HERTZ: AN ONLINE MUSIC STREAMING PLATFORM

Navaneeth A B (Roll No: 19Z229)
Sudharsan V (Roll No: 19z249)
Suriya Prasad P (Roll No: 19z250)
V S Tharun (Roll No: 19z254)
Vishwakjith I (Roll No: 19z260)
Rajesh G (Roll No: 20z232)
Srinath S (Roll No: 20z233)

Dissertation submitted in partial fulfillment of the requirements for the degree of

BACHELOR OF ENGINEERING

Branch: COMPUTER SCIENCE AND ENGINEERING

of Anna University

December 2021

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
PSG COLLEGE OF TECHNOLOGY
(Autonomous Institution)
COIMBATORE – 641 004

CONTENTS

CHAPTER **Page No.**

SYNOPSIS

Hertz

Online Music Streaming Platform

In this fast moving world, it is almost necessary for a person to carry music wherever one goes. So a music streaming service which can be accessed anywhere is absolutely vital. Likewise, it is also important for them to access their music though all their devices and have their progress synced across them. So the music streaming platform should be available to all the devices.

Hertz is a music streaming website that lets you play the music from anywhere with any device you have on you..
- Discover artists and tracks you'll love.
- You can organize tracks to your desire.
- Get music recommendations!

A website which lets you do all this and more.

Users can sign up with their credentials and start using the service from any device after that. All their progress: history, liked songs, personalized playlists and more will be stored and ready to be accessed when they login back.

Chapter 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

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

1.2 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.

1.3 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.

1.4 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)

1.5 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.

1.6 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.

1.7 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.

1.8 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.

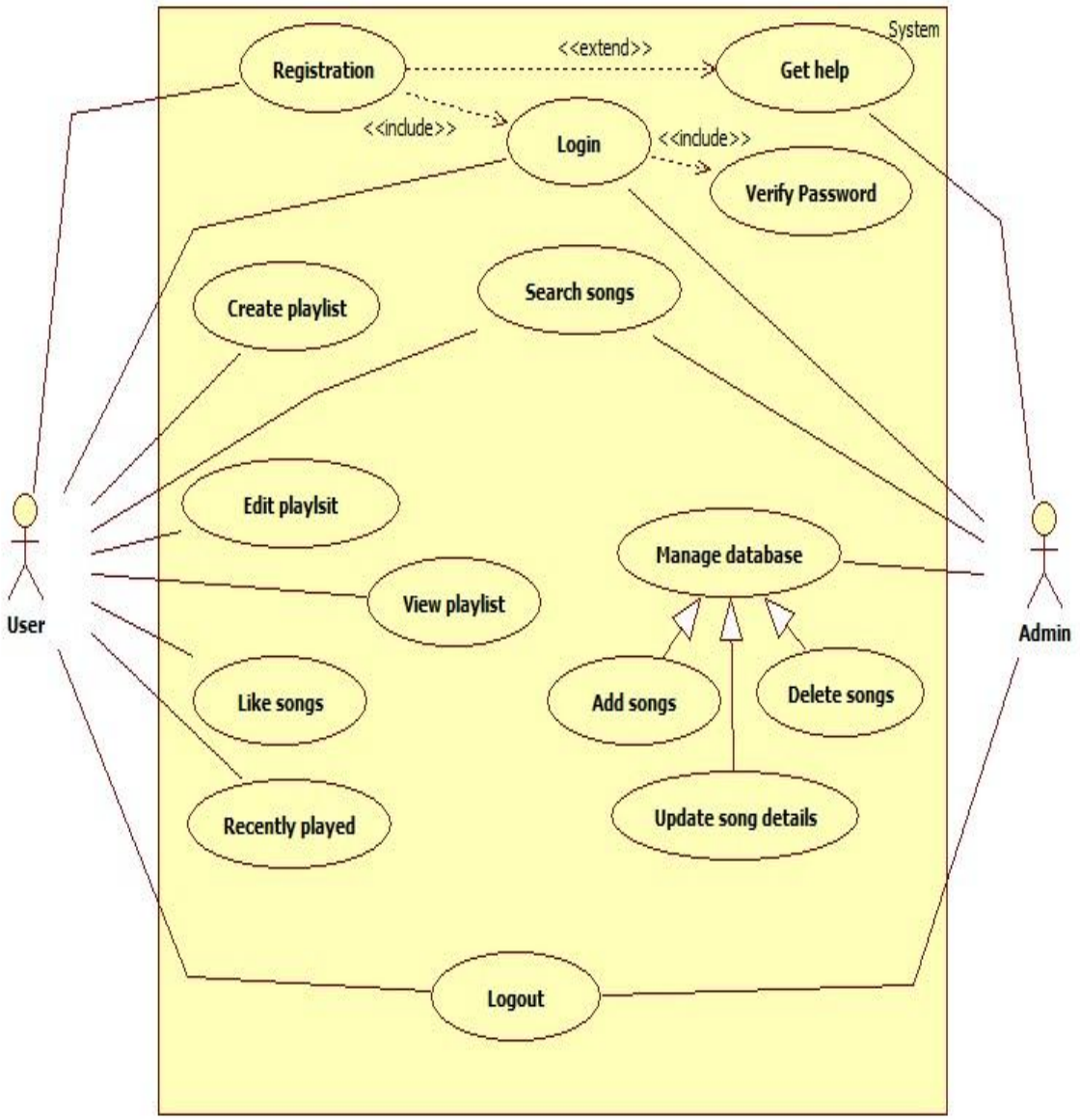
1.9 Assumptions and Dependencies

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.

Chapter 2

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)	Favourite 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 updation 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 updation 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.

2.1 Other Nonfunctional Requirements

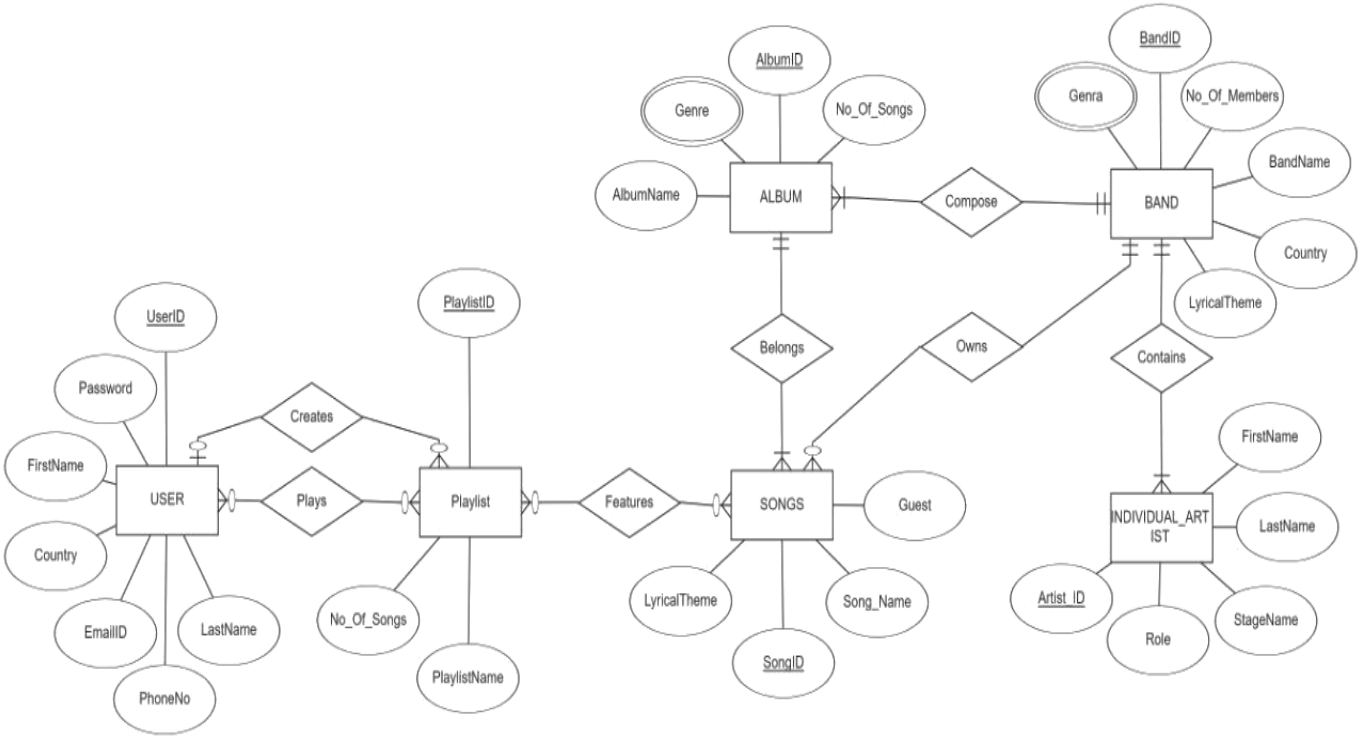
The non-functional requirements of the system are explained below.

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.

Chapter 3

3.1 ER Diagram



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.

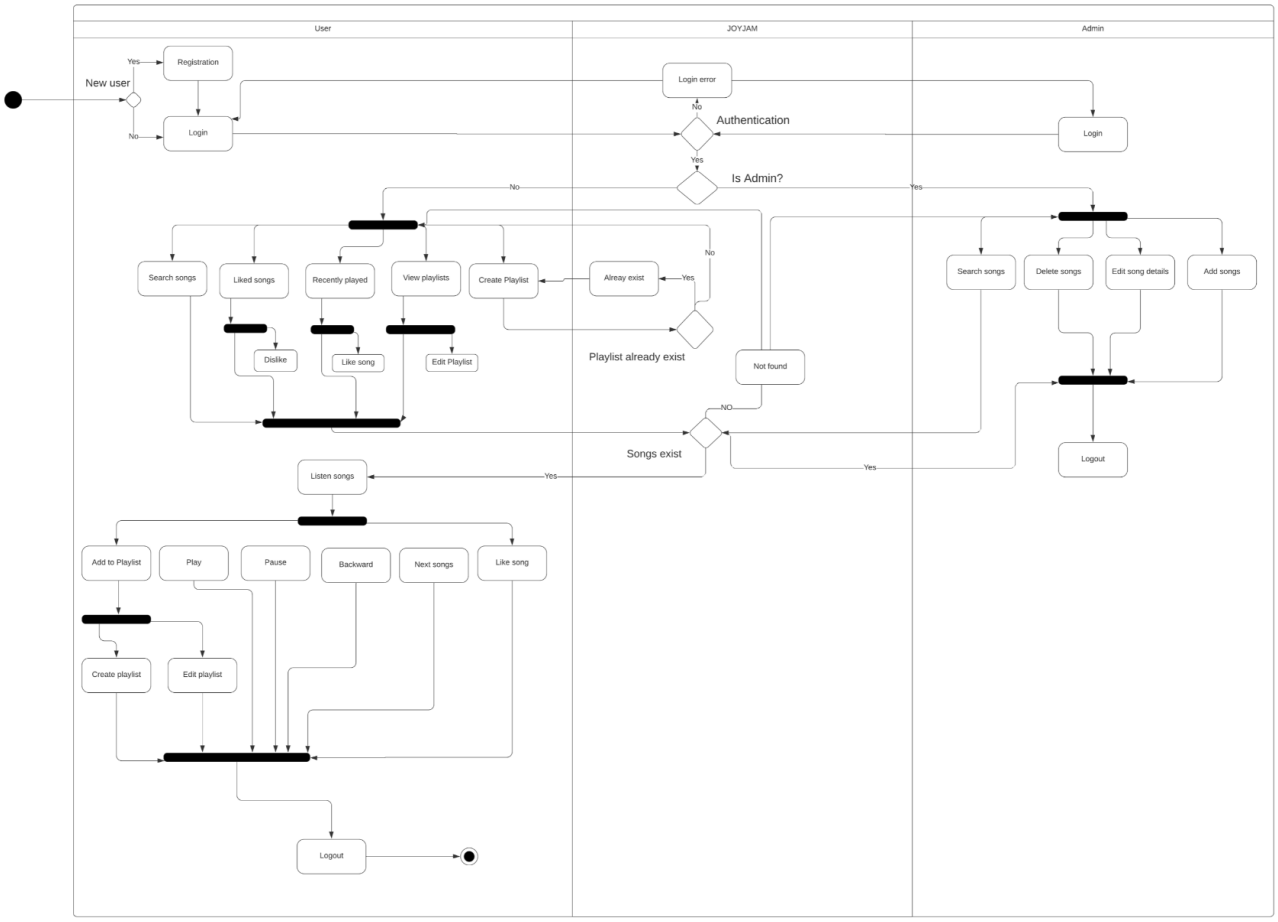
SONGS entity holds all the necessary data to represent the fundamental unit of the database. SONGS can be added to the Playlist.

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

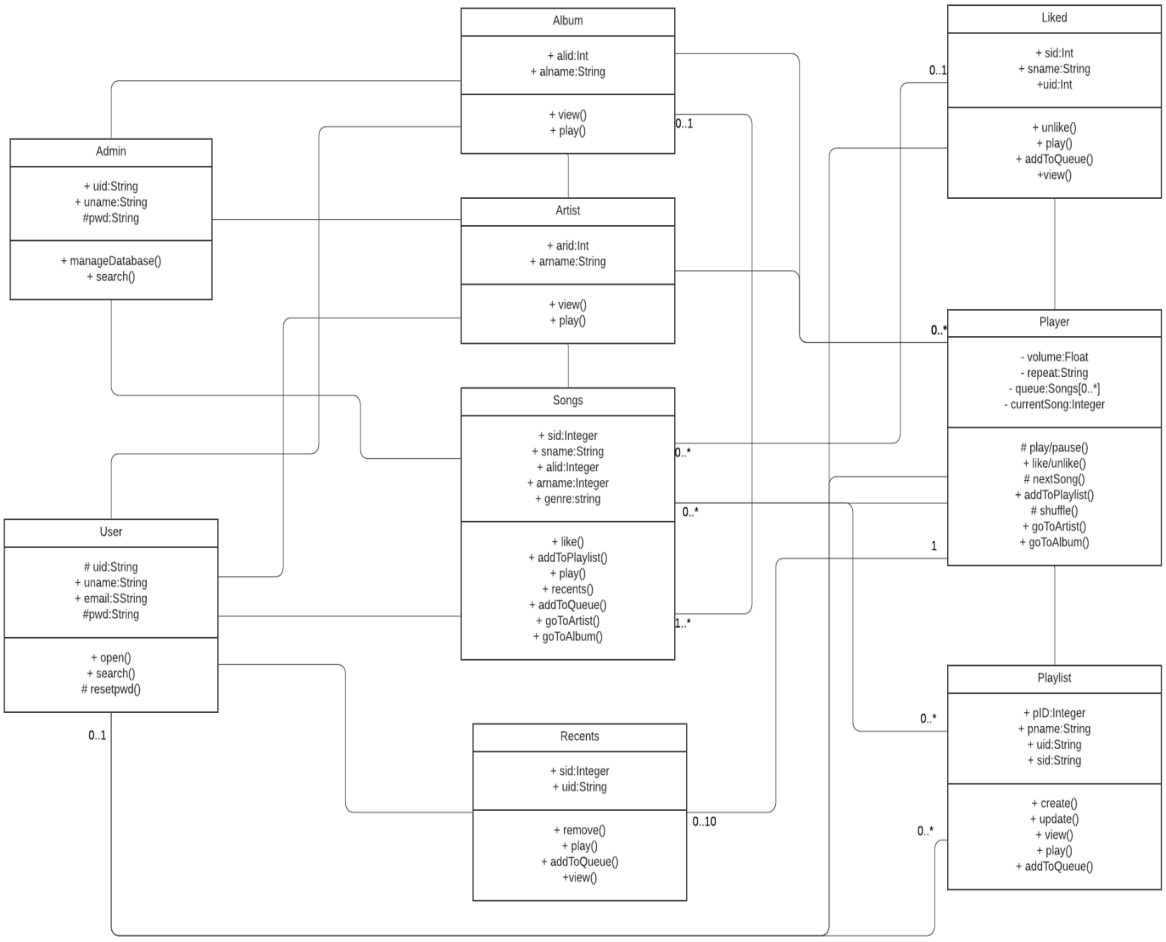
BAND entity represents a group of artists who create songs. This entity holds the information of the band and all their albums.

INDIVIDUAL_ARTIST represents an artist and his role.

3.2 Activity Diagram

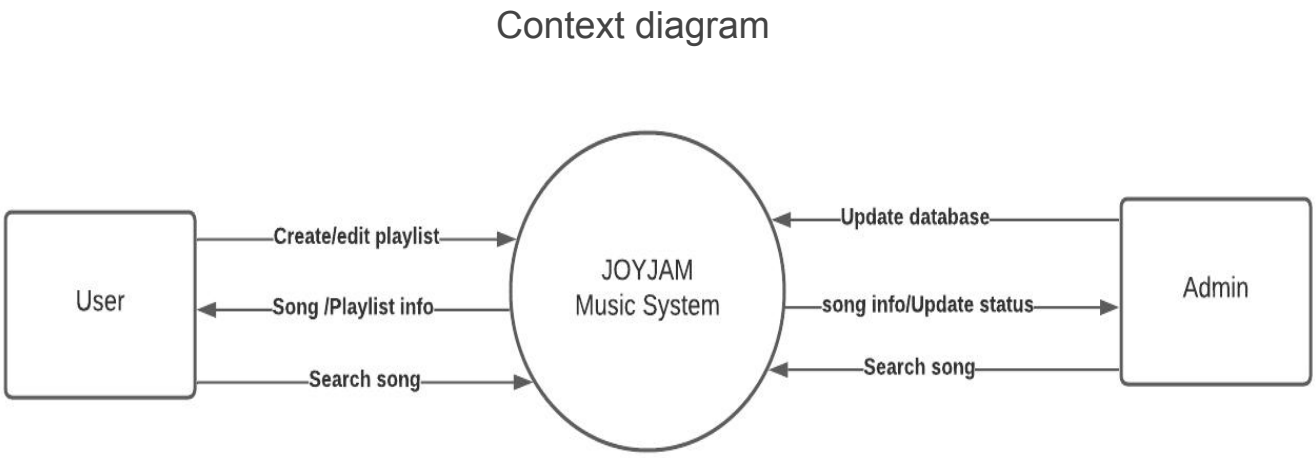


3.3 Class Diagram

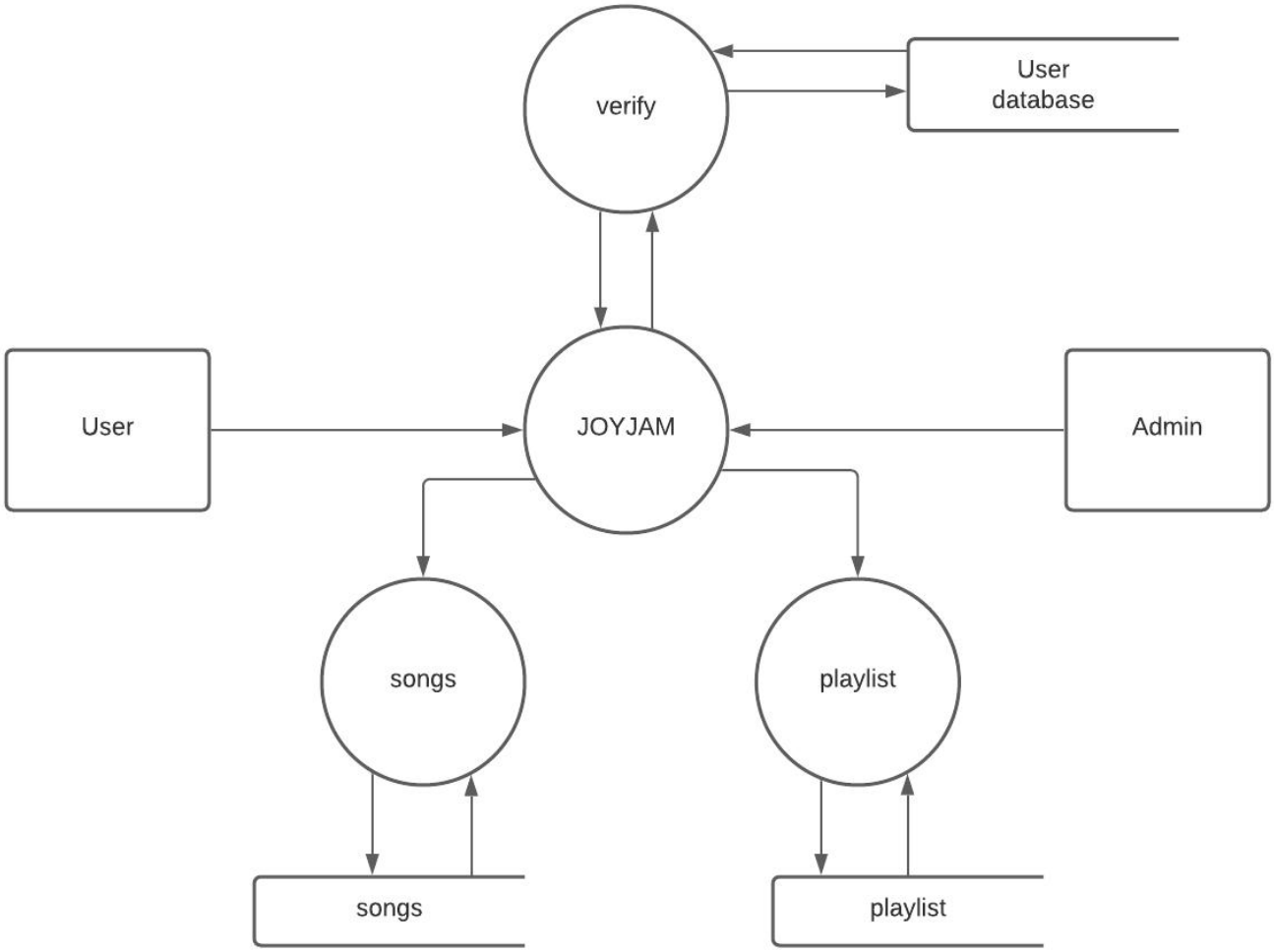


Chapter 4 - Implementation

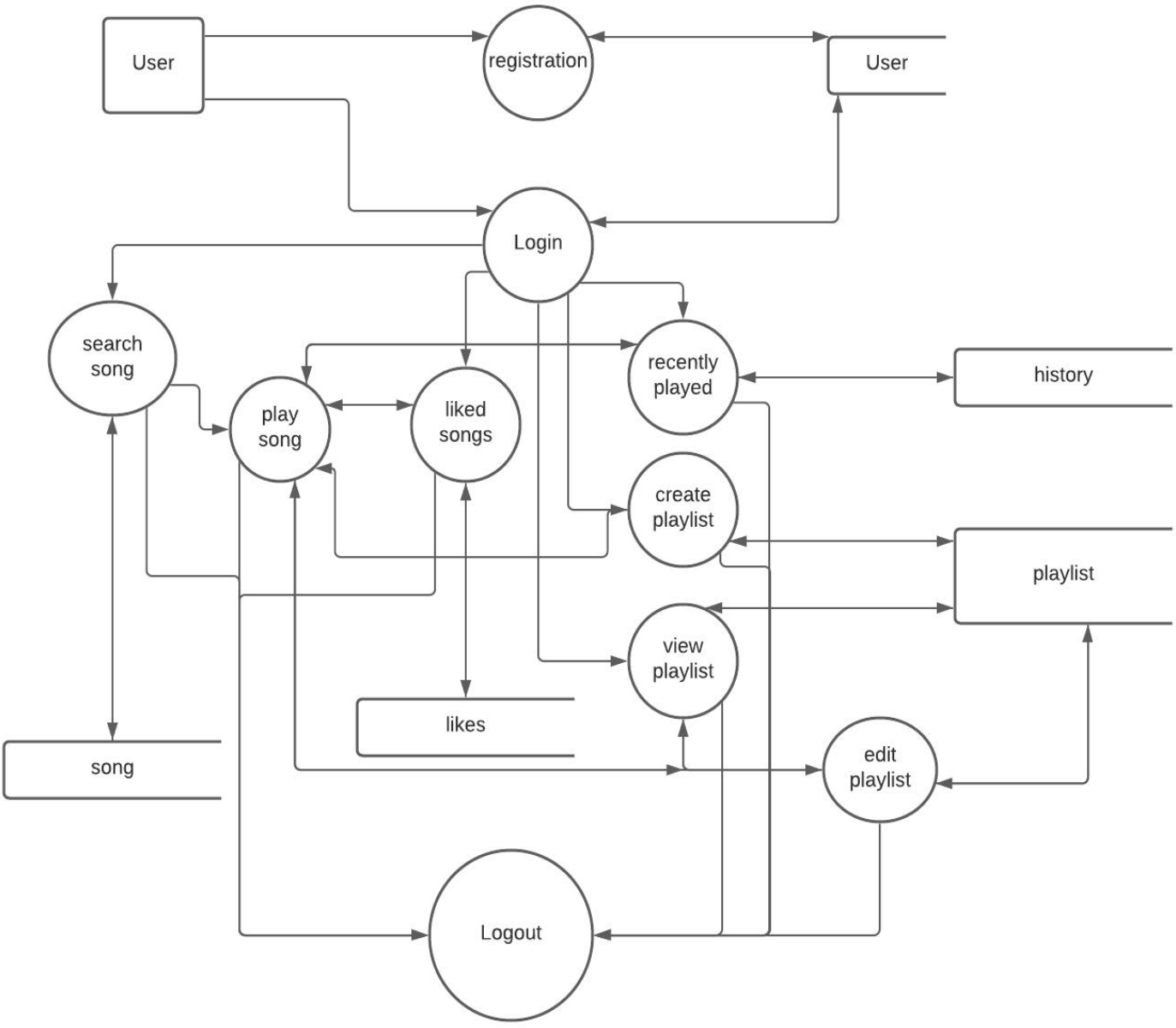
4.1 Data Flow Diagram



Level-1


verify
User database
User
JOYJAM
Admin
songs
playlist
songs
playlist

Level 2


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

Chapter 5 - Tools Used

Since our project is a web application which hosts music streaming services, our software development procedure was pretty standard. We started out by creating a database. Spent the initial days by planning a very efficient database and moved it from paper to **MySQL** after that.

Once our database was ready, we started working on Website development and database linking parallelly. For the development of our website, we used **HTML5** for website construction and **CSS 3**(Bootstrap) for website design. We also used JavaScript for a few features.

But most of the work was linking the database to the website. We used **PHP** to link the backend with the frontend. We also made use of **GitHub** to put our work together.

Once we had our website ready working on our device locally, we signed up for **Amazon Web Service** and with its help we were able to host our website and were able to access it from any device which had a browser.

Chapter 6 - Conclusion

Development

Initially we laid out the foundation, a normalized database and a basic website construction. Once we felt the ground was strong enough, we started brainstorming for creative ideas, both design and functionality wise. Since our approach was unorthodox, we did face a lot of setbacks while developing the website.

One of the most challenging problems we faced was how to design a very user-friendly website. With no experience in website design, we initially made an effort to learn about design through videos and articles. We referred to a lot of other websites with similar functionality. At times we weren't able to translate our ideas from paper to implementation. So after quite a struggle we did make a pretty user-friendly design.

The other challenging problem we faced was how to give the user music recommendations. We quickly understood what we needed to do in order to make it happen but it was hard to implement it because it tampered with the roots of the project. We had to add another table in the database called 'History' which tracks the songs a user listens to. With the help of those details, we were able to recommend music based on the user's previously heard genre and artists.

These were two of the most challenging problems we faced while developing the website. Some other minor challenges were effective normalization, play/pause tracks, hashing user passwords in the database and more. Working through these challenges made our learning experience much more interesting and effective.

How To Use

Since our project is a website there isn't a need to have directions to install it. Any user with any device with a functional browser will be able to access our website. For a new user, they will have to register using valid email ID and unique username. Recurring users can login with their credentials.

All the user's progress: history, personalized playlists, liked songs and more will be stored and synced through all the devices. When they login from any other device, their account with all their saved progress will be ready to be used again.

Bibliography

https://www.spotify.com/us/ - The Design Inspiration and Fundamental Functionalities

https://getbootstrap.com/ - Bootstrap

https://sweetalert.js.org/docs/ - A JavaScript functionality which raises alert on the existing page.

https://www.w3schools.com/php/

https://www.w3schools.com/js/

https://www.php.net/manual/en/book.session.php - Managing Sessions

https://www.php.net/manual/en/function.password-hash.php - Hashing Passwords

https://www.w3schools.com/jquery/jquery_css_classes.asp - Using CSS classes in jquery.

Appendix

Sample Code - Login Page (HTML & PHP)

```
<!doctype html>
<html lang="en">
  <head>
    <!-- Required meta tags -->
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width,
initial-scale=1, shrink-to-fit=no">
    <link
href="https://fonts.googleapis.com/css?family=Roboto:300,400&d
isplay=swap" rel="stylesheet">

    <link rel="stylesheet" href="fonts/icomoon/style.css">

    <link rel="stylesheet" href="css/owl.carousel.min.css">

    <!-- Bootstrap CSS -->
    <link rel="stylesheet" href="css/bootstrap.min.css">

    <!-- Style -->
    <link rel="stylesheet" href="css/style.css">

    <title>Login - form</title>
  </head>
  <body style="background-color: #ece0e0;">


  <div class="d-lg-flex half">
    <div class="bg order-1 order-md-2"
style="background-image:
url('images/pexels-photo-7626820.jpg');"></div>
    <div class="contents order-2 order-md-1">

      <div class="container">
        <div class="row align-items-center
justify-content-center">
          <div class="col-md-7">
```

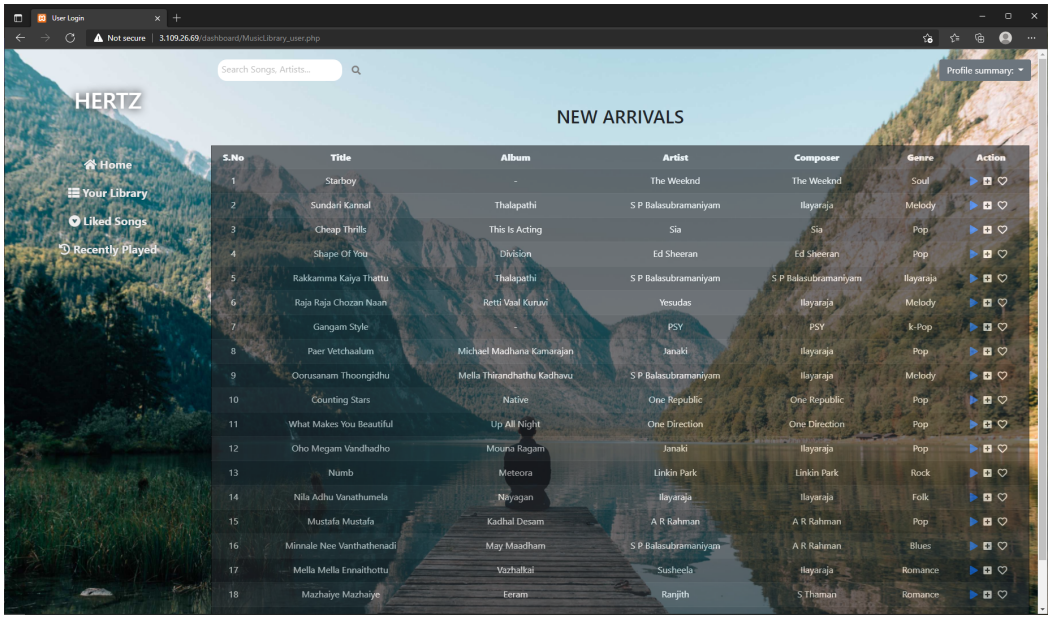
```
            <h3 style="text-align: center;"><strong
style="font-size:
2em;font-weight:900;color:navy;">HERTZ</strong></h3><br>
            <form action="<?php echo $_SERVER['PHP_SELF']; ?>"
method="post" name="loginform">
              <div class="form-group first">
                <label for="username">Username</label>
                <input type="text" class="form-control"
placeholder="Your Username" id="Uname" name="username">
              </div>
              <div class="form-group last mb-3">
                <label for="password">Password</label>
                <input type="password" class="form-control"
placeholder="Your Password" name="password" id="Pass">
              </div>

              <div class="d-flex mb-5 align-items-center">
                <label class="control control--checkbox
mb-0"><span class="caption">Remember me</span>
                    <input type="checkbox" />
                    <div class="control__indicator"></div>
                </label>
              </div>

              <input type="submit" value="Log In"
name="submit" class="btn btn-block btn-primary"><br>
              <label><button class="btn btn-block
btn-secondary" type="button" id="submit2" onclick
="javasript:NewUserRegistrationPage()">SIGN
UP</button></label>
            </form>
          </div>
        </div>
      </div>
    </div>


  </div>
```

```
    <script
src="https://unpkg.com/sweetalert/dist/sweetalert.min.js"></sc
ript>
    <script>
    //script to load new user registration page on click of
sign up button.
    function NewUserRegistrationPage() {
            window.location.assign("NewUserRegister.php");
    }
    </script>
    <script src="js/jquery-3.3.1.min.js"></script>
    <script src="js/popper.min.js"></script>
    <script src="js/bootstrap.min.js"></script>
    <script src="js/main.js"></script>
  </body>
</html>

<?php
    //creates session for each user
    session_start();

    //when user submits the form on clicking the submit
button, below code is executed in order to authenticate the
user by validating the user data in database.

    if(isset($_POST['submit']))
    {
        if (empty($_POST['username'])) {
            echo "<script>swal({title:'Enter
Username',icon:'info'});</script>";
        }
        else {
            $username = $_POST['username'];
            $password = $_POST['password'];
            $conn=connectDB();
            //validates if user with the entered credentials
exist in database. if yes proceed to next page.
            $query="select * from user where
user_name='{$username}'";
            $result=mysqli_query($conn,$query);
            if(mysqli_num_rows($result) == 1 ){
```

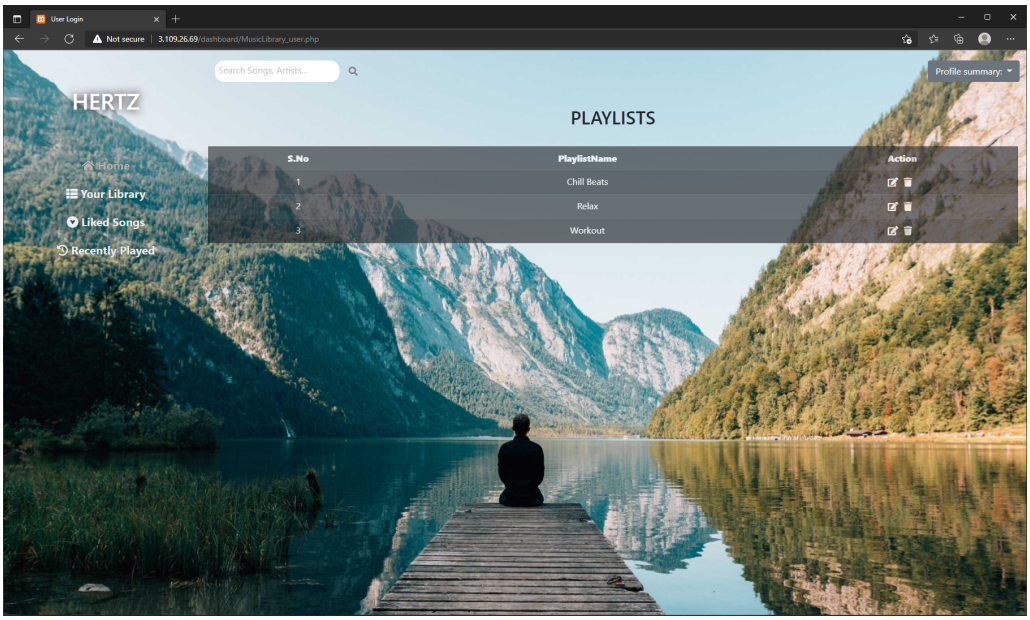
```
                $row=mysqli_fetch_array($result);
                if($row['userID'] == 1){
                    if($password == $row['password']){
                        echo "Success";
                        $_SESSION['userID'] = $row['userID'];
                    }
                    else{
                      echo "<script>swal({title:'Invalid
Username / Password',icon:'info'});</script>";
                    }
                }
                elseif($row['userID'] >= 1000){

if(password_verify($password,$row['password'])){
                        echo "Success";
                        $_SESSION['userID'] = $row['userID'];
                        //$_SESSION["user_name"] =
$[user_name];
                    }
                }
                else{
                  echo "<script>swal({title:'Invalid Username
/ Password',icon:'info'});</script>";
                }
            }
            else{
              echo "<script>swal({title:'Invalid Username /
Password',icon:'info'});</script>";
            }
        }

        //Once the user is succcessfully validated if user is
admin then redirected to MusicLibrary_admin page else to
MusicLibrary_user page.
        if(isset($_SESSION['userID'])) {
            if(($_SESSION['userID']) == 1){
                header("Location:MusicLibrary_admin.php");
            }
            else{
                header("Location:MusicLibrary_user.php");
            }
```

```
        }
    }
    //function to connect to the db with login details and the
database selection.
    //Modify the localhost,username,password,database name as
per individual credentials.
    function connectDB()
    {
        $conn = mysqli_connect("localhost:3306", "root", "",
"dbproject");
        //echo"connected DB"     ;
        if (!$conn)
        {
            echo "Error: Unable to connect to MySQL." .
PHP_EOL;
            echo "Debugging errno: " . mysqli_connect_errno()
. PHP_EOL;
            echo "Debugging error: " . mysqli_connect_error()
. PHP_EOL;
            exit;
        }
        return $conn;
    }
```

Sample Screenshots



Landing Page



Personalized Playlists