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

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:

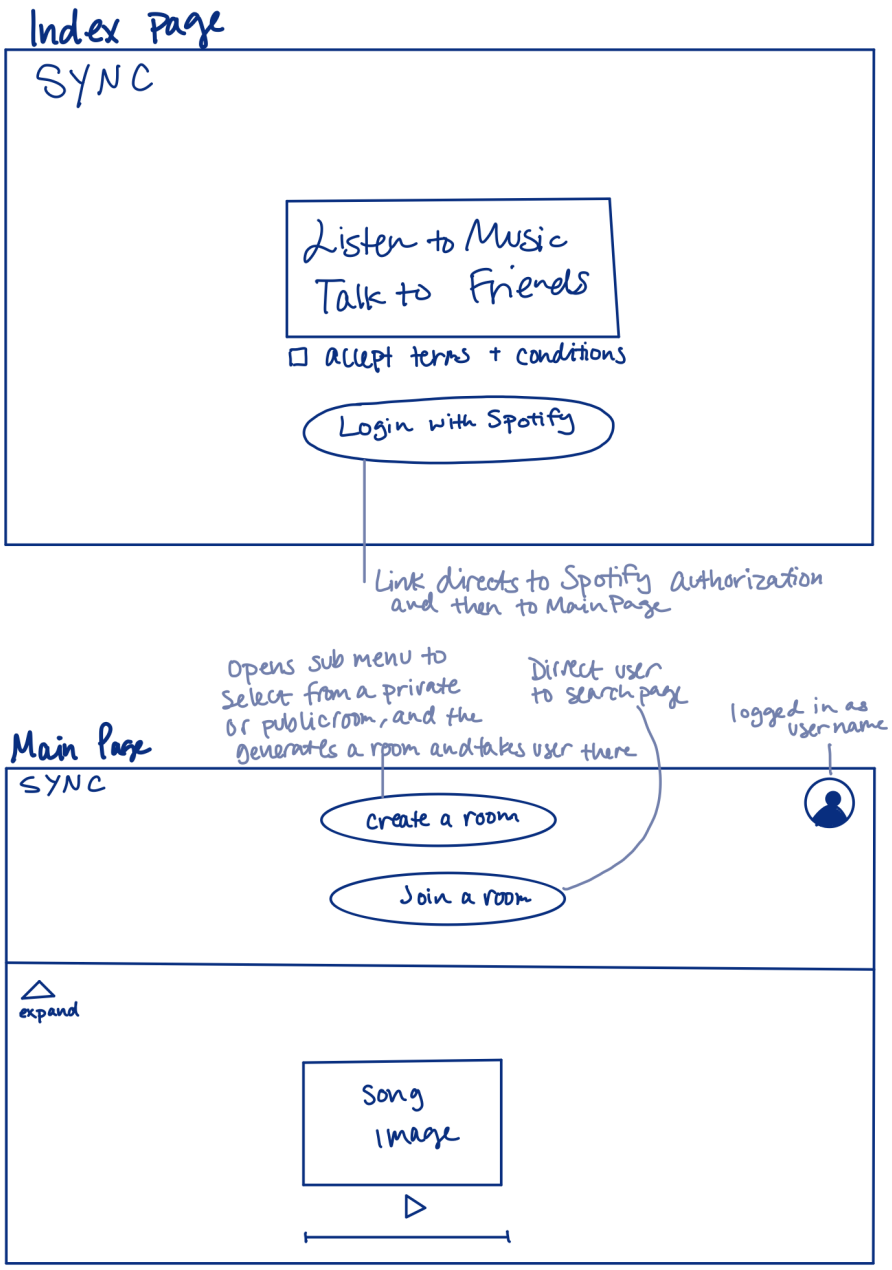
2. Registered users

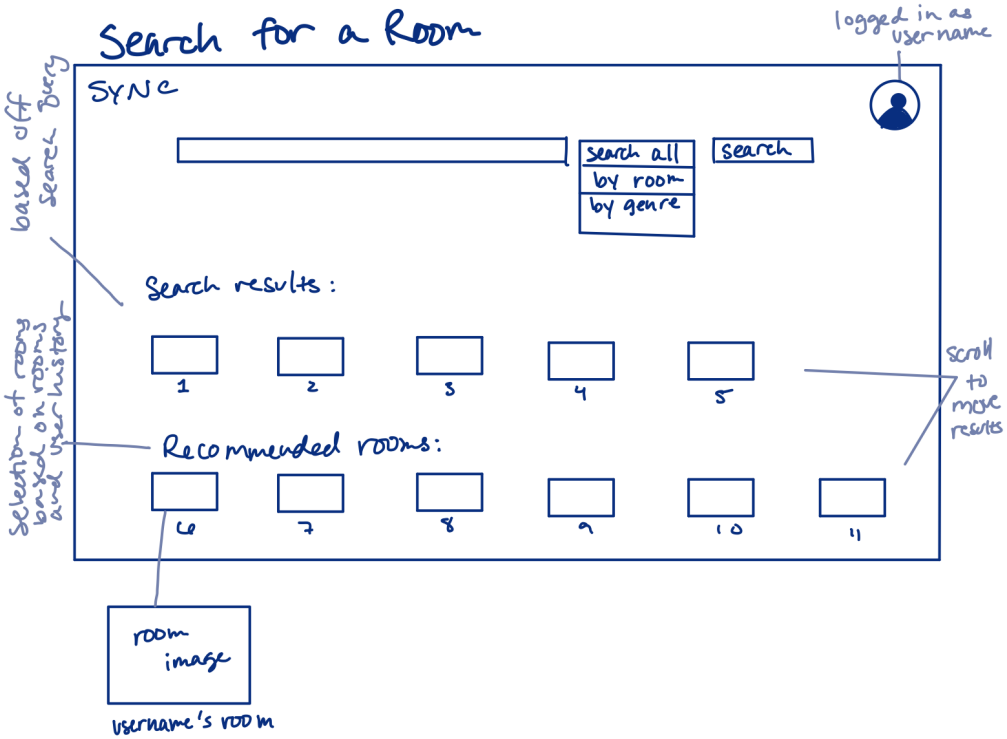
2.15. Registered Users shall be able to export the playlist of the room.
2.16. Registered Users shall be able to edit their SYNC profile.
2.17. Registered Users shall be able to change their SYNC usernames.
2.18. Registered Users shall be able to change their SYNC profile picture.
2.19. Registered Users 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

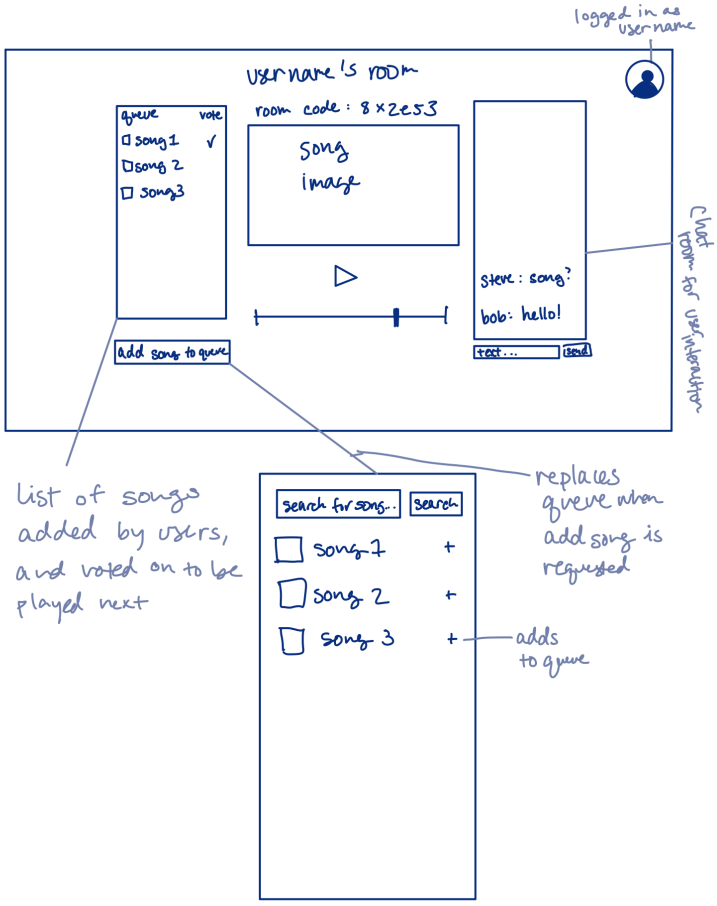
5. Website

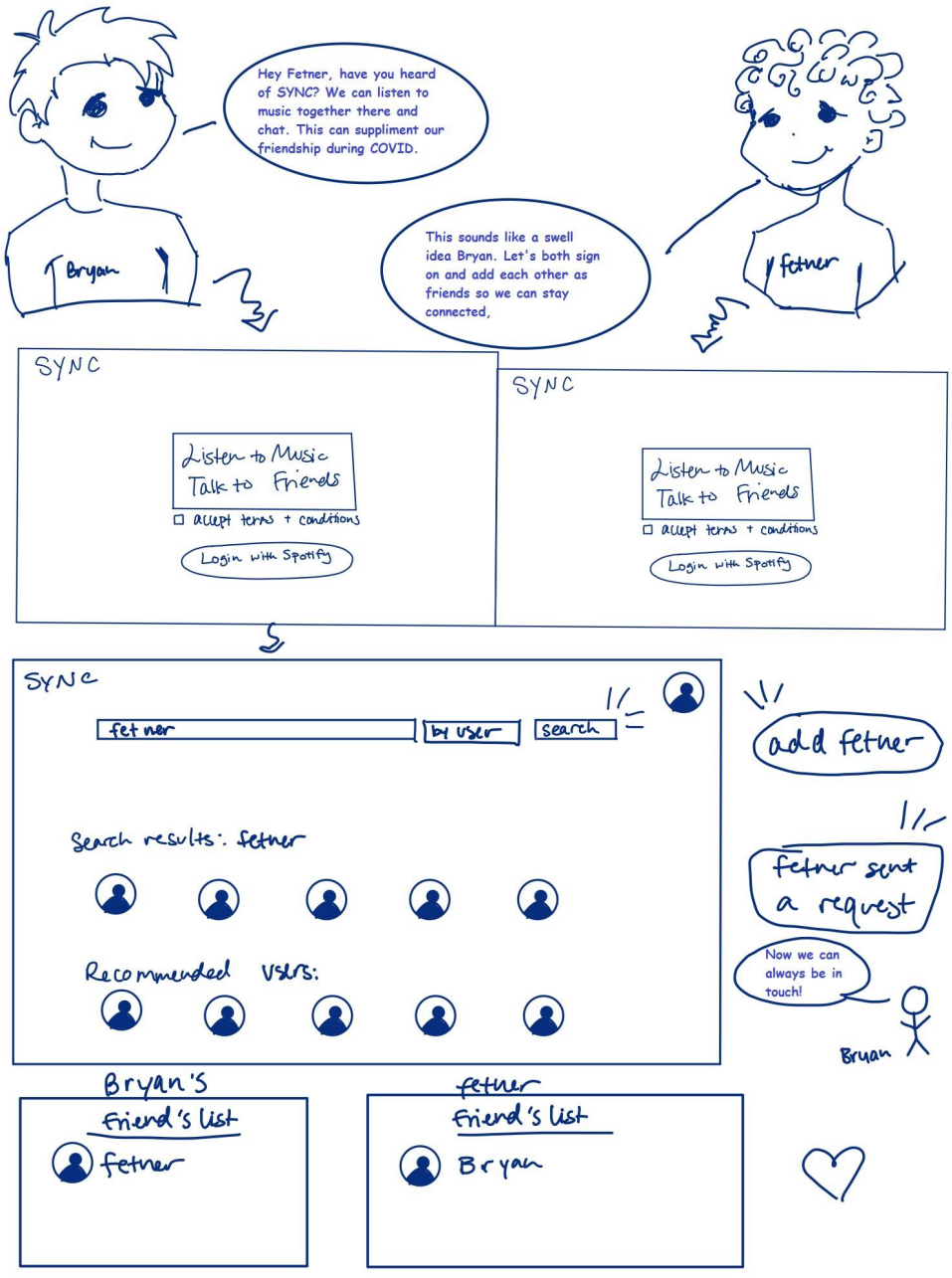
5.92. Website shall send notifications
5.104. Website shall have premiers of podcast
5.105. Website shall have premiers of song drops
5.106. Website shall have premiers on album drops

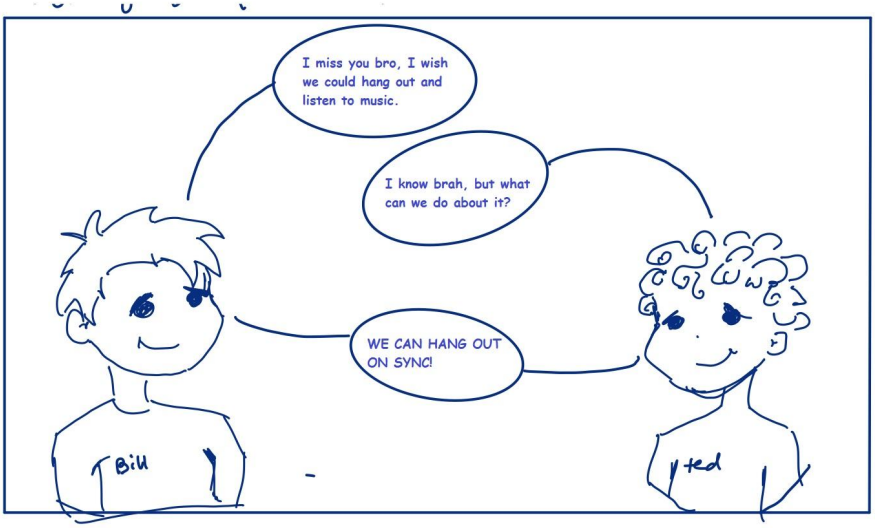
3. UI Mockups and Storyboards

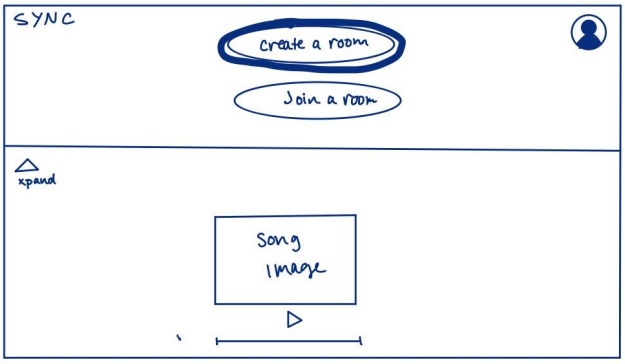


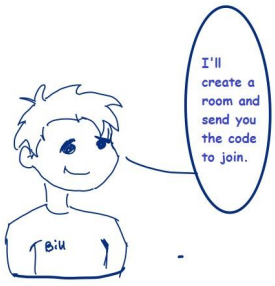

Search for a Room
logged in as username
SYNC
search all
by room
by genre
search
based off search query
Search results:
1
2
3
4
5
scroll to more results
selection of rooms based on rooms and user history
Recommended rooms:
6
7
8
9
10
11
room image
username's room


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

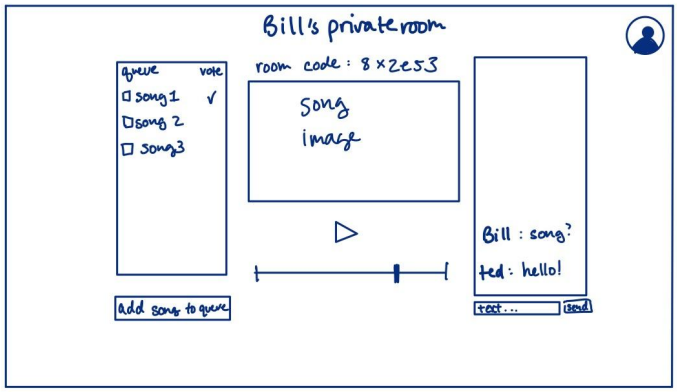

Hey Fetner, have you heard of SYNC? We can listen to music together there and chat. This can suppliment our friendship during COVID.
Bryan
This sounds like a swell idea Bryan. Let's both sign on and add each other as friends so we can stay connected,
fetner
SYNC
Listen to Music
Talk to Friends
accept 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

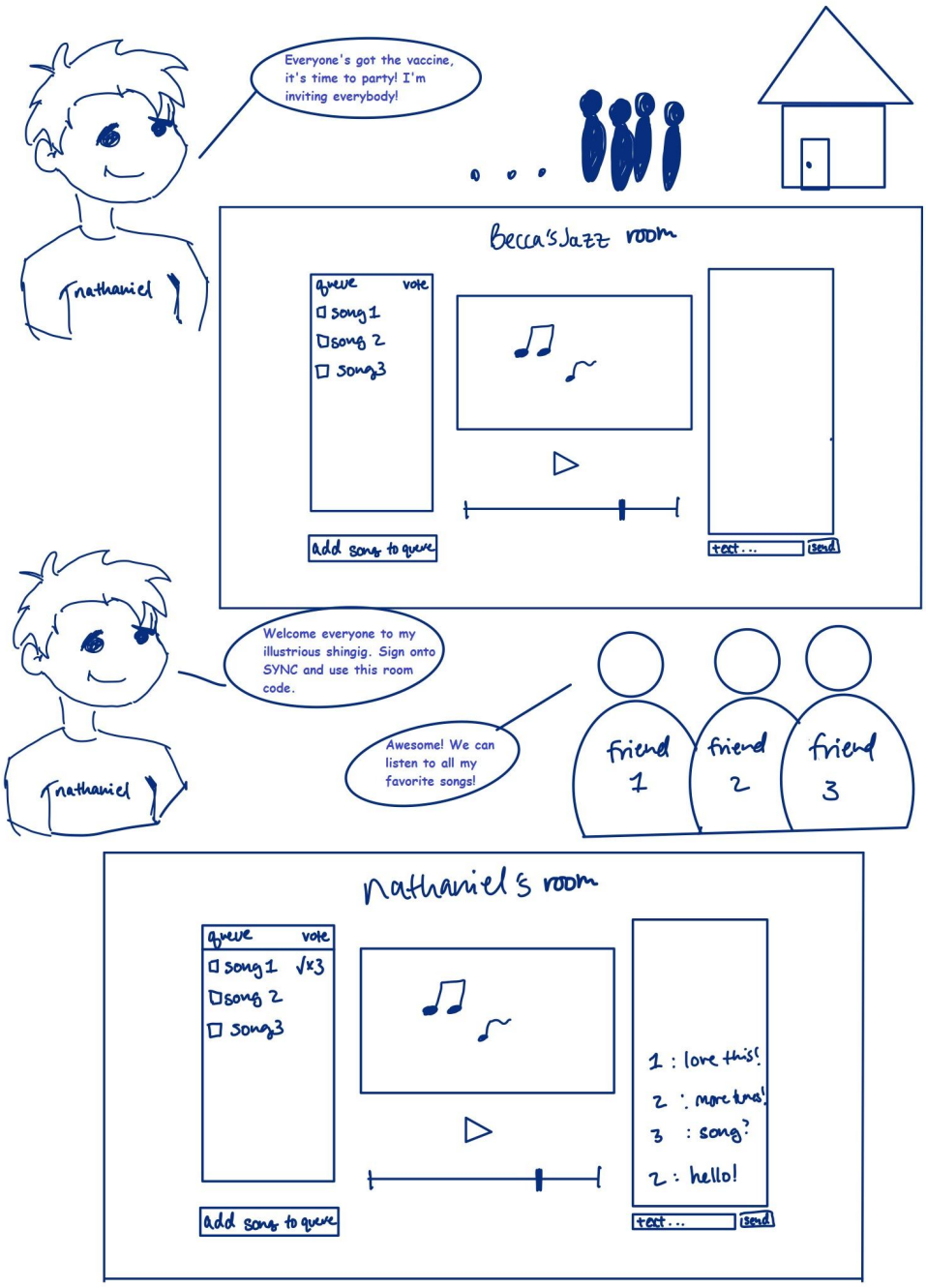

I miss you bro, I wish we could hang out and listen to music.
I know brah, but what can we do about it?
WE CAN HANG OUT ON SYNC!
Bill
ted

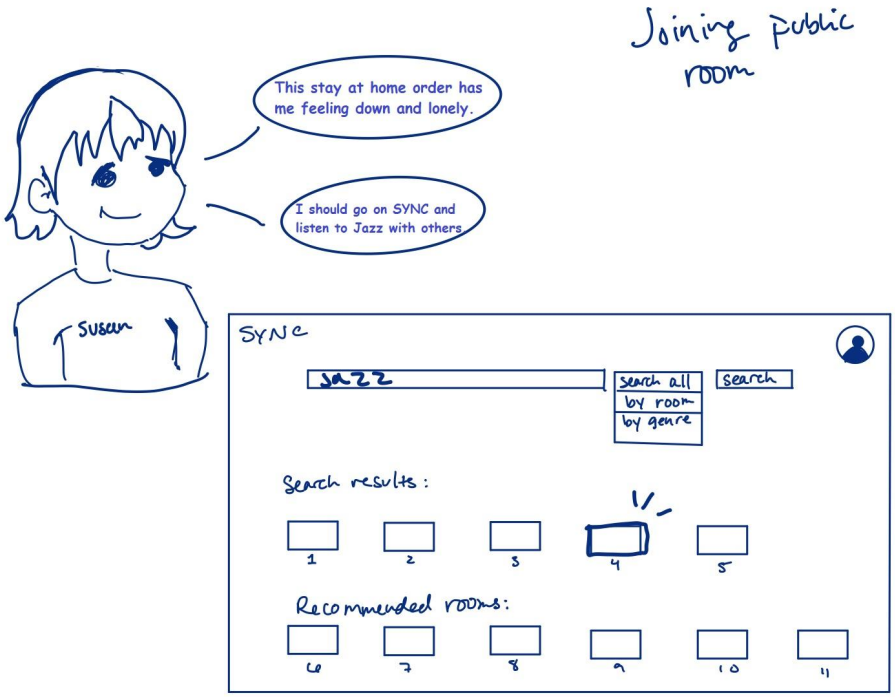

SYNC
create a room
Join a room
xpand
Song image

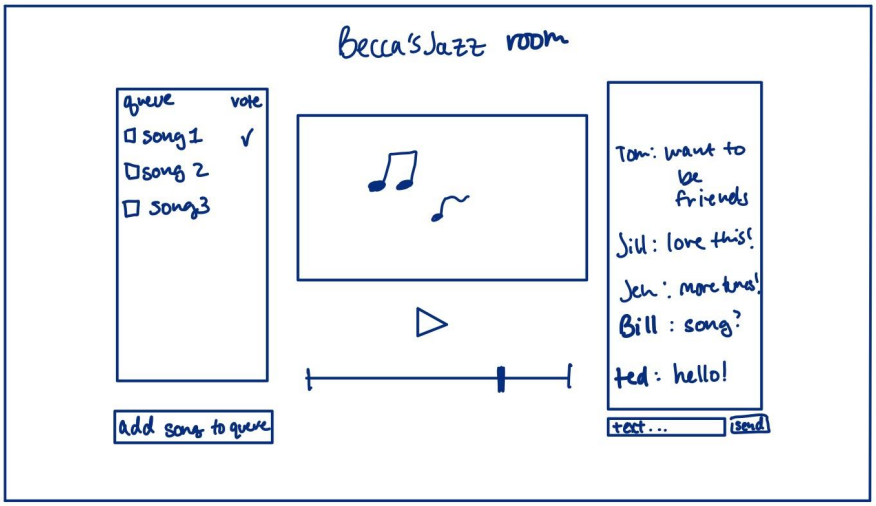

I'll create a room and send you the code to join.
Bill

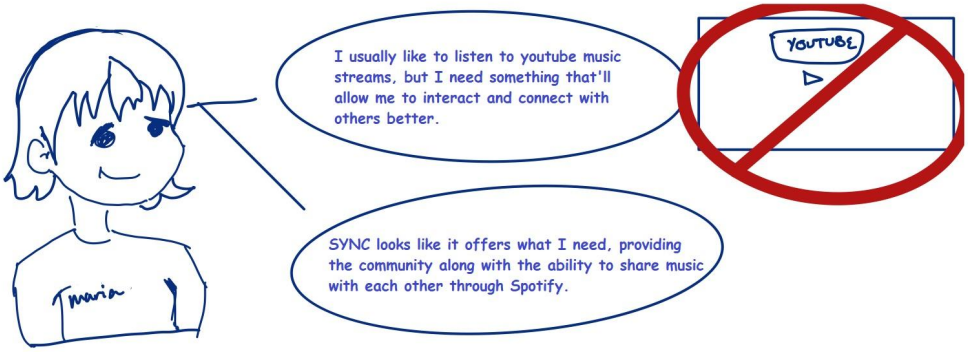

Bill: ~~~
Sweet, now we can hang!
ted

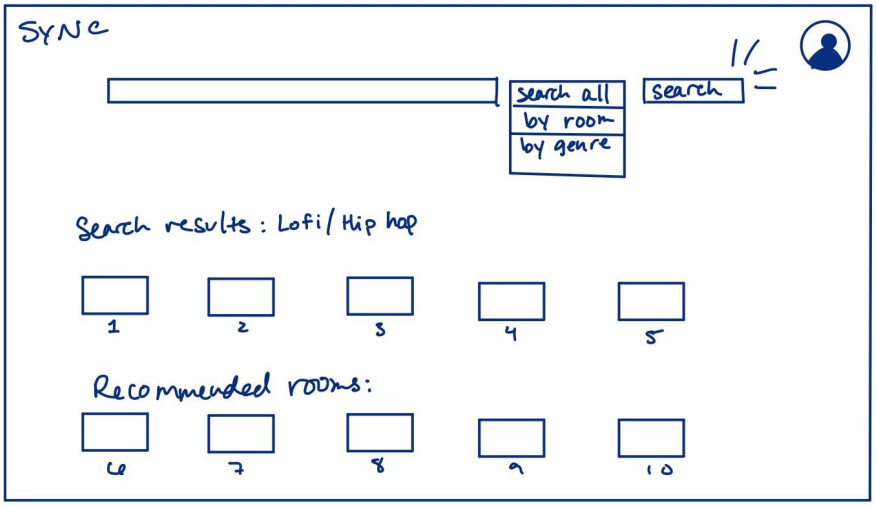

Bill's private room
queue
vote
song 1
song 2
song 3
room code: 8x2e53
Song image
Bill: song?
ted: hello!
text...
send
Add song to queue

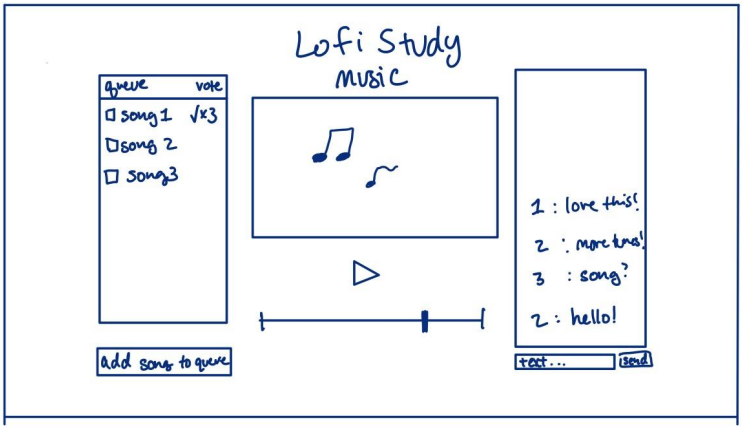

Everyone's got the vaccine, it's time to party! I'm inviting everybody!
nathaniel
Becca's Jazz room
queue
vote
song 1
song 2
song 3
Add song to queue
text...
send
Welcome everyone to my illustrious shingig. Sign onto SYNC and use this room code.
Awesome! We can listen to all my favorite songs!
friend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

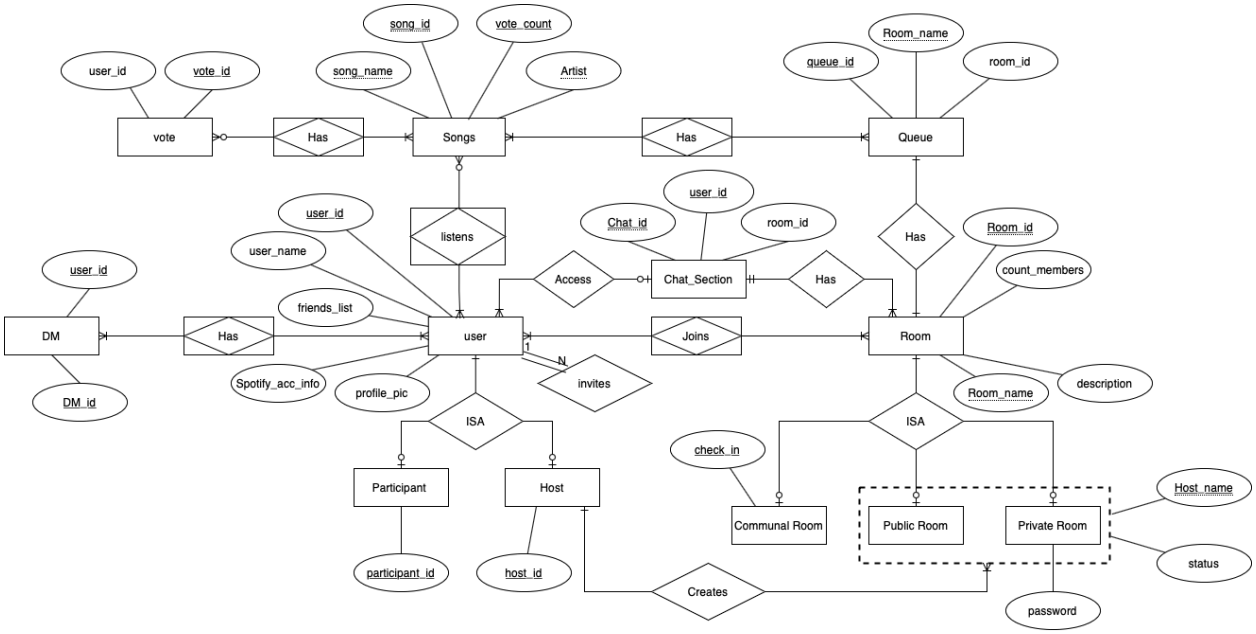
4. **Friends (weak)**
    a. Friends shall be able to be added from zero to many registered user's friend's list.
    b. Friends shall be able to be removed from zero to many registered user's friend's list.
    c. Friends shall be able to be blocked from zero to many registered user's friend's list.
    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

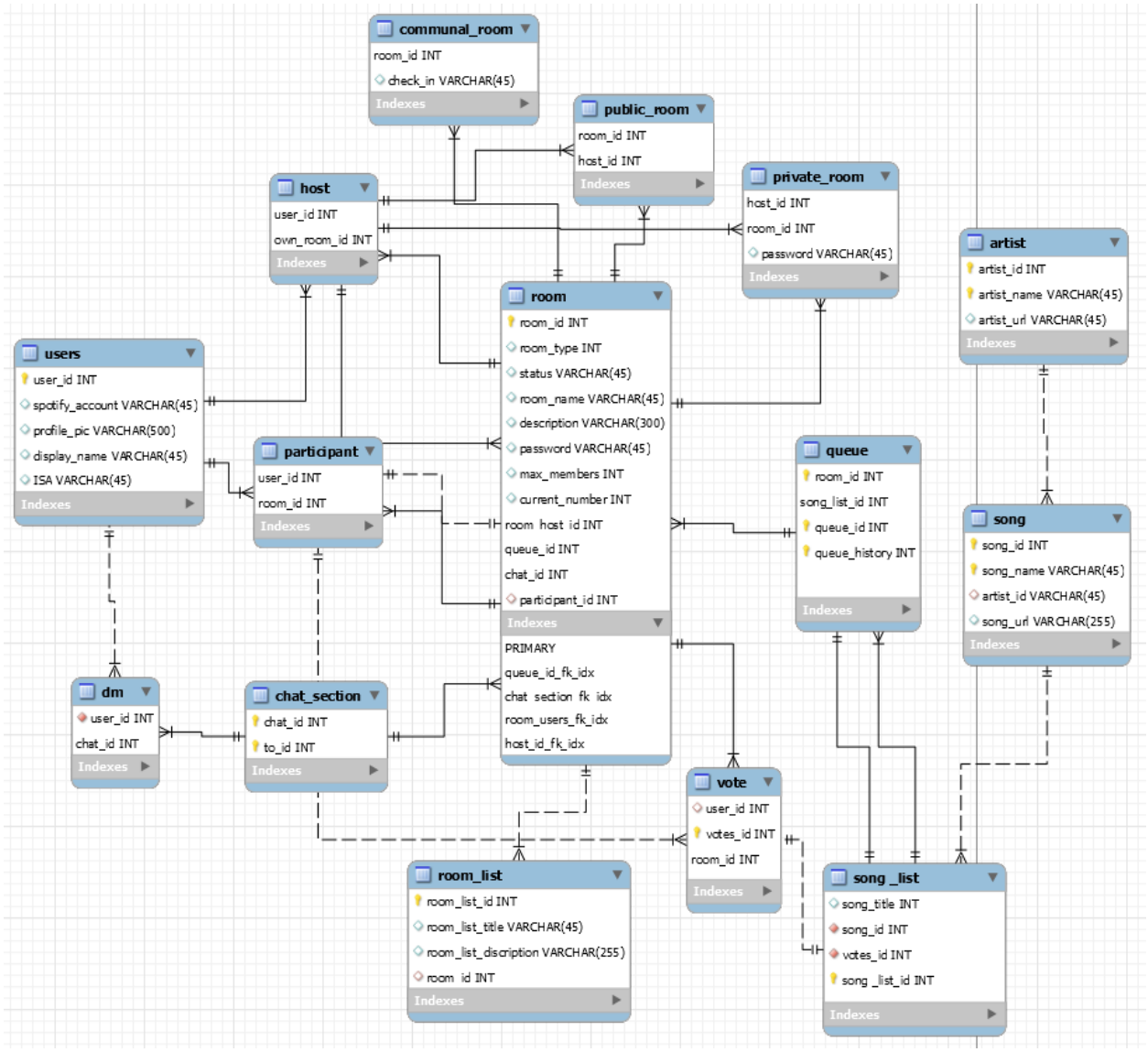
6. **Website (strong)**
    a. Website shall display zero to many Direct Messages
    b. Website shall display zero to many added friends
    c. Website shall be able to allow user to like zero to many playlists

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).

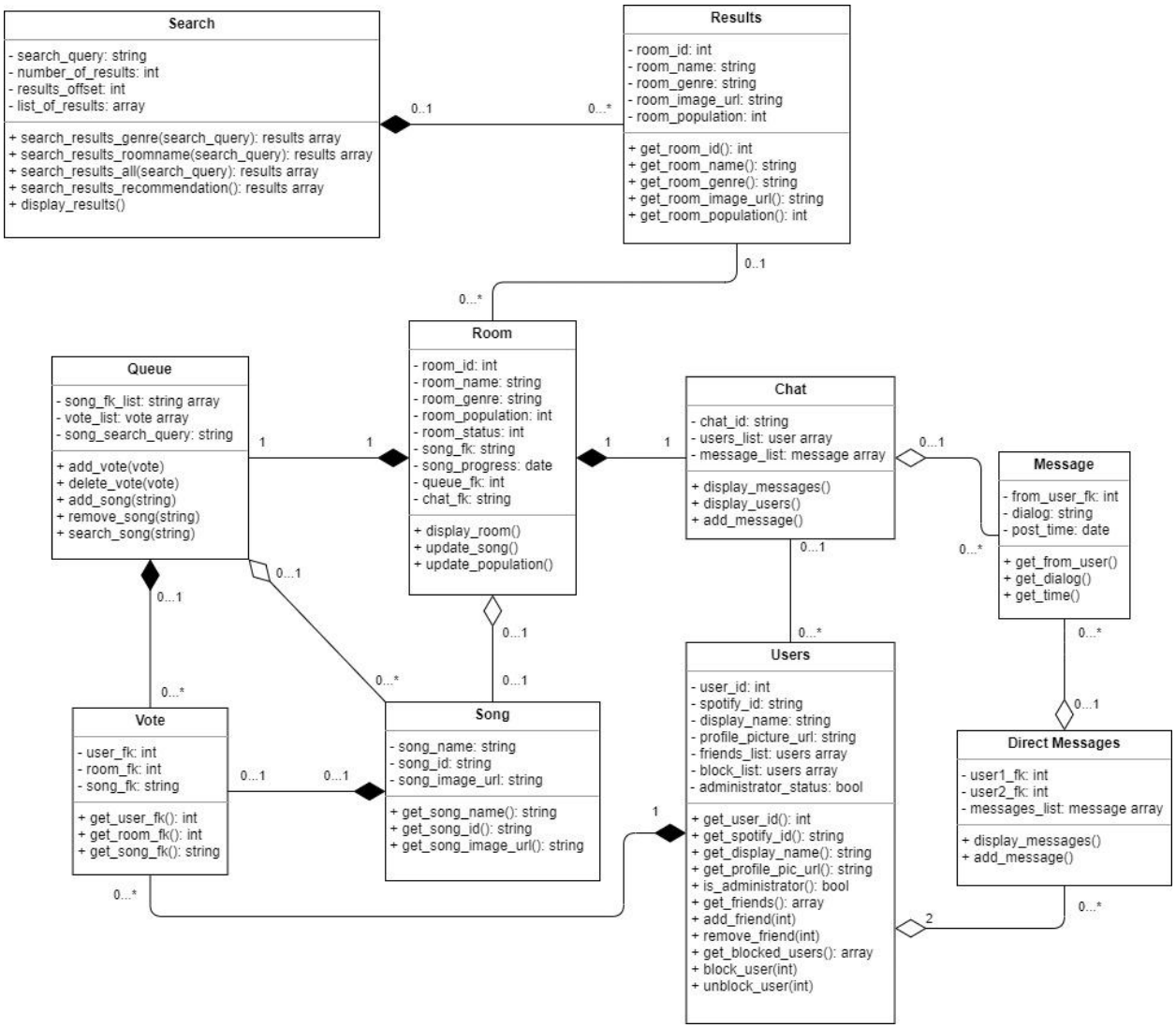
Inner-Joint Search: We connect data tables using foreign keys to create advanced search functions. We can combine information from multiple tables to provide more useful information when users search.

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.

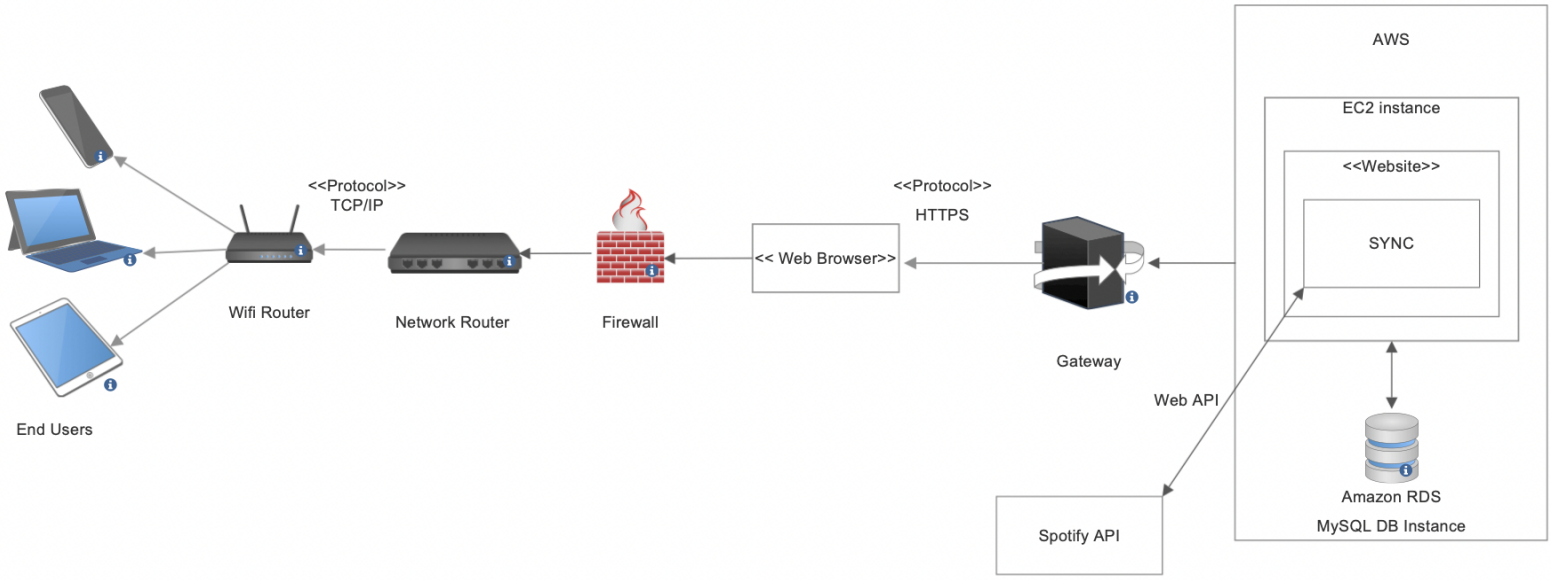
Search Spotify Data: Since Spotify provides many useful information (of users, songs, singers, playlists, etc,) for free of charge, we will implement our list of APIs with an endpoint from Spotify to utilize it.

6. High Level UML Diagrams

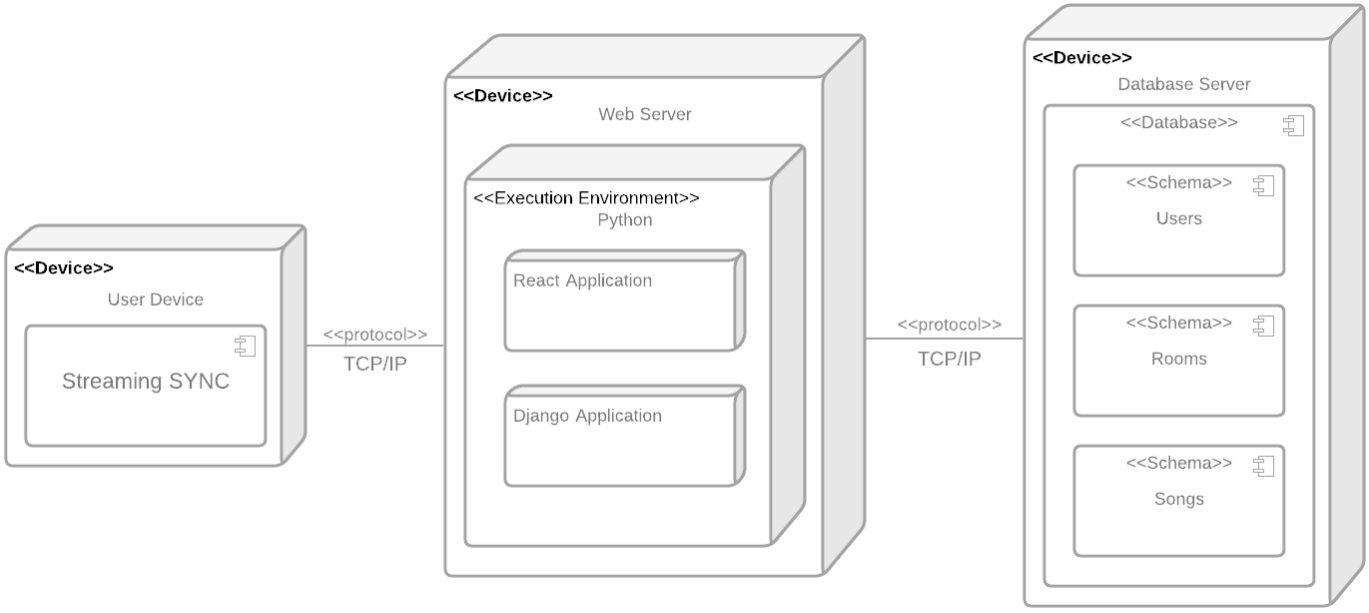


7. High Level Application Network and Deployment Diagrams

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.