API Documentation

a. DATABASE STRUCTURE:

```
{
 "users": {
   "user_id_1": {
     "username": "nicknameOrAlias",
     "rooms": {
        "roomID1": true,
        "roomID2": true
     },
   },


   "user_id_2": {
     "username": "ghopper",
     "rooms": {
        "roomID1": true
     }
   },
   "user_id_3": { ... }
 },

 "rooms": {
   "room_id_1": {
     "room_name": "inconspicuous group (real)",
     "room_description": 'asdasfsdfg',
     "last_message": "ghopper: Relay malfunction found. Cause:
moth.",
     "last_message_timestamp": 1459361875666,
     "isPublic": true,
     "themeImageUrl": 'https....',
     "dj": [
       "user_id_1",
       "user_id_2"
     ],
     "users": {
       "user_id_1": {
         "username": "askofsf",
         "owner": true
       },
       "user_id_2": {
         "username": "ghjk"
       },
       "user_id_3": {
         "username": "qwe"
       }
     }
   },
```

```
    "room_id_2": { ... },
    "room_id_3": { ... },
  },


  "messages": {
    "room_id_1": {
      "message_id_1": {
        "user_id": "asdoidjfj",
        "message": "aint no way.",
        "timestamp": 121234557
      },
      "message_id_2": {
        "user_id": "fdgojp",
        "message": "ghopper: Relay malfunction found. Cause: moth.",
        "timestamp": 131239128
      },
    },
    "room_id_2": { ... },
    "room_id_3": { ... }
  },

  "current_track":{
    "room_id_1":{
      "track_id": "12s35345ghjfghj98903dtg21409",
      "time_of_last_played": 12746,
      "is_current_track_playing": true
    },
    "room_id_2": { ... },
    "room_id_3": { ... }
  },

  "room_queue":{
    "room_id_1":[
        "track_id_1",
        "track_id_2",
        "track_id_3"
    ],
    "room_id_2": { ... },
    "room_id_3": { ... }


  "user_queue":{
    "user_id_1":[
        "track_id_1",
        "track_id_2",
        "track_id_3"
    ],
    "user_id_2": { ... },
    "user_id_3": { ... }
  }
}
```

b. CUSTOM SPOTIFY API FUNCTIONS:

Connects our app with the spotify API.

Usage case:

```
const userProfile = await GetCurrentProfile( {accessToken: 12345523} )
```

- `GetCurrentProfile( { accessToken} )` : Returns a JSON object as described in the Spotify developer API docs
- `GetUserPlaylists ( { accessToken, limit = 0, offset = 0} )` : Returns a JSON object as described in the Spotify developer API docs
- `GetPlaylistDetails( {accessToken, playlistID, limit = 20, offset = 0} )` : Returns a JSON Object as described in the Spotify developer API docs
- `GetPlaylistSongs( {accessToken, playlistID, limit = 20, offset = 0} )` : Returns a JSON Object as described in the Spotify developer API docs
- `GetTrack( {accessToken, trackId} )` : Returns a JSON Object as described in the Spotify developer API docs
- `GetRecentlyPlayed( {accessToken} )` : Returns a JSON Object as described in the Spotify developer API docs
- `SearchTrack( {accessToken, text} )` : Returns a JSON Object as described in the Spotify developer API docs
- `GetQueue( {accessToken} )` : Returns a JSON Object as described in the Spotify developer API docs

c. FIREBASE API FUNCTIONS:

Functions to connect our app with our Firebase database

1. **current_track_functions:** functions related to the current_track database

```
current_track_updateCurrentTrack({
   roomID,
   trackURL,
        timeOfLastPlayed,
        isCurrentTrackPlaying,
        songInfo,
}) :
```

Returns a current_track JSON object as described in our database. Updates the current_track data.

`current_track_removeFromRoom( {roomID} )` : Returns a current_track JSON object with roomID: roomID and deletes a current_track data.

`current_track_getCurrentTrack( {roomID} )`: Returns a current_track JSON object playing in the room identified by roomID: roomID

2. **message_functions:** functions related to the chatrooms in our JamRooms

`message_setMessage( {roomID, username, message, timestamp} )`: returns null. Updates the identified message JSON object with the given parameters.

`message_removeAllMessageInRoom( {roomID, username, message, timestamp} )`: Returns null. Removes all the messages in a room

`message_getMessage({roomID})` : Returns a list of sorted messages, sorted by timestamp.

3. **room_functions:** functions related to the JamRooms

```
room_updateRoom( {
     roomID,
     roomName,
     roomDescription,
     themeImageUrl,
     last_message,
     Last_message_timestamp,
     dj,
     isPublic,
     users,
```

```
    isOthersAddSongs,
    Broadcaster,
       etc

} ):
```

Returns a room JSON object as described in our database. This function either updates an existing room data or creates a new one based on the existence of the roomID parameter.

`room_getRoom( { roomID } )` : returns a room JSON object as described in our database

`room_fetchDJList( { roomID } )` : returns an array containing the usernames of all DJs in a given room

`room_fetchUserList( { roomID } )` : returns an array containing the usernames of all listeners in a given room

`room_removeRoom({ roomID })` : returns null.

`room_addUser( { roomID } )` : returns a room JSON object as described in our database with an updated user list, current user added

`room_updateDJ({ roomID, djArray })` : returns null. Updates the dj list array.

`room_removeUser( { roomID } )` : returns a room JSON object as described in our database with an updated user list, current user removed

`room_getAllRooms()` : returns all the rooms in the database

`room_checkIfOwner({ roomID, userID })` : returns a bool value. If the userID is the owner, return true, else return false.

`room_changeBroadcaster({ roomID, userID })` : returns null. Change the broadcaster data in the firebase database

4. user_functions

`user_updateUser( {userID, username, roomsObjects} )`: returns null. Updates a user JSON object identified by userID: userID with the given parameters.

`user_getRooms( {userID} )`: returns a list of room JSON objects identified by their roomIDs.

`user_getUsername( {userID} )`: returns a string containing the username parameter of the user JSON object identified by the given userID.

`user_addToRoom( {userID, arrayOfRoomIDs} )`: returns null. Updates room JSON objects identified by the arrayOfRoomIDs to add current user to given rooms.

`user_removeFromRooms( {userID, arrayOfRoomIDs} )`: returns null. Updates room JSON objects identified by the arrayOfRoomIDs to remove current user from given rooms.

5. user_queue_functions

`userQueue_updateQueue({userID, userQueueList = null})`: returns null. If the userID parameter exists, updates the existing user_queue, else create a new user_queue entry.

`userQueue_updateRoomQueue({roomID, userRoomQueueList = null})` returns null. If the roomID parameter exists, update the existing room_queue, else create a new room_queue entry.

`userQueue_getQueue({userID})` : Returns a list of the user_queue

`userQueue_getRoomQueue({roomID}): Returns a list on of the room_queue`

`userQueue_removeQueue({userID}): Returns null. Removes the whole user_queue.`

6. listeners

Listener functions provide a way to automatically update our local database when the remote database is updated.

usage case:

```
const listener = useRoomTrackListener(roomID)
```

`useRoomTrackListener(roomID)` : returns a JSON object of current_track data

`useRoomSongInfoListener(roomID)` : returns a JSON object of the song_info data

`useUserCurrentQueue(userID)`: returns a JSON object of the user_queue data

`useMessageListener(roomID)`: returns a list of all the messages in a room

`useRoomListener(roomID)`: returns a JSON object of the room data

`useTimeOfLastPlayedListener(roomID)`: returns an integer of the time_of_last_played data of a specific room

`useIsCurrentTrackPlayingListener(roomID)`: returns a boolean of whether the song is playing inside the room

`useRoomCurrentQueue(roomID)`: returns an array of current room's queue

`useRoomBroadcasterListener(roomID)`: returns a userID of the current broadcaster in the room