Serena Zafiris, Donovan Moini, Ian Lizarda, Ben Kern, Alexia Filler
Dr. Johnson
CMSI 401
November 13th, 2019

Software Design Description

TABLE OF CONTENTS

6.3. CSC and CSU Descriptions

6.3.1 Class Descriptions

The following sections provide the details of all classes used in the Amplify application.

6.3.1.1 Frontend Components

6.3.1.1.1 PlayerView Component: Creates the GUI for the room which contains the queue of songs in the room. The rest of the screen has the video player at the top of the screen and a list of users in the room which can be shown when the user icon is clicked.

6.3.1.1.2 CreateJoinView Component: Creates the visual that allows a user to create or join a room. The screen activates when a user opens the app. The component is a split screen where one side when clicked will prompt the user for a room code to join a room and the other will prompt the user for the necessary information to create a room.

6.3.1.1.3 CreateUserView Component: Allows a user to create a profile by prompting them to input their first name, last name, and email into the respective text boxes.

6.3.1.1.4 LoadingView Component: Loading Screen for transitions between components which shows the Amplify logo on a background of the color gradient used throughout the app.

6.3.1.1.5 SearchView Component: Search screen for songs which contains an input box for the users search query and then return a list of the appropriate songs in the default styling for songs shown in the PlayerView component.

6.3.1.1.6 ListView Component: List structure for songs that has the album on the far left with a circle in the bottom left of it that contain the initials of the user that added them. The album art is followed by three rows next to it which contain the song name, artists, and album name respectively.

6.3.2.1 Backend Components

6.3.1.2.1 searchTracks Endpoint: Takes in a search query and search type as inputs. Using the search terms, the available tracks are searched and filter for the appropriate search query and type. Once filtered, a list of appropriate tracks are returned.

6.3.1.2.2 getUser Endpoint: Takes in a user ID as an input. The user ID is validated to check if a user with that user ID exists. If a user with that user ID does not exist, an error is thrown. If a user with that user ID exists, that user is returned.

6.3.1.2.3 getRoom Endpoint: Takes in a room ID as an input. The room ID is validated to check if a room exists with that room ID. If a room with that room ID does not exist, an error is thrown. If a room with that room ID exists, that room is returned.

6.3.1.2.4 getRoomUsers Endpoint: Takes in a room ID as an input. The room ID is validated to check if a room exists with that room ID. If a room with that room ID does not exist, an error is thrown. If a room with that room ID exists, a list of that room's users is returned.

6.3.1.2.5 getUserCurrentRoom Endpoint: Takes in a user ID as an input. The user ID is validated to check if a user with that user ID exists. If a user with that user ID does not exist, an error is thrown. If a user with that user ID exists, the user ID is validated to see if the user with that user ID is in a room. If that user is not in a room, an error is thrown. If that user is in a room, that room is returned.

6.3.1.2.6 addTrack Endpoint: Takes in a user ID and a track input as inputs. The user ID is validated to check if a user with that user ID exists. If a user with that user ID does not exist, an error is thrown. If a user with that user ID exists, the user ID is validated to see if the user with that user ID is in a room. If that user is not in a room, an error is thrown. If that user is in a room, the endpoint retrieves the queue with the given user ID. Once found, the track input is added to the user's queue with the additional properties of which user added the track and the time the track was added. The updated queue is returned.

6.3.1.2.7 createUser Endpoint: Takes in a user ID, first name, and last name as inputs. The user ID is validated to check that another

user with the inputted user ID does not exist. If another user with the inputted user ID exists, an error is thrown. If another user with the inputted user ID does not exist, a new user is created with the inputted user ID, first name, and last name. The new user is returned.

6.3.1.2.8 createRoom Endpoint: Takes in a user ID and name as inputs. The user ID is validated to check that a user with that user ID does not exist. If a user with that user ID exists, an error is thrown. If a user with that user ID does not exist, then the user ID is validated to check if that user is not currently in a room. If the user is currently in a room, an error is thrown. If the user is not currently in a room, a unique four character room ID is generated. A new room is created with the generated room ID and inputted name. The user's currentRoom property is assigned the room ID. The newly created room is returned.

6.3.1.2.9 deleteRoom Endpoint: Takes in a room ID as an input. The room ID is validated to check if a room with that room ID exists. If a room with that room ID does not exist, and error is thrown. If a room with that room ID exists, it iterates through all the users in that room and sets their currentRoom property to the empty string. The room is then deleted and the endpoint returns true to indicate the successful deletion of the specified room.

6.3.1.2.10 joinRoom Endpoint: Takes in a user ID and room ID as inputs. The user ID is validated to check if a user with that user ID does exist. If a user with that user ID does not exist, an error is thrown. If a user with that user ID exists, it checks to validate if that user is currently not in a room. If that user is currently in a room, an error is thrown. If that user is not currently in a room, that room ID is validated to check if a room with that room ID exists. If a room with that room ID does not exist, an error is thrown. If a room with that room ID does exist, the endpoint sets that user's currentRoom property to the room ID and adds the user to the room's users property. The room is then returned.

6.3.1.2.11 leaveRoom Endpoint: Takes in a user ID and room ID as inputs. The user ID is validated to check if a user with that user ID does exist. If a user with that user ID does not exist, an error is thrown. If a user with that user ID exists, it checks to validate if that user is currently in a room. If that user is currently not in a room, an error is thrown. If that user is currently in a room, the endpoint gets the user's current

room. If that user is the room owner, the room is deleted by calling the deleteRoom endpoint. If that user is not the room owner, that user's currentRoom property is set to the empty string and the room removes that user from it users property. The room is then returned.

6.3.1.2.12 createUserRoomInfo Endpoint: Takes in a user ID as an input. The user ID is validated to check that the user exists. If a user with that user ID does not exist, an error is thrown. If a user with that user ID exists, a UserRoomInfo object is created from the inputted user ID. The newly created UserRoomInfo object is returned.

6.3.1.2.13 Database: Hosts the data for the Amplify App through Amazon RDS

6.3.2 Detailed Interface Descriptions

6.3.2.1 Frontend Components

6.3.2.1.1 PlayerView Component: Calls the SearchView and ListView components in order to create the search for songs screen and the list of songs that propagate on the player screen itself and the search screen.

6.3.2.1.2 CreateJoinView Component: Once the user decides to either create or join a room, this will route the user to the PlayerView component which will construct the desired room.

6.3.2.1.3 CreateUserView Component: Once the user is created, this component calls the CreateJoin room component and route the user to the CreateJoin screen.

6.3.2.1.4 LoadingView Component: This component is called by either the PlayerView, CreateJoinView, or the CreateUserView if the backend needs extra time to access and communicate with the database.

6.3.2.1.5 SearchView Component: This component is called by the PlayerView component when the user wants to search for a song. This component will call the PlayerView component when the user returns to the screen by pressing the back button in the top left corner of the screen.

6.3.2.1.6 ListView Component: This component is called by the PlayerView component and the SearchView component when a list of songs is returned from the backend.

6.3.2.1 BackendComponents

6.3.2.2.1 searchTrack EndPoint: This endpoint is called with two strings, a searchType and a searchTerm. It returns all tracks whose searchType property field contains the searchTerm.

6.3.2.2.2 getUser Endpoint: This endpoint is called with a single string: the userID. It returns the user object that has the specified userID. From this user object, the user's first name, last name, and current room are accessible.

6.3.2.2.3 getUserCurrentRoom Endpoint: This endpoint is called with a single string: the userID. It returns just the current room that the user with the matching userID is in.

6.3.2.2.4 getRoom Endpoint: This endpoint is called with a single string: a roomID. It returns the room object with the matched roomID. This room object can be used to call the room name, the users in the room, the tracks queued in the room, and the owner of the room.

6.3.2.2.6 addTrack Endpoint: This endpoint is called with a userID string and a trackinput object, which is a filtered form of telling the backend information about the track. It adds a track object to the current room of the user that matches the userID, and returns the current room's queue of tracks.

6.3.2.2.7 createUser Endpoint: This endpoint is called with a userID (Initialized by the frontend with a MAC address), a firstName string, and a lastName string. It creates a user with the information, and sets the user's current room to be blank.

6.3.2.2.8 createRoom Endpoint: This endpoint is called with a userID string, and a roomName string. It creates a room with a random roomID, the specified roomName, and puts the user with a matching userID field into the room, as well as sets that user as the owner of the room.

6.3.2.2.9 joinRoom Endpoint: This endpoint is called with a userID string and a roomID string. It puts the user associated with the userID into the room associated with the roomID.

6.3.2.2.10 leaveRoom Endpoint: This endpoint is called with a userID string. It kicks the user associated with the userID out of its current room. If the user is the owner of the room, it deletes the room.

6.3.2.2.11 createUserRoomInfo Endpoint: This endpoint is called by createRoom and the joinRoom endpoints to add a user to the room associated with the roomID in the argument.

6.3.2.2.12 Database: The Database is called by all the endpoints so they may receive information.

6.3.3 Detailed Data Structure Descriptions

6.3.3.1 Frontend Components (including but not limited to)

6.3.3.1.1 PlayerView Component - Contains arrays and objects that contain the room information, the user information, and the queue and track information.

6.3.3.1.2 CreateJoinView Component - Contains arrays and objects that the room information is propagated into when a user creates a room. It also contains an object containing the user information.

6.3.3.1.3 CreateUserView Component - Contains arrays and objects that store the users information when the user is created.

6.3.3.1.4 LoadingView Component - Contains an object that holds the information for the loading screen.

6.3.3.1.5 SearchView Component - Contains arrays and objects that host the data for the search query, and for all the returned tracks for that specified query.

6.3.3.1.6 ListView Component - Contains object that holds the information of all the different tracks in the queue.

6.3.3.2 Backend Components (including but not limited to)

6.3.3.2.1 searchTracks Endpoint

- Input:
  - searchTracks(searchQuery: String! searchType: String!): [Track!]!
  - In this example, searchQuery: “Camila Cabello” searchType: “artist”
- Response (JSON):


```
{
  "data": {
    "searchTracks": [
      {
        "provider": "YouTube",
        "providerID": "6-0vO8ZuW98",
        "name": "Liar",
        "artists": [
          "Camila Cabello"
        ],
        "album": "Liar - Single",
        "cover":
"https://img.youtube.com/vi/6-0vO8ZuW98/mqdefault.jpg"
,
        “addedBy” : {
          “ userID”: "193759372",
          “firstName”: "Ian",
          “lastName”: "Lizarda",
```

```
            "currentRoom": "ABCD",
          },
        "timeAdded": 1571874057700
      }
     ]
    }
   }
```

6.3.3.2.2 getUser Endpoint

- Input:
  - getUser(userID: String!): User!
  - In this example, userID: “123456789”
- Response (JSON):

```
{
 "data": {
  "getUser": {
   "userID": "123456789",
   "firstName": "Donovan",
   "lastName": "Moini",
   "currentRoom": "ABCD"
  }
 }
}
```

6.3.3.2.3 getRoomUsers Endpoint

- Input:
  - getRoomUsers(roomID: String!): [User!]!
  - In this example, roomID: “ABCD”
- Response (JSON):

```
{
 "data": {
  "getRoomUsers": [
   {
    "userID": "123456789",
    "firstName": "Donovan",
    "lastName": "Moini",
    "currentRoom": "ABCD"
   },
   {
    "userID": "193759372",
    "firstName": "Ian",
    "lastName": "Lizarda",
    "currentRoom": "ABCD"
   },
   {
```

```
        "userID": "987654321",
        "firstName": "Masao",
        "lastName": "Kitamura",
        "currentRoom": "ABCD"
       }
      ]
     }
    }
```


6.3.3.2.4 getUserCurrentRoom Endpoint

- Input:
  - getUserCurrentRoom(userID: String!): Room!
  - In this example, userID: “123456789”
- Response (JSON):


```
{
 "data": {
  "getUserCurrentRoom": {
   "roomID": "ABCD",
   "name": "Cool Room",
   "users": [
    {
     "userID": "123456789",
     "queue": [
      {
       "provider": "Spotify",
       "providerID":
"spotify:track:4n7jnSxVLd8QioibtTDBDq",
       "name": "On My Way",
       "artists": [
        "Alan Walker",
        "Sabrina Carpenter",
        "Farruko"
       ]
      }
     ]
    },
    {
     "userID": "193759372",
     "queue": [
      {
       "provider": "YouTube",
       "providerID": "qolmz4FlnZ0",
       "name": "Doin' Time",
       "artists": [
```

```
          "Lana Del Rey"
         ]
        },
        {
         "provider": "YouTube",
         "providerID": "6-OvO8ZuW98",
         "name": "Liar",
         "artists": [
          "Camila Cabello"
         ]
        },
        {
         "provider": "YouTube",
         "providerID": "tvTRZJ-4EyI",
         "name": "Humble",
         "artists": [
          "Kendrick Lamar"
         ]
        },
        {
         "provider": "Spotify",
         "providerID":
"spotify:track:4uTvPEr01pjTbZgl7jcKBD",
         "name": "NASA",
         "artists": [
          "Ariana Grande"
         ]
        }
       ]
      },
      {
       "userID": "987654321",
       "queue": [
        {
         "provider": "Spotify",
         "providerID":
"spotify:track:39LmTF9RgyakzSYX8txrow",
         "name": "imagine",
         "artists": [
          "Ariana Grande"
         ]
        },
        {
```

```
            "provider": "Spotify",
            "providerID":
"spotify:track:1TEL6MlSSVLSdhOSddidlJ",
            "name": "needy",
            "artists": [
              "Ariana Grande"
            ]
          }
        ]
      }
    ],
    "tracks": [
      {
        "provider": "Spotify",
        "providerID":
"spotify:track:4n7jnSxVLd8QioibtTDBDq",
        "name": "On My Way",
        "artists": [
          "Alan Walker",
          "Sabrina Carpenter",
          "Farruko"
        ]
      },
      {
        "provider": "YouTube",
        "providerID": "qolmz4FlnZ0",
        "name": "Doin' Time",
        "artists": [
          "Lana Del Rey"
        ]
      },
      {
        "provider": "Spotify",
        "providerID":
"spotify:track:39LmTF9RgyakzSYX8txrow",
        "name": "imagine",
        "artists": [
          "Ariana Grande"
        ]
      },
      {
        "provider": "YouTube",
        "providerID": "6-0vO8ZuW98",
```

```
          "name": "Liar",
          "artists": [
            "Camila Cabello"
          ]
        },
        {
          "provider": "Spotify",
          "providerID": "spotify:track:1TEL6MlSSVLSdhOSddidlJ",
          "name": "needy",
          "artists": [
            "Ariana Grande"
          ]
        },
        {
          "provider": "YouTube",
          "providerID": "tvTRZJ-4EyI",
          "name": "Humble",
          "artists": [
            "Kendrick Lamar"
          ]
        },
        {
          "provider": "Spotify",
          "providerID": "spotify:track:4uTvPEr01pjTbZgl7jcKBD",
          "name": "NASA",
          "artists": [
            "Ariana Grande"
          ]
        }
      ],
      "owner": {
        "userID": "123456789",
        "firstName": "Donovan",
        "lastName": "Moini"
      }
    }
  }
}
```


6.3.3.2.5 deleteRoom Endpoint

- Input:
    - deleteRoom(roomID: String!): Boolean
    - In this example, roomID: “ABCD”
- Response (JSON):

```
{
 "data": {
  "deleteRoom": true
 }
}
```


6.3.3.2.6 addTrack Endpoint

- Input:
  - addTrack(userID: String! track: TrackInput!): [Track!]!
  - In this example, userID: “123456789” track: {provider: "Spotify", providerID: "10283912A12", name: "TEST_SONG", artists: ["Donovan", "LIZZO"], album: "ALBUM_NAME", cover: "COVER_URL"}
- Response (JSON):


```
{
 "data": {
  "addTrack": [
   {
    "provider": "Spotify",
    "providerID": "spotify:track:4n7jnSxVLd8QioibtTDBDq",
    "name": "On My Way",
    "artists": [
     "Alan Walker",
     "Sabrina Carpenter",
     "Farruko"
    ],
    "album": "On My Way - Single",
    "cover":
"https://upload.wikimedia.org/wikipedia/en/a/af/Alan_Walker_-_On_My_Way.png",
    "addedBy": {
     "userID": "123456789",
     "firstName": "Donovan",
     "lastName": "Moini",
     "currentRoom": "ABCD"
    },
    "timeAdded": 1571874057631
   },
   {
    "provider": "Spotify",
    "providerID": "10283912A12",
    "name": "TEST_SONG",
    "artists": [
     "Donovan",
```

```
        "LIZZO"
       ],
       "album": "ALBUM_NAME",
       "cover": "COVER_URL",
       "addedBy": {
        "userID": "123456789",
        "firstName": "Donovan",
        "lastName": "Moini",
        "currentRoom": "ABCD"
       },
       "timeAdded": 1573684182473
      }
     ]
    }
   }
```


6.3.3.2.7 createUser Endpoint

- Input
  - createUser(userID: String! firstName: String! lastName: String!): User!
  - In this example, userID: “123454321” firstName: “Alexia” lastName: “Filler
- Response (JSON)


```
{
 "data": {
  "createUser": {
   "userID": "123454321",
   "firstName": "Alexia",
   "lastName": "Filler",
   "currentRoom": ""
  }
```


6.3.3.2.8 createRoom Endpoint

- Input
  - createRoom(userID: String! name: String!): Room!
  - In this example, userID: “123456789” name: “best room ever”
- Response (JSON)


```
{
 "data": {
  "createRoom": {
   "roomID": "FJ9R",
   "name": "best room ever",
   "owner": {
```

```
        "userID": "123456789"
      }
     }
    }
   }
```


6.3.3.2.9 joinRoom Endpoint

- Input
  - joinRoom(userID: String! roomID: String!): Room!
  - In this example, userID: “123454321” roomID: “FJ9R”
- Response (JSON)


```
{
 "data": {
  "joinRoom": {
   "roomID": "FJ9R",
   "name": "best room ever",
   "users": [
    {
     "userID": "123456789"
    },
    {
      “userID”: “123454321”
    }
   ],
   "owner": {
    "userID": "123456789"
   }
  }
 }
}
```


6.3.3.2.10 leaveRoom Endpoint

- Input
  - leaveRoom(userID: String!): Room
  - In this example, userID: “123454321” roomID: “FJ9R”
- Response (JSON)


```
{
 "data": {
  "leaveRoom": {
   "roomID": "FJ9R",
   "name": "best room ever",
   "users": [
    {
     "userID": "123456789"
    }
```

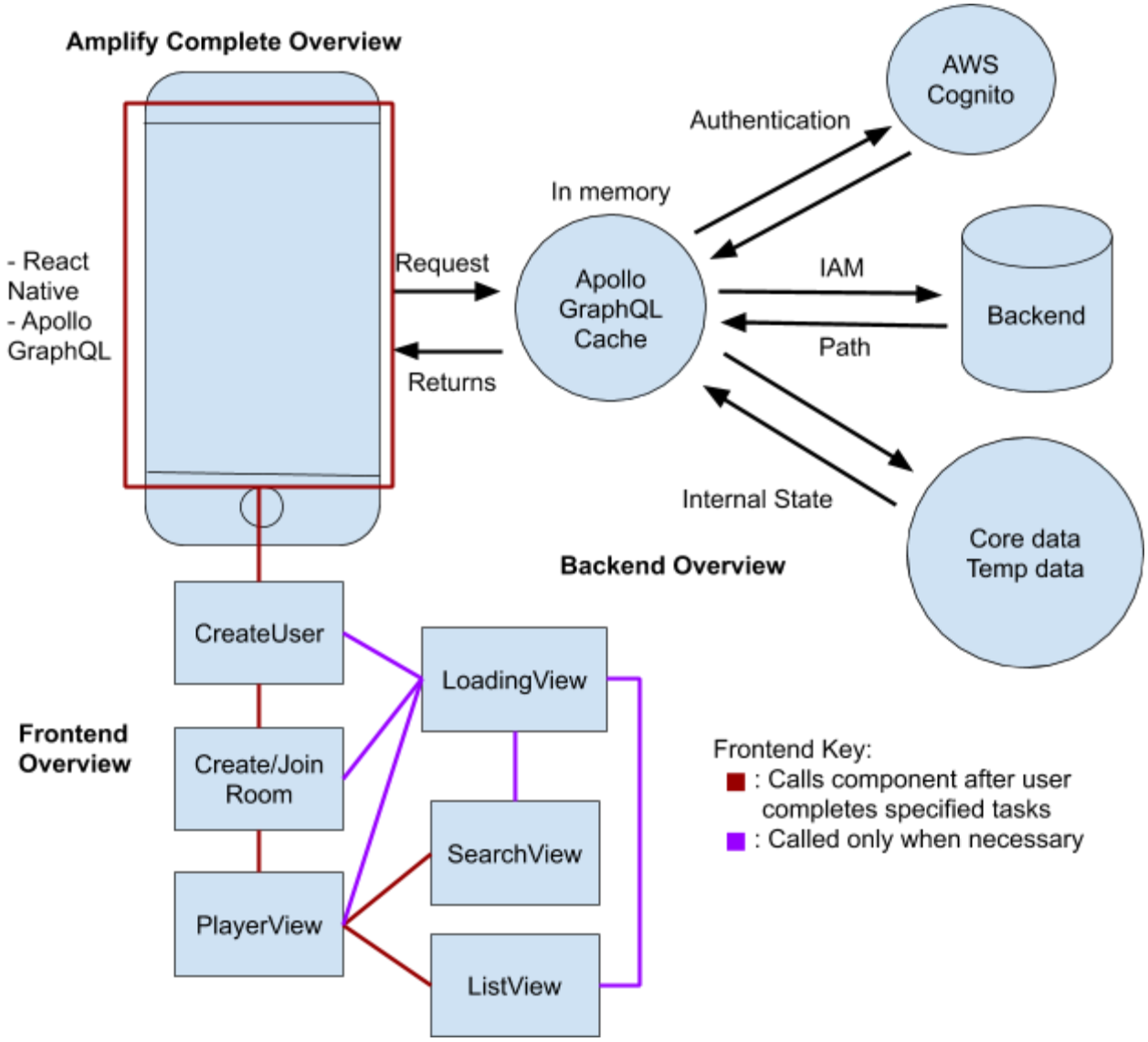
```
      ],
      "owner": {
        "userID": "123456789"
      }
    }
  }
}
```


6.3.3.2.11 createUserRoomInfo Endpoint

- Input
  - createUserRoomInfo(userID: String!): UserRoomInfo!
  - In this example, userID: “123456789”
- Response (JSON)


```
{
  "data": {
    "createUserRoomInfo": {
      "userID": "123456789",
      "queue": []
    }
  }
}
```

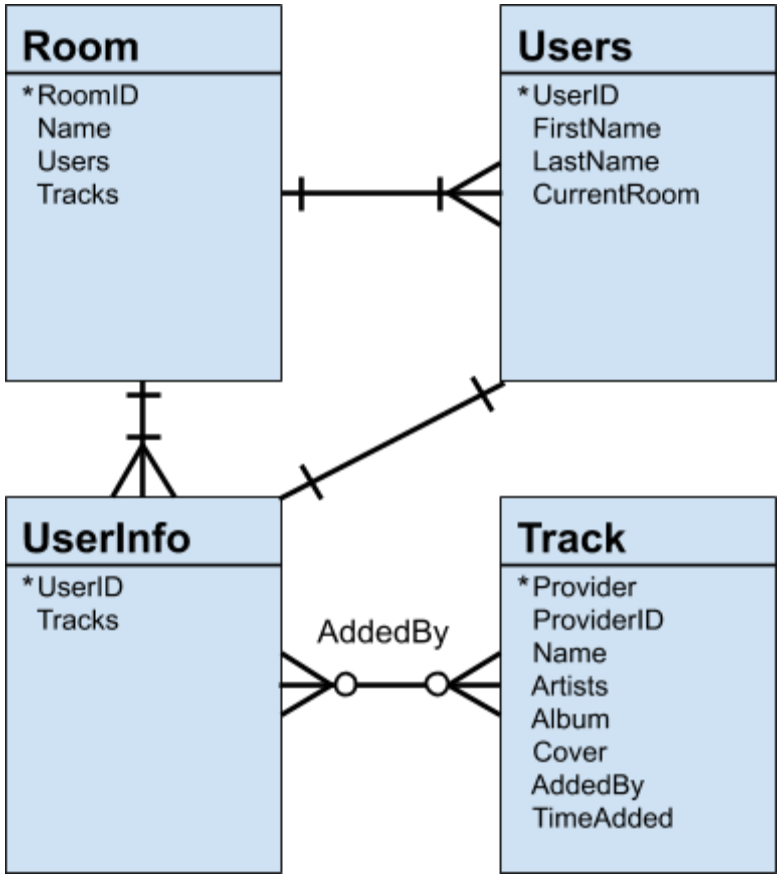

6.3.4 Detailed Design Diagrams



6.4 Database Design and Description

The database will use JavaScript, Apollo GraphQL, Amazon Cognito, Amazon Relational Database Services, Amazon API Gateway, AWS EC2, AWS Lambda, and AWS Amplify. The database engine being used is PostgreSQL. Potential users include any person who has access to Spotify and YouTube who has a passion for listening to music with friends.

6.4.1 Database Design ER Diagram



6.4.2 Database Access

Our database will be accessed through Amazon RDS and is hosted by Amazon EC2.

6.4.3 Database Security

The security of the database is taken care of through Amazon's EC2 service.