Serena Zafiris, Donovan Moini, Ian Lizarda, Ben Kern, Alexia Filler
Dr. Johnson
CMSI 401
November 6th, 2019

Software Design Description

TABLE OF CONTENTS

6.1 Introduction

This document presents the architecture and detailed design for the software for Amplify, which provides an application to allow registered users to create and join public playlist rooms. Amplify will be 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. Users will be able to join rooms and add songs of their choice from Spotify or YouTube to the community queue, which will be cycled through in round-robin style between all users' queued songs. Amplify will also have features useful to an individual user; the user will be able to import multiple playlists and rotate through those playlists via round-robin style.

6.1.1 System Objectives

The goal of Amplify is to provide the user with a means of sharing music with their friends. Amplify will allow a user to create a public playlist in which their friends can join and add their own music, which will be played in a round robin fashion. Users will be able to access music from both Youtube and Spotify. The application will be easy to use with it's minimalistic design.

6.1.2 Hardware, Software, and Human Interfaces

- Hardware
  - Amplify development will require a computer
  - Amplify will require a smartphone running either android or IOS
- Software
  - Node.js
  - React Native
  - iPhone Emulator
  - Android Emulator

    - TypeScript
    - PostgreSQL
- Human Interfaces
    - Amplify will require a smartphone running either Android or IOS

6.2 Architectural Design

6.2.1 Major Software Components

6.2.1.1 Frontend Components (including but not limited to)

6.2.1.1.1 PlayerView Component - Array, Object
6.2.1.1.2 CreateJoinView Component - Array, Object
6.2.1.1.3 CreateUserView Component - Array, Object
6.2.1.1.4 LoadingView Component - Object
6.2.1.1.5 SearchView Component - Array, Object
6.2.1.1.6 ListView Component - Object

6.2.1.2 Backend Components (including but not limited to)

6.2.1.2.1 searchTracks Endpoint - JSON
6.2.1.2.2 getUser Endpoint - JSON
6.2.1.2.3 getRoomUsers Endpoint - JSON
6.2.1.2.4 getCurrentRoom Endpoint - JSON
6.2.1.2.5 getCurrentRoomTracks Endpoint - JSON
6.2.1.2.6 addTrack Endpoint - JSON
6.2.1.2.7 createUser Endpoint - JSON
6.2.1.2.8 createRoom Endpoint - JSON
6.2.1.2.9 joinRoom Endpoint - JSON
6.2.1.2.10 leaveRoom Endpoint - JSON
6.2.1.2.11 createUserRoomInfo Endpoint - JSON
6.2.1.2.12 Database - JSON

6.2.2 Major Software Interactions

6.2.2.1 Frontend Components (including but not limited to)

6.2.2.1.1 PlayerView Component
- Current Tracks in Room: data.getUserCurrentRoom.tracks

6.2.2.1.2 CreateJoinView Component
- Grabs data for created or joined room: roomData.data

6.2.2.1.3 CreateUserView Component
- Calls createUser endpoint using the data of firstName, lastName, and Email

6.2.2.1.4 LoadingView Component
- Gets current room of user: data.getUser.currentRoom

6.2.2.1.5 SearchView Component
- Calls the searchTracks endpoint using the {searchQuery} data

6.2.2.1.6 ListView Component

- Get User Initials: {item.addedBy.firstName.substring(0, 1)}{item.addedBy.lastName.substring(0, 1)}
- Get Song Name: {item.name}
- Get Song Provider: {item.provider}
- Get Artist(s) Name: {item.artists && item.artists.map((artist, index) => ({artist})
- Get Album cover: item.cover

6.2.2.2 Backend Components (including but not limited to)

6.2.2.2.1 searchTracks Endpoint

- Given a search query and a search type (song name, artist, etc.), returns a filtered list of tracks that include the query of the search type specified.

6.2.2.2.2 getUser Endpoint

- Given a user ID, returns the User object with that user ID, if it exists.

6.2.2.2.3 getRoomUsers Endpoint

- Given a room ID, returns the list of Users that are associated with that room.

6.2.2.2.4 getCurrentRoom Endpoint

- Given a user ID, returns the Room that the User associated with the input user ID is currently a member of.

6.2.2.2.5 getCurrentRoomTracks Endpoint

- Given a room ID, returns the master queue of that Room.

6.2.2.2.6 addTrack Endpoint

- Given a TrackInput and a user ID, adds the Track into the User of the user ID's personal queue.

6.2.2.2.7 createUser Endpoint

- Given a user ID, a first name, and a last name, returns a User object containing the input information. This creates the User object.

6.2.2.2.8 createRoom Endpoint

- Given a user ID, creates a Room and assigns.

6.2.2.2.9 joinRoom Endpoint

- Given a user ID and a room ID, returns the Room with the specified room ID, updated to contain the User associated with the input user ID.

6.2.2.2.10 leaveRoom Endpoint

- Given a user ID, returns the Room that the User associated with the user ID is in. The returned Room is updated not to contain the User that has left the room.

6.2.2.2.11 createUserRoomInfo Endpoint

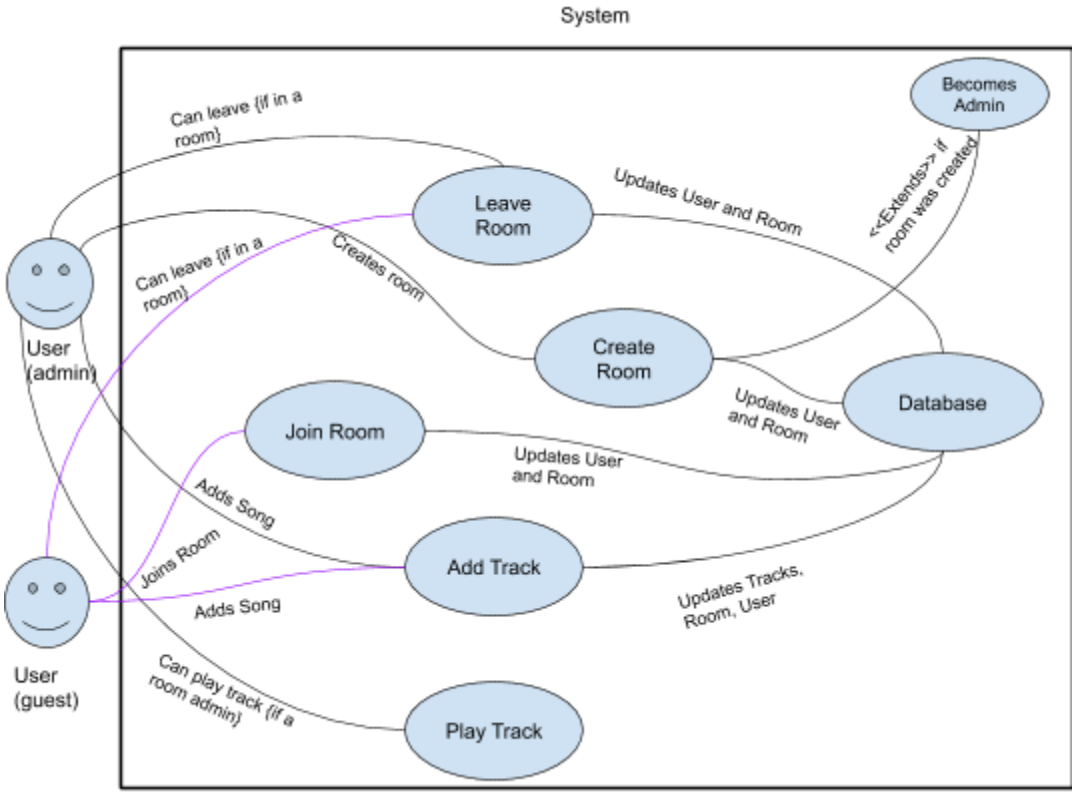
- Given a user ID, creates a UserRoomInfo for that user ID.

6.2.2.2.12 Database

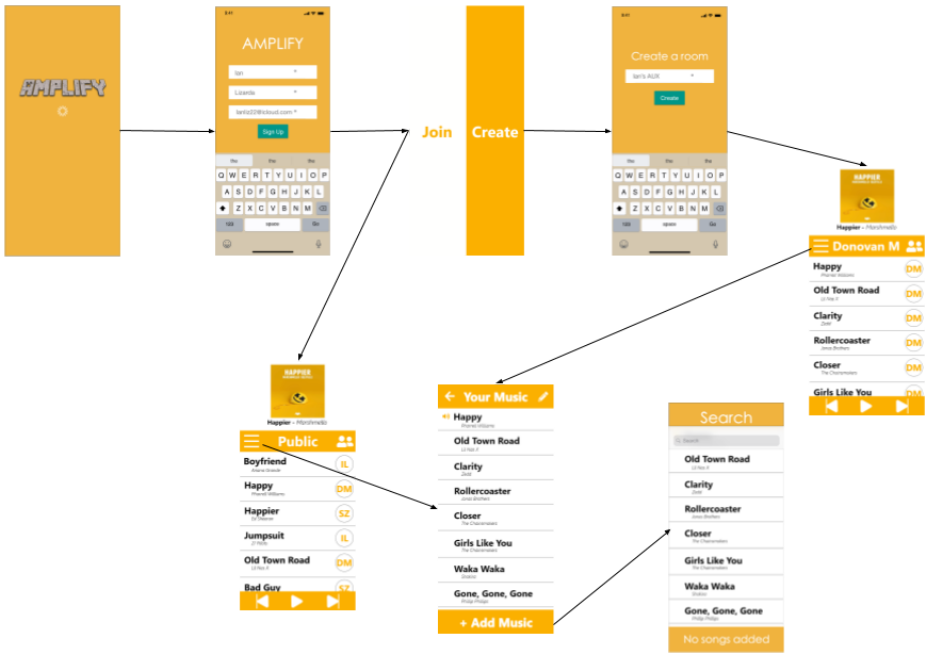
- An Amazon RDS instances using PostgreSQL.

6.2.3 Architectural Design Diagrams

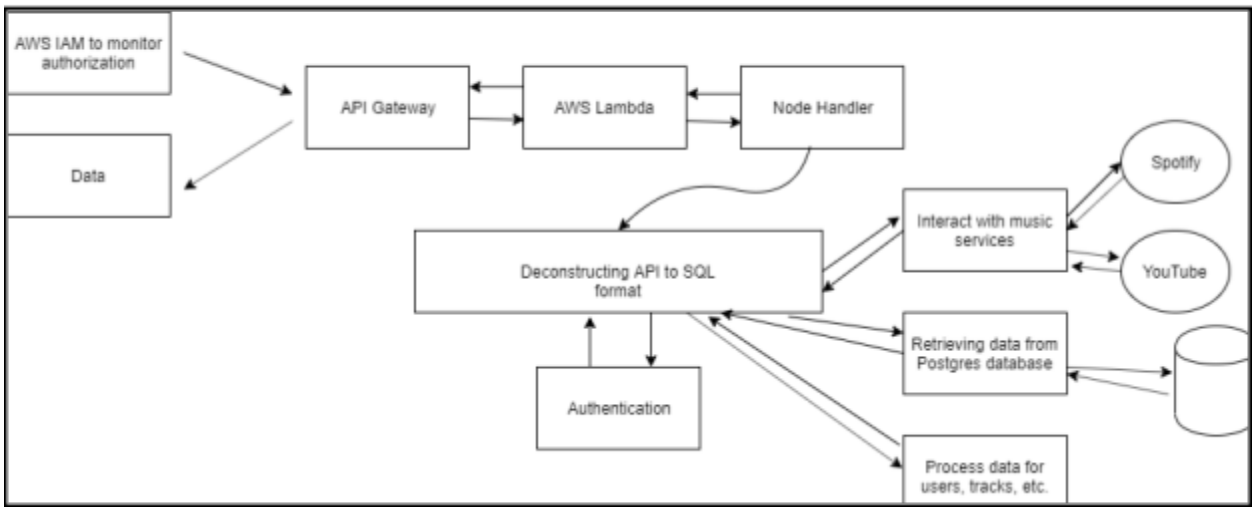
Use Case Diagram



User Interface Example



Backend Data Flow Diagram



Message Interaction Flow Diagram