# Software Requirements Document for [ SD1 ]

TEAM: SD1

AUTHORS: Marcin Lukanus, Dylan Mrzlak, Chris Pavlopoulos, Alex Thompson

| Version | Date | Author | Change |
|---|---|---|---|
| 0.1 | 4/16/20 | ML | Added bulk of information |
| 0.2 | 5/5/20 | ML/DM | Fixed docs in accordance with Mitra's suggestions |
| | | | |
| | | | |

# Table of Contents

### 1.1 Purpose

The purpose of this document is to establish how the application should interact with the end user, and establish all application requirements functional, and non functional. Once finalized, this document will state what must be accomplished for the application to be considered finished.

### 1.2 Scope

This SRS covers a number of potential use cases that users may encounter, as well as an overview of the project and its intended uses. It also includes information on the project's UI sketches, but the primary purpose is to give detailed descriptions of anticipated use cases.

### 1.3 Definitions, acronyms, abbreviations

| Term | Description |
|---|---|
| Party | A room in which members of the party are able to add Spotify songs to the queue |
| Host | The creator of a designated Party room. |

### 1.4 References

[None]

### 1.5 Overview

People at social gatherings often like to suggest songs at the expense of hassling whoever is currently utilizing the speakers/music devices. Our solution is to create an application that would allow both the host and attendees of social gatherings to collectively queue songs via their own devices to be played through the host's device. The Host will be allowed to create Party rooms and any user would be able to join that Party room via the Host's username.

### 1.6 Product Perspective

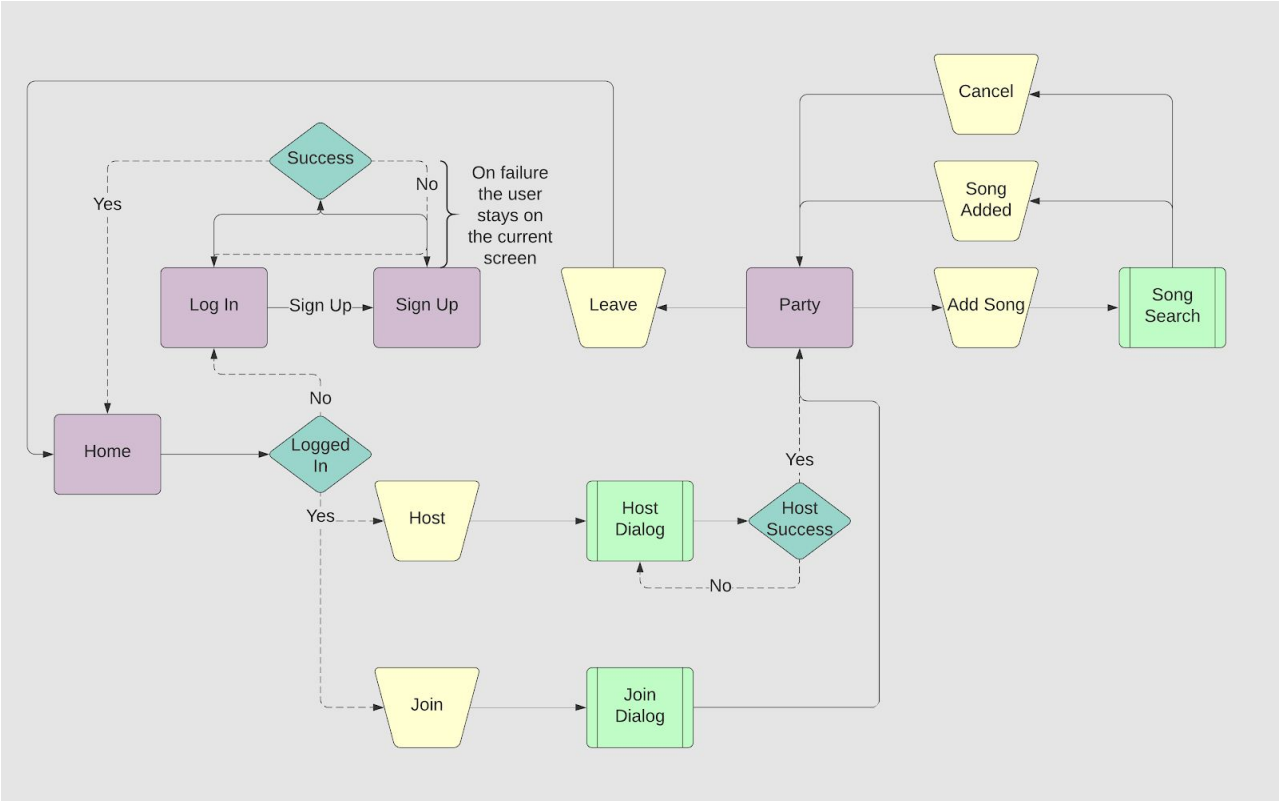
"Spotify", a popular music streaming app

#### 1.6.1.1 Concept of Operations

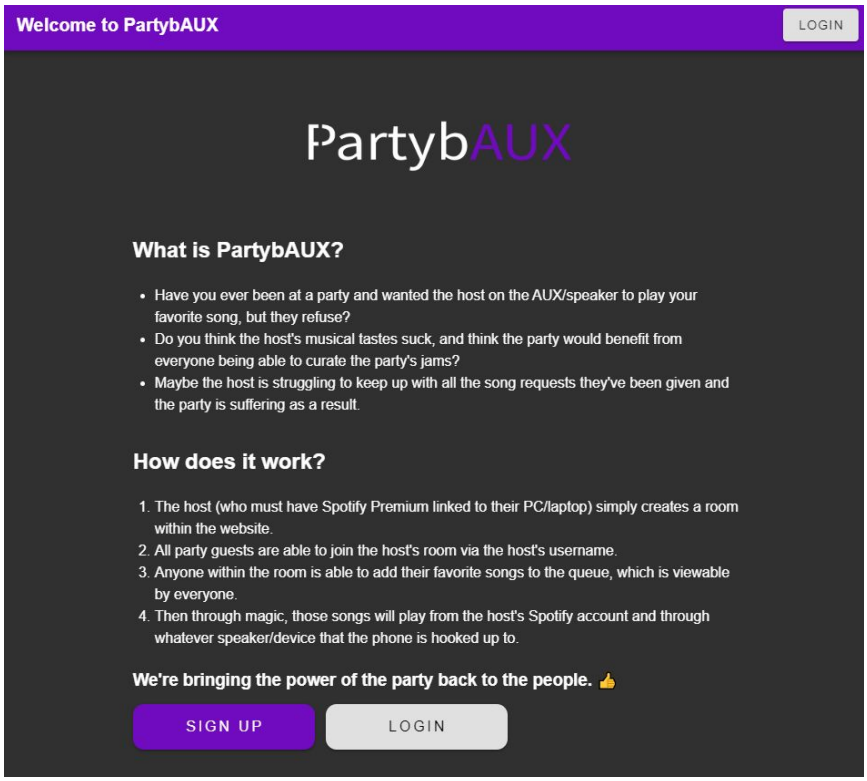
This project will be a web-based application that allows the user to create an account on a webpage, create and join Party rooms. If the user is the Host of a Party, they will be playing the songs in the queue through their own device that is connected to their Spotify Premium account. They will be able to skip or remove songs in the queue. Otherwise, users in a Party will be able to add songs to the queue.

This system will be supported by a database, which will store lists of users, parties, and songs. We utilize a cross-reference table to make communication between the databases easier for ourselves. The frontend will make requests to the backend's databases in order to function when creating and destroying Party rooms as well as associating Users to Parties and Songs to Parties.

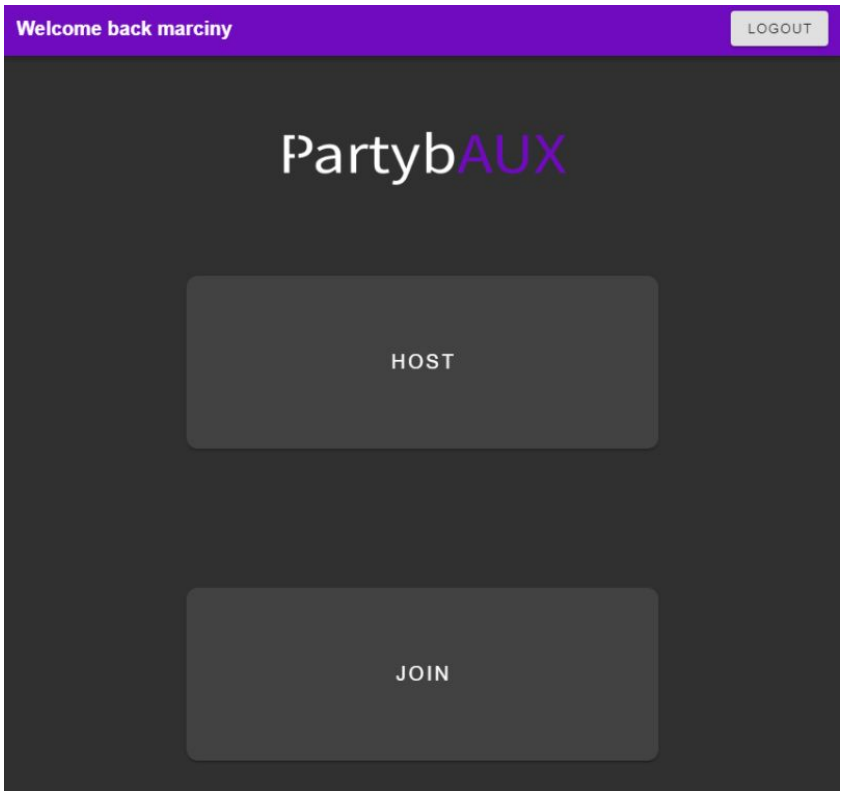
### 1.6.1.2 Major User Interfaces



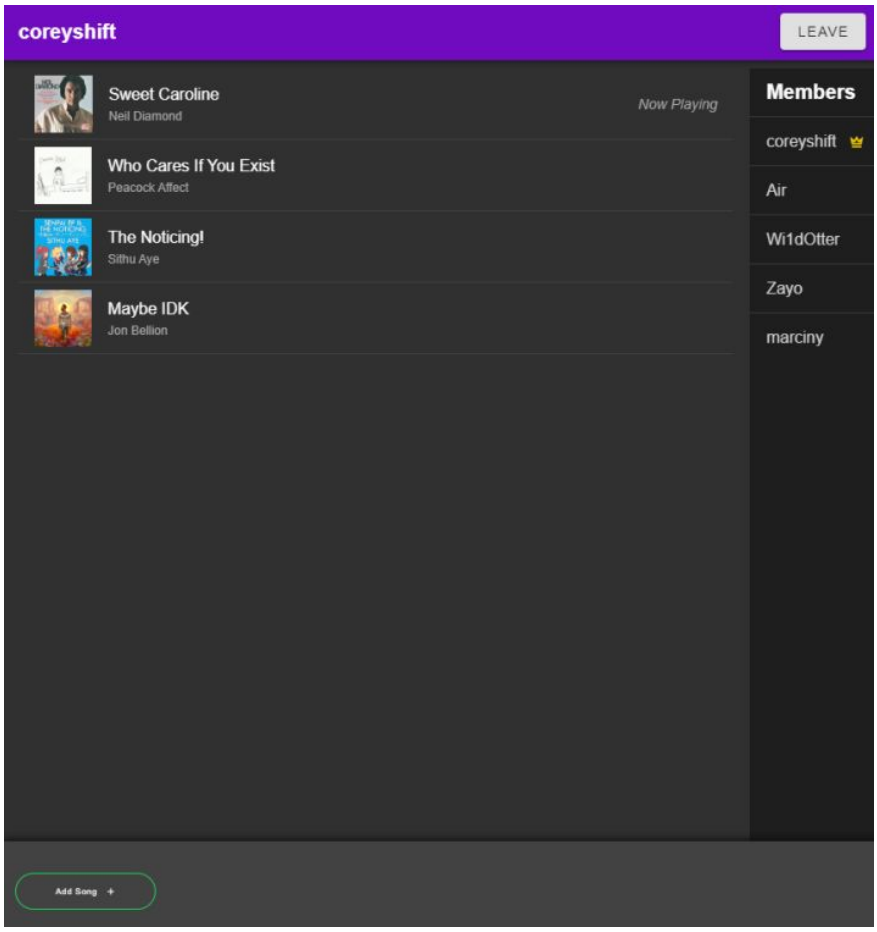
#### 1.6.1.3 Example Screenshot and description



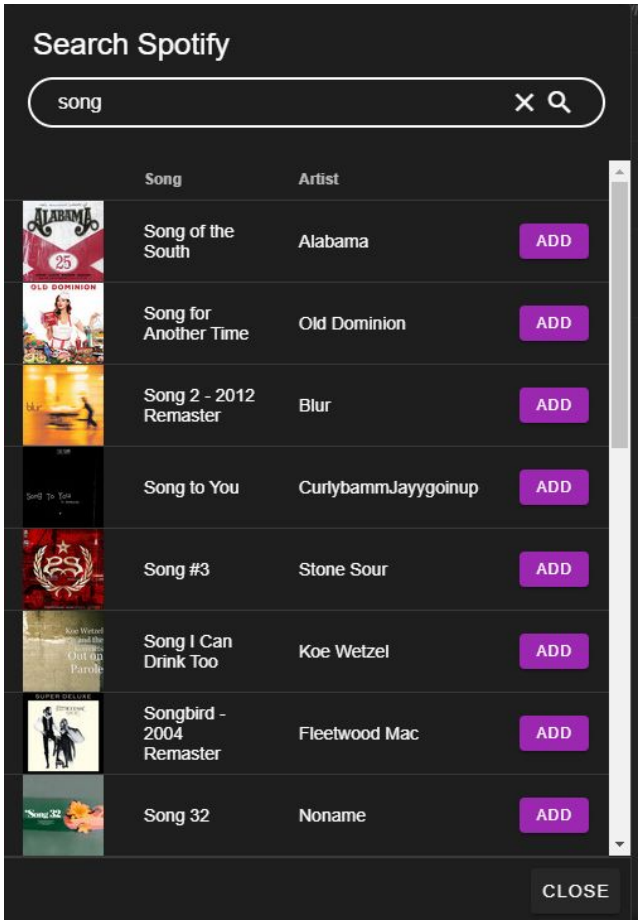

Home page when not logged in



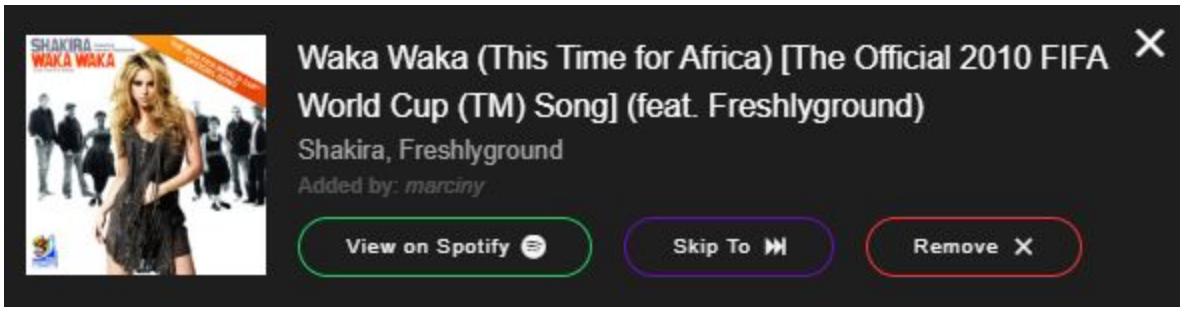

Home page when logged in



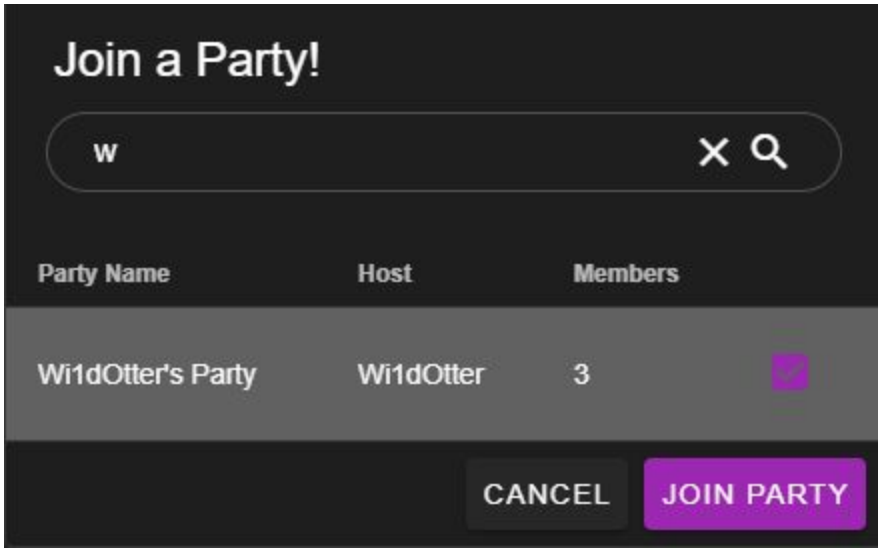

Party view with multiple members and multiple songs in the queue




Spotify Search Dialog




Song Preview Component

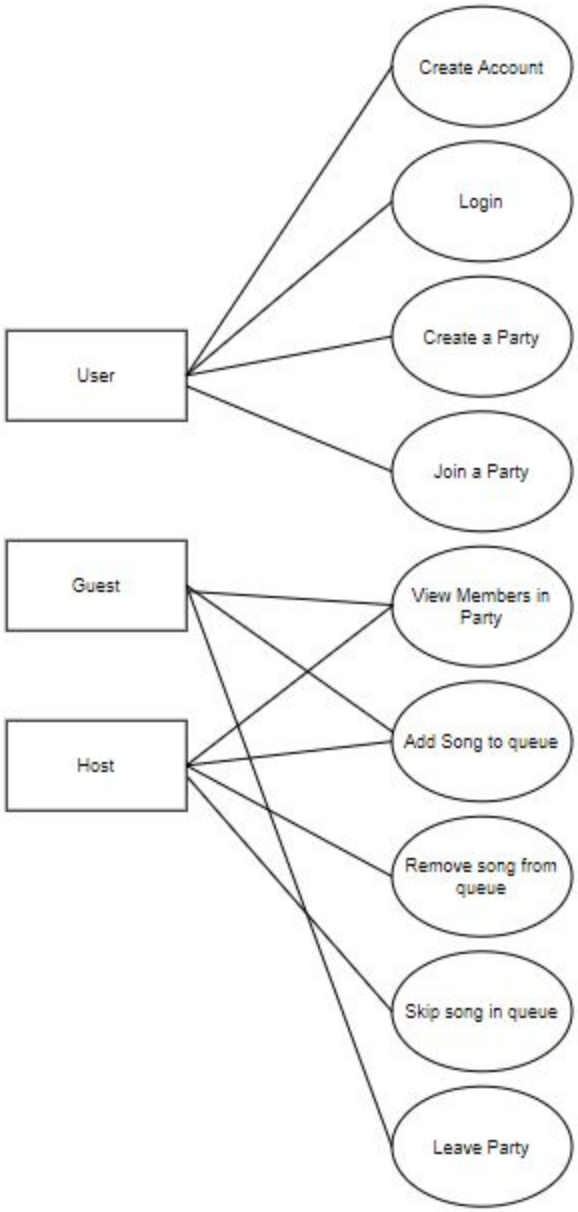



Join Party Dialog

1.6.2 Hardware Interfaces

1.6.2.1 Any non-mobile device that supports a modern web browser and enables javascript functionality.

## 1.7 PRODUCT FUNCTIONS



## 1.8 Use Cases

These use cases describe the steps that a user could take for creating or joining a Party in order to become a Host or a Guest. This will give them the functionality associated with adding songs, as well as the additional Host privileges of skipping and removing songs.

### 1.8.1 User creates a Party (User goal / User/Host)

Main scenario:

1) User goes to the application's website.
2) The user goes to the Login page.
3) The user enters his or her user credentials.
4) The user is redirected to the Home page.
5) The user clicks the Host button.
6) The user is prompted to name their Party, which defaults to "<username>'s Party"
7) The user names their Party and clicks the Create button.
8) The user has created a Party, making them into a Host

Extensions:

3a. User attempts to log in, but no account for the user exists.

3a1. The user instead navigates to the Sign Up page.

3a2. The user enters their desired username, email, and password.

3a3. The frontend sends this information to be stored in a Users database in the backend, which creates the user.

3a4. The user is redirected to the Home page and automatically logged in with their inputted information from the Sign Up page.

### 1.8.2 User wishes to join an existing Party (User Goal / User/Guest)

Main scenario:

1. The user clicks the Join button.
2. The user enters the name of the existing Party.
3. The frontend sends the input to the backend to parse the Party database for any entries that contain the input keywords.
4. The frontend displays a list of relevant Parties based on input.
5. User selects a Party and clicks the Join Party button.
6. User is now in the Party as a Guest.

1.8.3 Host or Guest in a Party wants to add a song to the queue. (User goal / Host/Guest)

Main scenario:

1. User clicks the Add Song + button.
2. User inputs the song title into the Search Spotify search bar.
3. User locates desired song, and click the Add button associated with that song.
4. The search dialog closes and the song is now added to the queue.

1.8.4 Create Account (User goal / All users)

Main scenario:

1. User navigates to Sign Up page.
2. User enters desired username, their email, and a password.
3. The frontend verifies that the information provided is in the correct format for storage.
4. The frontend sends the valid information to the backend to be stored as a User.

Extensions:

2a. User enters information that is not in the correct format

2a1. The frontend will display a validation error for any information that is left blank or invalid (i.e. email input not a valid email format).

3a. User enters information that cannot be stored. (Example: Account with that username already exists)

3a1. System will display an error signifying to the user that an account with that username already exists in the User database.

1.8.5 Guest wants to leave a Party (User goal / Guest)

Main Scenario:

1. A user is currently in a Party (and is not the Host).
2. The user clicks the Leave button.
3. The user is redirected to the Home page.
4. User is no longer a Guest in that previous Party (does not display under the Members list in that Party).

1.8.6 Host wants to leave their Party (User goal / Host)

Main Scenario:

1. A user is in a Party that they created (they are the Host).
2. The user clicks the Leave button.
3. The user is redirected back to the Home page.
    a. All Guests of that Party are also removed from the Party and redirected to the Home page.
4. That Party is removed from the Party database (including any songs left in that queue).

1.8.7 Host wants to skip a song in their Party (User Goal / Host)

Main Scenario:

1. The user is in a Party that they created (they are the Host).

2. There exists a song that is actively playing.
3. The user clicks the Skip icon in the Player Bar.
4. The song is removed from the queue.
    a. The next song (if applicable) begins to play.

Extensions:

1a No songs are in the queue.

1a1 If there are no songs in the queue, the Skip button does nothing.

### 1.9 User characteristics

1.9.1 Users are typically listeners of music.

1.9.2 Meant to be used at social gatherings.

### 1.10 Assumptions and Dependencies

– Assumptions
- Users have access to reliable Internet service.
- Users have existing Spotify accounts.

– Dependencies
- The ‘host’ of parties must have a valid Spotify Premium account.
- The ‘host’ of parties must be using a non-mobile device.

# 2. Specific Requirements

## 2.1. Features

2.1.1. Authentication

2.1.1.1. The Users database is all handled by the Django framework as it provides a robust and secure handling of sensitive user information.

2.1.2. Spotify Authentication

2.1.2.1. Upon signing in for the first time on a new device, we require users to authenticate their Spotify accounts (in compliance with Spotify's Terms of Service).

2.1.3. Song Search

2.1.3.1. Using the Spotify Web API to search for songs from Spotify.

2.1.4. Adding Songs

2.1.4.1. In each Party, there is a song queue that stores the information about each song on the frontend, with the song URIs being stored in the backend which is needed when a song is ready to be played.

2.1.5. Song Playing

2.1.5.1. Upon the creation of the party, the system will create an instance of the Spotify Player for the host. If there is music in the queue, the system will then begin playing the music.

2.1.6. Deployment With Heroku

2.1.6.1. Heroku allows us to host our site easily and allows the usage of pipelines. The pipelines don't rebuild the project, so we changed how the app will be deployed to satisfy the general way a pipeline would work.

## 2.2. Performance requirements

2.2.1. [None]

## 2.3. Design Constraints

2.3.1. The Spotify Web SDK

2.3.1.1. The SDK does not allow for mobile browser compatibility when playing songs. Mobile users may not host Parties.

2.3.2. Spotify Terms of Service

2.3.2.1. Needing to follow Spotify's Terms of Service, we needed to add new functionality to follow their requirements.

2.3.3. Heroku

2.3.3.1. Created limitations in terms of the project's file structure, leading us to move pieces of the project higher in the directory.

2.3.4. Request Forgery Protection

2.3.4.1. Created limitations in terms of the project's file structure due to different origin requests. This led us to move pieces of the project higher in the directory.

## 2.4. Software System Attributes

2.4.1. Reliability

2.4.1.1. Both frameworks we use are popular and well-established with continuous support, allowing us to continue developing with them without fear of near-future refactorization.

2.4.2. Availability

2.4.2.1. PartybAUX is available to anyone with an internet connection. Spotify handles regions themselves, so they will only be able to see songs in their market.

2.4.3. Security

2.4.3.1. Django supplies security middleware to handle passwords encryption and request forgery.

2.4.4. Maintainability

2.4.4.1. Django and Vue scale very well, so adding more features would not be difficult.

2.4.4.2. Both of our root frameworks are easy to learn. If another developer were to join the project, they could begin development rather quickly

2.4.5. Portability

2.4.5.1. App will be available to anyone who has a browser and internet connection. A user could use any device, though they may not be able to host a party.

## 2.5. Other Requirements

2.5.1. [None]