MUSIC STREAMING SYSTEM

PROJECT REPORT

Submitted in the partial fulfilment of the requirements for the award of degree in

BACHELOR OF COMPUTER APPLICATION

Submitted By,

Aswath Raj (220021084325)

Akshay Pradeep (220021084311)

Abhimanyu Vijayakumar (220021084307)

SEMESTER V

SWAMY SASWATHIKANANDA COLLEGE
POOTHOTTA

(Affiliated to Mahatma Gandhi University)
2024

SWAMY SASWATHIKANANDA COLLEGE

POOTHOTTA

(Affiliated to Mahatma Gandhi University)

CERTIFICATE

This is to certify that the project entitled “**Music Streaming System**” submitted in partial fulfilment of the requirements for the award of the degree in **Bachelor of Computer Application** is a Bonafide report of the project done by **Aswath Raj (220021084325), Akshay Pradeep (220021084311) and Abhimanyu Vijayakumar (220021084307)** during the 5th semester in the academic year 2024- 2025.

Internal Guide **Head of the Department**

Examiners:

1)..

2)..

College Seal **Department seal**

DECLARATION

I hereby declare that this project work entitled **"Ryythamwave"** is a record of the original work completed by me under the guidance of Ms. Deepa PN, **Assistant HOD**, Department of Computer Applications, **SWAMY SASWATHIKANANDA COLLEGE**, and the project work has not formed the basis for the award of any Degree/Diploma or similar title to any candidate of University.

Internal Guide	**Name and signature of candidate**
Deepa P N	Aswath Raj
	Akshay Pradeep
	Abhimanyu Vijayakumar

ACKNOWLEDGEMENT

At the outset, we thank God Almighty for making endeavour a success.

We express our gratitude to Prof. K. S. ULLAS, the principal of SWAMY SASWATHIKANANDA COLLEGE, POOTHOTTA for providing us with adequate facilities, ways and means by which we were able to complete the project work.

We express our sincere thanks to Assoc. Professor Ms. DIVYA R, Head of the Department of Computer Applications, and our project coordinators Ms. SHOBIMOL S, Ms.THUSHARAMOL P and our project guide Ms. DEEPA PN who has been showing deep interest in our project and inspired us through development by valuable suggestions, and all the faculty members of the department of Computer Applications for their sincere help and support.

CONTENTS

1)SYNOPSIS

The **Music Streaming System** project aims to create a free and ad-free music streaming website that offers users an enjoyable, uninterrupted listening experience. With music streaming becoming the dominant method for accessing music, this project provides a viable alternative to existing platforms like Spotify, Apple Music, and YouTube Music. These platforms, although popular, come with disadvantages such as subscription fees, ad interruptions, limited control, and privacy concerns.

The proposed system addresses these issues by offering cost-free access, customizable user experience, and enhanced privacy. Users can register, log in, and manage their playlists without worrying about intrusive ads or subscriptions. The system features a user authentication module for secure login and registration, and a music library module for streaming music, organizing tracks, and creating playlists.

Developed using **HTML**, **CSS**, **PHP**, and **JavaScript**, the system is technically and economically feasible. A database system (MySQL) handles user and music data, while an efficient web server hosts the site. The project ensures scalability, security, and a user-friendly interface to enhance user satisfaction. By offering a privacy-friendly and inclusive platform, this system aims to serve a wide range of users seeking a seamless music streaming experience.

2) Introduction

2.1) About the Project

The **Music Streaming System** is a web-based platform that offers users a free, ad-free experience to stream music online. It is designed to address the limitations of existing platforms, such as costly subscriptions, disruptive advertisements, and privacy concerns. This system provides a seamless interface where users can create accounts, log in securely, manage personalized playlists, and stream music. The project also includes robust security measures for protecting user data and is built to be scalable, ensuring a smooth and enjoyable music experience for a wide range of users.

2.2) Selection of Software

The software chosen for the **Music Streaming System** ensures efficiency, scalability, and security. **HTML** and **CSS** were selected for front-end development to create a visually appealing and responsive interface. **PHP** was used for server-side scripting, enabling the handling of user data, authentication, and database operations. **JavaScript** enhances interactivity, particularly in managing the music player and dynamic page updates. The backend uses **MySQL** to manage data storage for user accounts, music libraries, and playlists. This combination of technologies ensures the system is both user-friendly and capable of managing large volumes of data.

OPERATING SYSTEM

Software maintenance is the information of a software product after delivery to correct faults to improve performance or other attributes Maintenance is the ease with which a program can be corrected if any error is encountered adapted if its environment changes or enhanced if the customer desires a change in requirement Maintenance followers conversation to extend that changes are necessary to maintain satisfactory operations relative to changes in the user's environment .Maintenance often includes minor enhancements or correction to problems that surface in the system's operation Maintenance is also done based on fixing the problems reported changing the interface with other software or hardware enhancing the software.

PHP

PHP is a server-side scripting language that is used to develop Static websites or Dynamic websites or Web applications. PHP stands for Hypertext Pre-processor, that earlier stood for Personal Home Pages. PHP scripts can only be interpreted on a server that has PHP installed. The client computers accessing the PHP scripts require a web browser only.

Features of PHP

- PHP is open source and free.
- Short learning curve compared to other languages such as JSP, ASP etc.
- Large community document
- Most web hosting servers support PHP by default unlike other languages such as ASP that need IIS. This makes PHP a cost-effective choice.
- PHP is regular updated to keep abreast with the latest technology trends.
- Other benefit that you get with PHP is that it's a server-side scripting language; this means you only need to install it on the server and client computers requesting for resources from the server do not need to have PHP installed; only a web browser would be enough.
- PHP has in built support for working hand in hand with MySQL; this doesn't mean you can't use PHP with other database management systems. You can still use
  PHP with
- Oracle
- PHP is cross platform; this means you can deploy your application on several different operating systems such as windows, Linux, Mac OS etc.

MYSQL SERVER

MySQL the most popular Open-source SQL database management system is
developed distributed and supported by Oracle Corporation.
If that is what you are looking for should give it a try MySQL Server can run comfortably on a desktop or laptop, alongside your other applications, web server, and so on requiring little or no attention. If you dedicate an entire machine to MySQL you can adjust the settings to take advantage of all the memory, CPU power ,and 1/0 capacity available .MySQL can also scale up to clusters of machines networked together.
Although under constant development MySQL Server today offers a rich and useful set of functions. Its connectivity speed and security make MySQL Server highly suited for accessing databases on the internet.

Features of SQL

Tested with a broad range of different compilers.

- Works on many different platforms.
  Designed to be fully multi-threaded using kernel threads to easily use multiple CPUs if they are available.
- Provides transactional and non-transactional storage engines.
- Uses very fast B- tree disk tables (My ISAM) with index compression.
- Designed to make it relatively easy to add other storage engines this is useful if you want to provide an SQL interface for an in-house database.
- Uses a very thread-based memory allocation system.
- Executes very fast joins using an optimized nested loop join.
- Implements in-memory hash tables which are used as temporary tables.

> Implements SQL functions using a highly optimized class library that should be as fast as possible usually there is no memory allocation at all queries initialization

3) System Analysis

System analysis is the process of examining an existing system's components, structure, and operations to identify areas for improvement or design a new, more efficient system. It involves understanding the system's objectives, functionality, and problems. Through this analysis, developers gain insights into user requirements, constraints, and potential system upgrades. System analysis is crucial because it helps ensure that the proposed system will meet user needs, improve upon the limitations of the current system, and be both technically and economically feasible. It also forms the foundation for designing and implementing a more effective solution

3.1) Existing System

The current music streaming market is dominated by platforms like **Spotify**, **Apple Music**, and **YouTube Music**. These platforms offer extensive music libraries and personalized features, such as curated playlists and social sharing options.

DRAWBACKS OF EXISTING SYSTEM:

- **Subscription Costs**: Full access to features and music libraries typically requires a paid subscription, which may not be affordable for all users.
- **Ad Interruptions**: Free versions of these platforms feature advertisements, disrupting the music-listening experience.
- **Limited Customization**: Users often have limited control over playlist management and music downloads.
- **Privacy Concerns**: Many users are concerned about how large companies handle their personal data, including tracking music preferences and behavior.

3.2) Proposed System

The proposed **Music Streaming System** aims to address the drawbacks of existing platforms by providing an entirely **free and ad-free service**. This platform is designed to enhance user experience with features such as customizable playlists, music streaming, and privacy-focused data management.

ADVANTAGES OF PROPOSED SYSTEM

- **Cost-Free**: Users can access all music content without paying a subscription fee, making it inclusive for a wider audience.
- **Ad-Free Experience**: Users enjoy uninterrupted streaming with no advertisements.
- **Customizability**: Users can create and manage their own playlists, enhancing their experience.
- **Enhanced Privacy**: The system ensures that user data is kept secure, addressing privacy concerns.

Module Description

The most creative and challenging phase of the system development is system design. It provides the understanding and procedural details necessary for implementing the system recommended in the feasibility study. Design goes through the logical and physical stages of development.

System has 2 modules they are:

1. User Authentication Module
2. Music Library and Streaming Module

1. User Authentication Module

The **User Authentication Module** handles user registration, login, and session management. It ensures the secure storage of user data, including usernames, passwords, and email addresses.

Features include:

- **User Registration**: New users can sign up by providing details such as a username, password, and email.
- **Login**: Registered users can log in securely.
- **Password Management**: Users can reset or change their passwords.
- **Session Management**: Keeps users logged in during their interactions on the site.

2. Music Library and Streaming Module

The **Music Library and Streaming Module** allows users to browse, search, and stream music. Administrators can upload new music to the library, which is organized by genres, artists, and albums. Users can create playlists, stream tracks, and download songs.

Features include:

- **Music Upload**: Admins upload music tracks to the platform.
- **Music Catalog**: Tracks are categorized by genre, artist, and album.
- **Playlist Creation**: Users can create and manage personalized playlists.
- **Streaming**: Allows users to stream music directly from the website.
- **Player Controls**: Includes play, pause, skip, and volume controls.

3.3) Feasibility Study

A feasibility study is undertaken to determine the possibility or probability of either improving the existing system or developing a completely new system.it helps to obtain an overview of the problem and to get rough assessment of whether feasible solution exists. This is essential to avoid committing large resources to a project and need for Feasibility Study.

The feasibility study is need to

- Answer the question whether a new system is to be installed or not?
- Determine the potential of the existing system.
- Improve the existing system.
- Know what should be embedded in the new system.
- Define the problems and objective involved in a system.
- Avoid costly repairs at later stage when the system is implemented.
- Avoid crash implemented of a new system.

3.3.1) Economical Feasibility

This assessment typically involves a cost/benefits analysis of the project, helping organizations determine the validity, cost and benefits associated with a project financial resources are allocated.
Development costs are minimal, requiring time for coding, web hosting, and domain registration. Maintenance costs will involve updating the website and maintaining the server.

3.3.2) Technical Feasibility

Technical feasibility evaluates the technical complexity of the system and often involves determining whether system can be implemented with state-of-the-art techniques and tools. The project is technically feasible using common web development tools (HTML, CSS, PHP, JavaScript), with a database (MySQL) for user and music data management. A web server (Apache or Nginx) will handle requests, and a media server will facilitate audio streaming.

3.3.3) Social Feasibility

This system promotes inclusivity by offering an ad-free, free-of-cost music experience, ensuring users from various backgrounds can access the platform without financial or privacy concerns.

4) Requirement Analysis

Requirement analysis task is a process of discovery, refinement, modelling and specification both the developers and customer take an activity role in requirement analysis can be divided into:
4.1 Problem recognition
4.2 Problem evaluation & synthesis
4.3 Modelling

4.1) Problem Recognition

The existing music streaming platforms come with issues like subscription costs, ad interruptions, limited control over playlists, and privacy concerns. Users seek a cost-free, ad-free, and customizable music streaming platform that provides a seamless experience without compromising data privacy.

4.2) Problem Evaluation

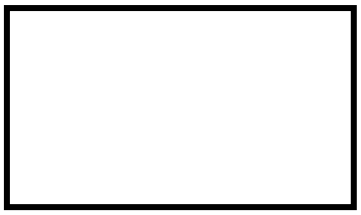
To address these issues, the proposed system will eliminate subscription fees and advertisements while offering users full control over playlist management. Additionally, it will prioritize privacy and data security, ensuring that user information is protected from unauthorized access.

4.3) Modelling

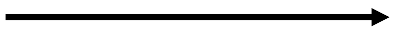
The system is modelled with two primary modules: a **User Authentication Module** for secure login and session management, and a **Music Library and Streaming Module** that enables music uploads, streaming, and playlist creation. This model ensures user accessibility and customization.

DATA FLOW DIAGRAM (DFD)

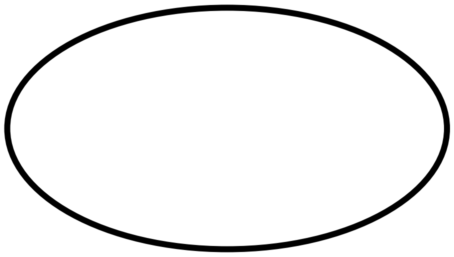
A DFD, also known as the bubble chart, has the purpose of clarifying system requirements and identifying major information that will become programs in system design. A FD is a pictorial representation of network that describes the flow of data through a system. The symbols used in Data Flow Diagram are:



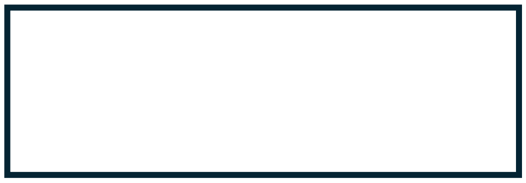
- It represents a data source or destination



- Represent flow of data



- Represents a process that transforms data



- Represent data storage

5) System Specification

5.1) Hardware Configuration

• **Processor**	:	Intel i5 or higher.
• **RAM**	:	8 GB or more.
• **Storage**	:	Minimum 500 GB HDD or SSD for fast performance.
• **Server**	:	Web server hosting (Apache/Nginx) with at least 2 GB of RAM.

5.2) Software Configuration

• **Operating System**	:	Windows, Linux, or macOS.
• **Development Environment**	:	VS Code, PHP 7+, MySQL, HTML5, CSS3, JavaScript.
• **Web Server**	:	Apache or Nginx.
• **Browser Support**	:	Chrome, Firefox, Safari.

6) System Design

System design is a process of developing specification for candidate system that meet the criteria established in the system analysis. Major step in design is the preparation of the input forms output reports in a form application to the user.
The main objective of the system design is to use the package easily by any computer operation.

System design is the creative act of invention, developing new inputs, a database, offline files, method procedure and output for processing business to meet an organization objective. System design builds information gathered during the system analysis.

6.1) Data Design

Data design creates a model of data or information that is represented at a higher level of abstraction. The structure of data has always been an important part of software design. The software design activities translate this requirement model into data structure at software component level. Data design required to manage the large volume of information. In this system, normalization process, the redundant field will be eliminated finally produce the efficient table.

The system is designed with multiple databases to handle:

- User details (user ID, username, email, password).
- Music library (music ID, title, artist, genre).
- Playlist management (playlist ID, user ID, music tracks)

6.2) Architectural Design

The architecture follows the client-server model. The client side (frontend) handles user interactions, while the server side (backend) manages data storage, user sessions, and music streaming functionalities.

6.3) Procedural Design

Procedural design or component level design occurs after data, architectural and interface design must be translated into operational software. The procedural design for each component, represented in graphical, tabular or text-based notation, is primary work product produced during component level design.

The website's procedures are organized into distinct modules:

- **User Authentication**: Handles login, registration, and session management.
- **Music Streaming**: Manages music uploads, streaming, playlist creation, and search functionalities

6.4) Interface Design

The interface design focuses on simplicity and user-friendliness. It features an intuitive layout with easy navigation. Users can access the music library, create playlists, and control playback with minimal effort. The design incorporates responsive elements, ensuring compatibility across devices, enhancing the overall user experience.

7) Coding

A coding has provided a brief identification of data item and replace longer description that would be more awkward to store and manipulate. A code can be defined as a group of characters used to identify an item of data, while identification is main function of a code. A code may also show relationships between items of data.
A code plan identifies the characteristics that needed to be contained within the code. Only information that makes possible efficient identification and retrieval of coded items should be chosen. The method chosen must have following features:

- Expandable : codes must provide space for additional entries that may be required entries that may be required
- Precise : The code must identify the specific item.
- Concise : The code must be brief, yet it should adequately describe the items.
- Meaningful : The code must be useful to that people dealing with it. If possible, it should indicate some characteristics for the item.
- Operable : The code should be compatible with present and anticipated methods of data processing.

A coding dictionary is often developed to make it easier for human to work with the codes. It is a listening of code and their corresponding data items. The dictionary allows one to translate the code into identification of data or to determine the code for a particular item.

8) System Testing

Testing is the process of executing the program with intend of findings error. System testing is the stage of implication which is aimed at ensuring that the system works accurately and efficiently before the live operation commences. The testing is vital to the success of system. System testing makes a logical assumption at the part of the system is correct the goal will be success achieved.

The recovery and the usability resets. A series of test, online, response, volume, stress, ready for the user acceptance test.

There are 3 types of testing being implemented:

- Unit testing
- Integration
- Validation testing

8.1) Testing Process

Software testing is the stage of implementation, which is aimed at ensuring that t
system works accurately and efficiently before the live operation commences. Testing is vital to the success of the system. Testing is the process if executing program with the explicit intention of finding errors that is making the program to fail. Analyst knows that an effective testing program does not guarantee system reliability. Therefore, reliability must be design into system. An elaborate testing of data is prepared, and system is tested using this test data. While testing error noted and correction are made. A series of testing are performed for the proposed system before the system is ready for the user acceptance testing.

8.2) Unit Testing

Unit testing involves testing individual components or modules of the **Music Streaming System** to ensure they function correctly in isolation. Each part, such as the user authentication module and music player, is examined using test cases that evaluate specific functionalities, such as login validation and playlist management.

Automated testing tools may be employed to streamline this process, enabling quick detection of errors or bugs within the code. By conducting thorough unit testing, developers can identify and rectify issues at an early stage, reducing the likelihood of defects in the final product and enhancing overall system reliability.

8.3) Integration Testing

Integration testing focuses on assessing the interactions between different modules of the **Music Streaming System**. After individual units have been tested, this phase ensures that combined components, such as user authentication and music library modules, function correctly together. Test scenarios will evaluate how well these modules communicate and share data, such as user credentials and music information.

Integration testing aims to identify any interface mismatches or logical errors that may arise when modules are combined. Successful integration testing helps to ensure that the entire system works as intended, providing a smooth user experience across all functionalities.

8.4) Validation Testing

Validation testing verifies that the **Music Streaming System** meets user requirements and functional specifications. It assesses system performance, security, and usability under various conditions. This phase ensures that the system delivers a satisfactory user experience, confirming that it behaves as expected and fulfils all outlined objectives

9) SYSTEM IMPLEMENTATION

9.1) IMPLEMENTATION PROCEDURE

A crucial phase in the **Music Streaming System** development life cycle is the successful implementation of the new system design. This phase involves configuring the system, setting up necessary infrastructure (such as hosting servers and databases), and ensuring user and admin modules function seamlessly. Additionally, the system requires user authentication to be integrated, along with features like music streaming, playlist management, and category organization, before going live.

The implementation process can only proceed after thorough testing of critical components, including the playback timeline, search functionality, and admin panel operations. This approach minimizes risks and ensures the system meets its specifications. For added security, the system is tested iteratively, and fallback measures are established to address potential errors or limitations in managing music files or user interactions.

Post-implementation, regular maintenance ensures the **Music Streaming System** continues to operate effectively. This includes monitoring system performance, updating the music library, fixing bugs, and implementing minor enhancements, such as improving search functionality or optimizing the user interface.

Maintenance also accounts for changes in user preferences or updates in technology. For example, incorporating Elasticsearch for advanced search capabilities or introducing new categories or playlists based on user demand ensures the system remains competitive and user-friendly.

10) SOFTWARE MAINTENANCE

Software maintenance is a critical phase in ensuring the long-term success and usability of the **Music Streaming System**. This phase involves addressing any issues that arise post-implementation and adapting the system to meet evolving user needs and technological advancements.

Key Maintenance Activities:

1. **Bug Fixes and Updates:**
   Regularly identifying and resolving bugs, such as playback errors, timeline inconsistencies, or admin panel glitches, ensures uninterrupted music streaming and user satisfaction.
2. **Feature Enhancements:**
   Introducing minor improvements, such as refining the search functionality using Elasticsearch or adding new playlist customization options, keeping the system dynamic and user-friendly.
3. **Database Management:**
   Periodically updating and optimizing the tracks database ensures efficient storage and retrieval of music files, categories, and user data.
4. **Security Patches:**
   Implementing security measures, such as updates to the user authentication module, protects user data and prevents unauthorized access.
5. **Performance Monitoring:**
   Monitoring the system's performance ensures that the music library, playback timeline, and admin panel continue to operate efficiently, even under increased user demand.

11) CONCLUSION

The **Music Streaming System** was developed to provide users with a seamless and enjoyable music streaming experience. By integrating key features such as user authentication, playlist management, music playback controls, and an admin panel, the system offers a user-friendly and efficient platform for discovering and enjoying music.

The implementation of robust technologies ensures smooth functionality, while thorough testing and iterative development have minimized potential errors. The use of dynamic features, such as category organization and a timeline slider for playback, enhances the overall user experience.

This project has not only demonstrated the feasibility of creating a modern music streaming platform but also highlighted the importance of regular maintenance and adaptability. Future enhancements, including advanced search functionality using Elasticsearch and personalized recommendations, will ensure users will continue to meet user demands and stay relevant in a competitive market.

Through this project, a deeper understanding of system design, database management, and web application development has been achieved, laying a strong foundation for further innovation in the field of music streaming.

12.1 APPENDIX A (Tables)

DATABASE DESIGN

TABLE NAME: admin_table

Field Name	Datatype	Constraint	Description
id	Int(11)	Primary key	Id for admin
username	Varchar(50)	Key2	Username for admin
password	Varchar(255)	Key5	Password for admin
email	Varchar(155)	Key5	Email for admin
profile_picture	Varchar(255)	Null	Admin Profile picture

TABLE NAME: albums

Field Name	Datatype	Constraint	Description
id	Int(11)	Primary key	Id for albums
title	Varchar(250)	Key2	Title for album
artist	Int(11)	Key5	Artist of the album
genre	Int(11)	Key5	Genre of the album
artworkPath	Varchar(500)	Key5	Artwork for album

TABLE NAME: artists

Field Name	Datatype	Constraint	Description
id	Int(11)	Primary key	Id for artist
name	Varchar(50)	Key5	Name of the artist

TABLE NAME: genres

Field Name	Datatype	Constraint	Description
id	Int(11)	Primary key	Id for genres
name	Varchar(50)	Key5	Name of the genres

TABLE NAME: playlists

Field Name	Datatype	Constraint	Description
id	Int(11)	Primary key	Id for plsylist
name	Varchar(50)	Key5	Name of the playlist
owner	Varchar(50)	Key5	Playlist owner name
dateCreated	datatime	Key5	Playlist created date

TABLE NAME: playlistsongs

Field Name	Datatype	Constraint	Description
id	Int(11)	Primary key	Id for plsylistsong
singsid	Int(11)	Key2	Song id
playlist	Int(11)	Key2	Playlist id
playlistOrder	Int(11)	Key5	Sort of playlist

TABLE NAME: songs

Field Name	Datatype	Constraint	Description
id	Int(11)	Primary key	Id for songs
title	Varchar(50)	Key5	Name of the song
artist	Int(11)	Key2	Id for artist
album	Int(11)	Key2	Id for album
genre	Int(11)	Key2	Id for genres
duration	Varchar(8)	Key5	Duration of the song
path	Varchar(500)	Key5	Path to the song
albumOrder	Int(11)	Key5	Sort albums
plays	Int(11)	Key5	Total playbacks

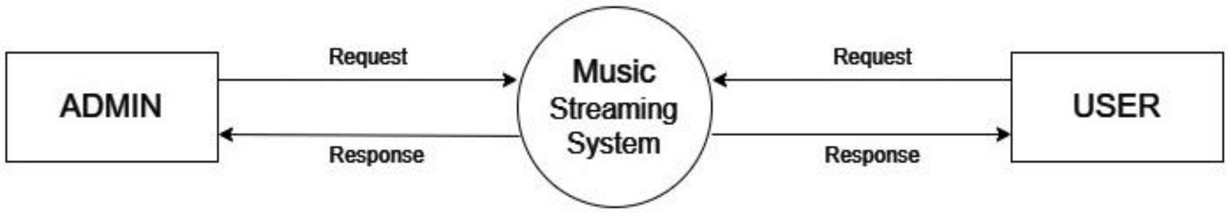
TABLE NAME: traffic

Field Name	Datatype	Constraint	Description
id	Int(11)	Primary key	Id for artist
page	Varchar(255)	Key5	Name of the page
date	datetime	Key5	Date of the login
ip_address	Varchar(45)	Key5	Ip address of the user

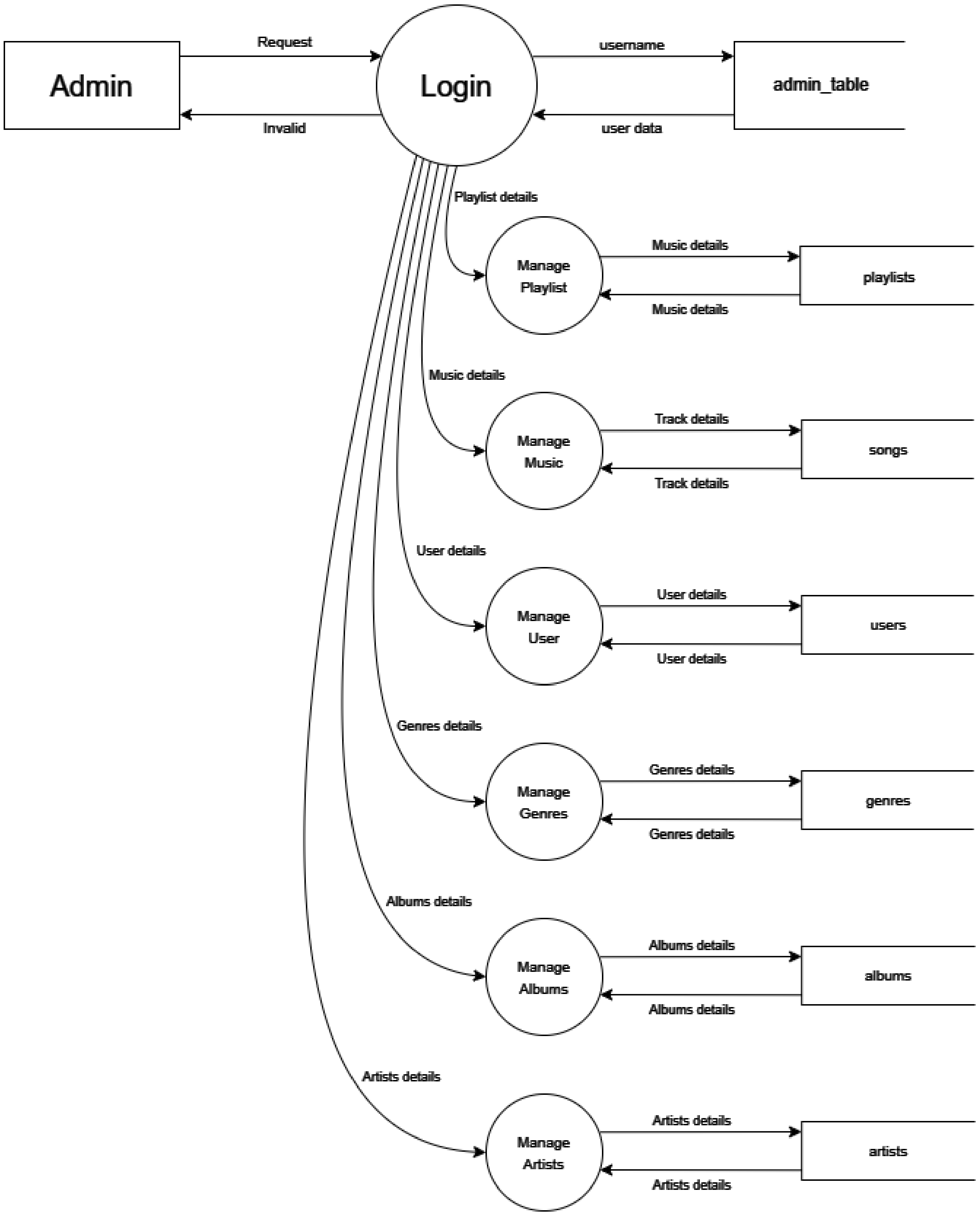
TABLE NAME: users

Field Name	Datatype	Constraint	Description
id	Int(11)	Primary key	Id for artist
username	Varchar(25)	Key5	Username for login
firstname	Varchar(50)	Key5	User First Name
lastname	Varchar(50)	Key5	User Last Name
email	Varchar(200)	Key5	User Email Address
password	Varchar(32)	Key5	User Password
signUpDate	datetime	Key5	SignUp date of User
profilePic	Varchar(500)	Key5	User profile Pic path

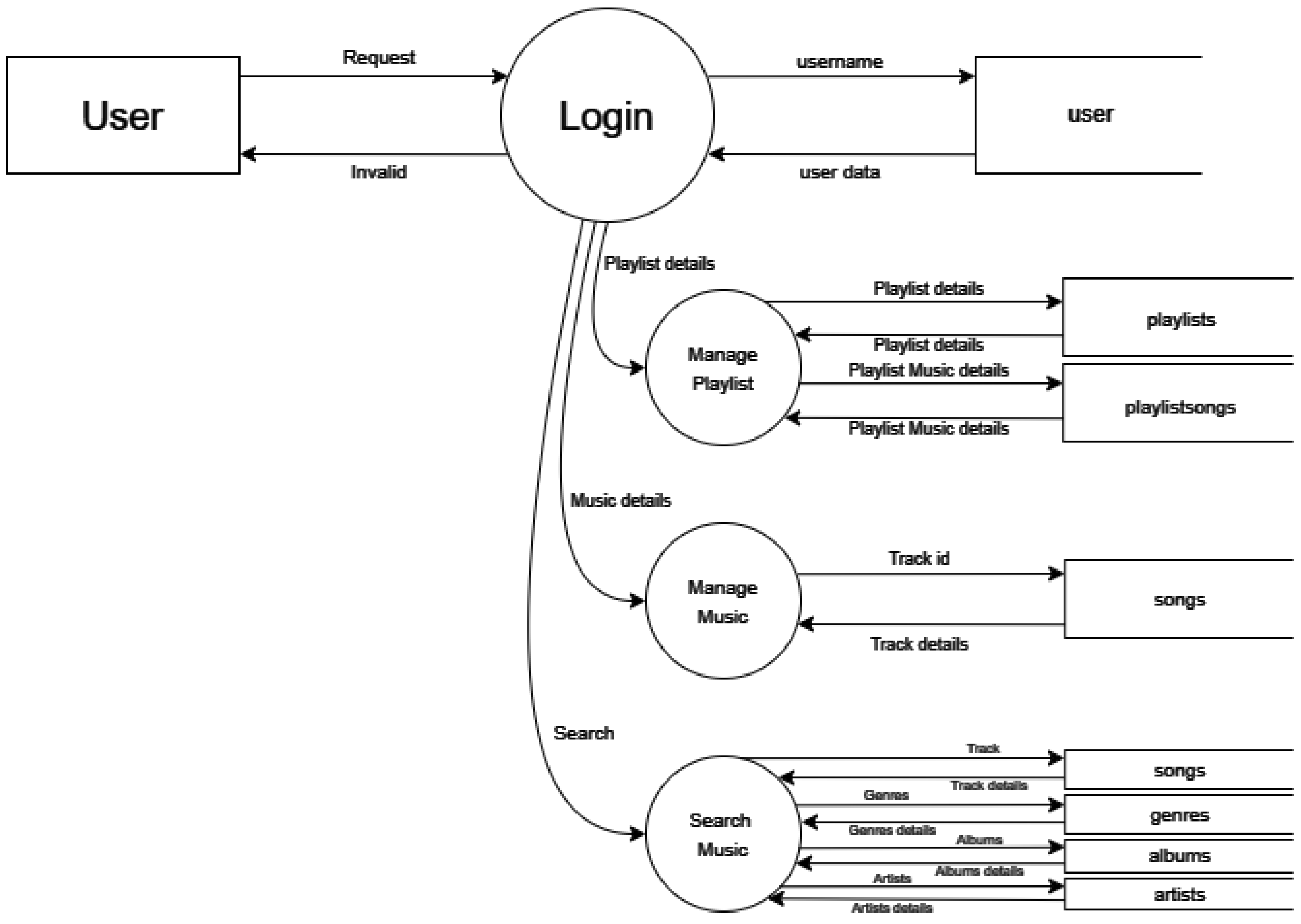
Context Level DFD (Level 0 DFD)



Level 1 DFD (For Admin)



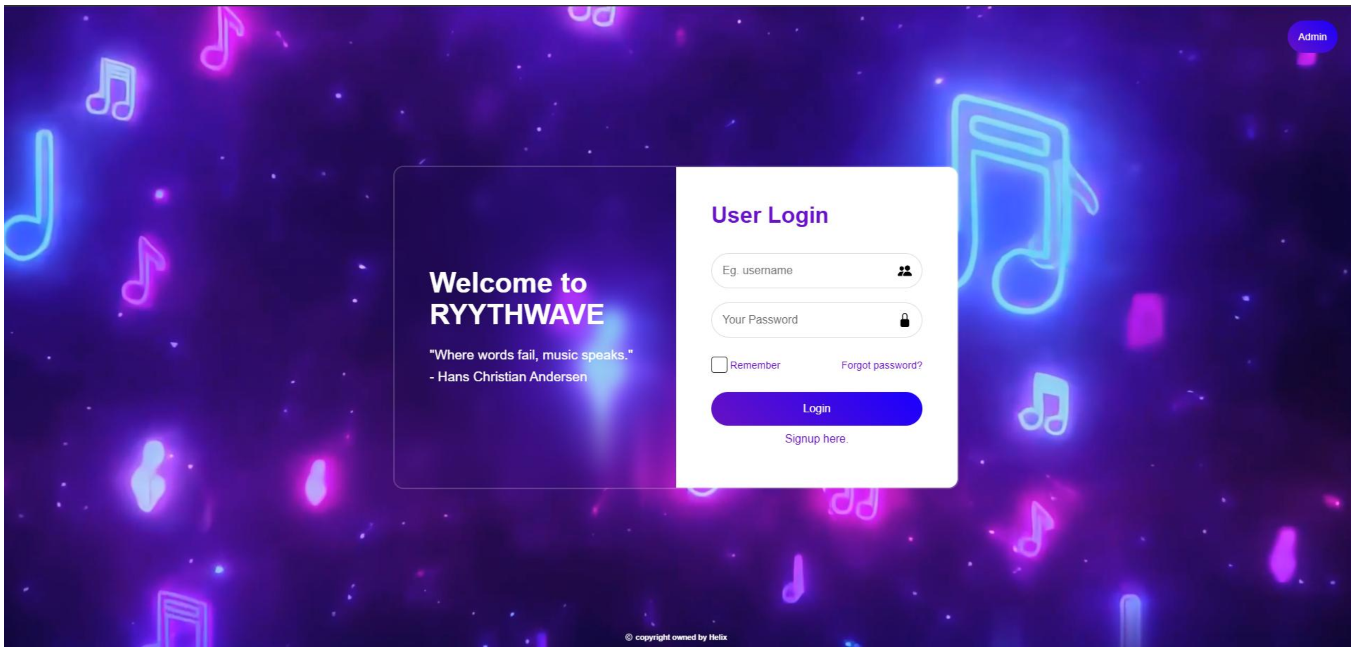
Level 1 DFD (For User)



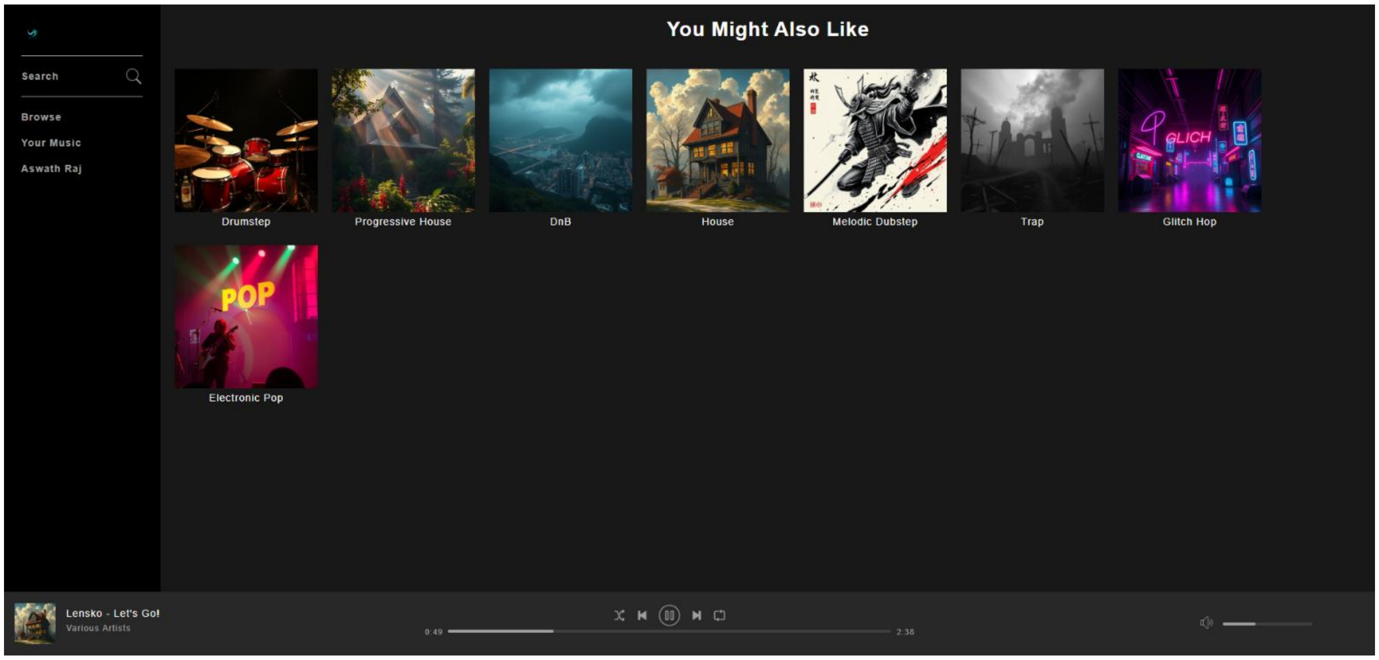
12.3 APPENDIX C

User

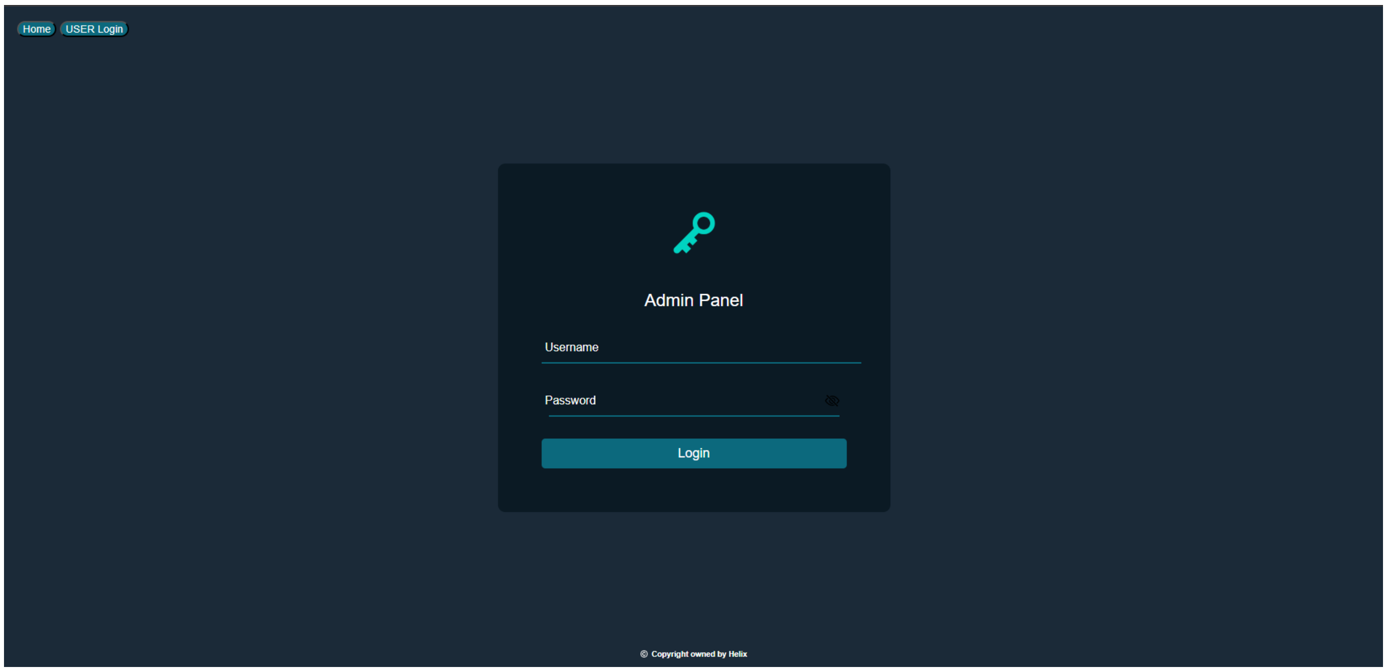
Login page for user



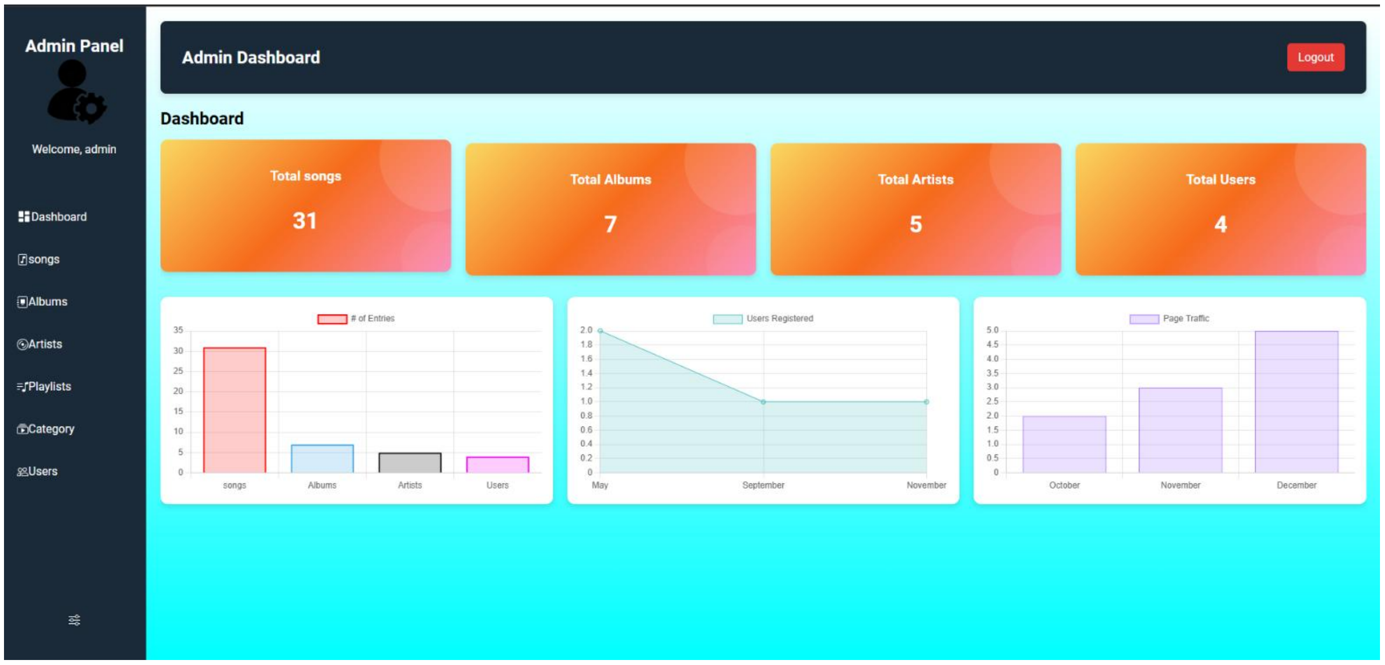
User home page



Login page for Admin



Admin Dashboard



12.4 APPENDIX D (CODING)

Admin Login

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Admin Panel Login</title>
  <link rel="stylesheet" href="public/assets/css/admin_login.css">
</head>
<body>
  <header class="admin-header">
    <div class="admin-header-div">
      <a href="test/home.php"><button type="button">Home</button></a>
      <a href="Login.html"><button type="button">USER Login</button></a>
    </div>
  </header>
  <div class="login-box">
    <div class="key-icon">
      <img src="public/assets/icons/key_icon.png" alt="Key Icon">
    </div>
    <h2>Admin Panel</h2>
    <form action="adminlogin.php" method="post">
      <div class="textbox">
        <input type="text" id="username" name="username" required>
        <label for="username">Username</label>
      </div>
      <div class="textbox password-container">
        <input type="password" id="password" name="password" required>
        <label for="password">Password</label>
        <img src="public/assets/icons/eye-slash.svg" alt="Show Password" class="eye-icon" id="eye-
icon">
```

```
            </div>
            <button type="submit" class="btn">Login</button>

            <!-- Error message displayed below the submit button -->
            <?php if ($error_message): ?>
                <p style="color: red; margin-top: 10px;"><?= htmlspecialchars($error_message) ?></p>
            <?php endif; ?>
        </form>
    </div>
    <div class="copy">
        <img src="public/assets/icons/toppng.com-copyright-symbol-png-white-copyright-logo-in-white-
2000x2000.png" alt="Copyright Symbol">
        <h6>Copyright owned by Helix</h6>
    </div>
    <script>
        document.getElementById('eye-icon').addEventListener('click', function() {
            var passwordField = document.getElementById('password');
            var eyeIcon = document.getElementById('eye-icon');

            if (passwordField.type === 'password') {
                passwordField.type = 'text';
                eyeIcon.src = 'public/assets/icons/eye.svg';
            } else {
                passwordField.type = 'password';
                eyeIcon.src = 'public/assets/icons/eye-slash.svg';
            }
        });
    </script>
</body>
</html>
```

Admin Login Process

```
<?php
session_start();

// Error reporting for debugging
error_reporting(E_ALL);
ini_set('display_errors', 1);
$error_message = ""; // Initialize error message variable
if ($_SERVER["REQUEST_METHOD"] == "POST") {
    // Sanitize input
    $username = isset($_POST['username']) ? trim($_POST['username']) : '';
    $password = isset($_POST['password']) ? trim($_POST['password']) : '';
    // Database connection
    $servername = "localhost";
    $dbusername = "root";
    $dbpassword = "";
    $dbname = "ryythmwave";
    $conn = new mysqli($servername, $dbusername, $dbpassword, $dbname);
    // Check connection
    if ($conn->connect_error) {
        die("Connection failed: " . $conn->connect_error);
    }
    // Prepare and execute query
    $stmt = $conn->prepare("SELECT * FROM admin_table WHERE username = ? AND password = ?");
    $stmt->bind_param("ss", $username, $password);
    $stmt->execute();
    $result = $stmt->get_result();
    if ($result->num_rows == 1) {
        $_SESSION['admin_logged_in'] = true;
        $_SESSION['admin_username'] = $username;
        header("Location: admin_panel.php");
        exit();
    } else {
        $error_message = "Invalid username or password."; // Set error message
    }
    $stmt->close();
    $conn->close();
}
?>
```

Admin Pannel (Dashboard)

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Admin Panel</title>
    <link rel="stylesheet" href="public/assets/css/admin_panel.css">
    <link href="https://fonts.googleapis.com/css2?family=Roboto:wght@300;400;500;700&display=swap"
rel="stylesheet">
    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-
awesome/5.15.4/css/all.min.css">
    <script src="https://cdn.jsdelivr.net/npm/chart.js"></script>
    <script>
        function showSection(sectionId) {
            document.querySelectorAll('.section').forEach(function (section) {
                section.style.display = 'none';
            });
            document.getElementById(sectionId).style.display = 'block';
        }
        window.onload = function() {
            showSection('dashboard');

            // Chart for songs, Albums, Artists, and Users
        const ctx = document.getElementById('myChart').getContext('2d');
        const myChart = new Chart(ctx, {
          type: 'bar',
          data: {
            labels: ['songs', 'Albums', 'Artists', 'Users'],
            datasets: [{
              label: '# of Entries',
              data: [<?php echo $songsCount; ?>, <?php echo $albumsCount; ?>, <?php echo
$artistsCount; ?>, <?php echo $usersCount; ?>],
              backgroundColor: [
                'rgba(255, 0, 0, 0.2)',
                'rgba(54, 162, 235, 0.2)',
                'rgba(0, 0, 0, 0.2)',
                'rgba(255, 0, 255, 0.2)'
              ],
              borderColor: [
                'rgba(255, 0, 0, 1)',
                'rgba(54, 162, 235, 1)',
                'rgba(0, 0, 0, 1)',
                'rgba(255, 0, 255, 1)'
              ],
              borderWidth: 2
            }]
          },
          options: {
            scales: {
              y: {
                beginAtZero: true
              }
            }
          }
        });

        // Chart for Monthly User Registrations
```

```
const userCtx = document.getElementById('userChart').getContext('2d');
const userChart = new Chart(userCtx, {
  type: 'line',
  data: {
    labels: <?php echo json_encode($monthly_user_labels); ?>, // Months from PHP
    datasets: [{
      label: 'Users Registered',
      data: <?php echo json_encode($monthly_user_data); ?>, // User count from PHP
      backgroundColor: 'rgba(75, 192, 192, 0.2)',
      borderColor: 'rgba(75, 192, 192, 1)',
      borderWidth: 1,
      fill: true
    }]
  },
  options: {
    scales: {
      y: {
        beginAtZero: true
      }
    }
  }
});
// Chart for Monthly Traffic
const trafficCtx = document.getElementById('trafficChart').getContext('2d');
const trafficChart = new Chart(trafficCtx, {
  type: 'bar',
  data: {
    labels: <?php echo json_encode($monthly_traffic_labels); ?>, // Months from PHP
    datasets: [{
      label: 'Page Traffic',
      data: <?php echo json_encode($monthly_traffic_data); ?>, // Traffic data from PHP
      backgroundColor: 'rgba(153, 102, 255, 0.2)',
      borderColor: 'rgba(153, 102, 255, 1)',
      borderWidth: 1
```

```
                    }]
                },
                options: {
                    scales: {
                        y: {
                            beginAtZero: true
                        }
                    }
                }
            });
        }
    </script>
</head>
<body>

<!-- Sidebar -->
<div class="sidebar collapsed">
<div class="profile">
        <h2>Admin Panel</h2>
        <img src="<?php echo htmlspecialchars($adminProfilePicturePath); ?>" alt="Admin Profile"
draggable="false" style="width: 100px; height: 100px; border-radius: 50%;">
        <p>Welcome, <?php echo htmlspecialchars($adminUsername); ?></p>
    </div>
    <hr>
    <ul>
        <li><a href="javascript:void(0);" onclick="showSection('dashboard')"><img id="dash"
src="public/assets/icons/dashboard.svg"><span class="text">Dashboard</span></a></li>
        <li><a href="javascript:void(0);" onclick="showSection('songs')"><img
src="public/assets/icons/file-music.svg"><span class="text">songs</span></a></li>
        <li><a href="javascript:void(0);" onclick="showSection('albums')"><img
src="public/assets/icons/journal-album.svg"><span class="text">Albums</span></a></li>
        <li><a href="javascript:void(0);" onclick="showSection('artists')"><img
src="public/assets/icons/disc.svg"><span class="text">Artists</span></a></li>
```

```
        <li><a href="javascript:void(0);" onclick="showSection('playlists')"><img
src="public/assets/icons/music-note-list.svg"><span class="text">Playlists</span></a></li>
        <li><a href="javascript:void(0);" onclick="showSection('category')"><img
src="public/assets/icons/collection-play.svg"><span class="text">Category</span></a></li>
        <li><a href="javascript:void(0);" onclick="showSection('users')"><img
src="public/assets/icons/people.svg"><span class="text">Users</span></a></li>
        <div class="settings">
          <li><a href="javascript:void(0);" onclick="showSection('edit-profile')"><img
src="public/assets/icons/Sliders.svg"></a></li>
        </div>
    </ul>
</div>
<div class="main-content">
    <header>
        <div class="header-title">
          <h1>Admin Dashboard</h1>
        </div>
        <div class="logout">
          <a href="logout.php">Logout</a>
        </div>
    </header>

    <!-- Dashboard Section -->
    <section id="dashboard" class="section active">
        <h2>Dashboard</h2>
        <div class="dashboard-stats">
          <div class="stat">
            <h3>Total songs</h3>
            <p><?php echo $songsCount; ?></p>
          </div>
          <div class="stat">
            <h3>Total Albums</h3>
            <p><?php echo $albumsCount; ?></p>
          </div>
          <div class="stat">
            <h3>Total Artists</h3>
            <p><?php echo $artistsCount; ?></p>
          </div>
          <div class="stat">
            <h3>Total Users</h3>
            <p><?php echo $usersCount; ?></p>
          </div>
        </div>

        <!-- Graph Section -->
        <div class="graph-container">
          <div class="chart-box">
            <canvas id="myChart"></canvas>
          </div>
          <div class="chart-box">
            <canvas id="userChart"></canvas>
          </div>
          <div class="chart-box">
            <canvas id="trafficChart"></canvas>
          </div>
        </div>
      </section>
    </div>
  </body>
</html>
```

Admin Pannel (Add Songs)

```
<?php
session_start();
if (!isset($_SESSION['admin_logged_in'])) {
    header("Location: adminlogin.php");
    exit();
}
```

```
if ($_SERVER['REQUEST_METHOD'] == 'POST') {
    $title = $_POST['title'];
    $album = $_POST['album'];
    $artist = $_POST['artist'];
    $path = $_POST['path'];
    $conn = new mysqli("localhost", "root", "", "ryythmwave");
    if ($conn->connect_error) {
        die("Connection failed: " . $conn->connect_error);
    }
    $sql = "INSERT INTO songs (title, album, artist, genre, path) VALUES ('$title', '$album', '$artist',
'$genre', '$path')";
    if ($conn->query($sql) === TRUE) {
        echo "New track added successfully";
    } else {
        echo "Error: " . $sql . "<br>" . $conn->error;
    }
    $conn->close();
}
// Fetch songs from the database
$conn = new mysqli("localhost", "root", "", "ryythmwave");
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}
$songs = $conn->query("SELECT * FROM songs");
?>

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Add Track</title>
    <link rel="icon" type="image/x-icon" href="public/assets/images/logo.png">
    <link rel="stylesheet" href="public/assets/css/edit_tracks.css">
```

```
</head>
<body>
  <div class="container">
    <div class="header">
      <a href="../../admin_panel.php" class="back-button">Back</a>
      Add New Track
    </div>
    <div class="form-container">
      <h2>Add Track</h2>
      <form action="add_track.php" method="post">
        <div class="form-group">
          <label for="title">Title:</label>
          <input type="text" id="title" name="title" required>
        </div>
        <div class="form-group">
          <label for="album">Album ID:</label>
          <input type="text" id="album" name="album" required>
        </div>
        <div class="form-group">
          <label for="artist">Artist ID:</label>
          <input type="text" id="artist" name="artist" required>
        </div>
        <div class="form-group">
          <label for="artist">Genre :</label>
          <input type="text" id="genre" name="genre" required>
        <div class="form-group">
          <label for="path">Track Path:</label>
          <input type="text" id="path" name="path" required>
        </div>
        <button type="submit" class="submit-btn">Add Track</button>
      </form>
    </div>

    <h2>Existing songs</h2>
```

```
        <div class="overtable">
            <table>
                <thead>
                    <tr>
                        <th>ID</th>
                        <th>Title</th>
                        <th>Album ID</th>
                        <th>Artist ID</th>
                        <th>Genre</th>
                        <th>Path</th>
                    </tr>
                </thead>
                <tbody>
                    <?php while ($track = $songs->fetch_assoc()) { ?>
                    <tr>
                        <td><?php echo $track['id']; ?></td>
                        <td><?php echo $track['title']; ?></td>
                        <td><?php echo $track['album']; ?></td>
                        <td><?php echo $track['artist']; ?></td>
                        <td><?php echo $track['genre']; ?></td>
                        <td><?php echo $track['path']; ?></td>
                    </tr>
                    <?php } ?>
                </tbody>
            </table>
        </div>
    </div>
</body>
</html>

<?php
// Close the database connection
$conn->close();
?>
```

User Home Page

```
<?php
include("includes/includedFiles.php");
?>
<h1 class="pageHeadingBig">You Might Also Like</h1>
<div class="gridViewContainer">
  <?php
    $albumQuery = mysqli_query($con, "SELECT * FROM albums ORDER BY RAND() LIMIT 10");
    while($row = mysqli_fetch_array($albumQuery)) {
      echo "<div class='gridViewItem'>
          <span role='link' tabindex='0' onclick='openPage(\"album.php?id=" . $row['id'] . "\")'>
            <img src='" . $row['artworkPath'] . "'>
            <div class='gridViewInfo'>"
              . $row['title'] .
            "</div>
          </span>
        </div>";
    }
  ?>
</div>
```

User Albums

```
<?php include("includes/includedFiles.php");
if(isset($_GET['id'])) {
  $albumId = $_GET['id'];
}
else {
  header("Location: index.php");
}
$album = new Album($con, $albumId);
```

```
$artist = $album->getArtist();
?>
<div class="entityInfo">
    <div class="leftSection">
        <img src="<?php echo $album->getArtworkPath(); ?>">
    </div>
    <div class="rightSection">
        <h2><?php echo $album->getTitle(); ?></h2>
        <p>By <?php echo $artist->getName(); ?></p>
        <p><?php echo $album->getNumberOfSongs(); ?> songs</p>
    </div>
</div>
<div class="tracklistContainer">
    <ul class="tracklist">
        <?php
        $songIdArray = $album->getSongIds();
        $i = 1;
        foreach($songIdArray as $songId) {
            $albumSong = new Song($con, $songId);
            $albumArtist = $albumSong->getArtist();
            echo "<li class='tracklistRow'>
                    <div class='trackCount'>
                        <img class='play' src='assets/images/icons/play-white.png' onclick='setTrack(\"" .
$albumSong->getId() . "\", tempPlaylist, true)'>
                        <span class='trackNumber'>$i</span>
                    </div>
                    <div class='trackInfo'>
                        <span class='trackName'>" . $albumSong->getTitle() . "</span>
                        <span class='artistName'>" . $albumArtist->getName() . "</span>
                    </div>
                    <div class='trackOptions'>
                        <input type='hidden' class='songId' value='". $albumSong->getId() . "'>
                        <img class='optionsButton' src='assets/images/icons/more.png'
onclick='showOptionsMenu(this)'>
```

```
                </div>
                <div class='trackDuration'>
                    <span class='duration'>" . $albumSong->getDuration() . "</span>
                </div>
            </li>";
        $i = $i + 1;
    }
    ?>
    <script>
        var tempSongIds = '<?php echo json_encode($songIdArray); ?>';
        tempPlaylist = JSON.parse(tempSongIds);
        console.log(tempPlaylist);
    </script>
  </ul>
</div>
<nav class="optionsMenu">

  <input type="hidden" class="songId">
  <?php echo Playlist::getPlaylistDropdown($con, $userLoggedIn->getUsername()); ?>
</nav>
```

User Playlist

```
<?php
include("includes/includedFiles.php");
?>
<div class="playlistContainer">
  <div class="gridViewContainer">
    <h2>PLAYLISTS</h2>
    <div class="buttonItems">
      <button class="button green" onclick="createPlaylist()">NEW PLAYLIST</button>
    </div>
    <?php
```

```
            $username = $userLoggedIn->getUsername();
            $playlistQuery = mysqli_query($con, "SELECT * FROM playlists WHERE owner =
'$username'");
            if(mysqli_num_rows($playlistQuery) == 0) {
                echo "<span class='noResults'>You don't have any playlist yet.</span>";
            }
            while($row = mysqli_fetch_array($playlistQuery)) {
                $playlist = new Playlist($con, $row);
                echo "<div class='gridViewItem' role='link' tabindex='0' onclick='openPage(\"playlist.php?id="
. $playlist->getId() ."\")'>
                        <div class='playlistImage'>
                            <img src='assets/images/icons/playlist.png'/>
                        </div>
                        <div class='gridViewInfo'>"
                                . $playlist->getName() .
                            "</div>
                        </div>";
            }
        ?>
    </div>
</div>
```

User Music Player Bar

```
<?php
$songQuery = mysqli_query($con, "SELECT id FROM songs ORDER BY RAND() LIMIT 10");

$resultArray = array();
while($row = mysqli_fetch_array($songQuery)) {
    array_push($resultArray, $row['id']);
}
$jsonArray = json_encode($resultArray);
?>
```

```
<script>
$(document).ready(function() {
    var newPlaylist = <?php echo $jsonArray; ?>;
    audioElement = new Audio();
    setTrack(newPlaylist[0], newPlaylist, false);
    updateVolumeProgressBar(audioElement.audio);


    $("#nowPlayingBarContainer").on("mousedown touchstart mousemove touchmove", function(e) {
        e.preventDefault();
    });
    $(".playbackBar .progressBar").mousedown(function() {
        mouseDown = true;
    });
    $(".playbackBar .progressBar").mousemove(function(e) {
        if(mouseDown == true) {
            //set time of song depending on the position of mouse
            timeFromOffset(e, this);
        }
    });
    $(".playbackBar .progressBar").mouseup(function(e){
            timeFromOffset(e, this);
    });
    $(".volumeBar .progressBar").mousedown(function() {
        mouseDown = true;
    });
    $(".volumeBar .progressBar").mousemove(function(e) {
        if(mouseDown == true) {
            var percentage = e.offsetX / $(this).width();
            if(percentage >= 0 && percentage <= 1) {
            audioElement.audio.volume = percentage;
            }
        }
    });
    $(".volumeBar .progressBar").mouseup(function(e){
        var percentage = e.offsetX / $(this).width();
        audioElement.audio.volume = percentage;
    });
    $(document).mouseup(function() {
        mouseDown = false;
    });
});
function timeFromOffset(mouse, progressBar) {
    var percentage = mouse.offsetX / $(progressBar).width() * 100;
    var seconds = audioElement.audio.duration * (percentage / 100);
    audioElement.setTime(seconds);
}
function prevSong() {
    if(audioElement.audio.currentTime >= 3 || currentIndex == 0 ) {
        audioElement.setTime(0);
    }
    else {
        currentIndex = currentIndex - 1;
        setTrack(currentPlaylist[currentIndex], currentPlaylist, true);
    }
}
function nextSong() {
    if(repeat == true) {
        audioElement.setTime(0);
        playSong();
        return;
    }
    if(currentIndex == currentPlaylist.length - 1) {
        currentIndex = 0;
    }
    else {
        currentIndex++;
    }
    var trackToPlay = shuffle ? shufflePlaylist[currentIndex] : currentPlaylist[currentIndex];
    setTrack(trackToPlay, currentPlaylist, true);
}
function setRepeat() {
    repeat = !repeat;
    var imageName = repeat ? "repeat-active.png" : "repeat.png";
    $(".controlButton.repeat img").attr("src", "assets/images/icons/" + imageName);
}
function setMute() {
    audioElement.audio.muted =! audioElement.audio.muted;
    var imageName = audioElement.audio.muted ? "volume-mute.png" : "volume.png";
    $(".controlButton.volume img").attr("src", "assets/images/icons/" + imageName);
}
function setShuffle() {
    shuffle = !shuffle
    var imageName = shuffle ? "shuffle-active.png" : "shuffle.png";
    $(".controlButton.shuffle img").attr("src", "assets/images/icons/" + imageName);

    if(shuffle==true) {
        // randomize playlist
        shuffleArray(shufflePlaylist);
        currentIndex = shufflePlaylist.indexOf(audioElement.currentlyPlaying.id);
    }
    else {
        // shuffle has been deactivated
        // go back to regular playlist
        currentIndex = currentPlaylist.indexOf(audioElement.currentlyPlaying.id);
    }
}
// Shuffle function/Algorithm taken from stack overflow
function shuffleArray(a) {
    var j, x, i;
    for(i = a.length; i; i--) {
        j = Math.floor(Math.random() * i);
        x = a[i - 1];
        a[i - 1] = a[j];
        a[j] = x;
    }
}
function setTrack(trackId, newPlaylist, play) {
        if(newPlaylist != currentPlaylist) {
            currentPlaylist = newPlaylist;
            shufflePlaylist = currentPlaylist.slice();
            shuffleArray(shufflePlaylist)
        }
        if(shuffle == true) {
            currentIndex = shufflePlaylist.indexOf(trackId);
        }
        else {
        currentIndex = currentPlaylist.indexOf(trackId);
        }
        pauseSong();
    $.post("includes/handlers/ajax/getSongJson.php", { songId: trackId }, function(data) {
        var track = JSON.parse(data);
        $(".trackName span").text(track.title);

        $.post("includes/handlers/ajax/getArtistJson.php", { artistId: track.artist }, function(data) {
            var artist = JSON.parse(data);
            $(".trackInfo .artistName span").text(artist.name);
            $(".trackInfo .artistName span").attr("onclick", "openPage('artist.php?id=" + artist.id + "')");
        });
        $.post("includes/handlers/ajax/getAlbumJson.php", { albumId: track.album }, function(data) {
            var album = JSON.parse(data);
            $(".content .albumLink img").attr("src", album.artworkPath);
            $(".content .albumLink img").attr("onclick", "openPage('album.php?id=" + album.id + "')");
            $(".trackInfo .trackName span").attr("onclick", "openPage('album.php?id=" + album.id + "')");
        });
        audioElement.setTrack(track);
        if(play == true) {
            playSong();
        }
    });
}
function playSong() {

    if(audioElement.audio.currentTime == 0) {
        $.post("includes/handlers/ajax/updatePlays.php", { songId: audioElement.currentlyPlaying.id });
    }
    $(".controlButton.play").hide();
    $(".controlButton.pause").show();
    audioElement.play();
}
function pauseSong() {
    $(".controlButton.play").show();
    $(".controlButton.pause").hide();
    audioElement.pause();
}
</script>
<div id="nowPlayingBarContainer">
    <div id="nowPlayingBar">
        <div id="nowPlayingLeft">
            <div class="content">
                <span class="albumLink">
                    <img role="link" tabindex="0" src="" class="albumArtwork">
                </span>
                <div class="trackInfo">
                    <span class="trackName">
                        <span role="link" tabindex="0"></span>
                    </span>
                    <span class="artistName">
                        <span role="link" tabindex="0"></span>
                </span>
              </div>
            </div>
          </div>
          <div id="nowPlayingCenter">
            <div class="content playerControls">
              <div class="buttons">
                <button class="controlButton shuffle" title="Shuffle button" onclick="setShuffle()">
                  <img src="assets/images/icons/shuffle.png" alt="Shuffle">
                </button>
                <button class="controlButton previous" title="Previous button" onclick="prevSong()">
                  <img src="assets/images/icons/previous.png" alt="Previous">
                </button>
                <button class="controlButton play" title="Play button" onclick="playSong()">
                  <img src="assets/images/icons/play.png" alt="Play">
                </button>
                <button class="controlButton pause" title="Pause button" style="display: none;"
onclick="pauseSong()">
                  <img src="assets/images/icons/pause.png" alt="Pause">
                </button>
                <button class="controlButton next" title="Next button" onclick="nextSong()">
                  <img src="assets/images/icons/next.png" alt="Next">
                </button>
                <button class="controlButton repeat" title="Repeat button" onclick="setRepeat()">
                  <img src="assets/images/icons/repeat.png" alt="Repeat">
                </button>
              </div>
              <div class="playbackBar">
                <span class="progressTime current">0.00</span>
                <div class="progressBar">
                  <div class="progressBarBg">
                    <div class="progress"></div>
                  </div>
                </div>
                <span class="progressTime remaining">0.00</span>
            </div>
        </div>
    </div>
    <div id="nowPlayingRight">
        <div class="volumeBar">
            <button class="controlButton volume" title="Volume button" onclick="setMute()">
                <img src="assets/images/icons/volume.png" alt="Volume">
            </button>
            <div class="progressBar">
                <div class="progressBarBg">
                    <div class="progress"></div>
                </div>
            </div>
        </div>
    </div>
  </div>
</div>
```

User Search

```
<?php
include("includes/includedFiles.php");

if(isset($_GET['term'])) {
  $term = urldecode($_GET['term']);
}
else {
  $term = "";
}
?>
<div class="searchContainer">
  <h4>Search for an artist, album or song</h4>
```

```
    <input type="text" class="searchInput" value="<?php echo $term; ?>" placeholder="Start typing..."
onfocus="this.value = this.value">
</div>
<script>
$(".searchInput").focus();
$(function() {
    $(".searchInput").keyup(function() {
        clearTimeout(timer);
        timer = setTimeout(function() {
            var val = $(".searchInput").val();
            openPage("search.php?term=" + val);
        }, 2000);
    })
})
</script>
<?php if($term == "") exit(); ?>
<div class="tracklistContainer borderBottom">
    <h2>SONGS</h2>
    <ul class="tracklist">
        <?php
        $songsQuery = mysqli_query($con, "SELECT id FROM songs WHERE title LIKE '$term%' LIMIT
10");
        if(mysqli_num_rows($songsQuery) == 0) {
            echo "<span class='noResults'>No songs found matching " . $term . "</span>";
        }
        $songIdArray = array();
        $i = 1;
        while($row = mysqli_fetch_array($songsQuery)) {
            if($i > 15) {
                break;
            }
            array_push($songIdArray, $row['id']);
            $albumSong = new Song($con, $row['id']);
            $albumArtist = $albumSong->getArtist();
            echo "<li class='tracklistRow'>
                    <div class='trackCount'>
                        <img class='play' src='assets/images/icons/play-white.png' onclick='setTrack(\"" .
$albumSong->getId() . "\", tempPlaylist, true)'>
                        <span class='trackNumber'>$i</span>
                    </div>
                    <div class='trackInfo'>
                        <span class='trackName'>" . $albumSong->getTitle() . "</span>
                        <span class='artistName'>" . $albumArtist->getName() . "</span>
                    </div>
                    <div class='trackOptions'>
                        <input type='hidden' class='songId' value='". $albumSong->getId() . "'>
                        <img class='optionsButton' src='assets/images/icons/more.png'
onclick='showOptionsMenu(this)'>
                    </div>
                    <div class='trackDuration'>
                        <span class='duration'>" . $albumSong->getDuration() . "</span>
                    </div>
                </li>";
            $i = $i + 1;
        }
        ?>
        <script>
            var tempSongIds = '<?php echo json_encode($songIdArray); ?>';
            tempPlaylist = JSON.parse(tempSongIds);
            console.log(tempPlaylist);
        </script>
    </ul>
</div>
<div class="artistsContainer borderBottom">
    <h2>ARTISTS</h2>
    <?php
    $artistsQuery = mysqli_query($con, "SELECT id FROM artists WHERE name LIKE '$term%' LIMIT
10");
    if(mysqli_num_rows($artistsQuery) == 0) {
            echo "<span class='noResults'>No artists found matching " . $term . "</span>";
        }
    while($row = mysqli_fetch_array($artistsQuery)) {
        $artistFound = new Artist($con, $row['id']);
        echo "<div class='searchResultRow'>
                <div class="artistName>
                    <span role='link' tabindex='0' onclick='openPage(\"artist.php?id=". $artistFound->getId() .
"\")'>
                    "
                    . $artistFound->getName() .
                    "
                    </span>
                </div>
            </div>";
    }
    ?>
</div>
<div class="gridViewContainer">
    <h2>ALBUMS</h2>
    <?php
        $albumQuery = mysqli_query($con, "SELECT * FROM albums WHERE title LIKE '$term%'
LIMIT 10");
        if(mysqli_num_rows($albumQuery) == 0) {
            echo "<span class='noResults'>No albums found matching " . $term . "</span>";
        }
        while($row = mysqli_fetch_array($albumQuery)) {
            echo "<div class='gridViewItem'>
                    <span role='link' tabindex='0' onclick='openPage(\"album.php?id=" . $row['id'] . "\")'">
                        <img src='" . $row['artworkPath'] . "'>
                        <div class='gridViewInfo'>"
                            . $row['title'] .
                    "</div>
                </span>
            </div>";
        }
    ?>
</div>
<nav class="optionsMenu">
    <input type="hidden" class="songId">
    <?php echo Playlist::getPlaylistDropdown($con, $userLoggedIn->getUsername()); ?>
</nav>
```

13.BIBILIOGRAPHY

Bootstrap Documentation: For styling components used in the UI. (https://getbootstrap.com/docs/)

PHP Manual: For PHP functions and syntax references. (https://www.php.net/manual/en/)

Stack Overflow: For resolving common coding issues and debugging tips. (https://stackoverflow.com/)

GeeksforGeeks: For learning programming concepts, web development tutorials, and problem-solving techniques. (https://www.geeksforgeeks.org/)

OpenAI ChatGPT (Rox): Provided support in solving doubts, debugging issues, and enhancing features during the development of the project. (https://openai.com/chatgpt)