**ITCS 6112 Fall 2017**

# Automatic Playlist Recommendation System

**Software Requirements Specification**

**Indrajeet Mishra, Sairam Rajagopalan, Priyanka Sharma, Dhiksha Ramkumar, Elizabeth Thomas**

**Version: 1.0** **Date: 11/15/2017**

## Table of Contents

# 1. Introduction

## 1.1 Purpose

The purpose of this document is to provide a detailed description of the Automatic Playlist Recommendation System. This will include discussion of the services the sytem is designed to provide, typical system users, and the constraints under which the system will operate. This document is designed to facilitate an understanding of the services that the software will provide, for both stakeholders and developers.

## 1.2 Scope

The automatic playlist recommendation system will provide users with a set of song recommendations, based on their previous listening history. More specifically, it will analyze their previous listening history in comparison with other users history to find song recommendations. The criteria for recommendations will be based on related user history of songs, specifically, it will recommend playlists based on the users top 5 most-listened songs. A playlist based on popularity of songs will also be given.

The server side of the software will consist of data analysis and algorithms to choose the recommendation results. The client side of the system will include a web application graphical interface in which users can receive recommendations.

The software will require users to have internet connection to access it. The software will interact with a collection of music data, detailing the songs each user has listened to as well as characteristics such as genre, artist, and user. This information will be stored in a database.

The end goal will be a service to users that improves their ability to find songs they may like, with ease and efficiency. This will broaden user music tastes and save users the time it would have taken for them to explore new music on their own.

## 1.3 Glossary

| Term | Definition |
|---|---|
| Software Requirements Specification | A document that completely describes all of the functions of a proposed system and the constraints under which it must operate. For example, this document. |
| Stakeholder | Any person with an interest in the project who is not a developer. |
| Collaborative Filtering | A technique used to provide personalized recommendations, wherein user preferences are compared with similar preferences of other users. |
| Admin | A user who holds special priveleges over the software, specifically, the ability to access the database. |
| Active User | A user who is accessing and using the system at a current |

| | |
|---|---|
| | moment in time, in order to receive music recommendations. |

## 1.4 References

IEEE. *IEEE Std 830-1998 IEEE Recommended Practice for Software Requirements Specifications.* IEEE Computer Society, 1998.

## 1.5 Overview

The following section, Overall Description, provides and explanation of the general functionality of the system. It is designed to give a holistic view of the services the system will provide in an easy to understand fashion. It will give context to the remainder of the document, especially for stakeholders who may not have a technical background.

The following section, Requirements Specification, is a technical description of the product and its constraints. It is designed to assist developers in their work, but should be intelligible for all readers of the document.

Both sections will serve the same purpose; that is, explicitly describing the system’s functionality, users, and constraints. However, each section has an intended audience, and therefore the two sections may use different types of description.

The final section will state the prioritization of requirements as well as the rationale behind the priorities chosen for each requirement.

# 2. The Overall Description

This section will provide a high-level, natural-language description of the services that the software will provide. It takes a holistic view and can serve as a context for further reading into the technical specifications (section 3).

## 2.1 Product Perspective

The playlist recommendation system will provide users with a group of songs based on their preferences and listening history. Users will be able to interact with the system through a web application, therefore users will require a personal device with internet connection. The goal is to help users discover new music.

For this system, there is a single actor, the user, who will use the product to gain music recommendations. The user will interact with a single system through the internet. The system will use the listening data associated with the active user and the

listening data of other users to produce a list of songs. See the diagrams below for a general overview of the system functionality.
An ideal future goal for this system is to be integrated with a music listening system and to dynamically collect data from the listening system. For now, however, the scope of the recommendation system does not include a music listening system. It will be considered as a stand-alone software.

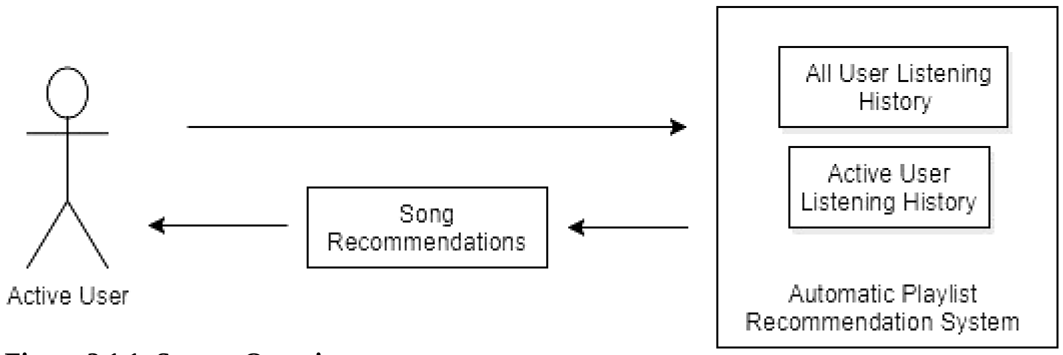

All User Listening History
Active User Listening History
Song Recommendations
Active User
Automatic Playlist Recommendation System

**Figure 2.1.1: System Overview**

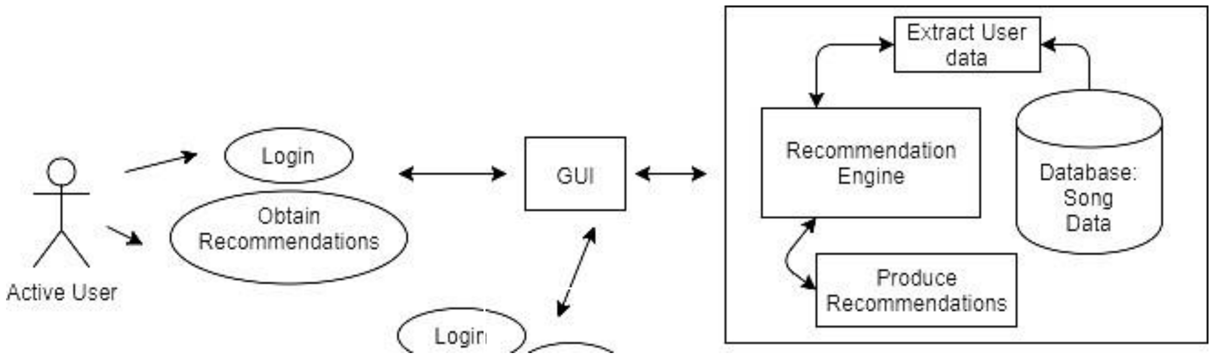


**Figure 2.1.2: Detailed System Overview 1**

### 2.1.1 System Interfaces

The system will need to interact with user data, stored in a database, in order to perform operations that will produce song recommendations for users. This data will be processed to produce the recommendations.
The system will be implemented as a web application, and it will therefore interact with different user browsers and operating systems.

### 2.1.2 User Interfaces

The system will only have one type of user. The user will perform some type of initiation of the software, such as logging in. Afterwards, the active user data will be retrieved and recommendations will be made based on specified algorithms. These operations will be carried out in the background, so that the user is not made aware of specific system operations. After the recommendations for the active user are determined, the system should display the user recommendations on the interface.

### 2.1.3 Hardware Interfaces

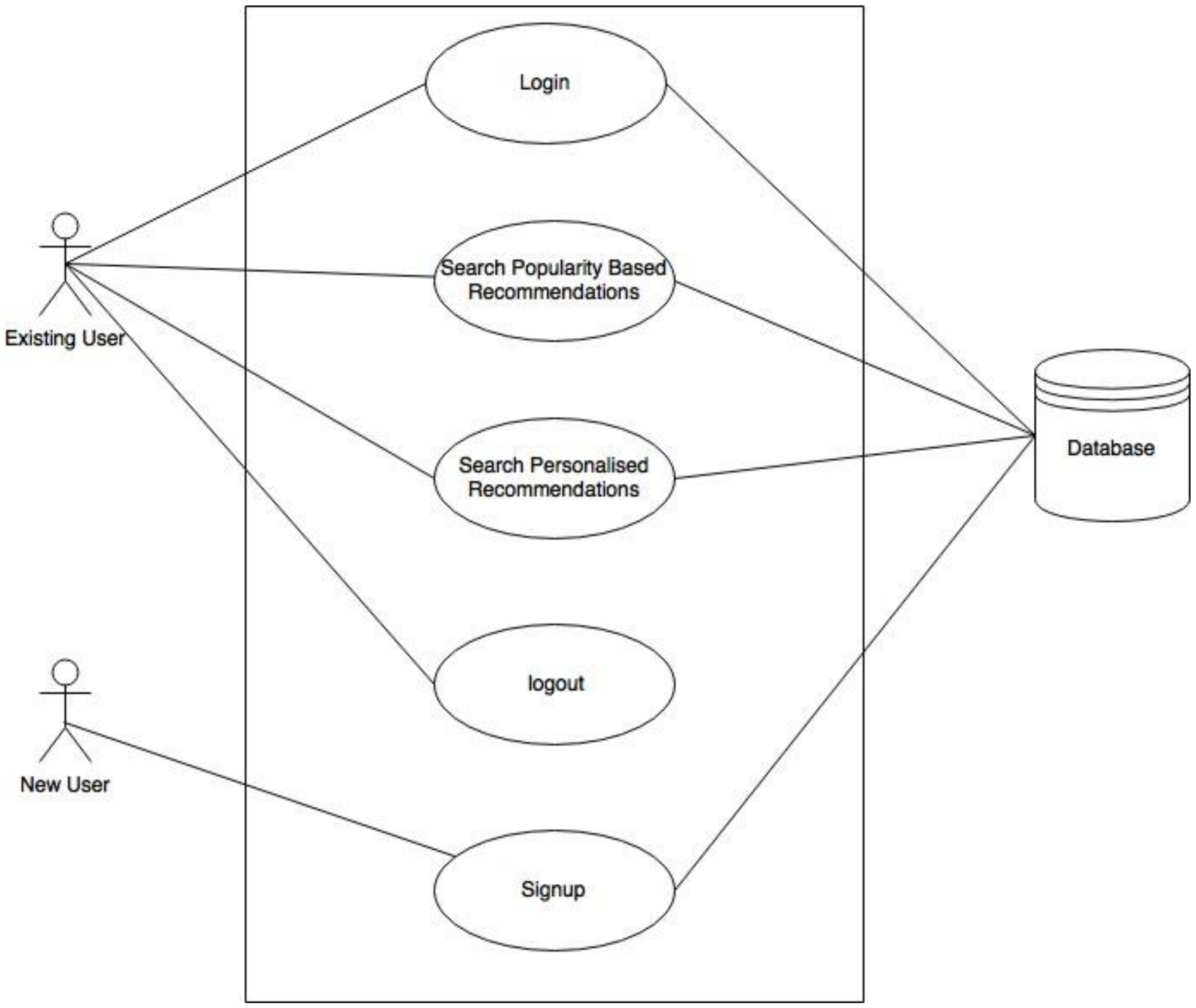
The software will run on any internet connected device, independent of hardware.

### 2.1.4 Software Interfaces

The software will run on an internet connected device, independent of operating system or browser. We will assume that the active user's device has some minimum functionality of processing speed and connectivity speed.

## 2.2 Product Functions

**Use Case Diagram:**


Login
Existing User
Search Popularity Based Recommendations
Search Personalised Recommendations
Database
logout
New User
Signup

**Use Case: Known User Login**
**Brief Description:**
The active user will input information and be taken to his or her page.
**Step-by-Step Description:**

1. The user logs in by entering username and password.
2. The system verifies the user is in the database.

3. The system displays a page specific to the user.

**Use Case: Known User Logout**
**Brief Description:**
The active user ends a session with the system.
**Step-by-Step Description:**
1. The user selects log out.
2. The system ends the particular user session.
3. The system displays the homepage.

**Use Case: Known User Search For Recommendations**
**Brief Description:**
The user will select a button to receive his or her recommendations. The system will display recommendations.
**Step-by-Step Description:**
1. The active user, who is logged in, will select to receive his or her recommendations.
2. The system will calculate recommendations.
3. The system will display song recommendations.

**Use Case: New User Sign Up**
**Brief Description:**
A new user creates an account with the system.
**Step-by-Step Description:**
1. The user, who is not already in the database, will enter his or her information.
2. The system will add the user information to the database.
3. The system will display a success message for the user.
4. The system will take the user to his or her page.

****Use Case: Admin Add song to Database (Relegated to future versions)**
**Brief Description:**
The admin will add a song the to the database.
**Step-by-Step Description:**

****Use Case: Admin Delete song from Database (Relegated to future versions)**
**Brief Description:**
The admin will delete a song from the database.
**Step-by-Step Description:**

****Use Case: Known User Subscribe to an Artist (Relegated to future versions)**
**Brief Description:**
The active user will choose an artist to follow. The system will display a song by that artist on the user’s page.
**Step-by-Step Description:**
1. The active user will search for an artist by name.
2. The system will search for that artist in the database.
3. If the artist is in the database, the system will display a success message to the user.
4. The system will show a song from the artist on the user’s page.

## 2.3 User Characteristics

The active users of the system are expected to be internet literate. They should have an interest in pursuing new music that may be similar to their recent tastes. Examples of potential entities who may use the software are fitness instructors, educators, therapists, personal music enthusiests.

## 2.4 Constraints

It will be important to secure user history data from outside parties, due to privacy concerns.
The system will need to have relatively high availability in order to gain notoriety and be competitive in its league. If users regularly have problems accessing the system, it will not be useful.
The system shall be relatively easy to operate, as users will not want to spend excessive time learning how to use the software.
The internet connection of users will be a constraint for accessibility of the system, since the application will require internet to function.
The ability for the software to produce recommendations will be constrained by the data within the system. If a user wishes to obtain a personalized playlist recommendation, he or she must have music history data in the system for this operation to succeed. If no prior history is known, the user will only receive popularity recommendations.
A primary concern is the accuracy of the song recommendations generated. The handling of data and algorithm design must produce accurate results.

## 2.5 Assumptions and Dependencies

The system relies on a set of user listening data. Due to the static nature of this data, new users who register an account will not be able to receive recommendations based on listening history, because they do not have listening history recorded in the system. Our assumption is that some day the system can be modified to interact with a music player software system, which will allow the system to collect dynamic data. Until that point, new users will be recommended songs based on a slightly different (popularity based) schema.

## 2.6 Apportioning of Requirements.

In the event that the project is delayed or finished without extra time, certain requirements or features may be relegated to later releases or abandoned completely. Decisions about requirements abandonment or delay will be made based on the prioritization as outlined later in this document (see section 4).

Our final system, at the current moment in time, will not include admin functionality or the “subscribe to an artist” feature. These system functionalities were chosen to be temporarily put on hold by our team, due to task prioritization and time constraints. Please see section 4 for further details.

# 3. Specific Requirements

This section provides a detailed and technical description of the system functionality and constraints. It is designed for developer use, but may be accessed by other stakeholders if desired.

## 3.1 Functional Requirements

The following content presents, in detail, the services that the system should provide. Diagrams have been provided for explicit understanding.

### 3.1.1 Detailed Use Case Elaboration

**3.1.1.1 Use Case: Search For Recommendations**

| | |
|---|---|
| **Use Case Name** | Search for Recommendations |
| **XRef** | Section 2.1 |
| **Trigger** | The active user, after logging in, requests a recommendation. |
| **Precondition** | The active user has music history data stored within the system, which can be referenced to produce recommendations. |
| **Basic Path** | 1. The system identifies the user and locates listening history.<br>2. Using collaborative filtering, the system compares active user music data with other users' music data to produce song recommendations.<br>3. The system provides the active user with an output list of song titles and corresponding artists. |
| **Alternative Paths** | **Potententially: If the user chooses, he or she may enter a specific song and receive recommendations. |
| **Postcondition** | The user receives recommendations. |
| **Exception Paths** | The attempt may be abandoned at any time. |
| **Other** | None |

**3.1.1.2 Use Case: Login**

| | |
|---|---|
| **Use Case Name** | Login |
| **XRef** | Section 2.1 |

| | |
|---|---|
| **Trigger** | The active user enters the system by entering a username and password. |
| **Precondition** | The active user has signed up as a user, and his or her information is in the database. |
| **Basic Path** | 1. The active user enters his or her username into the system.<br>2. The active user enters his or her password into the system.<br>3. The system identifies the user, using the information in the database.<br>4. If the user has not yet signed up and therefore is not in the database, the system will display a message.<br>5. If the user is in the database, the system will display a page where the user may take further action. |
| **Alternative Paths** | N/A |
| **Postcondition** | The user is able to view and interact with the system. |
| **Exception Paths** | The user may not have signed up, so his information may not be in the system. |
| **Other** | If user has not signed up, prompt that user to signup page. |

**3.1.1.3 Use Case: Logout**

| | |
|---|---|
| **Use Case Name** | Logout |
| **XRef** | Section 2.1 |
| **Trigger** | The active user leaves the system by clicking logout button. |
| **Precondition** | The active user is logged in currently. |
| **Basic Path** | 1. The active user clicks the logout button.<br>2. The system ends the user’s session and displays the home view (login page). |
| **Alternative Paths** | The session may expire after a period of time, logging the user out (without button trigger) |
| **Postcondition** | The login view is shown. The active user is no longer logged in. |
| **Exception Paths** | N/A |
| **Other** | None |

**3.1.1.4 **Use Case: Subscribe to Artist (Relegated to later versions)**

| | |
|---|---|
| **Use Case Name** | Subscribe to Artist |
| **XRef** | Section 2.1 |
| **Trigger** | The active user searches an artist and chooses to follow (subscribe) to that artist. |
| **Precondition** | The active user has not already subscribed to the particular artist. The active user is logged in. |
| **Basic Path** | 1. The active user searches an artist.<br>2. The active user selects an artist and selects the "follow" button.<br>3. |
| **Alternative Paths** | |
| **Postcondition** | The user receives random song recommendations from that artist whenever he or she logs in. |
| **Exception Paths** | The attempt may be abandoned at any time. |
| **Other** | None |

**3.1.1.5 Use Case: Signup**

| | |
|---|---|
| **Use Case Name** | Signup |
| **XRef** | Section 2.1 |
| **Trigger** | The active user registers with the system. |
| **Precondition** | The active user does not already have an account with the system. |
| **Basic Path** | 1. The active user selects "signup"<br>2. The active user enters his or her information into the system.<br>3. The system stores the active user's information in the database for future login. |
| **Alternative Paths** | N/A |
| **Postcondition** | The user now has an account and may use the system. |
| **Exception Paths** | The attempt may be abandoned at any time. |
| **Other** | None |

**3.1.1.6 **Use Case: Add Song to Database (Relegated to later versions)**

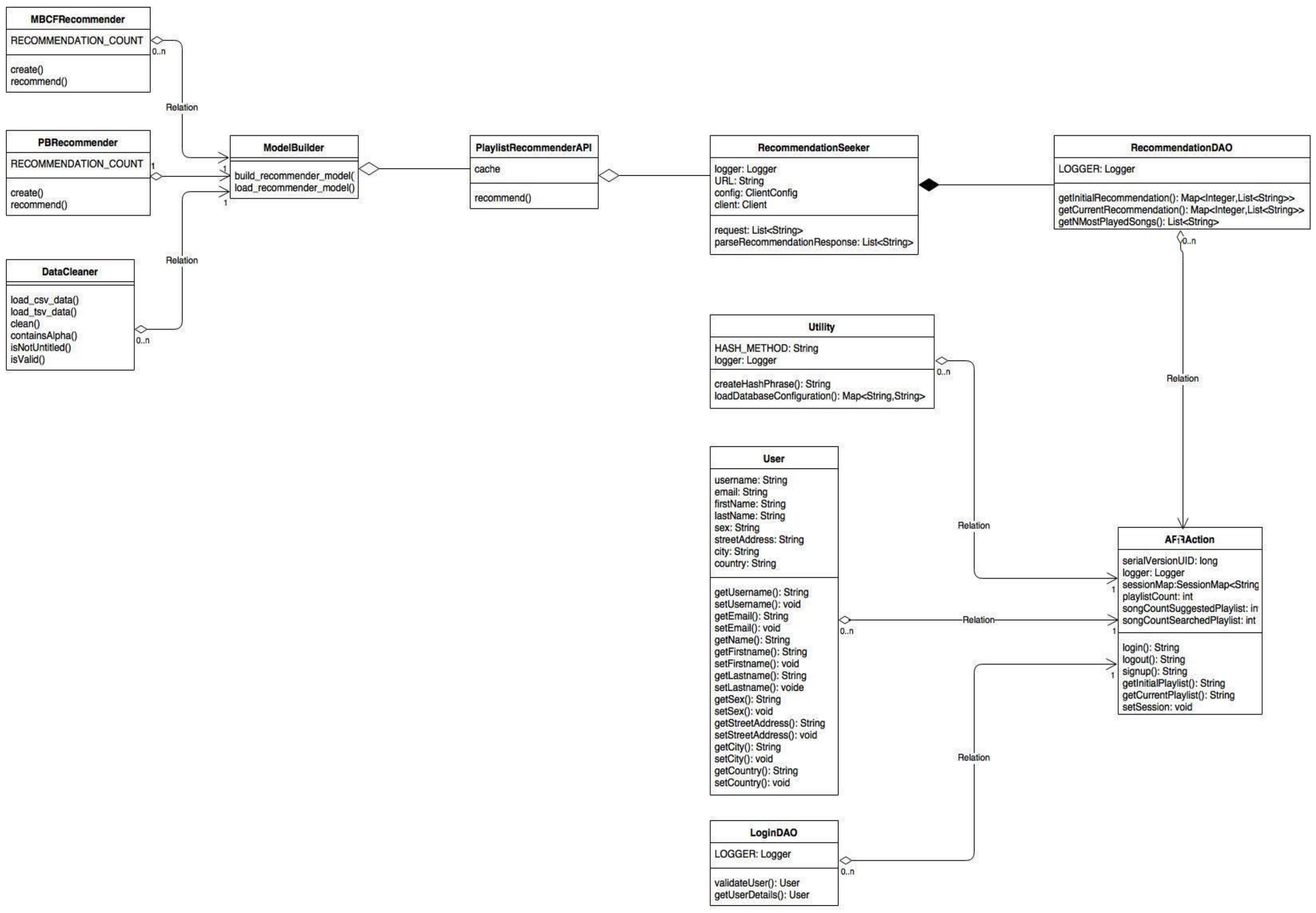
| | |
|---|---|
| **Use Case Name** | Add Song to Database |
| **XRef** | Section 2.1 |
| **Trigger** | The administrator logs in and selects "add song." |
| **Preconditi** | The administrator must be logged in. |

| | |
|---|---|
| **on** | |
| **Basic Path** | 1. The administrator logs in using his or her username and password.<br>2. The administrator selects “add song.”<br>3. The administrator enters song information.<br>4. The system saves the song information in the database.<br>5. The system returns to the home page. |
| **Alternative Paths** | N/A |
| **Postcondition** | The song is now in the system database. |
| **Exception Paths** | The attempt may be abandoned at any time. |
| **Other** | None |

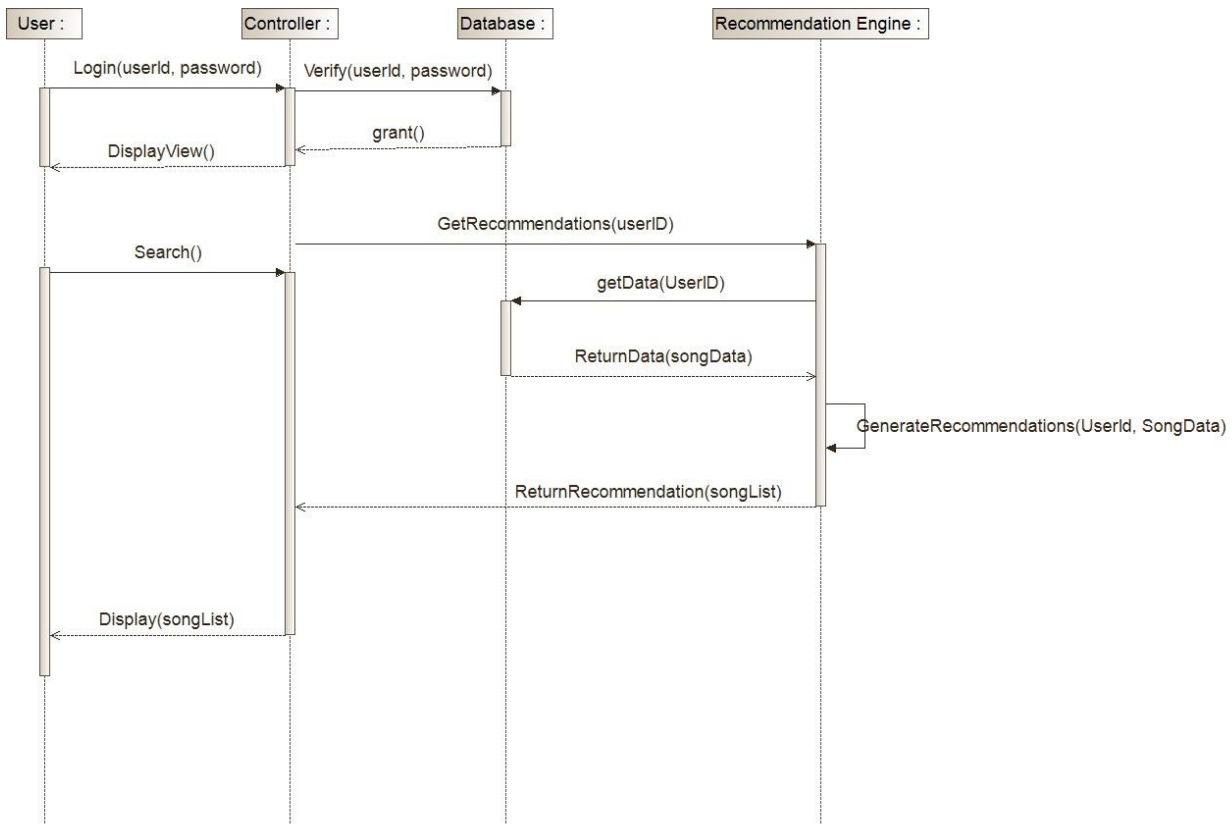
**3.1.1.7 Use Case: **Delete Song from Database (Relegated to later versions)**

| | |
|---|---|
| **Use Case Name** | Delete Song from DataBase |
| **XRef** | Section 2.1 |
| **Trigger** | The administrator logs in and selects “delete song.” |
| **Precondition** | The administrator must be logged in. The song must be in the database. |
| **Basic Path** | 1. The administrator logs in using his or her username and password.<br>2. The administrator selects “delete song.”<br>3. The administrator enters song title.<br>4. The system checks for the song in the database.<br>5. If the song is not in the database, the system displays a message.<br>6. If the song is in the database, the system prompts the administrator with a warning message that the action cannot be undone.<br>7. If the administrator accepts the warning message, the system deletes the song from the database.<br>8. The system returns to the homepage. |
| **Alternative Paths** | N/A |
| **Postcondition** | The song is no longer in the databse. |
| **Exception Paths** | The attempt may be abandoned at any time. |
| **Other** | None |

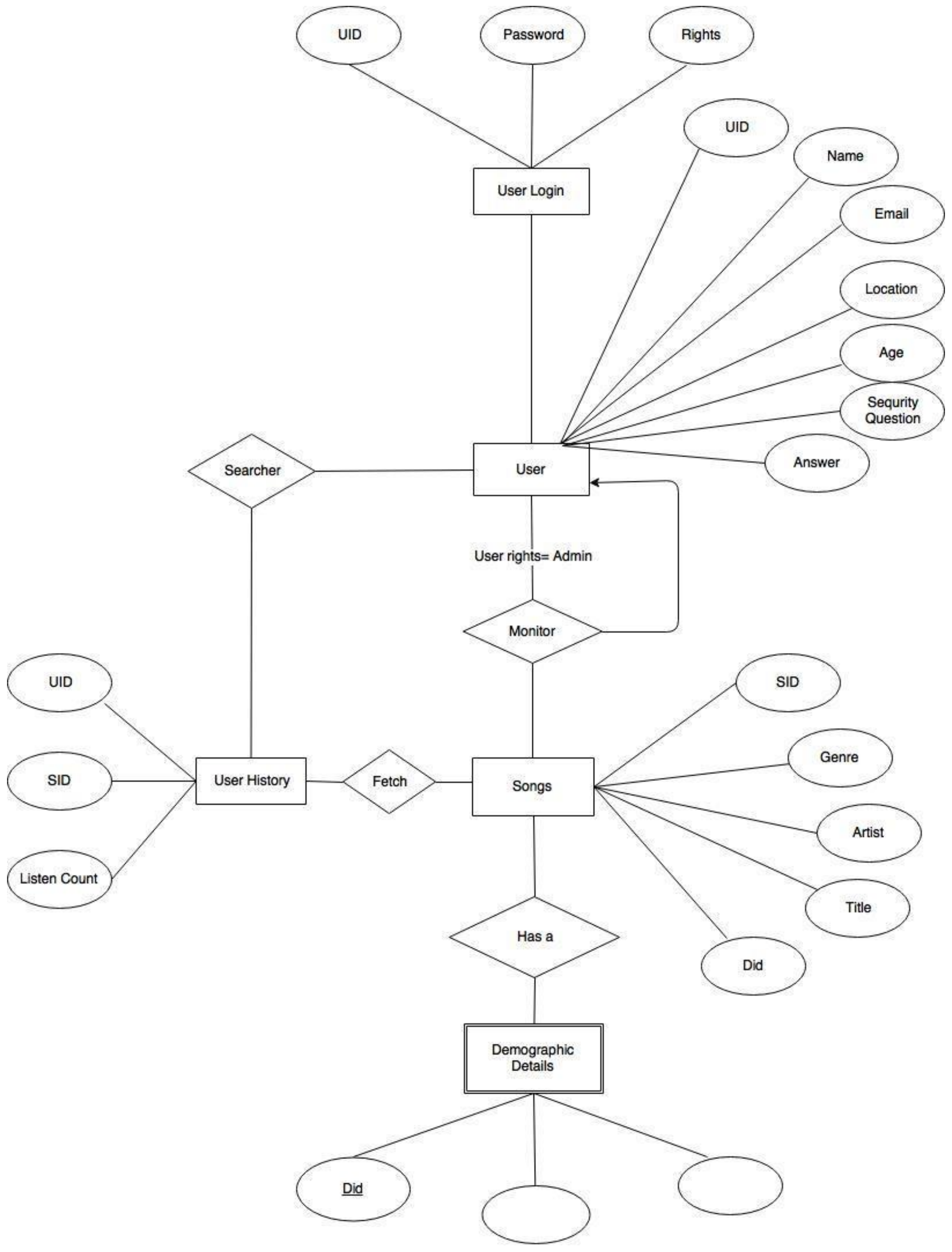
### **3.1.2 Domain Class Diagrams**

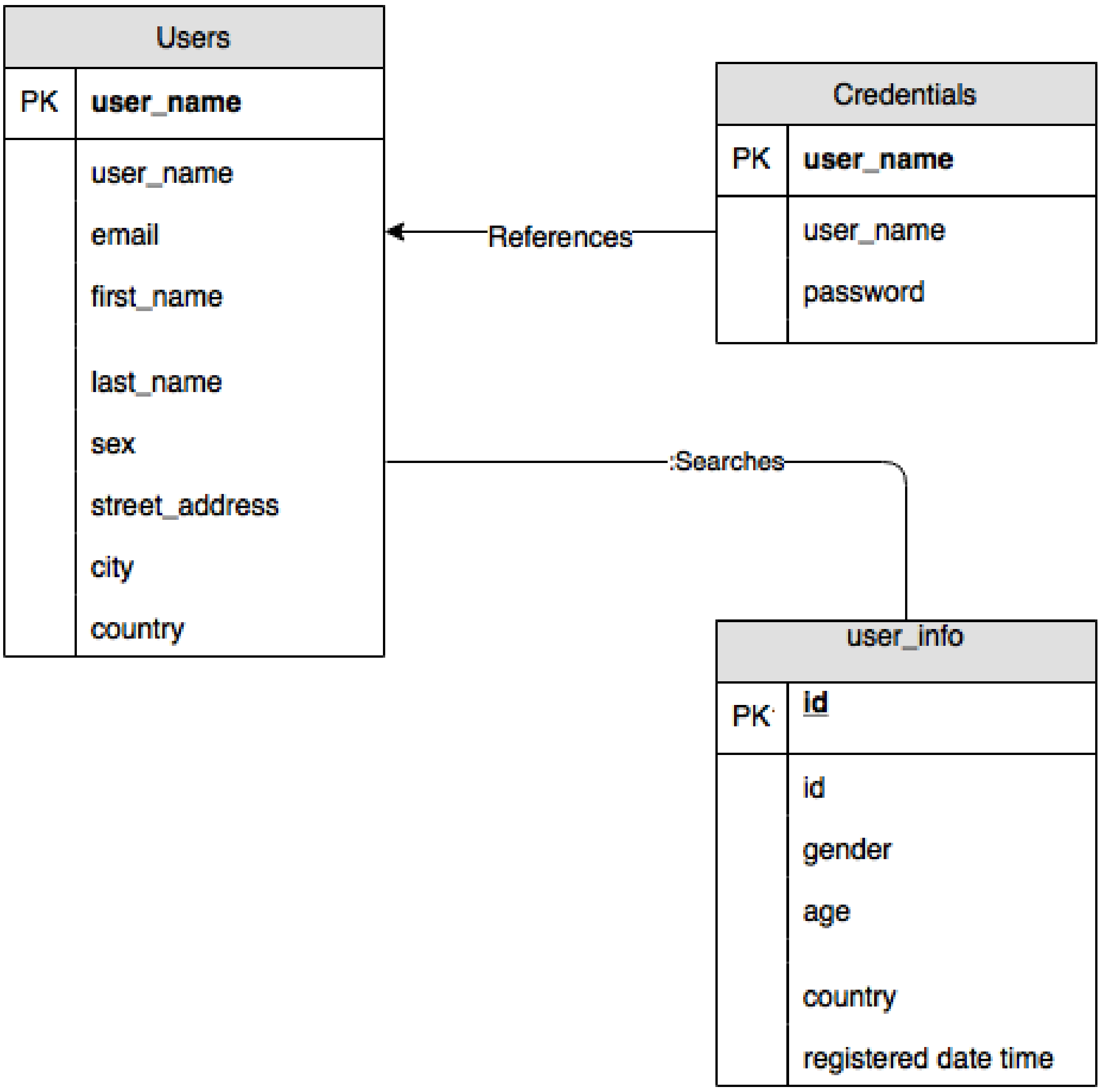


### 3.1.3 Sequence Diagram



### 3.1.4 Database Diagrams





## 3.2 Performance Requirements

The system shall not take more than 1 second to log the user into the system. The system shall not take more than 1 second to provide a list of recommendations to the user.
The system shall be secure and not have vulnerabilities that would allow a leak of user data stored in the database.
The system should recommend songs that are relevant to the interest of the user by correctly implementing known algorithms with a high degree of accuracy.

## 3.3 Logical Database Requirements

The database must interact with the recommendation engine portion of the project as well as the web service.
Please see the ER diagrams in section 3.2.4 (above) for further details about the logical structure of the database.
XREF Section 3.2.4

## 3.4 Design Constraints

### 3.4.2 Usage of the Application

The user interface of the web application should be simple to initiate, and the playlist output should be legible and easy to understand.

### 3.4.3 Response Time

No more than 1 second to log in and 1 second to recommend. The user registration process should not be so intensive that it takes a user more than 5 minutes to set up an account, assuming regular timing of entry for the forms on behalf of the user.

### 3.4.4 System Dependability

If the system loses connectivity or begins to malfunction, the user should be informed.
The admins of the system should also be informed, so that the system may be repaired.

## 3.5 Non Functional System Requirements

### 3.5.1 Reliability

The system should maintain itself in such a way that there are not frequent crashes and users are able to access the functionality.

### 3.5.2 Availability

In order to be competitive with other similar systems, the system must present high availability to the user. If the system is frequently offline, it is of no use to potential users. The system must also be relatively easy to understand, access, and operate, in order to attract users and compete with other software.

### 3.5.3 Security

The system must keep user information, stored in the database, secure. This includes protection against injection attacks and other malicious violoations of confidentiality.

#### 3.5.4 Maintainability

The system should be adaptable with time and built in such a way that it can be changed. This means that the code should be readable, clean, and commented. The system design should not take shortcuts that cause the system to be immutable.

#### 3.5.5 Portability

The system should function on a server, so that users may access it from multiple locations and systems without difficulty. Storage of the data should occur virtually.

#### 3.5.6 Summary of Non Functional Requirements

| ID | Characteristic | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | Correctness | ▨ | ▨ | ▨ | ▨ | ▨ | ▨ | ▨ | ▨ | ▨ | ▨ | ▨ | ▨ |
| 2 | Efficiency | ▨ | ▨ | ▨ | ▨ | ▨ | ▨ | ▨ | ▨ | ▨ | ▨ |  |  |
| 3 | Flexibility | ▨ | ▨ | ▨ | ▨ | ▨ | ▨ |  |  |  |  |  |  |
| 4 | Integrity/Security | ▨ | ▨ | ▨ | ▨ | ▨ | ▨ | ▨ | ▨ | ▨ | ▨ |  |  |
| 5 | Interoperability | ▨ | ▨ | ▨ | ▨ |  |  |  |  |  |  |  |  |
| 6 | Maintainability | ▨ | ▨ | ▨ | ▨ | ▨ | ▨ | ▨ | ▨ | ▨ | ▨ | ▨ | ▨ |
| 7 | Portability | ▨ | ▨ | ▨ | ▨ | ▨ | ▨ | ▨ | ▨ |  |  |  |  |
| 8 | Reliability | ▨ | ▨ | ▨ | ▨ | ▨ | ▨ | ▨ | ▨ | ▨ | ▨ | ▨ | ▨ |
| 9 | Reusability | ▨ | ▨ | ▨ | ▨ | ▨ | ▨ | ▨ | ▨ | ▨ | ▨ | ▨ |  |
| 10 | Testability | ▨ | ▨ | ▨ | ▨ | ▨ | ▨ | ▨ | ▨ | ▨ | ▨ |  |  |
| 11 | Usability | ▨ | ▨ | ▨ | ▨ | ▨ | ▨ | ▨ | ▨ | ▨ | ▨ | ▨ | ▨ |
| 12 | Availability | ▨ | ▨ | ▨ | ▨ | ▨ | ▨ | ▨ | ▨ | ▨ | ▨ | ▨ | ▨ |

### 3.6 Additional Comments

Many of these requirements, functional and non functional, may be relegated to later versions. See Function Prioritization and Change Management in Section 4 (below).

# 4. Function Prioritization and Change Management

In the event that a particular requirement is unable to be accomplished, the entire team should discuss and decide the course of action to be followed. Any changes to be made must be approved by all members of the design team and will receive final approval from the team leader Priyanka Sharma. All changes must be documented.

| Requirement | Priority Level | Change Plan of Action |
|---|---|---|
| Maintain secure user data. | High | Discuss the serious repercussions of not fulfilling this |

| | | |
|---|---|---|
| | | requirement. Notify all clients, notify users. |
| Implement a functioning web application with GUI | High | This is essential to the creation of the system. If not accomplished by deadline, a new deadline must be set and approved by the team. This requirement must be completed. |
| Provide song recommendations to user with accuracy. | High | Discuss ways to improve the recommendation accuracy, alter the algorithm or tools used to recommend. Update the database with a larger quantity of relevant data. |
| Maintain speed requirements (specified in the above sections) | Medium-High | Discuss a new deadline for completion. Discuss a new plan for future design that will ensure greater speed. |
| Allow new user to register | Medium-High | Potentially this can be left for a later date, at which point the system is integrated with a music player system. |
| Allow user to follow a chosen artist | Low | If this functionality is not added, it will reduce the competitive nature of the product. Discuss, evaluate, plan. |
| Allow admin to make changes to the database | Medium | This feature may be relegated until the system is integrated with a music player system. Discuss, evaluate, plan. |
| Present a sleek webpage design | Low | Refining web page may be an iterative |

| | | |
|---|---|---|
| | | process (future goal) |

## 5. Supporting Information

### Appendix A: Version Control

| Version No. | Date | Description | Sections | Name(s) |
|---|---|---|---|---|
| 0.1 | 9/21/2017 | Set up outline format, overview | All | Elizabeth Thomas |
| 0.2 | 9/22/2017 | Early diagrams, basic functionality descriptions | 1, 2, 3 | Elizabeth Thomas |
| 0.3 | 10/21/2017 | Use Case descriptions, formatting | 2 | Elizabeth Thomas |
| 0.4 | 10/22/2017 | Use Cases, Non-Functional Reqs, formatiing | 3 | Elizabeth Thomas |
| 0.5 | 10/24/2017 | Use case descriptions, updated use case diagram | 2, 3 | Elizabeth Thomas |
| 0.6 | 11/15/2017 | General Formatting, diagrams added | All | Elizabeth Thomas |
| 0.7 | 11/17/2017 | Overviewing for correctness | All | Indrajeet Mishra |
| 1.0 | 11/25/2017 | Final corrections and notes | All | Elizabeth Thomas |