Donovan Moini, Ian Lizarda, Ben Kern
Dr. Johnson
CMSI 486
10/16/2019

Preliminary Database Design

1.1 – Project description

- Our database will be the database for our CMSI 401 project, Amplify. Amplify is a community music streaming service that will allow users to create and join rooms that host public music playlists using Spotify and YouTube services, similar to platforms like plug.dj and Outloud. The database will use JavaScript, Apollo GraphQL, Amazon Cognito, Amazon Relational Database Services, Amazon Gateway, 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. Due to the large nature of our group, this database will be split into two parts so our group will be in two separate teams. Alexia and Serena will fully handle Rooms and partially handle Users. Donovan, Ben, and Ian will fully handle Tracks and UserInfo and partially handle Users. Due to the inherent connections between the two parts, there will be some redundant information between the two project proposals to allow for a solid and overarching understanding of the full construction of the database. This proposal was evenly constructed by both teams.

1.2 – Data description

- Room
  - roomID: A string that defines the ID of a room.
  - name: A string that defines the name of a room.
  - users: A list of strings that define the users in the room.
  - tracks: A list of *Tracks* in the current room.
- User
  - userID: A string that defines the ID of the user.
  - firstName: A string that defines the first name of the user.
  - lastName: A string that defines the last name of the user.
  - currentRoom: A string that defines the current room ID.
- Track
  - provider: A string that defines the music provider (Spotify, YouTube, etc…).
  - providerID: A string that defines the ID of the track from the specified provider (Spotify, YouTube, etc…).
  - name: A string that defines the name of the desired track.
  - artists: A list of strings that contains the artist(s) for a desired track.
  - album: A string that defines the name of the album containing the desired track.
  - cover: A string that links to the cover of the album containing the desired track.
  - addedBy: A string that specifies the userID of the user who added the *Track*.
  - timeAdded: A string that specifies the time the *Track* was added to a queue.
- UserInfo
  - userID: A string that defines the ID of the user.
  - tracks: A list of *Tracks* in the current room.

1.3 – Five examples of the type of data that it will provide to the user

- Room
  - tracks: Returns a list of the tracks in the current room.
  - rooms: Returns the list of rooms.
  - getRoom: Returns room of specified room ID.
  - getUserCurrentRoom: Returns current room of given user.
- User
  - users: Returns list of users.
  - getUser: Returns user given a user ID.
- Track
  - searchTracks: Returns list of tracks given a search query.
  - getTrackInfo: Returns the information stored for a given *Track*.
- UserInfo
  - getUserInfo: Returns *UserInfo* of given user ID.
  - track: Returns list of *Tracks* of specified user.

1.4 – A preliminary idea of the schema of the database (* = primary key)

- Rooms: Table of the different rooms
  - *roomID: Column that specifies room ID as a string.
  - name: Column that specifies room name as a string.
- Users: Table of the users
  - *userID: Column that specifies the ID of the user as a string.
  - firstName: Column that specifies the first name of the user as a string.
  - lastName: Column that specifies the last name of the user as a string.
  - currentRoom: Column that specifies the current room ID as a string.
- Tracks: Table of the different songs being used in rooms
  - provider: Column that specifies the music provider (Spotify, YouTube, etc…) as a string.
  - *providerID: Column that specifies the ID of the track from the specified provider (Spotify, YouTube, etc…) as a string.
  - name: Column that specifies the name of the track as a string.
  - artists: Column that specifies the artist(s) for a desired *Track* as a string separated by commas.
  - album: Column that specifies the name of the album of the track as a string.
  - cover: Column that specifies the cover of the album of the track as a string.
- UserInfo: Table of information linked to a specific user
  - *userID: Column that specifies the ID of the user as a string.
  - tracks: Column that specifies *Tracks* in the user's personal queue as a string of *Track* provider IDs separated by commas.

1.5 – Preliminary Entity-Relationship Diagram