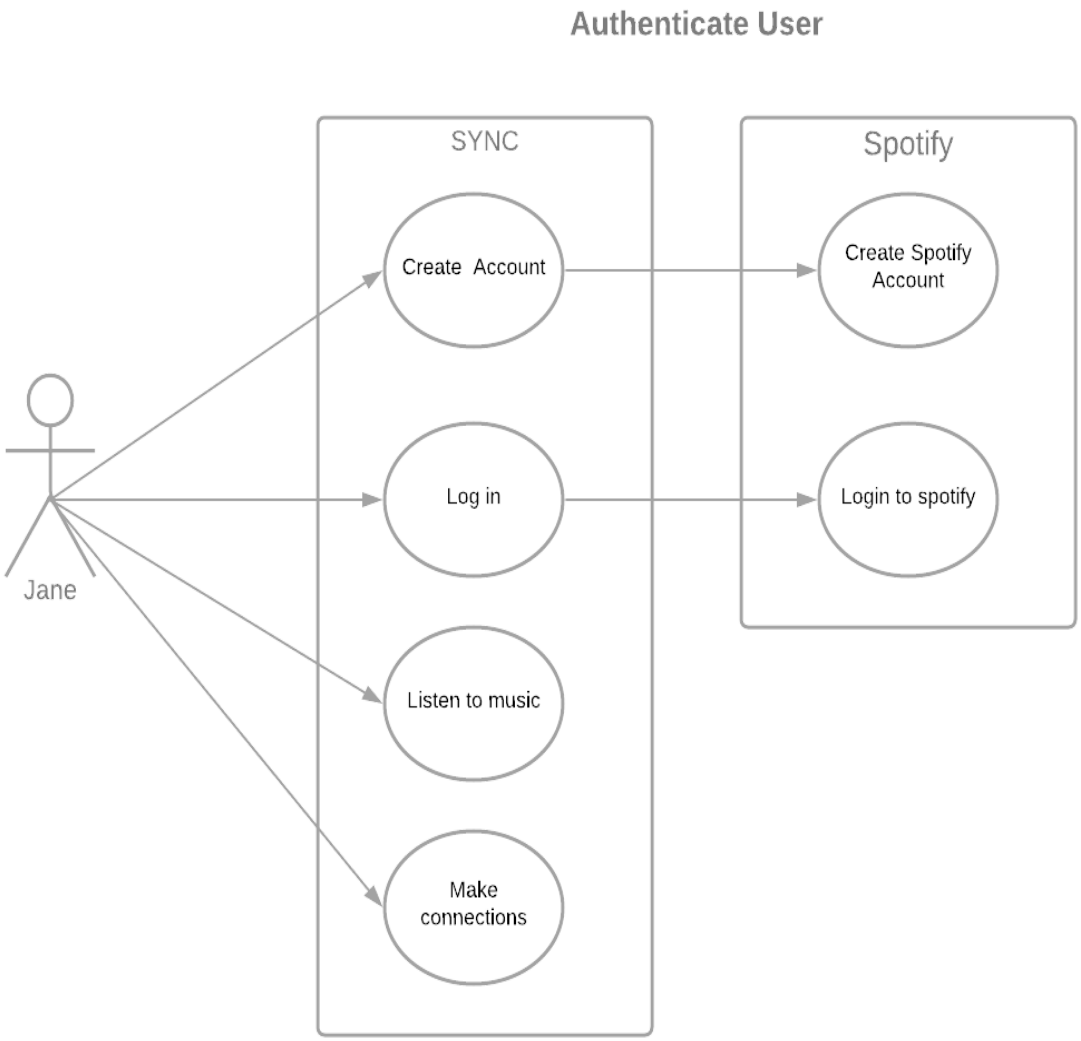
Title:	Joining public rooms
Actors:	Susan (Lonely music enthusiast)
Description:	After spending months home alone due to COVID-19 stay at home restrictions, Susan decides that she needs to fulfill her desire for social interaction. As a fan of music, she logs into SYNC using her Spotify account and browses the currently available public rooms through the search bar on the home page. Susan spots a room that has been playing songs from one of her favorite genres, and she chooses to enter the room. When entering the room she immediately starts hearing music with a visual of the current song playing. As well, she finds a community in the room's chat where she communicates with other listeners about her favorite songs.



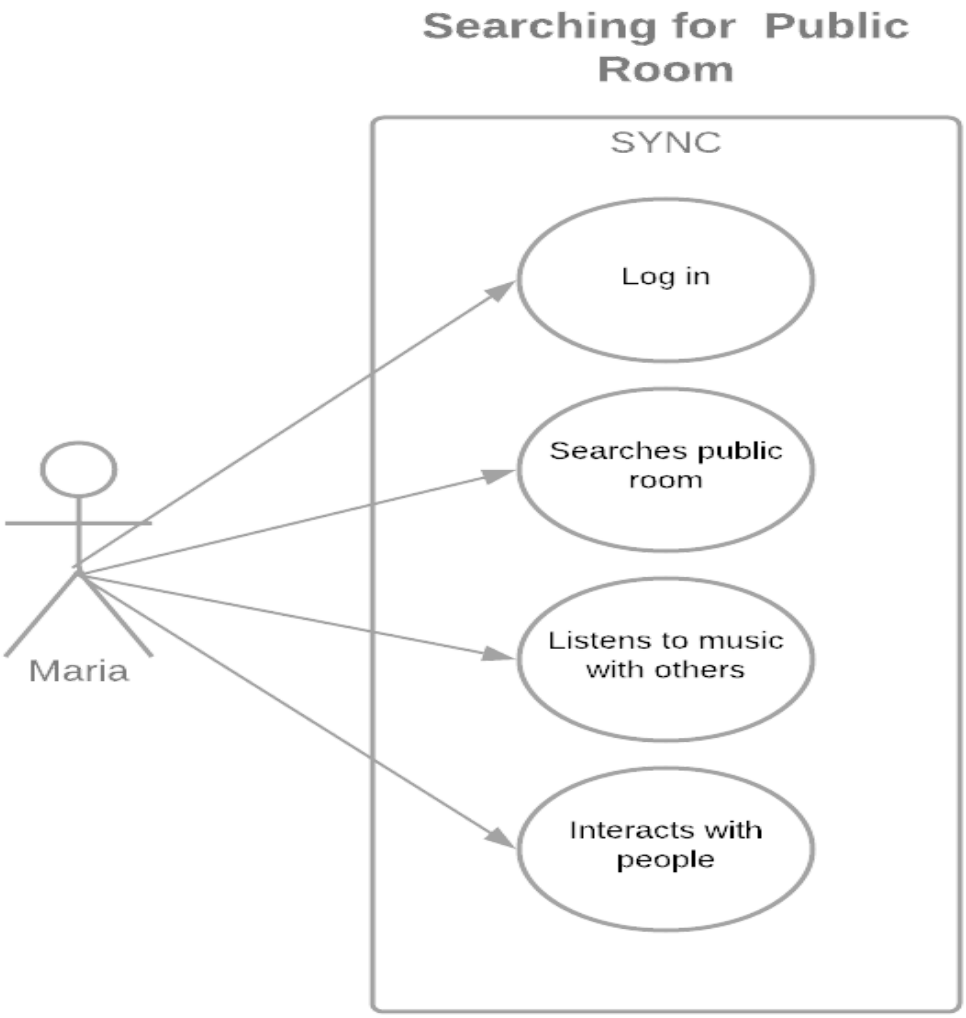
Title:	Using public room for large number of users
Actors:	Nathaniel and friends (Party friends)
Description:	COVID-19 is over, everyone has their vaccines. Now Nathaniel wants to throw a party at his parents' house while they are out of town. He invites all his friends to the house, and for music he decides to hook up his computer to the house sound system and loads up SYNC. He opens up a public room and gets a generated shared link. He airdrops the link to everyone at the party for them to join the public room. Now everyone at the party can recommend and vote through SYNC for the songs being played at the party, easily catering to those in attendance and keeping the vibes flowing.



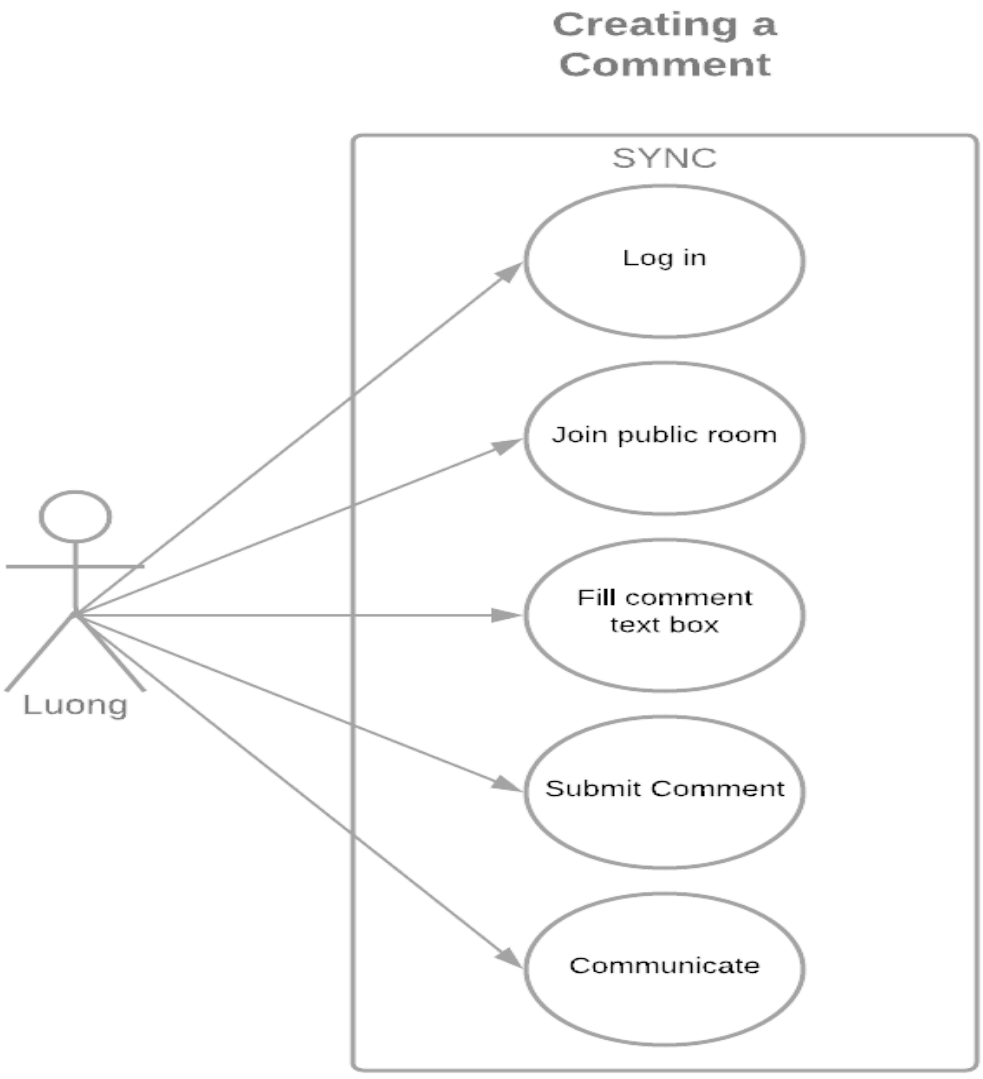
Title:	Authenticate user
Actors:	Jane (New user)
Description:	Jane wants to be able to create an account with SYNC so she can meet new people and listen to music together. After finding out that she needs a Spotify account to use SYNC, she creates a Spotify account. Once she has all the prerequisites, a Spotify account, she is able to navigate to the login page of SYNC to login in with her Spotify account. Login in now allows her to use SYNC, allowing her to listen to music with others and creating connections.



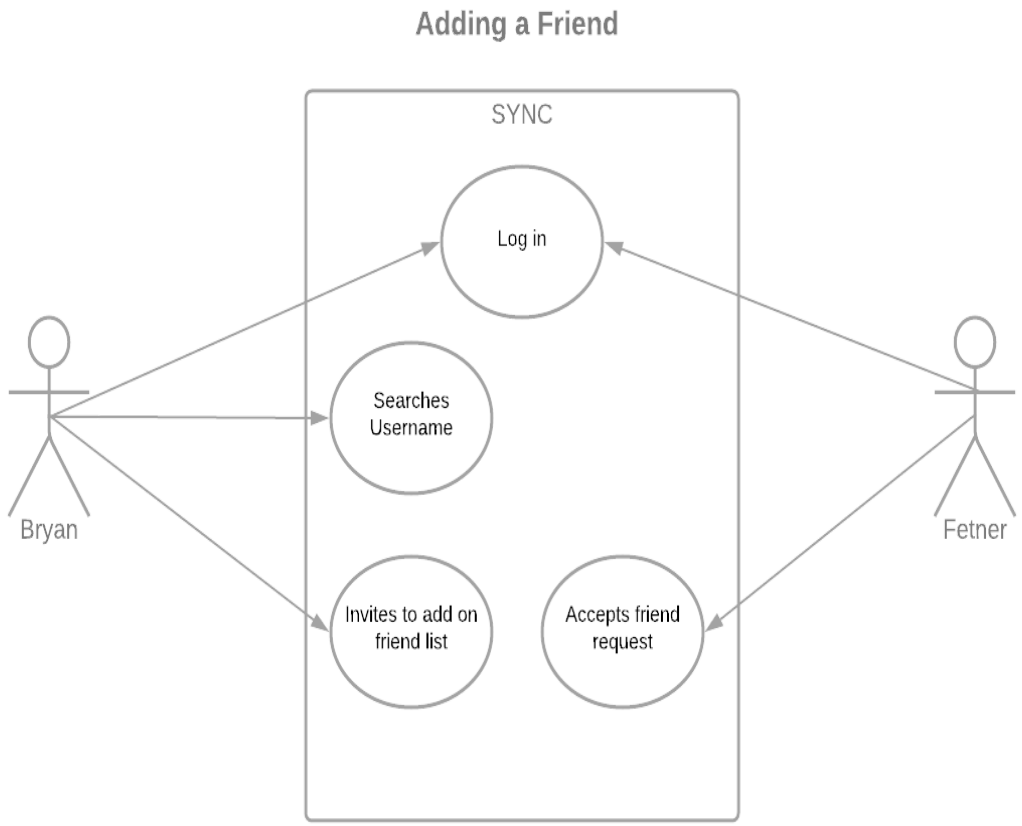
Title:	Searching for a public room
Actors:	Maria (Registered user)
Description:	Maria is a student who enjoys listening to music while studying. She would usually listen to live music streams on YouTube but she wants to make friends she can message and interact more with. When discovering SYNC she came to a realization that she can join public rooms allowing her to listen to music with others in real time. Using the search bar on the main page Maria searches for public rooms by entering the term "Lo-Fi Hiphop". She is now offered a selection of rooms featuring her search term. From here she selects the most appealing room where she can then interact with other people in the room and gets to know them.



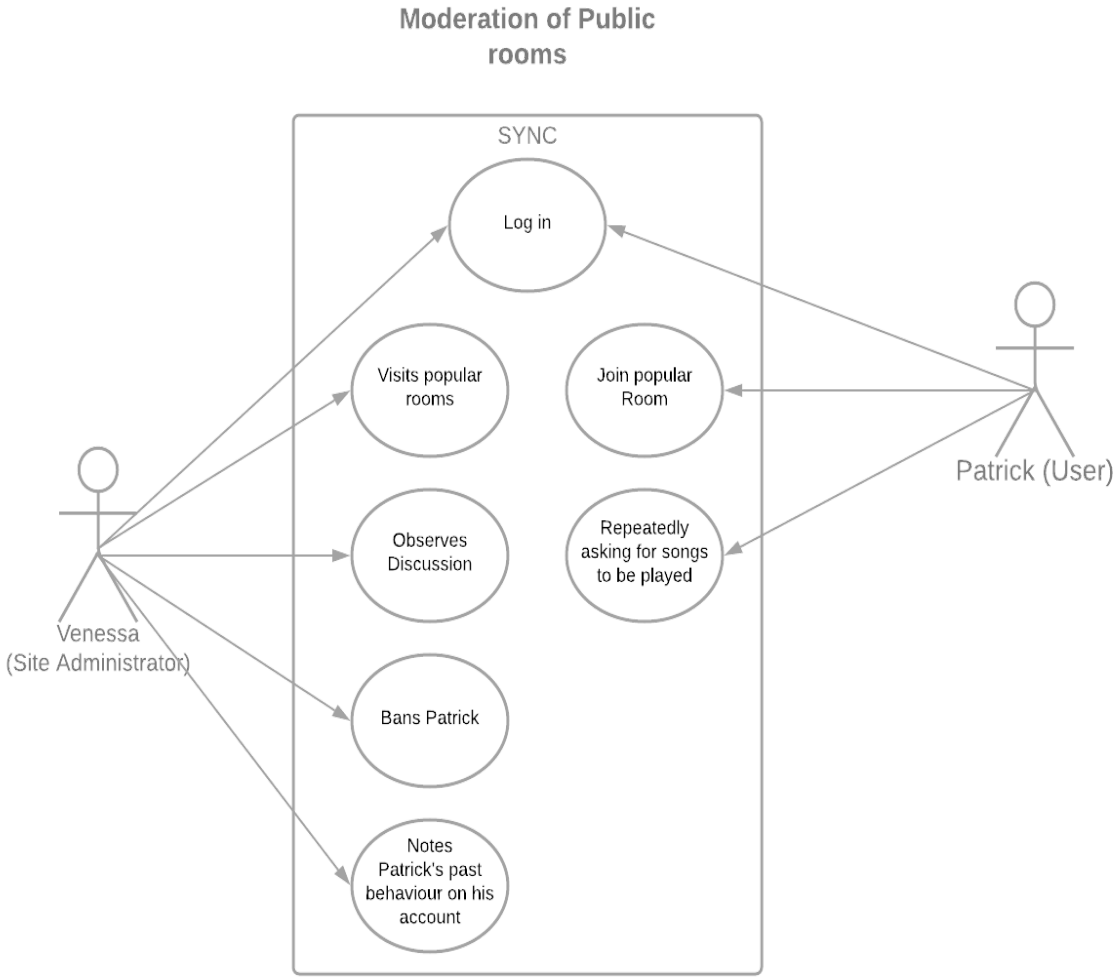
Title:	Creating a comment
Actors:	Luong (Room attendee)
Description:	Luong is a game fanatic who enjoys showing off his vast knowledge in video game soundtracks. He joins a public room that is playing the Cyber Punk Soundtrack. To show off his knowledge, Luong uses the comment text box option to comment on the current song and history of the artist. He then submits comments for others to react to. Luong now can communicate with others using comments within rooms.



Title:	Adding a friend through profile search
Actors:	Bryan (Registered user), Fetner (Registered user, Friend)
Description:	Bryan and Fetner are long term friends that miss each other during the COVID-19 pandemic. They decide to spend time together by listening to music through SYNC. They login into SYNC with their premium Spotify accounts. Bryan searches up Fetner's username to go to his profile page. Once he is on Fetner's profile page, Bryan sends an invite to be on Fetner's friend list. Fetner accepts Bryan's friend request and also sends a friend request to Bryan to be on his friend list. They are now added to each other's friend list.



Title:	Moderation of Public Rooms
Actors:	Vanessa (Site Administrator), Patrick (Registered User)
Description:	Vanessa is an administrator on SYNC. Due to her love of the site, she spends her free time moderating chat rooms. She visits the more popular rooms on the site to keep an eye on discussion. In a certain room she notices a user named Patrick repeatedly asking for a song to be played, clogging the chat with his requests. In order to give others a voice in the chat room, she gives Patrick a 10-minute ban by selecting his name to open a sub-menu which gives her this option. She does this to prevent him from spamming the chat for the time and discourages him from continuing this behavior. She is also given an option to make a note on his account regarding her reason for the ban for future reference. Also, Patrick receives a notification that he has been banned from commenting for the next 10 minutes.



3. List of main data items and entities

- Username
  - An entity where a user is identified uniquely.
- Logged in user
  - A user that is logged in on their Spotify account.
- Room host
  - A logged in user who created a room that has access to all features of the website, such as being able to chat, choose what song to play next, etc.
- Listeners of room
  - It shows the number of users that are currently listening in the room.
- Voters
  - Users who can vote on a song to play next while in a room public, communal, or private.
- Current song
  - This is the song that is currently playing in a room.
- Song queue
  - This shows the list of songs that are currently in line for listening.
- Spotify account information
  - This is the information needed to use SYNC website. The information will be used to customize recommended rooms.
- Podcasts
  - Might have to find a loophole from Spotify API since it's disabled.
- Artists
  - This shows who the artist of the song that is currently playing in the room.
- Song Title
  - This shows the name of the song currently playing in the room.
- Chats
  - This is accessible by anyone in the room and is used to communicate with each other.
- Public rooms
  - A room that is hosted by a logged in user that is accessible to any logged in user.
- Communal rooms
  - A room that is public that a logged in user will be put on randomly based on most listened genres.
- Private rooms
  - A room that is hosted by a logged in user that is invite only.
- Playlists
  - This shows the playlists that the logged in user has in their Spotify account.
- Genre preferences
  - This shows the genre preferences selected by each logged in user.
- Most listened songs
  - This shows the most listened songs by each logged in user.

- Most liked songs
  - This shows the most liked songs by each logged in user.
- Recommended rooms
  - This is a room where the website will recommend for the logged in user based on their song, genre, or artist choice.
- Profile
  - This is where the logged in user can change profile picture, and any public information that they want to display.
- Friends
  - These are the connections that are added by the logged in user.
- Friends list
  - This shows the list of added connections of the logged in user.

4. Initial list of functional requirements

Unregistered Users

1. Unregistered Users shall be able to log into their Spotify Premium.
2. Unregistered Users shall be able to access the homepage of the website.
3. Unregistered Users shall be able to access the About Us of the website.
4. Unregistered Users shall be able to access the FAQ of the website.
5. Unregistered Users shall be able to access the Contact page of the website.
6. Unregistered Users shall be able to get access to technical support.

Registered Users

General

7. Registered Users shall have a premium Spotify account.
8. Registered Users shall be able to login into their Spotify Premium.
9. Registered Users shall be able to listen to music in real time.
10. Registered Users shall be able to access the Homepage of the website.
11. Registered Users shall be able to access the About Us of the website.
12. Registered Users shall be able to access the FAQ of the website.
13. Registered Users shall be able to access the Contact page of the website.
14. Registered Users shall be able to get access to technical support.
15. Registered Users shall be able to export the playlist of the room.
16. Registered Users shall be able to edit their profile.
17. Registered Users shall be able to change their SYNC usernames.
18. Registered Users shall be able to change their SYNC profile picture.
19. Registered Users shall be able to change any information under the Profile page.
20. Registered Users shall be able to change status offline.
21. Registered Users shall be able to change status online.
22. Registered Users shall be able to report other users for misconduct.
23. Registered Users shall be able to share a room link though social media.
24. Registered Users shall be able to share link through Direct Message
25. Registered Users shall be able to share link through email
26. Registered Users shall be able to logout.

Host

27. Registered Users that create a room shall have the status of host.
28. Registered Users as hosts shall be able to name the room.
29. Registered Users as hosts shall be able to generate playlists.
30. Registered Users as hosts shall be able to control the music queue.
31. Registered Users as hosts shall be able to pause currently playing songs.
32. Registered Users as hosts of a room shall be able to disable sharing
33. Registered Users as hosts shall be able to set a limit to the number of users in the room.
34. Registered Users as host of a room shall be able to kick a user out of the room.

Rooms

35. Registered Users shall be able to create a “room” public
36. Registered Users shall be able to create a “room” private.
37. Registered Users shall be able to search a public room.
38. Registered Users shall be able to search a private room.
39. Registered Users shall be able to join a public room.
40. Registered Users shall be able to join a private room.
41. Registered Users shall be able to join a random public room.
42. Registered Users shall be able to invite people to their room.
43. Registered Users who created a room shall be able to choose what song to play.
44. Registered Users shall be able to search for songs.
45. Registered Users shall be able to choose the next song to play in the room.
46. Registered Users shall be able to change the background theme of the room.
47. Registered Users that create a room shall be able to close the room.

Chat

48. Registered Users shall be able to chat in all room types.
49. Registered Users shall be able to chat with people who joined in their created room.
50. Registered Users shall be able to create group chat.
51. Registered Users shall be able to text in group chat.

Friends

52. Registered Users shall be able to add friends.
53. Registered Users shall be able to DM a friend.
54. Registered Users shall be able to see their friends list.
55. Registered Users shall be able to remove friends.

56. Registered Users shall be able to block friends.

Administrators

57. Administrators shall be able to change SYNC usernames.
58. Administrators shall be able to reset user passwords.
59. Administrators shall be able to change user friends list
60. Administrators shall be able to open rooms with specific music selections.
61. Administrators shall be able to join rooms.
62. Administrators shall be able to review user comments.
63. Administrators shall be able to ban users.
64. Administrators shall be able to leave messages regarding reasons for user bans.
65. Administrators shall be able to ban songs.
66. Administrators shall be able to ban podcasts.
67. Administrators shall be able to delete rooms.
68. Administrators shall be able to delete accounts permanently.

Rooms

69. Rooms shall display the room name.
70. Rooms shall display if they are public or private.
71. Rooms shall display the hostname.
72. Rooms shall display a description of the room.
73. Rooms shall be up during its dedicated time set.
74. Rooms shall display the number of users in the room.
75. Rooms shall list all users in the room.
76. Rooms shall display the current song.
77. Rooms shall display the song queue.
78. Rooms shall display genre.
79. Rooms shall display chat.
80. Rooms shall display who commented in the chat.
81. Rooms shall prompt the voting system.

Website

82. Website shall have an About Us page.
83. Website shall have a FAQs page.
84. Website shall prompt users to login to Spotify.
85. Website shall prompt users to accept terms and conditions.
86. Website shall keep the user logged in.
87. Website shall have a technical support page.
88. Website shall display username.

89. Website shall display the user's account information.
90. Website shall show how we can be contacted.
91. Website shall show invites.
92. Website shall send notifications
93. Website shall show the user's most preferred genres/artists.
94. Website shall give the option to continue or cancel creation of the room.
95. Website shall allow user to send invitation link to rooms
96. Website shall display DMs
97. Website shall show the list of added friends.
98. Website shall display the number of SYNC friends the user has
99. Website shall show available public rooms.
100. Website shall show the history of rooms the users have been in.
101. Website shall show the history of rooms the users have been in.
102. Website shall show exported playlists.
103. Website shall be able to allow users to like playlists.
104. Website shall be able to allow users to add to the list of favorite playlists.
105. Website shall have premiers of podcast
106. Website shall have premiers of song drops
107. Website shall have premiers on album drops

5. List of non-functional requirements

Functionality

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

Security

6. Spotify username/ email address and password shall be required to use the web app.
7. Login prompt shall appear when the user first visits the website.
8. Private rooms shall only appear to invite Registered Users.

Privacy

9. Let users accept policies before creating an account.
10. Password and other personal information shall be kept hidden.
11. Collect Spotify data through API
12. We use API to dictate recommended rooms
13. We use API to group people together.
14. Authenticate users by checking with username and password?

Performance

15. The website shall be up all the time
16. The website shall make a search for a room easily
17. All users shall have their music synced with others having same preferences
18. Empty rooms should be destroyed automatically
19. The invite links to private rooms shall stay active till the room is active
20. The website shall not add more than 2 seconds to the time it takes to login to Spotify
21. Unused rooms shall be destroyed with 5 minutes of non attendance

System Requirements

22. The website shall work up to Version 88.0.4324.150 of Google Chrome.
23. The website shall work up to Version 85.0 of Mozilla Firefox.
24. The website shall work up to Version 81.0.416.64 of Microsoft Edge.
25. The website shall work up to Version 14.0 of Safari.
26. The website shall support Spotify music streaming.

Marketing

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

Content

30. The website shall have a navigation bar
31. The website shall have a search bar to search public rooms available.
32. The website shall present a general community room when users first sign in.
33. The website shall present recommended rooms for the specific users.
34. A navigation bar shall be present to direct users to other parts of the website
35. The website shall present an option to join private rooms

Scalability

36. The rooms shall be able to handle a large number of users.
37. The website shall have sufficient rooms to accommodate a growing number of users.
38. The webapp shall be developed using microservices architecture, contributing to the maintenance of the application.

Capability

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

Look and Feel

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

Coding Standards

54. The code shall be understandable
55. The code shall be organized
56. The code shall have proper working functions
57. The code shall have in-line comments
58. The code shall be pushed or pulled from branches in git properly
59. The code shall be well maintained in the relevant branches
60. The code shall have proper documentation
61. The code shall be maintained with one coding style
62. The code shall have proper formatting
63. The code shall have indentation

Availability

64. The website shall be active even if no users are active on the website
65. The website shall resync when loss of connection occurs
66. The website shall refresh when the website does not load.
67. The website will make general rooms unavailable within a set time or minimum user tolerance.
68. The website shall generate error messages when an error occurs
    1. Connection loss
    2. Incorrect Credentials

Fault tolerance

69. Website shall be able to refresh the Chat area when disconnected.
70. Room shall be able to be refreshed when disconnected.
71. Website shall reattempt accessing the database if a database query fails.

Storage

72. Store users' listening history on the database.
73. Store users' friends list on the database.
74. Store administrator notices about users.
75. Remove friends from the users friend list on the database if user removes
76. Store users favorited music on the database.
77. Remove the favorite list from database if user removes
78. Store chat while the room is open in the database.
79. Store users voting record.
80. Store usernames on the database.
81. Store passwords on the database.
82. Store users ignore lists in the database.
83. Store a banned word list on the database.
84. Store a banned user list on the database.
85. Store room name on database
86. Remove room name from database when room is destroyed
87. Store room description on database.
88. Remove room description from database when room is destroyed.
89. Store room song history on database.
90. Remove room song history from the database when the room is destroyed.
91. Store room voting history on database.
92. Remove room voting history from the database when the room is destroyed.

Expected Load

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

Legal

95. The website shall have Terms and Conditions that the user will be required to accept before they log in.
96. The website shall have privacy policies in the settings section.
97. The website shall have copyright notice.

6. Competitive Analysis

	Discord	Soundcloud.com	App.jqbx.fm	Beatsense.com	Listentogether.app
Strengths	Seamless onboarding experience.	Useful platform for aspiring artists.	Supports multiple services.	Users can jam with their music matches.	Nice UI
Weaknesses	Required to create an account and invite links to access servers.	Users cannot chat or make friends.	Hard time creating rooms, users need to have a Spotify premium account.	Very busy UI, confusing, -poor UX.	Lacking important services, poor user experience.
Pricing	Free, has option for Discord Nitro	Free, has option for Pro version	Free, needs Spotify premium account	Free	Free
Social Media	Facebook, Instagram, Twitter	Instagram, Snapchat, Facebook, Twitter	Discord, Facebook, Twitter	Blog posts, Facebook, Twitter	None
Onboarding	Simple, Smooth experience.	Takes quite time to follow.	Seamless onboarding	Too many steps to follow.	Moderate number of steps.
Usability	Famous among youth, gaming purposes	Used to share or create music online.	Easy to use	Bit complex to use	Usable.

Feature	Discord	Soundcloud.com	App.jqbx.fm	Beatsense.com	Listentogether.app	SYNC
Spotify support	+	-	+	-	+	+
UI simplicity	+	-	+	-	+	++
Chat	+	-	+	+	-	+
Public and private rooms	-	-	+	+	+	++
Voting System	-	-	-	-	-	+
Connections	+	+	+	+	-	+
Podcasts	-	+	-	-	-	+
Communal rooms	-	-	-	+	-	++
Top rooms	-	-	+	+	-	+

+ : Feature exists ++ : Feature is superior - : Feature doesn't exist

Summary of competitive analysis

SYNC is a platform in which users share their music data from their Spotify account. The 'SYNC' aspect is that these users are able to share and experience the music from their personal data accounts in real time. It enables a platform for users to gather and sync their music data that leads to creating a live experience.

SYNC's competitors are not mainstream sites or services due to the fact that their functionality is subpar. Our first mission as a company is to build a user base. Our top priority is to simulate a live listening concert experience where people will join a public room on our server and share their music with others through the chat. These features will be implemented with solid databases, acknowledging users' listening styles and habits and a UI that complements the data. The key is communication. Users would come back for the flow of interaction.

People have a wide range of emotions while looking for songs to listen to, and it is best to not distract their feelings. We plan to implement a simple UI with well embedded layout elements so that users may spend the least down time looking for their ideal room. We will focus on making a wide range of listening rooms to which the users can always find a room that best expresses their feelings. Moreover, since we will collect users' data, we implement data to create a better description for rooms.

One thing that Spotify, where we are taking our API from, does well is create recommended playlists. They are so well built out that they rely on them to direct the song choice to what they are listening to. This feature allows the users to also discover new music. But there is one problem with recommendation algorithms, is that they require data, which you have to generate through use time. To solve this, we let people who have been using Spotify (in other words, have been collecting songs) for a longer time to show off their collection on our website. The process will be interesting because the outcome will be limitless when people colabing on their music collection.

Most importantly, SYNC is cost efficient. We focus on speed, scale, and efficient ways to implement open source APIs. That fact does not affect our valuation as a company. Data is a new currency in the 21st century and we will generate a large data stream through our users. This 'currency' will be spent improving the performance of the site and how well it works for the user's experience.

7. High-level system architecture and technologies used

Storage: Firebase functions on the backend to authenticate users' Spotify profile. Also, we store player status on firebase to sync users. Store basic users' info and play log on sqlite from django

Front-End: Implement React to manipulate data from Spotify API.

Profile:

- Implement Spotify API to import users' info. Users can modify display info on our website, which will be stored on our database.

Information that will be imported from Spotify API:

1. Name
2. Age - restrict age in certain rooms
3. Spotify account
    1. Songs log
    2. Playlists

Information that will be stored on our server:

4. Active-Status
5. Active time
6. Connections
    a. Friendlist
    b. Direct Message
7. Song info
    a. Votes
    b. Play-rates (global)
    c. Click rates (from suggestion)
8. Room info
    a. Name of room
    b. Room ID
    c. Host of room
    d. What type of room
    e. Users in room
    f. Queued music
    g. Song suggestion
    h. Number of votes
9. Server Info
    a. Active room
    b. Room list
    c. Room removed when inactive

10. Chat
    a. Chat shall refresh
    b. Chat shall show who wrote what
    c. Support emojis

Technology:

Serve Host: AWS
Operating System: Ubuntu
Database: MySQL
Web Server: Apache or Nginx

Server-Side Language: Python

Additional Tech:

Front end framework: React
Web Framework: Django
IDE: Visual Studio Code, PyCharm
Web Analytics: Google Analytics
SSL Cert: Lets Encrypt (Cert Bot)
SASS: Bootstrap

8. Team Contribution

Team Lead, Document Master,	Rebecca Zumaeta	Contributed to writing half of the use cases of section 2. Main Use Cases and completed section 8. Assigned tasks to members on team and became available when needing feedback/guidance. Held and directed meeting towards document creation and researched along with team. Guided conversation and thought process per each member for each section of document.
Front End Lead, Document Assistant, mediator	Bryan Fetner	Wrote the entire summary of section 1. Contributed to writing and developing the use case scenarios for half of the written use cases of section 2. Main Use Cases. Discussed points and concepts to be implemented into nonfunctional and functional requirements. Gave suggestion on features and how to best implement them in documentation. Attended all meetings for the entirety of the time and continuously held conversation. Spoke out when passionate but open to other ideas when conversing with team.
Back End Lead, Document contributor	Luong Dang	Wrote out the mostly of section 7. High-level system architecture and technologies used. Contributed to section 3. List of main data items and entities making sure it connected back to section 7. Discussed concepts, features, and coding implementations into our project. Attended all meeting and contributed to conversation when felt most knowledgeable of subject matter. When asked for suggestion gave great concepts and ideas for other members to bounce from.
Front End Member	Malcolm Angelo De Villar	Wrote out half of section 3. List of Main Data Items and Entities, a good portion of section 4. Initial List of Functional requirements and a good portion of section 5. Non-functional requirements. In every

		section he wrote, he contributed to features and concepts. Contributed thoughts and input in other sections such as the Competitive Analysis section. Attended all meetings and gave input when he seemed best knowledgeable of the subject. Put out work when asked and continuously checked with the team when making decisions.
Front End Member	Hirva Patel	Built out diagram and wrote out paragraph of section 6. Competitive Analysis. Contributed to writing section 3. Main data items and section 5. Non-functional requirements. Contributed thoughts and implementations for the site as whole which lead to requirements and entities implemented in the document. Attended all meetings and contributed in conversation when she felt best fit and was passionate about the subject. Reported back to the team when she had updates on work and wanted feedback. Took constructive criticism well and worked off of it.
Github Master and back end member	Vishakha Tyagi	Drew out each Use Case Diagram for section 2. Main Use Cases and contributed to constructing other sections such as 4. Functional and 5. Non-Functional requirements. Gave input and suggestions towards implementation of the site which lead to requirements and entities involved in documentation. Attended all meetings and gave input when she felt best knowledgeable upon the subject and is passionate upon ideas. Checked with other members, especially with the team lead, when it comes to making any decisions within documentation.
Back End Member	Ashwini Managuli	Wrote out a good portion of section 4. List of Functional Requirements. Contributed to writing sections 3. Main data items and Entities and section 5. Non-Functional requirements. Contributed thoughts and

		input into sections 7. High-level System Architecture and Technologies Used. Attended all scheduled meetings and contributed into conversation when wanting to share her thoughts and experiences. Adapted quickly to the tasks given to her and put out great work. Was open to feedback and implemented constructive criticism well.

9. Checklist

- Team found a time slot to meet outside of the class

  Done

- Github master chosen

  Done

- Team decided and agreed together on using the listed SW tools and deployment server

  Done

- Team ready and able to use the chosen back and front end frameworks and those who need to learn are working on learning and practicing

  Done

- Team lead ensured that all team members read the final M1 and agree/ understand it before submission

  Done

- Github organized as discussed in class (e.g. master branch, development branch, folder for milestone documents etc.)

  Done

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	Luong Dang	ldang2@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	Ashwini Managuli	amanaguli@mail.sfsu.edu

Milestone 2

04/01/2021

History Table

Version	Date	Notes
M2V2	04/08/2021	
M2V1	04/01/2021	
M1V2	03/09/2021	
M1V1	03/05/2021	

Table of Contents

1. Data Definitions V2..... 3
2. Functional Requirements..... 6
3. UI Mocks and Storyboards……..... 10
4. High level database architecture and organization 18
5. High Level APIs and Main Algorithms..... 23
6. High Level UML Diagrams 25
7. High-level Application Network and Deployment Diagrams..... 26
8. Key Risks for Project thus far..... 27
9. Project Management 28
10. List of contributions. 29

1. Data Definitions V2

1. User: a person who has a spotify account that would like to listen to music with others in real time on web application named ‘SYNC’
    a. user_id: Unique number given to each registered user on SYNC
    b. spotify_id: A number given to registered user having spotify account
    c. profile_pic: A display picture of every registered user and they have the option to post a picture of themselves or not.
    d. display_name: Name of the user with which they wants to be identified as on SYNC
2. Profile: It has the information describing the registered user.
    a. user_id: Unique number given to each registered user on SYNC
    b. activity : It will show if the registered user is currently using SYNC or available online or not.
    c. profile_photo: A display picture of every registered user for which they have the option to post a picture of themselves or not.
    d. status : It defines if the person is listening to songs in the room as a participant who joined the room or as a host who created the room for others to join.
3. Participant: Any registered user who is using the SYNC for listening to songs in the real time and is present in the room but did not create the room.
    a. user_id: Unique number given to each registered user on SYNC
4. Host: The registered user who either created a private or a public room and sends invites to others to join the room. Also, this user has more control of the room than other participants.
    a. user_id: Unique number given to each registered user on SYNC
5. spotify_info: The information of the users imported from the spotify.
    a. auth_token:
    b. spotify_id:
    c. user_name: Name of the Spotify user.
    d. user_id: Unique number given to each registered user on SYNC
    e. playlist_list_id: Every playlist on spotify has a number attached to it

6. spotify_API
    a. Player
        i. connectivity (Online, disconnected, or error) : If the player is properly connected to the internet and is able to play songs as per the request of registered users.
        ii. current_song_title : The title of the song currently being played in the player.

iii. progress : The minute at which the song in the player is playing.

b. song
   i. song_title : The title of the song that can be searched or in queue.
   ii. artist : The creator of the song associated with song_title.
   iii. image_url : The image that is associated with the album.
   iv. genre : The category at which the song is considered in.
   v. album : The album name and which the song belongs in.

c. artists
   i. artist_name : The artist that is associated with the song.
   ii. songs : The songs that are created by the artist.
   iii. album : The album created by the artist.

d. genres
   i. image_url : Image that represents the genre.
   ii. genre_name : The name of the genre in which a song belongs to.
   iii. description : Describes what the genre is and gives a summary of what will be expected.
   iv. top_playlist : Shows the top playlist in the genre.

7. rooms
   a. room_type : It determines the kind of room the created room is.
      i. Public - available for all users through search and recommended results
      ii. Private - only available to other users through the sharing of the room id
      iii. Communal - A perpetual room.
   b. room_id : This is the unique room identifier.
   c. room_name : This is the room name in which the user set the room name to be.
   d. description : A description of the room in which the user decides to put.
   e. current_song : It shows what the current song is playing in the room. This is also shown in the previews of the rooms.
   f. room_host : It shows who created the room.
      i. user_id : This is the unique identifier of the user.
   g. password : This is the password set by the room_host for private rooms.
   h. status : Room status can either be Open or Closed, depending on the activity of the room.
   i. max_members : This is the max limit number of users that can join in a room, which is specified by the room_host.
   j. current_number : This is the current number of users that are in the room.
8. queue : The songs that are in queue in the room to be played and voted on.
   a. room_id : Identifies where that song queue belongs in.

    b. song_list_id : This is the unique queue list identifier.
9. song_list : This shows the
    a. song_title : This shows the title of the songs in the song_list
    b. song_id : This is the unique identifier of the songs in song_list.
    c. votes_id : This is the unique identifier for the votes in the song_list.
10. votes : This identifies what songs the users voted for to be played next in the room.
    a. user_id : This is the unique identifier for the user who voted for songs.
11. chat_section : This is the portion of the web application where it shows the users where a user can chat with.
    a. tab_id : This is the unique identifier for the chat_section to identify who the user is talking to, or what the current active tab is.
    b. tab_status : This identifies the current, active, or inactive chat tabs per user.
    c. server : This will be the server that will handle all realtime chat interactions.

2. Functional Requirements V2

Priority 1:

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 homepage of the website.
1.3. Unregistered Users shall be able to access the About Us 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.
2.11. Registered Users shall be able to access the About Us 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.
2.20. Registered Users shall be able to change status offline.
2.21. Registered Users shall be able to change status online.
2.26. Registered Users shall be able to logout.
2.27. Registered Users that create a room shall have the status of host.
2.28. Registered Users as hosts shall be able to name the room.
2.29. Registered Users as hosts shall be able to generate playlists.
2.30. Registered Users as hosts shall be able to control the music queue.
2.31. Registered Users as hosts shall be able to pause 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.
2.38. Registered Users shall be able to search a private 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.
2.41. Registered Users shall be able to join a random public room.
2.42. Registered Users shall be able to invite people to their room.
2.43. Registered Users who created a room shall be able to choose what song to play.
2.44. Registered Users shall be able to search for songs.
2.45. Registered Users shall be able to choose the next song to play in the room.

2.47. Registered Users that create a room shall be able to close the room.
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.
2.50. Registered Users shall be able to create group chat.
2.51. Registered Users shall be able to text in group chat.
2.52. Registered Users shall be able to add friends.
2.53. Registered Users shall be able to DM a friend.
2.54. Registered Users shall be able to see their friends list.
2.55. Registered Users shall be able to remove friends.
2.56. Registered Users shall be able to block friends.
2.57 Registered Users shall be able to vote on songs to be played next in queue

4. Rooms

3.69. Rooms shall display the room name.
3.70. Rooms shall display if they are public or private.
3.71. Rooms shall display the hostname.
3.72. Rooms shall display a description of the room.
3.73. Rooms shall be up during its dedicated time set.
3.74. Rooms shall display the number of users in the room.
3.75. Rooms shall list all users in the room.
3.76. Rooms shall display the current song.
3.77. Rooms shall display the song queue.
3.78. Rooms shall display genre.
3.79. Rooms shall display chat.
3.80. Rooms shall display who commented in the chat.
4.81. Rooms shall have the voting system.

5. Website

5.87. Website shall have a technical support page.
5.88. Website shall display username.
5.89. Website shall display the user's account information.
5.90. Website shall show how we can be contacted.
5.91. Website shall show invites.
5.93. Website shall show the user's most preferred genres/artists.
5.94. Website shall give the option to continue or cancel creation of the room.
5.95. Website shall allow user to send invitation link to rooms
5. 99. Website shall show available public rooms.

Priority 2:

1. Unregistered Users

1.6. Unregistered Users shall be able to get access to technical support.

2. Registered Users

2.14. Registered Users shall be able to get access to technical support.
2.32. Registered Users as hosts of a room shall be able to disable sharing
2.33. Registered Users as hosts shall be able to set a limit to the number of users in the room.
2.34. Registered Users as host of a room shall be able to kick a user out of the room.
2.46. Registered Users shall be able to change the background theme of the room.

3. Administrators

3.57. Administrators shall be able to change SYNC usernames.
3.58. Administrators shall be able to reset user passwords.
3.59. Administrators shall be able to change user friends list
3.60. Administrators shall be able to open rooms with specific music selections.
3.61. Administrators shall be able to join rooms.
3.62. Administrators shall be able to review user comments.
3.63. Administrators shall be able to ban users.
3.64. Administrators shall be able to leave messages regarding reasons for user bans.
3.65. Administrators shall be able to ban songs.
3.66. Administrators shall be able to ban podcasts.
3.67. Administrators shall be able to delete rooms.
3.68. Administrators shall be able to delete accounts permanently.

5. Website

5.96. Website shall display DMs
5.97. Website shall show the list of added friends.
5.98. Website shall display the number of SYNC friends the user has
5.100. Website shall show the history of rooms the users have been in.
5.101. Website shall show exported playlists.
5.102. Website shall be able to allow users to like playlists.

5.103. Website shall be able to allow users to add to the list of favorite playlists.

Priority 3:

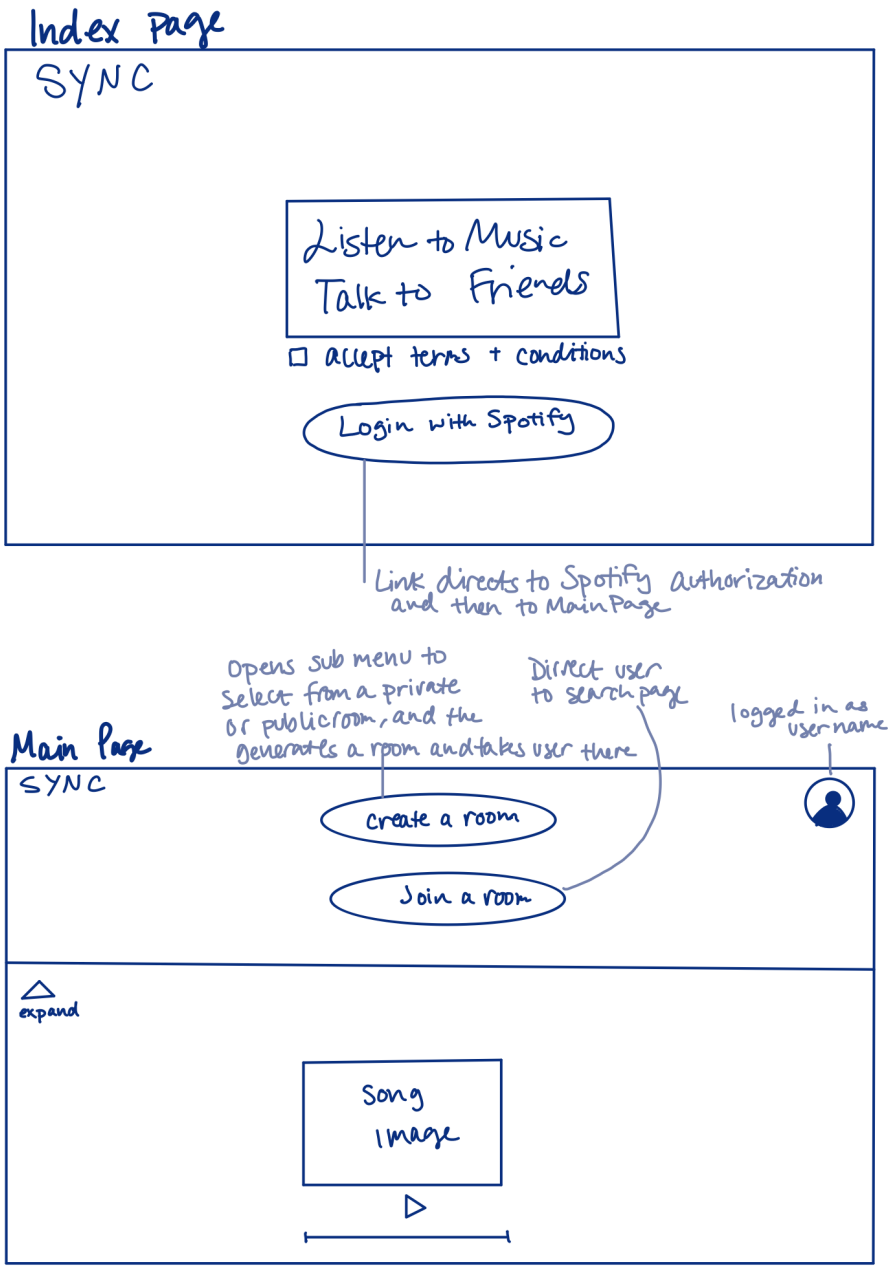
2. Registered users

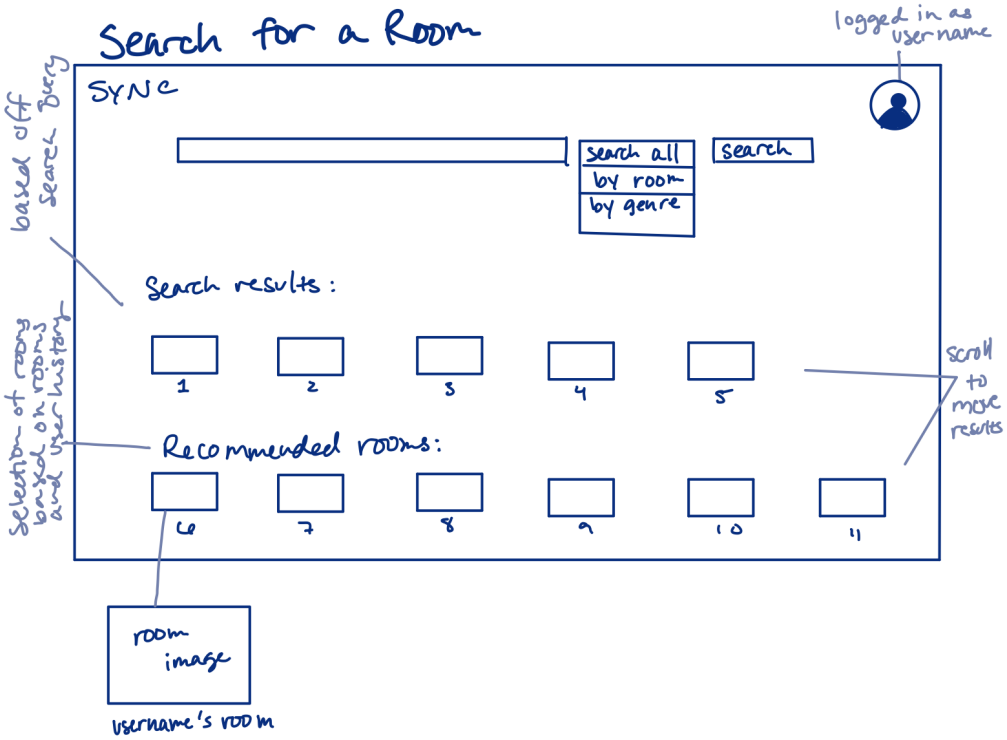
2.15. Registered Users shall be able to export the playlist of the room.
2.16. Registered Users shall be able to edit their SYNC profile.
2.17. Registered Users shall be able to change their SYNC usernames.
2.18. Registered Users shall be able to change their SYNC profile picture.
2.19. Registered Users shall be able to change any information under the Profile page.
2.22. Registered Users shall be able to report other users for misconduct.
2.23. Registered Users shall be able to share a room link though social media.
2.24. Registered Users shall be able to share link through Direct Message
2.25. Registered Users shall be able to share link through email

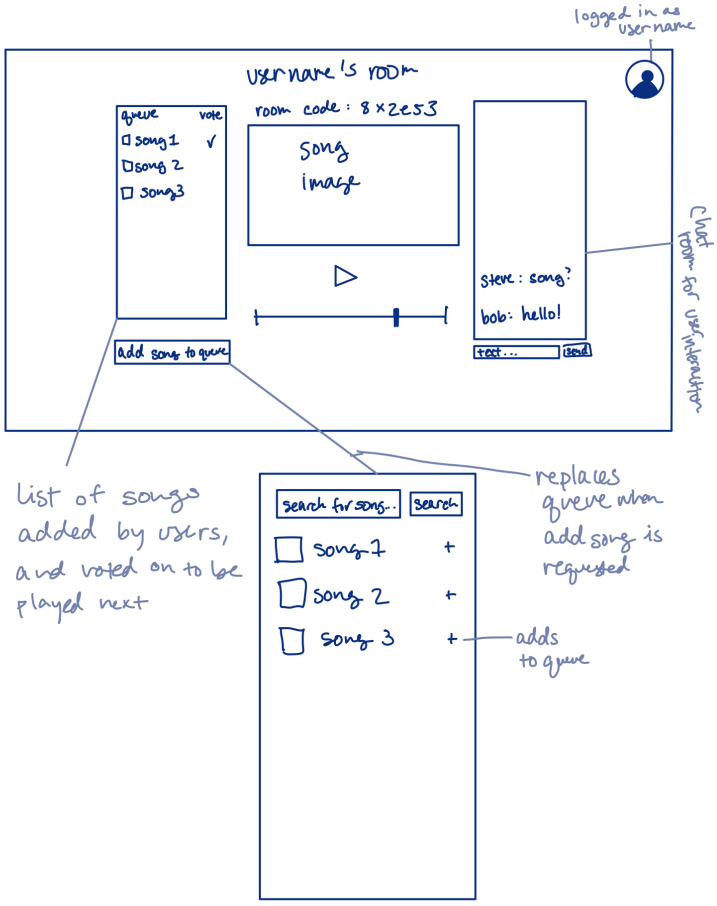
5. Website

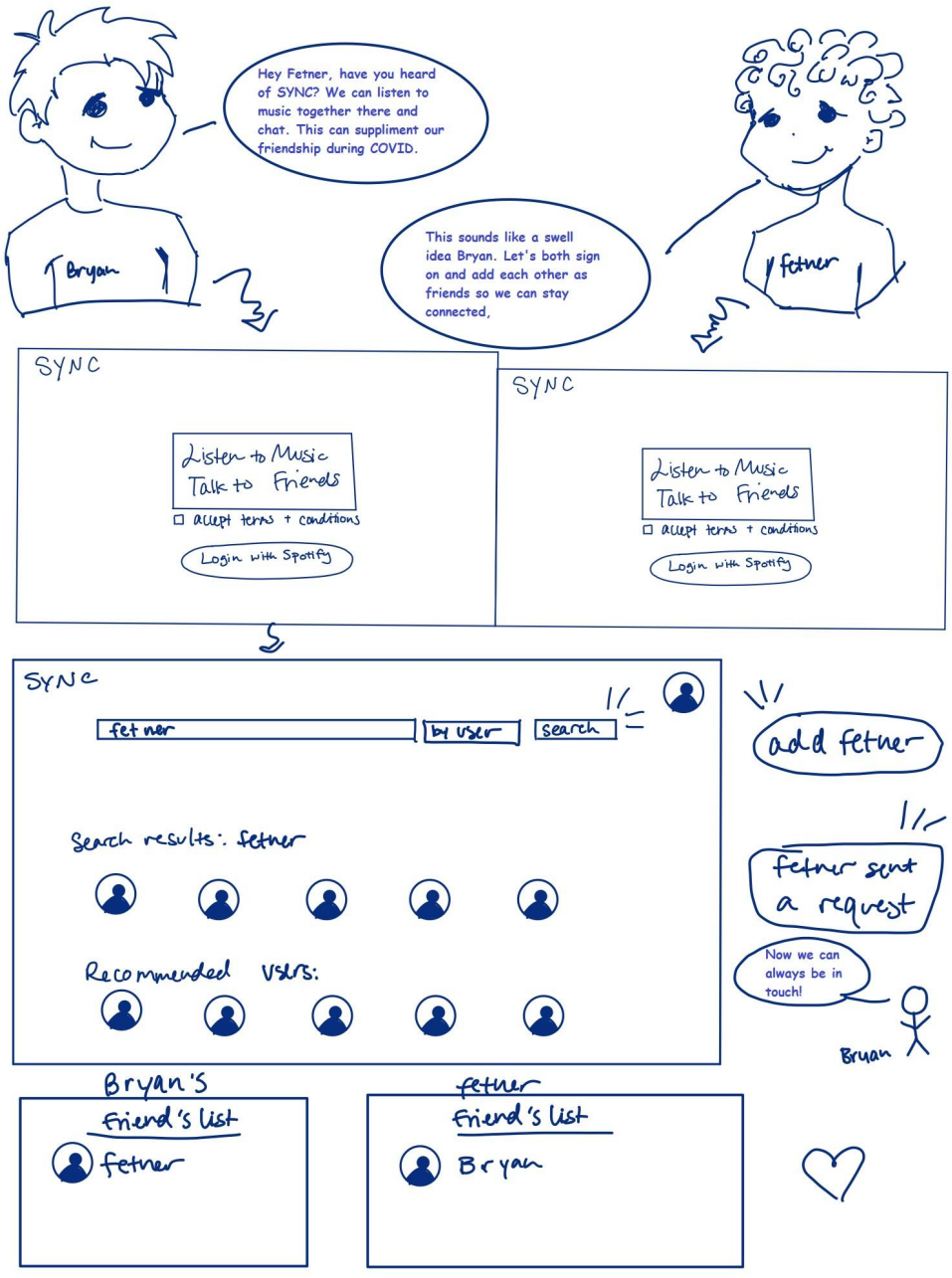
5.92. Website shall send notifications
5.104. Website shall have premiers of podcast
5.105. Website shall have premiers of song drops
5.106. Website shall have premiers on album drops

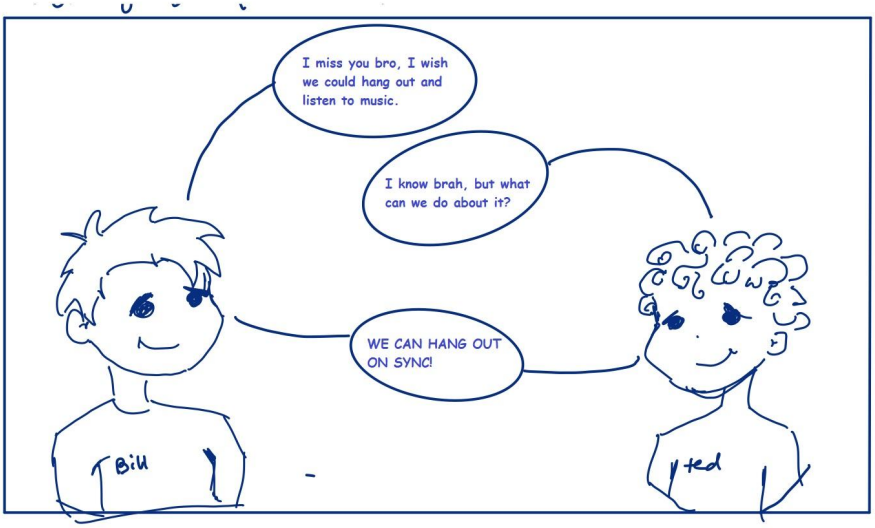
3. UI Mockups and Storyboards

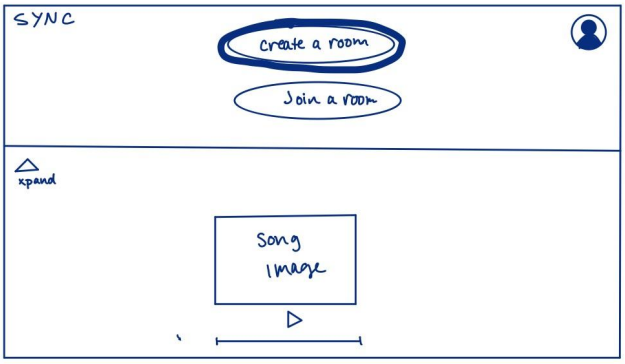


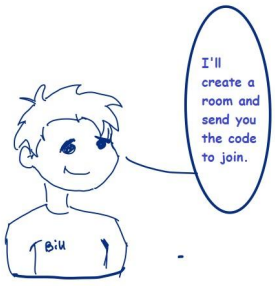

Search for a Room
logged in as username
SYNC
based off search query
search all
by room
by genre
search
Search results:
1
2
3
4
5
scroll to more results
selection of rooms based on rooms and user history
Recommended rooms:
6
7
8
9
10
11
room image
username's room


logged in as username
username's room
room code: 8x2e53
queue
vote
song1
song 2
song3
song image
steve: song?
bob: hello!
text...
send
add song to queue
chat room for user interaction
list of songs added by users, and voted on to be played next
search for song...
search
song 1
song 2
song 3
replaces queue when add song is requested
adds to queue

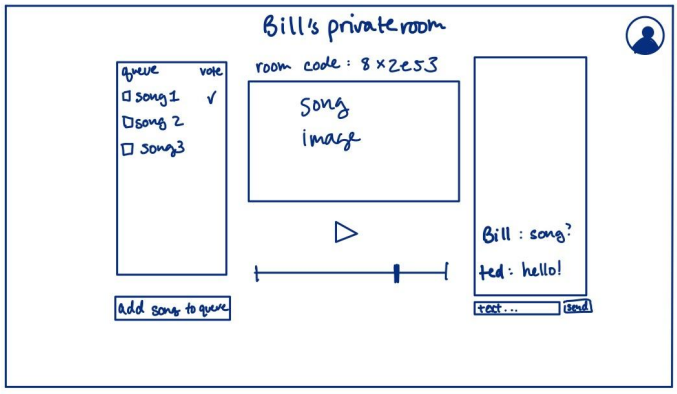

Hey Fetner, have you heard of SYNC? We can listen to music together there and chat. This can suppliment our friendship during COVID.
Bryan
This sounds like a swell idea Bryan. Let's both sign on and add each other as friends so we can stay connected,
fetner
SYNC
Listen to Music
Talk to Friends
accept terms + conditions
Login with Spotify
SYNC
Listen to Music
Talk to Friends
accept terms + conditions
Login with Spotify
SYNC
fetner
by user
search
Search results: fetner
Recommended users:
add fetner
fetner sent a request
Now we can always be in touch!
Bryan
Bryan's
Friend's list
fetner
fetner
Friend's list
Bryan

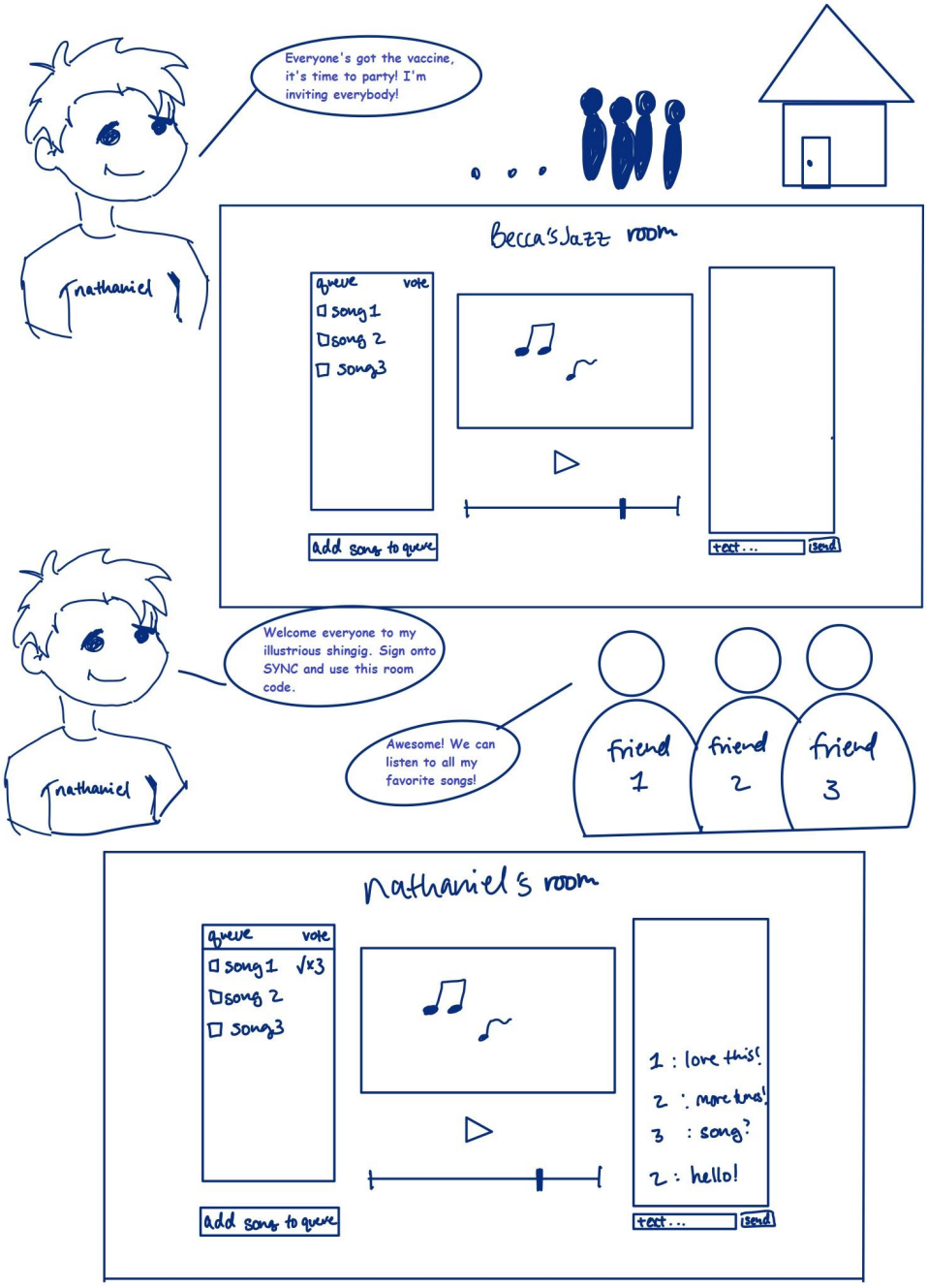

I miss you bro, I wish we could hang out and listen to music.
I know brah, but what can we do about it?
WE CAN HANG OUT ON SYNC!
Bill
ted

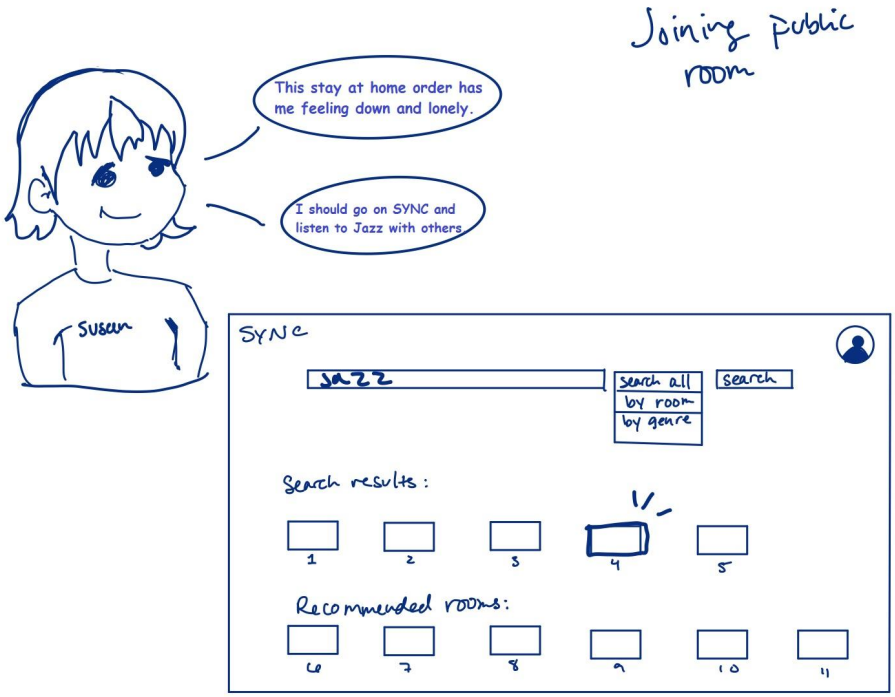

SYNC
create a room
Join a room
xpand
Song image

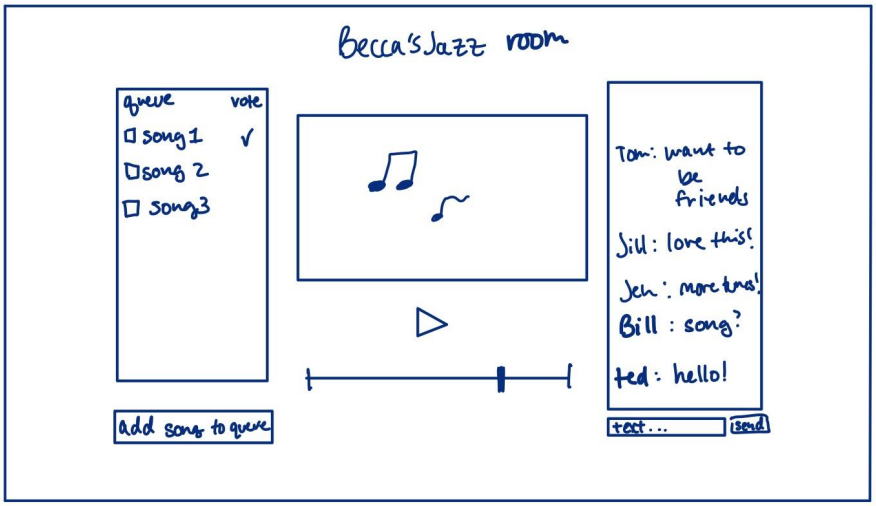

I'll create a room and send you the code to join.
Bill

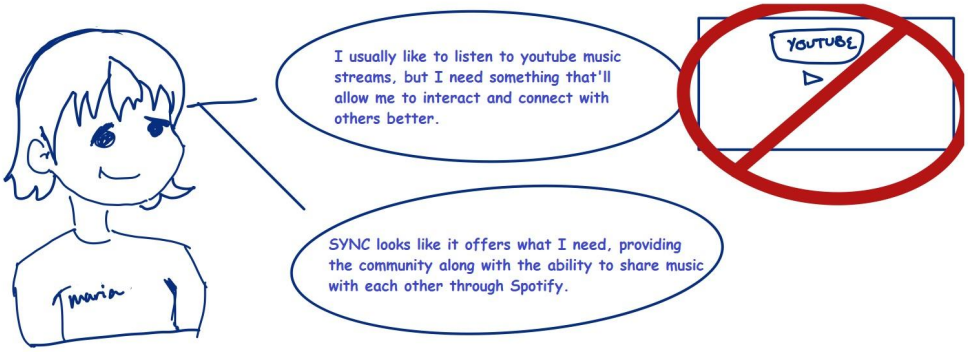

Bill DM:
Sweet, now we can hang!
ted

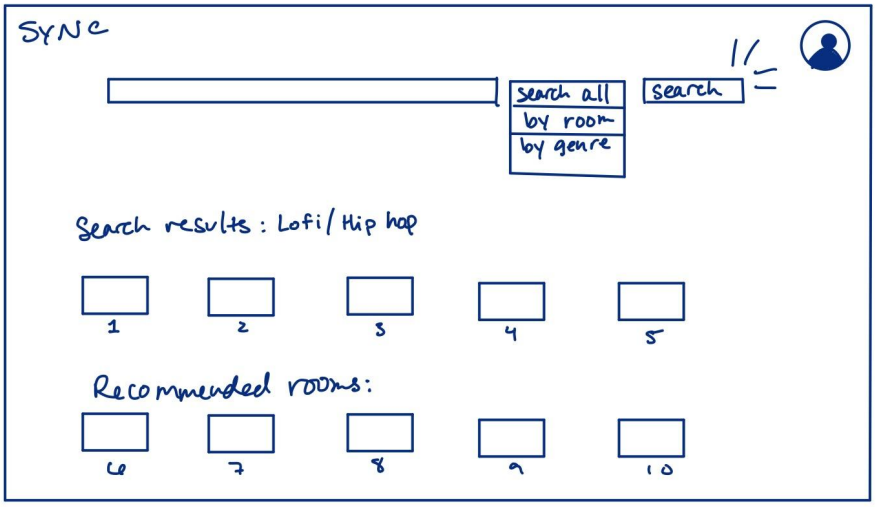

Bill's private room
queue
vote
song 1
song 2
song 3
room code: 8x2e53
Song image
Add song to queue
Bill: song?
ted: hello!
text...
send

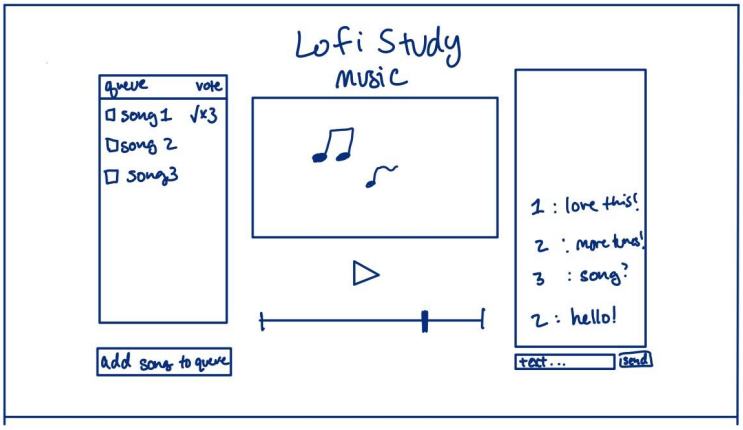

Everyone's got the vaccine, it's time to party! I'm inviting everybody!
nathaniel
Becca's Jazz room
queue
vote
song 1
song 2
song 3
Add song to queue
text...
send
Welcome everyone to my illustrious shingig. Sign onto SYNC and use this room code.
Awesome! We can listen to all my favorite songs!
friend 1
friend 2
friend 3
nathaniel's room
queue
vote
song 1 √x3
song 2
song 3
Add song to queue
1 : love this!
2 : more tmrs!
3 : song?
2 : hello!
text...
send


Joining public room
This stay at home order has me feeling down and lonely.
I should go on SYNC and listen to Jazz with others.
Susan
SYNC
Jazz
Search all
by room
by genre
Search
Search results:
1
2
3
4
5
Recommended rooms:
6
7
8
9
10
11


Becca's Jazz room
queue
vote
song 1
song 2
song 3
add song to queue
Tom: want to be friends
Jill: love this!
Jen: more tea!
Bill: song?
ted: hello!
text...
send


I usually like to listen to youtube music streams, but I need something that'll allow me to interact and connect with others better.
YOUTUBE
SYNC looks like it offers what I need, providing the community along with the ability to share music with each other through Spotify.
Maria


SYNC
search all
by room
by genre
search
Search results : Lofi / Hip hop
1
2
3
4
5
Recommended rooms:
6
7
8
9
10


Lofi Study
music
queue
vote
song 1
song 2
song 3
add song to queue
1 : love this!
2 : more lofi!
3 : song?
2 : hello!
text...
send

4. High level database architecture and organization

The DBMS we chose to create our database is MySQL because it is easy to understand and some of us have experience with it; it is an organized and widely used DBMS.

1. **User (strong)**
    a. Unregistered users has one Spotify account
    b. Registered user has one and only one username
    c. Registered user has one and only one user_id
    d. Registered user has zero to one display name
    e. Registered user has one and only one profile picture
    f. Registered user shall be able to send zero to many invite links
    g. Registered user shall be able listen 0 to one song in real time in room
    h. Registered user shall be be able to be or not be a host
    i. Registered user shall be able to be or not be a participant
    j. Registered user shall have only one profile page
    k. Registered user shall have zero to one profile picture
    l. Registered user shall be zero to one host
    m. Registered user shall be zero to one participant
    n. Registered user shall be able to create zero to many public rooms
    o. Registered user shall be able to be in zero to one public room
    p. Registered user shall be able to be in zero to one private room
    q. Registered user shall be able to be in zero to one communal room
    r. Registered user shall be able invite 0 to many users to their room
    s. Registered user shall be able choose 0 to many songs to play
    t. Registered user shall be able to choose zero to many created rooms

2. **Rooms (weak)**
    a. Rooms has one and only one display name
    b. Rooms shall display one and only one hostname
    c. Rooms shall display one to many songs in queue
    d. Rooms shall display one and only one current song
    e. Rooms shall display one and only one genre
    f. Rooms shall display one and only one chat box
    g. Rooms shall display one to many voted songs
3. **Host (weak)**
    a. Host has to be in one and only one room
    b. Host shall control one and only one room.
    c. Host shall create one and only one room
    d. Host shall name one and only one room

e. Host shall play one to many songs in room

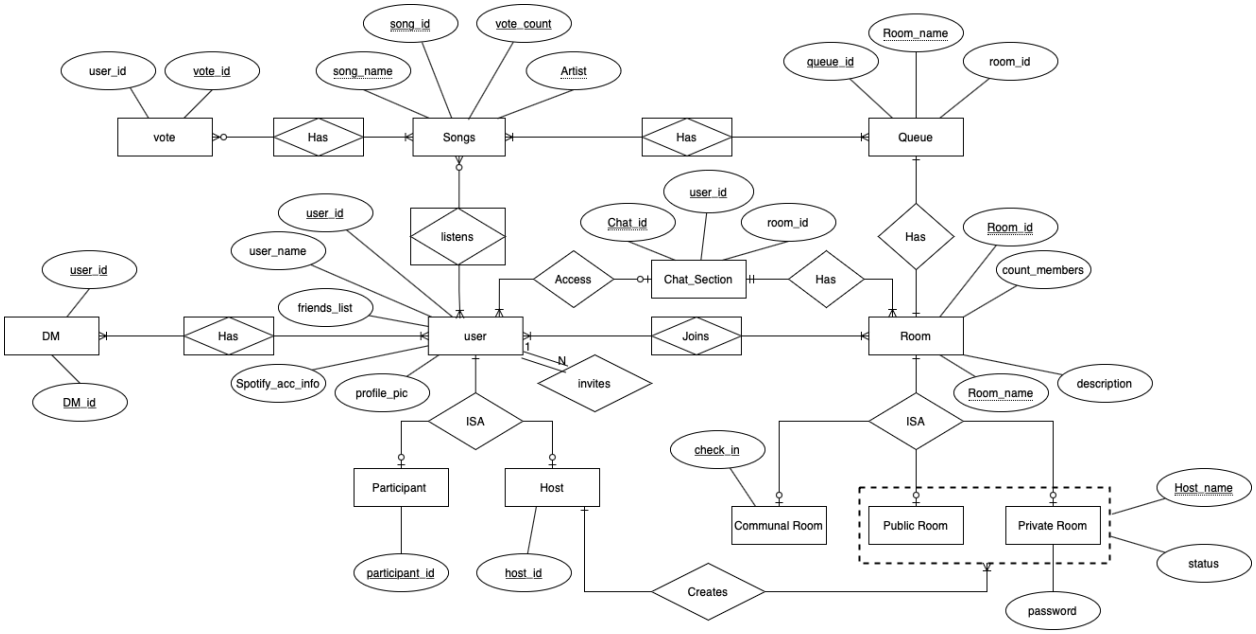
4. **Friends (weak)**
    a. Friends shall be able to be added from zero to many registered user's friend's list.
    b. Friends shall be able to be removed from zero to many registered user's friend's list.
    c. Friends shall be able to be blocked from zero to many registered user's friend's list.
    d. Friends shall Direct Message zero to many friends
5. **Chat (weak)**
    a. Chat box shall hold chat messages from zero to many users
    b. Chat box shall exist in one to many rooms

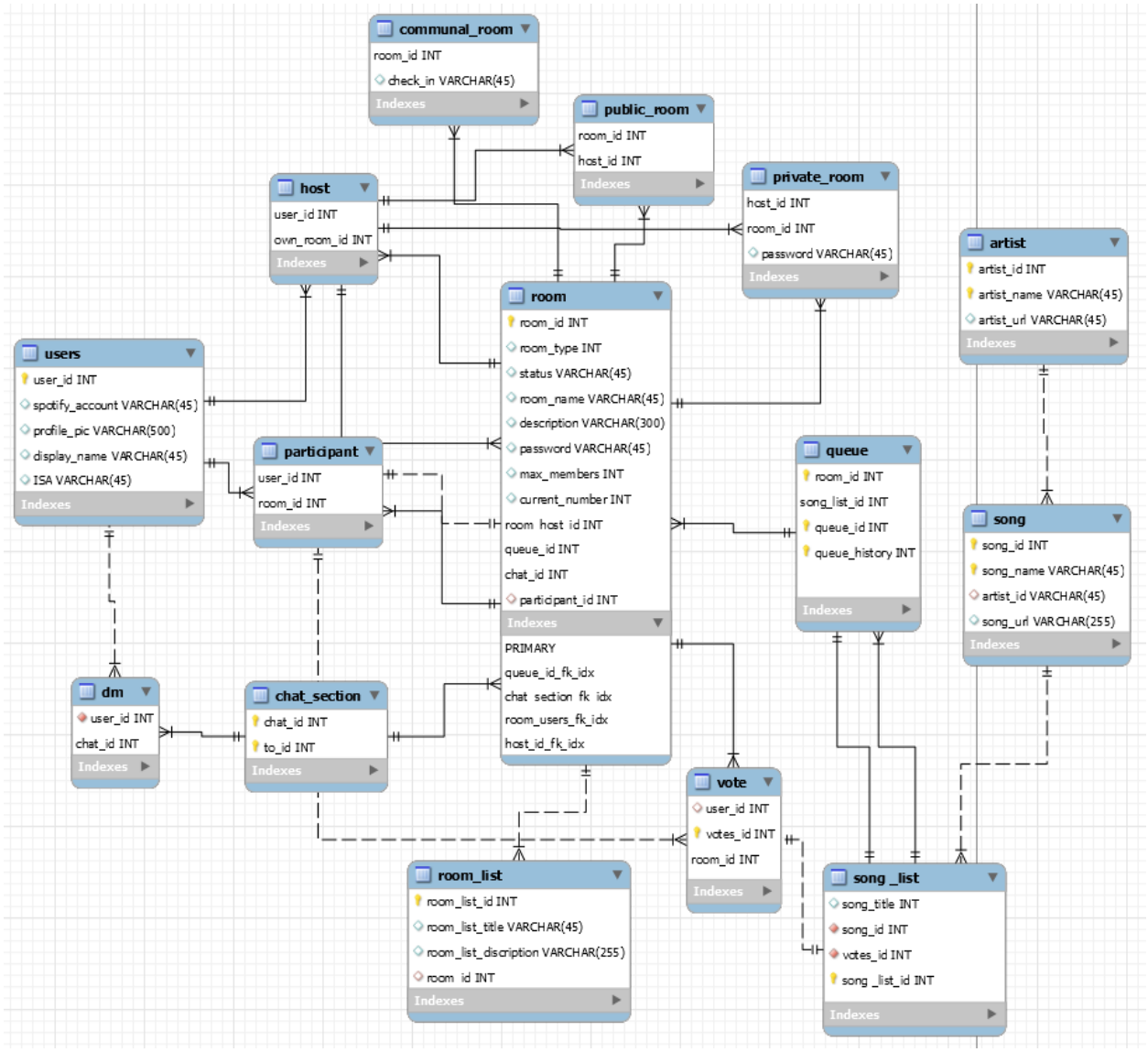
6. **Website (strong)**
    a. Website shall display zero to many Direct Messages
    b. Website shall display zero to many added friends
    c. Website shall be able to allow user to like zero to many playlists

1. **User (strong)**
    - User_id: strong key, numeric
    - spotify_id: weak key, numeric
    - profile_pic: weak
    - Display_name: alphanumeric
2. **Rooms (weak)**
    - room_type : alphanumeric
    - room_id : strong key, numeric
    - room_name : multivalue, alphanumeric
    - description : multivalue, alphanumeric
    - current_song :alphanumeric
    - room_host : alphanumeric
    - password : weak key, numeric
    - status : key, numeric
    - max_members : key, numeric
    - Current_number: key, numeric

3. **Host (weak)**
    - Host_id: strong key, numeric
4. **Chat (weak)**
    - tab_id : strong key, numeric
    - tab_status : key, numeric
    - server : key, numeric
5. **Profile:** It has the information describing the registered user.
    - user_id: strong key, numeric
    - activity : key, numeric
    - profile_photo:
    - status :key, numeric
6. **Spotify_API**
    - connectivity : key, numeric
    - current_song_title : alphanumeric
    - progress : key, numeric
    - song_title : multivalue, alphanumeric
    - artist : multivalue, alphanumeric
    - image_url : multivalue, alphanumeric
    - genre : multivalue, alphanumeric
    - album : multivalue, alphanumeric
    - artist_name : multivalue, alphanumeric
    - songs : multivalue, alphanumeric
    - album : multivalue, alphanumeric

- image_url :multivalue, alphanumeric
- genre_name : multivalue, alphanumeric
- description : multivalue, alphanumeric
- top_playlist : multivalue, alphanumeric




communal_room
room_id INT
check_in VARCHAR(45)
Indexes
public_room
room_id INT
host_id INT
Indexes
host
user_id INT
own_room_id INT
Indexes
private_room
host_id INT
room_id INT
password VARCHAR(45)
Indexes
artist
artist_id INT
artist_name VARCHAR(45)
artist_url VARCHAR(45)
Indexes
room
room_id INT
room_type INT
status VARCHAR(45)
room_name VARCHAR(45)
description VARCHAR(300)
password VARCHAR(45)
max_members INT
current_number INT
room host id INT
queue_id INT
chat_id INT
participant_id INT
Indexes
PRIMARY
queue_id_fk_idx
chat section fk idx
room_users_fk_idx
host_id_fk_idx
users
user_id INT
spotify_account VARCHAR(45)
profile_pic VARCHAR(500)
display_name VARCHAR(45)
ISA VARCHAR(45)
Indexes
participant
user_id INT
room_id INT
Indexes
queue
room_id INT
song_list_id INT
queue_id INT
queue_history INT
Indexes
song
song_id INT
song_name VARCHAR(45)
artist_id VARCHAR(45)
song_url VARCHAR(255)
Indexes
dm
user_id INT
chat_id INT
Indexes
chat_section
chat_id INT
to_id INT
Indexes
vote
user_id INT
votes_id INT
room_id INT
Indexes
room_list
room_list_id INT
room_list_title VARCHAR(45)
room_list_discription VARCHAR(255)
room id INT
Indexes
song _list
song_title INT
song_id INT
votes_id INT
song _list_id INT
Indexes

5. High Level API and Main Algorithms

API:

An API is an application programming interface - in short, it's a set of rules that lets programs talk to each other, exposing data and functionality across the internet in a consistent format.

High level API:

Auth: Authentication process which handles security and customization of profile for users. This will implement Spotify API to let users login with their existing Spotify account then import information related to the account from Spotify endpoints to our website. This API will make the authentication process quick, smooth, free, and also provides useful data of new users to our website.

RoomFeedManager: This API handles homepage, which displays list of existing rooms, suggestion rooms, enables creating new rooms, and sorting room lists.

RoomManager: This API helps users interact with the listening room. It handles display room's info, update room's info, display members in room, display chat, display queue, invite users, and kick/ban users.

Search:This API handles every type of searching operations. It helps the user search for existing rooms, songs, artists, and other users.

Queue: This API handles the voting system of a room. This API re-enforce a set of rules that will enable everyone in a room to distribute to the listening experience. It takes song suggestions from members of the room, keeps track of the votes, and is a helper API for the Player API.

Player: This API is a music player with familiar functions like play,pause,skip, and like. It will implement Spotify's WEB API to enable users to have the most enjoyable music listening experience.

Below is a list of general API that will be implemented. These will be implemented on multiple database's set, therefore, will have head of "*", which will be replaced by model name (e.g., user, playlist, room, etc.) in the development process.

1) *create

This API creates a new record to a specific data table after checking if the record is existing.

2) *createdetail

This API creates a new detail column to a specific data table. This will be mostly used for the chat feature

3) *update/id

This API retrieves a specified record, updates it and handles any error in the process.

4) *updatedetail/id

This API retrieves detail of a specified record, updates its detail and handles any error in the process

5) *retrieve

This API retrieves a list of specific record(s) by their id or detail(s) and reads the list.

6) *retrieve/id

This API retrieves details of a specific record by its id and reads it.

7) *delete/id

This API deletes a specific record or a list of it.

8) authentification

This API handles login and authenticates users' info while they are on the site.

9) spotifyplayer

This API retrieves a specific song from Spotify, plays them, handles functionality of a music player(e.g. play,skip,pause,add to favorite, etc.) and displays the current playing song's status.

The website will implement the general APIs from the above list differently with each data table for different tasks.

Main Algorithms:

Basic Search: When there are a huge amount of records in the database, scanning the entire table is not effective. We will implement index attributes, which are provided and automatically indexed by MySQL database to handle the retrieve function. More about index attribute, by using a tree data structure and binary search, the time complexity of data record searching by indexed DB term can be reduced to O(logn) from O(n).

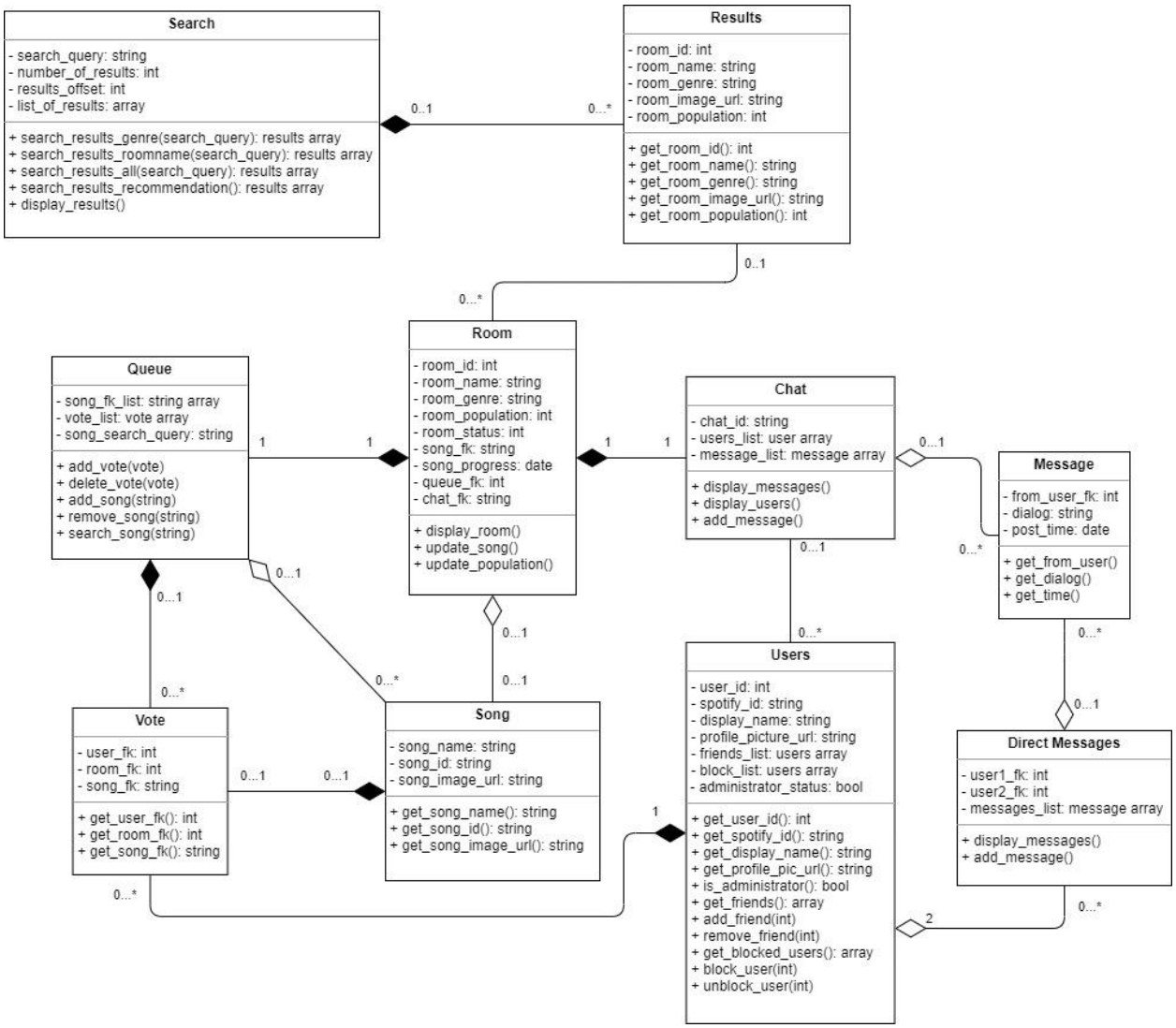
Inner-Joint Search: We connect data tables using foreign keys to create advanced search functions. We can combine information from multiple tables to provide more useful information when users search.

Authentication algorithm: Spotify offers an endpoint for authentication , which we will implement on our website. We can keep the login/register process easy and quick by offering login using a Spotify account. Moreover, we need users' authentication token, which is provided by Spotify after the users login, to be able to play songs from Spotify. Every song the users play on our website will be free of charge and therefore our website's operating expense is zero but still able to offer a quality music listening experience.

Extra(Authentication process on our website): Spotify Web API requires a developer's key which is provided and saved as an .env file on our server. When a user tries to login with a Spotify account, the request gets passed to the Spotify Web API, and the API returns an auth keylink which lasts only for one use section(meaning it will expire if the user goes inactive). Server will save the keylink under the authToken file, which will be a pass to interact on our site. Every function on the website will extract information from a personal authToken file to process.

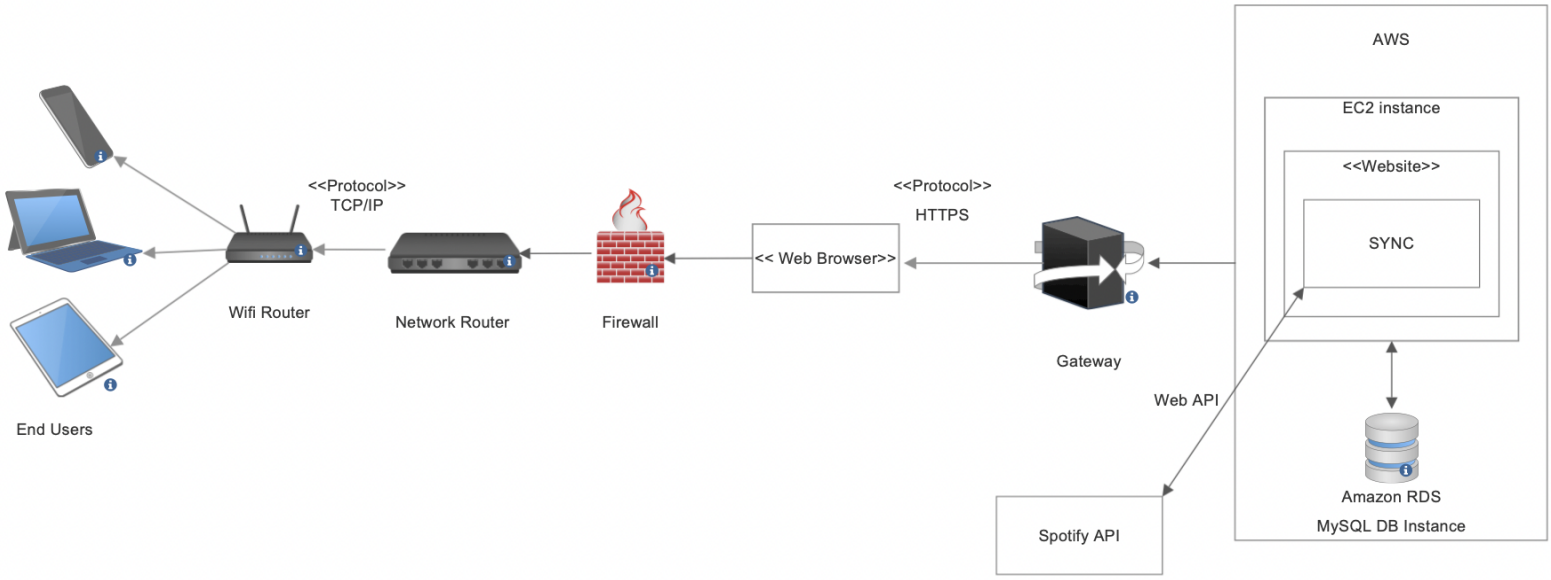
Search Spotify Data: Since Spotify provides many useful information (of users, songs, singers, playlists, etc,) for free of charge, we will implement our list of APIs with an endpoint from Spotify to utilize it.

6. High Level UML Diagrams

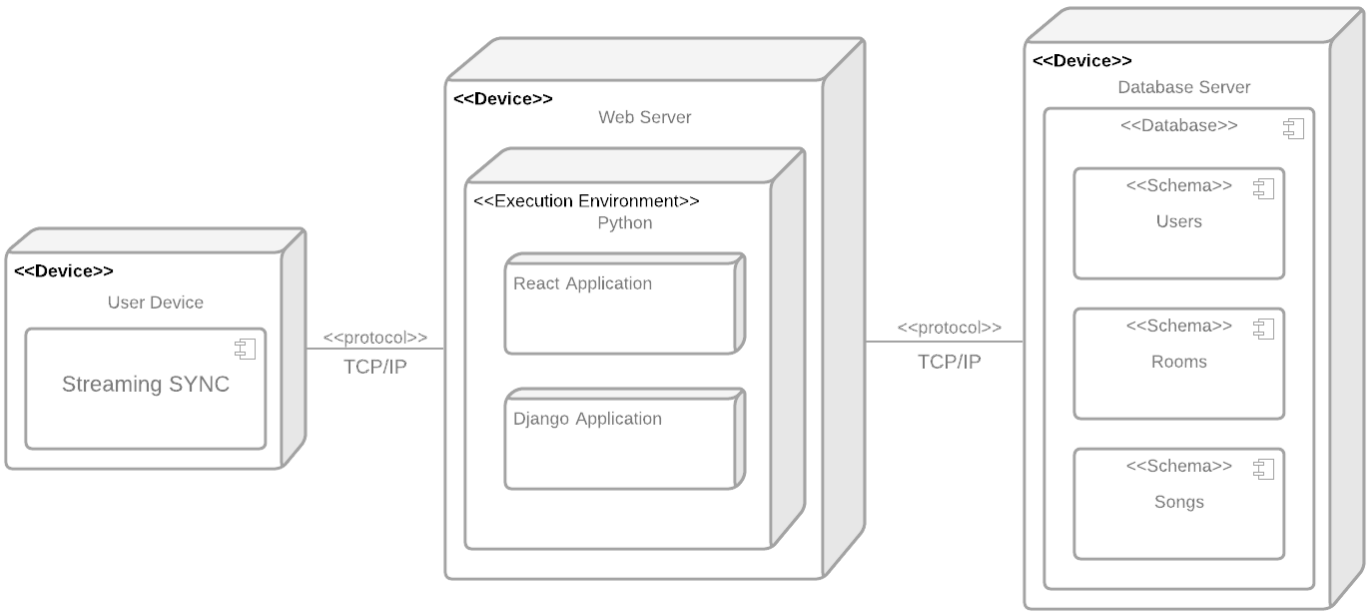


7. High Level Application Network and Deployment Diagrams

Application Network Diagram



Deployment diagram



8. Key Risks for your for project at this time

• Skills Risks - Some of us are new to creating a web application, and also new to the stack suggested by the team, which puts us risks on program development.

Solution: The team is working hard on studying and especially catching up on the stack suggested. Members also communicate most of the time if problems persist.

• Schedule Risks - While we are mostly available on our scheduled meeting times, the schedule risk we have is that some of us live from a different timezone, and some of us work during non-school days.

Solution: We find a common time slot that some of us are available in between the scheduled meeting time and day and work on development together. Moreover, we reach out if there are issues with scheduling, so we can plan ahead.

• Technical Risks - While we want to implement an integrated chat in our web application, we are running with problems on how to properly do a live synced chat, since we are not entirely sure if MySQL will be able to do it smoothly.

Solution: The team is currently working up a possible solution to how this should be implemented correctly. One thing we slightly agreed on is to use Firebase as the chat section's database to support real time messaging.

• Teamwork Risks - Communication was lost due to silence and no updates on progression of tasks leading to a last minute fixes and not creating a well done project.

Solution: Team lead plans to enforce keeping to the 'script' and ensuring we all are using the technologies stated, reporting diligently (every other day with progress) and creating a more organized space so that no one is behind/out of the loop/ or on their own. Teammates must be in contact with team lead with updates every other day to ensure progress and good communication and lastly, better team work.

• Legal/Content Risks- Since we will be implementing a chat portion in our application, one of the major risks is not being able to filter all messages sent, such as bots, viruses, malicious links, etc.

Solution: A possible solution we can do is filter external links, since that is where most risks will be coming from. Furthermore, we can add a mute chat option and/ or profanity filtering to prevent harassments.

9. Project management

Managing M2 should be done differently than with M1. It should be done with more structure and updates and even more teamwork. Team Lead should clearly state what is to be due and how it should and can be done. Team lead would give out high level tasks with pointers and ideas of how it should be done and divide the work to front and back end lead accordingly. Knowing that the front and back end leads have a better understanding of who is best at what in their respective teams, will assign roles to their members and report to the team lead. Throughout every step and every accomplishment even small should be reported to front and back end lead and then team lead. Zoom meetings would occur twice a week to have verbal affirmation, but side voice meetings though discord can and should be done with smaller teams periodically why teams needed to meet up for updates.

In theory this is a great way to keep everyone on track but it is now how it went. To improve for the next milestone, Team Lead plans to implement the steps above by enforcing with hard written deadlines to be checked off. This is going to be done through Trello with an integrated Trello bot on our discord server. I use trello in my other class team and it is quite unsuccessful. Members do not look at it including myself. I've noticed though that in both teams, people are quick to check discord. With the integration of a trello bot, I can see trello being successful and useful. Reading and understanding the milestone early with set tasks to be done accordingly can reduce time wasted and make our team more efficient for the next milestone.

10. Detailed list of contributions

Team Lead, Document Master	Rebecca Zumaeta	Assisted in writing various sections or documentation. Assigned tasks to members on team and became available when needing feedback/guidance. Guided member in tasks to be done for vertical prototype. Set and hosted all meetings, kept in contact with every member.
Front End Lead, Document Assistant, mediator	Bryan Fetner	Built a react front end with Malcom and Hirva. Also built a front in django, which is reflected on deployed sites. Built out and wrote sections 6 and 3 but assisted in many other sections such as 1 and 2. Attended all meetings for the entirety of the time and continuously held conversation. Spoke out when passionate but open to other ideas when conversing with the team.
Back End Lead, Document contributor	Luong Dang	Build working versions of sites using Spotify API and other functions we would use later in development. Entirely wrote out section 5.Attended all meeting and contributed to conversation when felt most knowledgeable of subject matter. When asked for suggestion gave great concepts and ideas for other members to bounce from.
Front End Member Document contributor	Malcolm Angelo De Villar	Assisted to part 8, 2 and 1 of the document. Assisted in designing and building the front end of the site that is deployed. Attended all meetings and gave input when he seemed best knowledgeable about the subject. Put out work when asked and continuously checked with the team when making decisions.
Front End Member	Hirva Patel	Build majority of Front end with Malcolm. Assisted in connecting front end to back with Ashwini. Attended all meetings and contributed to the conversation when she felt best fit and

		was passionate about the subject. Reported back to the team when she had updates on work and wanted feedback. Took constructive criticism well and worked off of it.
Github Master and back end member Document contributor	Vishakha Tyagi	Assisted Ashwini when possible, reflected back end work and deployment through document. Worked on document in sections 1, 2, and 7. Attended all meetings and gave input when she felt best knowledgeable upon subject and is passionate upon ideas. Checked with other members, especially with team lead, when it comes to making any decisions within documentation.
Back End Member	Ashwini Managuli	Connected back to a front end and was able to deploy. Built out back end using MYSQL workbench (tables and dummy data). Deployed site. Attended all scheduled meetings and contributed into conversation when wanting to share her thoughts and experiences. Adapted quickly to the tasks given to her and put out great work. Was open to feedback and implemented constructive criticism well.

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 3

04/25/2021

History Table

Version	Date	Notes
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. Data Definitions V3 3
2. Functional Requirements V3 6
3. Wireframes Based on Mockup/Storyboard…. 10
4. High level database architecture and organization V2 45
5. High Level Diagrams V2 ……..... 50
6. List of contributions 52

1. Data Definitions V2

1. **User**: a person who has a spotify account that would like to listen to music with others in real time on web application named 'SYNC'
    a. userId: Unique number given to each registered user on SYNC
    b. spotifyId: A number given to registered user having spotify account
    c. profilePic: A display picture of every registered user and they have the option to post a picture of themselves or not.
    d. displayName: Name of the user with which they wants to be identified as on SYNC
2. **Profile**: It has the information describing the registered user.
    a. userId: Unique number given to each registered user on SYNC
    b. activity : It will show if the registered user is currently using SYNC or available online or not.
    c. profilePhoto: A display picture of every registered user for which they have the option to post a picture of themselves or not.
    d. status : It defines if the person is listening to songs in the room as a participant who joined the room or as a host who created the room for others to join.
3. **Participant**: Any registered user who is using the SYNC for listening to songs in the real time and is present in the room but did not create the room.
    a. userId: Unique number given to each registered user on SYNC
4. **Host**: The registered user who either created a private or a public room and sends invites to others to join the room. Also, this user has more control of the room than other participants.
    a. userId: Unique number given to each registered user on SYNC
5. **spotifyInfo**: The information of the users imported from the spotify.
    a. authToken: key string for Spotify's player
    b. spotifyId: userID from Spotify's API endpoint
    c. userName: Name of the Spotify user.
    d. userId: Unique number given to each registered user on SYNC
    e. playlistListId: Every playlist on spotify has a number attached to it

6. **spotifyAPI**
    a. Player
        i. connectivity : If the player is properly connected to the internet and is able to play songs as per the request of registered users. Online, disconnected, or error
        ii. currentSongTitle : The title of the song currently being played in the player.

iii. progress : The minute at which the song in the player is playing.

b. song
   i. songTitle : The title of the song that can be searched or in queue.
   ii. artist : The creator of the song associated with songTitle.
   iii. imageUrl : The image that is associated with the album.
   iv. genre : The category at which the song is considered in.
   v. album : The album name and which the song belongs in.

c. artists
   i. artistName : The artist that is associated with the song.
   ii. songs : The songs that are created by the artist.
   iii. album : The album created by the artist.

d. genres
   i. imageUrl : Image that represents the genre.
   ii. genreName : The name of the genre in which a song belongs to.
   iii. description : Describes what the genre is and gives a summary of what will be expected.
   iv. topPlaylist : Shows the top playlist in the genre.

7. **rooms**
   a. roomType : It determines the kind of room the created room is.
      i. Public: available for all users through search and recommended results
      ii. Private : only available to other users through the sharing of the room id
      iii. Communal: A perpetual room.
   b. roomId : This is the unique room identifier.
   c. roomName : This is the room name in which the user set the room name to be.
   d. description : A description of the room in which the user decides to put.
   e. currentSong : It shows what the current song is playing in the room. This is also shown in the previews of the rooms.
   f. roomHost : It shows who created the room.
      i. userId : This is the unique identifier of the user.
   g. password : This is the password set by the roomHost for private rooms.
   h. status : Room status can either be Open or Closed, depending on the activity of the room.
   i. maxMembers : This is the max limit number of users that can join in a room, which is specified by the roomHost.
   j. currentNumber : This is the current number of users that are in the room.

8. **queue** : The songs that are in queue in the room to be played and voted on.
   a. roomId : Identifies where that song queue belongs in.

    b. songListId : This is the unique queue list identifier.
9. **songList** : This shows the
    a. songTitle : This shows the title of the songs in the songList
    b. songId : This is the unique identifier of the songs in songList.
    c. votesId : This is the unique identifier for the votes in the songList.
10. **votes** : This identifies what songs the users voted for to be played next in the room.
    a. userId : This is the unique identifier for the user who voted for songs.
11. **chatSection** : This is the portion of the web application where it shows the users where a user can chat with.
    a. tabId : This is the unique identifier for the chatSection to identify who the user is talking to, or what the current active tab is.
    b. tabStatus : This identifies the current, active, or inactive chat tabs per user.
    c. server : This will be the server that will handle all realtime chat interactions.

2. Functional Requirements V3

Priority 1:

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 homepage 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.
2.27. Registered Users that create a room shall have the status of host.
2.28. Registered Users as hosts shall be able to name the room.
2.29. Registered Users as hosts shall be able to add songs to queue
2.31. Registered Users as hosts shall be able to pause 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.
2.38. Registered Users shall be able to search a private 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 invite people to their room.
2.43. Registered Users who created a room shall be able to choose what song to play.
2.44. Registered Users shall be able to search for songs.
2.45. Registered Users shall be able to choose the next song to play in the room.
2.47. Registered Users that create a room shall be able to close the room.
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

3.58. Administrators shall be able to reset user passwords.
3.61. Administrators shall be able to join all rooms without a password.
3.63. Administrators shall be able to ban users.
3.67. Administrators shall be able to delete rooms.
3.65. Administrators shall be able to send messages to all users.
3.64. Administrators shall be able to leave messages regarding reasons for user bans.
3.66 Administrators shall be invisible (to other users in a room) when entering a room.

4. Rooms

4.69. Rooms shall display the room name.
4.70. Rooms shall display if they are public or private.
4.71. Rooms shall display the hostname.
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.
4.82. Rooms shall shut down if there is no host in it.
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.
5.90 Website shall let users register.
5.91 Website shall let users login.
5.92 Website shall let users reset their password.
5.93. Website shall display the website's contact info.
5.94. Website shall give the option to continue or cancel creation of the room.
5.95. Website shall allow user to send invitation link to 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 on Spotify.
5.101. Website shall support popular browsers

Priority 2:

1. **Unregistered Users**

2. **Registered Users**

2.20. Registered Users shall be able to change status offline.
2.21. Registered Users shall be able to change status online.
2.29. Registered Users as hosts shall be able to generate playlists.
2.33. Registered Users as hosts shall be able to set a limit to the number of users in the room.
2.34. Registered Users as host of a room shall be able to kick a user out of the room.
2.50. Registered Users shall be able to create group chat.
2.51. Registered Users shall be able to text in group chat.
2.52. Registered Users shall be able to add friends.
2.53. Registered Users shall be able to DM a friend.
2.54. Registered Users shall be able to see their friends list.
2.55. Registered Users shall be able to remove friends.
2.56. Registered Users shall be able to block friends.
2.72. Rooms shall display a description of the room.
2.73. Rooms shall be up during its dedicated time set.
2.74. Rooms shall display the number of users in the room.
2.75. Rooms shall list all users in the room.

3. **Administrators**

3.57. Administrators shall be able to change SYNC usernames.
3.59. Administrators shall be able to change user friends list
3.60. Administrators shall be able to open rooms with specific music selections.
3.62. Administrators shall be able to review user comments.
3.65. Administrators shall be able to ban songs.
3.66. Administrators shall be able to ban podcasts.
3.68. Administrators shall be able to delete accounts permanently.
3.69. Administrators shall be able to control the communal rooms.

5. **Website**

5.96. Website shall display DMs
5.97. Website shall show the list of added friends.

Priority 3:

1. Unregistered Users

1.6. Unregistered Users shall be able to get access to technical support.

2. Registered users

2.14. Registered Users shall be able to get access to technical support.
2.17. Registered Users shall be able to change their SYNC usernames.
2.22. Registered Users shall be able to report other users for misconduct.
2.23. Registered Users shall be able to share a room link though social media.
2.24. Registered Users shall be able to share link through Direct Message
2.25. Registered Users shall be able to share link through email
2.30. Registered Users as hosts shall be able to control the music queue.
2.41. Registered Users shall be able to join a random public room.
2.46. Registered Users shall be able to change the background theme of the room.

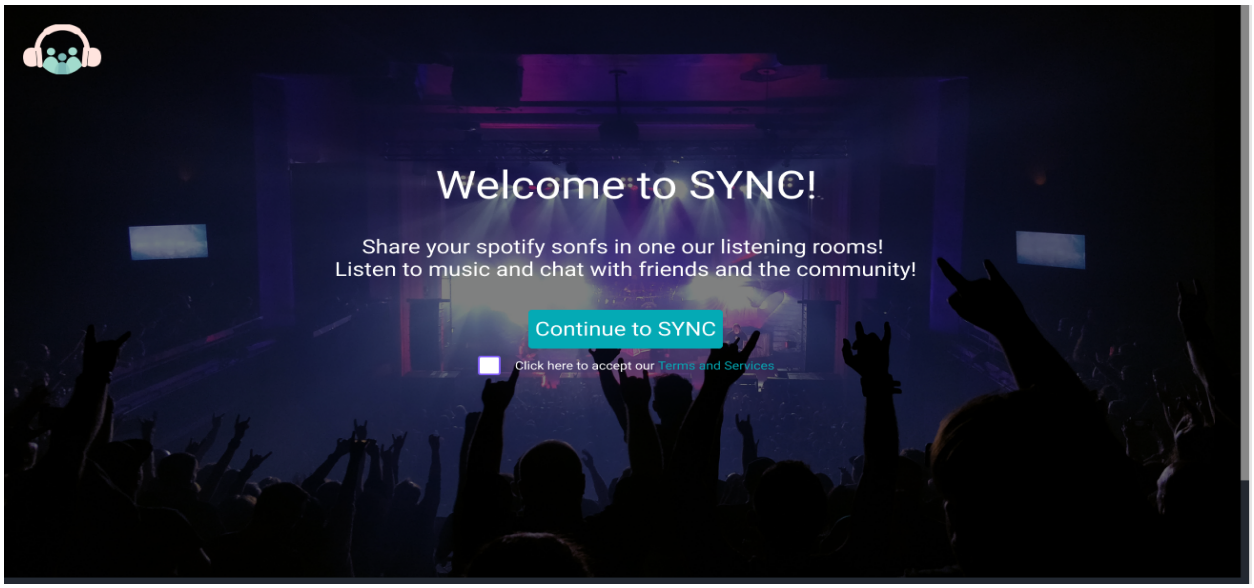
5. Website

5.87. Website shall have a technical support page.
5.91. Website shall show invites.
5.92. Website shall send notifications
5.93. Website shall show the user's most preferred genres/artists.
5.98. Website shall display the number of SYNC friends the user has
5.100. Website shall show the history of rooms the users have been in

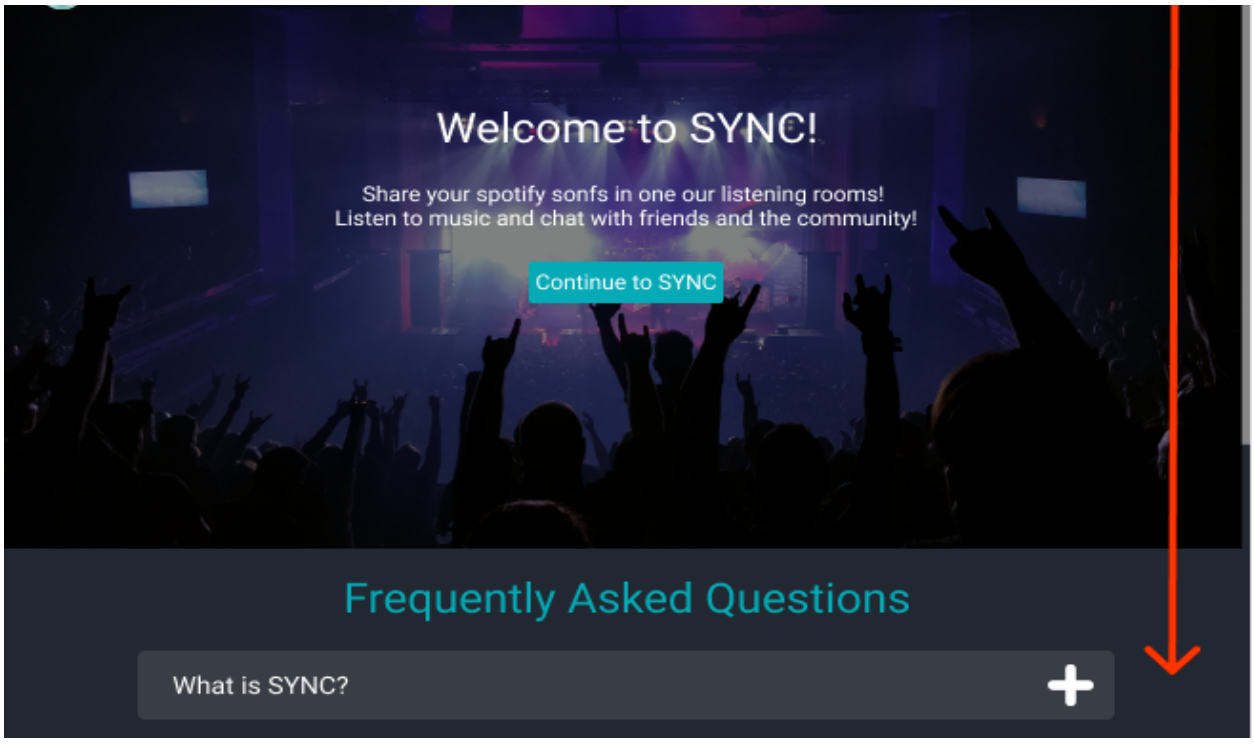
3. Wireframes Based on Mockups/Storyboards V2

1. View Frequently Asked Questions

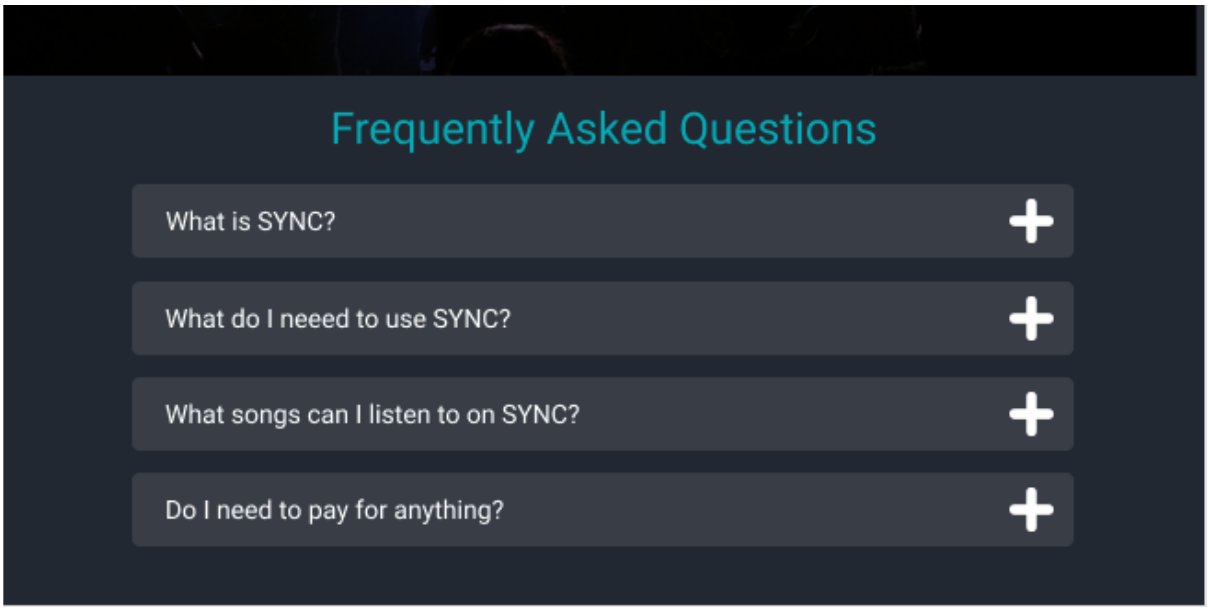
a. Navigate to http://3.17.66.0:3000/ it will bring you to this page



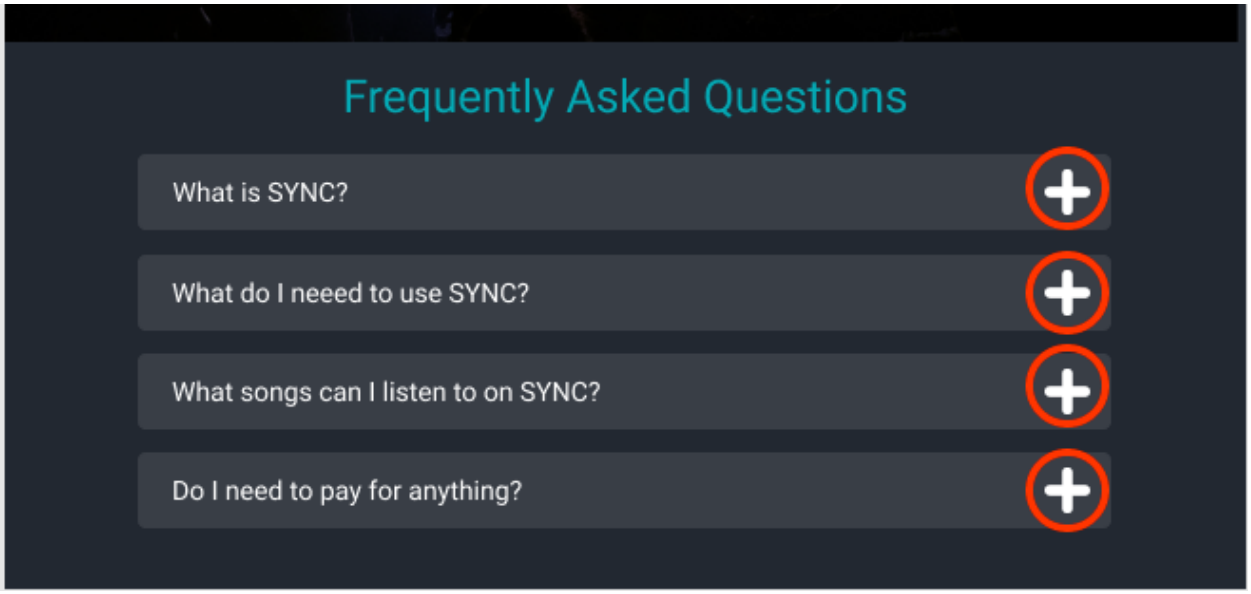
b. Scroll down the landing page



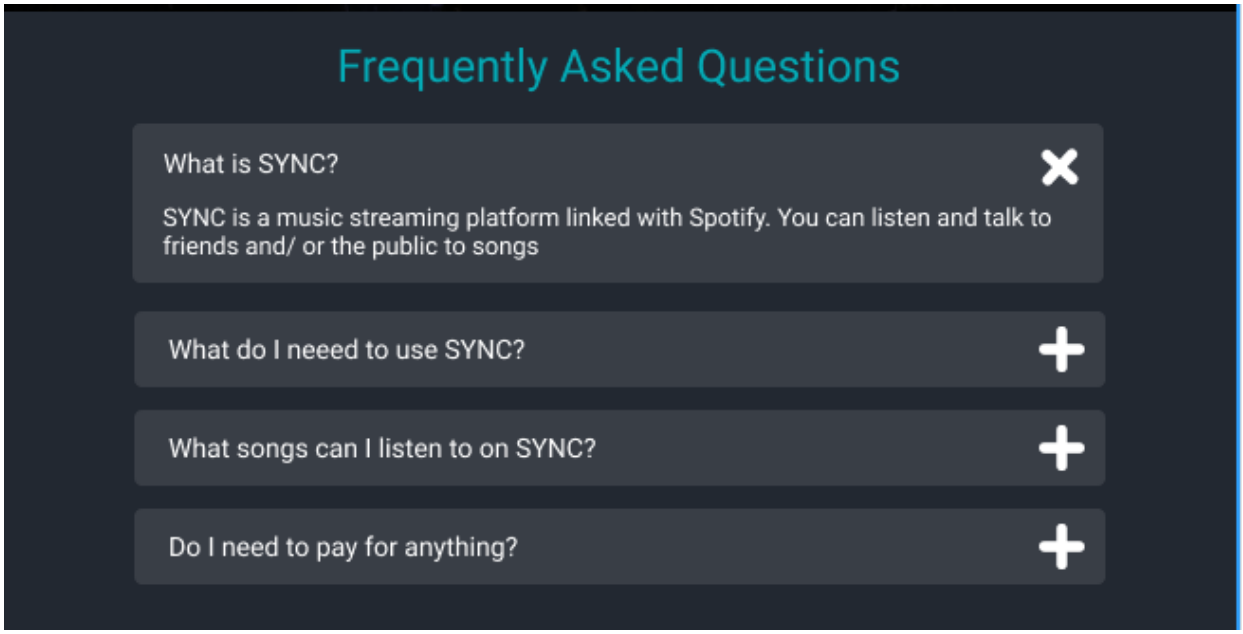
c. You will see a list of Frequently Asked Questions it will look like this



d. Select any of the '+' plus icons to view the answers the questions depicted on the left

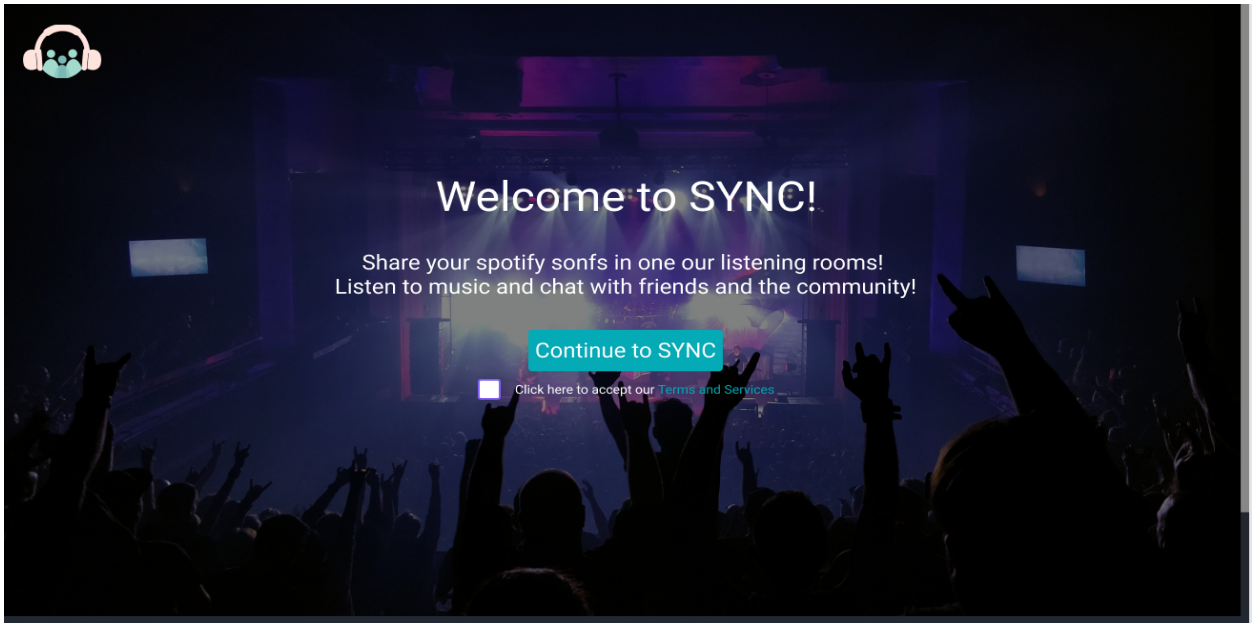


e. Once one of the '+' plus icons are selected you should be able to see the answer just as it is depicted below.

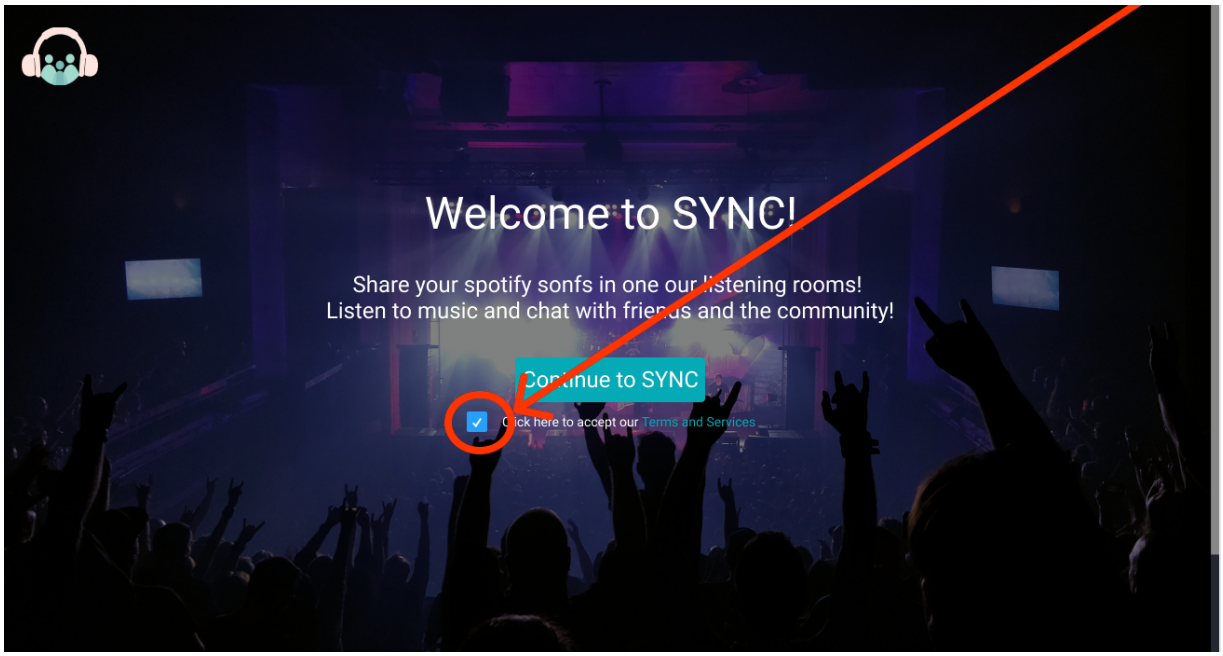

Frequently Asked Questions
What is SYNC?
SYNC is a music streaming platform linked with Spotify. You can listen and talk to friends and/ or the public to songs
What do I neeed to use SYNC?
What songs can I listen to on SYNC?
Do I need to pay for anything?

2. **Continue into SYNC and check off Terms of Services**

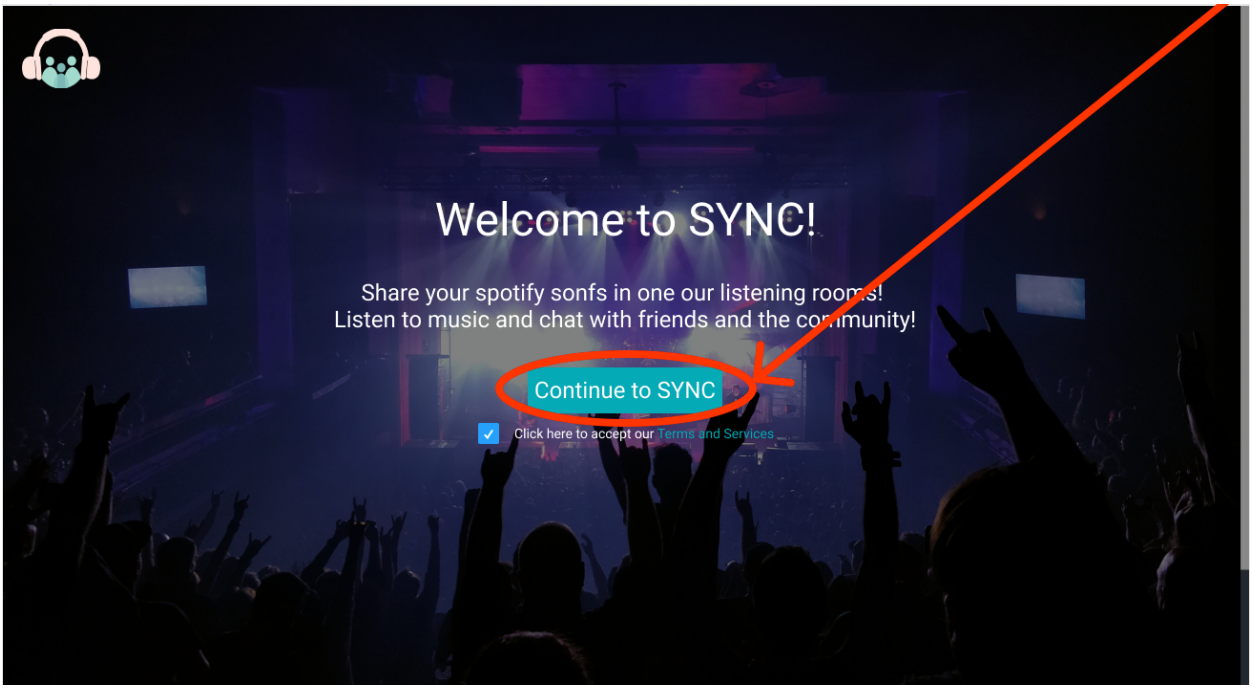
    a. Navigate to http://3.17.66.0:3000/ it will bring you to this page



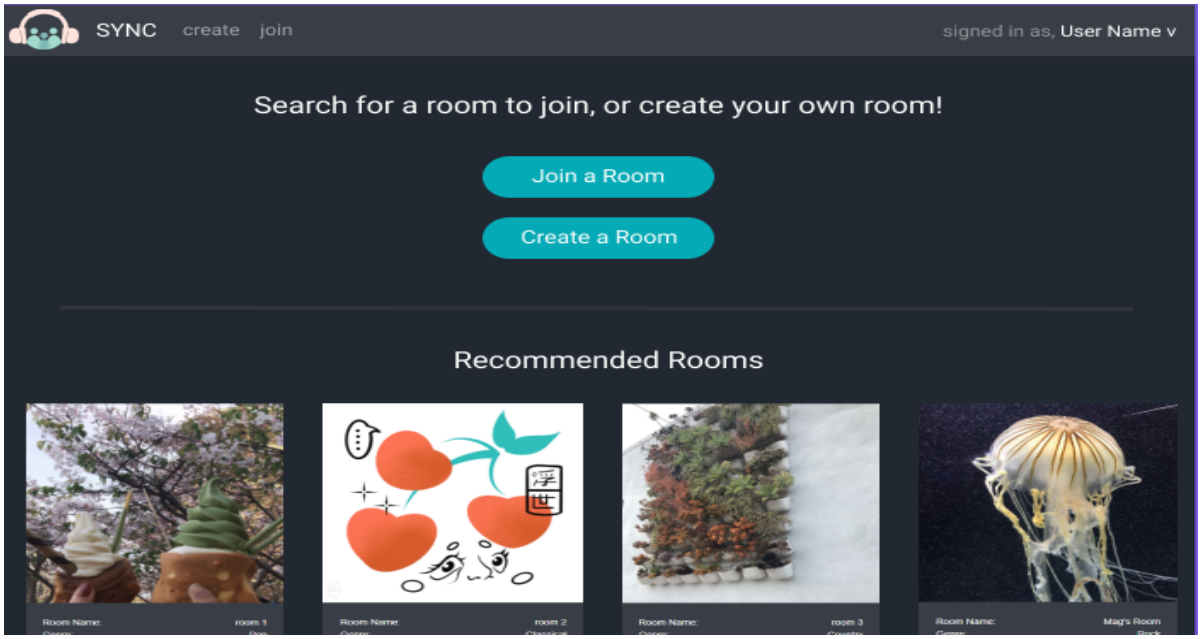
    b. To continue to the home page you must check off the terms and services. To do so you must check of the check box button located below the 'Continue to SYNC' button



c. Once that is checked select the ‘Continue to SYNC’ button to be redirected to SYNC’s home page.

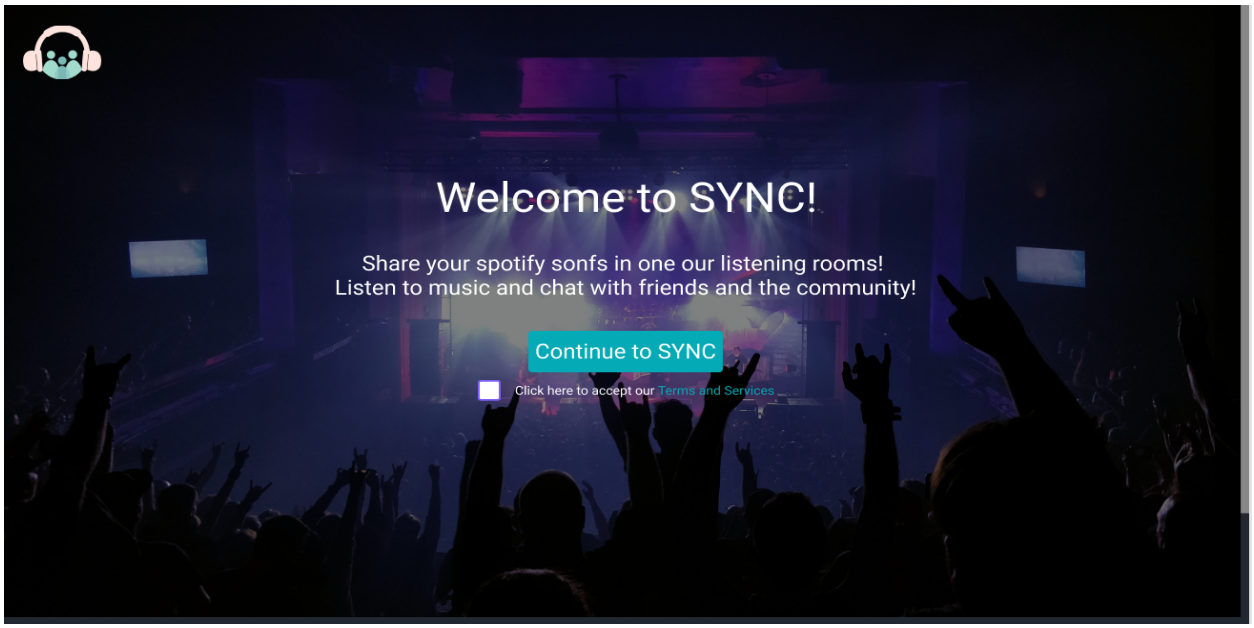


d. Directly after pressing the ‘Continue to SYNC’ button you will be redirected to the home page of SYNC!

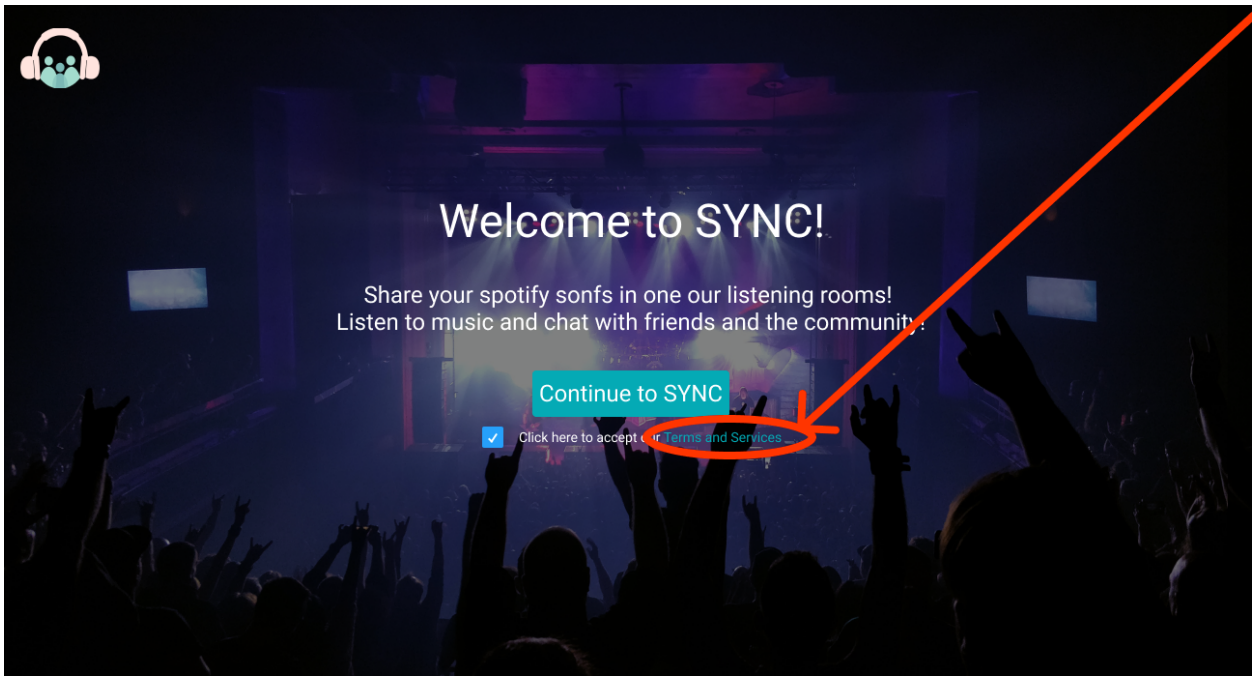


If you would like to read the Terms and Services, you can select the link provided on the landing page.

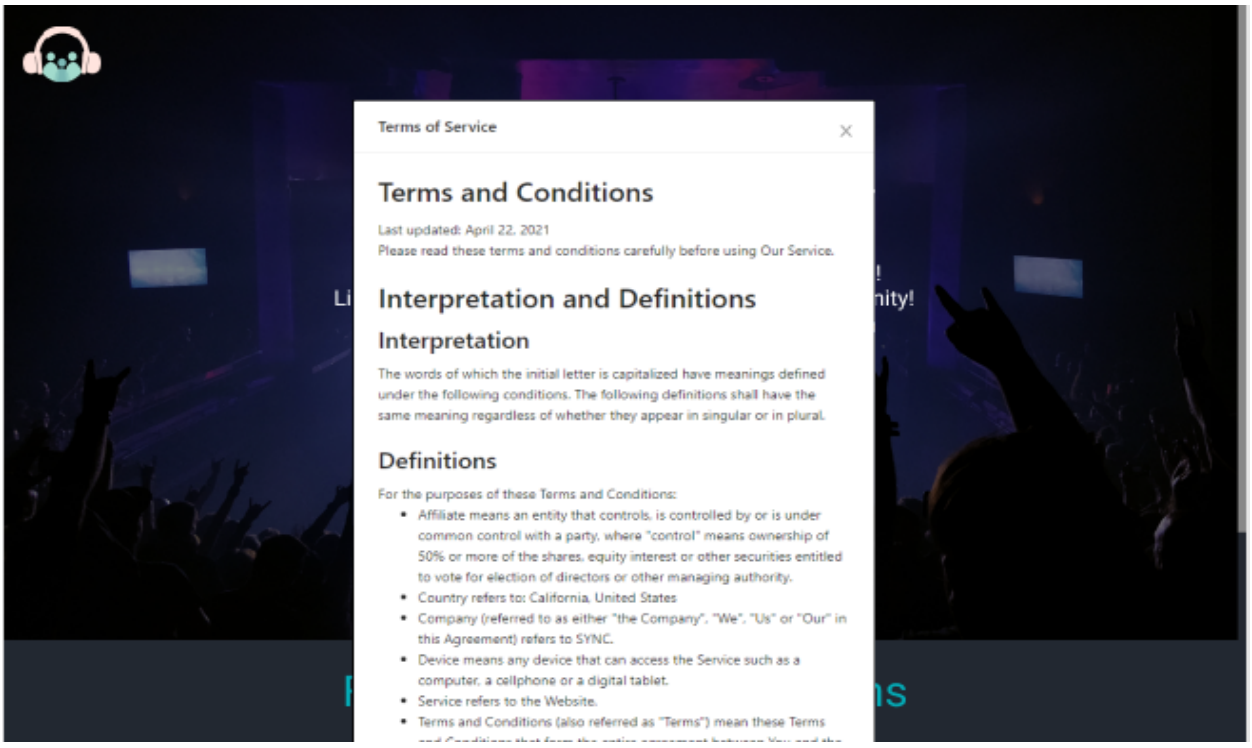
e. Navigate to http://3.17.66.0:3000/ it will bring you to this page



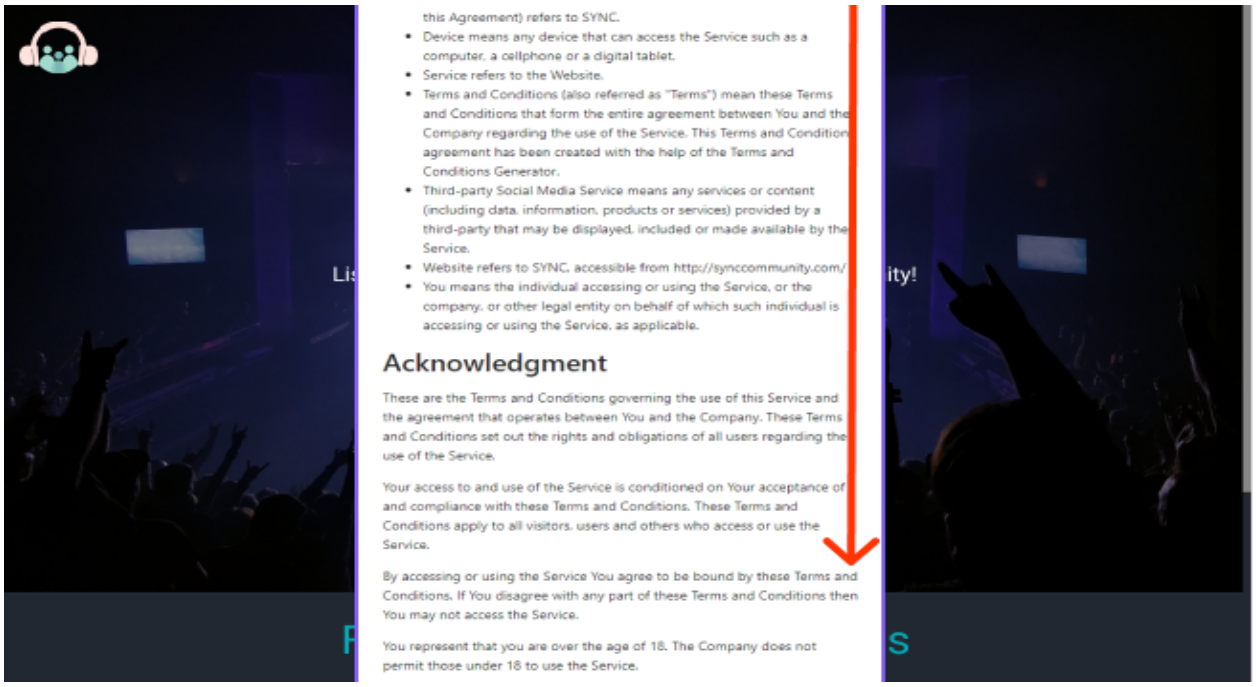
f. Select the ‘Terms and Services’ link located under the ‘Contine to SYNC’ button



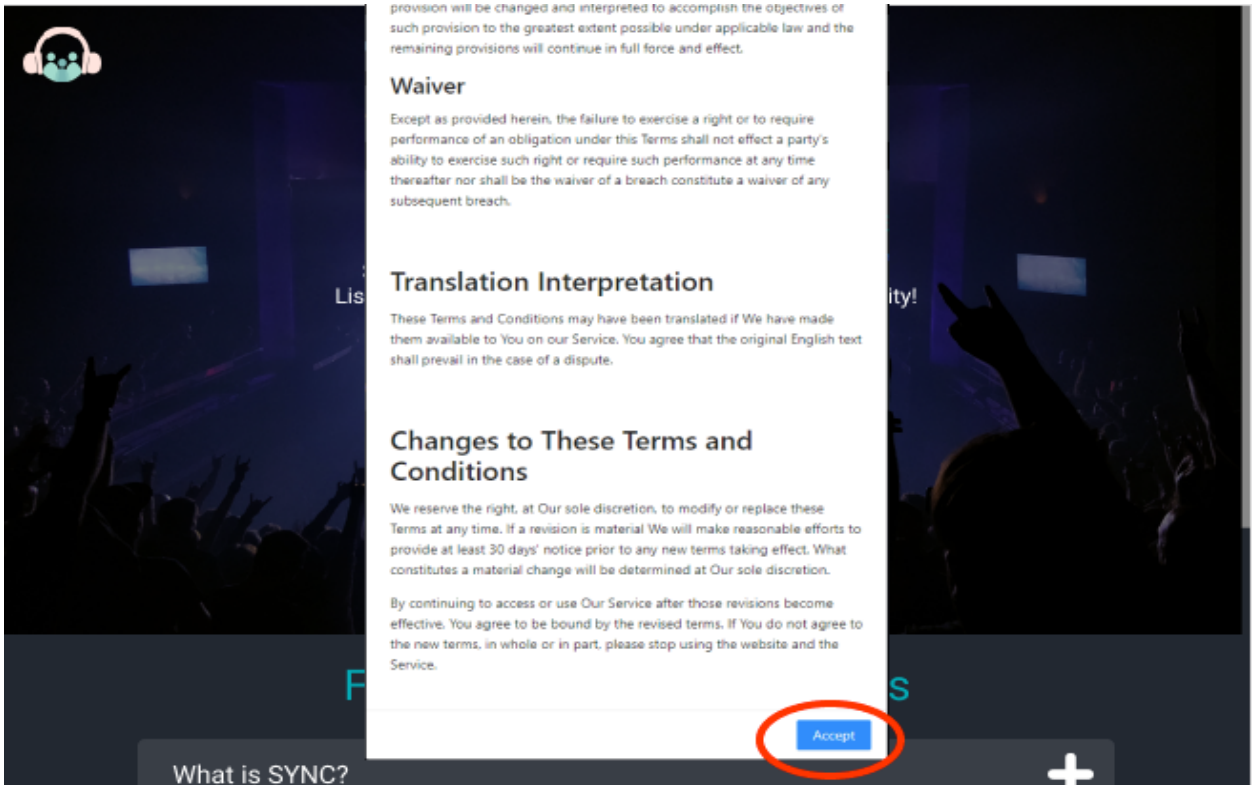
g. selected a pop up window will appear. It will look like this:



You will need to read and scroll through the Terms and Conditions to view all its content



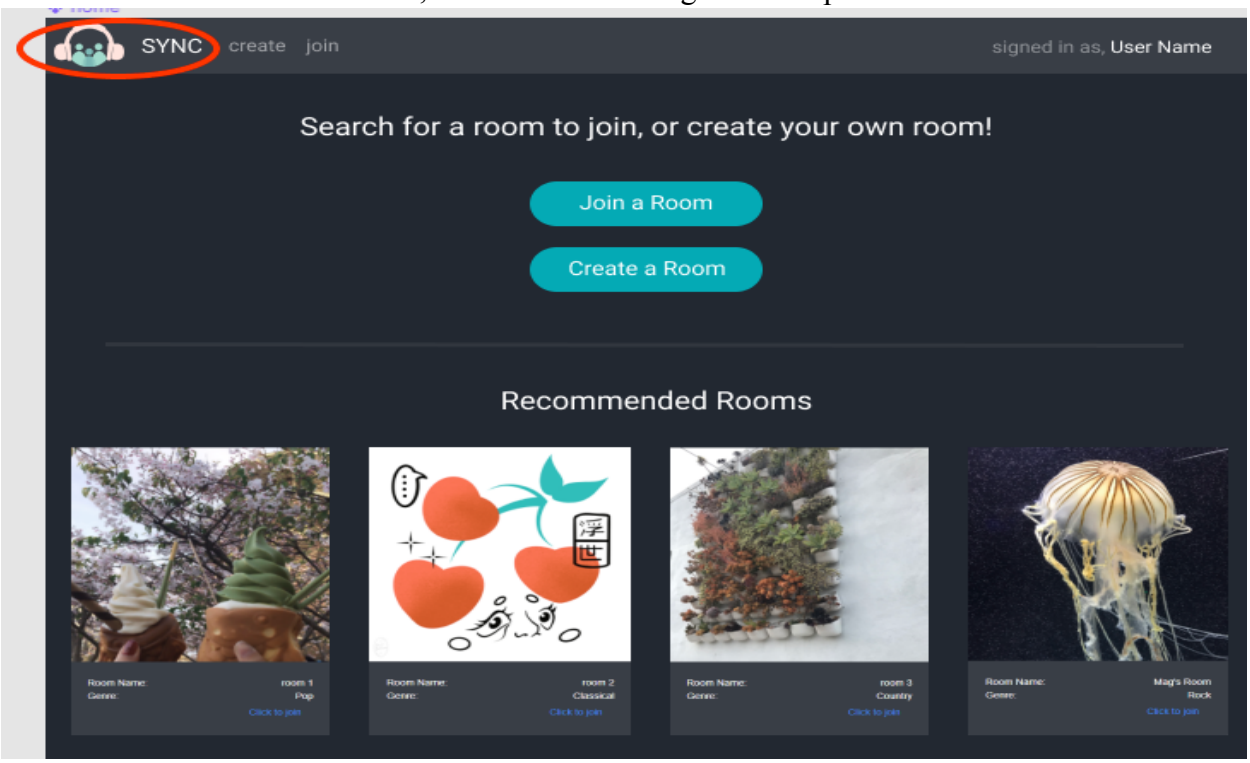
h. After scrolling and reading all the terms and conditions, select the accept button



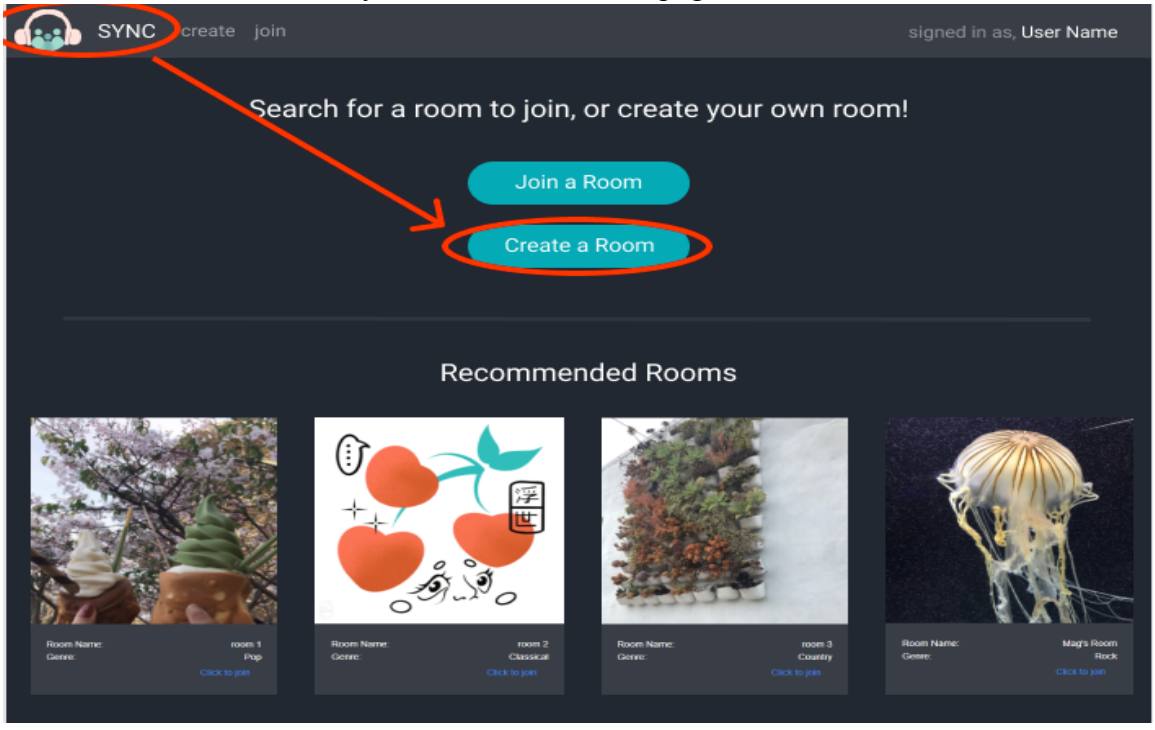
3. **Create a room**

   a. There are a couple places to create rooms. I will guide you through both locations

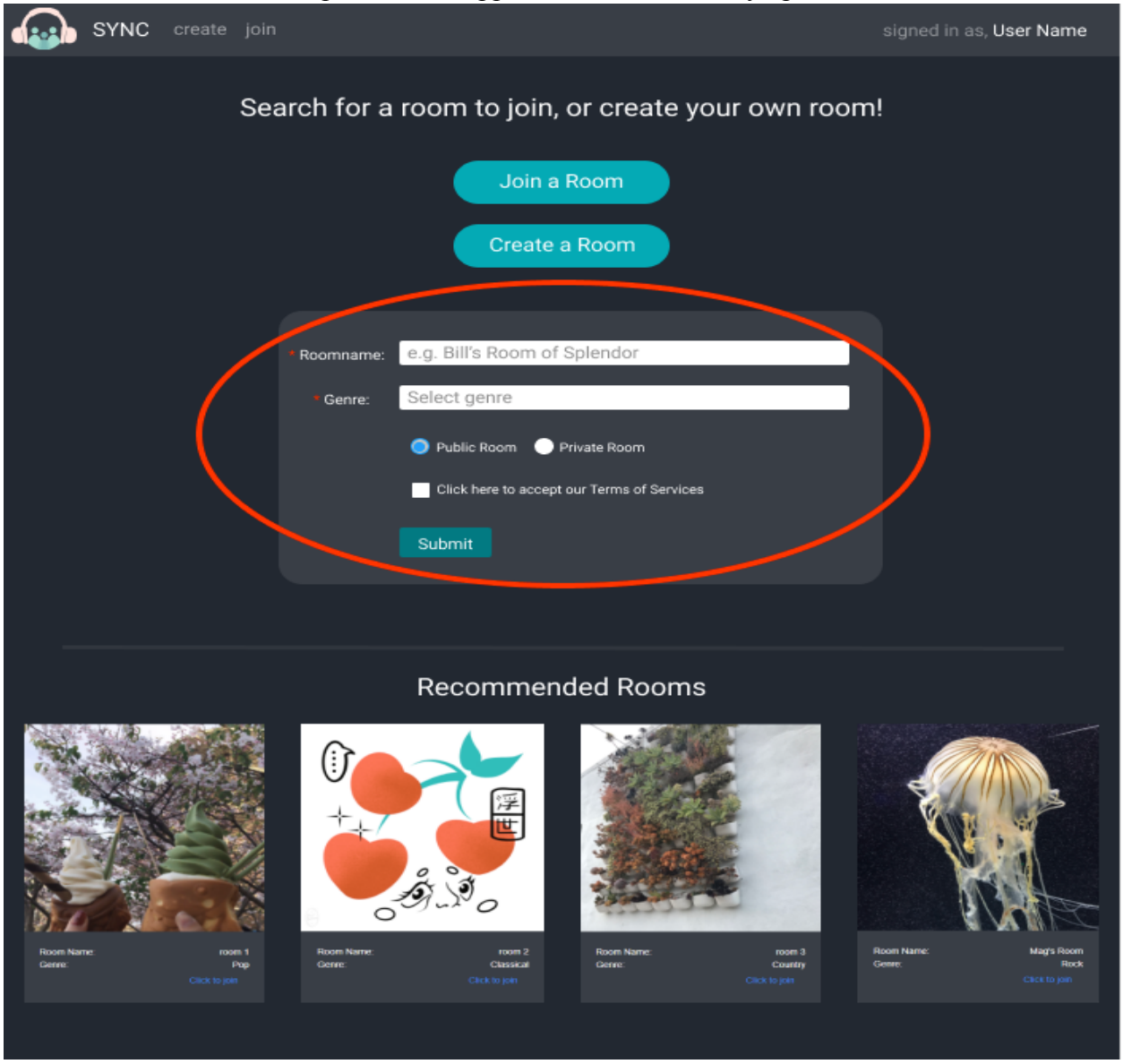
      i. Navigate to http://3.17.66.0:3000/Home by clicking this link or if already within site, select the SYNC logo at the top left



      ii. Once you arrive to the home page select the 'create a room' button

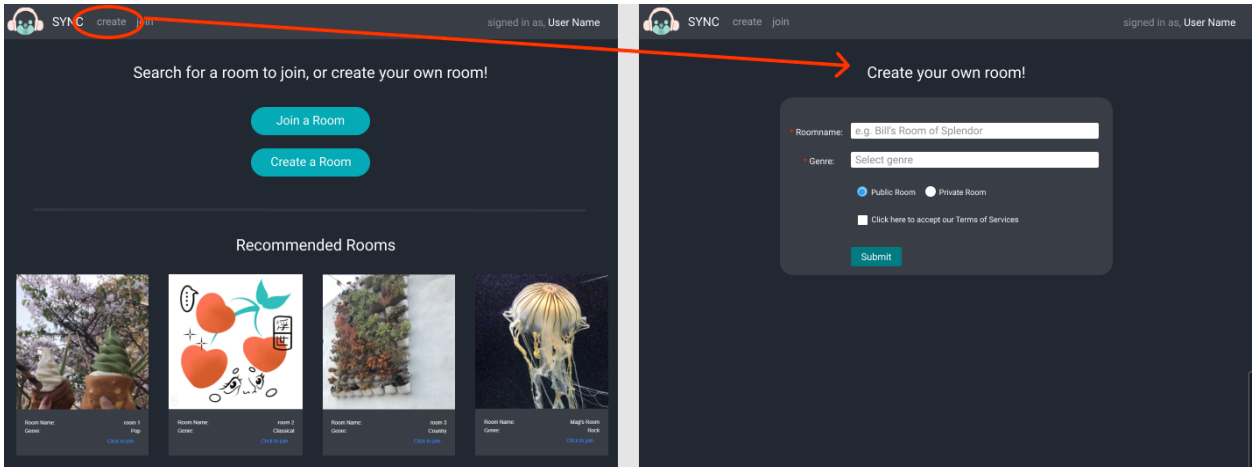


iii. A drop down will appear with the necessary options to create a room.



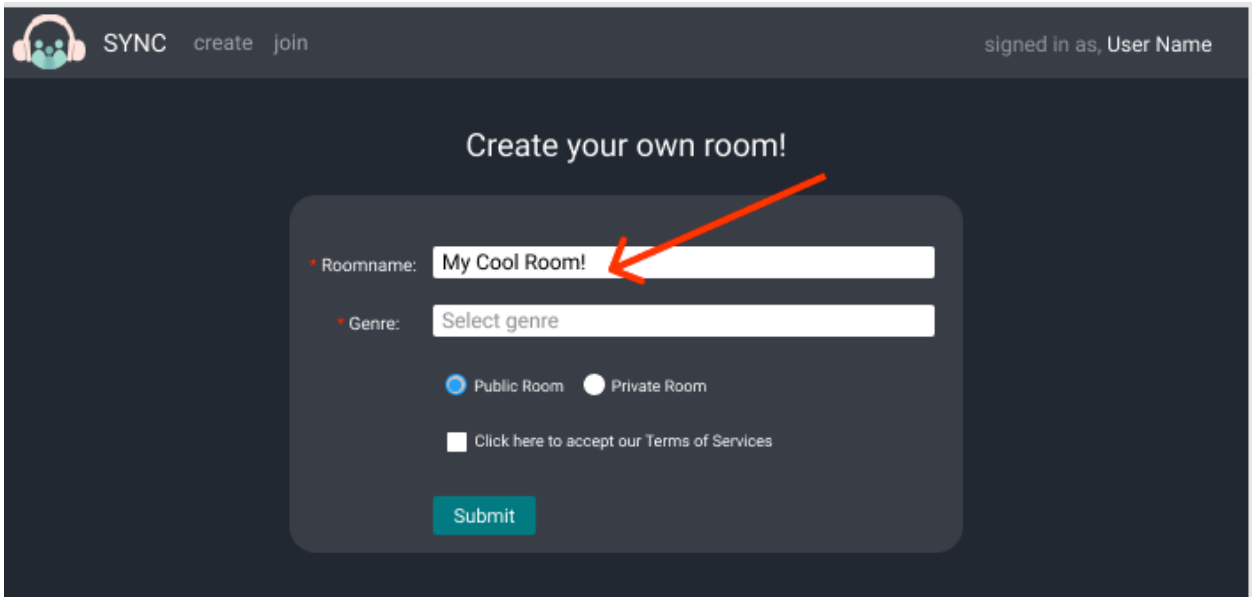
The **second** way to create a room is as such:

I. Navigate to http://3.17.66.0:3000/Createpage by selecting provided a link or (if already in site) selecting the ‘create’ button in the nav bar located at the top of every page within the site. It will take you to the crea a room page

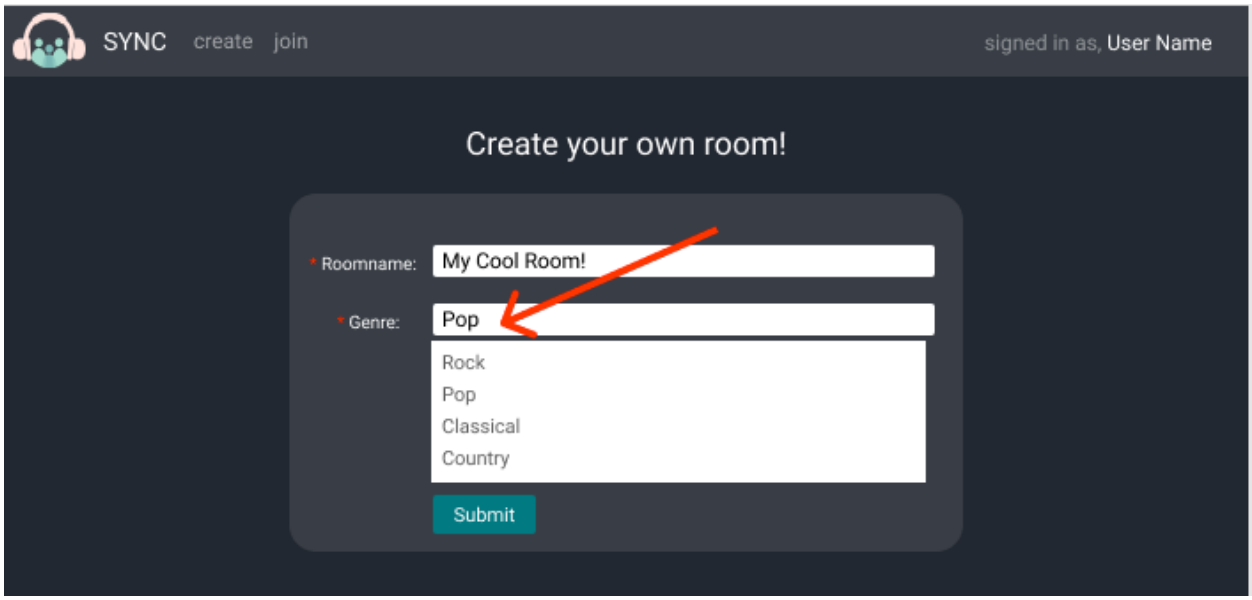


b. The next thing is to fill the text boxes.

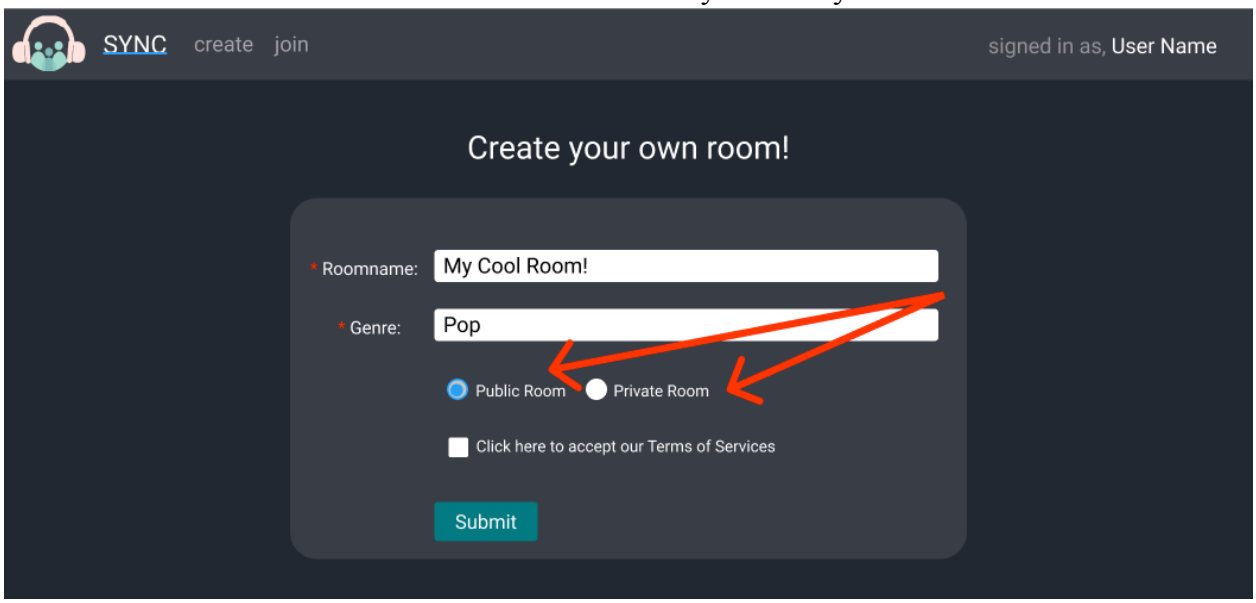
i. Type in a name for the room you are creating. This is uniquely yours.



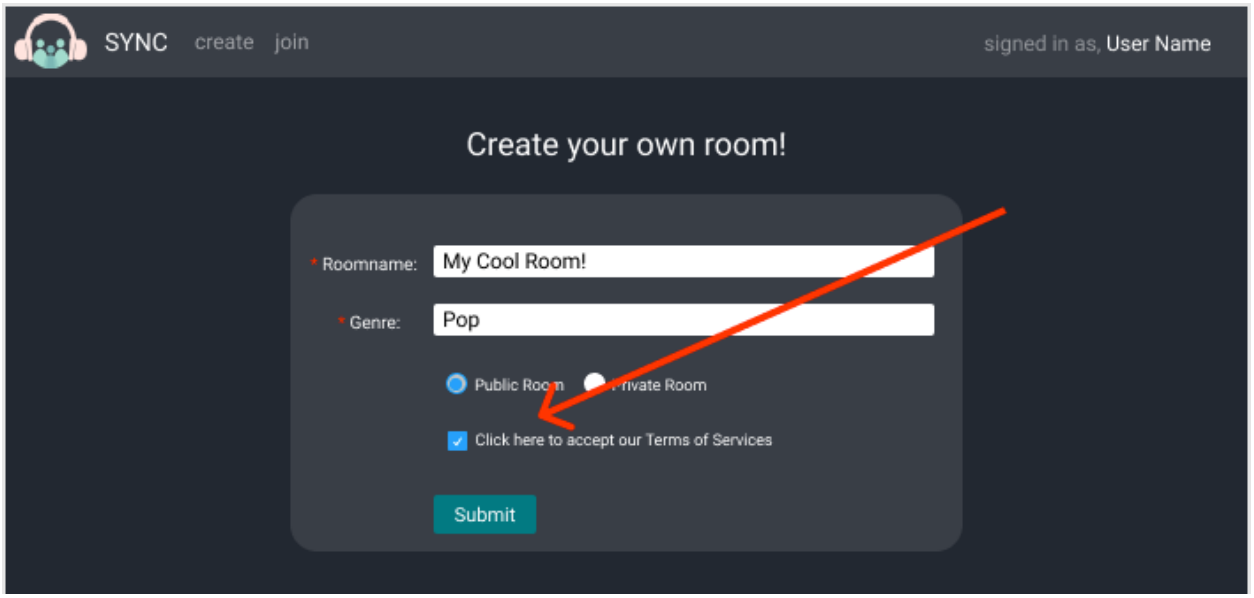
ii. Pick a genre: once you select the text box a drop down menu appears, allowing you to select genres such as pop or classical.



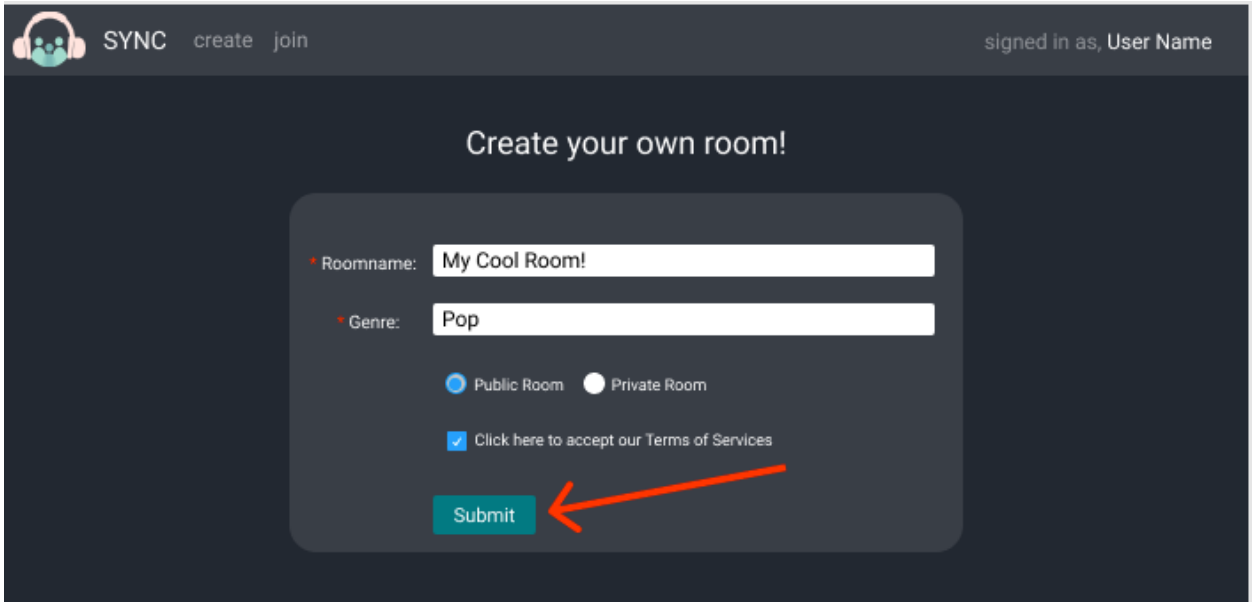
iii. Select ‘Public’ or ‘Private’ for the room type. This allows you to chose if all users can see the room or only to those you allow to see



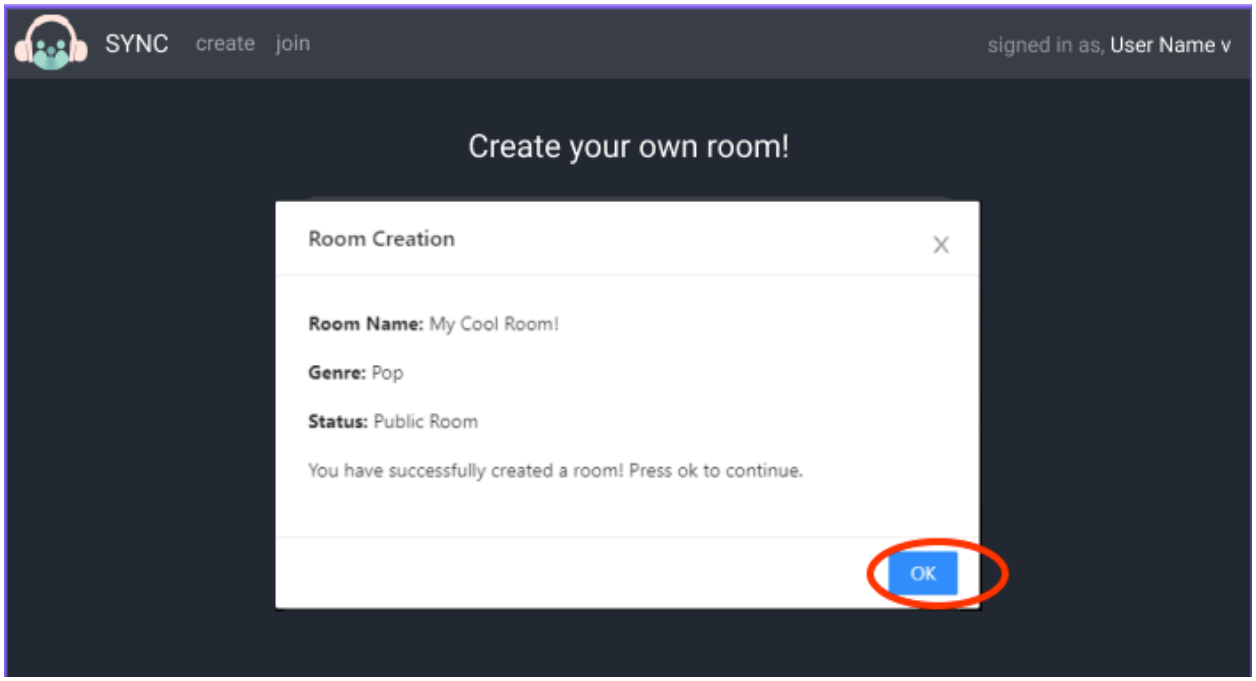
iv. Check off ‘Accept Terms of Services’ by selecting the box beside the text that reads ‘Click here to accept our Terms of Services’



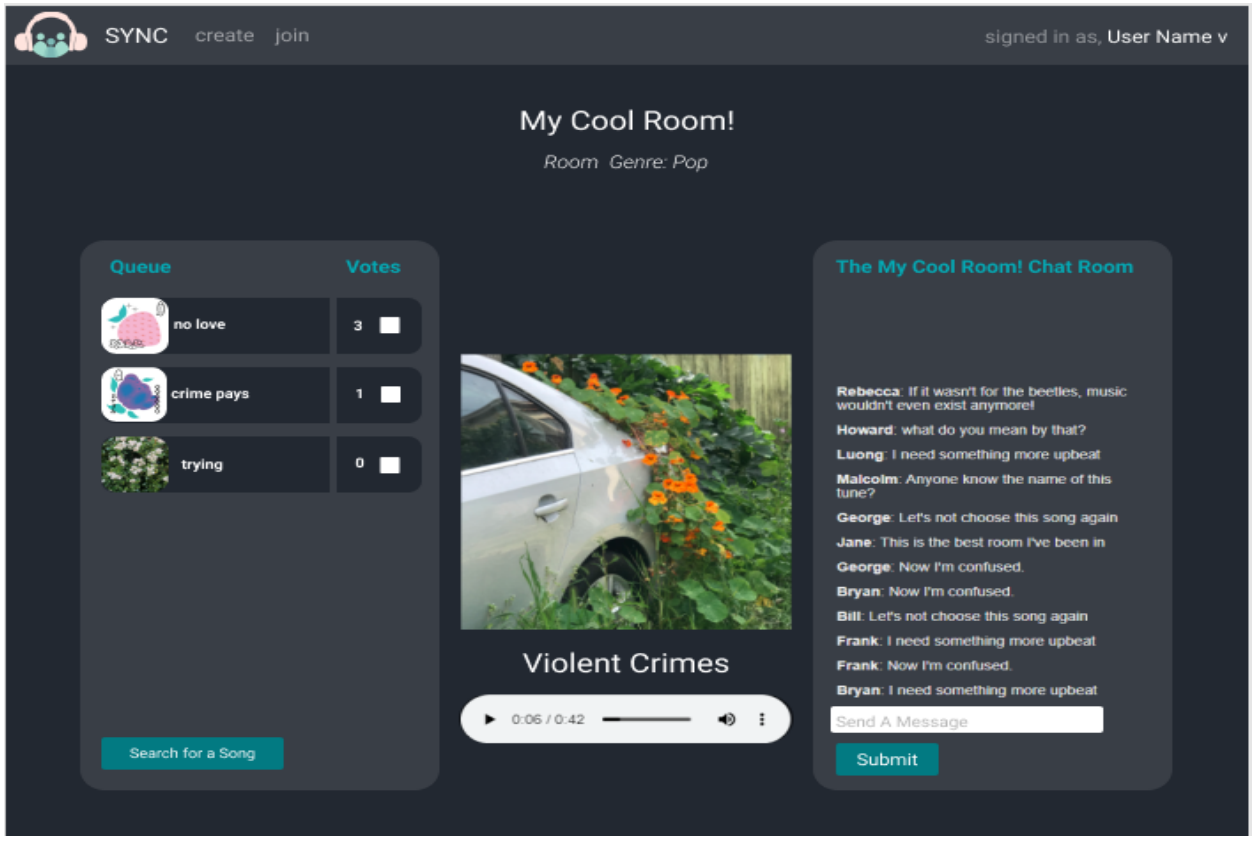
v. Select the ‘Submit’ button when you are satisfied with your selection



c. Select the ‘ok’ button the view room



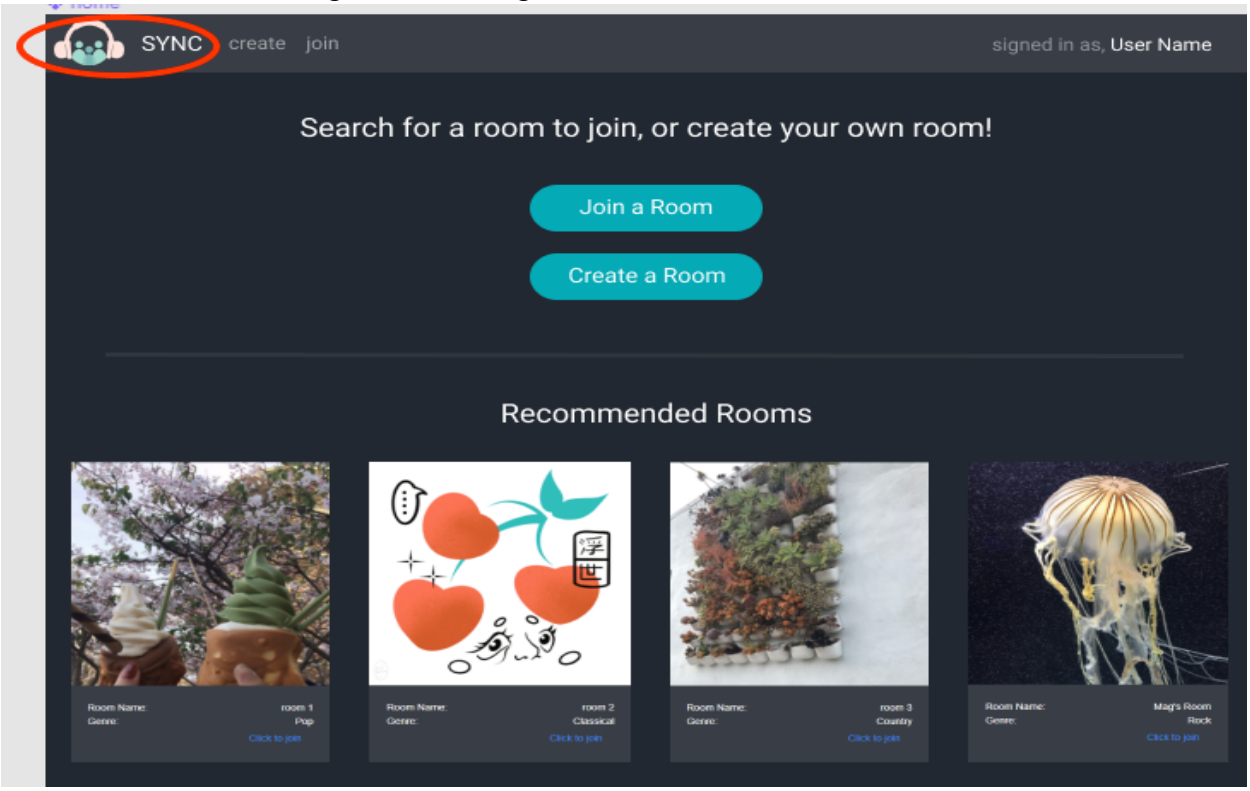
d. You are now in your created room!



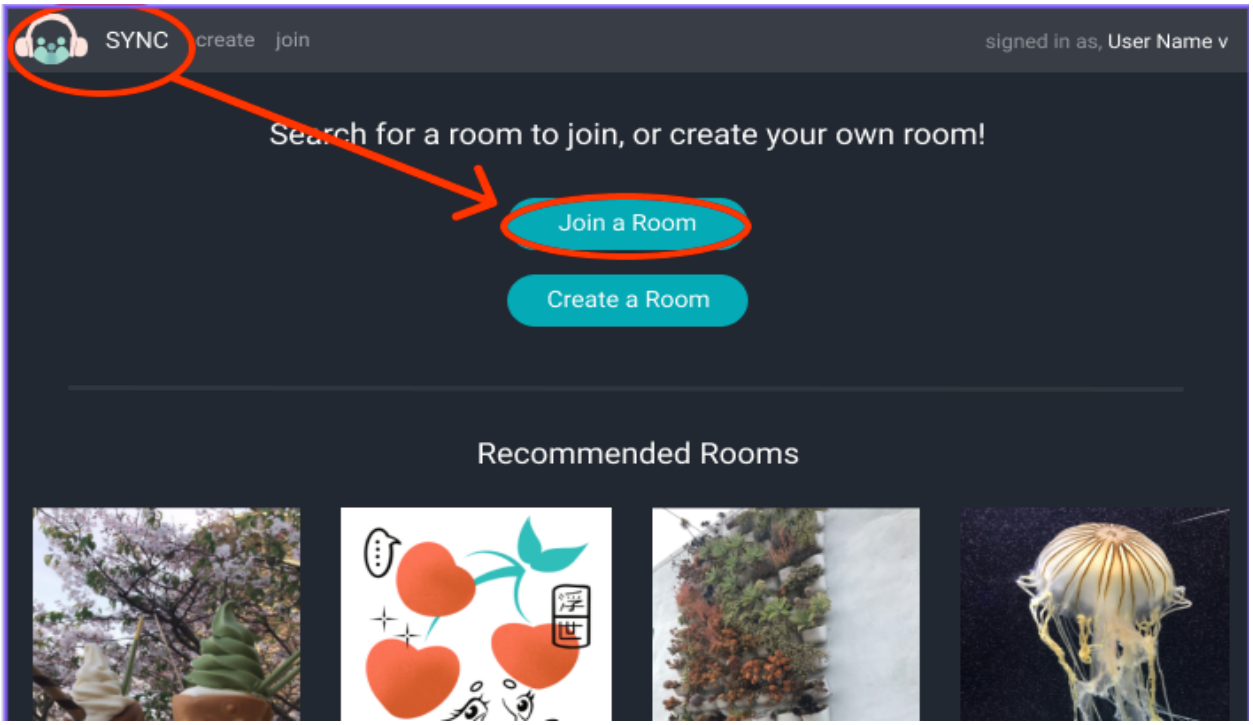
4. Search and Join a room

There are a couple places to join a room. I will guide you through both locations

a. Navigate to http://3.17.66.0:3000/Home by clicking this link or if already logged in, selecting the SYNC logo



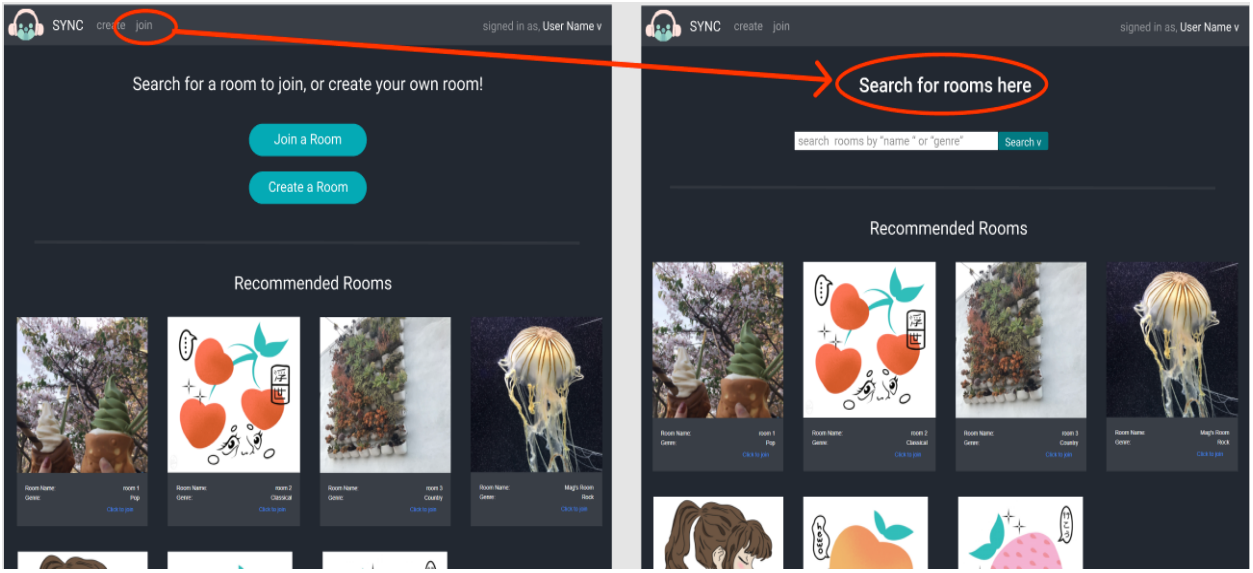
b. Once you arrive to the home page select the 'join a room' button



The **second** way to join a room is as such:

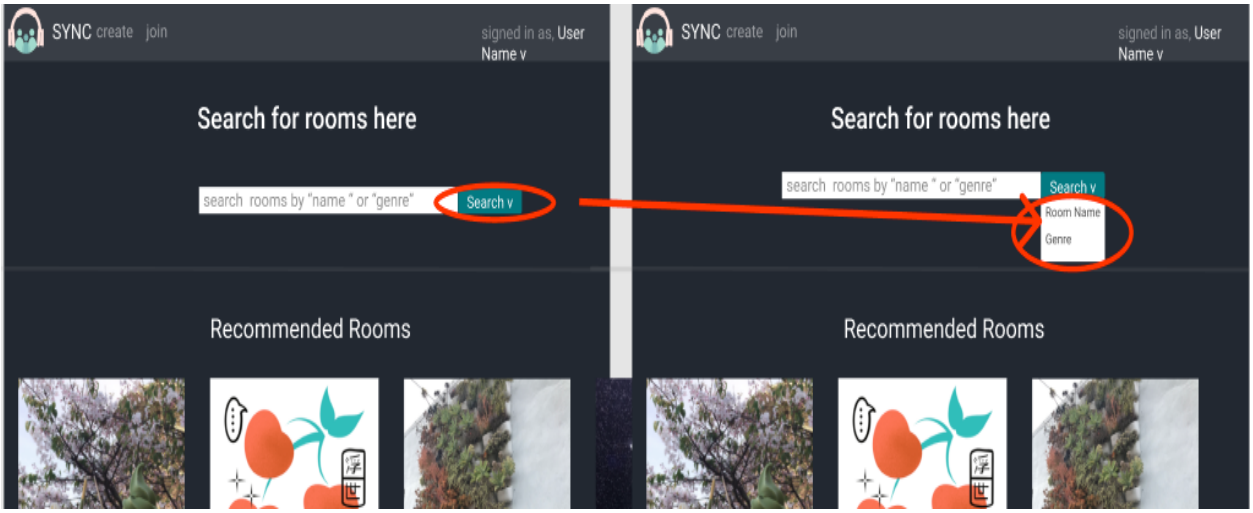
I. Navigate to http://3.17.66.0:3000/Join by selecting provided a link or (if

already in site) selecting the join button in the nav bar located at the top of every page within the site

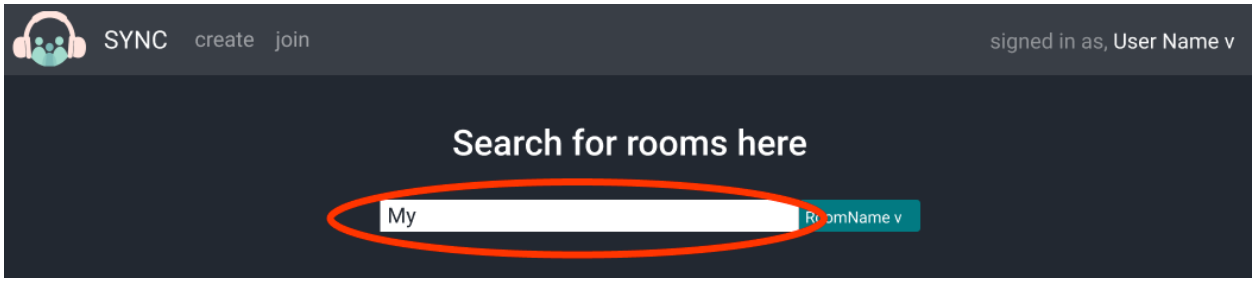


c. The next thing is to search for something! You can search the room you have created!

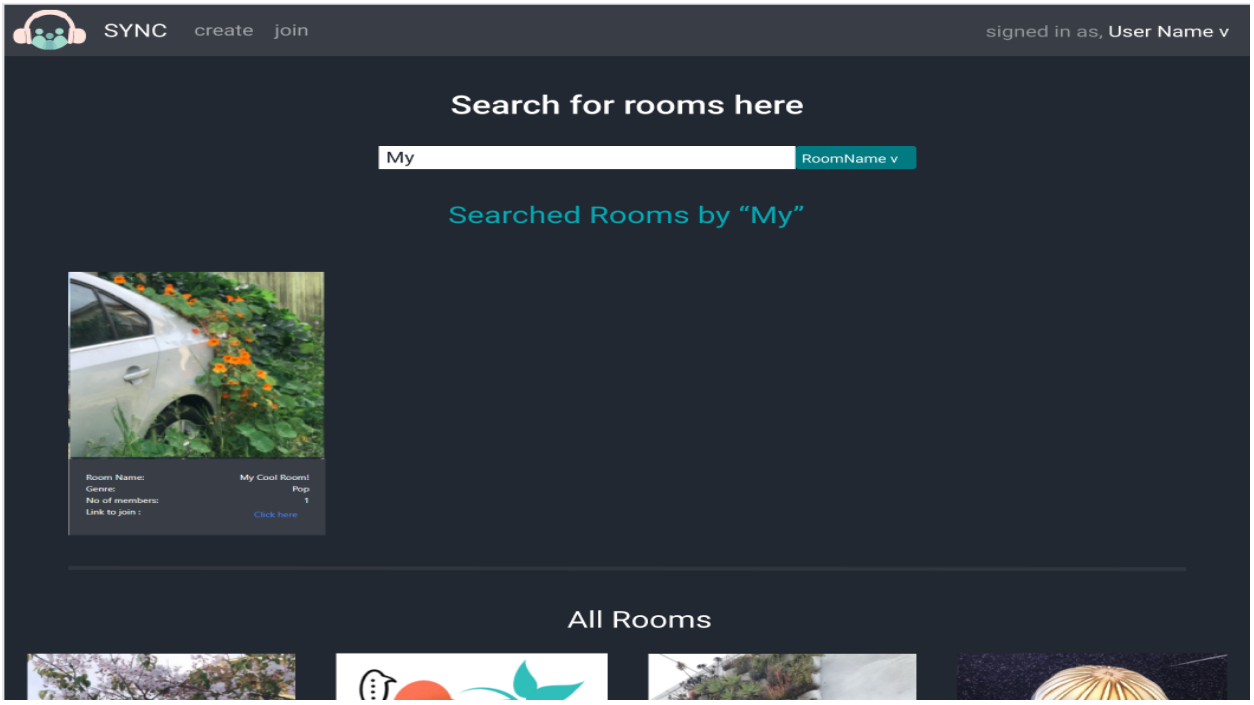
i. You may filter using the drop down menu beside the search bar. Select the turquoise 'Search v' button for the menu to appear. Select 'Room Name'



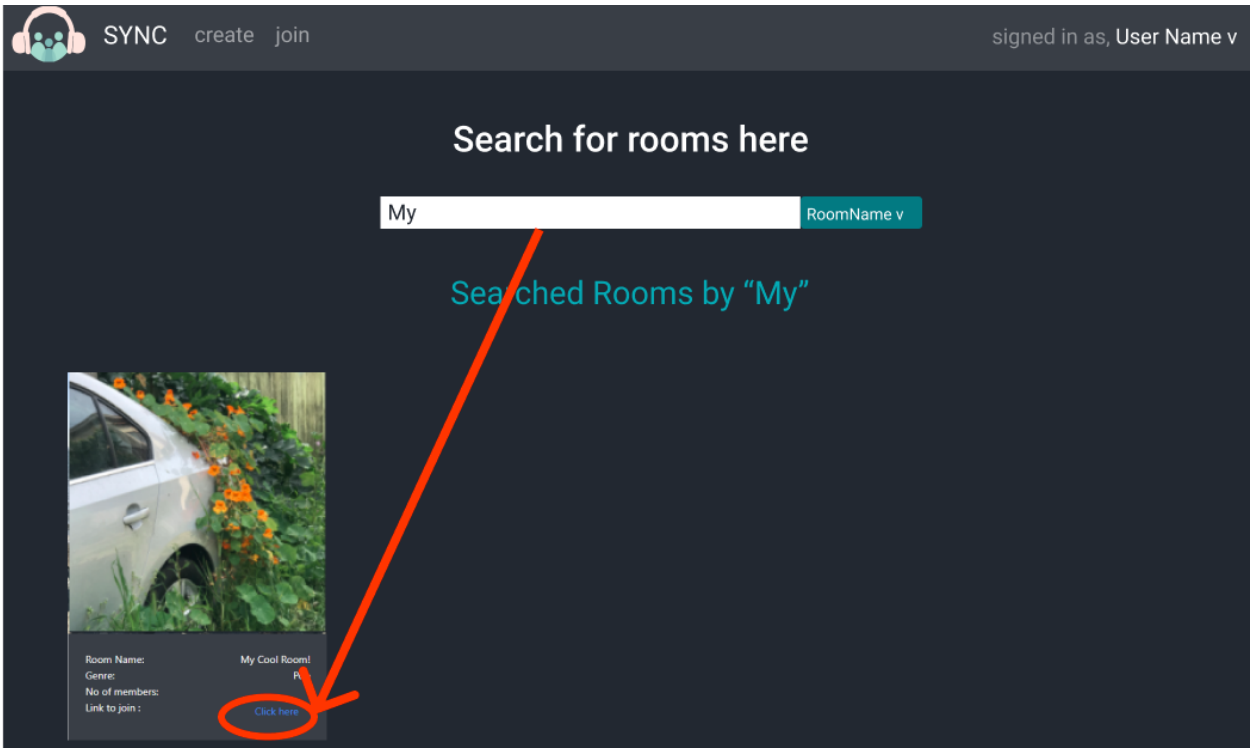
ii. Type in 'My'. or whatever you'd like to name your room



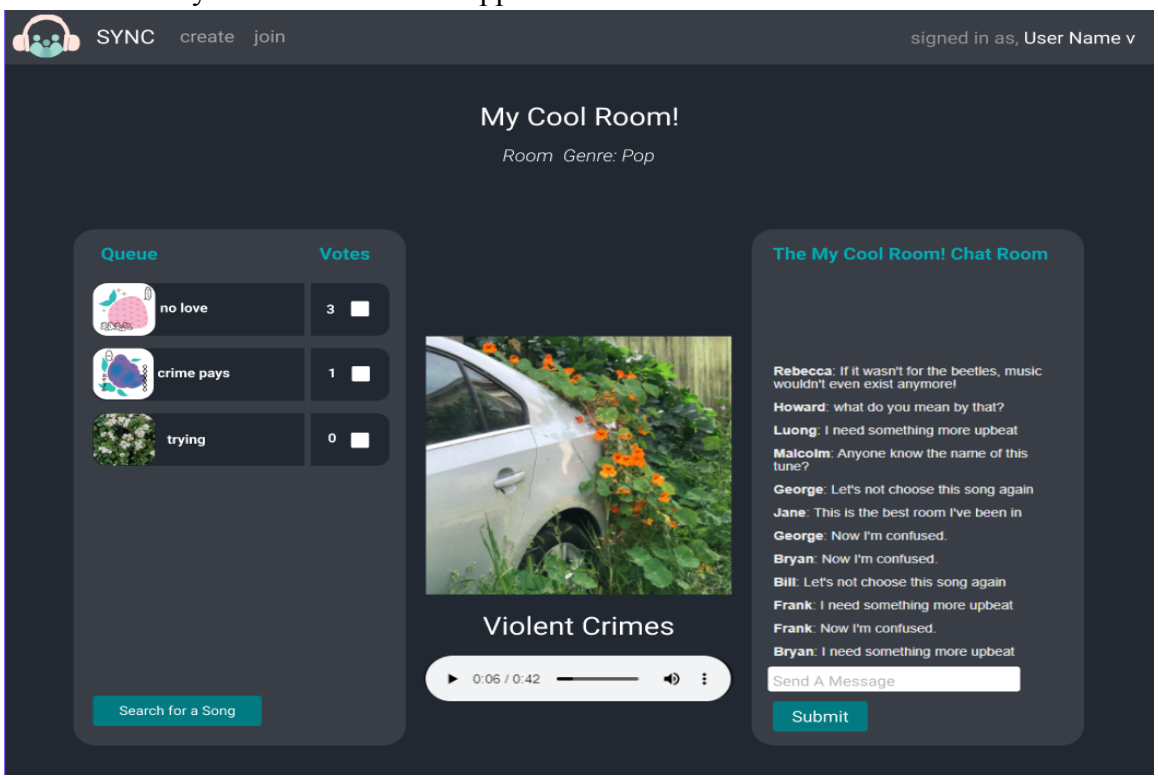
iii. Your room named ‘My Cool Room!’ will appear under the ‘Searched Rooms by “My” section. It should look like this:



d. On the room you would like to join there is a link to join the room. Select the blue link that reads ‘ Click here’ on the room you would like to join

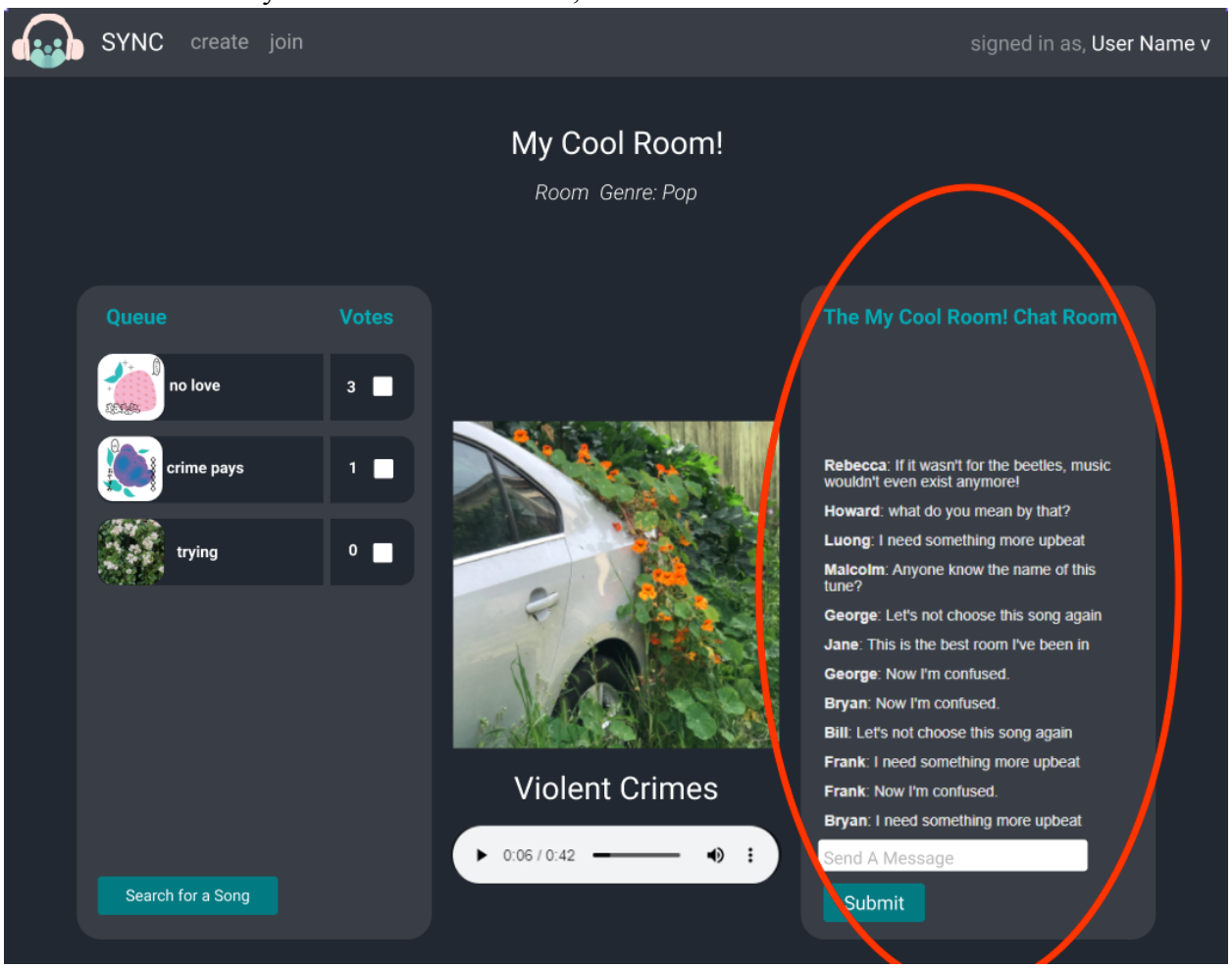


e. After selecting the ‘Click here’ link on a room card, you should be redirected into your room! It should appear in a new window as such:

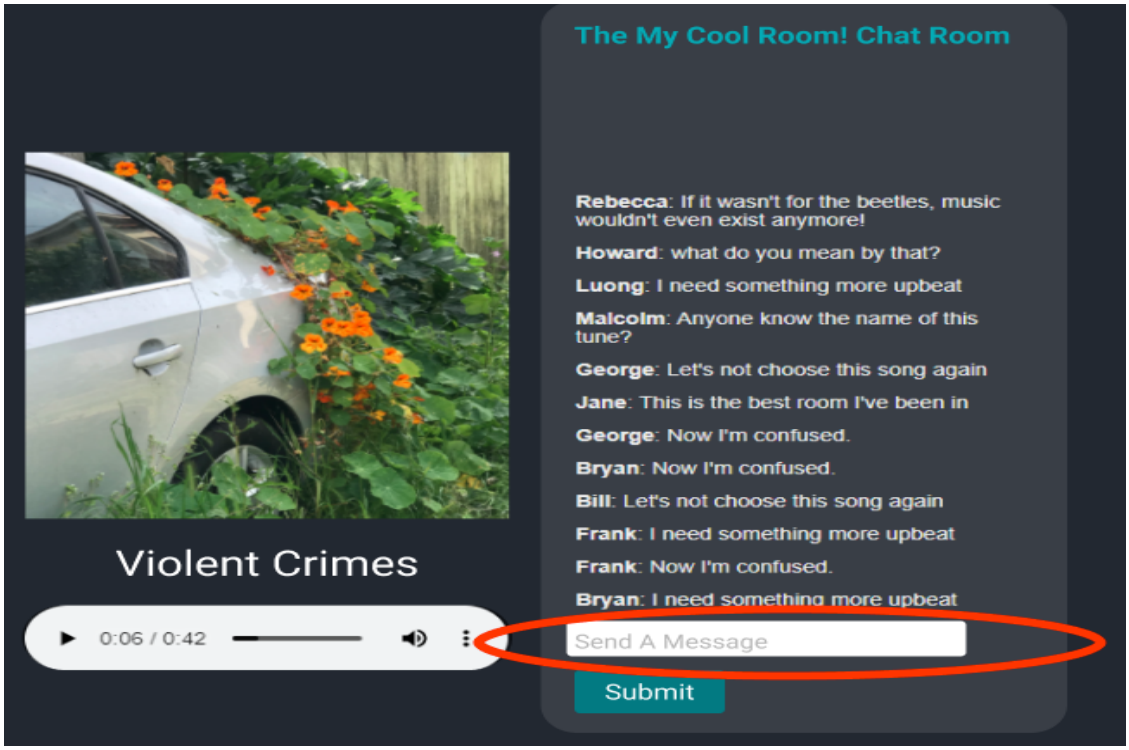


5. **Text in chat within a room**
   a. Navigate to your room or a room by following step 4.

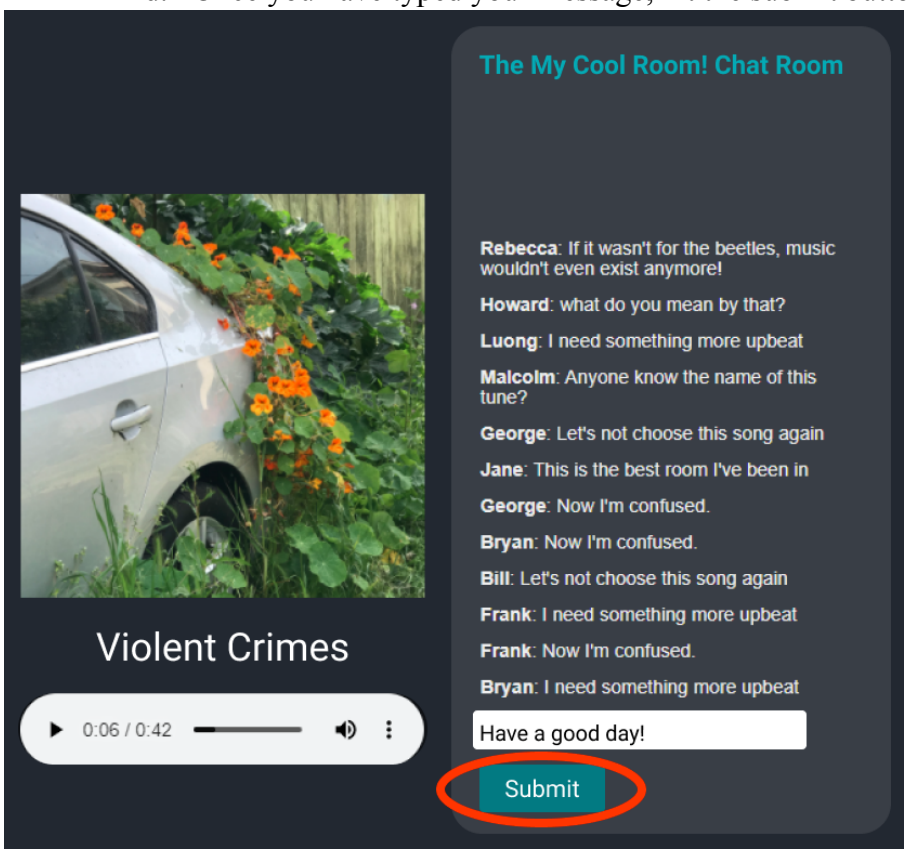
b. Once you have entered a room, locate the chat room



c. At the bottom of the chat window there is a text box where you are able to text something to other people in the room. Type your comment in the white text box.



d. Once you have typed your message, hit the submit button below the text be sent



e. After submitting the message, the message will displayed in the chat above for all to see

Violent Crimes

0:06 / 0:42

The My Cool Room! Chat Room

Rebecca: If it wasn't for the beetles, music wouldn't even exist anymore!

Howard: what do you mean by that?

Luong: I need something more upbeat

Malcolm: Anyone know the name of this tune?

George: Let's not choose this song again

Jane: This is the best room I've been in

George: Now I'm confused.

Bryan: Now I'm confused.

Bill: Let's not choose this song again

Frank: I need something more upbeat

Frank: Now I'm confused.

Bryan: I need something more upbeat

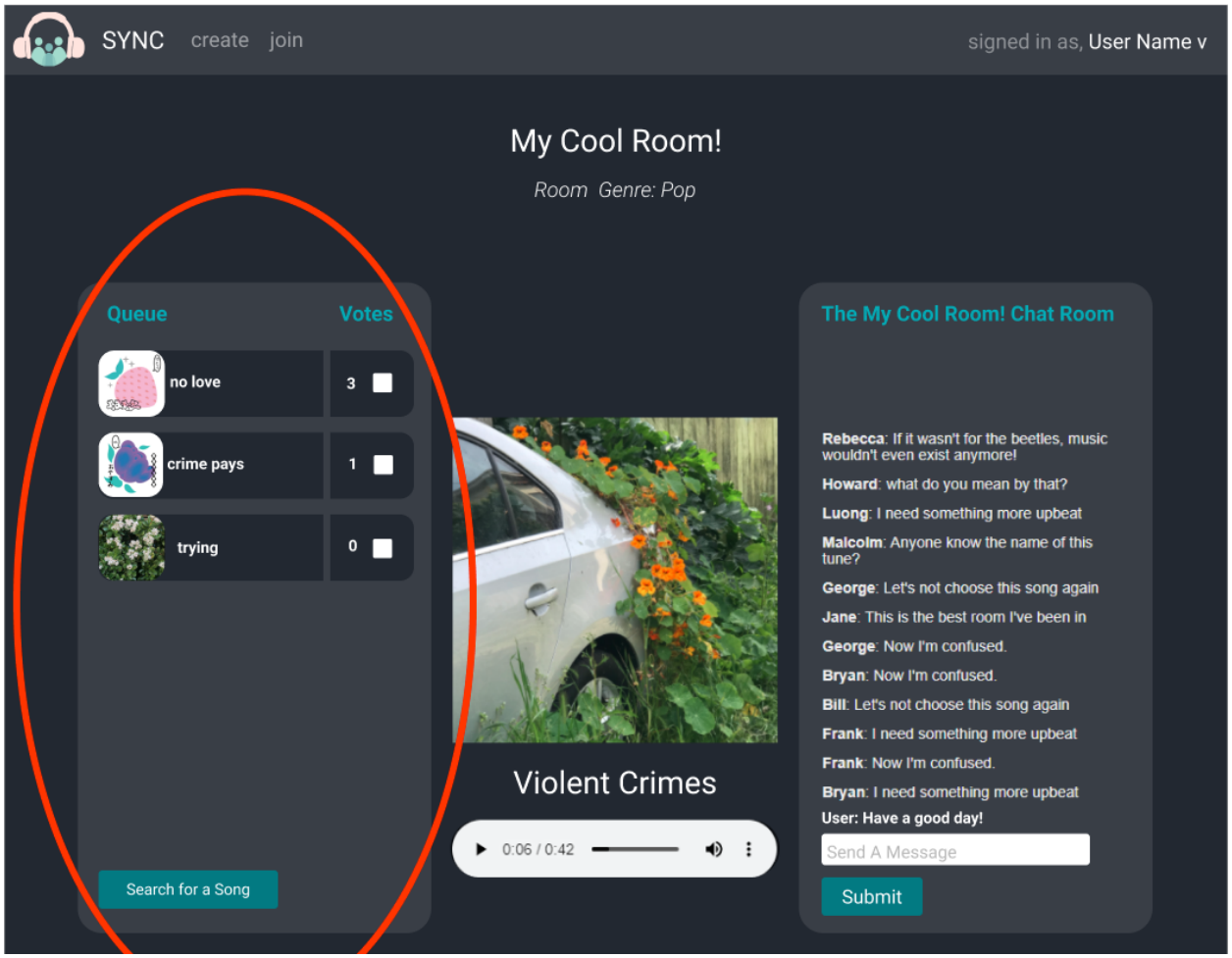
User: Have a good day!

Send A Message

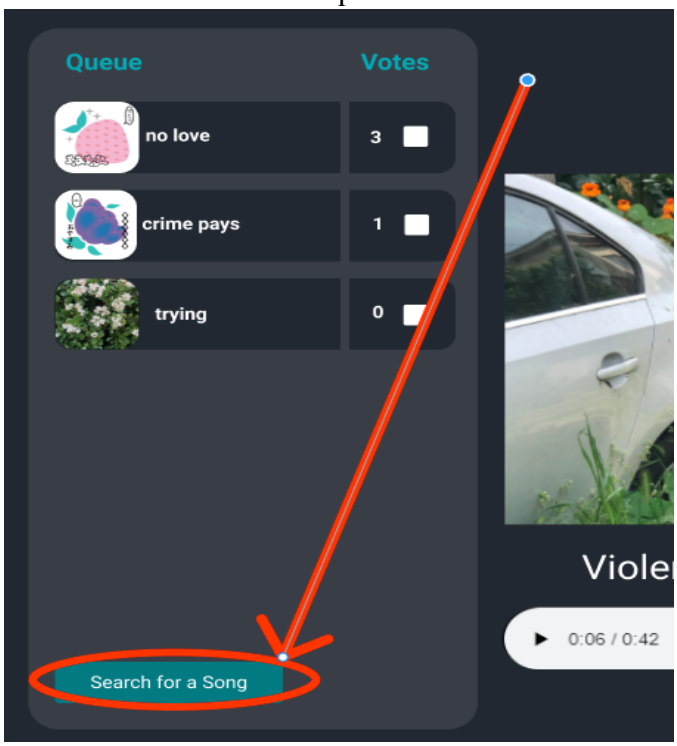
Submit

6. **Search and add song into queue**
    a. Navigate to your room or a room by following step 4.
    b. Once you have entered a room, locate the queue list located on the left of the screen

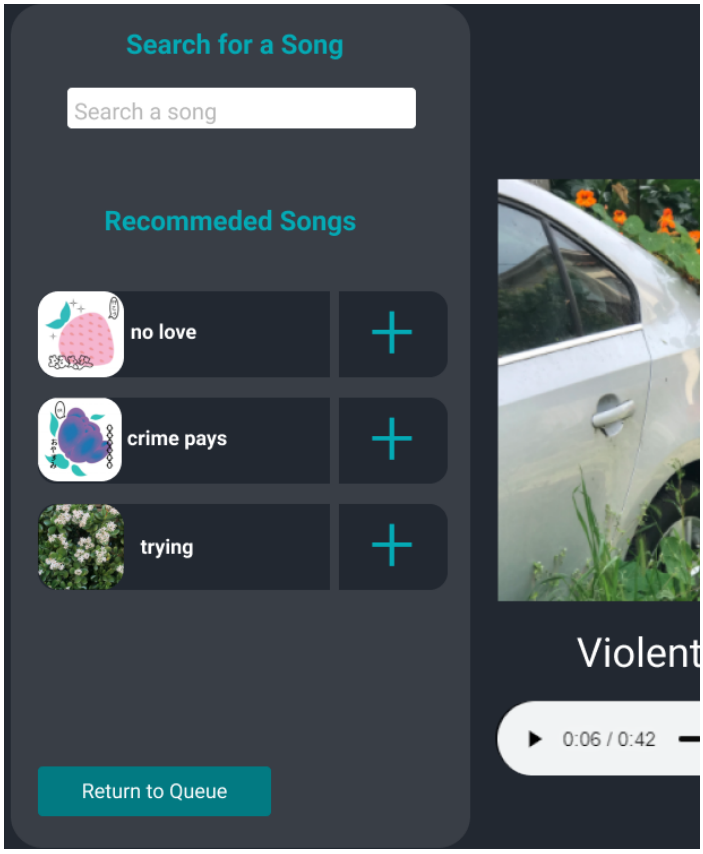
Here is listed all the songs in the queue, if any.



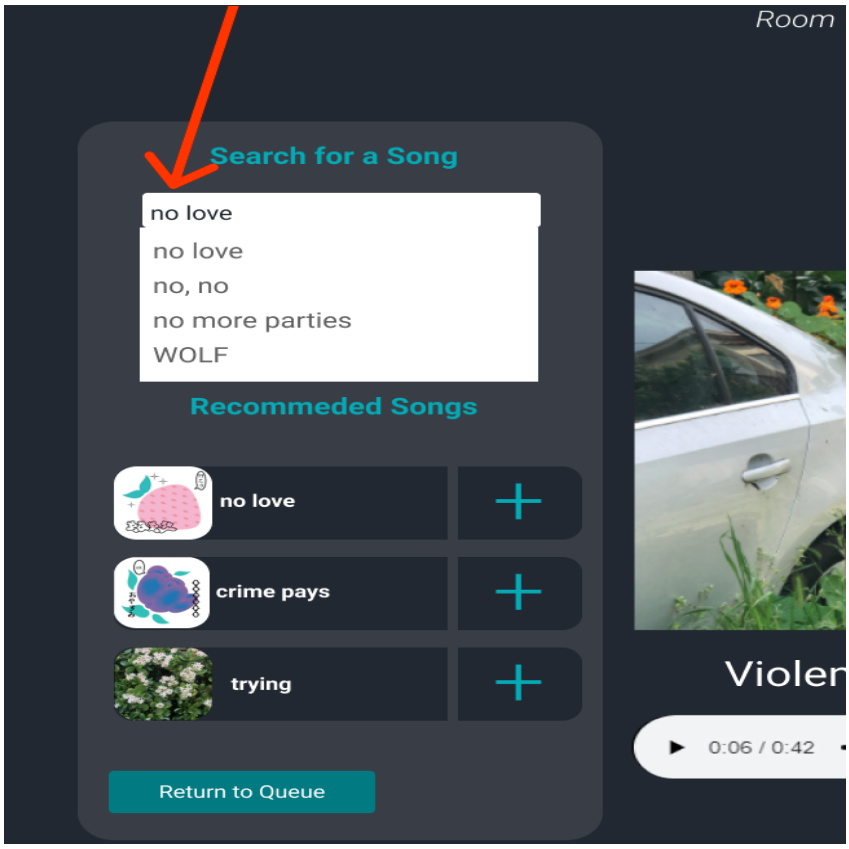
c. To add songs to queue select the ‘search for a song’ button located at the bottom left on the queue section



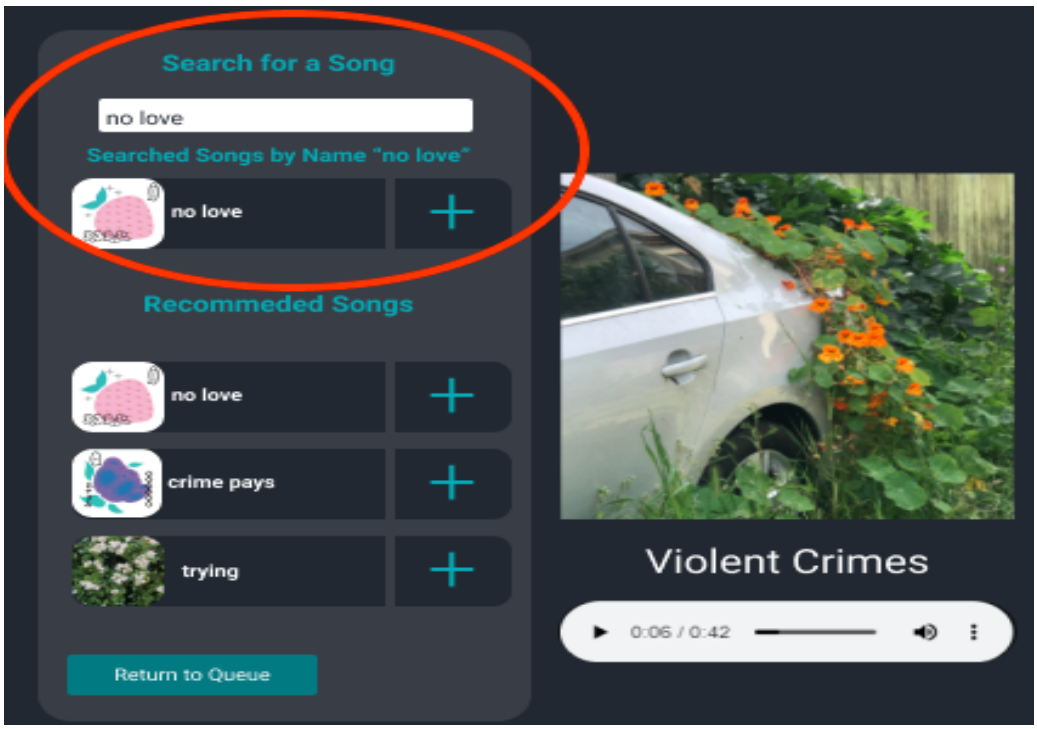
d. This will prompt a search bar above the queue to open.



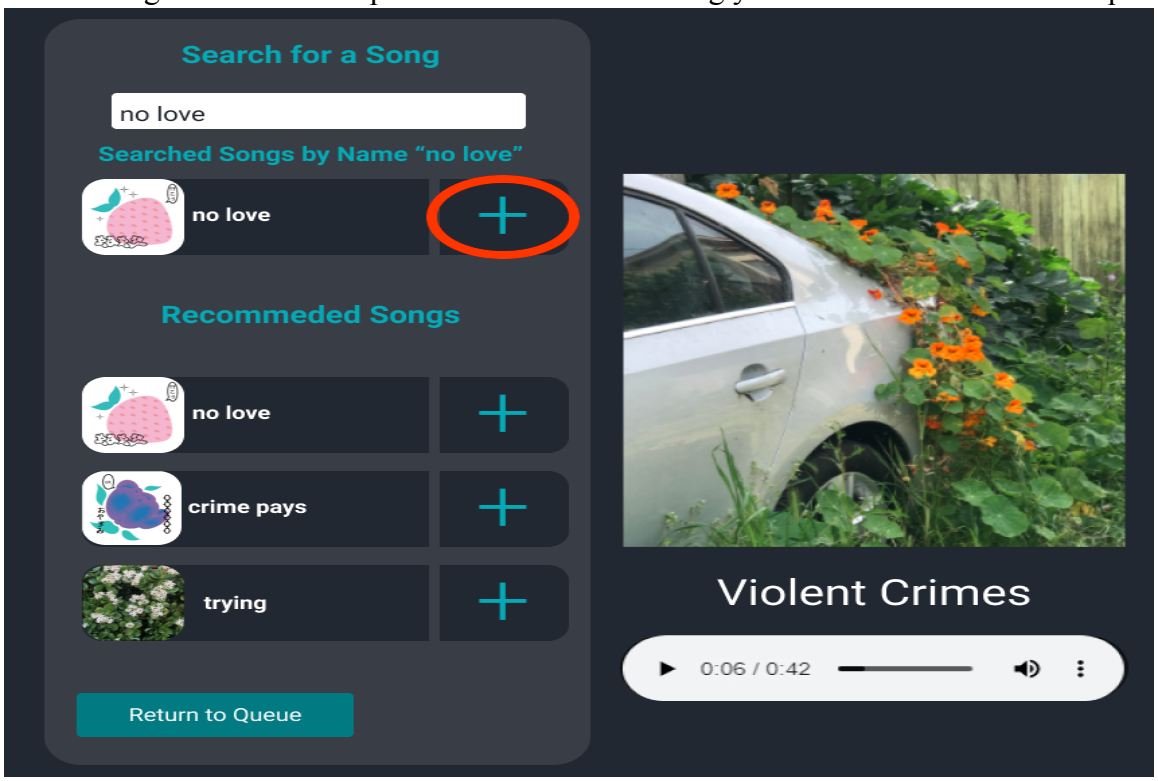
e. Type in the search bar a name of a song you would like to play. An example would be ‘no love’



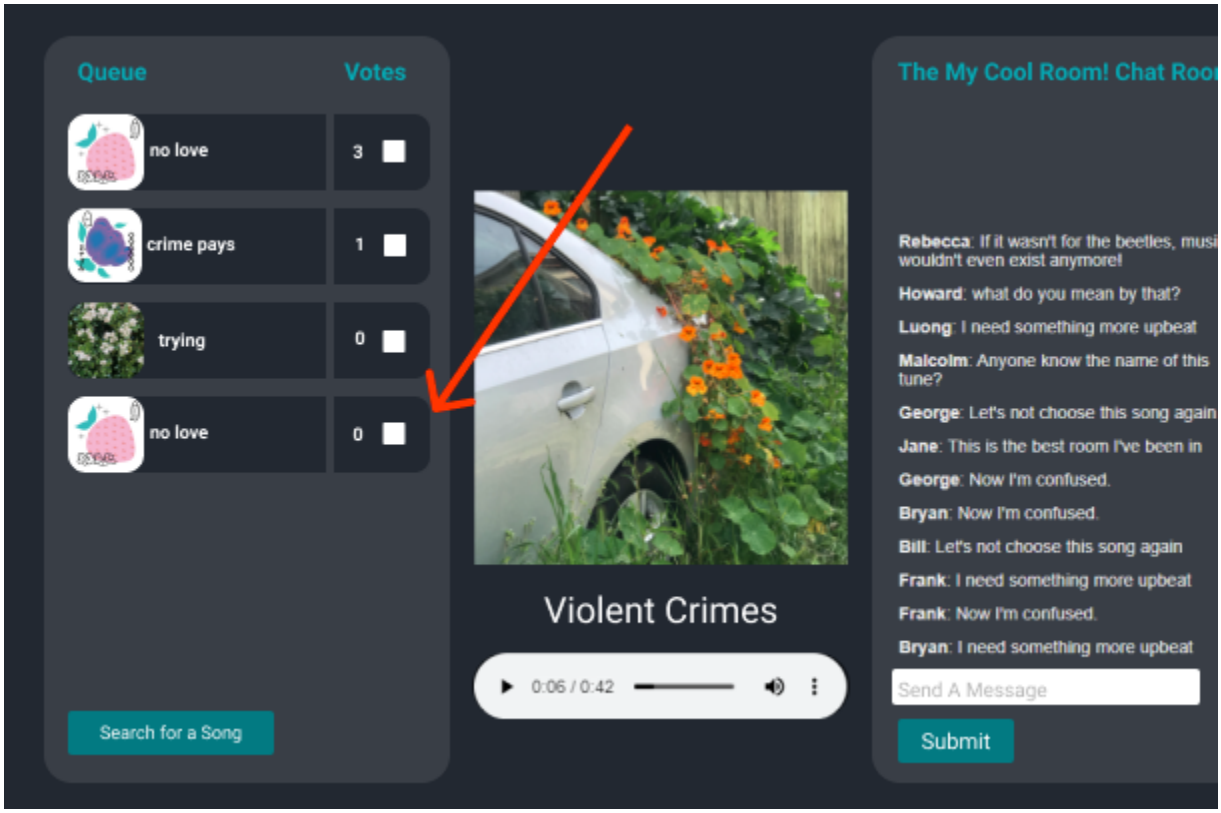
f. After typing in everything hit enter and the results would show up directly under the search bar



g. Select the '+' plus button next to the song you would like to add to the queue

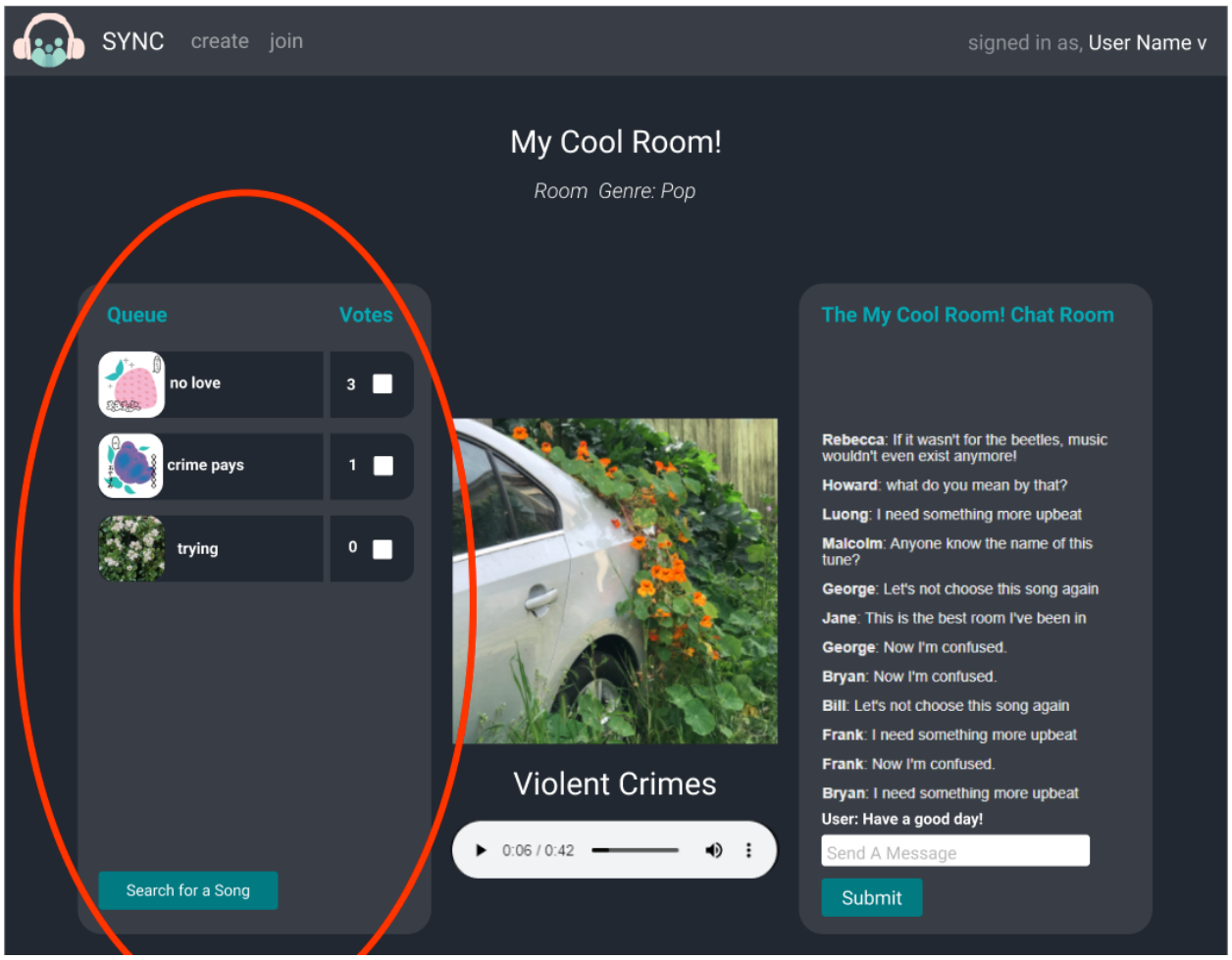


h. Once the plus button is selected the song added is displayed in the queue

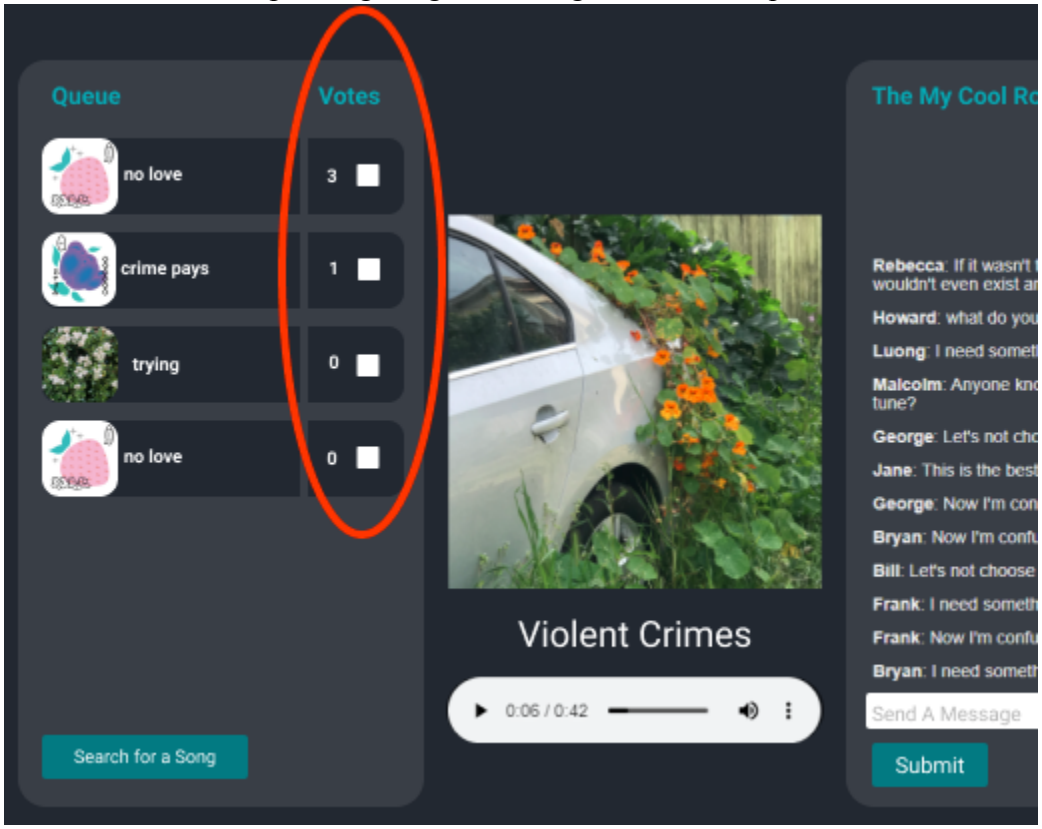


7. **Add a vote to a song in queue**
    a. Navigate to your or a room by following step 4.
    b. Once you have entered a room, locate the queue list located on the left of the screen

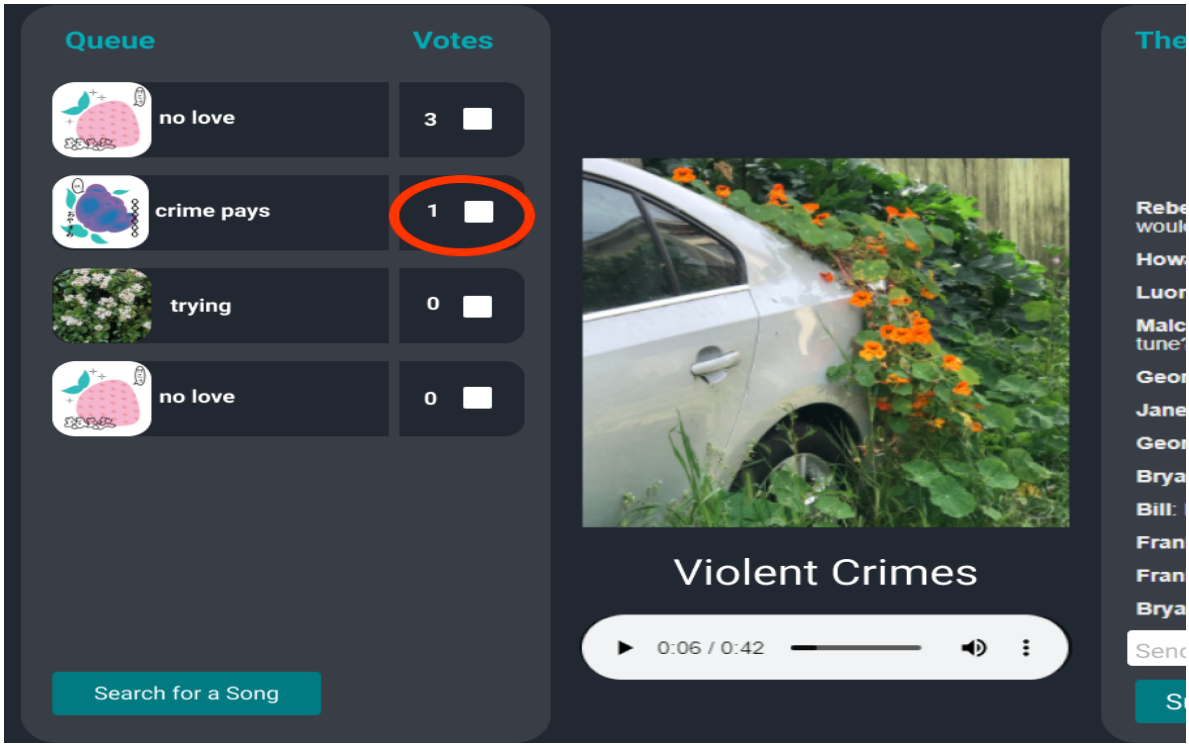
Here is listed all the songs in the queue if any.



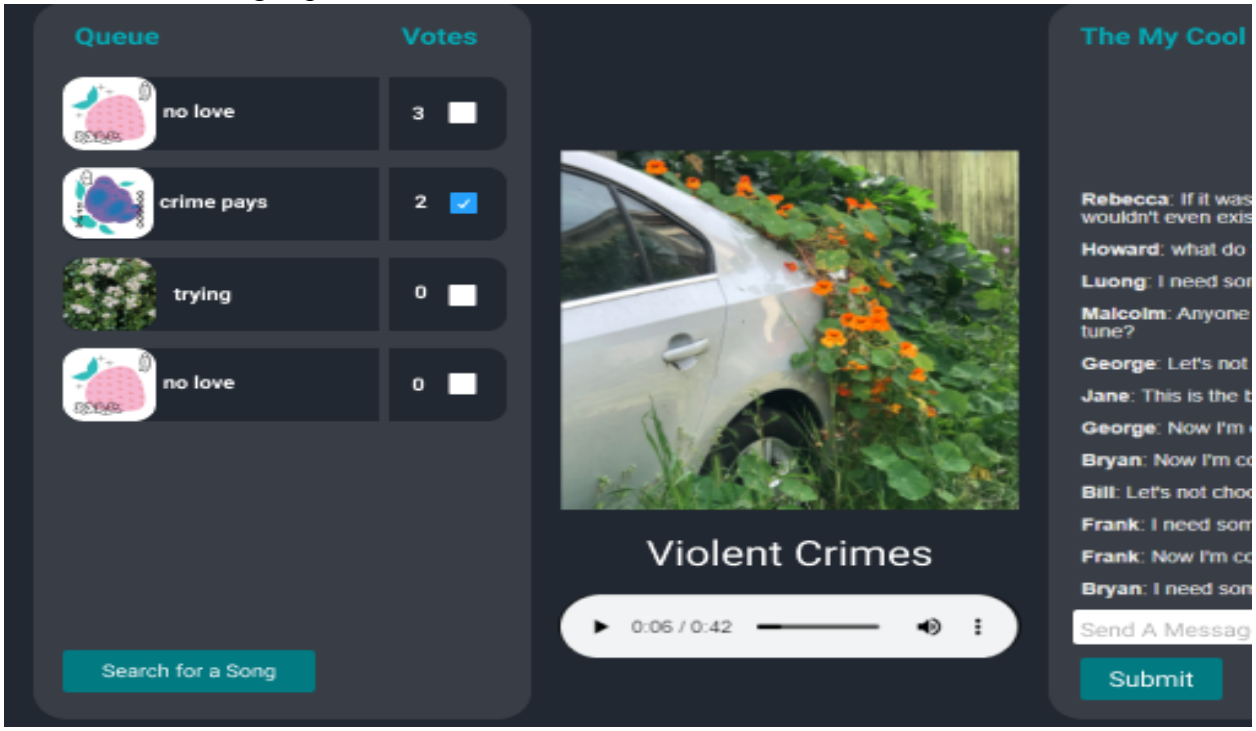
c. To add vote go to any of the songs in the queue: Navigate to the vote column to corresponding songs on the right side of the queue list



d. Choose a song you'd like to add a vote for, and select the vote check box corresponding to that song

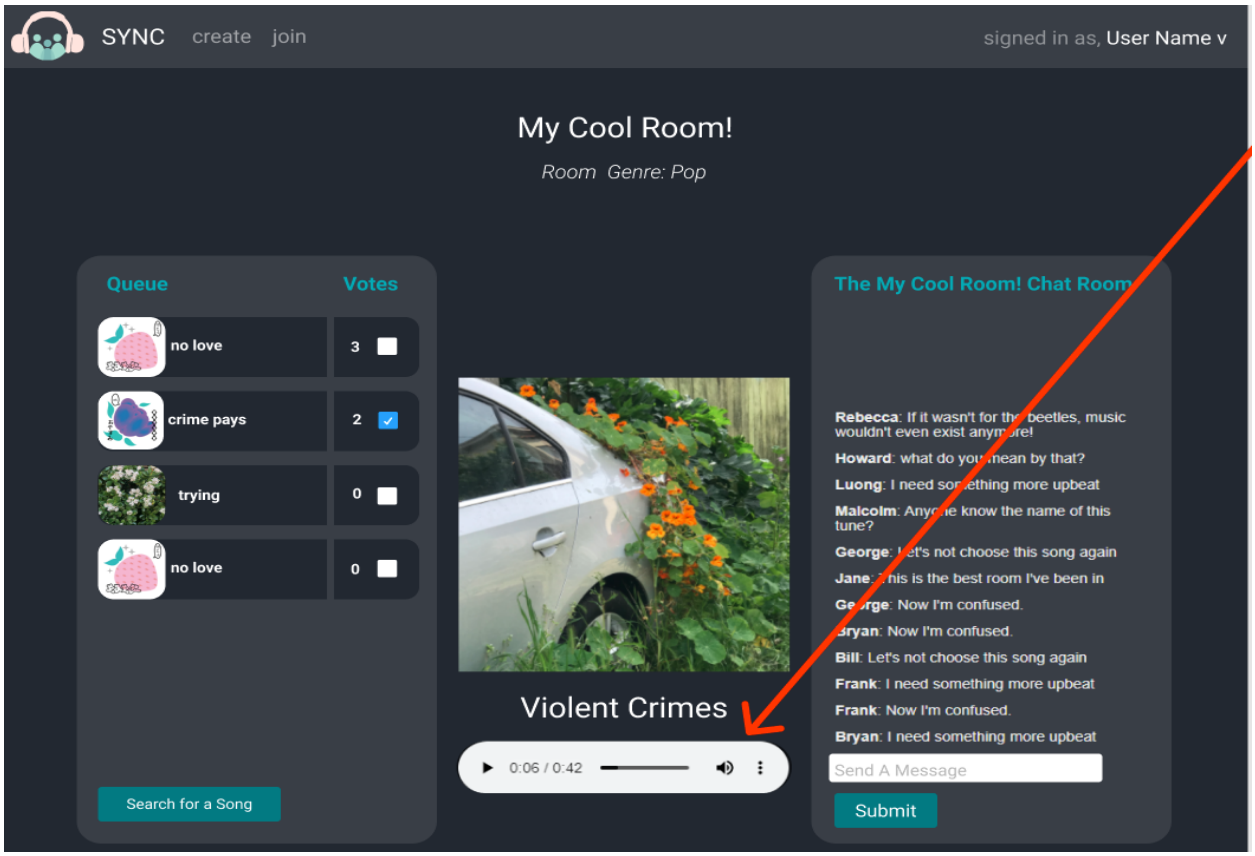


e. After selecting the checkbox, the box will appear checked and the number will change up one vote. You have voted!



8. **Change Volume in room**
   When you enter a room the volume is quite low, here is where you can change the volume to your liking!

   a. Navigate to your or a room by following step 4.
   b. Once you have entered a room, locate the player below the album cover of the currently playing song

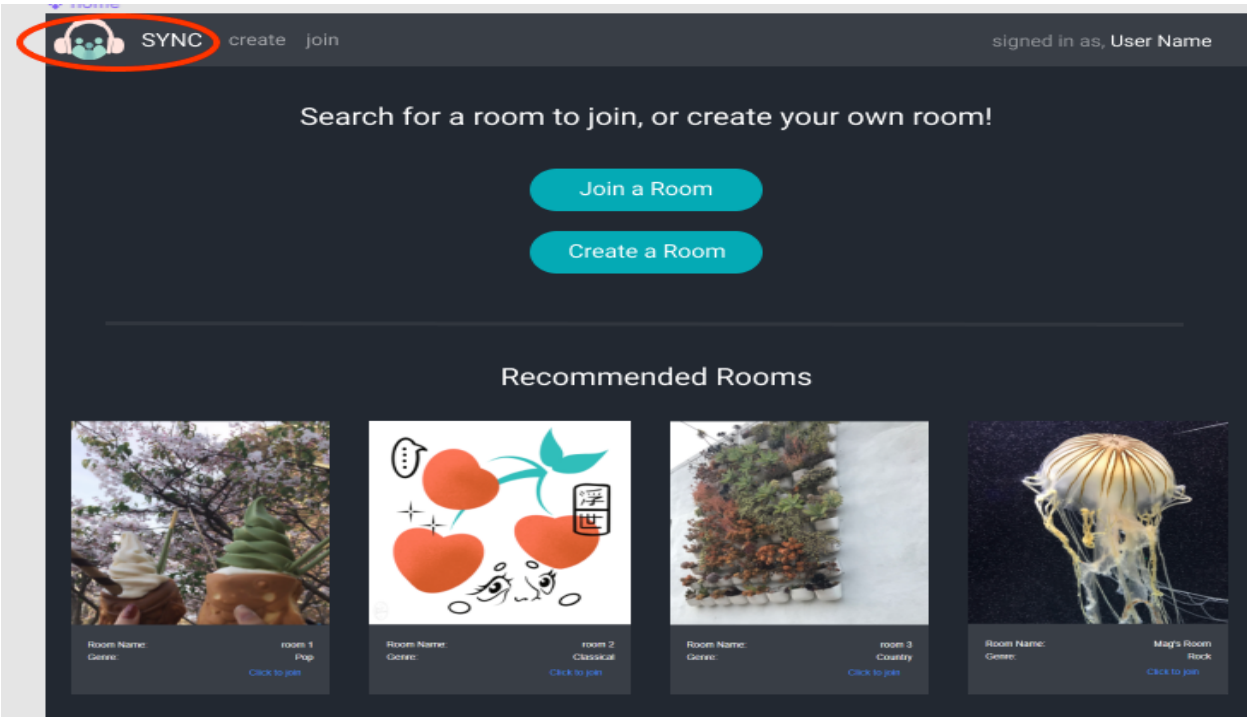


c. Locate the volume icon and select the slider button to change the volume (if you are on chrome, it will appear if you hover over the volume icon)

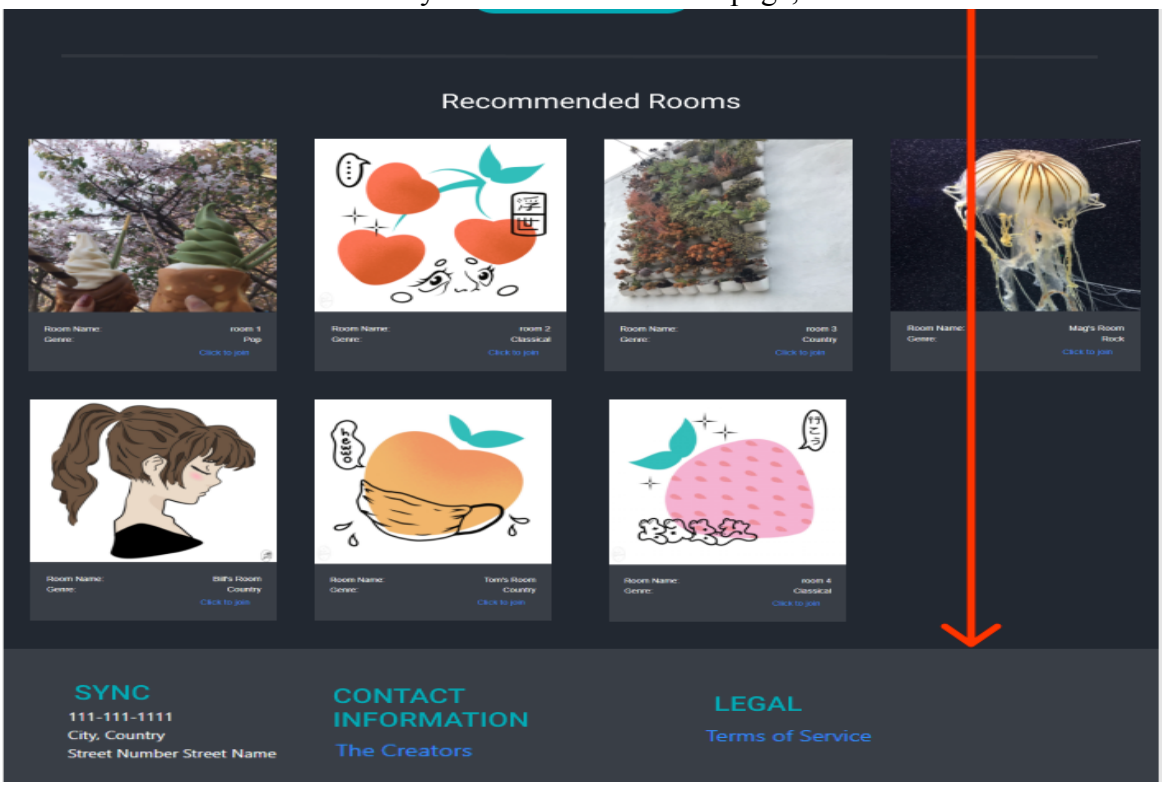
d. Slide left or right to lower the volume or bring it up.

9. **View ‘The Creators’ and ‘Terms of Service’**

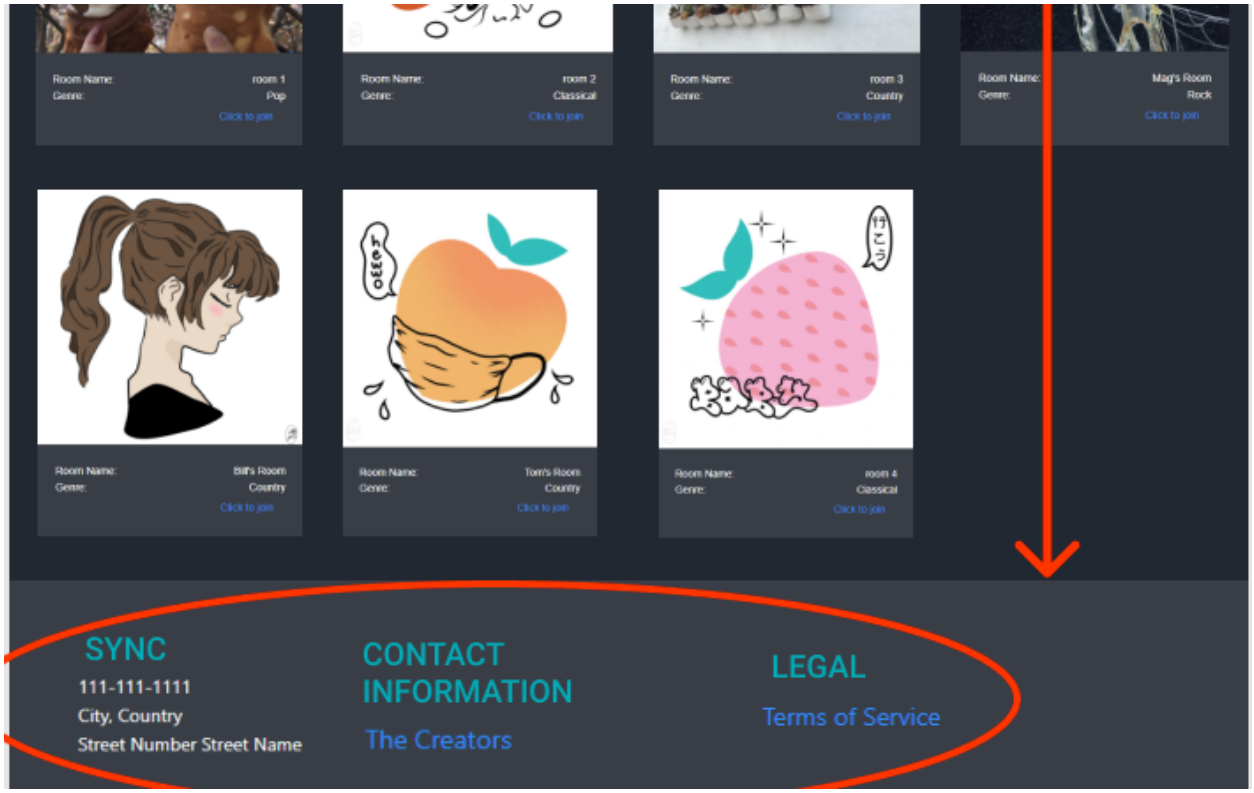
   a. Navigate to http://3.17.66.0:3000/Home or if already in SYNC, selecting the SYNC logo



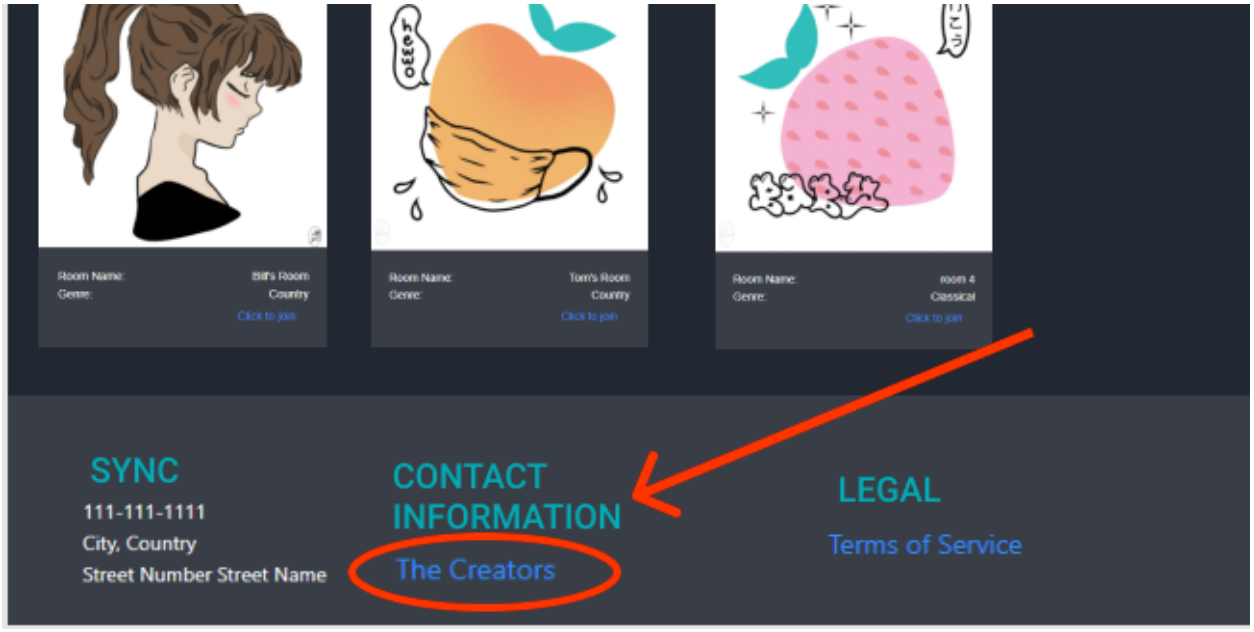
   b. Scroll down until you hit the footer of the page, it should look like this:



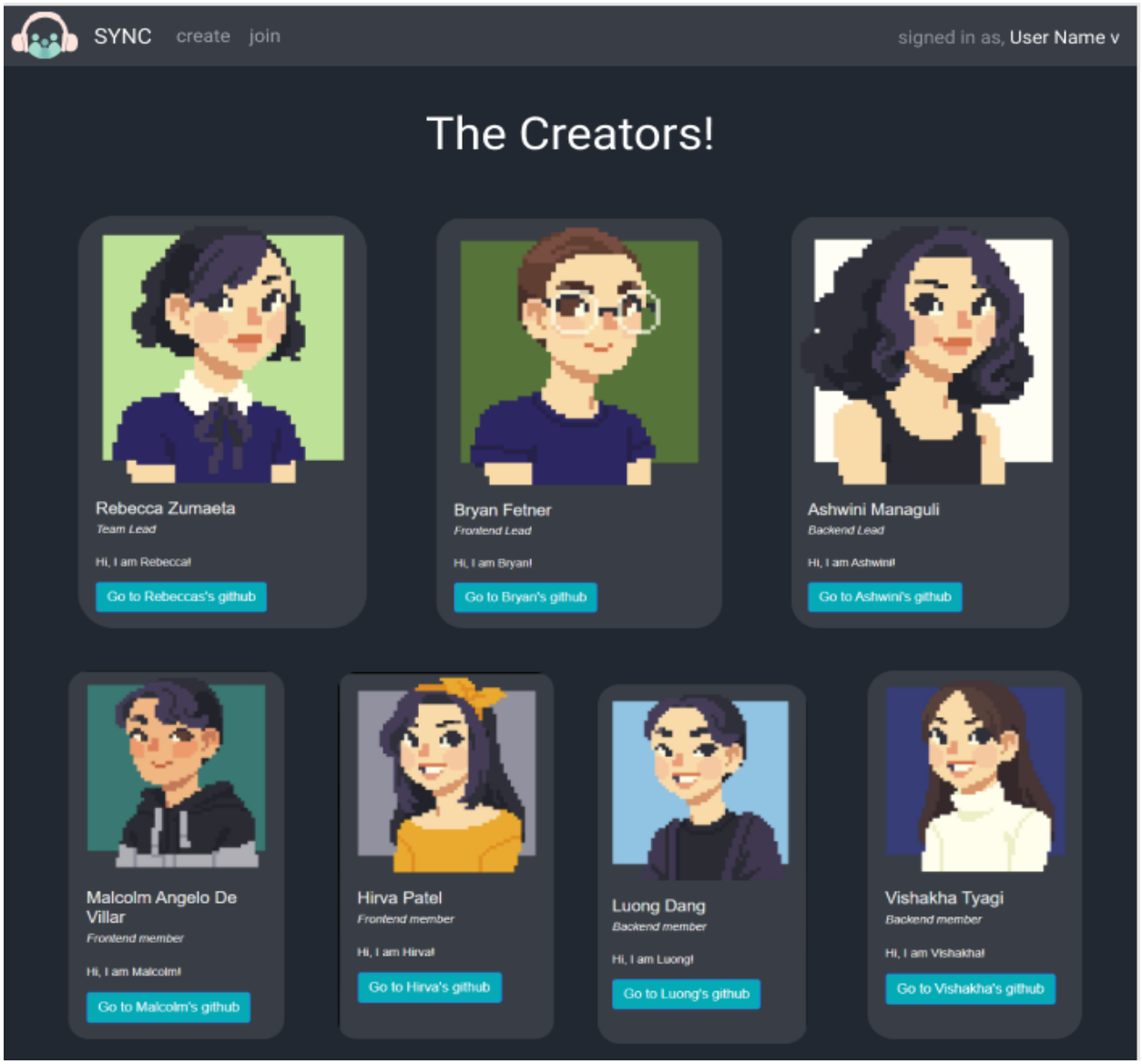
This is the footer :



c. Select ‘The Creators’ link to navigate to the creators page.

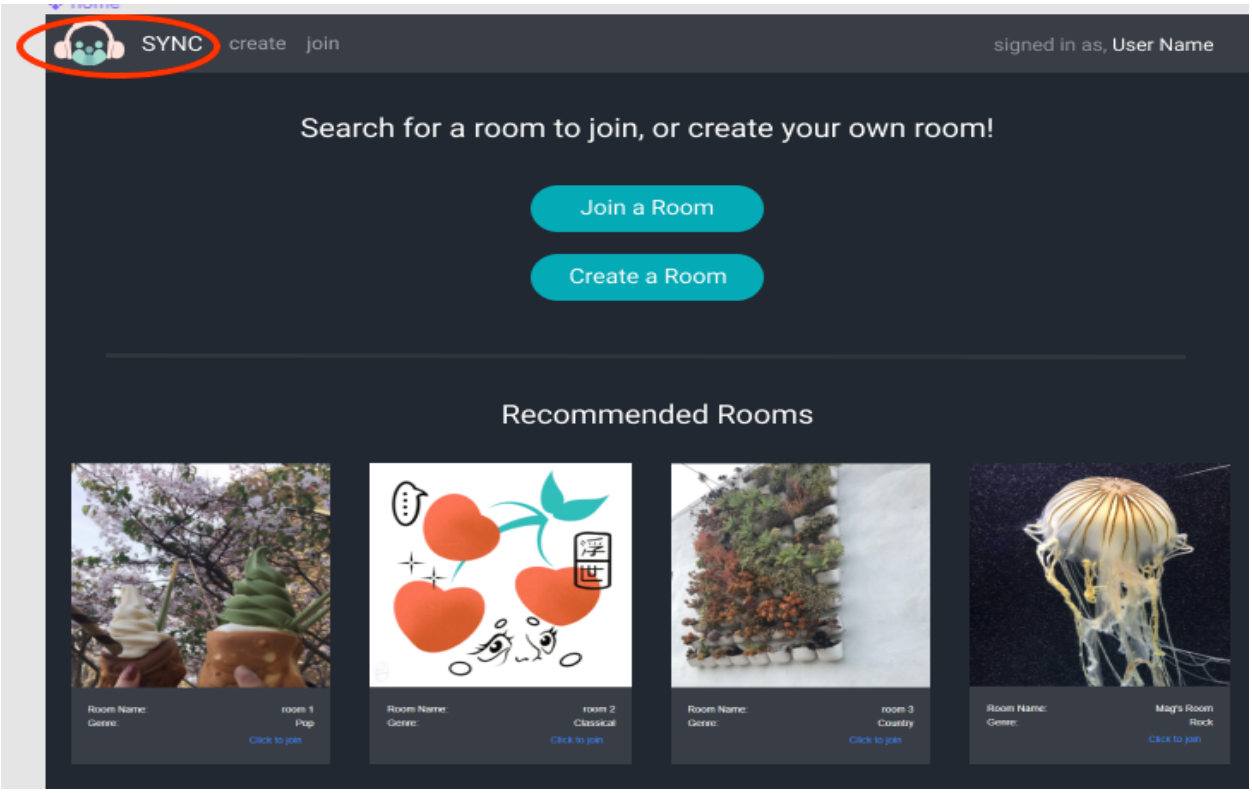


d. Once you click on ‘The Creators’ link, you are redirected to the Creators! page. It will look like this:

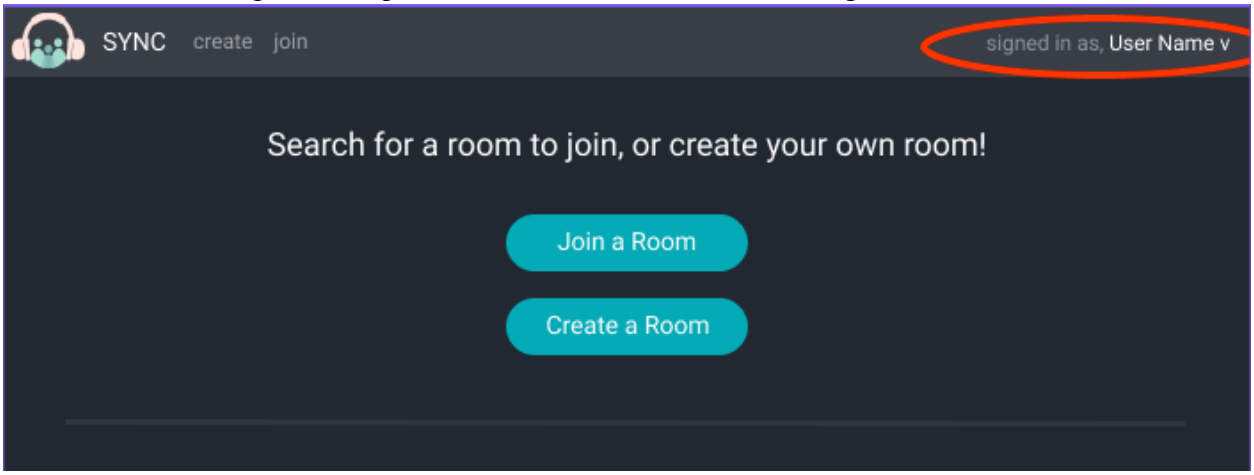


10. Log out

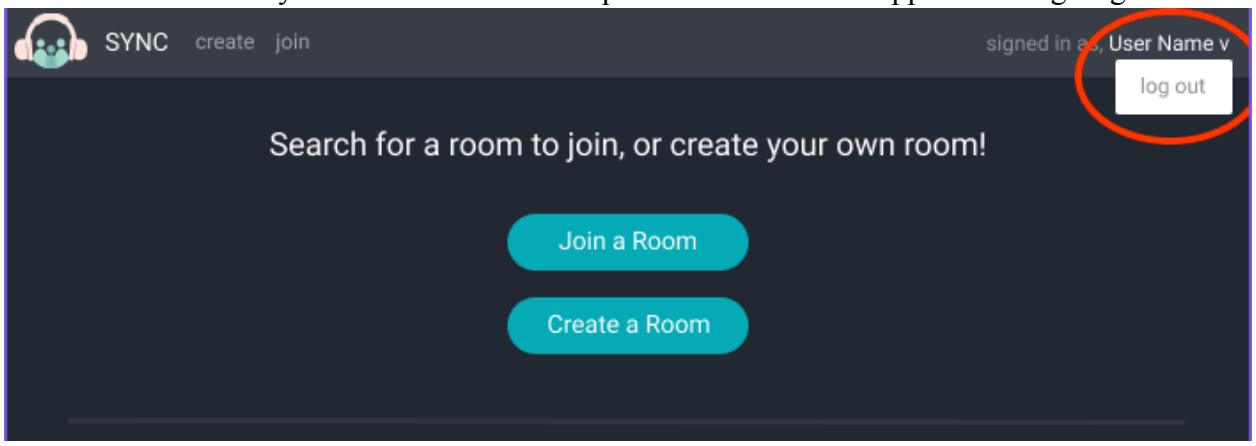
a. Make sure you are within the site, anywhere within the site. You can always navigate to the home page by selecting the sync icon on top of any page



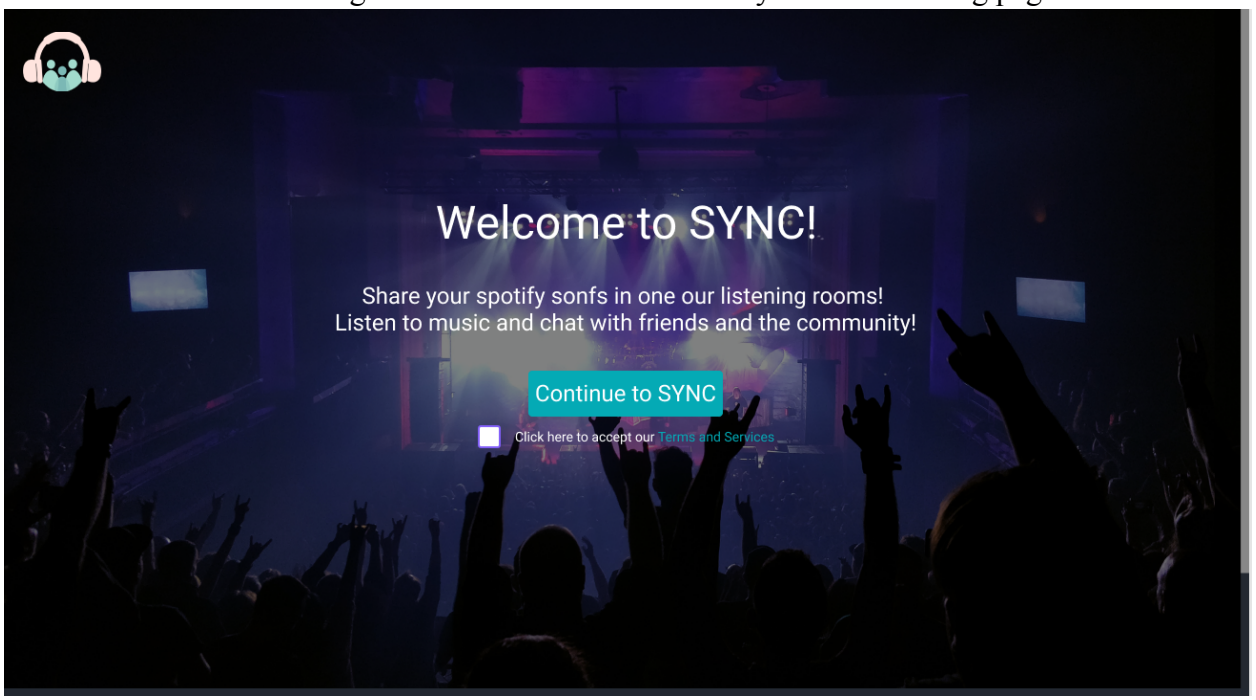
b. Go up to the top left of the screen where it reads 'signed in as *User Name*'



c. Select your username and a drop down menu should appear reading ‘log out’



d. Select the log out button and it will redirect you to the landing page!



Link to Figma, used to make wire frames:

https://www.figma.com/file/o1oD3F0Ud3COzAhv0U7R61/SYNC

4. High level database architecture and organization V2

The DBMS we chose to create our database is MySQL because it is easy to understand and some of us have experience with it; it is an organized and widely used DBMS.

1. **User (strong)**
    a. Unregistered users has one Spotify account
    b. Registered user has one and only one username
    c. Registered user has one and only one user_id
    d. Registered user has zero to one display name
    e. Registered user has one and only one profile picture
    f. Registered user shall be able to send zero to many invite links
    g. Registered user shall be able listen 0 to one song in real time in room
    h. Registered user shall be be able to be or not be a host
    i. Registered user shall be able to be or not be a participant
    j. Registered user shall have only one profile page
    k. Registered user shall have zero to one profile picture
    l. Registered user shall be zero to one host
    m. Registered user shall be zero to one participant
    n. Registered user shall be able to create zero to many public rooms
    o. Registered user shall be able to be in zero to one public room
    p. Registered user shall be able to be in zero to one private room
    q. Registered user shall be able to be in zero to one communal room
    r. Registered user shall be able invite 0 to many users to their room
    s. Registered user shall be able choose 0 to many songs to play
    t. Registered user shall be able to choose zero to many created rooms

2. **Rooms (weak)**
    a. Rooms has one and only one display name
    b. Rooms shall display one and only one hostname
    c. Rooms shall display one to many songs in queue
    d. Rooms shall display one and only one current song
    e. Rooms shall display one and only one genre
    f. Rooms shall display one and only one chat box
    g. Rooms shall display one to many voted songs
3. **Host (weak)**
    a. Host has to be in one and only one room
    b. Host shall control one and only one room.
    c. Host shall create one and only one room
    d. Host shall name one and only one room

e. Host shall play one to many songs in room

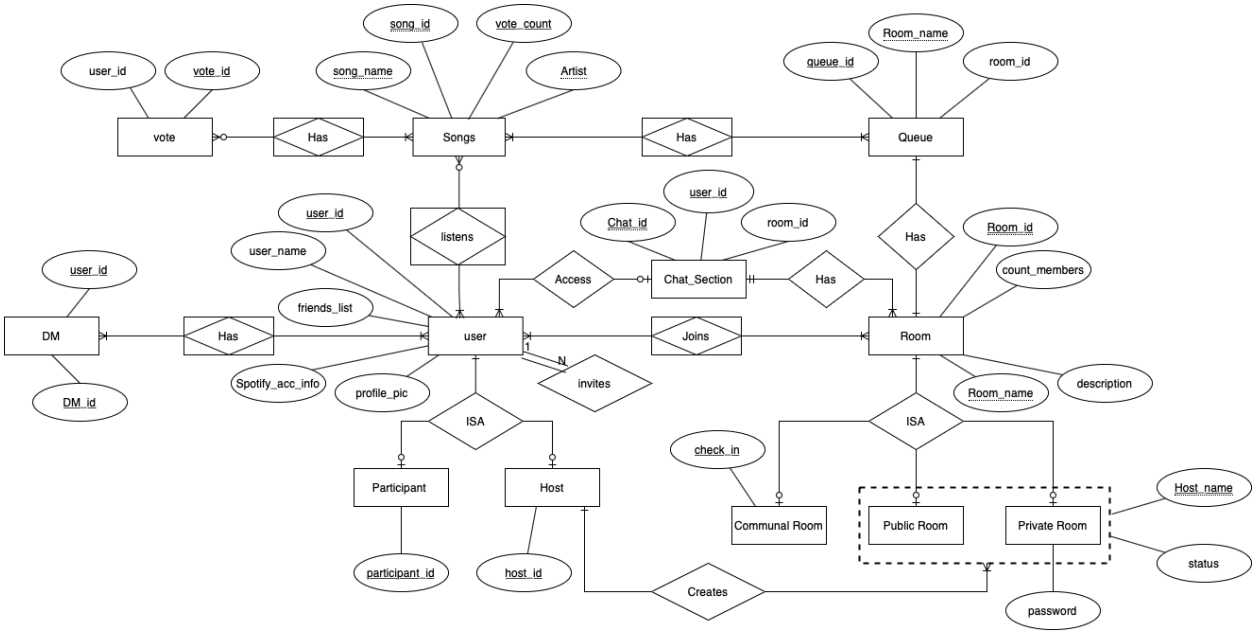
4. **Friends (weak)**
    a. Friends shall be able to be added from zero to many registered user's friend's list.
    b. Friends shall be able to be removed from zero to many registered user's friend's list.
    c. Friends shall be blocked from zero to many registered user's friend's list.
    d. Friends shall Direct Message zero to many friends
5. **Chat (weak)**
    a. Chat box shall hold chat messages from zero to many users
    b. Chat box shall exist in one to many rooms

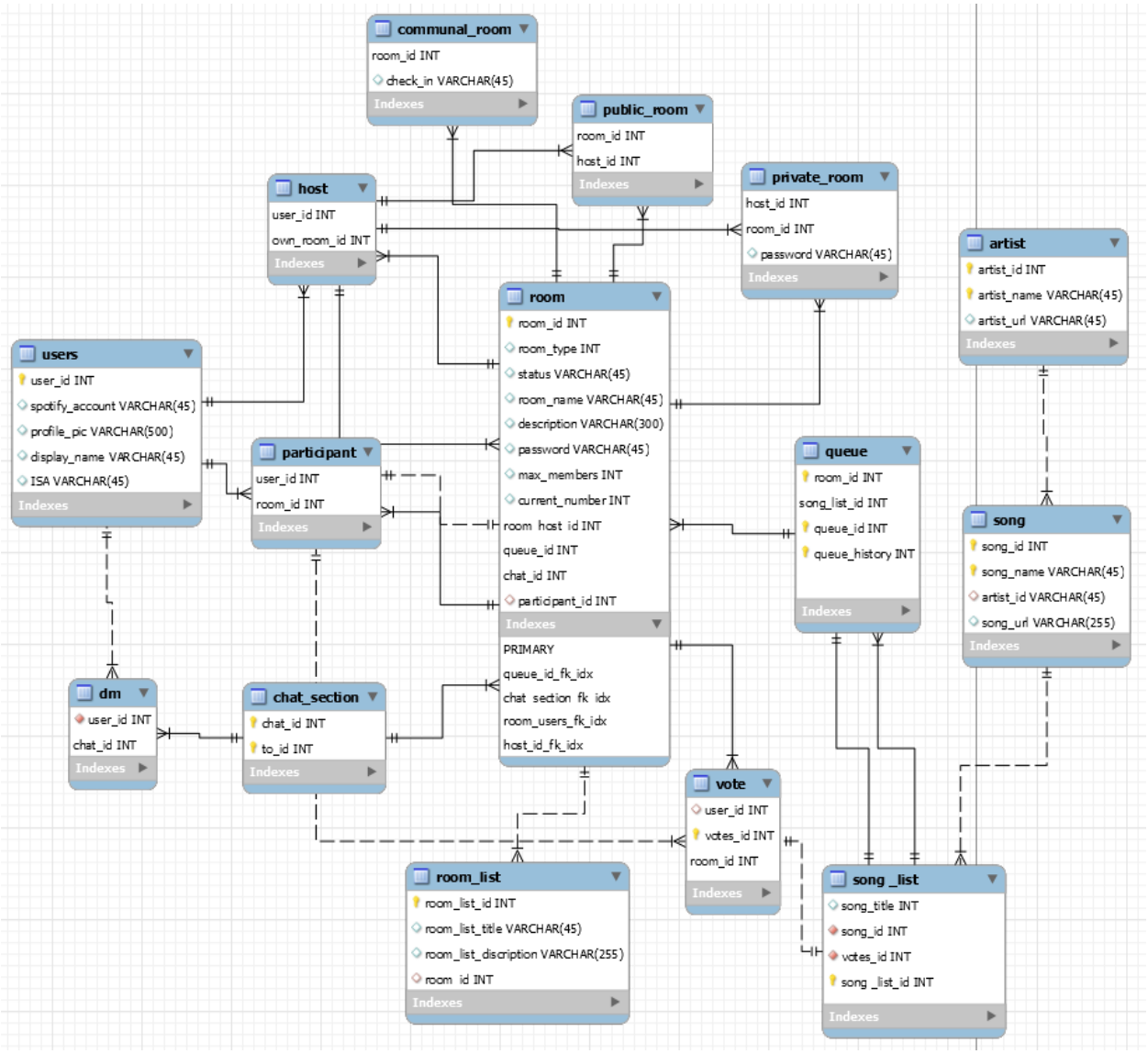
6. **Website (strong)**
    a. Website shall display zero to many Direct Messages
    b. Website shall display zero to many added friends
    c. Website shall be able to allow user to like zero to many playlists

1. **User (strong)**
    - User_id: strong key, numeric
    - spotify_id: weak key, numeric
    - profile_pic: weak
    - Display_name: alphanumeric
2. **Rooms (weak)**
    - room_type : alphanumeric
    - room_id : strong key, numeric
    - room_name : multivalue, alphanumeric
    - description : multivalue, alphanumeric
    - current_song :alphanumeric
    - room_host : alphanumeric
    - password : weak key, numeric
    - status : key, numeric
    - max_members : key, numeric
    - Current_number: key, numeric

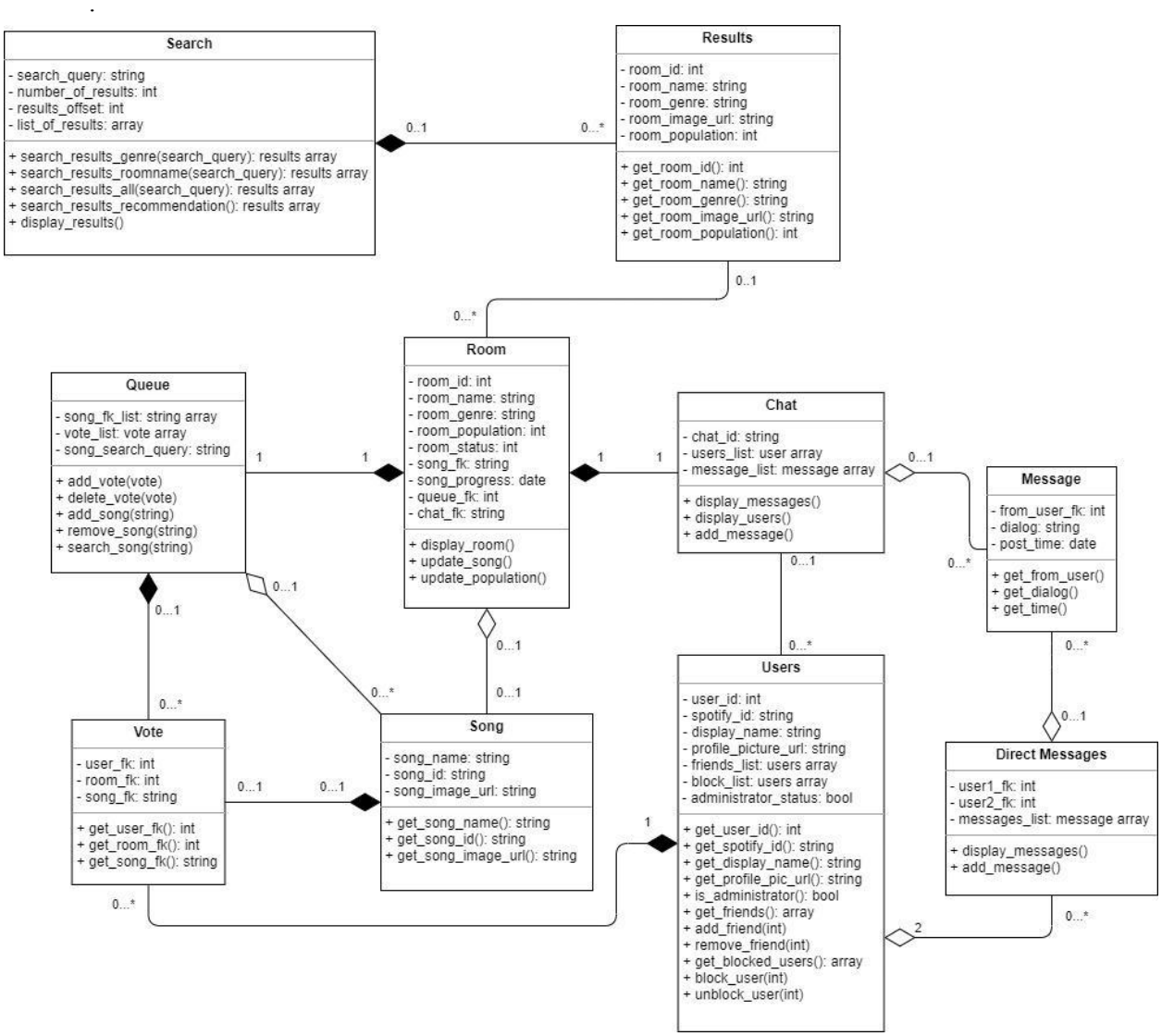
3. **Host (weak)**
    - Host_id: strong key, numeric
4. **Chat (weak)**
    - tab_id : strong key, numeric
    - tab_status : key, numeric
    - server : key, numeric
5. **Profile:** It has the information describing the registered user.
    - user_id: strong key, numeric
    - activity : key, numeric
    - profile_photo:
    - status :key, numeric
6. **Spotify_API**
    - connectivity : key, numeric
    - current_song_title : alphanumeric
    - progress : key, numeric
    - song_title : multivalue, alphanumeric
    - artist : multivalue, alphanumeric
    - image_url : multivalue, alphanumeric
    - genre : multivalue, alphanumeric
    - album : multivalue, alphanumeric
    - artist_name : multivalue, alphanumeric
    - songs : multivalue, alphanumeric
    - album : multivalue, alphanumeric

- image_url :multivalue, alphanumeric
- genre_name : multivalue, alphanumeric
- description : multivalue, alphanumeric
- top_playlist : multivalue, alphanumeric

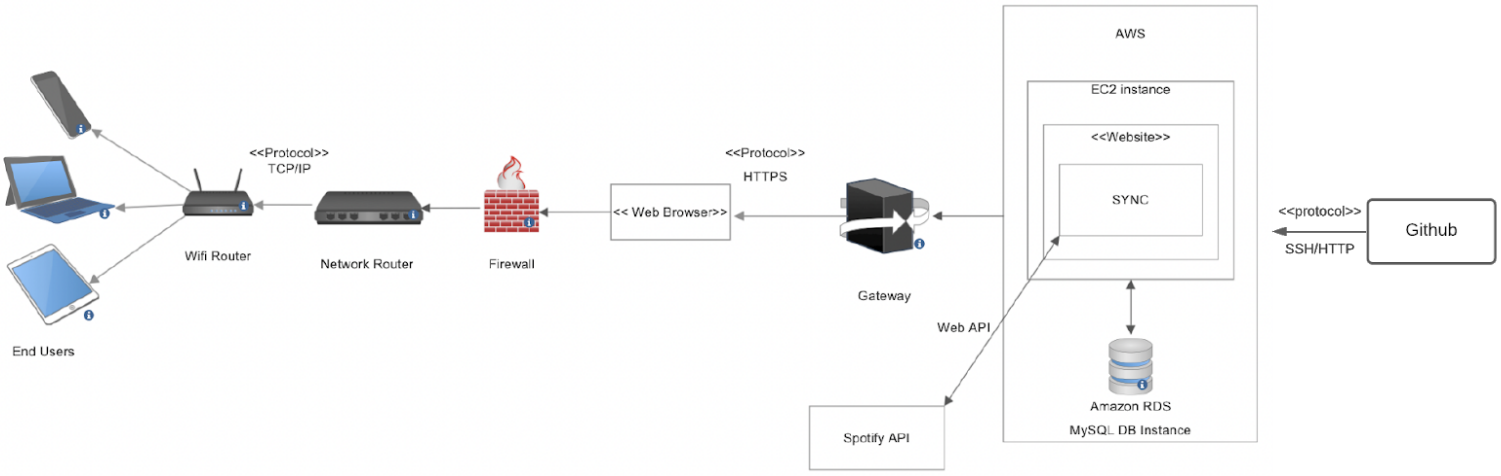



communal_room
room_id INT
check_in VARCHAR(45)
Indexes
public_room
room_id INT
host_id INT
Indexes
host
user_id INT
own_room_id INT
Indexes
private_room
host_id INT
room_id INT
password VARCHAR(45)
Indexes
artist
artist_id INT
artist_name VARCHAR(45)
artist_url VARCHAR(45)
Indexes
room
room_id INT
room_type INT
status VARCHAR(45)
room_name VARCHAR(45)
description VARCHAR(300)
password VARCHAR(45)
max_members INT
current_number INT
room host id INT
queue_id INT
chat_id INT
participant_id INT
Indexes
PRIMARY
queue_id_fk_idx
chat section fk idx
room_users_fk_idx
host_id_fk_idx
users
user_id INT
spotify_account VARCHAR(45)
profile_pic VARCHAR(500)
display_name VARCHAR(45)
ISA VARCHAR(45)
Indexes
participant
user_id INT
room_id INT
Indexes
queue
room_id INT
song_list_id INT
queue_id INT
queue_history INT
Indexes
song
song_id INT
song_name VARCHAR(45)
artist_id VARCHAR(45)
song_url VARCHAR(255)
Indexes
dm
user_id INT
chat_id INT
Indexes
chat_section
chat_id INT
to_id INT
Indexes
vote
user_id INT
votes_id INT
room_id INT
Indexes
room_list
room_list_id INT
room_list_title VARCHAR(45)
room_list_discription VARCHAR(255)
room id INT
Indexes
song _list
song_title INT
song_id INT
votes_id INT
song _list_id INT
Indexes

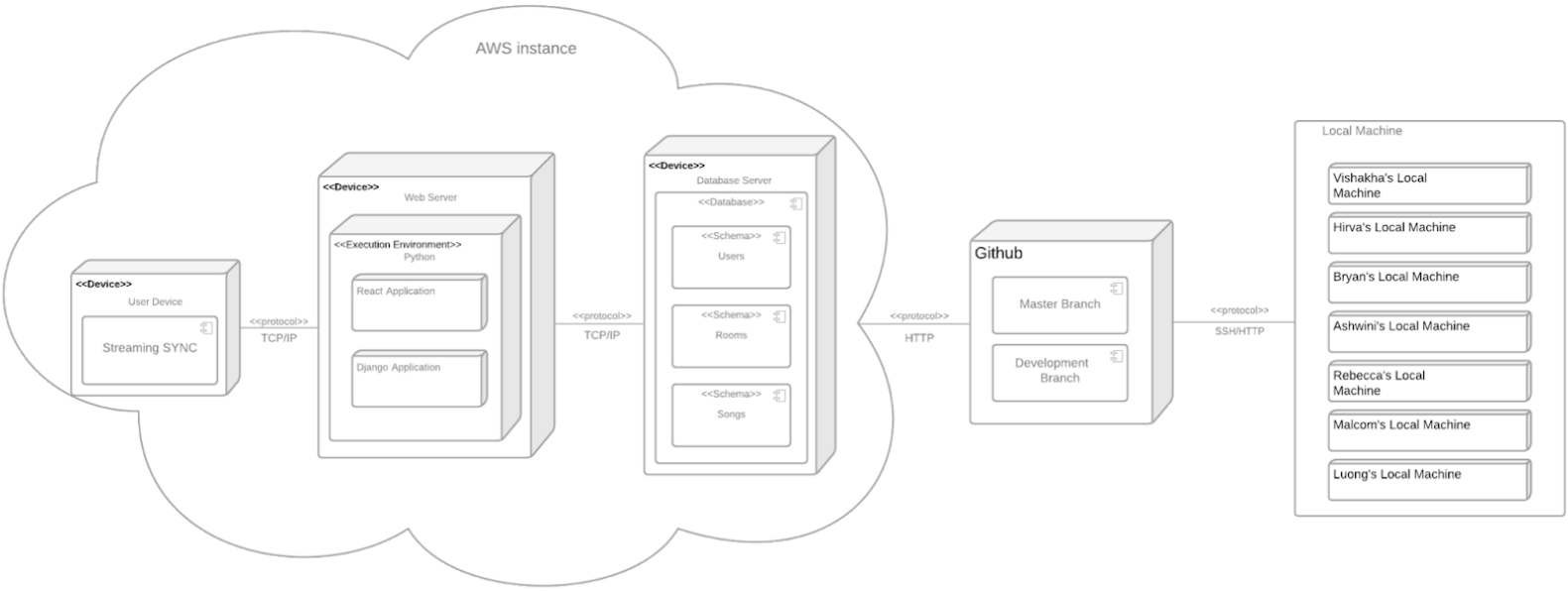
5. High Level Diagrams V2



Application Network Diagram



Deployment diagram



6. Detailed list of contributions

Team Lead, Document Master	Rebecca Zumaeta	Created Check points every 4 days and several tasks for members to fulfill to contribute to the final product, using Trello as a tool of organization. Used Figma to design and layout the WireFrames. Wrote out the directions of the wire frames. Managed the document entirely. Moved around P1's, P2's and P3's with the team. Tested horizontal prototype through the eyes of the user. Provided material for the horizontal such as personal images, artwork and free non copyrighted music. Only pushed documents onto the repository.
Front End Lead, mediator	Bryan Fetner	Built out a good amount of the functionality of the site: the pages connecting back to each other, music playing on entrance to a room, automated chat box, adding songs to queue and voting.. This includes .jsx files to connect it back to the database when needed. Made sure functionalities of the site worked in the front end but are compatible for back to connect back.Alongside Malcolm, designed the UX such as the nav bar, button placement and footer. Committed and pushed code to repo quite a few times to the front end of the project.
Back End Lead, Document contributor, Site Deployer	Ashwini Managuli	Deployed site when a push to the development branch had major updates. Connected back end sql to any updated to front end made (search bar for example). Dealt with errors of deployment and updated ram and other components with Ec2 when needed, and approved by Team Lead. Worked on leading back end members to other projects geared towards being implemented on M4 in local machines (one is updated on the testing branch).

Front End Member Document contributor	Malcolm Angelo De Villar	Worked on UI/UX of site: Designed navbar and footer and implemented links on them, Designed landing page with FAQs and TOS, assisted in designing room layout and pages create and join. Assisted Bryan in all button creations and functionality. Worked in mostly implementing CSS and have it be versatile for most functionality of the site. Implemented features that members deemed necessary to change for a better user experience such as when Team lead suggested fixing button sizes or text.
Front End Member	Hirva Patel	Worked heavily with the display of the rooms as shown as ‘recommended rooms’ in the grid that it had on pages ‘create’ and ‘join.’ Also designed with assistance with the back end of the search bar and it’s functionality of the drop down menu. Added images to rooms. Implemented features Front end Lead asked of her such as log out button and links to rooms when searching.Worked with Ashwini to connect back and front together though AWS.
Github Master and back end member Document contributor	Vishakha Tyagi	Assisted with documentation in fulfilling section 5. Built some functionalities for M5 such as login with Spotify with authentication on local machine. Helped in testing bugs in UI for front end to fix.
Back End Member	Luong Dang	Assisted in document in section 1. Built functions and api’s geared for implementation for M5 such as the spotify api integration (getting user info such as their liked songs and playlists), shown in the testing branch. Assisted in integrating pictures for Albums in recommended songs and in the room themselves.

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

05/15/2021

History Table

Version	Date	Notes
M4V2	05/15/2021	
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..... 3
2. Usability Test Plan..... 6
3. QA Test Plan…..14
4. Code Review…….....17
5. Self-Check on Best practices for security..…….....24
6. Self-Check: Adherence to Original Non-Functional Specs……...25
7. List of contributions 31

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 output 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'.

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

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

3	TASK	Description
	Task	Search for a song in a room
	Machine state	In a room with the song search component open.

	Successful completion	Song results based on song search query.
	Benchmark	5 seconds

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

5	TASK	Description
	Task	Chat in a room
	Machine state	In a room
	Successful completion	Inputted test will appear in the shared chat log.
	Benchmark	10 seconds

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

7	TASK	Description
	Task	Signing into Spotify

	Machine state	Landing page
	Successful 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**
1	Create a room	100%	none	Room creation complete
2	Search for a room	100%	none	Search for a room is complete
3	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.
4	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.
5	Chat in a room	80%	Some text will not appear in chat history	Does not take into special characters or emoticons
6	Share link to a room	100%	none	Copying on link within a room is implemented to share on other platforms
7	Signing into Spotify	70%	Some users are not allowed to proceed	Only premium spotify users are allowed to proceed to our site

Questionnaire

Test #	Question	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
1	There is not enough options to customize the room	1	2	3	4	5
1	Creating a room was a straightforward process.	1	2	3	4	5
1	I understood what creating a room is during this process.	1	2	3	4	5
2	The ‘Search for a Room’ function’ is placed in the wrong page	1	2	3	4	5
2	The searching process was easy.	1	2	3	4	5
2	The search results cater to your interests.	1	2	3	4	5
3	Finding the search function was fairly easy.	1	2	3	4	5
3	The room was easy to navigate.	1	2	3	4	5
3	They did not have the songs I wanted to play.	1	2	3	4	5
4	Voting allows me to have input in the next song played.	1	2	3	4	5
4	I was not satisfied with how the voting is done on songs.	1	2	3	4	5
4	The display of votes is user friendly	1	2	3	4	5
5	I knew how to use the chat immediately after seeing the chat.	1	2	3	4	5
5	I was confused when interacting in the rooms	1	2	3	4	5

5	The chat room was useless and unnecessary	1	2	3	4	5
6	It was confusing on how to copy the link of the room	1	2	3	4	5
6	I’m happy with the fact that I can share the room via a link	1	2	3	4	5
6	I had a hard time finding where in the room to get the room link	1	2	3	4	5
7	The sign in process took too long.	1	2	3	4	5
7	I understood immediately that I needed to Sign into my Spotify account	1	2	3	4	5
7	I would recommend this site to Spotify users	1	2	3	4	5

Results

Test #	Question	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
1	There is not enough options to customize the room	2	1	0	0	0
1	Creating a room was a straightforward process.	0	0	0	0	3
1	I understood what creating a room is during this process.	0	0	1	0	2
2	The ‘Search for a Room’ function’ is placed in the wrong page	0	0	0	0	3
2	The searching process was easy.	0	0	0	0	3
2	The search results cater to your interests.	0	0	0	0	3
3	Finding the ‘Search for a Song’ function was fairly easy.	0	0	0	2	1
3	The room was easy to navigate.	0	0	0	0	3
3	They did not have the songs I wanted to play.	0	0	0	1	2
4	Voting allows me to have input in the next song played.	0	0	0	0	3
4	I was not satisfied with how the voting function was executed.	2	1	0	0	0
4	The display of votes is user friendly	0	0	0	0	3
5	I knew how to use the chat immediately after seeing the chat.	0	0	0	0	3

5	I was confused when interacting in the rooms	2	1	0	0	0
5	The chat room was useless and unnecessary	0	3	0	0	0
6	It was confusing on how to copy the link of the room	1	0	2	0	0
6	I’m happy with the fact that I can share the room via a link	0	0	0	0	3
6	I had a hard time finding where in the room to get the room link	3	0	0	0	0
7	The sign in process took too long.	3	0	0	0	0
7	I understood immediately that I needed to sign into my Spotify account	0	0	0	3	0
7	I would recommend this site to Spotify users	0	0	0	0	3

3. QA Test Plan

Test Objectives

Our objective with these tests is to verify that information is being handled correctly throughout the site. Due to the fact that most of the site's focus is on organizing "listening rooms" and using data from Spotify as well as sharing that to various users, 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 its 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 the search query.
3. Getting song track information from Spotify API:
   This tests the retrieval of specific song information needed for app functionality from 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 join 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

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 through SYNC will give users Spotify Authentication.	User Spotify Premium account credentials	Users receive 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 general 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 a database.	Test acquisition of data into 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 interfaces 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 a 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 understandable and is efficient

```
 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 menus throughout the site,unless this is a necessary 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, technically 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, it really shows understanding of functions. using 'reassigning variable this to accomplish 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 console printing for errors at critical points for easy and quick 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 truly 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 times in between blocks or functions, but other than a couple naming conventions, it's clear, it's concise, it's 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 a roomname or genre that doesn't exist, wrong input then it shows a 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 - **DONE**
12. We use API to dictate recommended rooms - **DONE**
13. We use an API to group people together. - **DONE**
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 -**DONE**
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. - **ISSUE -** Spotify API in not compatible
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. - **ISSUE -** Spotify API in not compatible
25. The website shall work up to Version 14.0 of Safari. - **DONE**
26. The website shall support Spotify music streaming. - **DONE**

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. - **DONE**

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 - **DONE**
42. The website shall be capable to communicate efficiently with the users - **DONE**
43. The website shall be capable to recover from failures - **DONE**

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 - **DONE**
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 - **DONE**
62. The code shall have proper formatting - **DONE**
63. The code shall have indentation - **DONE**

Availability

64. The website shall be active even if no users are active on the website **-DONE**
65. The website shall resync when loss of connection occurs - **DONE**
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. **DONE**
81. Store passwords on the database. - **ISSUE** - Spotify API handles this function
82. Store users ignore lists 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 - **DONE**
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 the documentation, conducted Usability and QA tests, coordinated code review with members outside of the 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 Spotify login, song search with spotify API, etc. Assisted in confirming 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, assisted in building out all tables within DB diagram. Assisted in confirming 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 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 committed functions and to original non-functional specs.
Github Master and back end member Document contributor	Vishakha Tyagi	Assisted in building out all tables within the DB diagram like table for voting, assisted with document in section 4 and 3. Assisted in confirming 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 committed functions and to original non-functional specs.