# Software Requirements Specification

**for**

# Streamify

**Version 1.0 approved**

**Prepared by: Avinash Ratnam (PES1201800883),**

**Harichandana Magapu(PES1201801041),**

**Utsav Jolapara(PES1201801086)**

**PES University**

**30th January 2021**

# Table of Contents



# Revision History

| Name | Date | Reason For Changes | Version |
|---|---|---|---|
| | | | |
| | | | |

# 1. Introduction

## 1.1 Purpose

*The purpose of this document is to present a detailed description of the features and requirements of Streamify - a Music Player App. It will explain the interfaces, purpose and features of the system, as well as what the system will do, the constraints under which it must operate and how it will react to external stimuli. It also describes non-functional requirements and other factors necessary to provide a complete and comprehensive description of the requirements for the software.*

## 1.2 Intended Audience

*The document is intended for the marketing team, project managers, developers, testers, documentation writers, and other stakeholders. The rest of this document contains an overall description of the product, including the product perspective and functions, user classes and characteristics, operating environment, design and implementation constraints, assumptions made and dependencies. It also describes the external interface (User/Software/Communication) requirements, analysis models, system features, and non-functional requirements (performance, safety, security, quality, business).*

## 1.3 Product Scope

*This software system will be an online music player with a clean and intuitive user interface and an extensive library of not just mainstream music, but also music by lesser-known independent artists across multiple different genres. It will be an app made by music lovers, for music lovers. Any user can listen to the music they want when they want to. More specifically, it will allow a user to search for any song and listen to any song on-demand, queue songs to listen to later, share songs with friends and family, discover new songs via recommendations made by the app, save tracks to listen to offline, create playlists, and share them across various platforms.*

## 1.4 References

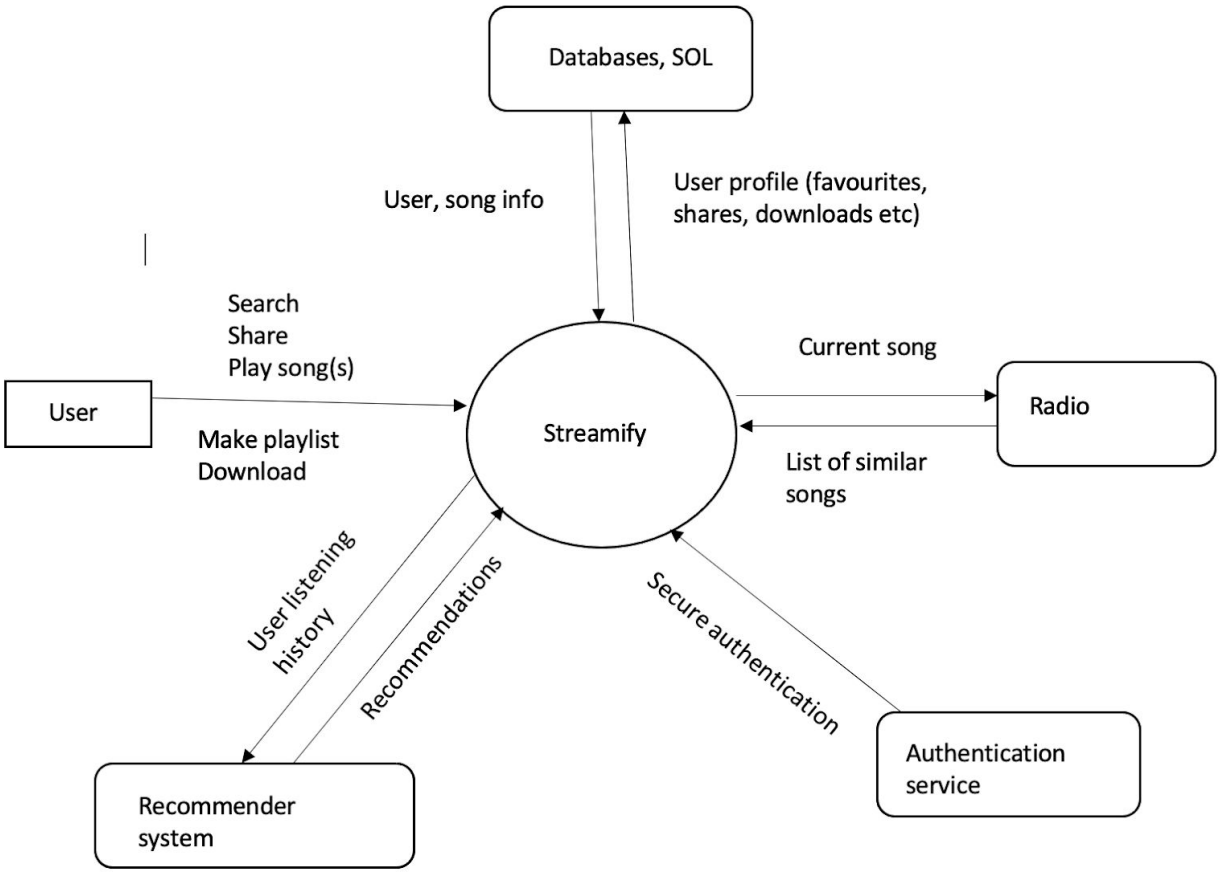
*https://wynk.in/music for description of a music streaming service*
*https://senior.ceng.metu.edu.tr/2014/cenghistoryx/documents/CengHistoryX_SRS.pdf - Software Engineering Project done by students att METU*
*https://www.ida.liu.se/~TDDC88/theory/srs-handout.pdf - Software Engineering Project done by students at Linköping University*
*SRS for STUB version 1.1 (project by students of IIT Delhi) - provided by faculty*

# 2. Overall Description

## 2.1 Product Perspective

*Our app is meant to be the best place for today's youth to indulge their love of music. We intend for our users to have it all - a clean and intuitive UI, a top-tier music recommender system, as well as offline access to their favourite songs, without an excessive amount of advertisements interrupting their experience. Our goal is to replace the existing music platforms such as Spotify, Wynk, and JioSaavn.*

## 2.2 Product Functions



## 2.3 User Classes and Characteristics

*By Age:*

- *3-10 - They might have difficulty reading and understanding English, so default settings and visual interaction is a must. Parental guidance is required, as well as blocking of explicit music*
- *11-15 - very active in building and sharing playlists. Likely to stick to one genre. Controlled exposure to explicit music required.*
- *16-25 - target demographic. Likely to explore multiple genres and share songs/playlists across multiple different platforms. More likely to be frequent users.*
- *26-30 - Similar to the 16-25 demographic, but slightly less likely to explore new genres.*
- *30-45 - likely to listen to music they are already familiar with, less emphasis on playlists, sharing, or discovery.*
- *45+ - likely to listen to music they are already familiar with, no emphasis on playlists, sharing, or discovery. Likely to be technically challenged, may need more help navigating the interface.*

*By frequency of usage:*

- *Frequent - use the app at least once or twice a day*
- *Occasional - sporadically use the app*

*By Reason for use:*

- *Regular - use the app to listen to music, cannot upload their own songs*
- *Business - Independent artists as well as record labels; can upload songs*

## 2.4 Operating Environment

*This app will run on smartphones which support Android (versions 10.0 and above) as well as iOS (version 13.7 and above). It must peacefully coexist with social media apps such as Twitter, Facebook, Instagram, and Whatsapp for sharing of songs/playlists.*

## 2.5 Design and Implementation Constraints

- *Since we need user profile data while developing the product, finding real time and sufficient data can be a problem for developers because of regulatory policies.*
- *Huge amounts of data will be required for training the recommender system and radio models, hence lots of disk space and clusters are necessary. The accuracy of these models is also extremely important, and this may be difficult with sparse and huge data.*
- *The app gathers real-time user profile data, whose security is of utmost importance.*
- *The system is being developed in Python, so there may be compatibility issues with certain APIs and other systems.*

## 2.6 Assumptions and Dependencies

*Here we are assuming that internet connection is consistently available as well as stable for a smooth streaming experience.*

*We also assume that the authentication service is compatible with our development environment, is secure, and works effectively.*
*Another assumption that we are making is that the phone will have a volume control option which the app can access.*

# 3. External Interface Requirements

## 3.1 User Interfaces

There will be one type of user. Therefore there are no differences between users in terms of functionality, visualisation and interface. The user interface is only dependent on web developers and designers.
The website that we are working on can offer unique experiences to its users.
For example, the user can choose a unique profile image, nickname and avatar. However, the architecture will be the same for all users.
The interface will consist of a certain number of steps such as User Login, Top Hundred lists, Search options, custom playlists etc.
The first step is the user login, using which the recommendation system is activated. The recommendations are specified by an algorithm that is dependent on users actions. These actions include the genre of music the user is listening to, artists similar to the current artist, artists who collaborated with the contemporary artist etc.
Some other miscellaneous actions include previously listened to music, downloaded music, favorited music. If a song is downloaded or listened to multiple times, the songs' rating in the album will increase. These ratings can be represented by a few highlighted bars, comparable to that of a typical graphic rating system.
Other specific graphical interface elements that are included are:

1. Top left corner: Home icon. Redirects the user to the Home page from wherever they are.
2. Top Right Corner: Search Icon. Allows the user to search for songs, albums, artists etc.
3. Bottom:
    a. Music Player. Usually, when the user scrolls through the app, it will be in its compact minimal form, revealing only necessary information like the song name, artist and the ability to play/ pause the song.
    b. Navigation icons:
        - Home: Take the user home.
        - Radio: Allow users to access different radio channels and services.
        - Browse: Find new music based on genre, region, language, artist etc.
        - My Music: Collection of all the users added music. (Includes user-created playlists, albums/ singles the user has liked etc.)
        - Settings: Configure anything about the app the user needs.

## 3.2 Software Interfaces

The app will run on any platform, such as iOS, Android, Windows, macOS etc.
Since our app is based on streaming music from an external off-shore server, an internet connection is required.
The application will be coded mostly in Python along with Python Libraries. The website will be made in HTML, CSS, JS and JS Libraries. For the database, MySQL will be used for accessing and managing purposes.

| **Name** | **Version Numbe**r | **Source of Software** |
|---|---|---|
| MySQL | MySQL 8.0 | dev.mysql.com |
| HTML | HTML 5 | - |
| Windows | Windows 8 or Windows 10 | Microsoft Company |
| Python | 3.9.0 | - |

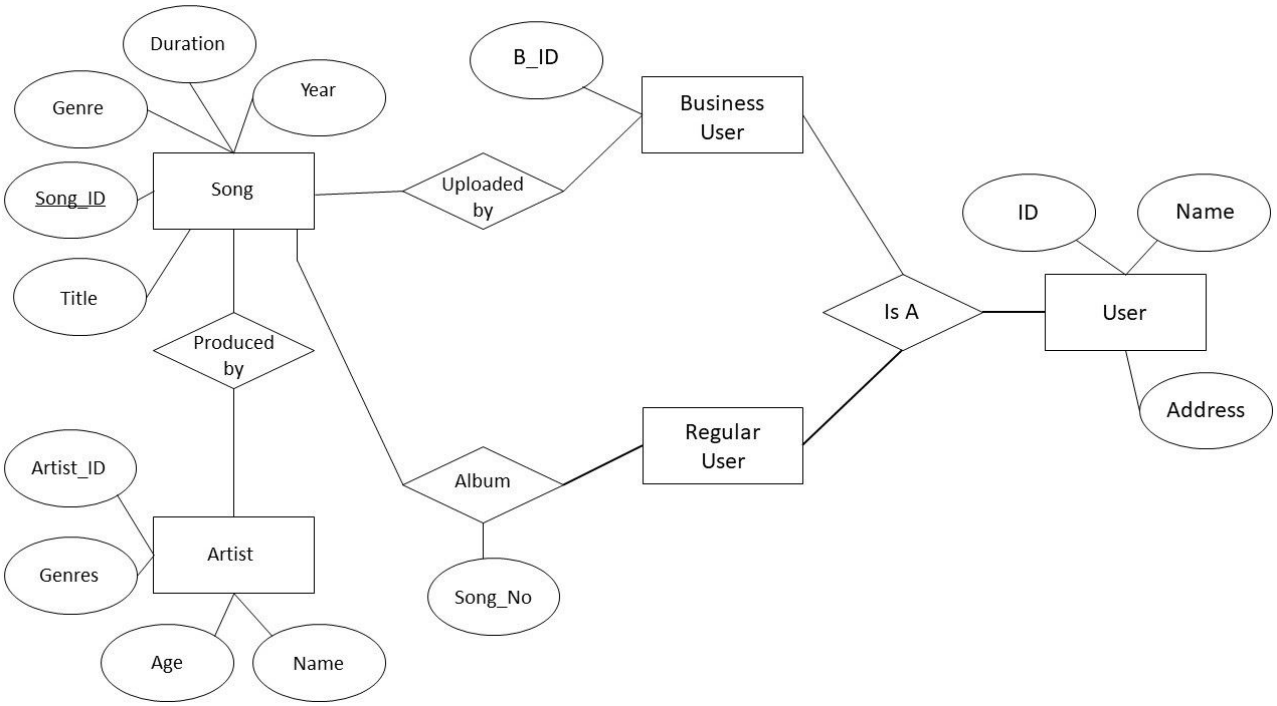
## 3.3 Communications Interfaces

Protocols usage:

| **Service** | **Protocol Name** | **Service Name** |
|---|---|---|
| Email | IMAP (Internet Mail Access Protocol)<br>POP3 (Post Office Protocol) | - |
| Streaming Music from Server | RTSP (Real-Time Server Protocol)<br>RTCP (Real-Time Control Protocol) | - |
| Database | TCP/IP (Transmission Control Protocol/ Internet Protocol) | MySQL |
| Website | HTTP/HTTPS | HTML and JS |

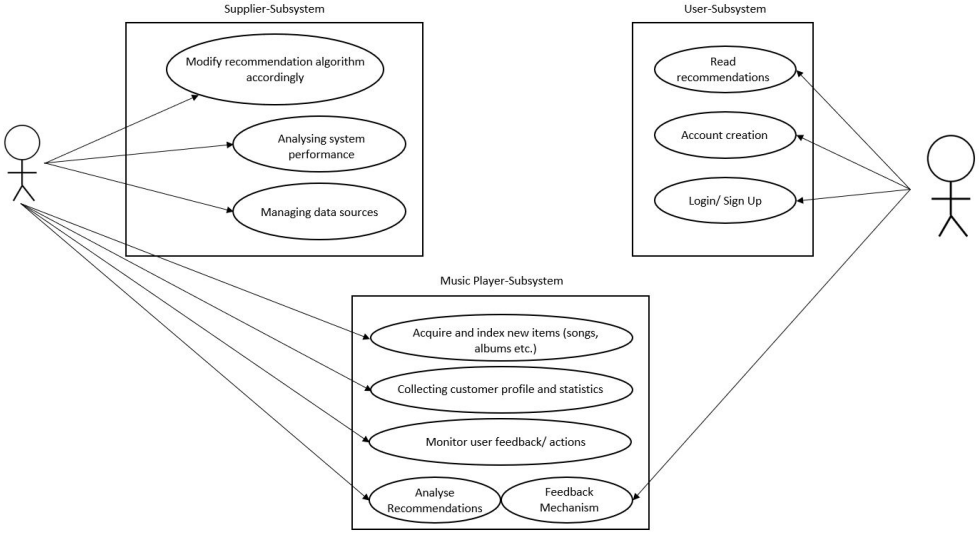
# 4. Analysis Models

## 4.1 ER Diagram




## 4.2 Use Case Diagrams

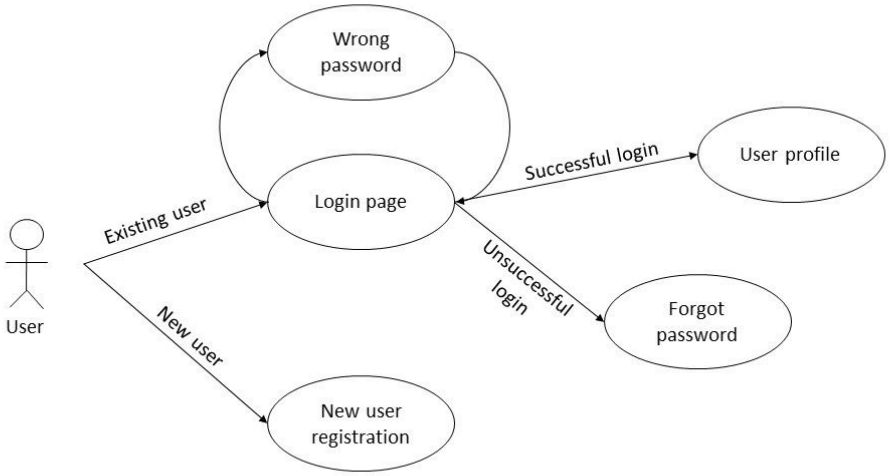
Overall use/ general diagram:



### 4.2.1 Login

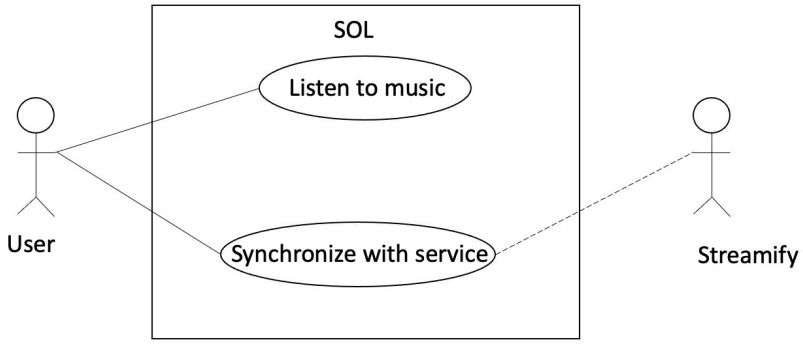
If the user already has an account with the streaming service, they are redirected/ prompted to log in with the correct credentials and go to the landing page. If the user is new, they are required to make an account to access the app.

Fig: User Login Procedure




### 4.2.2 Synchronise with service

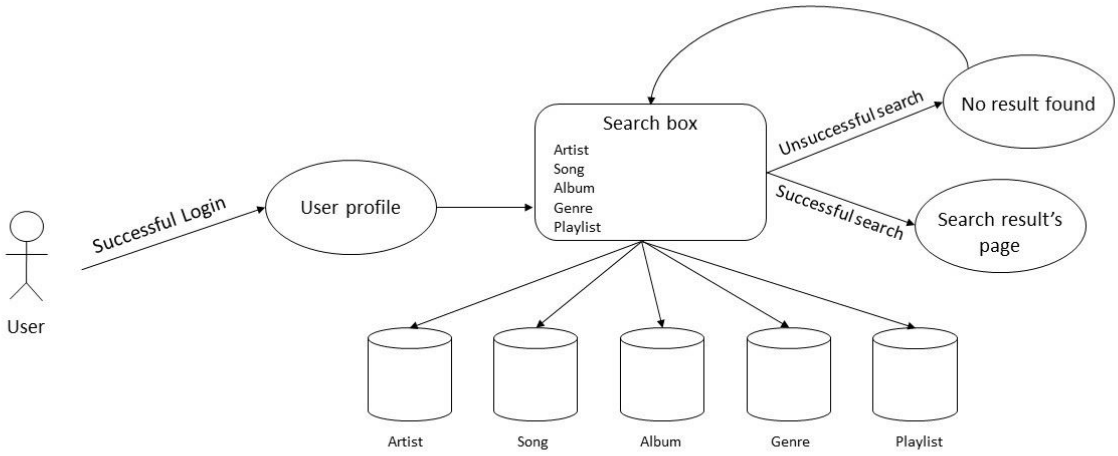
When the user opens Streamify while online, the playlists and music are automatically synchronised with their account. Synchronisation conflicts are resolved by prompting the user with questions. When SOL is left, the user is asked whether they wish to prepare for offline listening (if this is not the default SOL setting).



### 4.2.3 Search

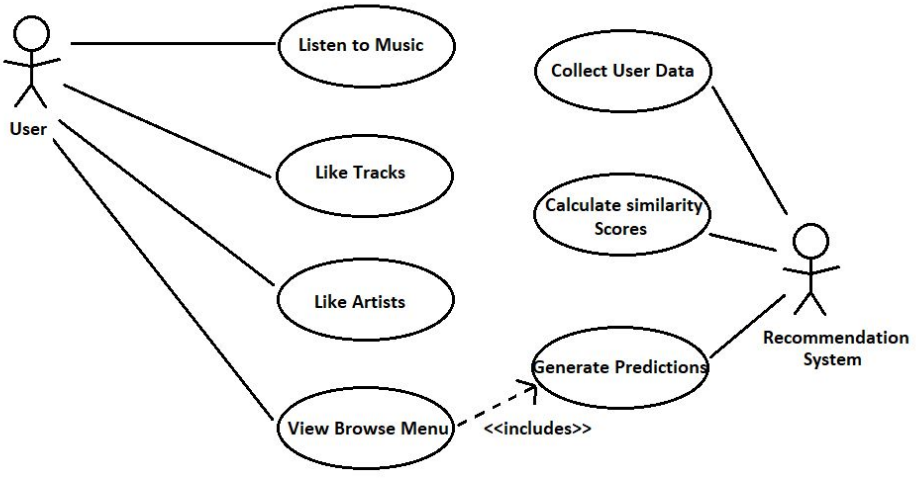
If the user has successfully logged in, they have the option to search for any songs, artists or albums they'd like. All the different information is stored in separate databases which are accessible by the same search bar. If the search element is found, the user is redirected to that search results page, else they are prompted to search again.

Fig: Search Procedure



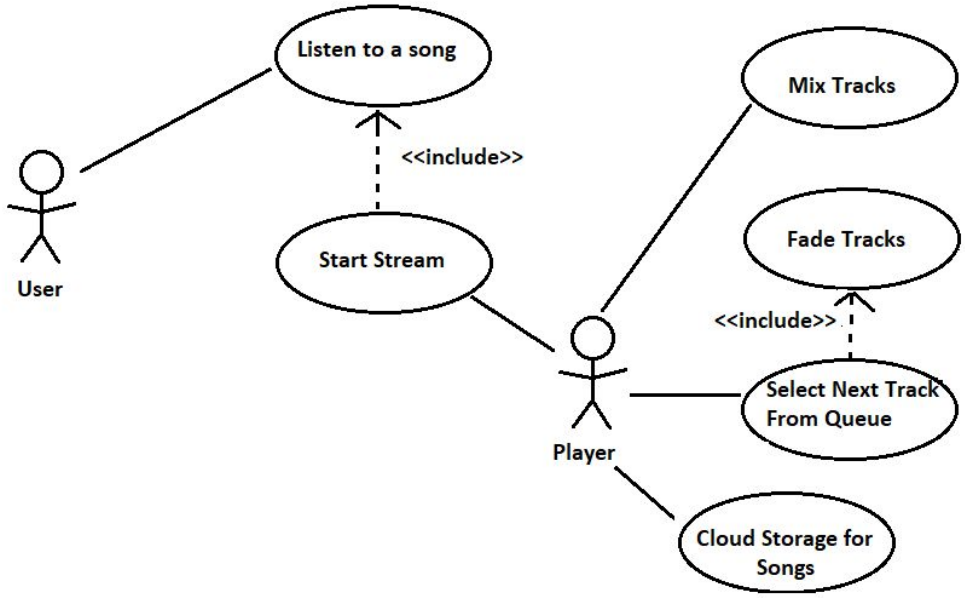

### 4.2.4 Recommendation System

The app collects the user's listening history, likes, shares, and downloads. It uses this information as input to a Machine Learning model, which will generate a list of songs that the user has not heard, but is likely to love.



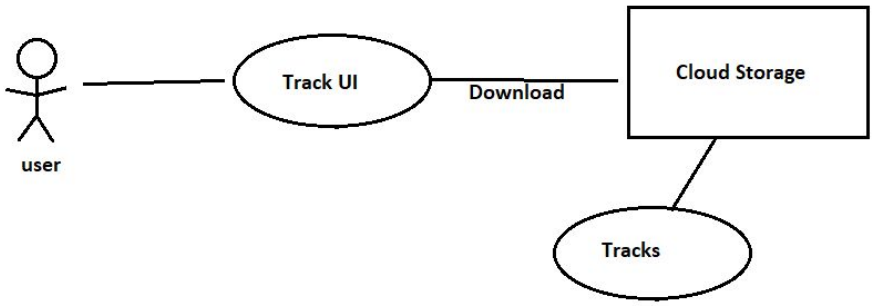
### 4.2.5 Stream

A user can listen to any song available in the SOL, i.e they can use the internet to stream the song(s) of their choosing



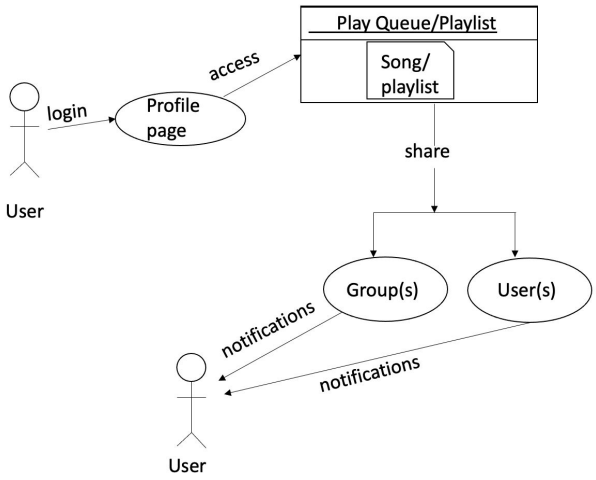

### 4.2.6 Download

A user can save any song in the SOL to their device so that it will be available even when they do not have access to internet



### 4.2.7 Share

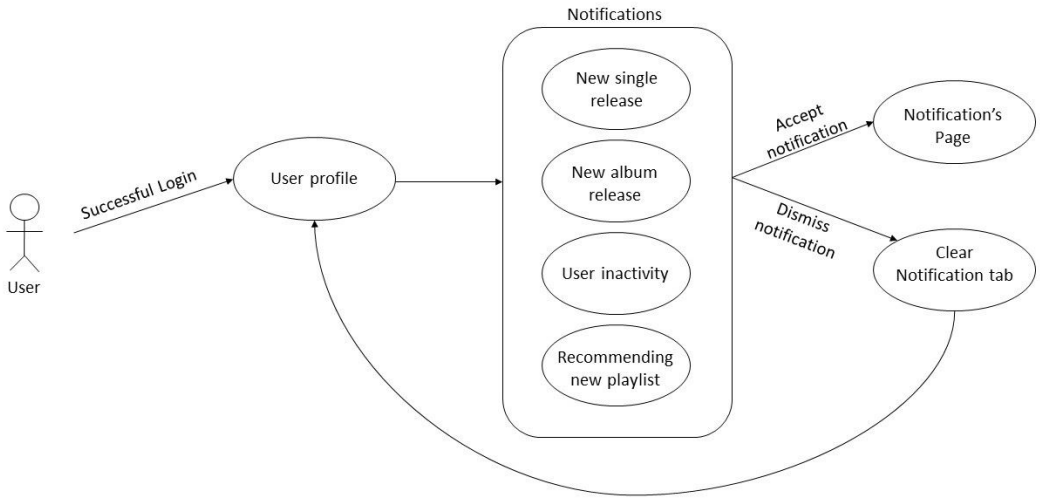
Users can share a song they are currently listening to, a song in their play queue, a song in a saved playlist, or an entire playlist with single or multiple users/groups via social media apps such as Instagram, Twitter, and Facebook.




### 4.2.8 Notification

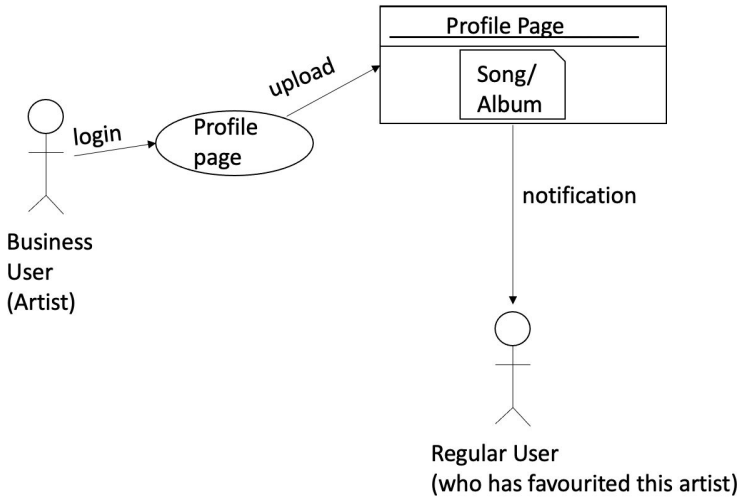
Only while the user is logged in will they receive notifications, or if the settings that are chosen allow for notifications. Notifications are sent primarily for new songs, albums or recommending playlist. If the user is logged in but hasn't used the app in several days, a notification is sent reminding them they are still logged in, and there is more music to be discovered.

Fig: Notification Procedure



### 4.2.9 Upload

Business users can upload their songs and albums, and notify those users who have favorited their profile.




# 5. System Features

## 5.1 On-Demand Music Streaming

### 5.1.1 Brief Description and Priority

**Brief Description**: Users can stream and listen to songs on the music app without downloading them by choosing any song via the browse menu or searching for it.

**Priority**: High priority feature. The app is built around the ability to deliver music on-demand to users worldwide.

### 5.1.2 Stimulus/Response Sequences

Stimulus:

- User searches for a song /finds one in the browse section.
- User taps on on a song of their choice.

Response Sequences:

- Based on the search keyword a relevant list of songs is displayed.
- The application communicates to the servers that hold the music files—requesting for a particular file on the user clicking on any song.

- The CDN associated with the cloud service that houses the files, starts to deliver data to the user in small amounts buffering the data.
- The user gets pre-buffered music that has been pre-buffered a few minutes or seconds before playing a song.

Given that the user has a good internet connection, streaming technology provides an uninterrupted listening experience.

### 5.1.3 Functional Requirements

Tag: [ODS-REQ-#]: On-Demand Streaming-Requirement-#number

**ODS-REQ-1:** Searching Mechanism: The application must communicate with the backend cloud service and produce a search result based on a keyword, artist or song name.

**ODS-REQ-2:** SOL on Cloud: Large library of songs must be stored on any cloud service such as Amazon S3, next cloud etc.

**ODS-REQ-3:** CDN: linked with the cloud service provider to send data in a streamed manner to the client.

**ODS-REQ-4:** Use of reliable protocols: TCP can be chosen due to its dependable data transfer properties.

**ODS-REQ-5:** Basic Audio Files Codec Support: the application must handle a basic variety of audio file formats. An appropriate file format must be chosen when storing the audio files on the cloud service. Example: Vorbis, AAC LC, FLAC, Mp3 etc.

## 5.2 Search

### 5.2.1 Brief Description and Priority

**Brief Description**: Users search for the desired song by entering a keyword example: name, artist, album etc. Based on the search parameters, a list is displayed.

**Priority**: High priority feature. The ability to search through an extensive database of songs is crucial.

### 5.2.2 Stimulus/Response Sequences

Stimulus:

- User logs in, accesses the app.
- User searches for a song using song name, category, artist, album etc.
- User clicks on a song of their choice from the displayed list.

Response Sequences:

- Upon receiving the search keyword, the system formulates a query.
- This query is then dispatched to the servers containing the song library.

- The query is processed and the server bundles the most relevant results.
- The most relevant results are sent back to the client-side app, and the list is displayed.

### 5.2.3 Functional Requirements

Tag: [SER-REQ-#]: Search-Requirement-#number

**SER-REQ-1:** Search box to take the input (keyword) from the user.

**SER-REQ-2:** SOL on Cloud: Large library of songs must be stored on any cloud service such as Amazon S3, next cloud etc.

**SER-REQ-3:** CDN: linked with the cloud service provider to send data in a streamed manner to and from the client.

**SER-REQ-4:** A searching algorithm to return top matches quickly and efficiently.

**SER-REQ-5:** Use of reliable protocols: TCP can be chosen due to its dependable data transfer properties.

## 5.3 Music Recommendation

### 5.3.1 Brief Description and Priority

**Brief Description**: Predicts and recommends songs and artists that the user would like based on their profile (listening history, favourites, shares, downloads).

**Priority**: High priority feature. Recommendation systems help with the problem of information overload.

### 5.3.2 Stimulus/Response Sequences

Stimulus:

- User logs in and accesses the music app.
- User searches for a song using song name/genre/artist/album etc.
- User clicks on a song of their choice from the displayed list.
- Different songs, artists and genres are favourited and liked by the user.

Response Sequences:

- System stores the various artists that the user searches for on the backend.
- The system also stores the different songs and artists liked/favourited by the user on the servers.
- Based on this data gathered from the user, a machine learning model is run on the data to predict/recommend different artists and songs that the user may like.
- The recommended entities are bundled together and are sent to the client-app to display on the browse section of the application.

### 5.3.3 Functional Requirements

Tag: [REC-REQ-#]: Recommendations-Requirement-#number

**REC-REQ-1:** Cloud Storage: Large amounts of data (user profiles, liked songs, artists genres and searches) for n number of users must be stored on any cloud service such as Amazon S3, next cloud etc.

**REC-REQ-2:** Model to predict new songs, artists based on the various parameters gathered from the user and the song or artist they listen to.

**REC-REQ-3:** CDN: linked with the cloud service provider to send data in a streamed manner to and from the client.

**REC-REQ-4:** A searching algorithm to return top matches quickly and efficiently.

**REC-REQ-5:** Use of reliable protocols: TCP can be chosen due to its reliable data transfer properties.

## 5.4 Offline Access [Track Downloads]

### 5.4.1 Brief Description and Priority

**Brief Description**: Users can access songs and playlists of their choice offline by downloading the songs onto their device.

**Priority**: Medium priority feature. The ability to download songs and have them on the go is an excellent feature to include in the application. It falls under Medium priority as it is not crucial to the on-demand streaming service principle that the application is built on.

### 5.4.2 Stimulus/Response Sequences

Stimulus:

- User logs in and accesses the music app.
- User searches for a song on the global search or in a playlist using keywords.
- User clicks on the download button next to the song name and download begins.

Response Sequences:

- (Global Search)
- Upon receiving the search keyword, the system formulates a query.
- This query is then dispatched to the servers containing the song library.
- The query is processed and the server bundles the most relevant results.
- The most relevant results are sent back to the client-side app and the list is displayed. (Local Playlist)

- The song is searched in the local playlist and is displayed.
  (Download)
- The desired song is chosen and the download button on clicking issues a query to the cloud service.
- The CDN responds with a stream of data from the server, and the song is stored in a specified directory on the user's device.
- The UI is updated to show that the particular song is a downloaded song and is available for offline access.

### 5.4.3 Functional Requirements

Tag: [OFA-REQ-#]: Offline Access-Requirement-#number

**OFA-REQ-1:** UI to indicate downloaded songs.

**OFA-REQ-2:** SOL on Cloud: Large library of songs must be stored on any cloud service such as Amazon S3, next cloud etc.

**OFA-REQ-3:** CDN: linked with the cloud service provider to send data in a streamed manner to and from the client.

**OFA-REQ-4:** A searching algorithm to return top matches quickly and efficiently.

**OFA-REQ-5:** Use of reliable protocols: TCP can be chosen due to its dependable data transfer properties.

**OFA-REQ-6:** Space (storage) on the user's local device.

## 5.5 Playlist Creation

### 5.5.1 Brief Description and Priority

**Brief Description**: Users can group songs to create a playlist. Users are allowed to create as many playlists as they want.

**Priority**: High priority feature. The feature to create a playlist is one of the most basic features every music application must-have.

### 5.5.2 Stimulus/Response Sequences

Stimulus:

- User logs in and accesses the music app.
- User searches for a song using song name, category, artist, album etc.
- User clicks on a song of their choice from the displayed list.
- Using the UI, the user is prompted to add the song to an existing playlist or create a new playlist with the song.

(Existing playlist)
- The user chooses an existing playlist, and the song is added.

(New playlist)
- The user enters a name for the playlist.
- The playlist is created with the selected song in it as default.

(Create a new playlist from playlists menu)
- Using the UI, the user chooses an option to create a new playlist.
- The user enters the name of the playlist.
- An empty playlist is created with the given name.

Response Sequences:

- Upon receiving the search keyword, the system formulates a query.
- This query is then dispatched to the servers containing the song library.
- The query is processed and the server bundles the most relevant results.
- The most relevant results are sent back to the client-side app, and the list is displayed.
- (Existing Playlist)
- UI redirects the user to select a playlist from the list of playlists stored (on the local device).
- Once chosen, the song id is added to the playlist data structure.
- (Create a playlist via song UI)
- UI redirects the user to create a new playlist.
- The name is taken in as input, and a new instance of an empty playlist is created on the local device.
- The id of the song that the menu was accessed via is added to the new playlist instance.
- (create a playlist from playlist menu)
- The name is taken in as input, and a new instance of an empty playlist is created on the local device.

### 5.5.3 Functional Requirements

Tag: [PLA-REQ-*#*]: Playlists-Requirement-*#*number

**PLA-REQ-1:** Data structure to store song IDs and song details.

**PLA-REQ-2:** Storage space on the local device to store the playlists.

**PLA-REQ-3:** Cloud storage to keep a copy of the playlists. (Linking an account to their playlists.)

## 5.6 Sharing Tracks and Playlists

### 5.6.1 Brief Description and Priority

**Brief Description**: Users can pick a song or their playlist and share it with their friends on the music application.

**Priority**: Medium priority feature. Ability to share songs or playlists among one's peers is a sought-after feature in most apps, but since it's not essential for the application to function it is a medium priority feature.

### 5.6.2 Stimulus/Response Sequences

Stimulus:

- User logs in and accesses the music app.
- User searches for a song using song name, category, artist, album etc.
- User clicks on a song of their choice from the displayed list.
- User selects the platform to share on

Response Sequences:

- Upon receiving the search keyword, the system formulates a query.
- This query is then dispatched to the servers containing the song library.
- The query is processed and the server bundles the most relevant results.
- The most relevant results are sent back to the client-side app, and the list is displayed.
- On opting the song to be shared, a list of social media platforms is displayed.
- After a choice is made, the sharing API for the selected platform is called, and the user can share the song to users/groups on those platforms.

### 5.6.3 Functional Requirements

Tag: [SHA-REQ-#]: Sharing-Requirement-#number

**SHA-REQ-1:** Social Network sharing APIs: to facilitate sharing across different platforms

**SHA-REQ-2:** Cloud Storage: to store relations between users for sharing within the pla

**SHA-REQ-3:** CDN: linked with the cloud service provider to send data.

**SHA-REQ-4:** Social Network App authentication and permissions: user is logged into the sharing apps and has given Streamify permission to access their contacts.

## 5.7 Lyrics

### 5.7.1 Brief Description and Priority

**Brief Description**: Users can listen to songs and view their lyrics.

**Priority**: Low priority feature. While having access to lyrics of various songs is a bonus feature and is not required to make the app function, neither is it a feature in most music applications.

### 5.7.2 Stimulus/Response Sequences

Stimulus:

- User accesses the music app.
- User searches for a song using song name, category, artist, album etc.
- User clicks on a song of their choice from the displayed list.
- Using the UI, the user can click to view lyrics of the song as it plays.

Response Sequences:

- Upon receiving the search keyword, the system formulates a query.
- This query is then dispatched to the servers containing the song library.
- The query is processed and the server bundles the most relevant results.
- The most relevant results are sent back to the client-side app, and the list is displayed.
- When a song is playing the server also sends the song's lyrics in anticipation of the user invoking a function to view it.
- When the call is made the UI updates and the lyrics are displayed.

### 5.7.3 Functional Requirements

Tag: [LYR-REQ-#]: Lyrics-Requirement-#number

**LYR-REQ-1:** Cloud Storage: Large database to store lyrics of songs available on the app.

**LYR-REQ-2:** CDN: linked with the cloud service provider to send the lyrics over to the client app.

## 5.8 Radio

### 5.8.1 Brief Description and Priority

**Brief Description**: Users can listen to "song radios", i.e. a collection of songs based on any artist, album, playlist, or song of the user's choice. It even updates over time to keep fresh.

**Priority**: Low priority feature. Access to song radios is merely a bonus feature and is not a feature of the on-demand streaming service.

### 5.8.2 Stimulus/Response Sequences

Stimulus:

- User accesses the music app.
- Users plays a single song
- User chooses to start a radio based on the current song.

Response Sequences:

- Application communicates with the backend-servers.
- Servers return a playlist of songs with a high degree of similarity.
- Radio is generated.

### 5.8.3 Functional Requirements

Tag: [RAD-REQ-#]: Radio-Requirement-#number

**RAD-REQ-1:** Machine Learning Model for creation and updation of the song radios.

## 5.9 Notifications

### 5.9.1 Brief Description and Priority

**Brief Description**: Users receive notifications if the artists whose profiles they have favourited release new music.

**Priority**: Low priority feature. The feature is only meant to target hardcore fans of various artist and those avid music listeners who do not want to miss a beat.

### 5.9.2 Stimulus/Response Sequences

Stimulus:

- User accesses the music app.
- User favourites an artist and opts for notifications from the settings.

Response Sequences:

- DB stores the favourited artist of every user.
- When the artist uploads new music or the license to certain songs and albums are secured, a notification is sent out to users with the respective artist in their favourited list.

### 5.9.3 Functional Requirements

Tag: [NOT-REQ-#]: Notification-Requirement-#number

**NOT-REQ-1:** Push notifications to be enabled on the local device.

## 5.10 Play/Pause/Seek

### 5.10.1 Brief Description and Priority

**Brief Description**: Users have control over the tracks playing in real-time. They have the ability to pause or play from a given position on the track. They have control which enables users to seek to a desired position.

**Priority**: High priority feature. Minimum requirement of a music player.

### 5.10.2 Stimulus/Response Sequences

Stimulus:

- User accesses the music app.
- User chooses a song.
- User pauses/plays/seeks the song.

Response Sequences:

- On selection of the desired song it begins streaming.
- Upon given input by the user the song is paused or played from the current position on the track, or played from the desired position (in case of seek)

## 5.11 Previous/Next

### 5.11.1 Brief Description and Priority

**Brief Description**: Users have control over the tracks playing in real-time. They have the ability to skip tracks or traverse back to a track that has been queued in the playlist.

**Priority**: High priority feature. Minimum requirement of a music player.

### 5.11.2 Stimulus/Response Sequences

Stimulus:

- User accesses the music app.
- User chooses a song.
- User chooses the next option.
- User chooses the previous option.

Response Sequences:

- On selection of the desired song it begins streaming.
- The next song in the queue is played.
- The previous song the queue is played.

# 6. Other Nonfunctional Requirements

## 6.1 Performance Requirements

On-Demand Streaming Service: Real-time service. Requires reliable and quick data transfer in a streamed manner from the cloud to the end device.

Search: Must be quick and efficiently performed as it is a real-time service provided by the application.

Connection to the service: Must be completed within is 3 seconds.

Time limit between pressing play button and hearing the song: no greater than 1 second

Any stimulus that relating to a song (Pause, Play, Next, Previous etc.): no greater than 0.5 seconds

Login and authentication: must take no greater than 2 seconds.

Functional Requirements:

**ODS-REQ-2:** SOL on Cloud: Large library of songs must be stored on any cloud service such as Amazon S3, next cloud etc.

- Cloud service that does not have long periods of down-time.
- Multiple servers at different locations to create reliability with redundancy.
- Access times of entities stored on the cloud must be as low as possible to provide a seamless experience.

**ODS-REQ-3:** CDN: linked with the cloud service provider to send data in a streamed manner to the client.

- Use of reliable data transfer protocols in the delivery of the data.

**ODS-REQ-5:** Basic Audio Files Codec Support: the application must be able to handle a basic variety of audio file formats. An appropriate file format must be chosen when storing the audio files on the cloud service. Example: Vorbis, AAC LC, FLAC, Mp3 etc.

- Formats for storing high-quality music in a compressed manner must be chosen.

**PLA-REQ-1:** Data structure to store song IDs and song details.

**PLA-REQ-2:** Storage space on the local device to store the playlists.

- The structure must be well-defined and capable of storing necessary details.
- The structure must be designed in an efficient way to store the details in a compressed manner.

**SER-REQ-4:** A searching algorithm to return top matches quickly and efficiently.

## 6.2 Safety Requirements

Music with explicit content is present on the platform and is tagged as “Explicit”. Users have the option to filter out explicit content. Parental advisory is required.
The content we identify as explicit is based on information we receive from rights-holders, so we can’t guarantee that no explicit content will play with this setting. If a user spots a song that needs an explicit tag, they will be prompted to report it.

Warning will be given when current volume settings are unsafe for human ears.

## 6.3 Security Requirements

**Authentication Requirements:**

Data required: your username, email address, phone number, birth date, and country, password.

**Privacy (Data we collect):**

User Data:

This is the personal data that is provided by you or collected by us to enable you to sign up for and use the Streaming Service. Depending on the type of service plan you sign up for, this may include your email address, phone number, birth date, and country.

Some of the personal data we will ask you to provide is required to create your account. You also have the option to provide us with additional personal data to make your account more personalised.

The exact personal data we will collect depends on the type of service plan you sign up for, how you create an account, and whether you use third-party services (such as Facebook) to sign up and use the Streaming Service. If you use a third-party service to create an account, we will receive personal data via that third party service but only when you have consented to that third party service sharing your personal data with us. Please note that the available plans and sign-up options may differ by country.

Usage Data:

This is the personal data that is collected about you when you’re accessing and/or using the Streaming Service, including:

- Information about your type of Service plan.

- Information about your interactions with the Streaming Service, such as your search queries (including the date and time of any requests you make), streaming history, playlists you create, your library, browsing history, and interactions with the Streaming Service, content, other application users. This also may include details of your use of third-party applications in connection with the Service.
- Inferences are drawn about your interests and preferences based on your usage of the Streaming Service.

Certain technical data, which may include:

- URL information;
- online identifiers including cookie data and IP addresses;
- information about the types of devices you are using such as unique device IDs, network connection type (e.g. wifi, 3G, LTE, Bluetooth), provider, network and device performance, browser type, language, information enabling digital rights management, operating system, and application version;
- device attributes of devices on your wifi network that are available to connect to the Application Service (such as speakers);
- your non-precise location, which may be derived or inferred from specific technical data (e.g., your IP address, the language setting of your device, or payment currency), to comply with geographic requirements in our licensing agreements, and deliver personalised content and advertising to you; and
- motion-generated or orientation-generated mobile sensor data (e.g. accelerometer or gyroscope) is required to provide specific features of the Application Service to you.

**Protection against Copyright Infringement:**
Plagiarism checks for content uploaded by business users so no copyright infringement takes place and protection of intellectual property is ensured.

## 6.4 Software Quality Attributes

**Availability:** Being a streaming service, the application requires internet connection in order to work. There is the feature of offline access present which users must pay for; it enables users to download the songs on their local devices.

**Interoperability:** The application can communicate and work with different devices present on your network that have the streaming application installed. One device can be used to control another e.g.: control the music using your phone as the audio is played from your TV speakers or Car's audio system.

**Portability:** Being an On-demand streaming service, the application can be installed on a host of different devices that support it. e.g., Laptops, TVs, Phones etc.

**Reliability:** with the use of CDNs, datastores on reliable cloud services such as Google Cloud, Amazon AWS etc. add reliability with redundancy among other features ensuring that there is never a hiccup.

**Maintainability:** The servers are maintained by third-party services and ensure that the down-time is minimal, and with the presence of a large host of servers, maintenance is scheduled so that the streaming service is never interrupted.

**Usability:** Application provides a stunning yet intuitive UI, which makes it easy for users to operate.

### 6.5 Business Rules

Securing Licenses:

- Licensing Teams: Ensure that proper licenses are purchased in order to stream a given artist's content on the platform.

Streams of Revenue:

- Ad-Supported Service:
  Members who do not pay for a premium subscription are able to access the Ad-Supported Service. Such users have limited on-demand online access to the company's music catalogue. Users' streaming experience is interspersed with advertisements.
- Premium Subscriptions:
  The application offers audio-streaming services to both paying and non-paying members. However, paying members, known as Premium Subscribers, are able to enjoy unlimited online and offline access to the entire catalogue of music without having to listen to commercials.

# 7. Other Requirements

Other requirements include:

1. Ensuring artists and albums are verified, and not infringing on copyrights before uploading music onto the app.
2. Availability of certain types of music in other regions.
3. Users should be of some minimum age to create an individual account; else they must be under parental supervision.

# Appendix A: Glossary

SOL - Special Object Library. Provides specialized information resources on a particular subject (in this case, music), serves a specialized and limited clientele, and delivers specialized services to that clientele.

CDN - Content Delivery Network. Geographically distributed network of proxy servers and their data centers. Their goal is to provide high availability and performance by distributing the service spatially relative to end users.

Favourite - an indication that a user likes something; could be an artist profile, an album, a playlist, or a song

Radio - list of similar songs generated by one parameter (could be an artist, genre, or a song)

# Appendix B: Field Layout

**User:**

| Field | Length | Data Type | Description | Is Mandatory |
|---|---|---|---|---|
| user_id | 16 | Numeric | | y |
| email | 25 | Alphanumeric | | y |
| phone | 10 | Numeric | | n |
| Username | 10 | Alphanumeric | | y |
| Password | 8 | Alphanumeric | | y |
| DOB | 8 | Date | Date of Birth | y |
| Country | 2 | String | Country code (2 letter) | y |
| Type | 1 | String | R for regular, B for Business | y |

**Song:**

| Field | Length | Data Type | Description | Is Mandatory |
|---|---|---|---|---|
| song_id | 16 | Numeric | | y |
| name | 25 | Alphanumeric | | y |
| genre | 10 | String | | n |
| publish_date | 8 | Date | | n |
| artist_id | 16 | Numeric | | y |
| album_id | 16 | Numeric | | n |
| distributor | 25 | String | | y |
| credits_id | 16 | Numeric | Details of people involved y in the making of the song | y |

**Album:**

| Field | Length | Data Type | Description | Is Mandatory |
|---|---|---|---|---|
| name | 25 | Alphanumeric | | y |
| genre | 10 | String | | n |
| publish_date | 8 | Date | | n |
| artist_id | 16 | Numeric | | y |
| album_id | 16 | Numeric | | y |

**Credits:**

| Field | Length | Data Type | Description | Is Mandatory |
|---|---|---|---|---|
| performed by | 25 | String | | y |
| written by | 125 | String | | y |
| produced by | 125 | String | | y |
| source | 25 | String | distributor | y |