**Play My Song**

Software Requirements Specification

PlayMySong\_SoftwareRequirementsSpecification.doc

Draft 0.1

Contributors: Kenny Cauthen,

John-Paul Sprouse, Connor Wilkinson

September 14, 2016

[ DevSoMo ]



Revisions

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Primary Author(s) | Description of Version | Date Completed |
| Alpha | Kenny Cauthen, John-Paul Sprouse, Connor Wilkinson | Working Copy | 09/14/16 |

[1 Introduction](#_srid46d8n7gc)

[1.1 Purpose](#_efjunjgfg6s0)

[1.2 Scope](#_qocr1cgs7wfm)

[1.3 Definitions, Acronyms, and Abbreviations](#_mh98dffyw2rs)

[1.4 References](#_9ho5b237x9oi)

[1.5 Overview](#_vftse7k0d0jo)

[2 Overall Description](#_s9dmp1wn3df)

[2.1 Product Perspective](#_syl2hrrv0cgi)

[2.1.1 System Interfaces](#_ktgw60kno2gp)

[2.1.2 User Interfaces](#_m9opabbuf5i0)

[2.1.3 Hardware Interfaces](#_maic0m65l0on)

[2.1.4 Software Interfaces](#_ilzob5eltips)

[2.1.5 Communications Interfaces](#_fmru9lmox2wx)

[2.1.6 Memory Constraints](#_72zskg2ww90x)

[2.1.7 Operations](#_g7zmra8trh5v)

[2.2 Product Functions](#_mxbfb4tdfh04)

[2.3 User Characteristics](#_qxnp5fdy7i3f)

[2.4 Constraints](#_363yl3kq62jr)

[2.5 Assumptions and Dependencies](#_xu2f4hwcltw2)

[2.6 Apportioning of Requirements](#_ozrgm4obxc8o)

[3 Specific Requirements](#_s3a9vmdsrgn9)

[3.1 External Interface Requirements](#_j2l1x790rh4u)

[3.1.1 User Interfaces](#_rawieaqciamw)

[3.1.2 Hardware Interfaces](#_xmskg5ognc50)

[3.1.3 Software Interfaces](#_s1rrh2196k3m)

[3.1.4 Communications Interfaces](#_2h2we8jx5sam)

[3.2 Software Product Features](#_r5ufhtu1obof)

[3.2.1 Event Management System](#_1poy6pnx7ryt)

[Purpose](#_n8xp5rz4ewjq)

[Stimulus/Response Sequence](#_hq0ea292di7)

[Associated Functional Requirements](#_bnbfmaza806d)

[3.2.1.1 Find Event Functional Requirement](#_tctai3weh90)

[Introduction](#_qapdraamxedr)

[Inputs](#_9rb9wrrjvmbp)

[Processing](#_uzqs9ds2i0ms)

[Outputs](#_h7opoxzh2z3h)

[3.2.1.2 Join Active Event Functional Requirement](#_e3m4hshz6xml)

[Introduction](#_la79hk396e63)

[Inputs](#_m2kwstg69fnv)

[Processing](#_wl8dejd9imeh)

[Outputs](#_r2giuh86a4w3)

[3.2.1.3 Edit Events Functional Requirement](#_wsyzyrgkd51h)

[Introduction](#_j4n1pgth99yb)

[Inputs](#_awqsoevkao68)

[Processing](#_h7g7wuf5cl58)

[Outputs](#_styvptgzdo8h)

[3.2.2 Sign-In Feature](#_yxr6sg6tch1m)

[Purpose](#_cu1vc1jrm72x)

[Stimulus/Response Sequence](#_na5jgvxf7pdj)

[Associated Functional Requirements](#_k86ny21ghr)

[3.2.2.1 Functional Requirement (Sign Up)](#_o4em07fee08s)

[Introduction](#_21wq6a4771es)

[Inputs](#_q3d34wpxs794)

[Processing](#_ryfxy4qiraiv)

[Outputs](#_998r3gizktu6)

[3.2.2.2 Functional Requirement (Sign In)](#_fxuo9bx8qkok)

[Introduction](#_zephuu1y89bu)

[Inputs](#_x93o49prf9h0)

[Processing](#_urgjs1h67dl1)

[Outputs](#_162gn8344hx7)

[3.2.2.3 Functional Requirement (Edit Account)](#_ga6e4uk6f34s)

[Introduction](#_hnnupdntbku3)

[Inputs](#_3r68gyxamdeq)

[Processing](#_86o6hkepiogf)

[Outputs](#_ts9ifrqt21oi)

[3.2.3 Song Request Features](#_wao9g3voy6d6)

[Purpose](#_i1siq5jw6i6k)

[Stimulus/Response Sequence](#_e5lnu3z1aol4)

[Associated Functional Requirements](#_rzelv0thqtzr)

[3.2.3.1 View Song Request Queue Functional Requirement](#_xh1n0wyhmf5n)

[Introduction](#_l2sbh5tiqe3i)

[Inputs](#_uuooi08ku495)

[Processing](#_lbasvj9sneq)

[Outputs](#_nocii97wf4db)

[3.2.3.2 Find Search Song Functional Requirement](#_xr3ula69xvlx)

[Introduction](#_gu22oe470atp)

[Inputs](#_m3p19qd6im1c)

[Processing](#_743b653y9fl3)

[Outputs](#_fel1244r18ep)

[3.2.3.3 Request Song Functional Requirement](#_zc9hbsm4a567)

[Introduction](#_j3z3ao8ova7c)

[Inputs](#_kg5uk2foxmqk)

[Processing](#_2tve6ay2nd6v)

[Outputs](#_9kh2qdq86dc0)

[3.2.3.4 Reject Song Functional Requirement](#_kwhzxn2bzusf)

[Introduction](#_w2jjh3vdjegb)

[Inputs](#_3vq15cqwwzv0)

[Processing](#_mdvig4n8ybya)

[Outputs](#_dr5f10u4ywht)

[3.2.3.5 Acknowledge Song Functional Requirement](#_gl8ikthkmovk)

[Introduction](#_cqagiwljfy8t)

[Inputs](#_v5wtitqd77n)

[Processing](#_3th3g6co4sef)

[Outputs](#_s46qdqss53o)

[3.2.4 Needed List Feature](#_ezgjsvq0b1w7)

[Purpose](#_44irfmkg22vs)

[Stimulus/Response Sequence](#_z8qmj9g3og54)

[3.2.4.1 View Needed List Functional Requirement](#_93f95mzaysbe)

[Introduction](#_5060hcl6ku7w)

[Inputs](#_qihg06bh8skk)

[Processing](#_kqx67qxy4g0l)

[Outputs](#_me7uoiluolpz)

[3.2.4.2 Add Song to Needed List Functional Requirement](#_cpi7s171qqhg)

[Introduction](#_5q2zmkx6d771)

[Inputs](#_jp4yd7po3pva)

[Processing](#_43bhog2b3vsu)

[Outputs](#_aty5hfmmurw2)

[3.2.4.3 Remove Song from Needed List Functional Requirement](#_71qtiwp40hcw)

[Introduction](#_kjie37xtxi78)

[Inputs](#_jfb7une61x18)

[Processing](#_g2vcd0dn29um)

[Outputs](#_v0t1mcz4c0t6)

[3.3 Performance Requirements](#_qcxn2eju1qmt)

[3.4 Design Constraints](#_vqjoitoc7way)

[3.5 Software System Attributes](#_19zl8qf2qm)

[3.5.1 Reliability](#_d8c6irff7gis)

[3.5.2 Availability](#_96l23jn1b0qn)

[3.5.3 Security](#_vmtwxjhpubsg)

[3.5.4 Maintainability](#_2cq00rqc5s5g)

[3.6 Logical Database Requirements](#_3l7mf1wbg267)

[3.7 Other Requirements](#_2gz3f5aegr47)

# 

# **1 Introduction**

## **1.1 Purpose**

The application will facilitate digital interactions between an event, DJ, and the patrons. The idea behind this app is to address a problem most DJ’s have while playing; people coming up and interrupting the set to requests songs.

## **1.2 Scope**

The system will be called Play My Song. DJ’s will have an administrator account that will have the ability to create location and time/date specific events. Event attendees will be able to register as users associated with the event. The attendees will be able to request songs to be played. The DJ will receive the request and either accept or deny it. If accepted the song will be added to a queue that is visible to both the DJ and the attendee. The event will record metrics regarding requests and the genres thereof. The data generated could then be used by DJ’s for analytics purposes. The system will not interface directly with the DJ’s equipment. The DJ will be responsible for playing music independent of the system.

## **1.3 Definitions, Acronyms, and Abbreviations**

**DJ** - Disc Jockey, responsible for event creation, song acceptance/denial, and overall event administration.

**VO** - Venue Owner, proprietor of establishment

**Patron** - Event Attendee, User, responsible for making requests.

**Pro-User** - DJ System administrator

**Event** - Location and time/date specific affair in which a DJ is responsible for playing music.

**GUI** - Graphical User Interface

**Active Event** - An event where the current time is within the start and end time/date of the Event.

**Request Queue** - list of requested songs from during an active event

**Needed List** - DJ specific list of songs that are not available in their library but have been requested

## **1.4 References**

Kenny Cauthen - AKA James Klenser, Experience: 10 years as a Disc Jockey and 13 years as a music producer. Current residence at Jerzees in Athens Georgia.

## **1.5 Overview**

This document will provide the requirements necessary for making the ‘Play My Song’ application.

# **2 Overall Description**

In this section, describe the general factors that affect the product and its requirements. This section should contain background information, not state specific requirements.

## **2.1 Product Perspective**

The Play My Song system will be independent and self-contained.

### **2.1.1 System Interfaces**

The system will have to interface with a database of song titles, an active event session based request database, and a DJ specific needed song database.

### **2.1.2 User Interfaces**

The user will utilize a GUI web-based Interface with mobile friendly views.

### **2.1.3 Hardware Interfaces**

The system will utilize HTML5 to ensure functionality and ubiquity of user access.

### **2.1.4 Software Interfaces**

The Play My Song system will need to interface with a database of current songs titles. The database should not contain actual song files. The purpose for the interacting with the database is so that users to are able to quickly find songs they want to request instead of allowing users to manually type songs in.

### **2.1.5 Communications Interfaces**

The system will communicate over the internet.

### **2.1.6 Memory Constraints**

Memory constraint are limited to accessing current browser. The system resource footprint should be expected to be the same as a typical web-based system.

### **2.1.7 Operations**

Backups of the database are performed at predetermined intervals to protect data from loss, corruption, etc.

## **2.2 Product Functions**

Patron:

* Find and check into local events that use Play My Song
* View the active song request list for the event
* Find any song to add to the event request list

Pro User:

* Create and manage events for patrons to check into and request songs
* Accept or Reject songs in the request queue
* Add unavailable songs to a wish list to download later

## **2.3 User Characteristics**

Pro-Users are familiar with the characteristics and methods involved in being a DJ.

They have experience with managing events and handling song requests.

Patron/Users are familiar with reading and pressing buttons. They should be expected to have experience using websites that employ universal web standards. No technical experience required.

## **2.4 Constraints**

The system should function on a standard smartphone.

The system should function with multiple users connected.

The system should update in real time based on input from user devices.

# **3 Specific Requirements**

## **3.1 External Interface Requirements**

Music Database (MusicBrainz)

* Purpose: provide an updated database of all known songs with limited scope so that patrons and DJs can search for songs that they would want to request.
* Data can be sourced from a Music database source such as MusicBrainz.
* Database will be accessed via backend of the web interface.
* Data Fields
  + Artist
  + Song Title
  + Release Date
  + Genre

### **3.1.1 User Interfaces**

Mobile Web Browser

* Acts as an interface between the user and software
* Input is the user
* Output is the GUI for the program
* GUI is organized as a portrait-style display

Laptop and Desktop Computer Web Browser

* Acts as an interface between the user and software
* Input is the user
* Output is the GUI for the program
* GUI is organized as a landscape display

### **3.1.2 Hardware Interfaces**

Hardware Interface will be specified by the web browser capable of handling HTML 5 web interface.

### **3.1.3 Software Interfaces**

Software Interface will be specified by the web browser capable of handling HTML 5 web interface.

### **3.1.4 Communications Interfaces**

Communication Interface will be specified by the web browser capable of handling HTML 5 web interface.

## **3.2 Software Product Features**

The features are broken down by subsystems which are color coded as described in the following legend. The listed features are grouped by the users who are responsible for their execution.

**Subsystems** (Color Key)

1. Event Management System
2. Sign-in System
3. Song Request System
4. Needed List System

All Users:

* Sign Up
* Sign In
* Find/Select Song (Also used in Needed List System)
* Request Song
* View Active Request Queue
* Join Event
* Find(Search) Event

DJ and Event Managers:

* DJ Portal:
  + Create Event
  + View Needed List
  + (Instance of Find Search Event for Event Manager)
* Edit Event
* Acknowledge(Accept) Song
* Reject/Needed Song
* Reject/Lame Song
* Add Song to Needed List
* Remove Song from Needed List

System:

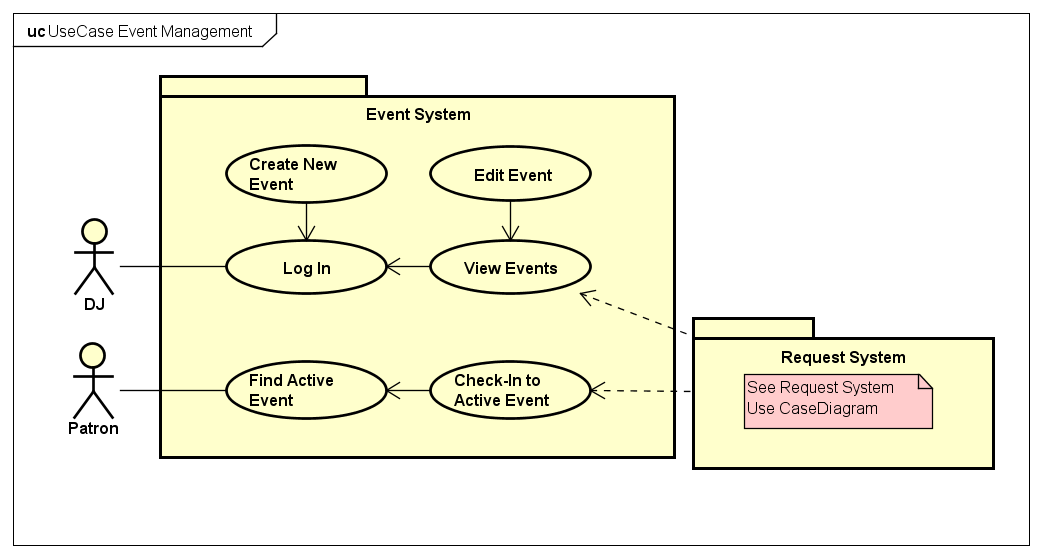
* Activate Event
* Close Event

### 

### 

### **3.2.1 Event Management System**

### **Purpose**

Create, edit, find, and join location-based events which song requests will be associated with.

#### **Stimulus/Response Sequence**

DJ presses/clicks a button to create an event.

DJ enters following fields:

* Event Name
* Event Description
* Event Address
* Local Start time
* Local End time

The event is registered in the database using the fields listed above with a unique identifier and the DJ/Event Manager that has created the event.

#### **Associated Functional Requirements**

##### **3.2.1.1 Find Event Functional Requirement**

Patron will find events they want to join. Event Managers will find/view events that they have created.

###### **Introduction**

Patrons will find events they want to join. The Patron will be able to quick view events that are near them as well as manually typing in an address to find an event near that address. This function should be available immediately after a patron signs in.

Event Manager will view the events that they have created. This will be previously closed events and/or new/pending events that they have created. The system will search for events created by the Event Manager.

###### **Inputs**

Patron can manually type the address for an area to search for an event.

###### **Processing**

For the Patron, the system will automatically process the events available near the user based on location services. If the Patron wants to search a specific area, the system should search the vicinity of the area the Patron has manually typed for events.

For the Event Manager, the system will find events that was created by the Event Manager.

###### **Outputs**

The system outputs these events.This list is sorted by:

* Events near the Patron
* Events near the area specified by the Patron.

System displays events created by the DJ or event Manager. This list is sorted by:

* Upcoming Events - Events that have been created but the date has not passed.
* Previous Events - Most recent limited number of events where the end time and date of the events has expired.

##### **3.2.1.2 Join Active Event Functional Requirement**

Patron and Event Manager will join an event to view the request list. Patrons and Event Managers can only join events that are active or where the current time is within the start and end time/date of the event.

###### **Introduction**

After searching for an event, the user will join the event so that they are able to view the request queue list for that event.

###### **Inputs**

Press/Click a button or action that selects the event the user wants to join.

###### **Processing**

Patron and Event Manager will be locked to current event so that any song requests made will be made to this event only. System will check if the current time is within the start and end time/date(Active).

###### **Outputs**

If the Event is active, Patron will be able to join the Event with a welcome message displayed. If not the system will display an error message the event has not started yet.

##### **3.2.1.3 Edit Events Functional Requirement**

Event Manager will edit events that they have created.

###### **Introduction**

Event Managers will edit events they have created where the event is not active. After viewing all the events that the Event manager has created, they will be able to edit events that have not started (Current time has not surpassed the Start time of the Event) to change the following fields:

* Event Name
* Event Description
* Event Address
* Local Start time
* Local End time

Once an event has started and/or closed, the event can not be edited.

###### **Inputs**

Click a button or action that selects the event the user wants to edit. Manual entry to change the fields. Press/Click button to save changes or cancel to exit with saving changes.

###### **Processing**

System will update all entries that had been changed for the event if user submits the changes. No changes will be committed if the user cancels the the transaction.

###### **Outputs**

No recognizable output at this time.

### **3.2.2 Sign-In Feature**

### **Purpose**

Provide an interface for users to identify themselves to the system.

#### **Stimulus/Response Sequence**

The DJ will input their user ID and password

The system will validate and verify the user ID and password

The system will display the DJ home screen.

#### **Associated Functional Requirements**

##### **3.2.2.1 Functional Requirement (Sign Up)**

###### **Introduction**

User opens the program. A sign in page displays. The user clicks a button. A sign up page displays. The user enters valid information and clicks submit. The information is stored and the user proceeds to the sign in page.

Usernames contain only letters, numbers and underscores. They are no longer than 24 characters.

Passwords contain at least 6 characters and must contain at least 1 symbol, 1 letter, and 1 number. Passwords cannot contain the username, or part of the username if it is separated by an underscore.

###### **Inputs**

The user will type in a username and password.

The user will press enter, or click submit.

###### **Processing**

The username and password will be checked against an encrypted database for any duplicate entries. They will be checked for appropriate length and composition. (See introduction)

###### **Outputs**

A page will be displayed if the username and password are unique and correctly composed.

An error will be displayed if the username is not unique or composed correctly.

An error will be displayed if the password is not composed correctly.

##### **3.2.2.2 Functional Requirement (Sign In)**

###### **Introduction**

User opens the program. A sign in page displays. The user enters a username and password. The program will check the username and password. Upon validation the remaining features are accessible to the user.

###### **Inputs**

The user will type in a username and password.

The user will press enter, or click submit.

###### **Processing**

The username and password will be checked against an encrypted database for validity.

###### **Outputs**

A page will be displayed if the username and password are verified.

An error will be displayed if either the username or password are not verified.

##### **3.2.2.3 Functional Requirement (Edit Account)**

###### **Introduction**

User opens account settings page. User clicks a button labeled ‘edit’. User updates account information. User presses enter or clicks a button. Account information is updated in the database if it is valid. (See sign up)

###### **Inputs**

The user will type in a new password.

The user will press enter, or click submit.

###### **Processing**

The new password will be validated against the constraints specified in section 3.2.2.1.

###### **Outputs**

A message will be displayed if the password is verified.

An error will be displayed if the password is not verified.

### **3.2.3 Song Request Features**

#### 

#### **Purpose**

Provide an interface for users interact with song request features.

#### **Stimulus/Response Sequence**

DJ or Patron select song

Song is added to request queue

DJ and Patron view request queue

DJ marks song as rejected or accepted

-If rejected reason (lame or not in library) is selected

-If reason is not in library song is added to DJ needed list

#### **Associated Functional Requirements**

##### **3.2.3.1 View Song Request Queue Functional Requirement**

There must be a list of registered song titles from which the DJ and Patron can search for songs to request.

###### **Introduction**

The Event Manager and Patron will view a list of songs in the Events song request queue. The list will not reflect what songs the DJ has available but rather is an updated database of all possible songs that have been requested.

###### **Inputs**

The DJ or Patron will type the title of the desired song.

The DJ or Patron will select the correct song from a list of possible matches.

###### **Processing**

The song title will be searched for in the database song title list.

###### **Outputs**

A list of possible song matches will be displayed. If no possible matches are found then a prompt informing the user of the no match condition will appear.

##### **3.2.3.2 Find Search Song Functional Requirement**

###### **Introduction**

DJ and Event Manager will Search the Play My Song Database for current songs. This list does not reflect songs that the DJ has but is a way to easily search for and add songs to the request queue.

###### **Inputs**

The DJ or Patron will type the title of the desired song.

###### **Processing**

The song title will be searched for in the database song title list.

###### **Outputs**

A list of possible song matches will be displayed. If no possible matches are found then a prompt informing the user of the no match condition will appear.

##### **3.2.3.3 Request Song Functional Requirement**

###### **Introduction**

Event Manager or DJ will select a song to be added to the Event’s Request Queue

###### **Inputs**

The DJ and patrons will click a button from within the song search screen that will then add it to the queue.

###### **Processing**

System will see if the song already is in the queue, if not, the selected song will be added to the Event’s song request queue.

###### **Outputs**

System will notify user that the song has been added to the queue. If song has already been requested once then the system will alert the user that the song has already been added.

##### **3.2.3.4 Mark Song Functional Requirement**

###### **Introduction**

The DJ will open song request queue. Dj will select an option to mark a song. The DJ can mark a song in the following manner:

* Acknowledge(Accept) Song
* Reject/Needed Song
* Reject/Lame Song

###### **Inputs**

The DJ will select a song to mark.

###### **Processing**

System will handle the songs in the following manner:

* Acknowledge(Accept) Song - System will mark a song which signifies that it is in the database and that DJ will play (Marked Green or Check Mark)
* Reject/Needed Song - System will mark a song so that it signifies that the DJ does not have the song but needs to song. System will then add the song to the DJ’s Needed List.
* Reject/Lame Song - System will signify that the song will not be played and that the DJ

###### **Outputs**

A rejected song will display in a manner that suggests it has been rejected.  
After a time period of 2 minutes the song will be removed from the queue for that event permanently.

### **3.2.4 Needed List Feature**

### **Purpose**

Provide an interface that a DJ can store and interact with a list of songs to consider procuring.

#### **Stimulus/Response Sequence**

* The DJ will click a button to enter a view of the needed list.
* The DJ will click a button to search for a song in the database
* The DJ will click a button to add the song to the needed list found from the database.
* The DJ will click a button to reject a song from the request queue and system will automatically add the song the Needed list.
* The DJ will click a button to remove the song from the list

##### **3.2.4.1 View Needed List Functional Requirement**

DJ/ Event Manager will view list of songs in the Needed list

###### **Introduction**

DJ/ Event Manager will view the needed list which will be specific to that user. The view event button will be accessible from the DJ’s starting screen.

###### **Inputs**

User will click a button from the DJ portal to view the Needed List.

###### **Processing**

System will populate songs that are in the Users needed list.

###### **Outputs**

Event song title additions will be reflected when the needed list view is accessed.

##### **3.2.4.2 Add Song to Needed List Functional Requirement**

###### **Introduction**

Event Manager will add song to their Needed List after searching for the song through the Play My Song Music Database.

###### **Inputs**

DJ will click a button to select to add a song to the needed list.

###### **Processing**

System will add the selected song’s title and artist to the Needed list table for the user.

###### **Outputs**

System will display that the song has been added the their Needed List.

##### **3.2.4.3 Remove Song from Needed List Functional Requirement**

The DJ will remove a song from the needed list.

###### **Introduction**

The DJ will be able to remove selected songs from the needed list.

###### **Inputs**

The DJ will select a song from the needed list.

The DJ will confirm removal of selected song.

###### **Processing**

The needed list database will remove the song entry and update the view to reflect the deletion.

###### **Outputs**

System will display that the song has been removed from their Needed List.

Event song title additions will be reflected when the needed list view is accessed.

The updated list will be displayed upon changes from within the needed list view.

## **3.3 Performance Requirements**

This system supports at least 1,000,000,000 users at a time. Ambitious, I know. Updates to the request queue should be reflected within 3 seconds of the user's input.

## **3.4 Design Constraints**

The system will be constrained by current HTML5 and web platform service that are available at this time.

## **3.5 Software System Attributes**

### **3.5.1 Reliability**

This system will not have any errors. EVER.

### **3.5.2 Availability**

This system will be accessible at least 90% of the time on industry standard hardware running the latest OS for that hardware.

### **3.5.3 Security**

DJ user ids and passwords will be stored securely using cryptographic methods.

Payment information will also be stored using cryptographic methods.

### **3.5.4 Maintainability**

This system will be operational within ten minutes of a hardware failure. There will be database backups in place to ensure there is no data loss. These backups will be made every [insert time measurement] and a total of 3 backups will be kept at one time.

## **3.6 Logical Database Requirements**

Song data is stored with information about what event it was requested at. This data will be accessed and searched through frequently. Therefore this data will be searchable by the application in order for it to provide insight into trending songs.

Data will be retained indefinitely order to provide statistics for this application’s users.

## **3.7 Other Requirements**

No other requirements at this time.