Title Page

CS495 Spring 2018

Software Design Document & Presentation

Juked

Francisco Rovelo, Laura Phillips, John Mills, Mohammed Rahman, Noah Patton

02/06/2018

Revision Page

Revision History

#	Revisioner:	Comment:	Date:
1	Francisco Rovelo	Added Title, Revision, Contents page, & Format	02/06/18
2	Laura Phillips	Added Introduction	02/08/18
3	Mohammed Rahman	Added function , Non function, Comparison	02/16/18
4	John Mills	Added project description	02/16/18
5	Mohammed Rahman	Updated non functional requirement	02/25/18
6	Noah Patton	Reviewed overall paper	02/16/18
7	Noah Patton	Made powerpoint presentation	02/25/18
8	John Mills	Added class diagram and Use cases	02/26/18
9	Laura Phillips	Editing formatting and TOC	2/26/18
10	Laura Phillips	Added activity diagrams	2/26/18
11	Laura Phillips	Updated definitions	2/26/18
12	John Mills	Updated class diagram	3/1/18
13	All	Final pass before first submission	3/1/18
14	Mohammed Rahman	Added observations, reviewed the paper	3/1/18
15	Laura Phillips	Activity diagram descriptions	3/1/18
16	Noah Patton	Added use case description	3/1/18
17	John Mills	Added class diagram description	3/1/18
18	Noah Patton	Reviewed function and nonfunctional	3/1/18
19	Laura Phillips	Reviewed additions to § 3 & 6	3/1/18
20	Francisco Rovelo	Reviewed/edited/added information to § 4	3/1/18
21	Laura Phillips	Added prospective database information	3/1/18
22	Laura Phillips	Edited requirements document using feedback	3/28/18
23	Laura Phillips	Uploaded all team members' sequence diagrams	3/28/18
24	Francisco Rovelo	Detailed join/create lobby sequence diagrams	3/28/18

25	Laura Phillips	Detailed skip song/remove song/has voted diagrams	3/29/18
26	Mohammed Rahman	Added use cases	3/29/18
27	Noah Patton	Added detail to sequence diagrams	4/2/18
28	John Mills	Updated class diagram	4/3/18
29	Laura Phillips	Updated table of contents, reviewed for submit	4/3/18

Table of Contents

3.0 Introduction

3.1 Purpose

Juked is an application to give the power of music to the people. When a group of friends are together, the task of picking the music they listen to typically falls to one person. The host of a party or gathering can't relax and enjoy their company because of the stress of constantly choosing music everyone will enjoy. Juked allows the host to share this responsibility by allowing the entire group to pick their favorite songs and vote on their friends' choices so everyone gets to listen to music they like. Aside from making

a party more interactive, the app also provides opportunity for people to venture out of their comfort zone. It also possible for people to taste something new they were not aware of before, and prevents one person from monopolizing the party.

3.2 Goals

Our goals for Juked are to produce an application that is useful, user friendly, convenient, and fun. We want Juked to be intuitive and easy to understand so that any group of people can make use of it. We envision this application to be useful in everyday life, especially for young adults and teenagers that have frequent get togethers with friends. Having a clean and clear UI will help accomplish these goals, so that parties can be joined easily and songs can be found quickly and concisely. When an application is cluttered or hard to navigate, users are easily turned away. The application has potential for growth because once a party host decides to use Juked, any party guests that wants to have input on the the song selection will also download Juked. Going forward, all of those party guests can suggest Juked at a different party, where all of those attendees will download the application, and so forth.

3.3 System Scope

Juked is an Android based application. It uses a NodeJs server and a MySQL database. Juked allows partygoers to select song to be played. It allows the users to up or down vote song selection, making it an interactive get together. Juked allows host to have input from the guests, helping to make a more successful party.

3.3 Definitions

- API - Application programming interface
- Guest - Guest of a party, can join a lobby and request songs
- Host - Party host, has guest privileges and administrative controls of party
- Lobby - The host creates a lobby where the host and guests can select options and view the songs that have been chosen by the other guests.
- Queue - List of songs chosen by the party guests and host, played in a first in, first out order
- Soundcloud - 3rd party music listening application, hosts independent music
- Spotify - 3rd party music listening application, requires premium (paid) subscription
- UI - User interface, what is displayed on the screen that the user will interact with to use the application
- UX - User experience, the experience a user has while interacting with the application. The design and layout of the app should be chosen with the user experience in mind.

4.0 Project Description

Juked will allow one phone to act as a host, and let anyone connected to that host's lobby to choose the songs that will be played from the host phone. When the host creates a lobby, the

app assigns it a four digit number from 0000 to 9999. The host can let the party goers know the generated lobby code. The guests then use that number to join the lobby. Every user gets to choose one song in a given turn. Each guest may upvote or downvote any song in the queue and the top voted song goes to the top of the queue, while negative voted songs get pushed to the bottom (hence it is collaborative). If a song obtains a low enough score it is removed from the queue. If the host forgets to close out the lobby at the end of the party, the app's back end automatically closes the room after a certain amount of inactive time.

4.1 Feature Overview

Guests have minimal features, of which include selecting songs, viewing the party's song history, and voting up/down songs. The host has all the perks of a user, but more administrative powers.

4.2 Generating Lobby

At the main screen, the host chooses the option to host a lobby. A random four digit number is assigned to the host to give to guest. If the app becomes popular we will increase available lobby numbers exponentially.

4.2-A Selecting Available Music Median

The host will also be given the option to limit the guest to choose from just Spotify, just Soundcloud, or both (possibly YouTube if time allows). The host will be required to have the Spotify or Soundcloud application downloaded to their phone in order to use their streaming services in Juked. Additionally. Spotify will only be available for premium Spotify users.

4.2-B Host Options

The host is allowed to skip songs or delete songs that he deems not fit in the queue. The host can also pause and resume playback as necessary.

4.3 Guest Menu

Once a guest chooses to join a lobby, they will be prompted to enter the four digit code that the host gives out. Once in the lobby, the guest will choose both a nickname and an avatar. Avatar options will include either a pre-uploaded one(default avatar) or upload one from the device

4.3-A Searching and Selecting

The guest will have a search bar on the guest screen to search for songs. The search uses either Spotify's search or Soundcloud's to pull up the closest match to the song being typed. The guest then has the opportunity to pick the song and it will automatically be added to the queue.

4.3-B Upvoting and Downvoting

The guests will also have the ability to see songs that are in the queue. They can upvote or downvote any and all songs, but may only vote on each song once. The more upvotes a song gets, the higher it will move in the queue. If a

song gets 5 (or other predetermined amount) or more downvotes, it is removed from the queue entirely.

5.0 Functional & Non-Functional Requirements

5.1 Functional Requirements and Priority Levels

a. High- The app will give user the option either to create a lobby or join a lobby.
b. High- User will be able to select a song from Spotify. In the future, we will be open to adding more platforms such as Soundcloud, Youtube or Google Play Music - (Low priority).
c. High- User will be able to upvote or downvote a song. The user can change the vote at any time by either pressing the same button to cancel the vote or pressing the opposite button to do the opposite action. However, the user will only get one vote per song. A user may change their vote from up to down and vice versa, but they cannot upvote a song twice.
d. High- User will have the opportunity to add another song selection to the queue after their previously chosen song has been played.
e. Med- Host will be able to remove a song from the queue. This will help to ensure that host maintains absolute control over the playlist.
f. Med- Lobby codes will be (at least) 4 digits. If the app becomes popular, this number can change to add lobby options.
g. High- Host may not create more than one lobby at a time. Currently we have only 10,000 possible lobby codes. This would ensure better usage of limited resources.
h. High- User may not choose more than one song in each turn. If the user has a song selection in the queue, they cannot make another selection until their first one has been played or voted out.
i. Low - If a lobby has an empty queue for a certain amount of time, the lobby will be destroyed and all users removed, including the host. Since the app only allows 10,000 lobbies at present, this will make sure we are using the number of available lobbies efficiently.
j. Medium - Delete any system cache after the party closes ensuring data security.

5.2 Non-Functional Requirements and Priority Levels

a. High- Keep track of all the lobbies created. The lobby numbers have to be unique, otherwise user's songs may be played in a different room then they are in.
b. Medium- The system must use resources efficiently. Given the possible number of rooms, votes and actions at once, keeping things efficient is a must.
c. High- System need to be able to use Spotify API/SDK. Spotify has the largest market share of the music streaming industry and will provide a large library of songs for users to select from.
d. Medium - Have a smooth user experience that is clear and easy to follow.

6.0 Functionality Comparison

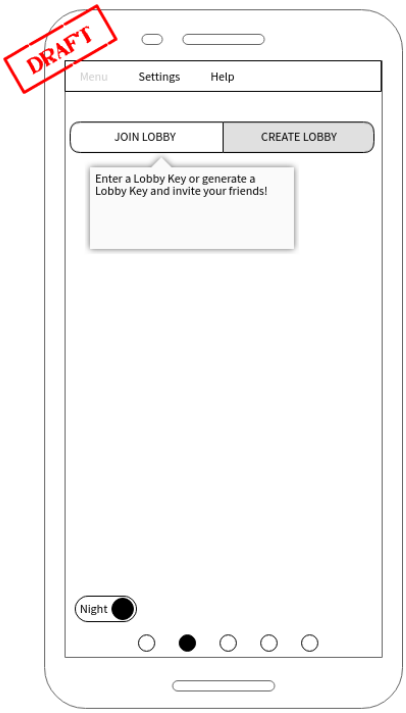
We view our biggest competitors of the market to be Spark.dj, Flo, Troppo, Festify, Jukestar. Many of these applications are only available on iOS or are web based applications. Since our application will be developed for Android, this diversifies our audience from the users of many other apps. Most of the apps other than Flo do not have spam prevention, which stops users from adding the same song multiple times. Juked prevents this by allowing each user one song in the queue at a time. While Flo also prevents this, it does not allow up or down votes to affect playback order. Troppo allows to play song from guests own phone via wifi, which can be a potential security hazard or violate file sharing guidelines. Juked is also one of few apps that will prevent guests song choices form affecting their auto generated playlists by using Spotify in a private session. Spotify has algorithms that use listening history to build playlists and song mixes based on a user's taste in music, and using Juked will not effect these algorithms. One shortcoming of Jukestar is that the application requires too separate applications, one for a host and one for a guest. The host application is only for party management, requiring a host to also download the guest application in order to add music.
Overall, Juked intends to have better functionality and user experience in comparison to other apps.

App/Req	Platform	Password Protection	Song Selection	Popular song	Spam Prevention	Affect Host's personal ranking?
Spark.dj	Apple	No	Spotify	Up/Down Vote	No	No
Flo	Apple	No	SoundCloud/Spotify	No	Host permission	Yes
Troppo	Android	Share via Wifi	User Device, online	Up/down vote	No	N/A
Festify	Web based	Party code	Spotify	Up/down vote	No	N/A
Jukestar	Apple/ Android	Yes	Spotify	Up/ Down vote	No	No

Juked	Android	Random 4 digit code	Spotify	Up/Down Vote	Yes	No

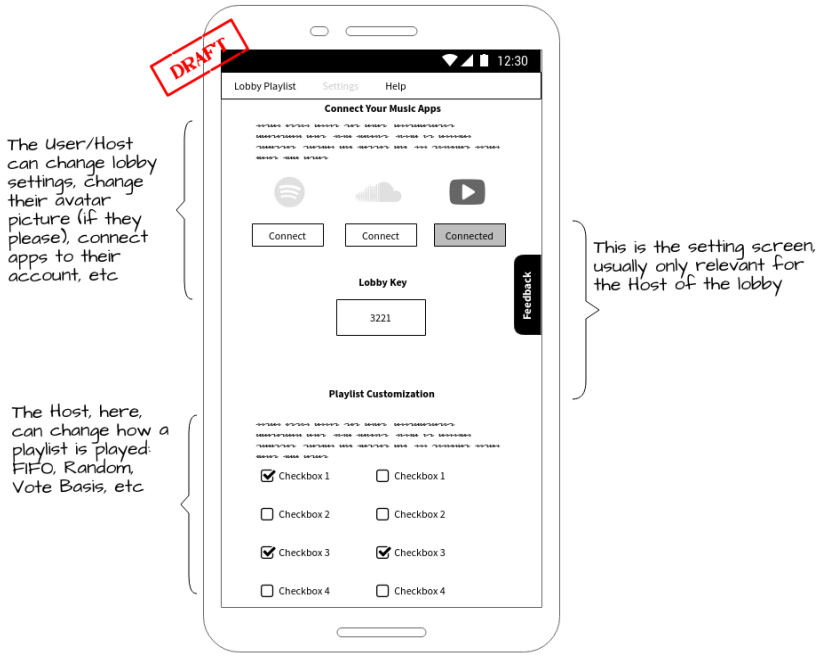
7.0 Analytical Diagrams and Description

7.1 - UI Mockups

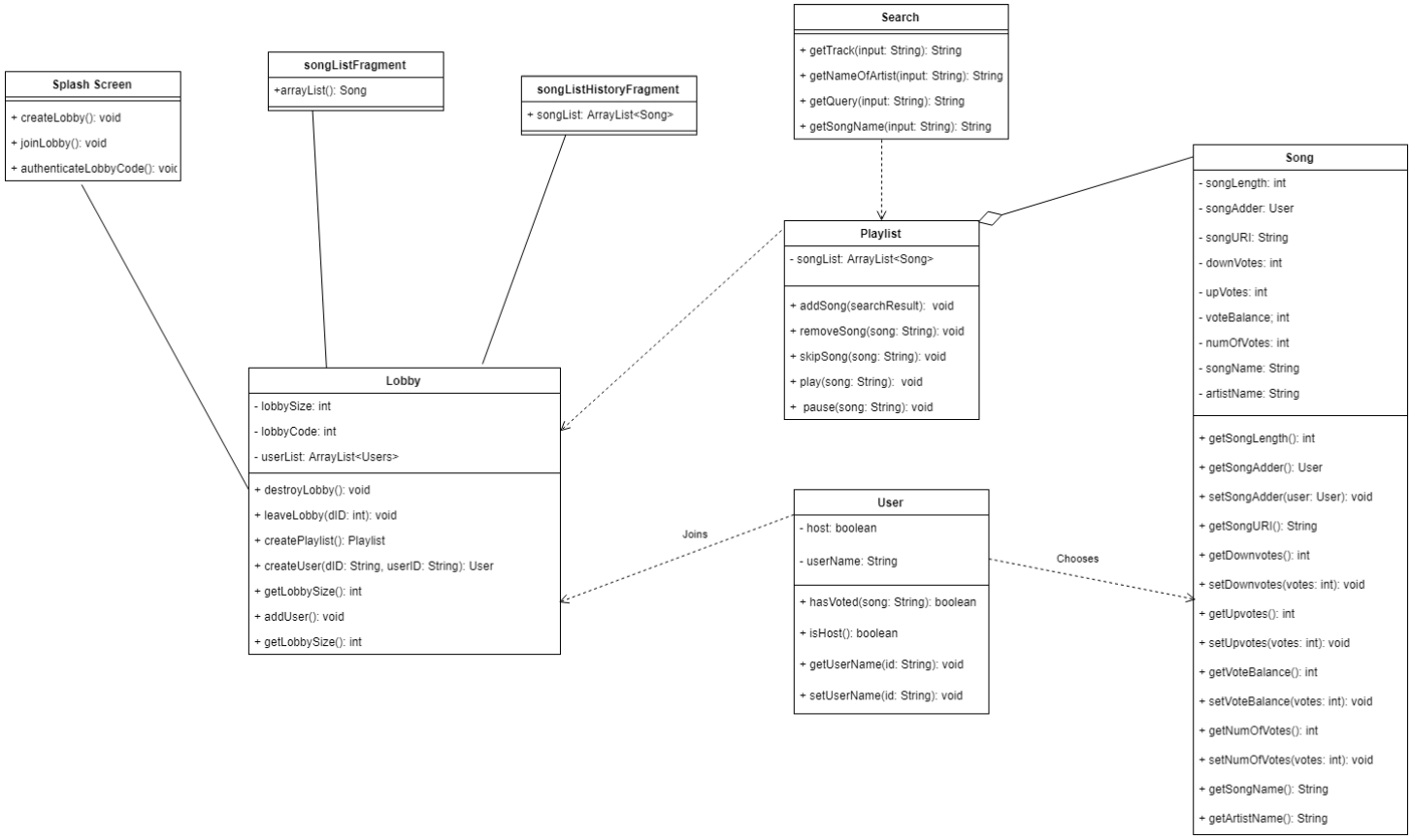


The UI/UX is designed to remain minimalist while maintaining a large amount of functionality on as little screen space as possible. One of the key points of our app vs the competition is is ease of usability.

We hope to use a form of javascript for responsive UI.



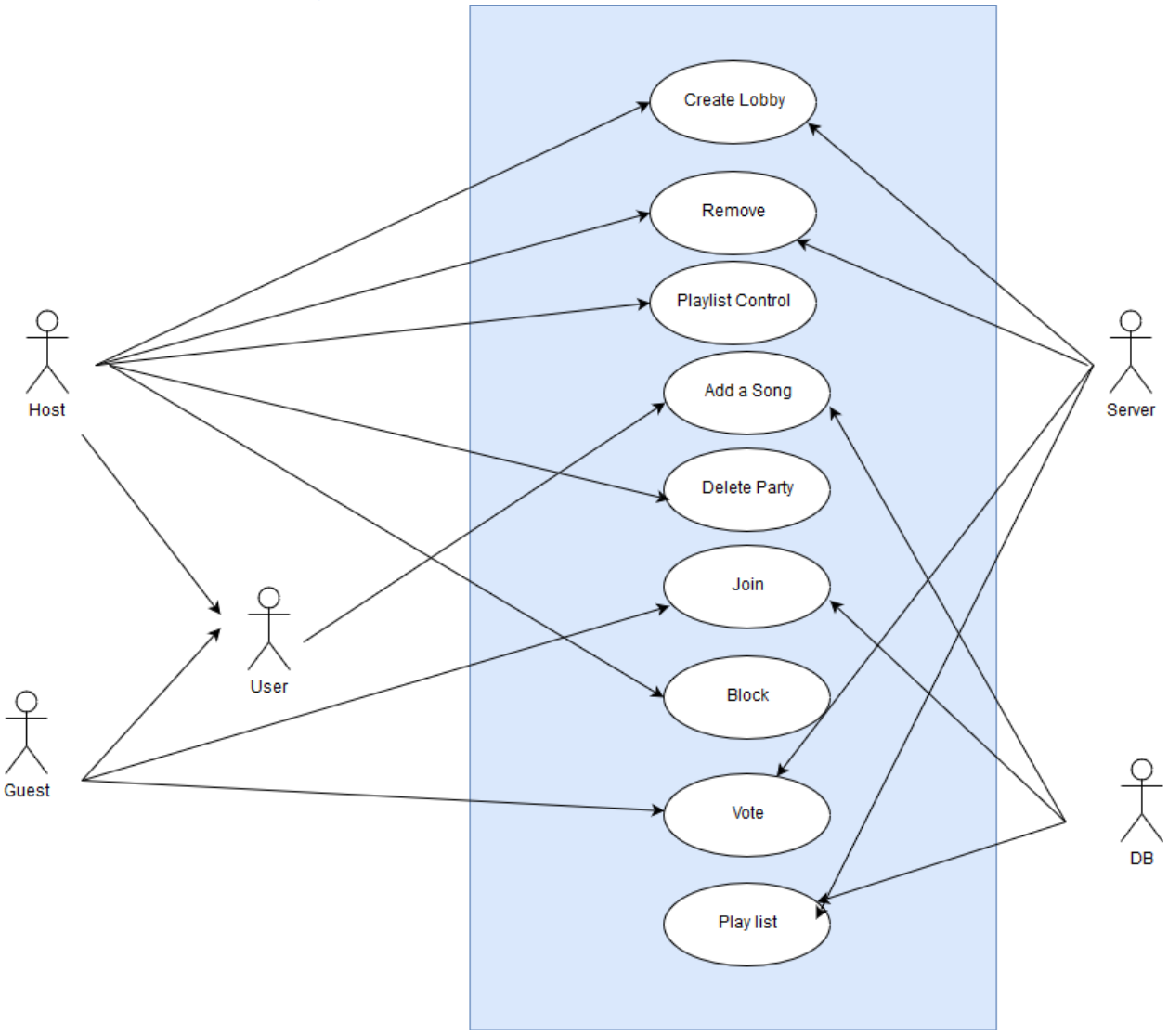
7.2 Class Diagram



We currently plan on using five classes for this app. As seen, a user class will be a base class. It will hold a userName (or nickname) and a key that identifies it uniquely. The user class will currently have methods that allow them to search songs, select a song from the search results, and also show other users that are currently in the lobby and the song that they chose. A host will be the same as a user but have a boolean field that determines if it is a host. The host will also be able to kick a user if he has not been cooperating for any reason. A song class will create a song object consisting of only items that pertain to being played. Some of these are the song name, number of votes, the balance of the votes, and also the URI to Spotify or Soundcloud. The playlist class will hold the songs that users have chosen to be played. This is also where the methods that control how the songs are moved up, down, in, and out of the queue. The playlist will be a queue that is comprised of song objects, and has methods that allow for the changing of objects in the queue. Lastly, the lobby will hold all of the other objects. The lobby class creates the user, host, and playlist. It also holds a lobbyID that is stored in a database to uniquely identify that lobby. The splash screen class will hold the opening screen and a few methods. These methods include createLobby, joinLobby, and authenticateLobbyCode. The two songFragment classes have been added that display the current playlist of songs and history of songs. A Search class has been added that will search for the song, and consult the playlist class to be added into the queue. Search will read

in a users input and query the spotify api, which returns a json object. The json is parsed and sent to a Song object. The song object is sent to the songFragmentScreen where the user may choose the song. Once the song has been chosen, it is sent to the playlist class to await being played.

7.3 Use Case Diagram



Use Cases:

1. **Use Case:** Create a lobby.

Description: Host creates a lobby so that users can join the party.

Actors: Host, Server

Main Success Scenario:

a. Host initiates creating a lobby.

b. Server searches for existing lobby. Creates a lobby and return its number to host. Host then passes that number to the guests for them to join.

2. **Use Case:** Enter's a lobby.
Description: Guest users enter an existing lobby.
Actors: Guest, Server.
Main Success Scenario:
    a. Guests manually receives a lobby code from the host, enters that number and a nickname to join the lobby.
    b. Server updates the database with the user's nickname in the respective lobby.

3. **Use Case:** Creates a lobby number.
Description: Server creates a unique lobby number.
Actors: Server.
Main Success Scenario:
    a. Upon receiving a request from host, the server will search through its existing lobby number and return a unique number.

4. **Use Case:** Create a playlist.
Description: Playlist actions in various stage of the lobby.
Actors: Server.
Main Success Scenario:
    a. Backend work by the server.
    b. The playlist is created the moment the lobby is created, updates during the party and is destroyed when the host deletes the lobby.
    c. Users select song from the Spotify search results, the server checks the database to ensure the song is not on the queue. If the song is on the queue than the server asks the user to enter a new song. Otherwise the song is added at the bottom of the playlist.
    d. The playlist can be manipulated by the host user.

5. **Use Case:** Add a song
Description: The host or the guests adds a song to the playlist.
Actors: Host, guests, server

Main Success Scenario:

a. User search a song, the Spotify returns search results, the host or the guests, select a song to be added to the playlist.

6. **Use Case:** Vote
Description: The users can up or down vote song.
Actors: Guest, Host, Server, Database
Main Success Scenario:

a. Users (host or guest) can up or down vote any song on the playlist. Only constrain being each user get to vote only once in each song.
b. The server updates the vote count in database.
c. Song moves and up or down in the playlist according to the vote.

7. **Use Case:** Playlist control
Description: Host control the playlist.
Actors: Database, Server, Host
Main Success Scenario:

a. Host can delete a song.
b. Host also can skip a song even if it is playing at the time.

8. **Use Case:** Block user
Description: Host can remove or block a user.
Actors: Host, database, user
Main Success Scenario:

a. Host has the ability the remove a user.
b. Host also have the ability to unblock a user.

9. **Use Case:** Leave a party
Description: User leaves a party.
Actors: User.
Main Success Scenario:

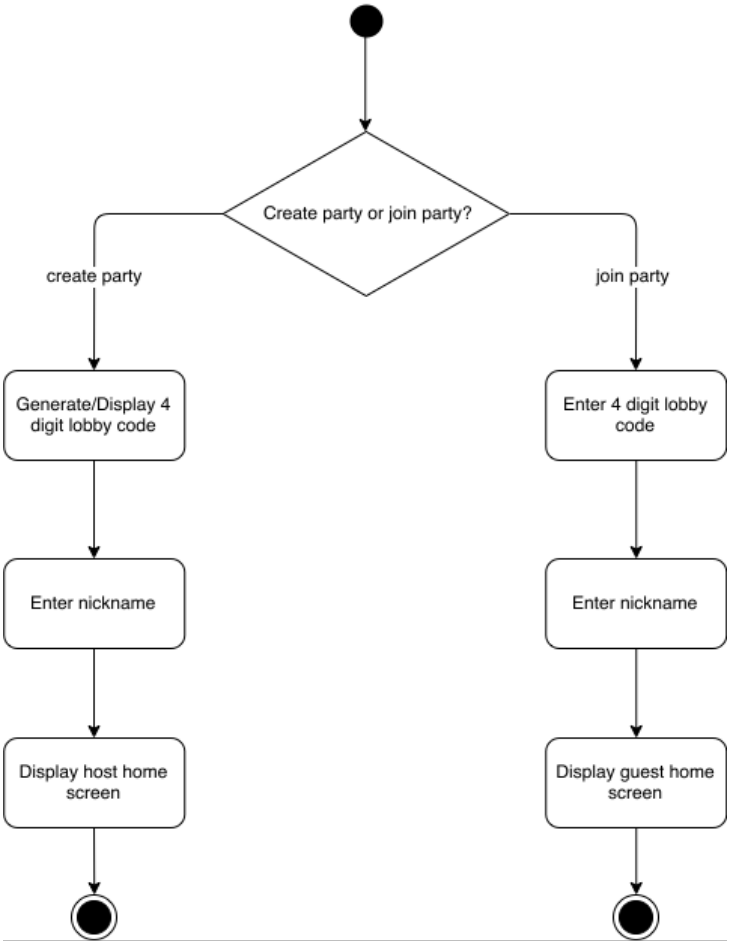
a. User may leave a party at any time.
b. User may join the same party back using the lobby code.

10. **Use Case:** Delete a lobby
Description: Host deletes a lobby.
Actors: Host, server, database
Main Success Scenario:

a. Host leaves the party or chooses to close the lobby.
b. Server clears the database, deletes the playlist.
c. Server clears the lobby code from the database and it is available to next host.

7.4 Activity Diagrams

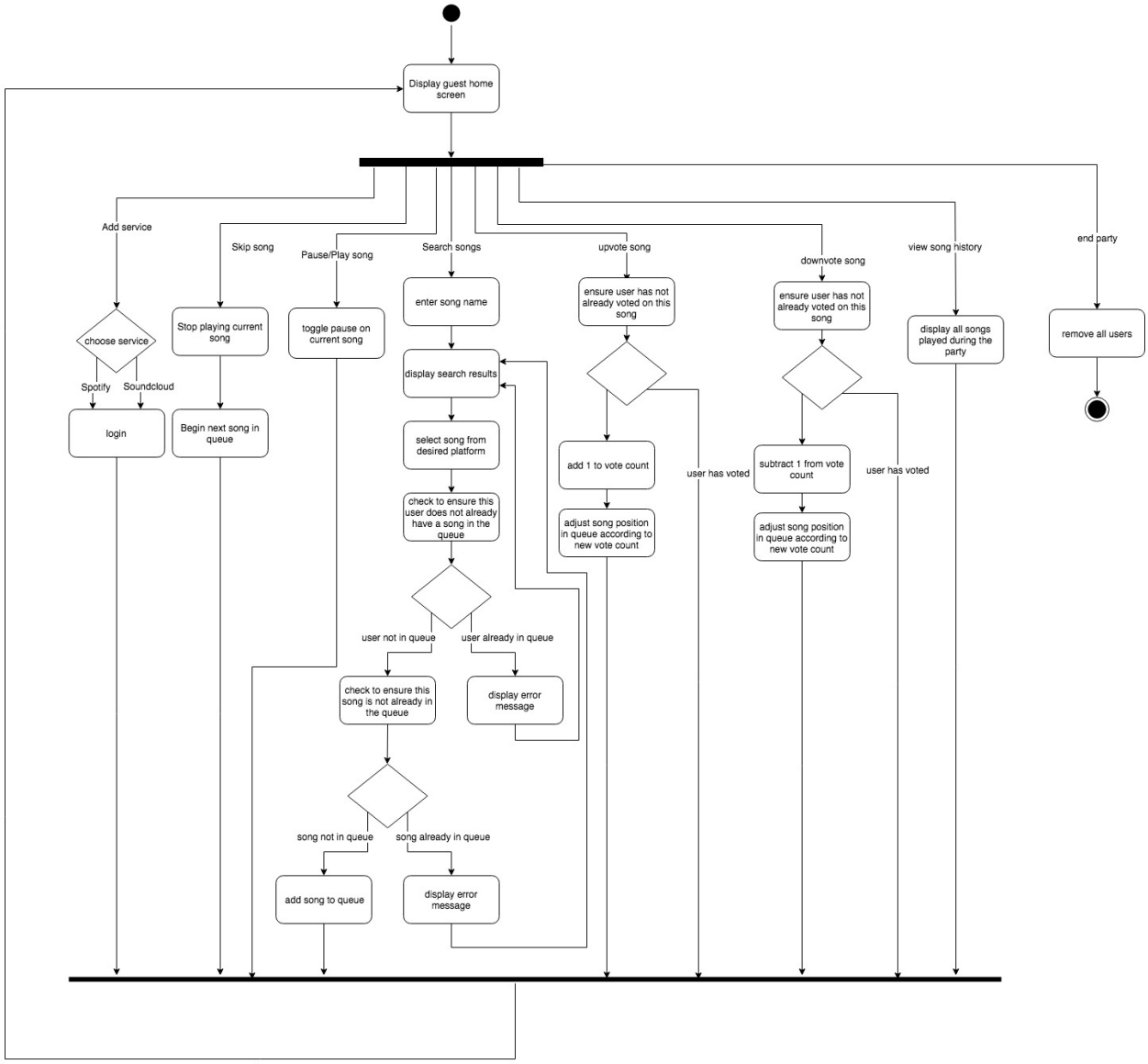
7.4-A Splash Screen Activity Diagram



The above activity diagram shows the user's options upon application startup. If the user is hosting a party, they will need to create a new party, or lobby. After selecting the host option, a four digit code will be generated for the new lobby. As covered in functional requirements, (Section 5.1 g) this will be a four digit number. The digit will be displayed to the host, so that they can share the code with their guests. From this screen, the host will enter a nickname that will serve as an identifier to all guests of the party. Once the nickname has been entered, the host will be taken to the main/home page of the Juked app.

Conversely, if a user opens the app and wants to join an existing party, they will select the guest option. The user will then be prompted to enter the four digit lobby code that was obtained by the party host. Once a valid party code has been entered, the guest will be prompted to enter a nickname, just like with a host. Once the guest has entered their chosen nickname, they will be taken to an abridged version of the home page.

7.4-B Host Home Page Activity Diagram



The diagram above shows the activities available to a host in the Juked application. All available actions will be available from the the host's 'home page'. The home page will display the current queue of songs that have been selected by the party guests, and different buttons to allow interaction with the app. The host will have a few more privileges than the guest to allow more control over the party and music selection.

The host will have 8 different options from the home screen:

1. Add service

   The host will choose to add a streaming service. The host will then choose to log in to their Spotify or Soundcloud account. Once the host has successfully logged in to a streaming service, they will be redirected back to the home screen.

2. Skip song

   While any song is playing, the host has an ultimate "veto power", and can choose to skip the song. The song will stop playing immediately and the next song in the queue will begin playing.

3. Toggle play

   If the host needs needs to stop the music, they can choose to pause playback. If the music is paused, the host can resume playback right where the music left off.

4. Search song

   The user can type in the name of an artist, song, or album into a search bar. The Juked app will search through the libraries of the streaming services that have been added (Spotify and/or Soundcloud) and the results matching the search will be displayed. The user can then choose which exact song and which streaming service they wish to use. Once a song has been selected, the Juked app will then ensure that the user does not already have a song selection in the queue, and that the selected song has not already been placed in the queue by another user. (If a song has already been played during a party, it can be chosen again, so long as it is not currently in the queue). If either of these conditions fail, the user will receive an error notification and be redirected to the search screen so that they may search for another song, or wait until the have no selections in the queue. Once the song has successfully been selected, the user is redirected to the home screen.

5. Upvote song

   The user has the option to vote once on each song. If the user likes the song choice, they can choose to upvote it. When the user selects to upvote, the Juked app will check to ensure that the user has not already voted on the song. If the user has already voted on the song, no change will be made. Once the user has

successfully voted, the song's vote count will be incremented, and the song will move up the queue until it reaches a song with a higher vote count.

6. Downvote song

   The downvote option works identically to the upvote option, except that the vote count will be decremented, and the song will be moved down the queue until a song with a lower vote count is found.

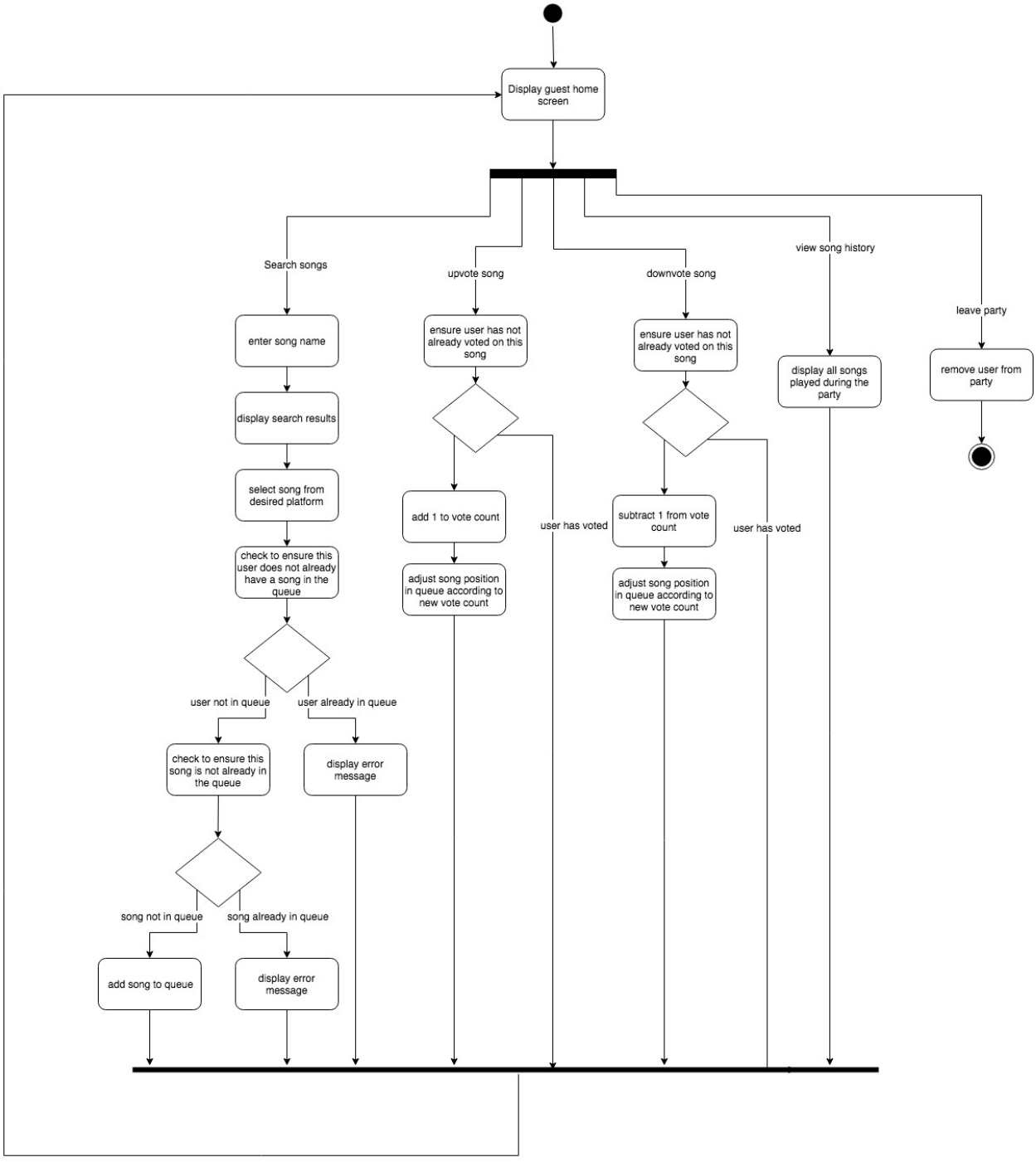
7. View song history

   The user can select the 'View Song History' option to view a list of all the songs that have been played since the party was created.

8. End party

   The host can choose the end a party. This will remove all users from the party and all songs from the queue. All of the song and user information from the party will be lost. Once a party has been ended, the host and guests will be redirected to the splash screen.

7.4-C Guest Home Page Activity Diagram



The above diagram shows the activities available from the guest home screen. The guest has 5 options to choose from, and 4 of these are identical to the hosts options (see 7.4-B description).

1. Search songs - identical to host option 4
2. Upvote song
   Identical to host option 5
3. Downvote song
   Identical to host option 6
4. View song history
   Identical to host option 7
5. Leave party
   The guest may choose to leave a party at any time. When a guest leaves a party, their song choice is removed from the queue and the user is redirected to the splash screen.

7.5 Prospective Database Setup

Users

User_ID	User_Nickname	Host
12345	“lauraphillips”	true

Playlist

Song_URI	User_ID	VoteCount
song_12345	12345	2

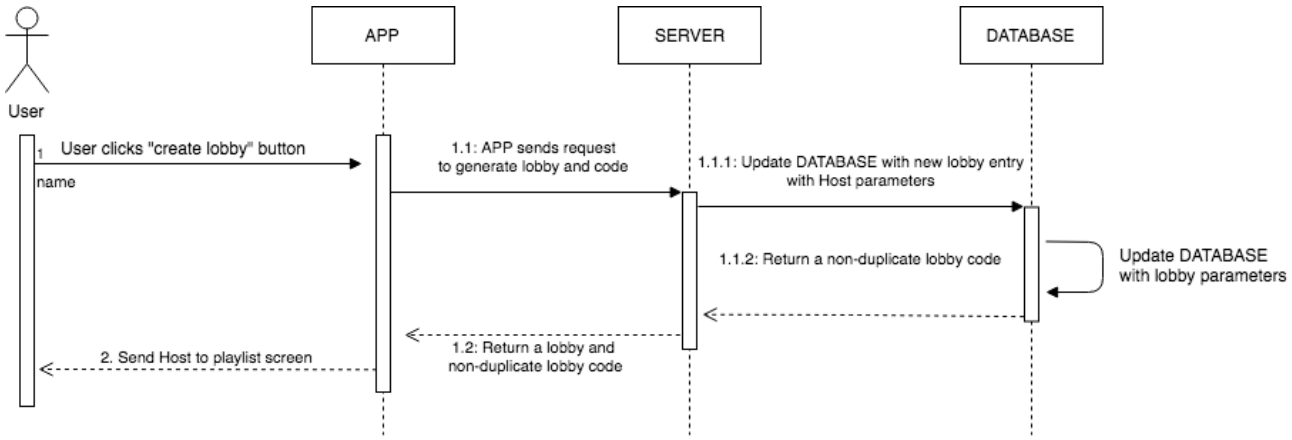
Votes

User_ID	Song_URI	UpOrDown
12345	song_12345	“up”

The above is a basic overview of the SQL database setup for the Juked application. When a host creates a lobby, the above tables will be created. Each time a user joins a party, the server will be contacted and a new user will be generated. The User_ID will be auto-generated and unique. The User_nickname will correspond with the nickname entered by the user. The Host column will be a boolean value identifying whether the user is a host (true) or a guest (false). The playlist database will keep track of the queue. This table will contain an entry for each song added by party guests. The table will record the song URI (provided by the Spotify or Soundcloud API), the User_ID for the user that selected the song, and the voteCount integer. This table will allow the functionality to make sure a user does not add more than one song (by checking for the User_ID), and ensure that the same song is not added twice (by checking the Song_URI). The VoteCount column will keep a running overall score for each song, so that it can be placed in the queue accordingly. The final table keeps records of the votes. The columns here (User_ID, Song_URI, and UpOrDown) ensure that a user can only vote on a song once. The UpOrDown column will allow the user to change their vote once it has

been selected, but will not allow for multiple of the same vote; a user can not continually upvote a song, but they can switch an upvote to a downvote. With all of this information being stored on the Juked server, it can easily be retrieved to achieve the desired functional requirements.

8.0 Sequence Diagrams

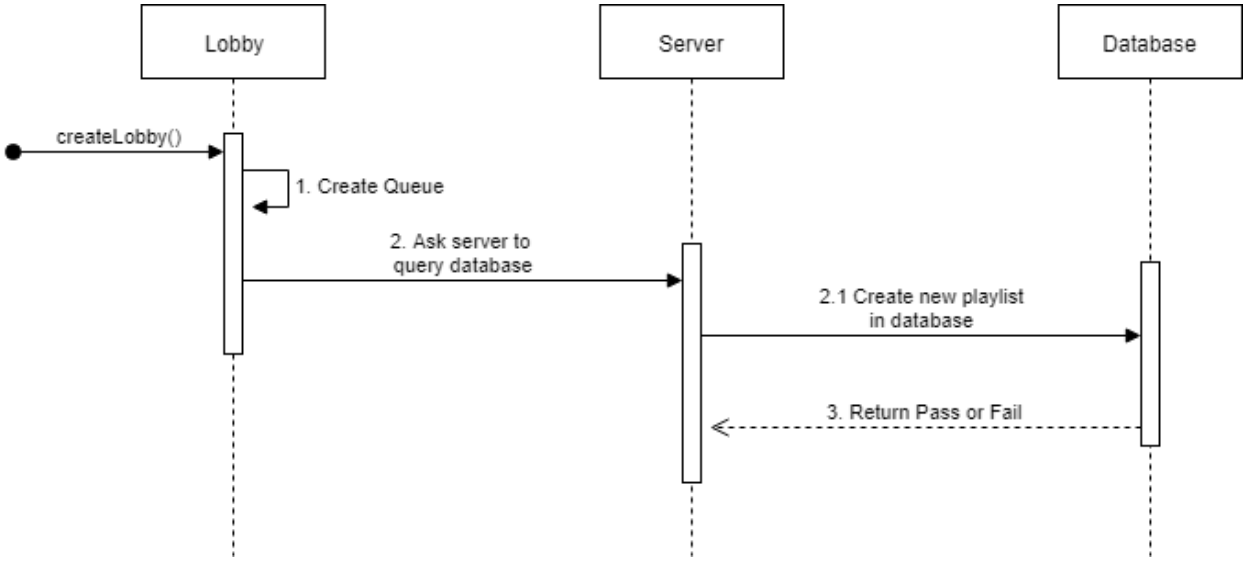


8.1 - Create Lobby

When a user intending to be a **host** clicks the “Create Lobby” button it will in turn send a request to the server to create a lobby/playlist as well as generate and a host object and a unique four digit access code(i.e., 4353) that will enable other users to join the session. The database will ensure that the four digit code that is generated is unique. This ensures that multiple parties will not have the same identifier, and their songs/users will not overlap.

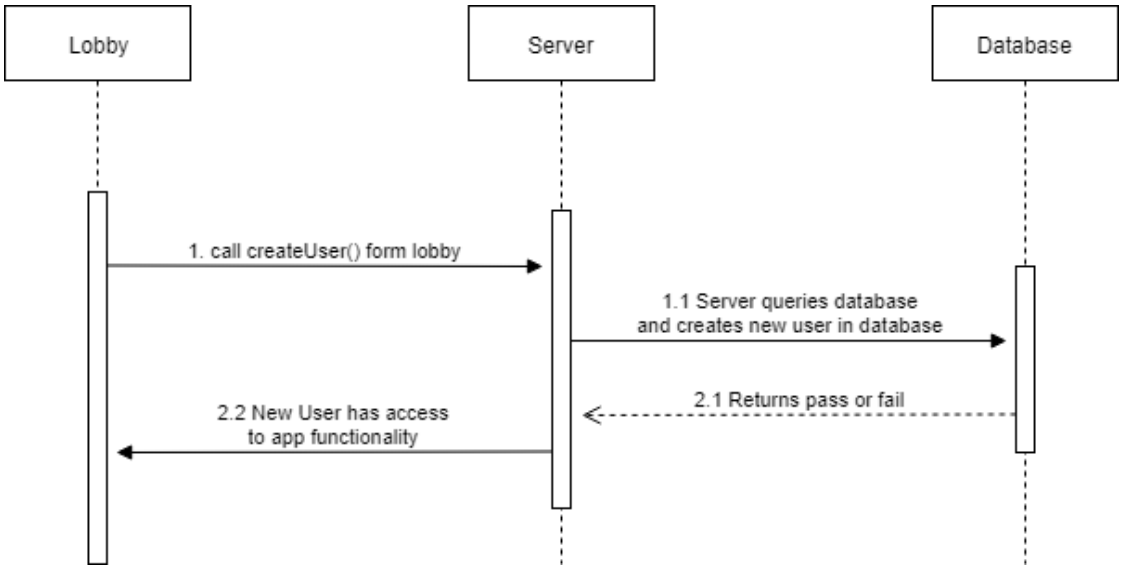
Once a unique code is generated and all lobby parameters(methods, objects, classes, etc) are created without error, the host will then be prompted to enter a username that is associated with their unique deviceID. They will also be prompted to connect their Spotify account.

8.2 - Create Playlist



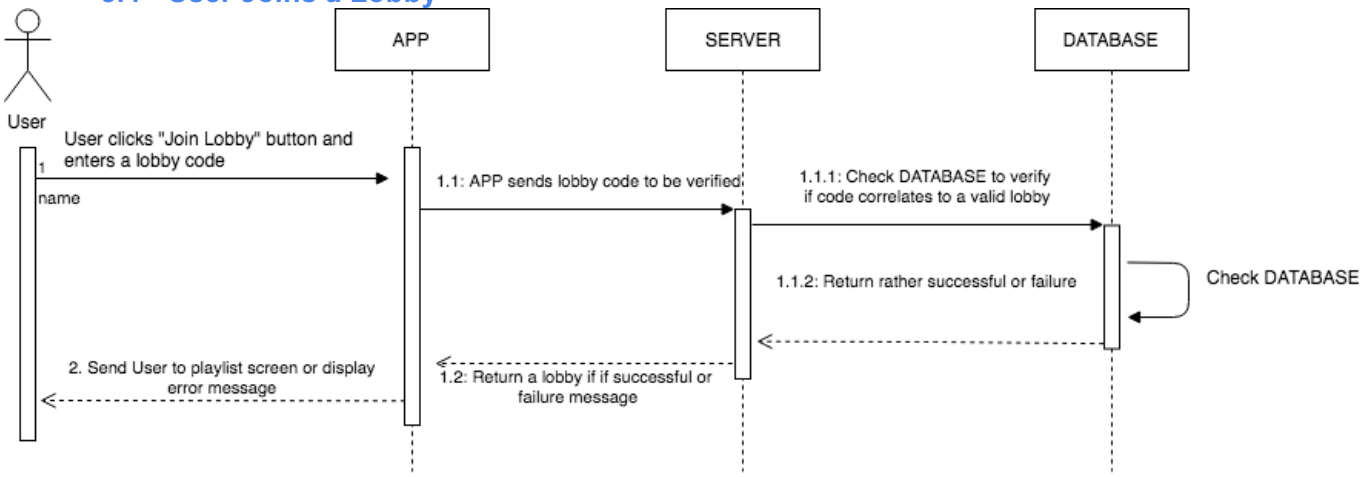
After a user clicks the “Create Lobby” button, a playlist is automatically created on the backend. An ArrayList is created in the program while the server queries the database. The playlist’s ArrayList will store all the songs that have been selected through their respective searches. The ArrayList holds a song object and is populated after each round of song choosing.

8.3 - Create User



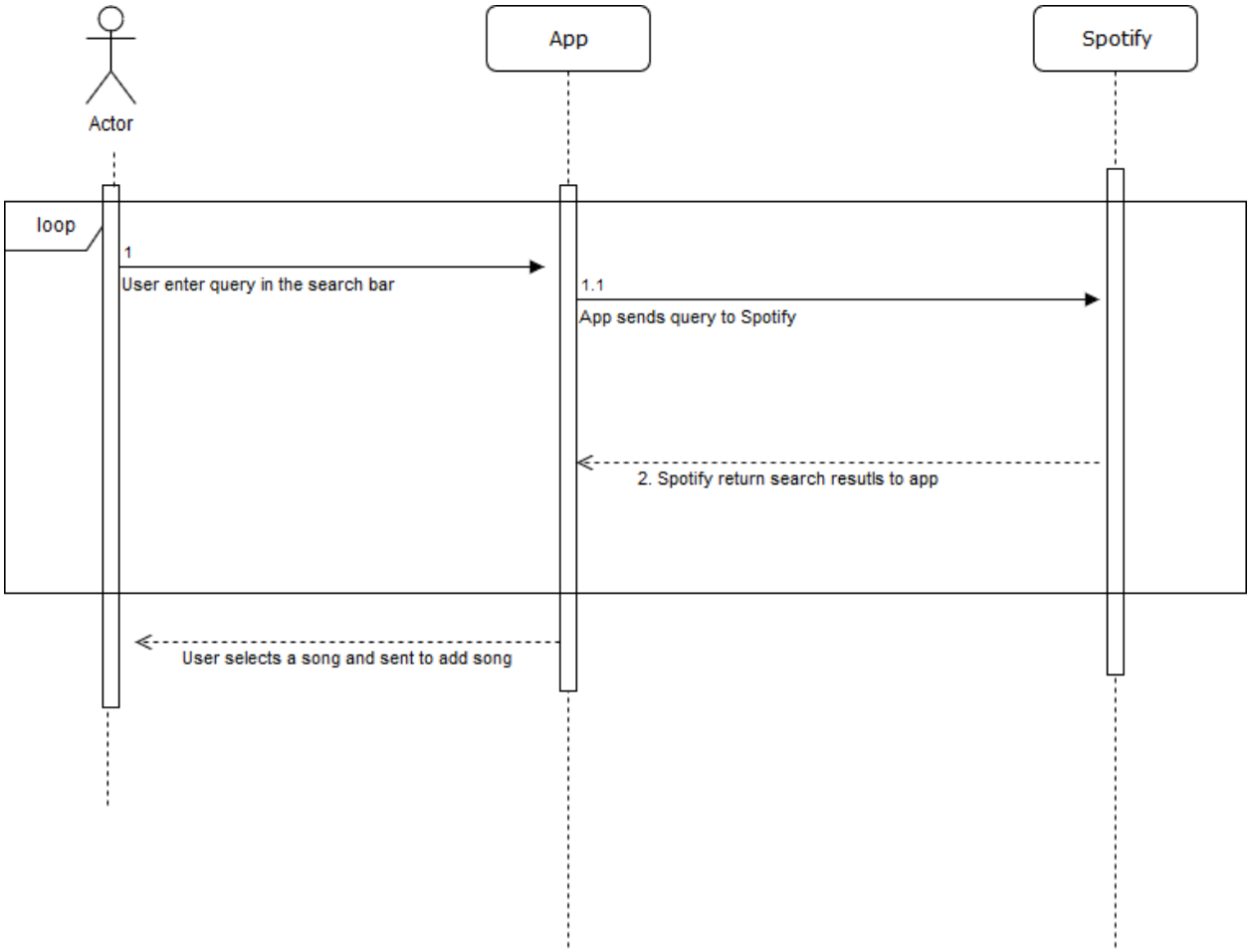
A user object is created when a user clicks the “Join Lobby” button. This in turn invokes the createUser method. This method is passed the “nickname” that was entered by the user, and their device ID. The device ID can be used as a unique identifier for the user. This user is then sent to the server. The server takes this information and adds the user to the lobby.

8.4 - User Joins a Lobby



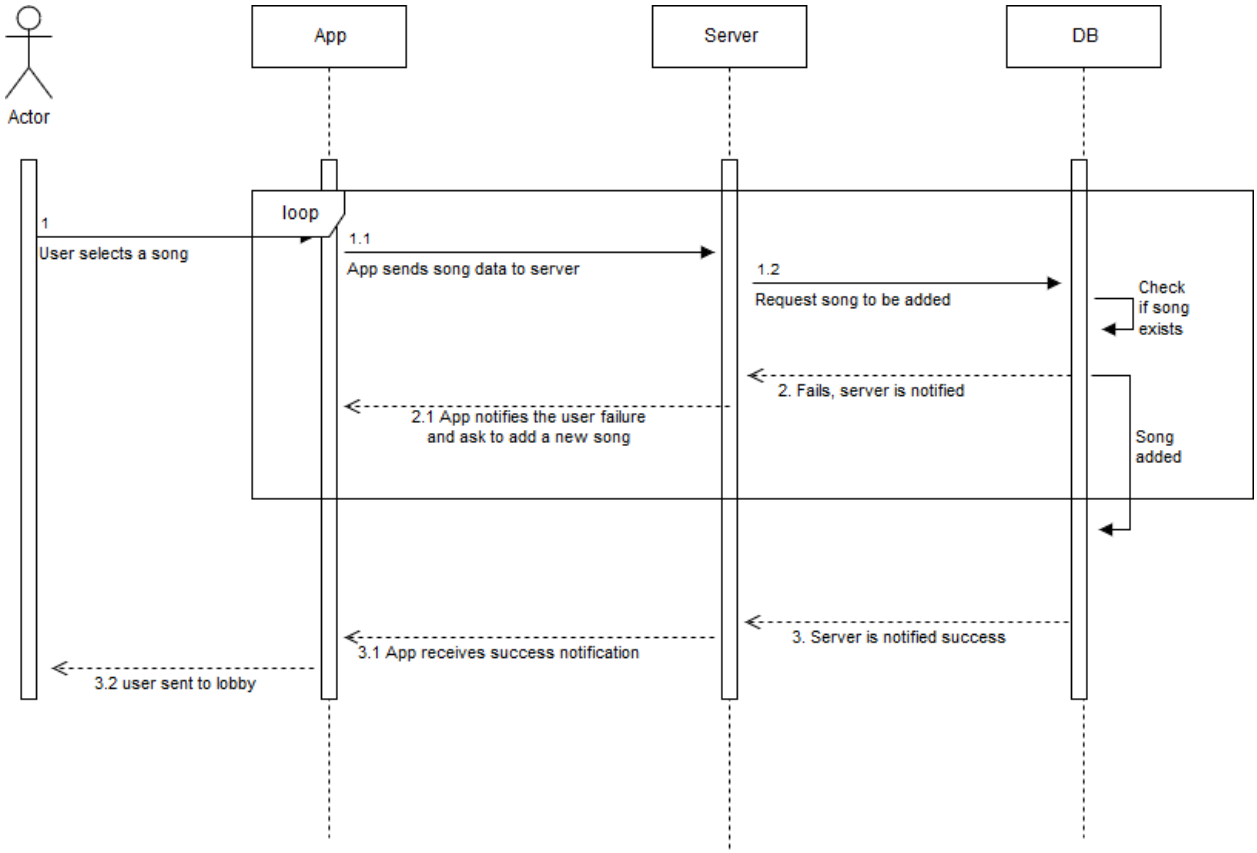
When a user attempts to join a lobby with a provided access code, they will select the “Join Lobby” button. This will then prompt the user to enter the four digit lobby code that they have obtained from the party host. This code is then sent to the server for validation. If such a lobby exists that is associated with the passed lobby code, it will then return as a success. In this case, the application will send the user to the party lobby, also known as the playlist screen. From here, they will then be prompted to enter a username that will be associated with their unique deviceID. Otherwise, an error message will be displayed- more than likely from an invalid lobby access code.

8.5 - Search For Song



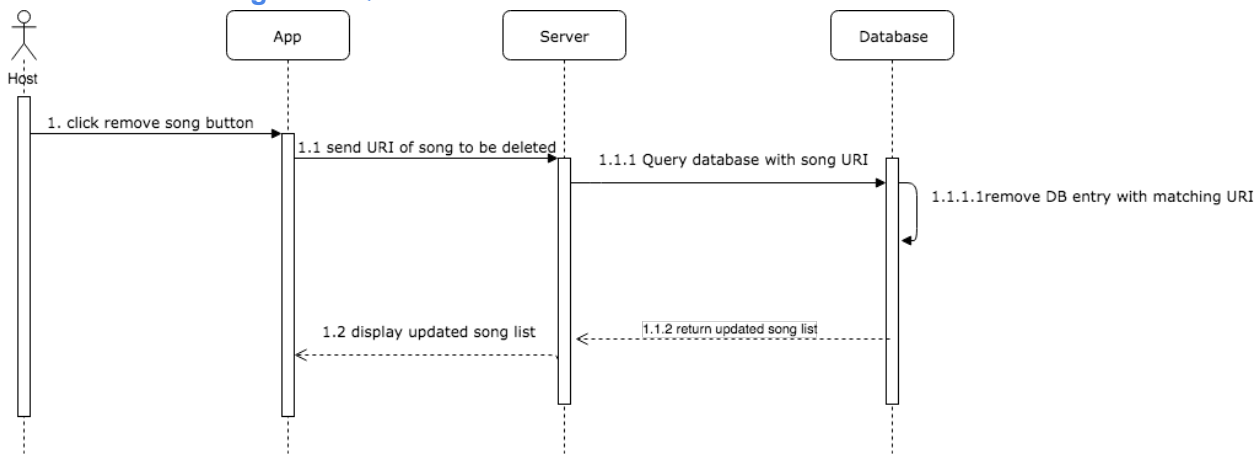
Search for a song: The user clicks the search bar, which pulls up qwerty keyboard. Using the keyboard, user enters query. The app sends this query to Spotify API. Spotify performs the search and the results are displayed on the app's screen. User than may choose a song among the search results, the search loop then terminates. User may choose to perform a new search or exit out of the search all together.

8.6 - Add A Song To the Queue



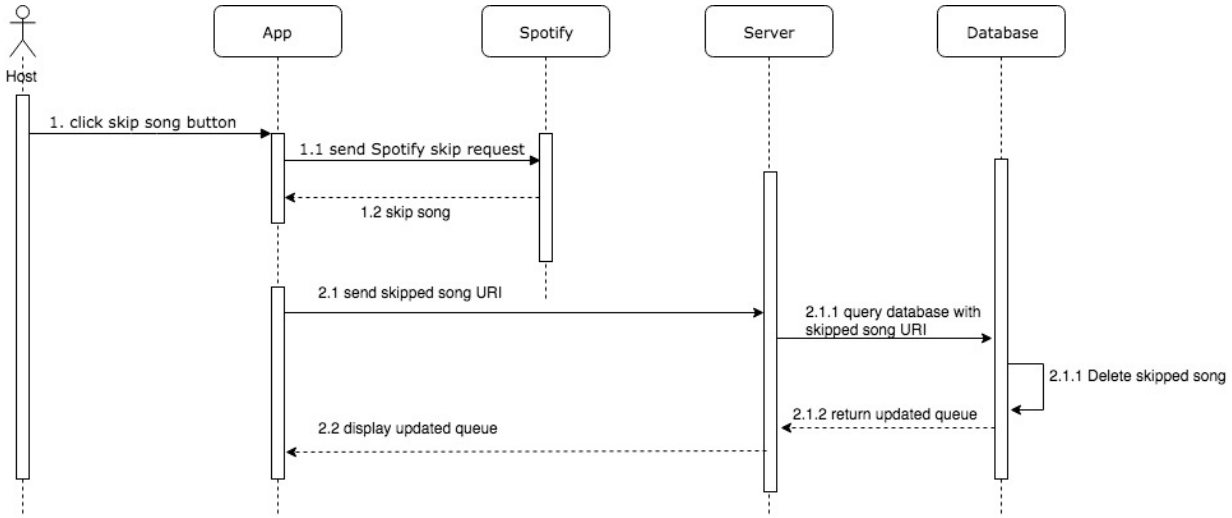
This task is performed at the backend. User is not aware of the process in this sequence. The process starts after the user selects a song. App sends the song to the backend server. Backend server checks the database to ensure the song has not been added previously. If the song exists in the database, then user is asked to select a different song, else the song is placed in the database playlist.

8.7 - Remove Song from Queue



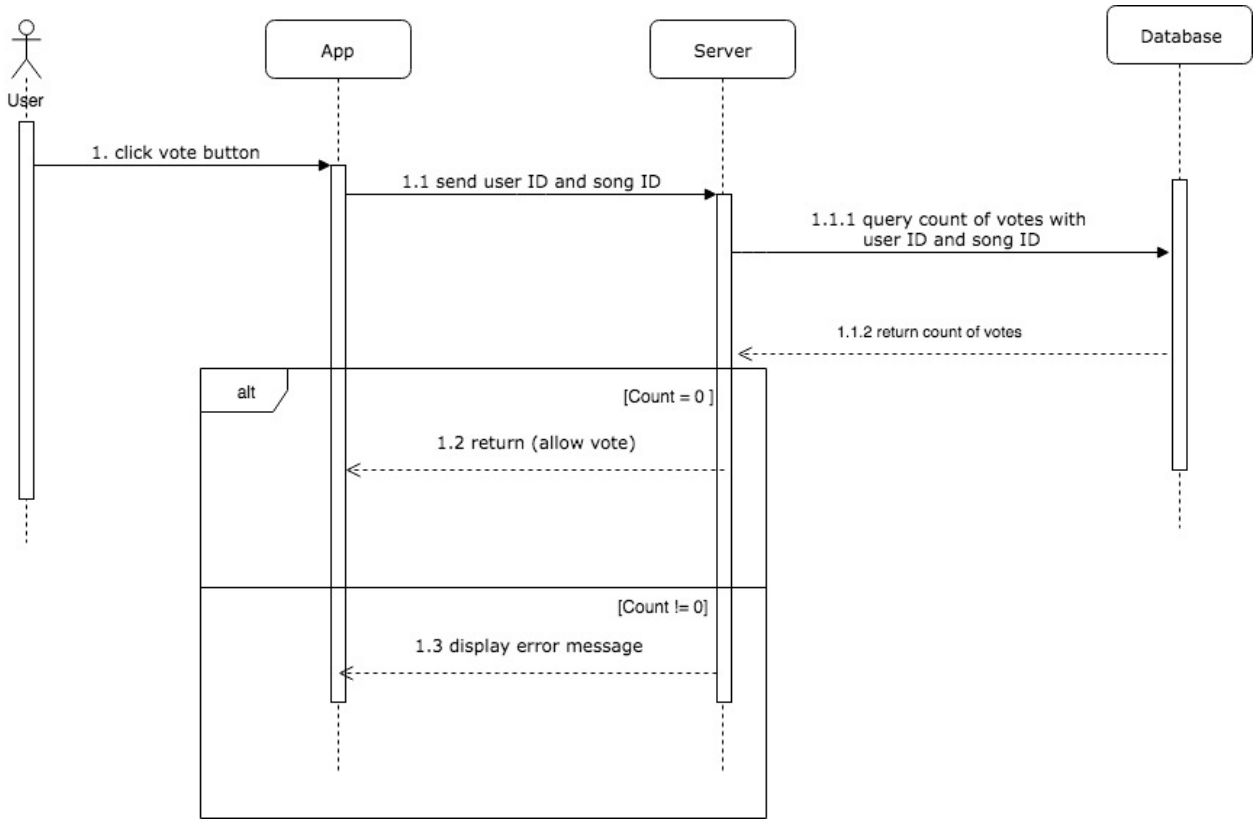
If the host wishes, they have the ability to remove a song from the queue before it plays. The host can click a remove song button for the corresponding song. When this button is pressed, Juked will contact the server with the song identifier. The server will in turn query the database to remove any entries in the song queue matching that song identifier. Once the song has been removed, the server will send the updated queue of songs. The app will then display the updated queue on all devices for users in that lobby, successfully removing the song and preventing it from being played.

8.8 - Host Skips a Song



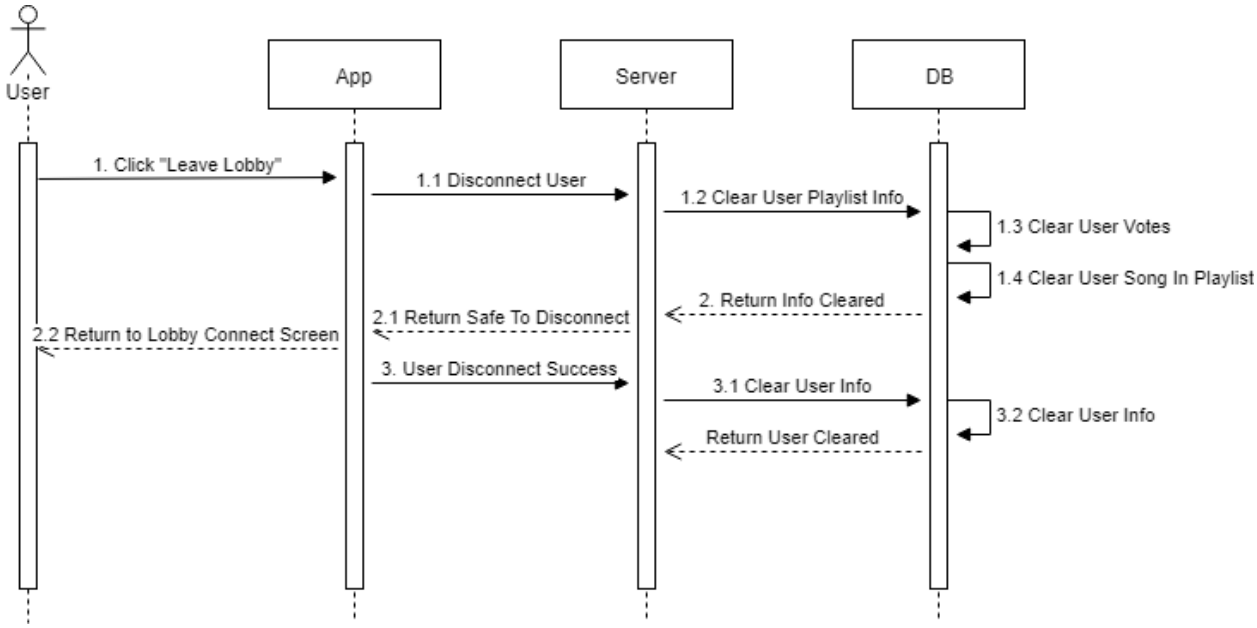
If the host wants to skip the song that is currently playing, the app will contact the NodeJS server with skip request and the song ID. The server will first contact the Spotify app using the API to skip to the next song. The server will then contact the database to remove the skipped song from the queue, using the song's identifier. The database entry matching the skipped song will be deleted, and the updated queue will be sent back to the server. The server will then send the updated song queue to the users in the lobby and display the queue on their devices.

8.9 - Check If User Has Voted



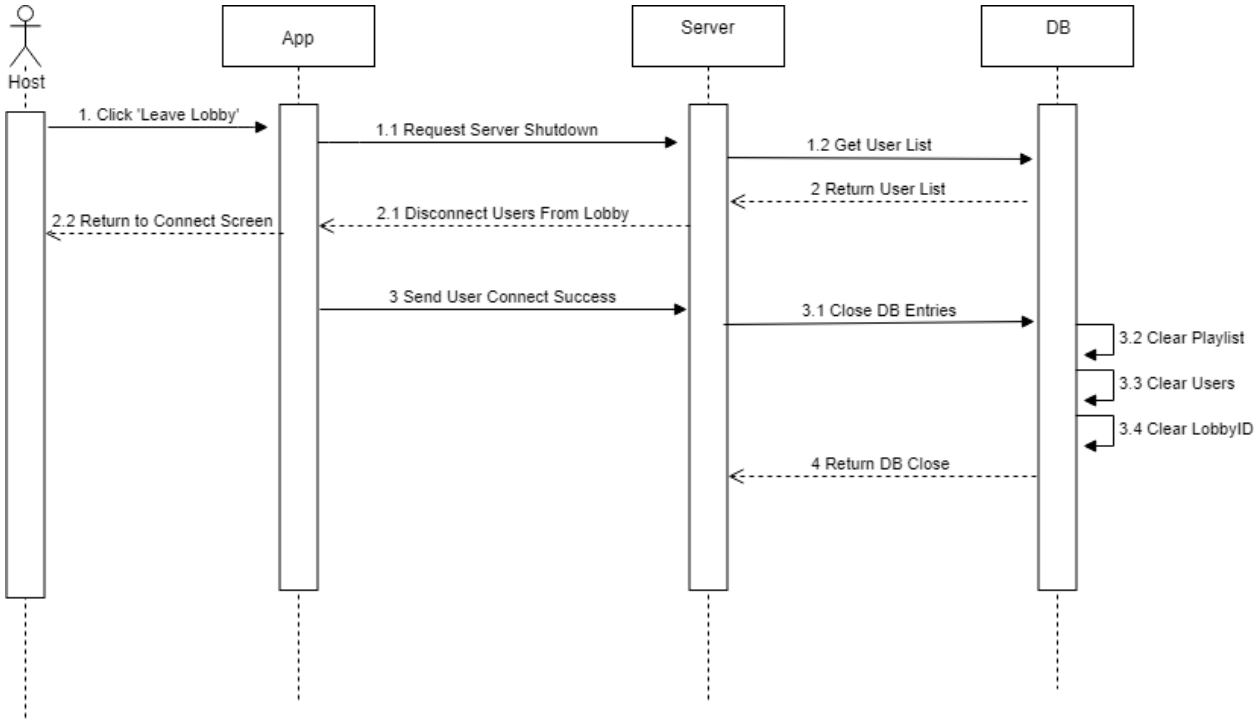
A user should only have the option to vote once per song in the queue. Without this functionality, a user could flood a song with upvotes or downvotes, and the queue wouldn't accurately reflect the choices of the group, but rather that one individual. That being said, when a Juked user clicks the upvote or downvote button for the song, the server is contacted with the song ID and the voter's ID. The server will query the database for a vote that matches the user ID AND song ID. If the database returns an entry, this means that the user has already voted on that specific song, and the application will display an error message. If the database returns no entries, the vote is valid and the vote sequence begins.

8.10 - Leave Lobby



When user clicks leave lobby, the app sends a message to the server to begin the process of disconnecting the user. The server accesses the database and has the database clear the user's playlist information. The playlist information is cleared which consisted of the user's votes for songs and their song in the playlist. Once that information is cleared, the database returns back to the server. The server sends a message to the application that disconnecting is now safe, and so the application disconnects, sending the user back to the lobby screen. Once the user is successfully disconnected, the application notifies the server. The server then has the database clear out the rest of the user information the database. Once that is done, the database returns to the server and the program continues as normal.

8.11 - Destroy Lobby



When the host clicks leave lobby, the app passes the shutdown server request to the server. The server gets the user list from the database. The server then uses that information to pass a disconnect prompt to each user's connected application. Each user is returned to the connection screen. When each user is disconnected, it informs the server. When all users are disconnected, the server has the database close all the open entries which include the playlist, users, and lobbyID. Once that is done, the database informs the server that it is shutdown, freeing the lobbyID for future lobbies.