1.

auth.php

-this is a get request that retrieves the access token needed to communicate with the apps spotify account

Send:

type of request- GET

url- <http://www.playlistparty.live/api/auth.php>

header- “content-type”:“application/json”

Receive:

{

“token”:“access token will be here”

}

2.

createRoom.php

-this will create the room with the necessary information on our database

Send:

Type of request- POST

url- <http://www.playlistparty.live/api/createRoom.php>

header- “content-type” : “application/json”

body-

{

"playlistid":"insert spotify id of playlist here",

"joincode":"insert generated join code here",

"playlistname":"insert user entered playlist name here"

}

Receive:

{

“id”:”this will return the room id, which is needed for most database calls”

}

3.

checkduplicateJoincode.php

-this checks if the generated joincode is already being used

Send:

Type of request- POST

url- <http://www.playlistparty.live/api/checkduplicateJoincode.php>

header- “content-type” : “application/json”

body-

{

"joincode":"insert generated join code here"

}

Receive:

{

“status”:”this will return either 200 or 400. If 200, the joincode can be used, if 400 a different joincode is needed”

}

4.

joinRoom.php

-this will retrieve the room info for a user if the joincode they enter is valid

Send:

Type of request- POST

url- <http://www.playlistparty.live/api/joinRoom.php>

header- “content-type”:“application/json”

body-

{

"joincode":"insert user entered join code here"

}

Receive:

{

“id”:”this will return the room id, which is needed for most database calls”,

“playlistid”:”this will return the spotify id of the rooms playlist”,

“playlistname”:”this will return the name of the playlist”

}

5.

checkduplicatePlaylist.php

-this checks if a song to be added is already in the playlist

Send:

Type of request- POST

url- <http://www.playlistparty.live/api/checkduplicatePlaylist.php>

header- “content-type”:“application/json”

body-

{

"roomid":"the id of the current room here”,

“songid”:”the spotify song id of the song in question”

}

Receive:

{

“status”:”this will return either 200 or 400. If 200, the song is not a duplicate, if 400 the song is a duplicate”

}

6.

checkBlacklist.php

-this checks if a song is in the blacklist

Send:

Type of request- POST

url- <http://www.playlistparty.live/api/checkBlacklist.php>

header- “content-type”:“application/json”

body-

{

"roomid":"the id of the current room here”,

“songid”:”the spotify song id of the song in question”

}

Receive:

{

“status”:”this will return either 200 or 400. If 200, the song is not blacklisted, if 400 the song is blacklisted”

}

7.

addtoPlaylist.php

-this adds a song to our playlist table

Send:

Type of request- POST

url- <http://www.playlistparty.live/api/addtoPlaylist.php>

header- “content-type”:“application/json”

body-

{

"roomid":"the id of the current room here”,

“songid”:”the spotify song id of the song in question”

}

Receive:

{

“status”:”this will return either 200 or 400. If 200, the song has been added, if 400 the song has not (server error)”

}

8.

getplaylistInfo.php

-this retrieves the playlist information from spotify, to fill tables with the rooms playlist info

Send:

Type of request- POST

url- <http://www.playlistparty.live/api/getplaylistInfo.php>

header- “content-type”:“application/json”

body-

{

"token":"access token used for sites spotify account”,

“playlistid”:”the spotify id of the room playlist”

}

Receive:

{

“items”:[ // items is a json array (requires iteration)

“track”:{ //track is a json object (no iteration), sub of items

“album”:{ //album is a json object (no iteration), sub of track

“images”:{ //is a json array, however we will always use index [2]

“url”:”image url here” //url is a json object

}//end images array, sub of album

“name”:”album name here” //name is a json object, sub of album

}//end album object, sub of track

“artists”:{ //is a json array, however will always use index [0]

“name”:”artist name here” //name is a json object, sub of artists

}//end artists array, sub of track

“name”:”song name here” //name is a json object, sub of track

}//end track object, sub of items

]//end items array

}

9.

removefromPlaylist.php

-this removes a song from our playlist table

Send:

Type of request- POST

url- <http://www.playlistparty.live/api/removefromPlaylist.php>

header- “content-type”:“application/json”

body-

{

"roomid":"insert id of room here”,

“songid”:”the spotify id of the song to be removed”

}

Receive:

{

“status”:”this will return either 200 or 400. If 200, the song has been removed, if 400 the song has not (server error)”

}

10.

addtoBlacklist.php

-this adds a song to the blacklist table

Send:

Type of request- POST

url- <http://www.playlistparty.live/api/addtoBlacklist.php>

header- “content-type”:“application/json”

body-

{

"roomid":"insert id of room here”,

“songid”:”the spotify id of the song to be blacklisted”

}

Receive:

{

“status”:”this will return either 200 or 400. If 200, the song has been blacklisted, if 400 the song has not (server error)”

}

11.

closeRoom.php

-this deletes all information in our database for a room

Send:

Type of request- POST

url- <http://www.playlistparty.live/api/addtoBlacklist.php>

header- “content-type”:“application/json”

body-

{

"roomid":"insert id of room here”

}

Receive:

{

“status”:”this will return 200, meaning room has been deleted”

}