

## 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

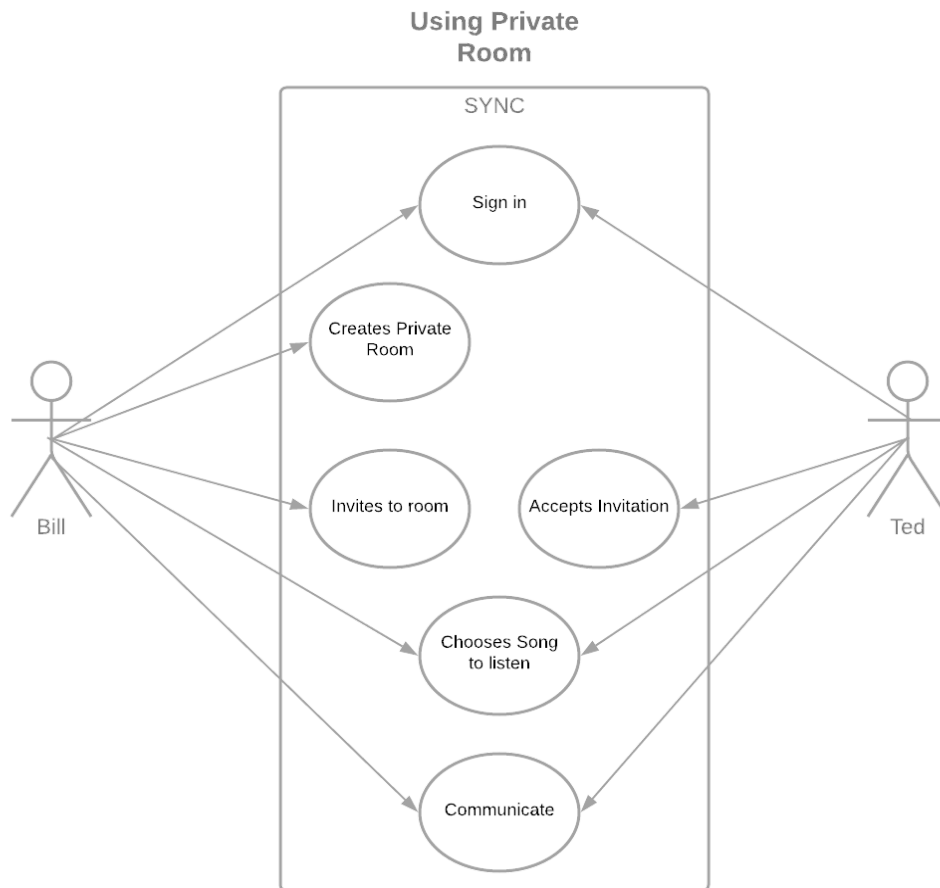
## **1. Executive Summary**

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.

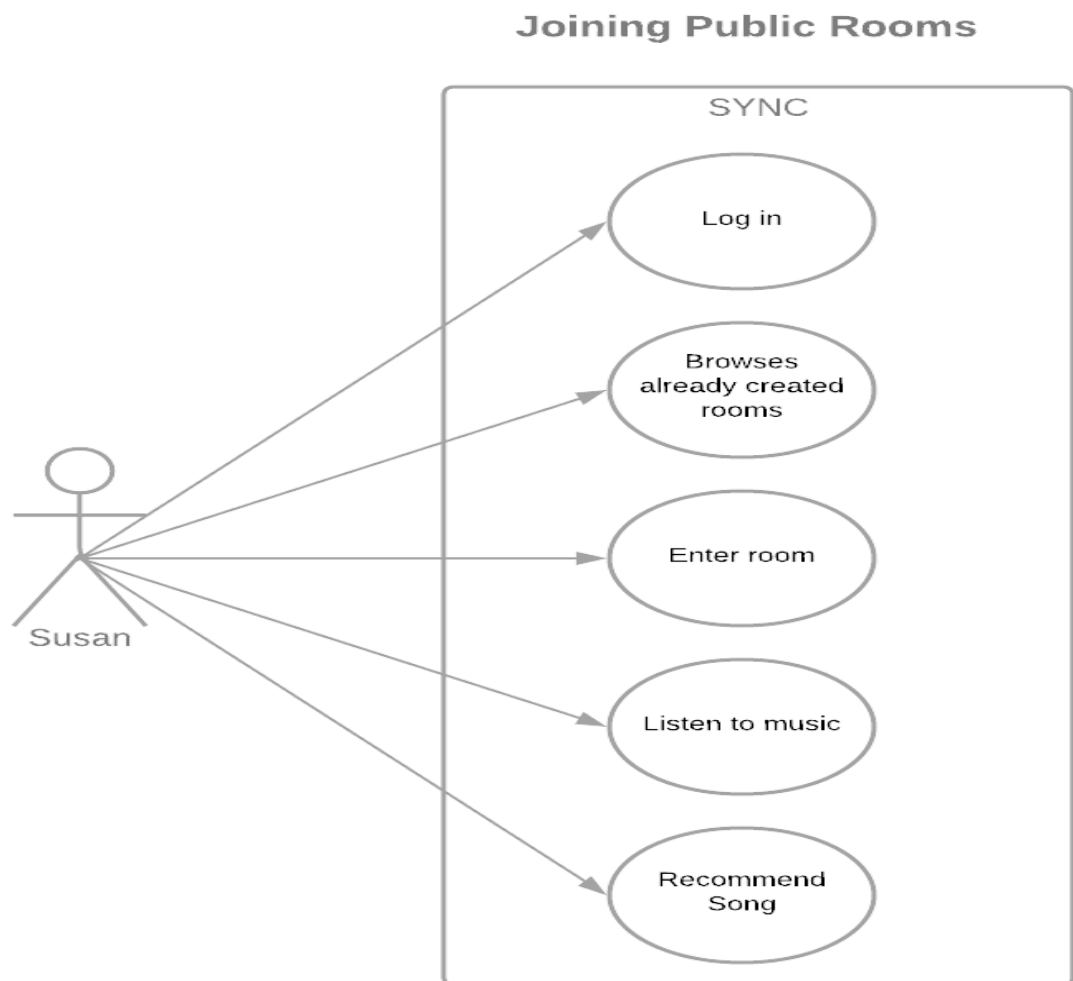
Now, you might be wondering, what does this mean exactly? Simply put, we put the control in the hands of the users. Most often the case with these types of web applications, is the control is often put in the hands of a single "DJ" while others in the room must simply idly sit as a single person chooses 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

<b>Title:</b>	Using Private Rooms
<b>Actors:</b>	Bill and Ted (Friends who share similar music preferences)
<b>Description:</b>	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.



<b>Title:</b>	Joining public rooms
<b>Actors:</b>	Susan (Lonely music enthusiast)
<b>Description:</b>	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.

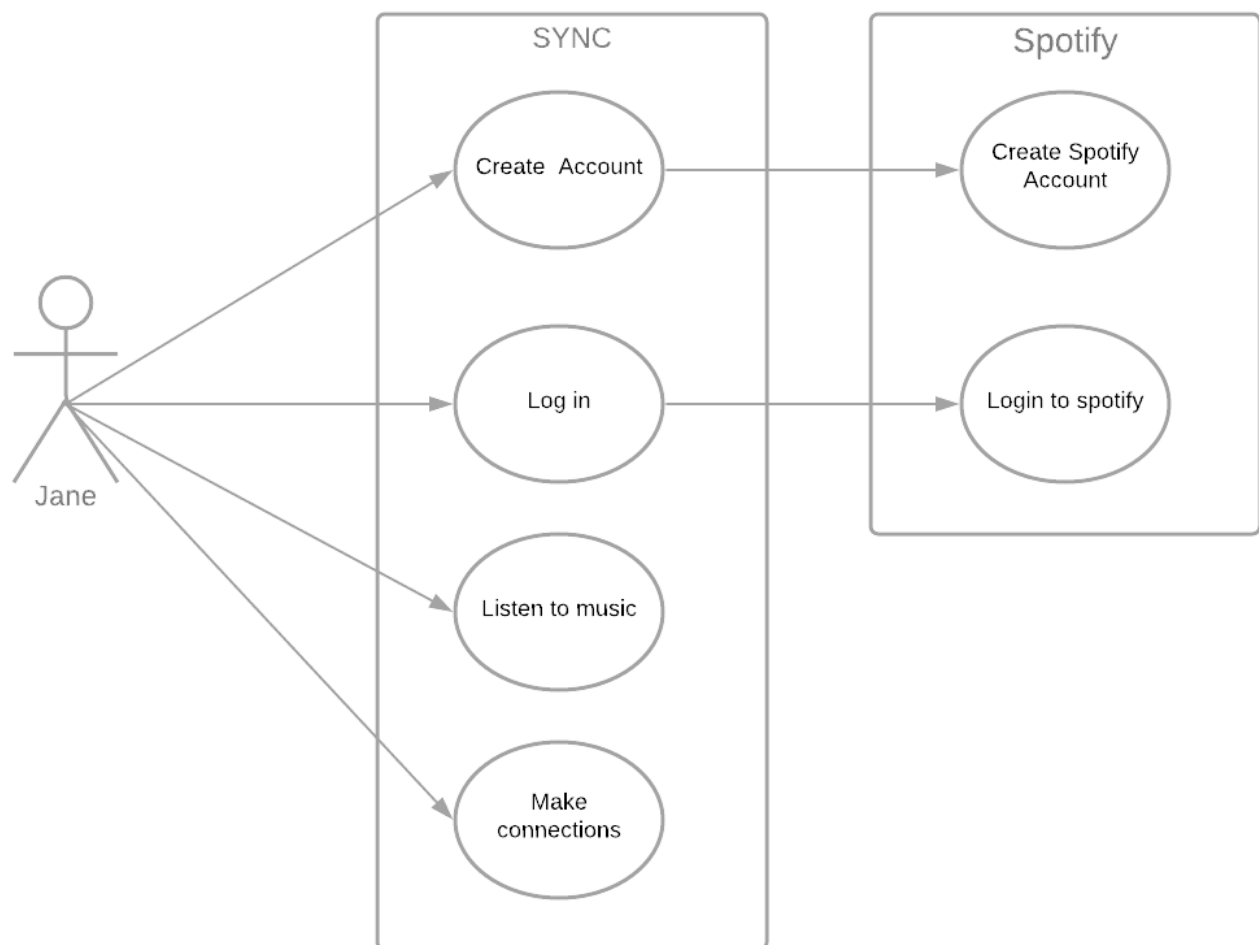


<b>Title:</b>	Using public room for large number of users
<b>Actors:</b>	Nathaniel and friends (Party friends)
<b>Description:</b>	COVID-19 is over, everyone has their vaccines. Now Nathaniel wants to throw a party at his parents' house while they are out of town. He invites all his friends to the house, 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.

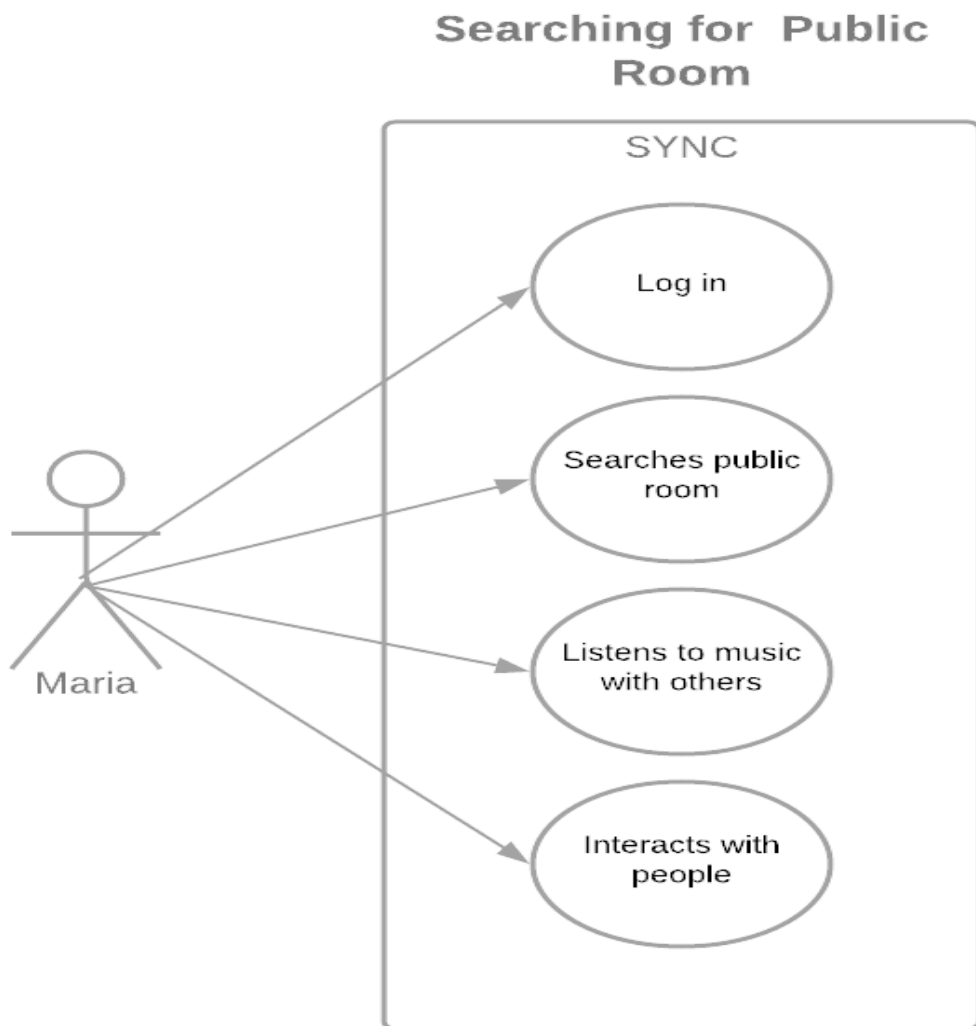


<b>Title:</b>	Authenticate user
<b>Actors:</b>	Jane (New user)
<b>Description:</b>	Jane wants to be able to create an account with SYNC so she can meet new people and listen to music together. After finding out that she needs a Spotify account to use SYNC, she creates a Spotify account. Once she has 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.

### Authenticate User

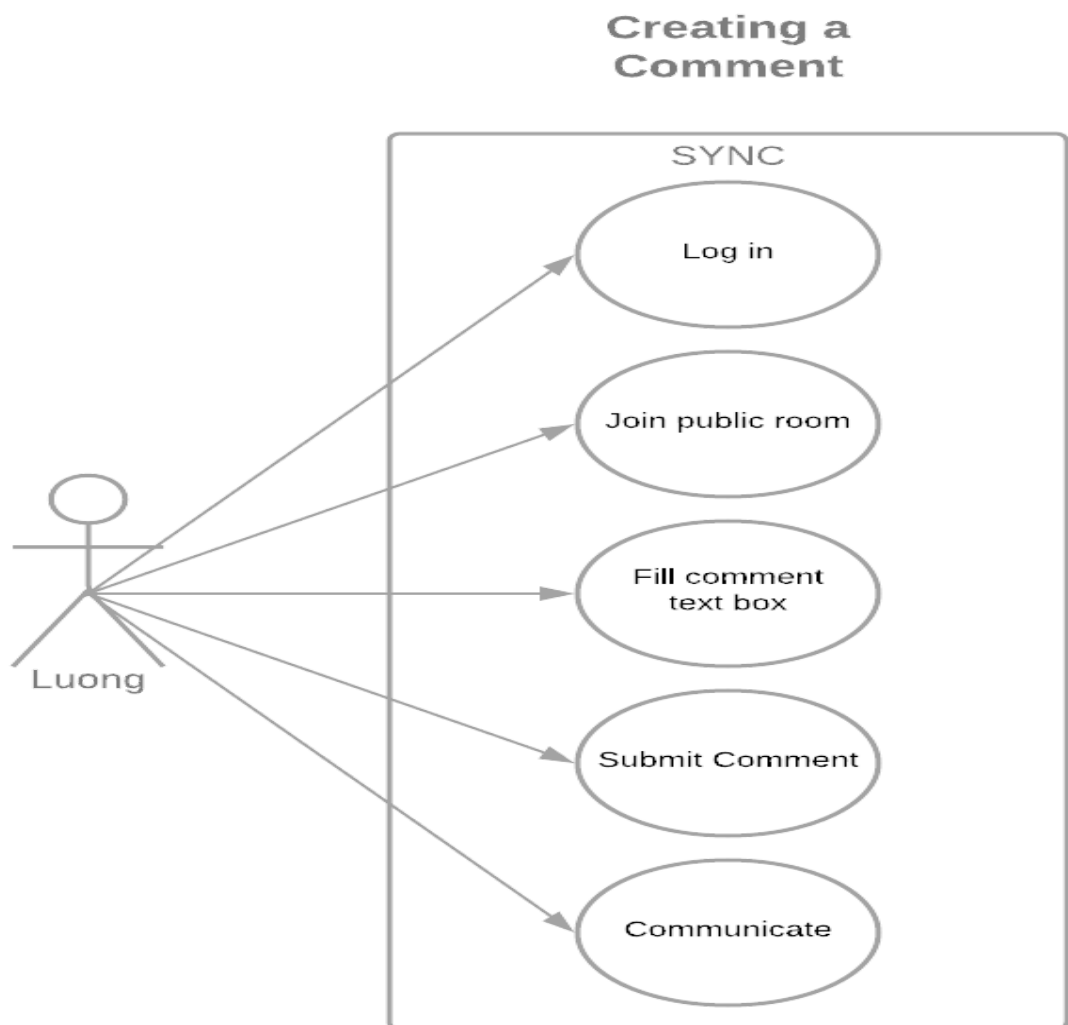


<b>Title:</b>	Searching for a public room
<b>Actors:</b>	Maria (Registered user)
<b>Description:</b>	Maria is a student who enjoys listening to music while studying. She would usually listen to live music streams on YouTube but she wants to make friends she can message and interact more with. When discovering 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.

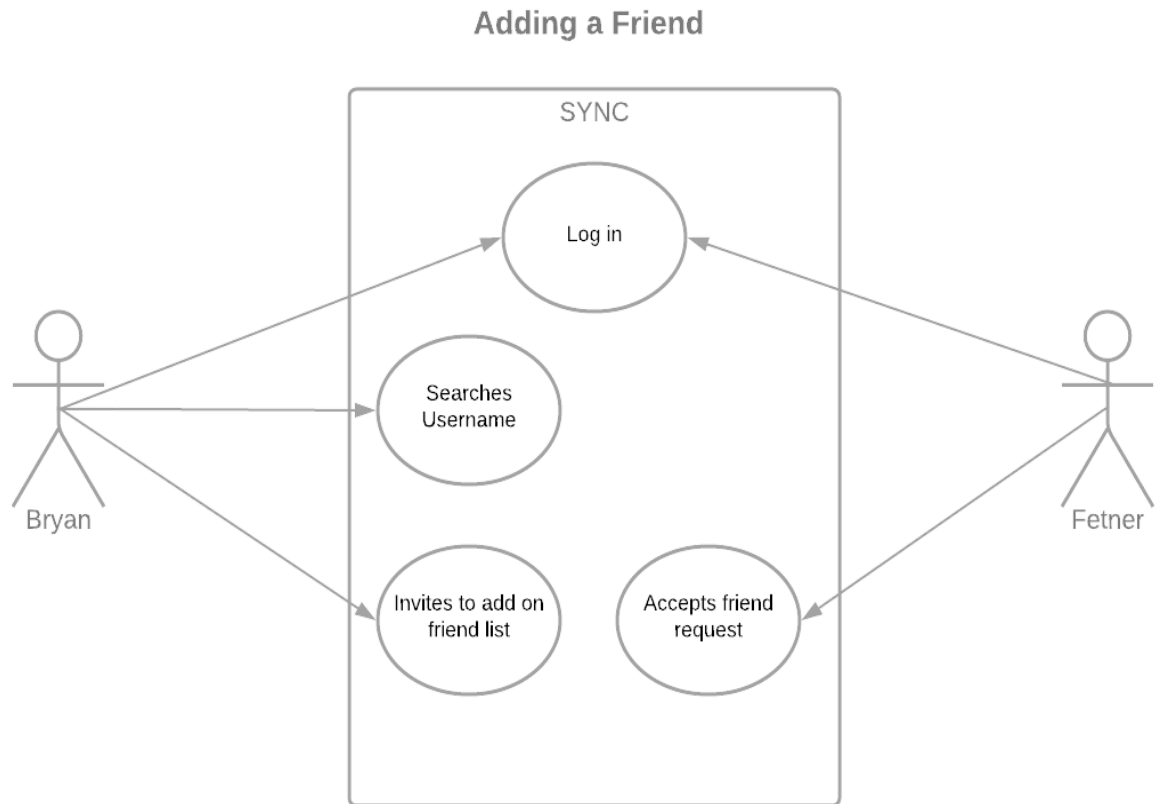




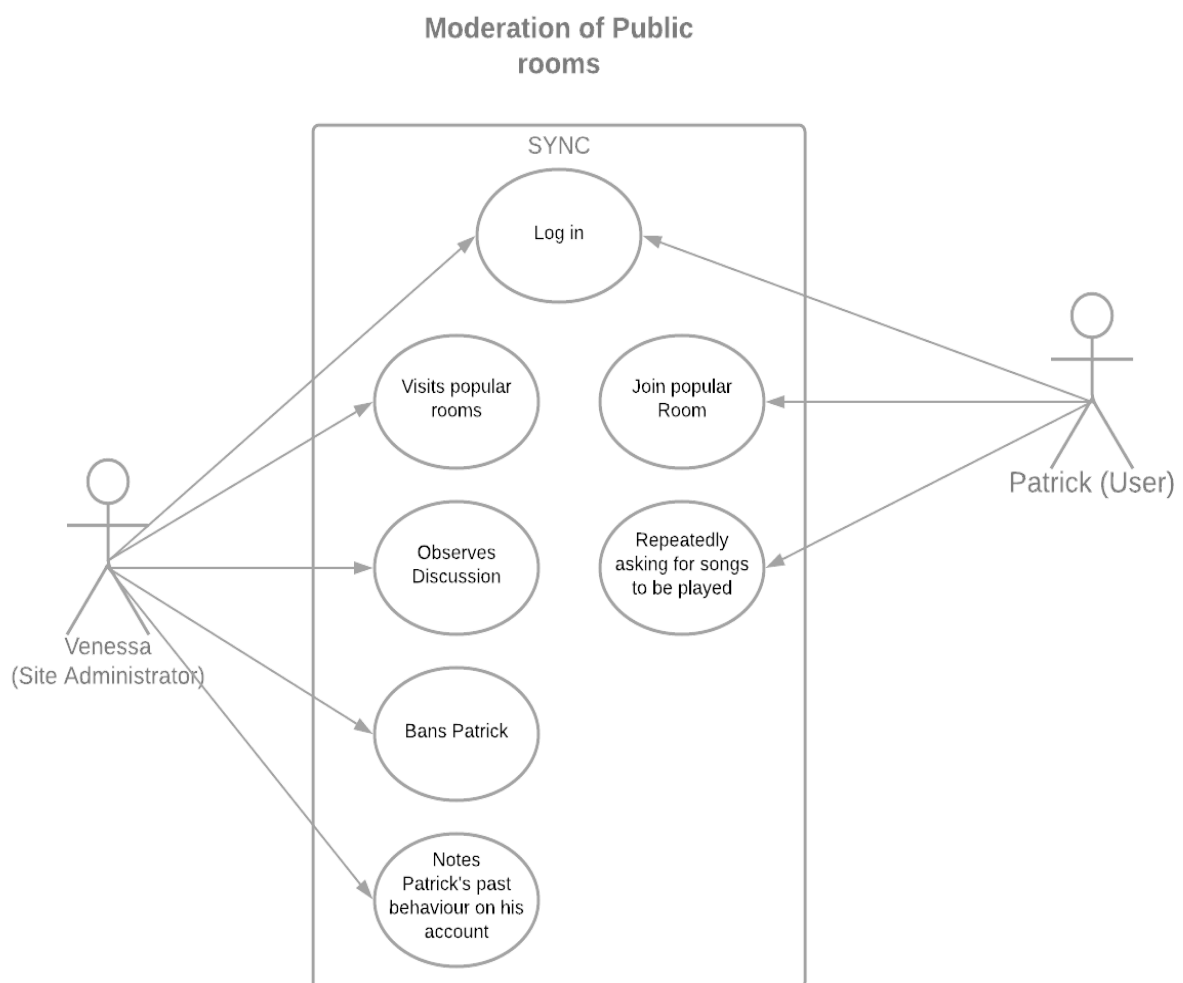
<b>Title:</b>	Creating a comment
<b>Actors:</b>	Luong (Room attendee)
<b>Description:</b>	Luong is a game fanatic who enjoys showing off his vast knowledge in video game soundtracks. He joins a public room that is playing the Cyber Punk Soundtrack. To show off his knowledge, Luong uses the comment text 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.



<b>Title:</b>	Adding a friend through profile search
<b>Actors:</b>	Bryan (Registered user), Fetner (Registered user, Friend)
<b>Description:</b>	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.



<b>Title:</b>	Moderation of Public Rooms
<b>Actors:</b>	Vanessa (Site Administrator), Patrick (Registered User)
<b>Description:</b>	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 database if a database query fails.

## **Storage**

- 72. Store users' listening history on the database.
- 73. Store users friends list on 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 list on 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

	<b>Discord</b>	<b>Soundcloud.com</b>	<b>App.jqbx.fm</b>	<b>Beatsense.com</b>	<b>Listentogether.app</b>
<b>Strengths</b>	Seamless onboarding experience.	Useful platform for aspiring artists.	Supports multiple services.	Users can jam with their music matches.	Nice UI
<b>Weaknesses</b>	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.
<b>Pricing</b>	Free, has option for Discord Nitro	Free, has option for Pro version	Free, needs Spotify premium account	Free	Free
<b>Social Media</b>	Facebook, Instagram, Twitter	Instagram, Snapchat, Facebook, Twitter	Discord, Facebook, Twitter	Blog posts, Facebook, Twitter	None
<b>Onboarding</b>	Simple, Smooth experience.	Takes quite time to follow.	Seamless onboarding	Too many steps to follow.	Moderate number of steps.
<b>Usability</b>	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.ap p	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

		<p>section he wrote, he contributed to features and concepts. Contributed thoughts and input in other sections such as the Competitive Analysis section. Attended all meeting and gave input when he seemed best knowledgeable of subject. Put out work when asked and continuously checked with team when making decisions.</p>
Front End Member	Hirva Patel	<p>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 site as whole which lead to requirements and entities implemented in document. Attended all meetings and contributed in conversation when she felt best fit and was passionate about the subject. Reported back to team when she had updates on work and wanted feedback. Took constructive criticism well and worked off of it.</p>
Github Master and back end member	Vishakha Tyagi	<p>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 suggestion towards implementation of site which lead to requirements and entities involved in documentation. Attended all meetings and gave input when she felt best knowledgeable upon subject and is passionate upon ideas. Checked with other member, especially with team lead, when it comes to making any decisions within documentation.</p>
Back End Member	Ashwini Managuli	<p>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</p>

		<p>Architecture and Technologies Used.</p> <p>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.</p>
--	--	---

## 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**