Software Requirements Specification

for

SongHaven

**Version 1.2 approved**

**Prepared by :D**

**Eric Krenz**

**Christy King**

**Jordan Goetze**

**Corey Arneson**

**North Dakota State University**

**10/14/2014**

**Table of Contents**

**1. Introduction 1**

1.1 Purpose 3

1.2 Document Conventions 3

1.3 Intended Audience and Reading Suggestions 3

1.4 Project Scope 3

1.5 Definitions and Acronyms 4

1.6 References 4

**2. Overall Description 2**

2.1 Product Perspective 5

2.2 Product Features 7

2.3 User Classes and Characteristics 11

2.4 Operating Environment 12

2.5 Design and Implementation Constraints 12

2.6 User Documentation 12

2.7 Assumptions and Dependencies 12

**3. Functional Requirements 3**

3.1 Music Queue 13

3.2 Searching Music Library 14

3.3 New User Registration 14

3.4 Uploading to Music Library 15

3.5 Messaging Box 15

3.6 User Skip Votin 16

3.7 Music Library Management 16

3.8 User Account Management 17

3.9 Administrator Skip 17

3.10 AutoFill 19

**4. External Interface Requirements 4**

4.1 User Interfaces 20

4.2 Hardware Interfaces 20

4.3 Software Interfaces 20

4.4 Communications Interfaces 20

**5. Other Nonfunctional Requirements 5**

5.1 Performance Requirements 21

5.2 Safety Requirements 21

5.3 Security Requirements 21

5.4 Software Quality Attributes 22

**6. System Architecture** 23

**7. Test Plan** 23

**8. Project Management**

8.1 Updated Risk Management 24

8.2 Updated Project Plan 25

8.3 Meeting Minutes 25

8.4 Progress Report 25

**Revision History**

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Date** | **Reason For Changes** | **Version** |
| :D | 10/08/14 | Created | 1.0 |
| :D | 10/10/14 | Additional information added | 1.1 |
| :D | 10/13/14 | Additional information added | 1.2 |

# Introduction

## Purpose

The purpose of this project is to create a working system that will function as a radio station controlled by a web site titled SongHaven.

## Document Conventions

This document follows IEEE standards and conventions. Hierarchy is indicated by a numerical system and font sizing.

## Intended Audience and Reading Suggestions

This document will be written in such a fashion that it will be understandable by any person with a background in technology. However, someone familiar with business logic should be able to understand the document as well. The table of contents may be referenced for any reader looking for specific information, otherwise the authors suggest that the material is read in order.

Sections 1 and 2, the Introduction and Overall Description, are the most pertinent for the common SongHaven user. For technical information sections 3 through 7 should be utilized; Functional Requirements, External Interface Requirements, Nonfunctional Requirements, System Architecture, and the Test Plan. For the reader interested in project management, section 8 is the most pertinent; covering Risk Management, the Project Plan, Meeting Minutes, and the most current Progress Report.

## Project Scope

The goal of the software being created is to provide entertainment in a location where the server lies. A similar system, which has inspired this one, is currently in place in the ACM lounge.

However, there are some issues with the current system, Sonic Flow. Sonic Flow uses no authentication system, and audio stream based music sourcing. It is our hope that with the implementation of the ability to upload personal music files, all issues related to network connection will go away, and functionality will even remain during network outages.

User management and administration will be a key design principle behind SongHaven. Frequent issues arise with users submitting troll songs, such as Rick Astley’s “Never Gonna Give You Up” multiple times in a row. A system will need to be in place to track users who submit these songs and ban or time out the users who are abusing the system.

A chat room and message queue front in will be the main page that users interact with on the site. Users will be able to communicate and leave messages about songs as they are played, and the intended purpose is for the messages to persist even when the page is reloaded.

## Definitions and Acronyms

|  |  |
| --- | --- |
| **Term** | **Definition** |
| MP3 | MPEG-2 Audio Layer III |
| ASCAP | American Society of Composers, Authors, and Publishers |
| ACM | Association for Computing Machinery |
| Troll songs | A song with the intention of making other users unhappy or aggravated |
| IIS | Internet Information Services |
| NDSU | North Dakota State University |

## References

### GitHub Repository

"Jghibiki/SongHaven." *GitHub*. N.p., n.d. Web. 10 Oct. 2014.

<https://github.com/jghibiki/SongHaven.git>

### GitHub Tutorial

"Try Git." *Try Git*. N.p., n.d. Web. 10 Oct. 2014.

<http://try.github.io>

### ASCAP Licensing Information

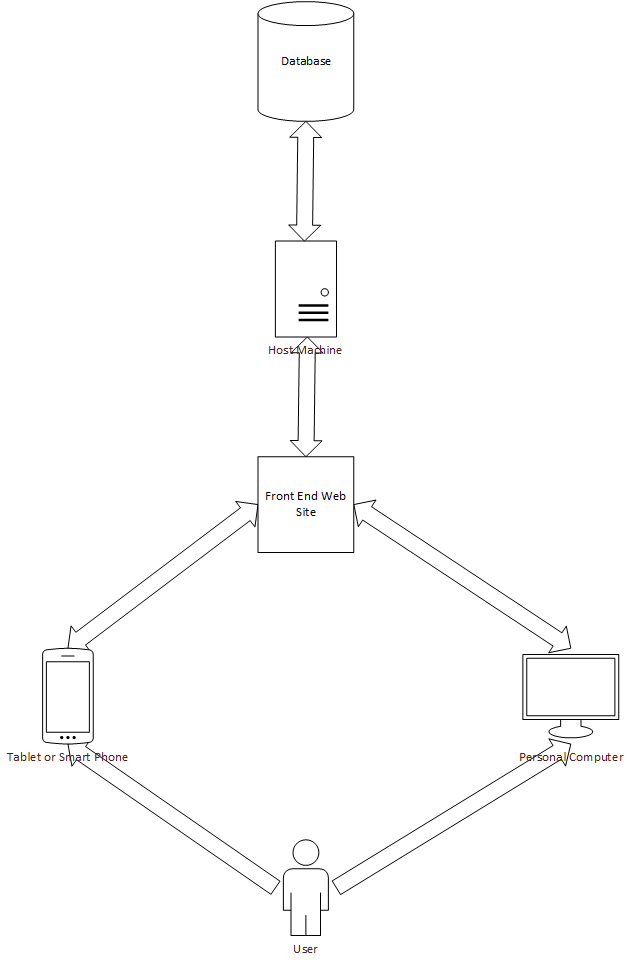
"ASCAP Licensing FAQs." *Www.ascap.com*. N.p., n.d. Web. 10 Oct. 2014.

<http://www.ascap.com/licensing/licensingfaq.html#general>

# Overall Description

## Product Perspective

SongHaven developed as a replacement solution to a relatively unstable software system named SonicFlow that is currently in use by the NDSU chapter of the ACM. This product comprises of a local music service that will play queued music on a localized server machine in the ACM Lounge area. Music can be queued up by users on web accessible devices with access to the SongHaven website. The music library on the server can also be broadened by allowing users to upload their own music files though the website. For more details see system overview below:



**Figure 1:** A Simple Diagram

## Product Features

### User Features:

#### Account Registration (Login Page):

Prospective Users will access the SongHaven homepage at: [www.songhaven.ndacm.org](http://www.songhaven.ndacm.org)

and will be directed to a Login page.This page will consist of a New User option and a Returning User option.

The Returning User section will have a Username and Password field as well as a Login button. The User will log in using these components to access any functionality of SongHaven.

The New User section will have fields for Email, Username, Password, Password Reentry, and Name. Once all fields are filled, the User can select the Register button and a new account will be created using the entered data. If there are invalid data in the fields or there is already an account with the desired Username, then the account will not be created and the User will be notified of the reason it was not created.

Once the account is registered, Users can proceed to all user features in SongHaven.

#### Music Queueing (Queue View Page):

The Music Queue will consist of a list of music files that have been queued by users who have logged into SongHaven. The music player will take the first music file from the queue and begin playback on that file.

A maximum number of songs may be added to the queue and this maximum size will be set by an Administrator.

When playback of the current file is completed or when a file’s playback is terminated by an Administrator Skip or User Skip Vote, that file will be unloaded from the music player, removed from the queue and each file in the queue will be promoted. The first song in the new queue will be loaded into the music player, and playback of the first file will begin.

#### Searching Music Library (Search Page):

A logged in User will have access to a search box that will search the Library for relevant music files to the user entered search terms. This page will consist of a Search Box with a Search button, a scrollable Results List below the search box, and several option buttons to the right of the list such as Add to Queue, Start a New Search, and Return to Queue View.

Search terms will consist of Text bounded to ASCII letters, symbols, and numerals. The searching protocol will search Music File Name first, then Group / Artist Name, then finally search the for a match in Album Name.

The Search will return a list of relevant matches and display them below the Search Box and the User can scroll the list to view the possible matches. If there are no matches or invalid input characters were detected the User will be redirected to the search box while displaying an error message.

ex.

“ No matches were found for the entered search criteria.”

or

“ Invalid search criteria was detected. Please try again using only valid characters, numbers, and symbols (! ? , . $ & : # )”

Once the results have been displayed in the list, the User can select any music file in the list and the details of the file will be displayed.

When the User selects a result in the list that file will become highlighted and the Add to Queue button will become active (not grayed-out). If the User selects this option then the music file will be added to the Queue.

If the User left-clicks outside of any interactive component, the current item in the results list will become unselected, and the Add to Queue option will become deactivated (grayed-out).

If at any time, the User selects the Start New Search option, then the current results list will be deleted and the Search Box will be cleared. Similarly, if the User selects Return to Queue View the search page will be cleared and the User will be redirected to the Queue View page.

#### Uploading Files to the Music Library (Upload Page):

Logged in Users can also select the Upload to Library option on the Queue View page, and then they will be redirected to the Upload Page.

The Upload Page will consist of a several data entry boxes, Cancel and Confirm buttons, as well as a Choose a File button and a Return to Queue View button.

The required fields will include:

* Music File Name
* Artist/Group Name
* Album Name

Other optional fields will include:

* Year of Release
* Publisher
* Recording Studio
* Search tags (country, edm, funny, intense, rock, etc.)

The User must also upload a local music file using the Choose a File option before confirming. When this option is selected, a File Browser will open allowing the user to navigate their local file system to find the intended music file. Once selected, the User will left-click the Ok button and the file will be added to the Upload Page.

Once the User has entered input into all of the required fields the Confirm option will become active (not grayed-out). The User can then select Confirm and the file will be uploaded to the Music Library. A confirmation message will then be displayed stating that the file was successfully uploaded. If there are invalid fields or the file was of an invalid file type, the confirmation will fail and a failure message will be displayed stating the reason the upload failed.

At any time the user can choose the Return to Queue View option and any current information in the fields will be cleared and the Queue View page will be loaded.

#### Messaging Box(Queue Page):

On the Queue Page, there will be a Messaging Box that will log all actions on SongHaven, including queueing of files, skip requests, admin skips, system actions, voting results, and user chat.

Any User that is currently logged in can type a message into the message line and select the Say option. Once the Say option has been selected the Message box will create a new line composed of “:- <User Name> User Message”.

Chat abuse is a problem that can be foreseen and will be handled by an Admin specified chat limit for each User Account. This will be a three stage setting of Unrestricted where a user can post as often as they want, Limited where the User can only post up to a certain times in a minute, or Revoked where the User cannot post any messages until an Admin changes the status manually.

#### User Skip Voting(Queue Page):

While logged in, Users will be able to start a Skip Vote on any music file currently in queue, including the current file being played by the music player. When a vote is created on a particular file, a vote message will be displayed in the Messaging Box consisting of the name of the file being skipped as well as the current number of votes.

ex.

“A User Skip Vote on ‘Music File Name’ has been created.

This vote will be concluded in ‘30’ seconds.

Please Vote Now (‘Skip’ or ‘Keep’)

Current vote status:

Skip : 1

Keep: 0 ”

Each vote will be open for user input from the Messaging Box for a specified amount of time. This time period will be set by an Administrator.

Users who are logged in will be allowed to reply to the vote message by typing in either a ‘skip’ or ‘keep’ command. The voting system will track all replies to the Messaging Box of whose message consisted of one these commands. Each time a vote is received a skip or keep counter will be incremented and the username of the voting user will be added to a ‘voted’ list and will not be accepted again until after the conclusion of the current vote.

At the conclusion of the voting time period, a comparison of the skip and keep counts will determine if the file will be removed from the queue and the results will be displayed in the Messaging Box.

ex.

“ The results of the vote are as follows:

Skip : 4

Keep: 2

The file ‘ Music File Name’ has been skipped“

If there are more skips than keeps, the file will be unloaded from the music player and removed from the queue. All files in the queue after the removed file will be promoted.

If there are more keeps than skips, the file will NOT be removed and the queue will remain unchanged. Similarly, in the case of a draw the file will NOT be removed from the queue.

Only one vote will be allowed to be executed at a time and another vote will not be allowed to start until the current vote is concluded.

**Administrator Features:**

#### Music Library Management (Admin Page):

The administrator will be able to make changes to the music database as s/he sees fit. If a song is being abused by a particular group of users, an administrator will be able to remove songs and block songs with that particular tag from being uploaded in the future. The administrator will likely have access to the host machine, so they will be able to transfer large amounts of files directly.

#### User Account Management(Admin Page):

#### The administrator will have full control over the other users in the system. They will be able to time out, ban, temporarily ban, and remove users from the system entirely. Users must wait out the ban or create a new account to continue using the service. For future plans, we are also looking at the ability to ban someones IP address if continued abuse occurs.

#### Administrator Skip (Admin Page):

An administrator who has logged into the system will be able to access the Music Queue and can ‘Skip’ any music file in the queue without the consent of users through the use of a vote. When an administrator skip is performed, a message will be displayed in the Messaging Box stating the name of the music file skipped. This file will be unloaded from the music player and removed from the queue and all files after the removed file will be promoted.

ex.

“Administrator has skipped ‘Music File Name’ and it has been removed from the Queue.”

**System Features:**

#### AutoFill:

An administrator who has logged into the system will be able to enable an AutoFill mode. When this mode is enabled the system will switch to a parallel queue of a random variety of songs if there are no songs currently in the queue. In this way SongHaven will continuously play music, regardless of if users are manually queueing it. The users may still manually add songs to the queue when this mode is activated. In this event the system will switch back to the User filled queue and play the added song immediately.

## User Classes and Characteristics

* Standard User
  + Queues up music
  + Uploads music to be played by the server
  + Interacts with other users via the website
  + Modifies meta-data
  + Can be banned or timed out
* Administrator
  + Can remove text from chat
  + Can remove songs from queue
  + Can ban, time out, or permanently ban users
  + Can remove songs from the database

## Operating Environment

The system will be using a server machine running Windows Server 2012 with IIS installed. The host machine will be running all of the web services, and will be hosting the web site. The website will be hosted through the ACM’s hosting and the URL will be ‘songhaven.ndacm.org.’ The project will be designed off of the MVC design pattern, and created using ASP.NET’s MVC 5 framework and using Razor view pages to generate HTML. The system must work alongside a media player of some sort in order to playback the songs that are selected to be played by the radio station.

## Design and Implementation Constraints

* **Hardware Limitations:** There is limited storage space on the machine, and therefore it will be limited to the amount of hard drive space available (internal and external additions). The server will only be able to host a certain amount of users, so it could overload if too many users try to access the service at the same time.
* **Software Limitations:** The software will be limited by the libraries available to the asp.net c# environment, and if other libraries are needed they will have to be created manually. The team has decided to switch to a Windows machine rather than Linux machine based on unforeseen complexities in implementing some of the more obscure libraries.
* **Human Limitations:** Although there are implemented spam prevention and moderation protocols in place, people will still find ways around them. Administrative functionality will not be accessible when an administrator is not currently logged in and this will allow Users to perform ‘abusive’ or ‘improper’ actions on the system with little to no realtime repercussions.
* **Music Distribution Limitations:** SongHaven is exempt of music licensing fees given that it remains a nonprofit educational project.

## User Documentation

For the third deliverable, due December 8th 2014, the team intends to create a user manual for the completed product. The user manual will inform users how to access and utilize SongHaven’s functions. The use of figures and diagrams will be added to assist in user understanding.

## Assumptions and Dependencies

The assumptions and dependencies are as follows.

* A stable line of power and Internet will stay attached to the host system.
* Only trustworthy administrators will be selected to moderate the system.
* The server will never run out of disk space.
* The server will have enough processing power to carry out all of the user requests in a reasonable amount of time.
* The operating system was installed correctly, and there will be no operating system level defects.
* The system will be used for non-profit educational purposes, therefore exempt from music licensing.

# Functional Requirements

## Music Queue

3.1.1 Description and Priority

The Music Queue is a HIGH priority component and consists of a Queue of Music Library files and a Music Player component that will handle the playback of the first file in the queue.

3.1.2 Stimulus/Response Sequences

* User logs into the SongHaven website.
* Queue Page is loaded by default.
* User views the Queue or accesses other features (see 3.2 - 3.6).
* System handles the updating of the Queue and the playback of the first file in the Queue. If AutoFill (see 3.9) is enable the System will add files to the queue as necessary to meet its limit.

3.1.3 Functional Requirements

MQ-1: A queue component will accept a music file from the library and add it to the queue.

MQ-2: The music player component will access the first file in the queue component and will begin playback of the file.

MQ-3: The queue will have an Admin set maximum size and Users cannot add files once that size has been reached.

MQ-4: The music player will remove the current file from the queue when playback has finished.

MQ-5: The queue and music player components will accept a skip command on a given queue position and will remove that file from the queue or music player.

MQ-6: If AutoFill is enabled by the Admin and the queue is empty, the Music player will request a random file from the Library and start playback on that file.

MG-7: If AutoFill is enabled and a User queues a file, the currently file in the music player will fade out and the first file in the queue will be loaded into the music player and begin playback.

## Searching Music Library

* + 1. Description and Priority

The priority on searching the music library is medium high to high. Our system will be able to function without the ability to perform a search, but it will be very hard to queue up a song without first knowing what songs are in the library.

* + 1. Stimulus/Response Sequences
* User logs into the SongHaven website.
* User views queue and realizes that s/he may add his/her favorite song to the queue.
* User uses the library search function.
* User enters a search term relevant to his/her interest.
* System handles return of relevant data to user.
* User chooses result to add to queue.
* System adds chosen element to queue.
  + 1. Functional Requirements

SML-1: A search will accept any number of characters and digits, including most ASCII characters and some recognized symbols.

SML-2: A search will query all fields of the database (universal search) and return the most relevant information to the user.

SML-3: When a returned song is selected by the user, it will be added to the end of the queue by the system.

## New User Registration

* + 1. Description and Priority

New users will use this feature in order to use the service. This is how they will be able to authenticate against the service. The priority on this feature is high, because it is a core feature of the application

* + 1. Stimulus/Response Sequences
* User enters the web site
* User realizes that s/he cannot access and of the features because s/he is not registered
* User decides that they wish to access these features and registers for the web site
* User fills in required forms
* system adds user to database
  + 1. Functional Requirements

NuR-1: System will be able to validate whether or not the username has been used before

NuR-2: System will give user access to core site features once their new account is validated

NuR-3: Database is updated with new user immidiately after form is filled out and fields are validated

NuR-4: User will be able to be banned or timed out at admin’s discretion once they start using the website.

## Uploading to Music Library

* + 1. Description and Priority

The priority on uploading the music library is medium. Our system will be able to function without the ability for users to upload, but we will be limited to what we add to the machine manually with flash drives

* + 1. Stimulus/Response Sequences
* User realizes s/he needs to add their song in order to play it.
* User clicks the upload a song button.
* User inputs their file.
* System downloads file from user, and places it into the library folder.
  + 1. Functional Requirements

UML-1: The system will check and make sure the chosen file is of a valid format.

UML-2: The system will validate whether or not the file has a valid file name.

UML-3: The system will be able to play the song, and the search will be able to find the song, once it is uploaded immediately.

## Messaging Box

* + 1. Description and Priority  
        The priority on a messaging box is low. It is not a core feature of the product.
    2. Stimulus/Response Sequences
* User logs into the website.
* User views existing messages in the box.
* User may choose to respond to the messages in the box, or make an original comment.
* Admins may remove messages, and ban users based on their comments.
  + 1. Functional Requirements

MsgB-1: The system will filter only accept messages of a certain length.

MsgB-2: The system will not accept erroneous characters.

MsgB-3: The system will display who posted the comment and when.

MsgB-4: The messaging box will remove comments after a certain number of new ones have been entered.

## User Skip Voting

* + 1. Description and Priority

The priority on the User skip voting component is low. It is not a core feature, but will be implemented if at all possible as it fixes issues with the system that is being replaced.

* + 1. Stimulus/Response Sequences
* User logs into the website.
* User views the queue and finds a file to be objectionable.
* User Selects the Skip Vote option and chooses the file that they want skipped.
* The System notifies all Users through the messaging box that a Skip Vote has been requested on that file.
* Users then can type in either the commands ‘Skip’ or ‘Keep’ and the updated vote tally is displayed in the messaging box without displaying the User name or command choice.
* The System will accept commands from Users for a time period set by an Admin.
* The System will display the results in the messaging box and if necessary send a skip command to the Music Player, when the time period has elapsed.
  + 1. Functional Requirements

USV-1: The System will log a skip vote on a particular file in the queue when a User requests it.

USv-2: The System will only allow one skip request to be actively voted on within one voting period.

USV-3: The System will only accept votes from logged in Users through ‘Skip’ and ‘Keep’ commands in the Messaging box.

USV-4: The System will log all votes in a skip and keep values, as well as track the usernames of voters in a voted list.

USV-5: The System will only allow Users whose username is not in the current voted list to actually increment the vote values.

USV-6: The System will only accept votes within a admin set period of time and will send a skip request if necessary to the music player at the end of the voting period.

USV-7: The System will clear the skip, keep, elapsed time, voters list, and target file values when the current vote is complete and will then be able to accept another vote request.

## Music Library Management

* + 1. Description and Priority

Administrator ability for Music Library Management is of medium high priority. The Administrator’s should be able to access the music database and edit as needed.

* + 1. Stimulus/Response Sequences
* Administrator logs into the website.
* Administrator accesses Music Library.
* Administrator edits or deletes files as needed.
  + 1. Functional Requirements

MLM-1: Only the Administrators will be able to manage the music library.

MLM-2: An Administrator may edit a song’s information.

MLM-3: An Administrator may delete a song from the music library.

## User Account Management

* + 1. Description and Priority

The priority for Administrator’s ability to manage User accounts is high. The system SongHaven is replacing, SonicFlow, did not have a user authentication system in place, thus the repeated playing of troll songs was not preventable.

* + 1. Stimulus/Response Sequences
* Administrator logs into the website.
* Administrator selects User from database.
* Administrator completes some action on User account: time out, ban, temporarily remove user.
  + 1. Functional Requirements

UAM-1: Only the Administrator may apply these restrictions to User accounts.

UAM-2: The system will deny restricted users according to action applied.

## Administrator Skip

* + 1. Description and Priority

Administrator skip is the functionality for a logged in and authenticated administrator to skip any song currently playing. This will be in place to skip songs when there are not enough people around to vote, or a group of individuals are abusing the system simultaneously and consist of the majority. Priority for the feature is low, it is not a core feature of the system.

* + 1. Stimulus/Response Sequences
* Administrator logs into the website.
* Administrator wishes to skip the current song.
* Administrator presses his skip button.
* The system recieves the action and performs the skip.
  + 1. Functional Requirements

ASkiP-1: The system will only allow this feature to be performed if the user is an authenticated administrator.

ASkip-1: The system will call the skip function directly on the currently playing song immediately when the skip is chosen.

## AutoFill

* + 1. Description and Priority

The priority of the AutoFill mode is medium. The system will be able to function without randomly queued music, but unless Users are queueing manually, the system will not play anything.

* + 1. Stimulus/Response Sequences
* Administrator logs into SongHaven website.
* Queue Page is loaded by default.
* Administrator enables AutoFill mode.
* System fades out to a parallel mode that randomly selects music from the library.
* If a song is manually queued the system will fade back in to the User defined queue.
  + 1. Functional Requirements

AFR-1: If AutoFill is enabled by the Admin and the queue is empty, the Music player will request a random file from the Library and start playback on that file.

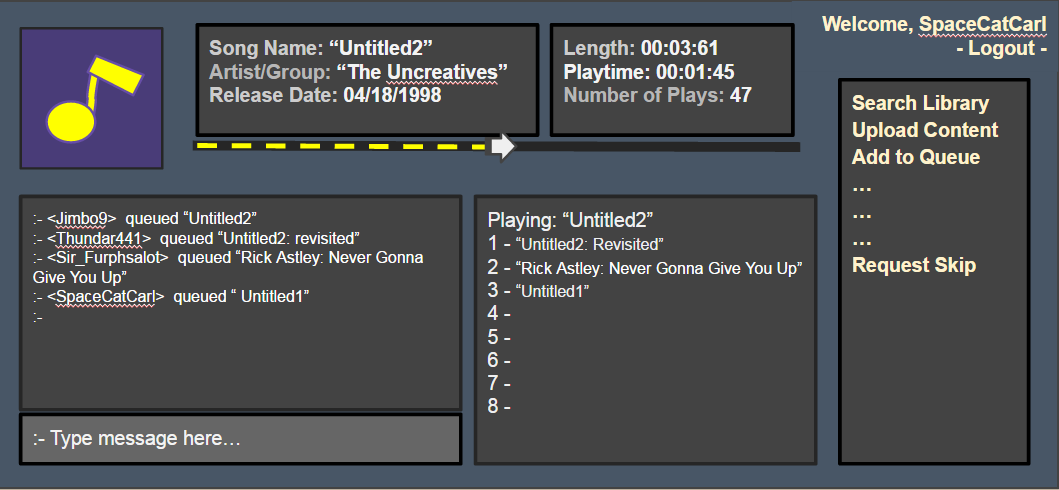
AFR-2: The music player components will accept a skip command on a given queue position and will remove that file from the queue or music player.

AFR-3: If AutoFill is enabled and a User queues a file, the currently file in the music player will fade out and the first file in the queue will be loaded into the music player and begin playback.

AFR-4: The music player component will access the first file in the queue component and will begin playback of the file.

# External Interface Requirements

## User Interfaces



**Figure #:** User Interface Mock-up

## Hardware Interfaces

The hardware required for this system will consist on a user’s personal device, a host machine (in our case this is a computer borrowed from the school.), and speakers to play the music being hosted.

## Software Interfaces

The web interface uses IIS to host html pages and web services. The interface will have a home page that displays the current queue of songs and the song that is currently playing. There will be a search page that allows users to search for a song, then the interface will display the search results. From this point the user may request a displayed song to be played next. There will be a page that allows users to upload songs to the database. There will be a library page that allows users to browse the music library.

## Communications Interfaces

Communications will be relayed over HTTP and HTTPS via forms and get/post requests made by the front end to the back end controller. A user will use a web browser to control the host machine, and the host machine will relay back music to those who are present in it’s area of effect. Administration services will only be available if a user is logged into a validated account, and will be protected as such. Database calls will be made locally, and therefore will not require any kind of secure line.

# Other Nonfunctional Requirements

## Performance Requirements

Our machine will be required to maintain functionality and not slow down even when various different users are accessing the system simultaneously. The database queries must be optimized in such a way that there will not be any bottlenecks on the queries, however this should not be an issue with the database located on the same machine as the web server.

## Safety Requirements

The prospect of any loss, damage, or harm coming from the use of SongHaven is unlikely. The server host machine, however, does need to be set up to avoid fire and electrical hazards. In addition, users of SongHaven should take into consideration the possibility of hearing loss from prolonged or loud exposure to music.

## Security Requirements

* + 1. Administrative Authentication

As this system will be using claim based authentication, the authentication process for an administrator is no different than for a standard user.

* + 1. User Identity Authentication

When a user logs, in the programmatic background for the system will receive an OAuth token which will provide information about the user. When a user tries to do something on the site, the system will use this token to verify that the user account has been assigned the required scopes/claims (symbolic entities which represent permission to exhaust a resource). If the user has the required claim, they are allowed to exhaust the resource, else they are denied access to the resource.

* + 1. Music Licensing

According to the American Society of Composers, Authors, and Publishers, also known as ASCAP, the use of a song does not need to be licensed if it is part of a teaching activity for non-profit educational purposes. Since SongHaven is a product designed for NDSU’s 413 Software Engineering class, it falls into the category of educational. In addition, the creators of SongHaven, nor NDSU, will be profiting from the product.

However, this product would need to be licensed if it was to be used commerically. This is because it would fall under the category of a ‘public performance.’ ASCAP defines a public performance as:

“...one that occurs either in a public place or any place where people gather. A public performance is also one that is transmitted to the public…”

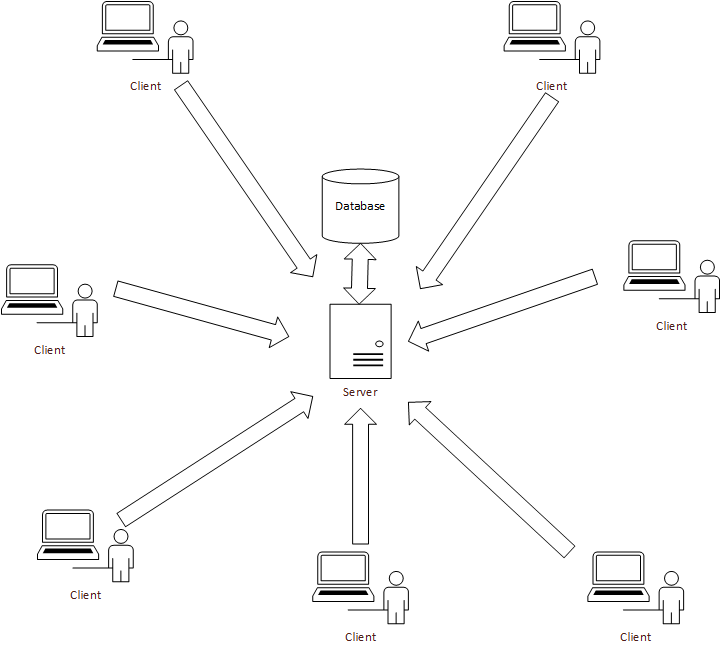
In addition, licensing would be required if the music is used as part of training seminars, conventions, or other commercial or business presentations.

## Software Quality Attributes

Quality Attributes in decreasing priority:

* Maintainability of System Components (Developer and Admin)
* Reliability of the System Components (Developer, Admin, and User)
* Availability of the System to Users (User)
* Robustness of Exception Handling (Developer, Admin, and User)
* Ease of Use (User and Admin)

# System Architecture



# Test Plan

This service will be tested using a series of unit tests running periodically on a build server. The web interfaces will be manually, and procedurally functionally tested. The production code is running at <http://songhaven.saves-the-whales.com/songhaven> and the build server can be seen at <http://songhaven.saves-the-whales.com:8080/>

# Project Management

## Updated Risk Management

One risk for the team is a variety of experience levels, thus limiting what certain members are able to contribute. Prevention strategies for this risk include completing tutorials individually or as a group on areas of weakness. In addition, team members are going to keep open channels of communication in case assistance is needed or questions arise. Resources found helpful by team members will be shared with the group for reference.

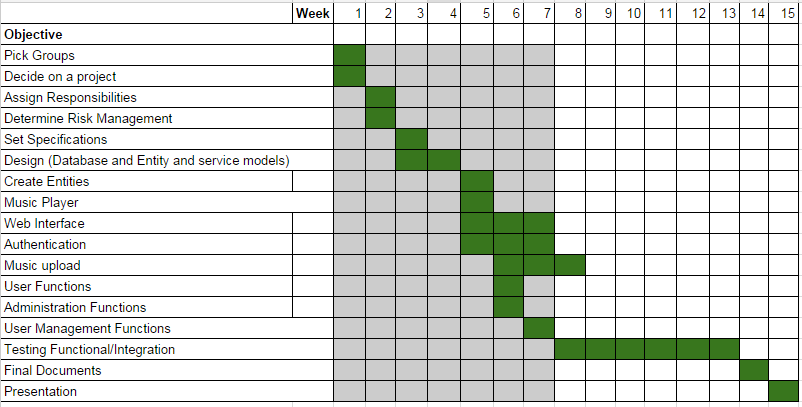
Another risk throughout this project is the possibility of personnel issues. Personnel issues may arise when a team member decides to drop the course or is unable to attend meeting. This risk may be prevented with proper communication between parties. In the regards to meeting attendance, the team has mostly utilized texting and Facebook messaging to keep in touch. In the event that a group member does drop the class, the individual will make all work and documentation available to their teammates.

Hidden complexity is a possible risk for this project. Hidden complexity entails the project being more complicated than previously thought or planned for. The main risk is that the team may become stuck solving complex problems in order to implement basic functionality. This risk could cause the project to fall behind schedule. In order to prevent this risk the team will stay on top of the schedule previously defined, and not put off important work. In addition, all team members will work to become familiar with the system’s architecture in an attempt to predict any issues caused by choice of framework.

The team may run into other scheduling issues, due to other pressing obligations and constrained meeting availability. In order to apprehend these risks, the team will work to be in contact outside of official meeting times and keep the team calendar up to date. If needed, the team will devise a strategy to split up tasks so that unavailable members are still able to assist outside of meetings.

Lastly, the team may run into unexpected project fault. The main risk is that functionality could cease to exist if the project is changed. In some situations, the system may be unrecoverable. It is possible that the system may also be corrupted by user error and misuse. The prevention strategies for this risk is to keep logs of all processes and database transactions. With the aid of GitHub, the team will keep a version history and always have available backups via the repository.

## Updated Project Plan



**Figure #:** Updated Gantt Chart

## Meeting Minutes

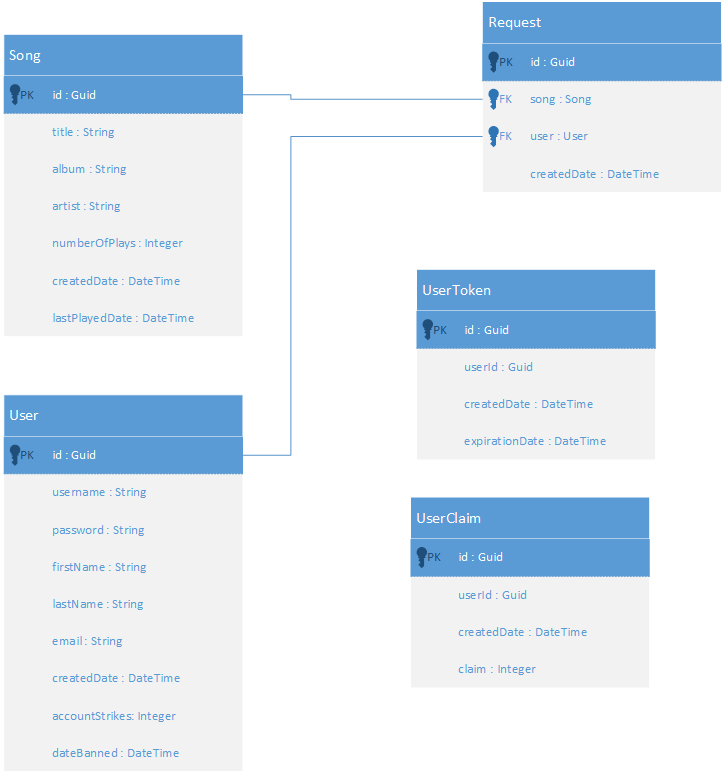
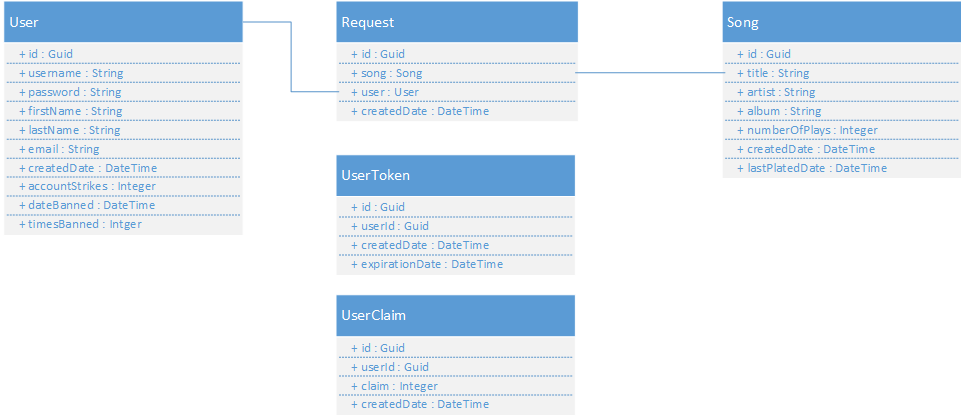
All :D Meeting Minutes can be found in the repository.

## Progress Report

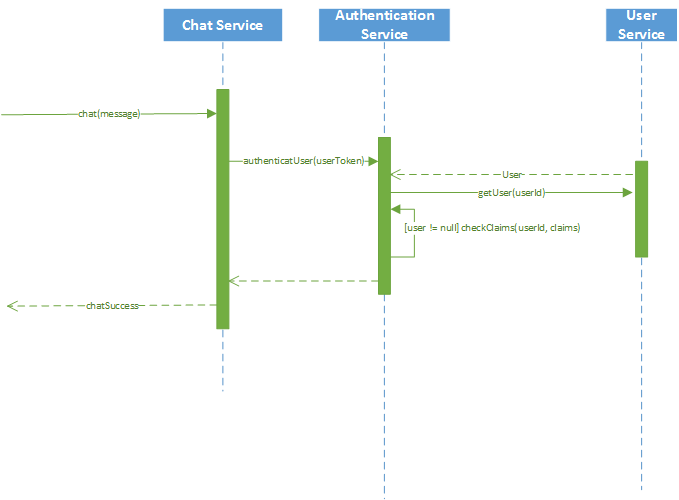
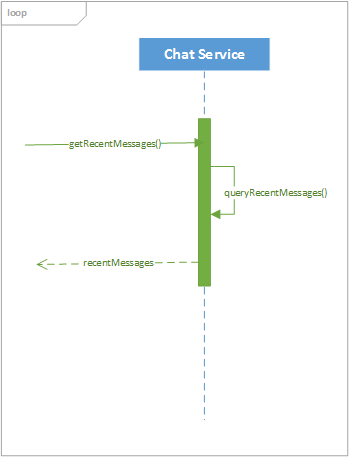
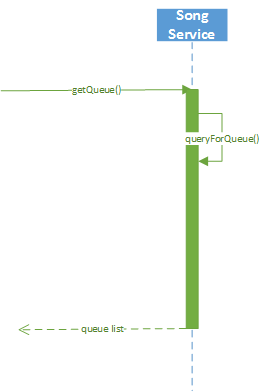
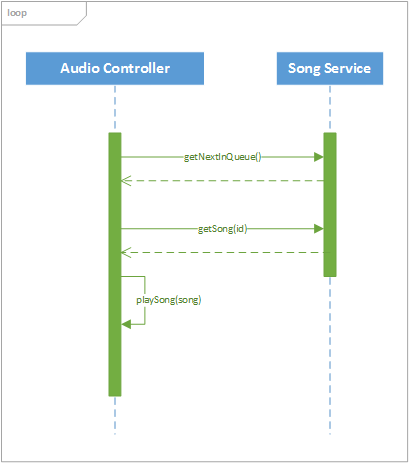
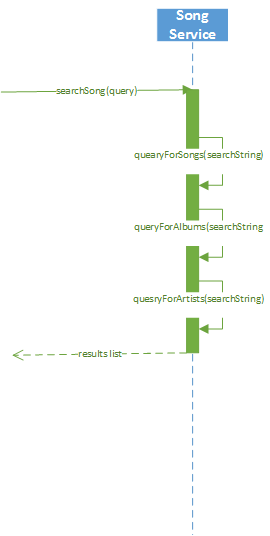
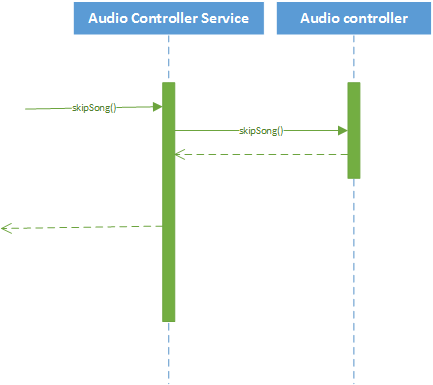
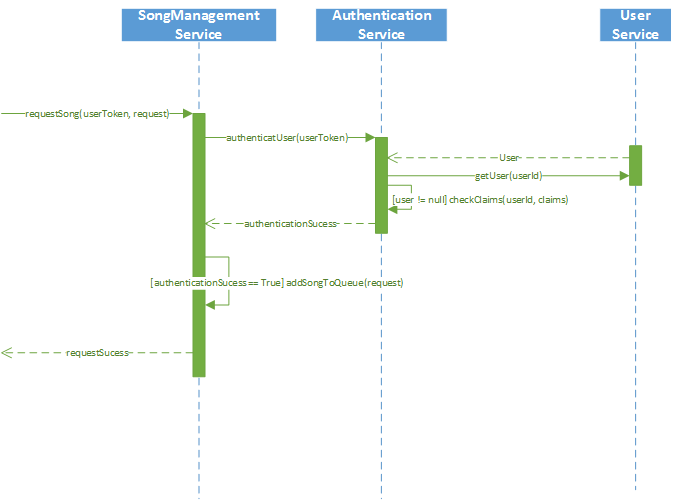
Soon, we will have a working exposed server. We have chosen to rewrite our application in a different framework because of the difficulty involved with setting up our previous option. Our databases are currently modeled and on a working MySQL database on our machine, and we have our use cases analyzed and ready to be written.

**Appendix B: Analysis Models**

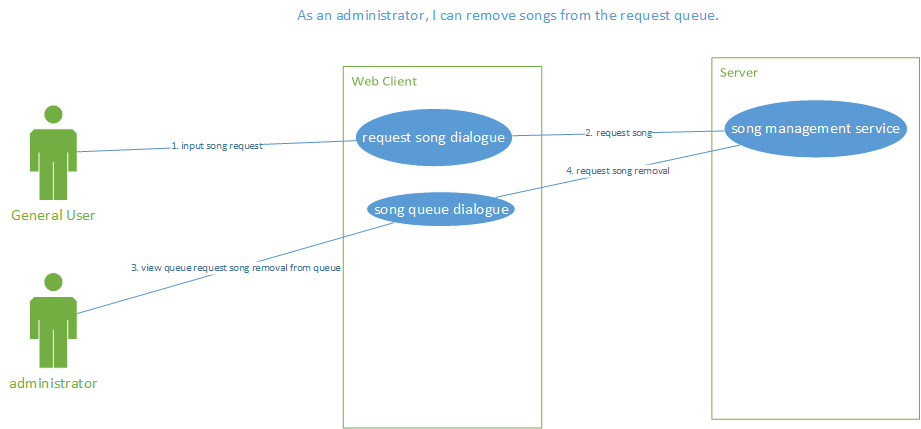
Class Model Diagrams:



Sequence Diagrams:



Use Case Diagram:



**Appendix C: Issues List**

Whether or not to include some kind of online streaming provider.

Whether or not to include some kind of meta-data reference provider.