HTTP Requests/Routes

GET /user/:userId/account_info, body is undefined
What it does: gets user data of the user associated with userId from the mock database
What resources are created/modified: does not create/modify any resources
Who is authorized to use it: anyone

GET /user/:userId/playlists, body is undefined
What it does: gets playlist array of the user associated with userId from the mock database
What resources are created/modified: does not create/modify any resources
Who is authorized to use it: anyone

POST /createroom/:hostId [roomData], where [roomData] is a JSON object containing the room id of the room to be created
What it does: creates a new room if the room id doesn't already belong to another room, and if the host isn't already hosting another room
What resources are created/modified: Creates a new /room
Who is authorized to use it: anyone

POST /joinroom/:userId [roomData], where [roomData] is a JSON object containing the room id of the room to be joined
What it does: if a room with the room id already exists, it adds the user associated with userId to the room if he/she isn't already in a participant in the room
What resources are created/modified: Updates /room/participants
Who is authorized to use it: anyone

POST /room/data [roomData], where [roomData] is a JSON object containing the room id of the room whose information is being queried
What it does: gets room data (playlist, participants) of the room associated with the room id
What resources are created/modified: Does not create/modify any resources
Who is authorized to use it: anyone

POST /room/save [roomData], where [roomData] is a JSON object containing a room id, the user id of the current logged in user, and the name of the playlist within the room associated with the room id
What it does: creates a new user playlist from the current room playlist for the logged in user
What resources are created/modified: Updates /user/:userId/playlists
Who is authorized to use it: anyone

POST /room/:songId/new_song [roomData], where [roomData] is a JSON object containing the room id of the room in which the song is to be added and user id of the user that is adding the song
What it does: if the song associated with songId is not already in the room associated with room id, the song gets added to the room
What resources are created/modified: Updates /room/playlist
Who is authorized to use it: anyone

POST /room/song_like [roomData], where [roomData] is a JSON object containing a room id, the song id of the song to be liked, and the user id of the user that is liking the song
What it does: if the user associated with the user id hasn't already liked the song associated with song id within the room, then add a like to that song by that user
What resources are created/modified: Updates /room/:songId
Who is authorized to use it: anyone

POST /room/participants [roomData], where [roomData] is a JSON object containing the room id of the room whose participants are being returned
What it does: Returns a list of the participants of the room associated with the room id
What resources are created/modified: Does not create/modify any resources
Who is authorized to use it: anyone

GET /song/:songId, body is undefined
What it does: gets metadata of song associated with songId through the SoundCloud API
What resources are created/modified: Does not create/modify any resources
Who is authorized to use it: anyone

POST /resetdb, body is undefined
What it does: calls the resetDatabase function on the server side database, resetting data back to original state
What resources are created/modified: Does not create/modify any resources
Who is authorized to use it: anyone

DELETE /room/:roomid/participants/:participantid
What it does: remove logged in user from the current room, allowing them to "exit"
What resources are created/modified: Updates /room/participants
Who is authorized to use it: anyone

Individual Contributions

Aarsh Patel

- GET /user/:userId/account_info ⇒ gets user account info
- GET /user/:userId/playlists ⇒ gets user playlists
- POST /createroom/:hostId ⇒ creates new room
- POST /joinroom/:userId ⇒ allows user to join room
- POST /room/data ⇒ gets room data for a particular room
- POST /room/save ⇒ saves room playlist
- POST /room/:songId/new_song ⇒ adds new song to room playlist
- POST /room/song_like ⇒ adds like to a song
- POST /room/participants ⇒ gets all participants of the room
- GET /song/:songId ⇒ gets song metadata for a particular song
- Implemented the search functionality in the RoomPlaylist Component
- Embedded SoundCloud Player Component to play songs for our app

Bhavik Jain

- POST /createroom/:hostId ⇒ creates new room
- JSON Tokens and Authentication

Siddarth Patel

- POST /resetdb ⇒ resets the mock database

Justin Martinelli

- POST /createroom/:hostId ⇒ creates new room

Lynn Samson

- DELETE /room/:roomid/participants/:participantid ⇒ removes logged in user from room

Ronit Arora

Lingering Bugs

- Need to add authorization to some routes
- Sidebar component does not immediately re-render once playlist has been saved, have to refresh the page
- No way for a host to currently delete his/her own room
- Room does not get deleted if all participants exit room
- User is able to join multiple rooms
- Currently just using placeholder values for some song metadata such as “artist” and “album”
- Form on Account Info page does not update the user info
- Songs in user playlist do not automatically play when clicked on
- Songs in room playlist do not automatically play