Software Requirements Specification

Hertz

Online music Streaming Platform

Team Members:

- Navaneeth A B 19z229
- Sudharsan V 19z249
- Suriya Prasad P 19z250
- Tharun VS 19z254
- Vishwakjith I 19z260
- Rajesh G 20z432
- Srinath S 20z433

Table of contents

Product: Project Hertz

Description: An online music streaming platform

Status: Waiting for Review

Development Status: design phase

Revisions

Latest

```
Current Version: 1.0
Current Status: Work in Progress
Date: 15-08-2021
```

Revision History

- Version 1.0
    - Date: 15-08-2021
    - Reason for Changes: initial changes

1. Introduction

1.1 Purpose

The purpose of this document is to provide a debriefed view of requirements and specifications of the project called "Hertz".

Goal of this project is to provide a website that can handle multiple functionalities to provide users a smooth and easy experience.

This document discusses the whole system from backend to user interactions.

The tools used in this project are described in this document as follows:

- Libraries used for back-end control of application
- Libraries used for UI and UX design of the application
- Database used for music referencing
- Classifiers used for classifying data and yielding recommendations

1.2 Document Conventions

- All terms are in italics style.
- Main features or important terms are in bold style.
- TBD means "To be Decided", these are the components that are not yet decided
- For more references see Terminology.

1.3 Intended Audience and Reading Suggestions

- Anyone with some basic knowledge of programming can understand this document. The document is intended for Developers, Software architects, Testers, Project managers and Documentation Writers. But anyone with a programming background and some experience with UML can understand this document.
- It is divided into 5 phases with sections 3, 4, 5 being intended for developers and software managers but other sections can be understood by anyone having little knowledge about software.

This Software Requirement Specification also includes:

- Overall description of the product
- External interface requirements
- System Features
- Other non-functional requirements

1.4 Product Scope

- Hertz is a digital extension of a *library*, with more flexible rules. Instead of books, we deal with digital copies of music. Admin can upload the crafts to the database and any user will be able to access it.
- Another key aspect of enjoying music, is the freedom of the users to compile their favorites into one folder, called a ‘Playlist’. Our project brings all the necessary elements together to present a fundamental framework of the required database for implementing the ideal vision.
- Music recommendation will involve recommendation of familiar tracks and familiar genres of music available on the internet.
- Name of the project is "Hertz". It is a Website.
- It plays music and provides suggestions based on the track which the user is listening to from the cloud.

Advantages

- It provides suggestions from the cloud.
- It uses Hertz database for getting metadata of all the music present in the user's local library and recommends tracks.
- it is not platform or service specific
- it is not bounded with any music provider services so it is suggestions are not limited to particular service
- There is no specific audience for this software. Anyone can install it and use it.

1.5 References

- This document is written in GitHub flavored Markdown.
- IEEE. IEEE Std 830-1998 IEEE Recommended Practice for Software Requirements Specifications. IEEE Computer Society, 1998.

1.6 Terminology

Term	Description
User	Any living being who is interacting with the software is a *user*. (Aware only of the front-end)
System	The package of all the components which takes input and gives output to demonstrate the features of the software is called System.
Database	Collection of information on different topics related to each other.
Playlist	The collection of songs, more like a directory.
Metadata	The set of data which describes and gives information about the sound track.

Recommender system	A system which takes a track as input and outputs a set of tracks closely related to the input.
Tags	A label attached to the track which gives extra information about it.
NIC	A network interface card (NIC) is a computer circuit board or card that is installed in a computer so that it can be connected to a network.

2. Overall Description

2.1 Product Perspective

This system consists of two components packed as a single website with multiple pages and multiple functionalities:

- Website: For searching and playing songs, View recently played and liked songs.
- Cloud: It contains the database which has the data of users and the songs as a whole.

With the website the user could login to access all the resources.

2.2 Product Functions

Using this app, users can play tracks available in offline libraries (future implementation). While playing music users can get a list of suggested tracks which are most closely related with the present track in terms of their metadata tags like singer, genre, release year, rating etc. These tracks may be present in offline libraries or online sources.

User can perform following actions:

- play/pause/stop/seek, control volume
- add tracks/ remove tracks
- get recommendation (future implementation)

2.3 User Classes and Characteristics

Almost any user will be able to easily get going with this application as it is perfectly meant for an average music lover. Users with poor internet connectivity will benefit even more because the only place we require internet connection is where we are required to update the metadata of the music for giving correct suggestions.

2.4 Operating Environment

System Requirements

- The Operating System should be capable of running browsers for playing music.
- Internet Connection is required for suggestions and metadata updation.

2.5 Design and Implementation Constraints

- For constructing the website, we have HTML 5 and for designing it we used CSS 3 (bootstrap framework).
- Cloud has the MySQL database of the website which is likely the backend.
- The backend and the frontend are linked with the help of PHP.
- MySQL databases could be viewed and altered only by the ADMIN.

2.6 User Documentation

- There is a user manual that lists all the features available for the user and methods to access them.
- "Help" option will be available in the user interface which will redirect to the HELP webpage.

2.7 Assumptions and Dependencies

We used various online open-source material for most of our project work. The following lists the various open-source material we had referred to:

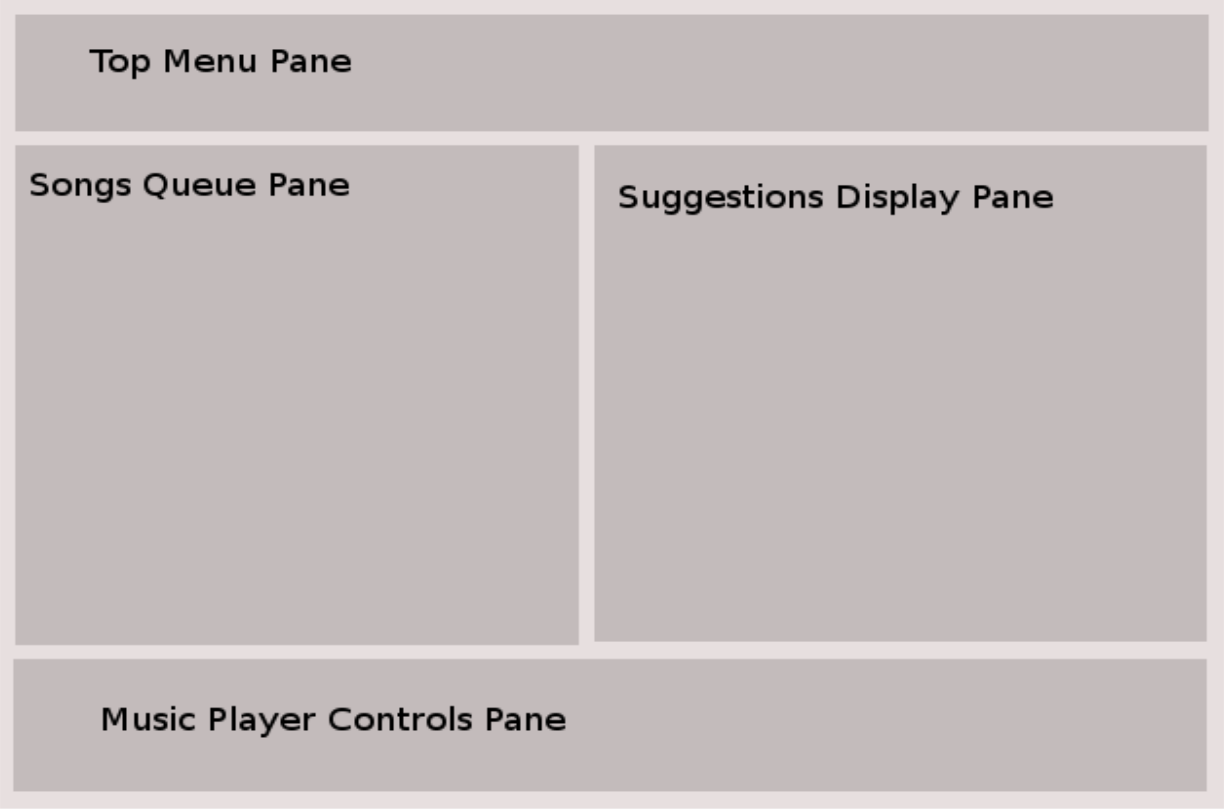
- Spotify (For Reference).
- MySQL.
- PHP with MySQL.
- HTML 5.
- CSS Twitter Bootstrap.

3. External Interface Requirements

3.1 User Interfaces

User interface is implemented in HTML, CSS, JS and PHP to link the data with the server. There is one front page which interacts with the user and it is divided into frames for different functions as shown below.

UI Mockup



3.2 Hardware Interfaces

- Input device is needed for the user to interact with the system.
- Software needs a display device to interact with users.
- Music player needs a playback device for sound output.
- Wi-Fi modem or Ethernet is needed for internet connectivity.

3.3 Software Interfaces

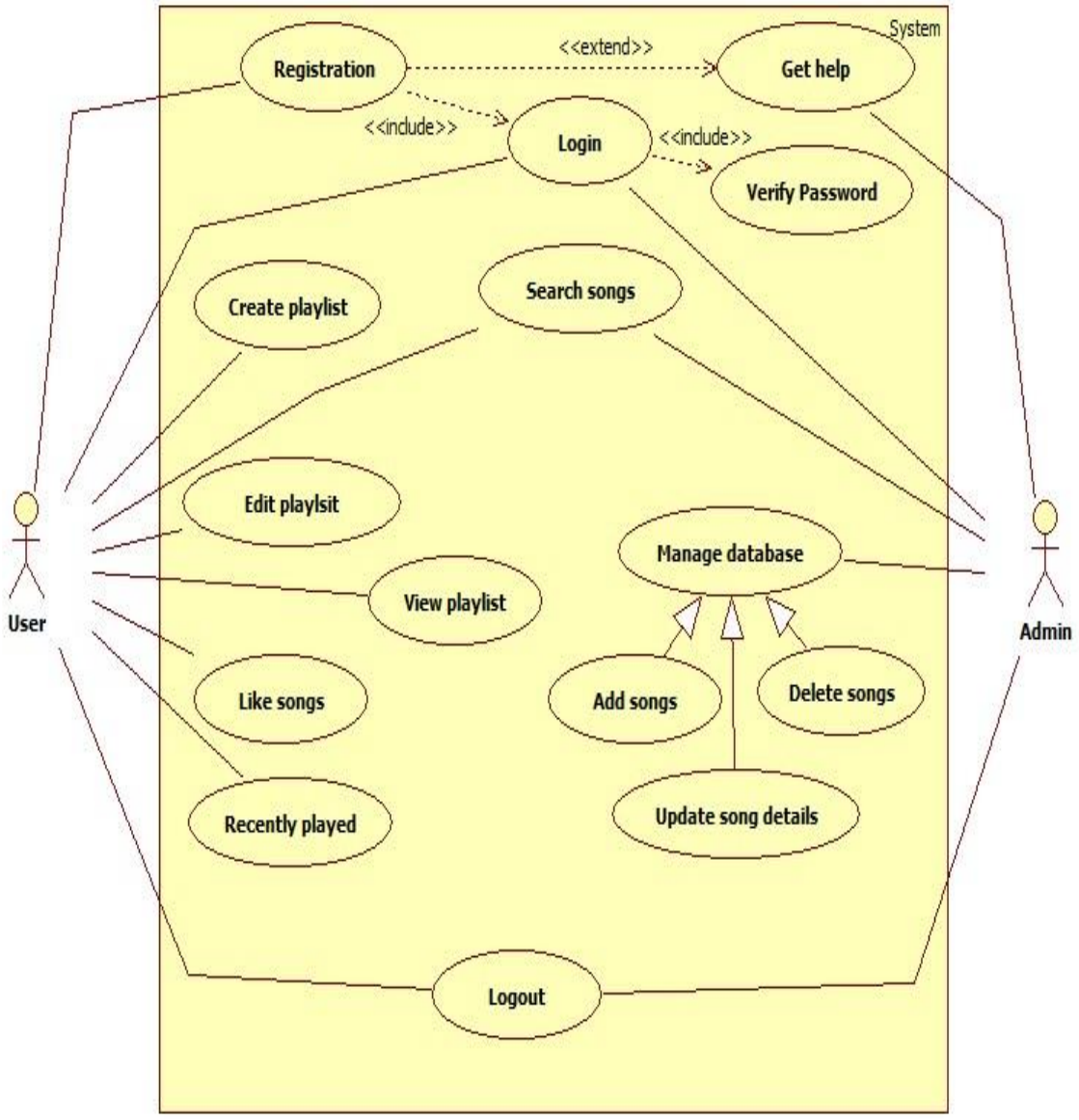
TBD (Will be Declared in Future)

3.4 Communications Interfaces

- The Internet connection is used by User/Admin to communicate with Database.
- Internet communication will be Encrypted.
- User and server databases are linked with JS and PHP

4. System Features

Following is the use case diagram for the application



Use case description table

Use Case Title (ID)	Description
Registration (UC1)	Register to the platform
Listen Music (UC2)	Generalization of control songs
Create playlist (UC3)	User can create playlist
Edit playlist (UC4)	User edits playlist
Update database (UC5)	Admin can update the database for details.
Recently played (UC6)	Last 10 songs played by users reside here
Like songs (UC7)	Favorite songs can be added to another playlist.
View playlist (UC8)	All the playlists and its songs can be viewed and heard.
Search songs (UC9)	Songs, playlists, artists, genre, album could be searched.
Login (UC10)	User can access the resources after login

Logout (UC11)	Users are restricted from accessing the website resources.
Remove Songs (UC12)	Admin can remove songs from the database.
Add songs (UC13)	Admin can add songs to the database.

Functional Requirements

Identifier for Requirement	Functional Requirement Name	Description
RQ 01	Manually update metadata	The User will be able to update the metadata of any track manually.
RQ 02	Volume control	The User will be able to increase or decrease or mute the volume of the playing track
RQ 03	Play music	The User will be able to play the track by selecting it or clicking on Play
RQ 04	Pause music	The User will be able to pause the track being able to play it again from the same timeline
RQ 05	Seek track	The User will be able to move anywhere in the timeline of the track

RQ 06	Stop music	The User will be able to stop the track which will close the track, in order for the user to play another track or exit software
RQ 07	Go to the next track	The User will be able to play the next track
RQ 08	Go to the previous track	The User will be able to play the previous track
RQ 09	Add songs	The User will be able to add songs into the desired playlist.
RQ 10	Remove songs	The User will be able to remove any track from the playlist.

Project Recommend comes with the following set of system features:

Hertz Database (Cloud)

- The Hertz database is the primary database containing user details and songs with its details.

User actions

This section will state all the use cases of the application and what the user will be able to do with the website.

Music management

The user will be able to manage music, playlists and liked songs. The Admin will be able to add/remove music files into the database with valid information.

The user can control music in the music player component by the following actions:

- Play music
- Pause music
- Control volume
- Go to the next track
- Go to the previous track
- Seek a playing track

Manual updating of metadata

The user will be able to manage metadata of the tracks that he/she chooses.

Manual recommendation of music

The user can get recommendations of a track manually by clicking on it.

A summary of the direct actions that the user can take is as follows:

User Interactions

The following are the use cases and how the actor: user interacts with them

Use case: Edit Playlist

Brief Description

The user manages tracks, can add and remove them from the playlist.

User interaction

- Add tracks from the metadata.
- The user can remove the track from the playlist.

Use case: Listen Music

Brief Description

The user can control music which means he/she can play, pause, stop, go to the next track, go to the previous track, control the volume

User interaction:

- Click events trigger all controlling of music operations.
- On playing a track, the play process triggers an event that gets recommendations for the track
- The system checks whether the metadata is already updated or not, if not then the metadata is updated in the track
- The system connects to the Hertz database to update the track metadata.
- The system then adds the songs to the recently played metadata.

Use case: Manually get recommendation

Brief Description

The user can manually get recommendations of a track other than the track that is being played.

User interaction:

- The user triggers the event of getting a recommendation of a track. The system checks whether the metadata is already updated or not, if not then the metadata is updated in the track
- The system connects to the Hertz database to update the track metadata.
- The system fetches tags of the track.
- After metadata update the local store is updated for the current track.
- The system then runs a classifier on the track and gets all suggestions for that track.
- The suggestions are updated into the local store.

Use case: Create Playlist

Brief Description

The user can also manually create a playlist.

User interaction

- The user creates a playlist to add songs to his wish.

Use case: Manually update metadata

Brief Description

The user can manually trigger updating of metadata of a track such that he/she can simply click on a track and update the metadata.

User interaction

- The user triggers an event of manually updating the metadata of a track.
- The system connects to the Hertz database.
- The system fetches the metadata of a track from the database.
- The system then updates the metadata into the local store.

5. Other Nonfunctional Requirements

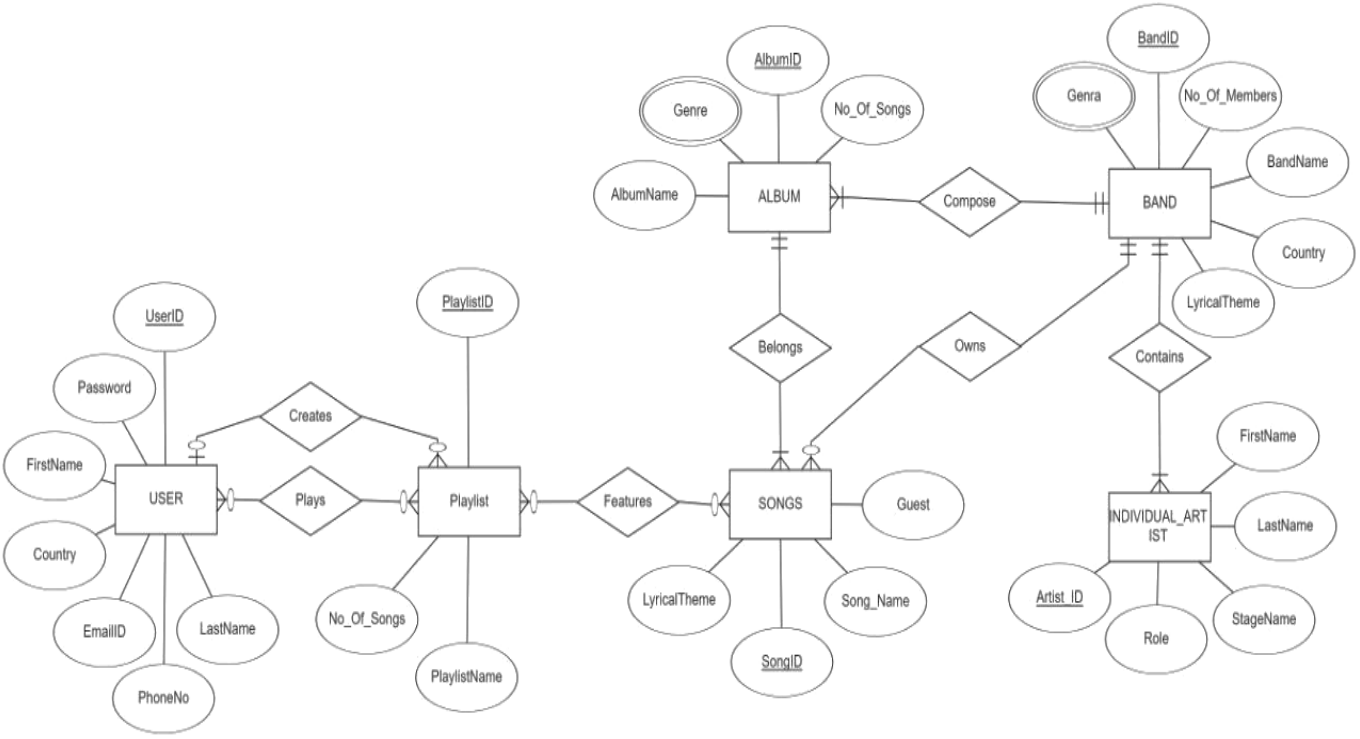
The non-functional requirements of the system are explained below.

Non-Functional Requirements	Name	Description
5.1 Performance Requirements		
NR_01	Quickness	System should be fast enough to play music and respond to any of the user actions in any way without any shattering or buffering, else it will not be a good experience.
NR_02	Robustness	System should be robust to deal and act accordingly with common error scenarios like no internet connection, unavailable metadata, unsupported file types.
NR_03	Failure Handling	In case of failures, it should be able to fail or recover quickly.
5.2 Safety Requirements		

NR_04	Exception Handling	The software should be able to restrict or warn (in the first place) the user from doing things not suitable, like, increasing volume beyond threshold, or exiting the software w/o saving the changed data.
5.3 Security Requirements		
NR_05	Encrypted Connection	Connection between user and Hertz servers should be Encrypted (HTTPS/TLS).
5.4 Software Quality Attributes		
NR_06	Memory Management	System should not leak memory.
NR_07	Compatibility	System should peacefully coexist with other software
NR_08	Error Handling	System should not cause or trigger any events that will leave Operating System in unrecoverable state
5.5 Business Rules		
NR_09	Open Source	This software is Open-Source software.

6. Other Requirements

6.1 Analysis Models



See below for reference.

USER_DETAILS entity represents anyone who is willing to access music from the Database.

PLAYLISTS entity represents the means through which the USER can access music.

USER can either create one's own playlist or play any public playlist.

SONGS entity holds all the necessary data to represent the fundamental unit of the database. SONGS can be added to the Playlist.

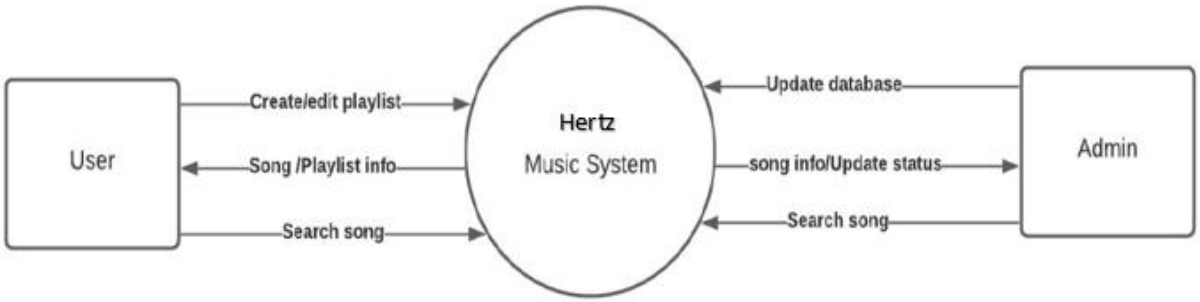
ALBUM entity doesn't represent only the parent folder of songs, but also the overall view of the album, for example: genre, artists, etc.

BAND entity represents a group of artists who create songs. This entity holds the information of the band and all their albums.

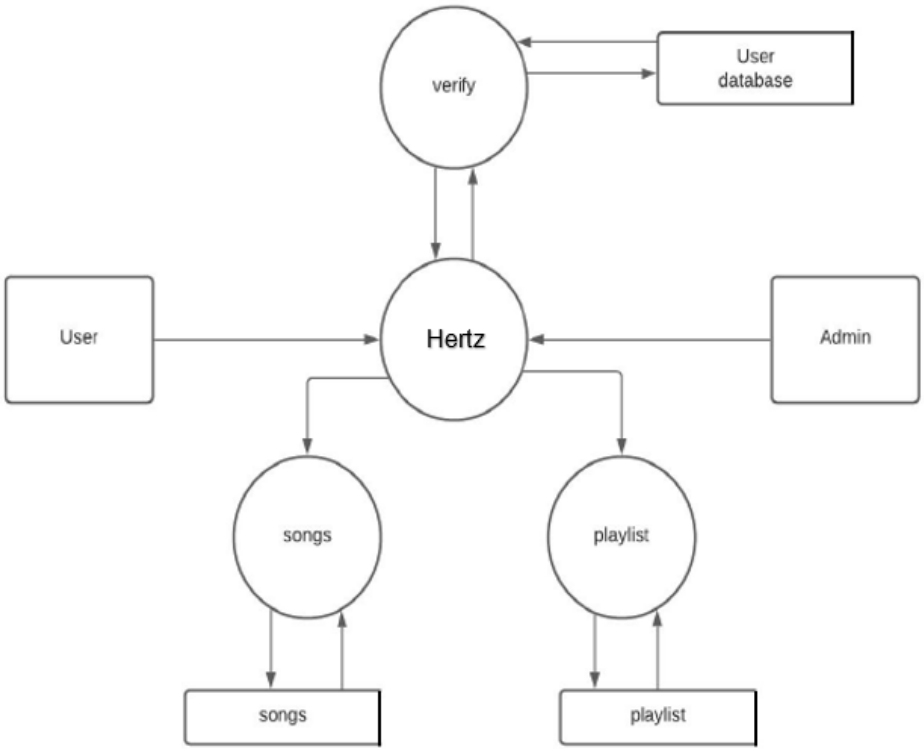
INDIVIDUAL_ARTIST represents an artist and his role.

6.2 Data Flow Diagram

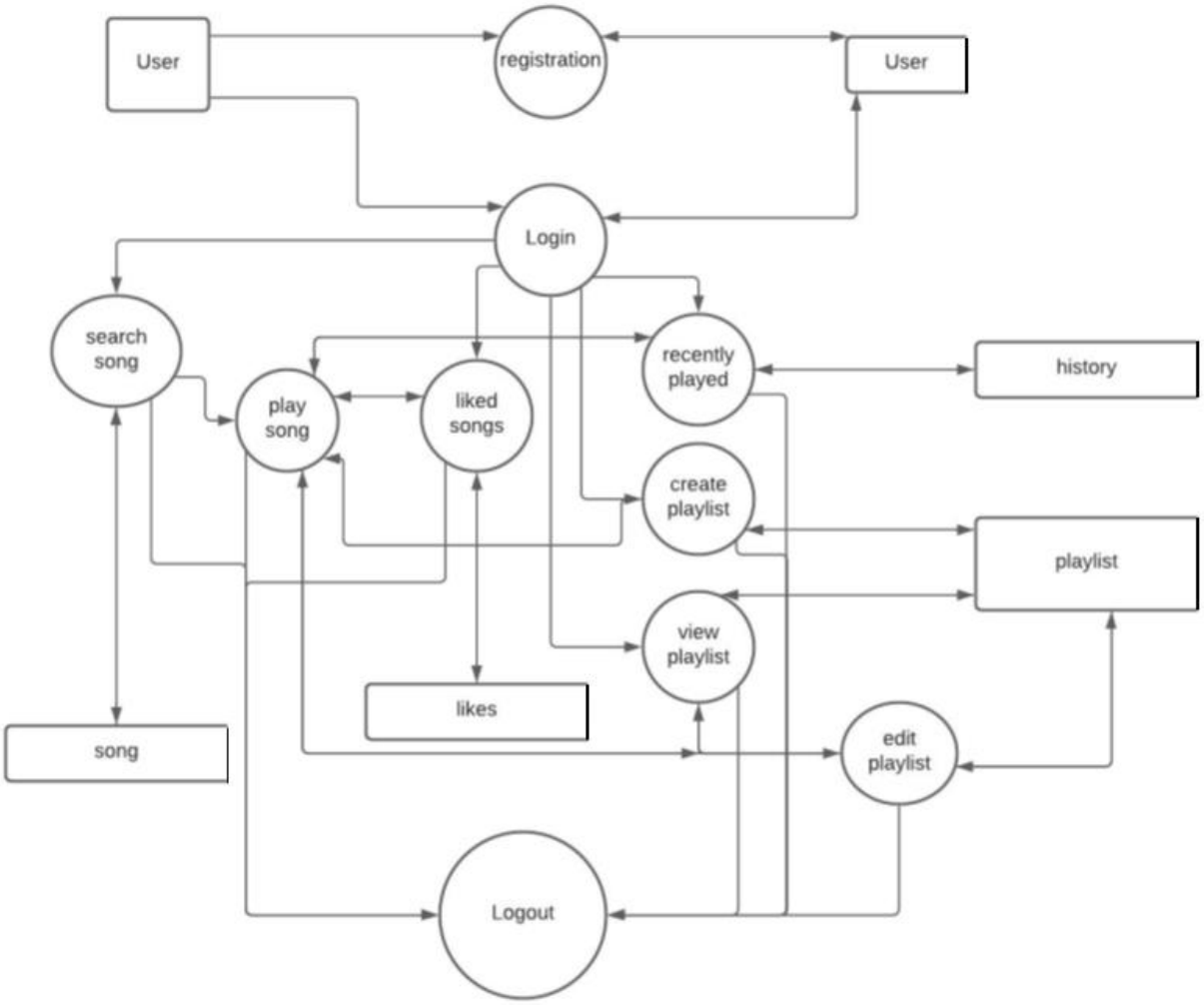
Context Diagram



Level 1 DFD



Level 2 DFD


User
registration
User
Login
search song
play song
liked songs
recently played
history
create playlist
playlist
view playlist
likes
song
edit playlist
Logout