M.S. RAMAIAH INSTITUTE OF TECHNOLOGY

MSR NAGAR, BENGALURU,560054

A Mini Project Report on

MUSIFY – A MUSIC SYSTEM

Submitted in partial fulfilment of the OTHER COMPONENT requirements as a part of the DBMS Lab for the IV Semester of degree of ***Bachelor of Engineering in Information Science and Engineering***

Submitted by

SUHAS R
1MS21IS106

VARSHITH R
1MS21IS124

VIDWAN GOWDA HM
1MS21IS126

Under the Guidance of

Faculty Incharge
Mrs. Kusuma S
Assistant Professor
Dept. of ISE

Department of Information Science and Engineering
Ramaiah Institute of Technology
2022 – 2023

Ramaiah Institute of Technology

MSR Nagar, Bengaluru-560054

DEPARTMENT OF INFORMATION SCIENCE & ENGINEERING

CERTIFICATE

This is to certify that the Mini project report entitled ***MUSIFY: A MUSIC SYSTEM*** has been successfully completed by **SUHAS R, VARSHITH R, VIDWAN GOWDA HM** bearing USN **1MS21IS106, 1MS21IS124, 1MS21IS126** respectively, presently IV semester students of **Ramaiah Institute of Technology** in partial fulfilment of the OTHER COMPONENT requirements of the DBMS Laboratory during academic year 2022 – 2023. The mini project report has been approved as it satisfies the academic requirements.

Mrs. Kusuma S

Faculty Incharge

ABSTRACT

“Musify” is a web-based music app that allows users to explore and enjoy songs, albums, and artists. It is built using the XAMP backend, which consists of PHP files that handle the user interface and the database operations. The app uses a MySQL database to store user and music-related data in different tables.

Users have to register and log in to access the app’s features, which include a list of trending songs on the index page and a search function that can find songs by name, artists, or albums. The app queries the database using SQL statements and retrieves the relevant data from the “track,” “album,” and “artist” tables.

The app also handles audio files and cover images efficiently by storing their file paths in the database instead of the actual files. This way, the app can display the songs and album covers on the front-end without loading too much data.

“Musify” is a comprehensive music platform that demonstrates the use of robust database management techniques and best practices. It provides a user-friendly and secure interface for users to discover and listen to their favorite songs and artists.

TABLE OF CONTENTS

LIST OF FIGURES

ABBREVIATIONS

CSS	-	Cascading style sheets
DBMS	-	Database Management System
ER	-	Entity Relationship
HTML	-	Hypertext Markup Language
HTTP	-	Hypertext Transfer Protocol
ID	-	Identification
JS	-	JavaScript
PHP	-	PHP Hypertext Preprocessor
SQL	-	Structured Query Language
XAMPP	-	X-operating system, Apache, Mysql, Php, Perl

Chapter 1

INTRODUCTION

1.1 Background

A **database** is an organized collection of data, generally stored and accessed electronically from a computer system. Where databases are more complex they are often developed using formal design and modeling techniques.

The database management system (DBMS) is the software that interacts with end users, applications, the database itself to capture and analyze the data and provides facilities to administer the database. The sum total of the database, the DBMS and the associated applications can be referred to as a "database system". Often the term "database" is also used to loosely refer to any of the DBMS, the database system or an application associated with the database. The DBMS manages three important things: the data, the database engine that allows data to be accessed, locked and modified and the database schema, which defines the database's logical structure. These three foundational elements help provide concurrency, security, data integrity and uniform administration procedures. Typical database administration tasks supported by the DBMS include change management, performance monitoring/tuning and backup and recovery. Many database management systems are also responsible for automated rollbacks, restarts and recovery as well as the logging and auditing of activity.

1.2 Introduction to Music Management System "MUSIFY"

Welcome to Musify, your ultimate destination for music lovers! Musify is a cutting-edge web-based music application that aims to revolutionize the way you discover and enjoy music. Developed as a part of a DBMS mini-project, Musify offers a seamless and immersive musical experience that caters to every user's unique preferences.

At Musify, we believe that music has the power to uplift, inspire, and connect people from all walks of life. Our platform is designed with a user-centric approach, providing a sleek and intuitive interface that makes exploring and interacting with music a breeze.

Key Features:

1. Trending Songs: On the homepage, you'll find a handpicked list of the hottest and most popular tracks. Stay up-to-date with the latest music trends and discover new favorites in just a few clicks.

2. Personalized User Accounts: To unlock the full potential of Musify, simply create a user account. By registering, you gain access to a range of personalized features and recommendations tailored to your musical tastes.

3. Dynamic Search: Our powerful search functionality empowers you to find your favorite songs, artists, and albums effortlessly. Search by song name, artist name, or album title to find precisely what you're looking for.

4. Detailed Artist and Album Pages: Delve into the world of your favorite artists and explore their discography on dedicated artist and album pages. Learn about their background, genre, and artistic journey.

5. Secure and Reliable: Musify ensures the utmost security for your data and privacy. We utilize the latest encryption protocols and robust authentication mechanisms to safeguard your information.

6. Seamless Audio Playback: Experience uninterrupted playback of your favorite tracks. With Musify, you can enjoy your music without any disruptions.

Chapter 2

E R DIAGRAM AND RELATIONAL SCHEMA DIAGRAM

2.1 Description of ER Diagram

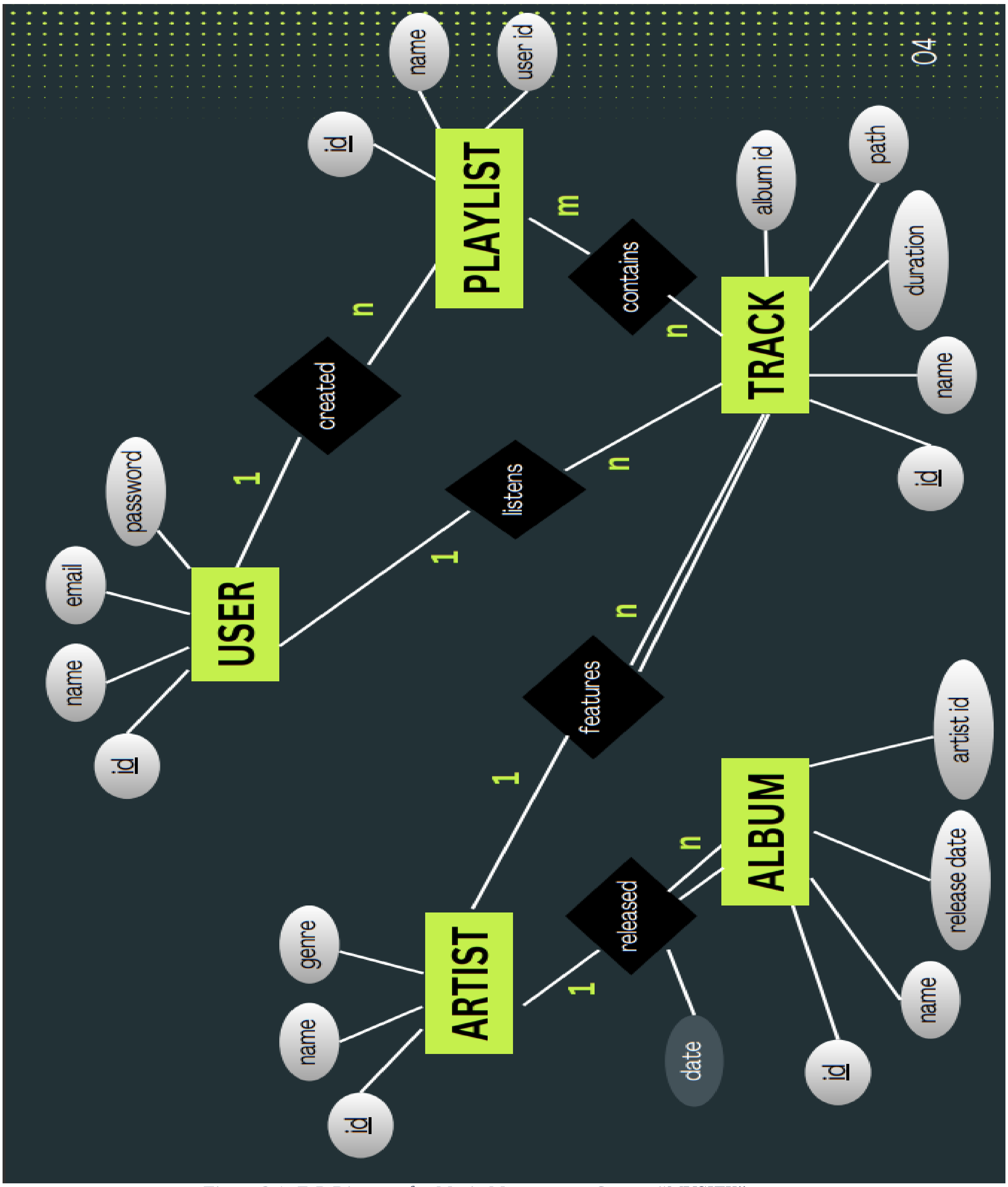


Figure 2.1: E-R Diagram for Music Management System "MUSIFY"

Entity relationship diagram displays the relationships of entity set stored in a database. In other words, we can say that ER diagrams help you to explain the logical structure of databases. At first look, an ER diagram looks very similar to the flowchart. However, ER Diagram includes many specialized symbols, and its meanings make this model unique.

2.1.1 Components of MUSIFY(MUSIC Management System), E-R Diagram

Entity types like **USERS,ALBUM,TRACK,PLAYLIST and Artist** are in rectangular boxes.

1. Relationships like **CONTAINS,RELEASED** and **CREATED** are in diamond boxes, attached to entity types with straight lines.
2. Attributes are shown in ovals like **Name,Email** and **Genre**, each attached by a straight line to entity or relationship type.
3. Primary Key attributes like **Id** are underlined.
4. The foreign key **albid** in the Tracks table references the album to which the track belongs, using the album_id from the Albums table.
5. The foreign key **artid** in the Albums table references the artist who created the album, using the artist_id from the Artists table.

2.1.2 E-R Diagram Relationships Description

- **USER:PLAYLIST** - The “USER” table has a one-to-many relationship with the “PLAYLIST” table. This means that one user can have many playlists, but each playlist belongs to only one user. The foreign key for this relationship is “user id” in the “PLAYLIST” table, which references the “id” field in the “USER” table.
- **ARTIST:ALBUM** - The “ARTIST” table has a one-to-many relationship with the “ALBUM” table. This means that one artist can have many albums, but each album belongs to only one artist. The foreign key for this relationship is “artist id” in the “ALBUM” table, which references the “id” field in the “ARTIST” table.
- **ALBUM:TRACK** - The “ALBUM” table has a one-to-many relationship with the “TRACK” table. This means that one album can have many tracks, but each track belongs to only one album. The foreign key for this relationship is “album id” in the “TRACK” table, which references the “id” field in the “ALBUM” table.
- **PLAYLIST:TRACK** - The “PLAYLIST” table has a many-to-many relationship with the “TRACK” table. This means that one playlist can have many tracks, and one track can belong to many playlists. The foreign key for this relationship is not stored in either table, but in a separate junction table called “PLAYLIST_TRACK”. This table has two fields: “playlist id” and “track id”, which reference the “id” fields in the respective tables.

2.2 Description of Relational Schema Diagram

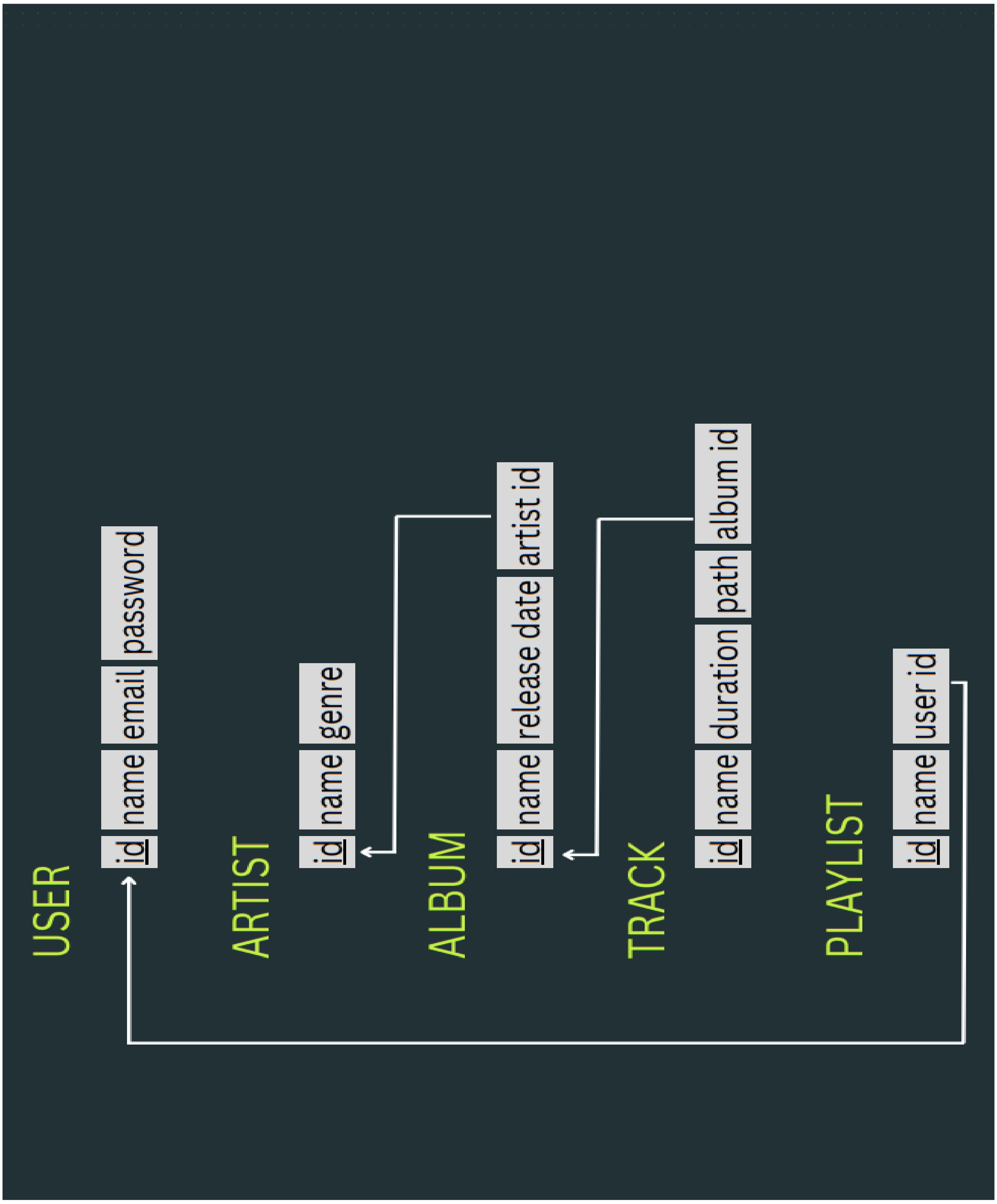


Figure 2.2 Relational Schema Diagram for Music Management System "MUSIFY"

2.2.1 General Constraints

1. **NULL Constraint**: Attributes that are under NOT NULL constraints have to be filled compulsorily. Almost all the attributes in the project are under NOT NULL constraint.
2. **Entity Integrity Constraint**: This constraint makes sure that no primary key can have a NULL value assigned to it. The primary keys involved in the project include:
    - Users.id
    - Users.email
    - Track.id
    - Playlist.id
    - Album.id
    - Artist.id
3. **Referential Integrity Constraints**: A table in the back end of the project may have references pointing to an attribute in another table. The various tables are also linked with multiple foreign keys which are all set to cascade any update or delete operation on the attribute in the main table. The various Foreign Key attributes are:
    - user id in the PLAYLIST table, which references the id field in the USER table
    - artist id in the ALBUM table, which references the id field in the ARTIST table
    - path in the TRACK table, which references the name field in the ALBUM table

2.2.2 Schema Description

The above Figure.2.2 shows the relational schema of Music Management System "MUSIFY". It has the following entities.

1. **USER:** This table contains details of user like user_id, password, username,email.
2. **ARTIST:** This table contains the details of the artist like artist_id, artist_name, genre
3. **ALBUM:** This table contains the details of the album like album_id, album_name,release_date,artist_id(Id of artist to whom this song belongs)
4. **TRACK:** This table contains the details of the track like track_id, track_name, duration, path, album_id(id of album to which this track belongs)
5. **PLAYLIST:** This table contains the details of the playlist like playlist_id, playlist_name, user_id(id of the user to whom playlist belongs)

Chapter 3

SYSTEM DESIGN

3.1 Table Description

USER

FIELD	TYPE	NULL?	KEY	DEFAULT
Id	INT(11)	NO	PRIMARY	None
Password	VARCHAR(15)	NO		None
Username	VARHAR(15)	NO		None
Email	VARCHAR(15)	NO		None

ARTIST

FIELD	TYPE	NULL?	KEY	DEFAULT
Id	INT(11)	NO	PRIMARY	None
Name	VARCHAR(15)	YES		NULL
GENRE	VARCHAR(10)	YES		NULL

ALBUM

FIELD	TYPE	NULL?	KEY	DEFAULT
Id	INT(11)	NO	PRIMARY	None
Release_date	DATE	NO		NULL
Name	VARCHAR(20)	NO		NULL
Artist_id	INT(11)	NO	FOREIGN	NULL

TRACK

FIELD	TYPE	NULL?	KEY	DEFAULT
Id	INT(11)	NO	PRIMARY	None
Name	VARCHAR(20)	YES		NULL
Duration	TIME	YES		NULL
Path	VARCHAR(20)	YES		NULL
Album_id	INT(11)	YES	FOREIGN	NULL

PLAYLIST

FIELD	TYPE	NULL?	KEY	DEFAULT
Id	INT(11)	NO	PRIMARY	None
Name	VARCHAR(20)	NO		None
User_id	INT(11)	NO	FOREIGN	None

Chapter 4

IMPLEMENTATION

4.1 Front-end Development

The front-end is built using a combination of technologies such as Hypertext Markup Language (HTML), JavaScript and Cascading Style Sheets (CSS). Front-end developers design and construct the user experience elements on the web page or app including buttons, menus, pages, links, graphics and more.

4.1.1 Hypertext Markup Language

HTML is a computer language devised to allow website creation. These websites can then be viewed by anyone else connected to the Internet. It is relatively easy to learn, with the basics being accessible to most people in one sitting; and quite powerful in what it allows you to create. HTML is the standard markup language for creating Web pages. It stands for Hyper Text Markup Language. It describes the structure of a Web page. It consists of a series of elements. It elements tell the browser how to display the content. It elements are represented by tags. HTML tags label pieces of content such as "heading", "paragraph", "table", and so on. Browsers do not display the HTML tags, but use them to render the content of the page.

4.1.2 Cascading style sheets

CSS stands for Cascading Style Sheets. It is a style sheet language which is used to describe the look and formatting of a document written in markup language. It provides an additional feature to HTML. It is generally used with HTML to change the style of web pages and user interfaces. CSS is used along with HTML and JavaScript in most websites to create user interfaces for web applications. Before CSS, tags like font, color, background style, element alignments, border and size had to be repeated on every web page. This was a very long process. CSS solved that issue. CSS style definitions are saved in external CSS files so it is possible to change the entire website by changing just one file. CSS provides more detailed attributes than plain HTML to define the look and feel of the website.

4.1.3 JavaScript

JavaScript is a dynamic computer programming language. It is lightweight and most commonly used as a part of web pages, whose implementations allow client-side script to interact with the user and make dynamic pages. It is an interpreted programming language with object-oriented capabilities. Client-side JavaScript is the most common form of the language. The script should be included in or referenced by an HTML document for the code to be interpreted by the browser. It means that a web page need not be a static HTML, but can include programs that interact with the user, control the browser, and dynamically create HTML content. The JavaScript client-side mechanism provides many advantages over traditional CGI server-side scripts. The JavaScript code is executed when the user submits the form, and only if all the entries are valid, they would be submitted to the Web Server. JavaScript can be used to trap user-initiated events such as button clicks, link navigation, and other actions that the user initiates explicitly or implicitly. Advantages are: **Less server interaction, immediate feedback to the visitors, increased interactivity** and **richer interfaces.**

4.2 Back-end Development

Backend is server side of the website. It stores and arranges data, and also makes sure everything on the client-side of the website works fine. It is the part of the website that you cannot see and interact with. It is the portion of software that does not come in direct contact with the users. The parts and characteristics developed by backend designers are indirectly accessed by users through a front-end application. Activities, like writing APIs, creating libraries, and working with system components without user interfaces or even systems of scientific programming, are also included in the backend.

4.2.1 PHP

PHP, which stands for "Hypertext Preprocessor," is a widely used server-side scripting language designed for web development. It was originally created by Rasmus Lerdorf in 1994 and has since evolved into a powerful and versatile programming language. PHP is embedded within HTML code and executed on the server, generating dynamic content that is then sent to the user's web browser.

One of PHP's key strengths is its ability to interact with databases, making it an excellent choice for building dynamic web applications and content management systems (CMS). It supports various database systems such as MySQL, PostgreSQL, SQLite, and more, allowing developers to create robust and data-driven websites.
PHP is an open-source language, which means its source code is freely available, and a vast community of developers continuously contribute to its improvement and maintenance. This

active community has led to a rich ecosystem of libraries, frameworks, and tools that make PHP development more efficient and streamlined.

Additionally, PHP is relatively easy to learn and use, making it accessible to both beginners and experienced developers. Its syntax is similar to C and other programming languages, making it familiar to many programmers.

As the backbone of numerous popular websites, PHP has played a significant role in shaping the modern web landscape. Websites like Facebook, WordPress, Wikipedia, and many others rely on PHP to deliver dynamic and interactive user experiences.

Whether you are creating a simple personal website or a complex web application, PHP offers the flexibility and scalability needed to bring your web projects to life. Its vast community support, extensive documentation, and frequent updates ensure that PHP remains a relevant and powerful tool for web development.

4.2.2 Web Server – Apache

Apache is an open-source and free web server software that powers around 46% of websites around the world. The official name is Apache HTTP Server, and it's maintained and developed by the Apache Software Foundation. It allows website owners to serve content on the web — hence the name "web server". Although we call Apache a web server, it is not a physical server, but rather a software that runs on a server. Its job is to establish a connection between a server and the browsers of website visitors (Firefox, Google Chrome, Safari, etc.) while delivering files back and forth between them (client-server structure). Apache is a cross-platform software, therefore it works on both UNIX and Windows servers.

4.2.3 Database – MySQL

MySQL is a fast, easy-to-use RDBMS being used for many small and big businesses. It is developed, marketed and supported by MySQL AB, which is a Swedish company. It is released under an open-source license. So you have nothing to pay to use it. It is a very powerful program in its own right. It handles a large subset of the functionality of the most expensive and powerful database packages. It uses a standard form of the well-known SQL data language. It works on many operating systems and with many languages including PHP, PERL, C, C++, JAVA, etc. It works very quickly and works well even with large data sets. It is very friendly to PHP, the most appreciated language for web development. MySQL supports large databases, up to 50 million rows or more in a table. The default file size limit for a table is 4GB, but you can increase this (if your operating system can handle it) to a theoretical limit of 8 million terabytes (TB). It is customizable. The open-source GPL license allows programmers to modify the MySQL software to fit their own specific environments.

4.3 User Flow Diagram

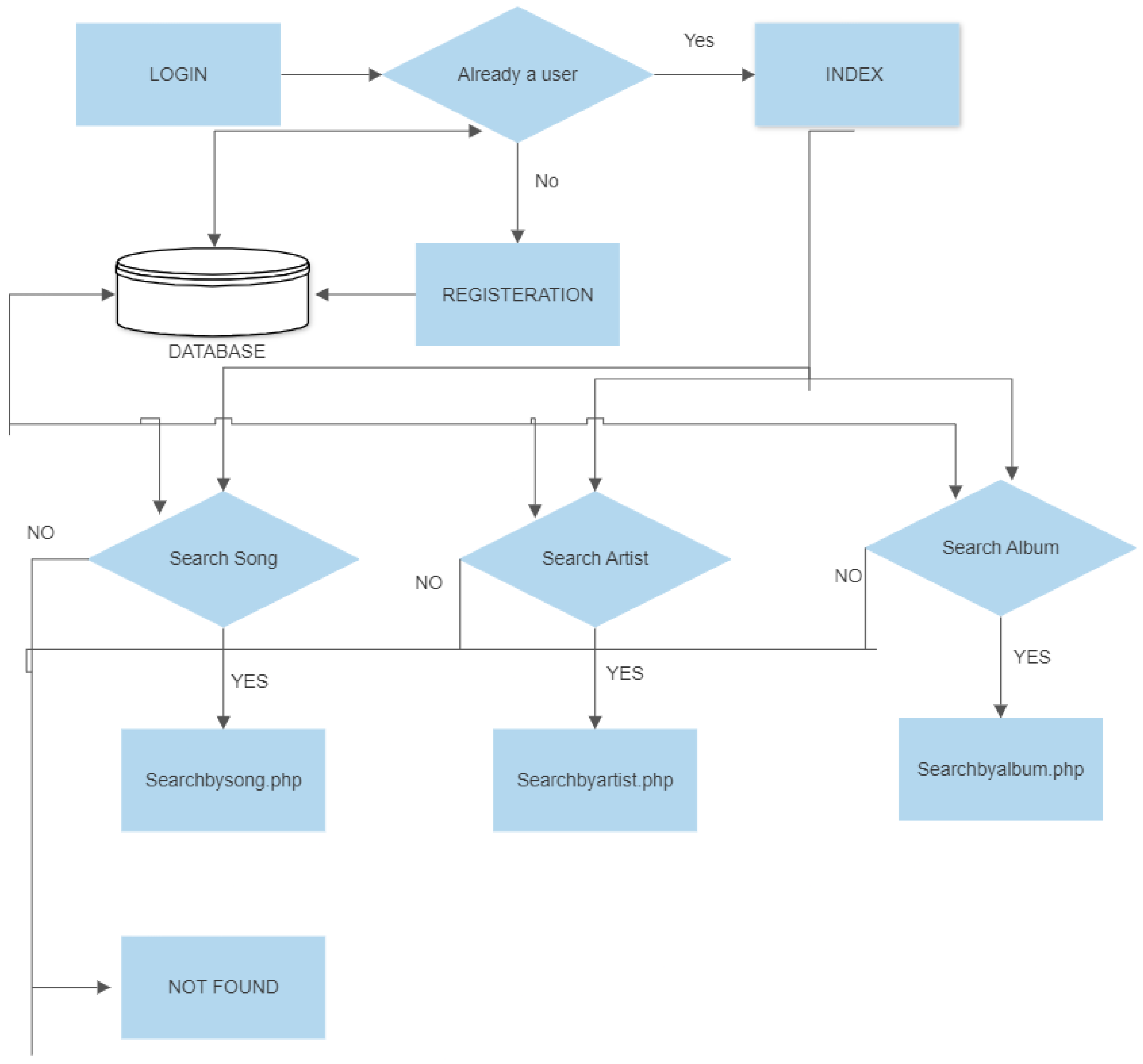


Figure 4.1 Successful Booking user flow diagram

The above figure 4.1 shows the user flow diagram for the project

1. Landing Page:

- New User: The user arrives on the landing page. If they are a new user, they see options to "Register" for a new account.

- Existing User: If the user already has an account, they can proceed to "Login" with their credentials.

2. Registration:

- The user enters their desired username, email, and password to create a new account.

- Upon successful registration, the user is redirected to the login page.

3. Login:

- The user enters their registered email and password to log in to the Musify app.

- If the credentials are correct, the user is authenticated and directed to the main dashboard.

4. Main Dashboard:

- The dashboard displays a list of trending songs that users can listen to.

- Navigation Bar: The dashboard includes a navigation bar with search options for songs, artists, and albums.

5. Song Search:

- The user enters the name of the song they want to search for in the "Song Search" bar.

- The app performs a search query and displays matching song results.

6. Artist Search:

- The user enters the name of the artist they want to search for in the "Artist Search" bar.

- The app performs a search query and displays matching artist results.

7. Album Search:

- The user enters the name of the album they want to search for in the "Album Search" bar.

- The app performs a search query and displays matching album results.

8. Artist Page:

- If the user clicks on an artist's name from the search results, they are directed to the artist's page.

- The artist's page shows details about the artist, such as their name, genre, and a list of albums they have contributed to.

9. Album Page:

- If the user clicks on an album from the search results, they are directed to the album's page.

- The album page displays details about the album, including its name, cover image, and a list of tracks associated with it.

10. Song Playback:

- When the user clicks on a song from the search results or trending list, the song starts playing with audio controls.

- The user can pause, play, and control the volume of the song.

11. Logout:

- The user can log out from their account by clicking on the "Logout" option in the navigation bar.

- After logging out, the user is redirected to the login page.

4.4 Discussion of Code Segment

This section talks about the important code sections and modules that are implemented in the Music Management System “MUSIFY” project. These modules add logic to the complete system, and make it function the way it is supposed to. It also talks about the integration between the front end HTML code and the back end MySQL database.

Login Page

localhost/exdbms/login.php

```
<?php include('server.php') ?>
<form method="post" action="login.php">
<?php include('errors.php'); ?>
        <div class="input-group">
                <label>Username</label>
                <input type="text" name="username" >
        </div>
        <div class="input-group">
                <label>Password</label>
                <input type="password" name="password">
        </div>
        <div class="input-group">
                <button type="submit" class="btn" name="login_user">Login</button>
        </div>
        <p>
                Not yet a member? <a href="register.php">Sign up</a>
        </p>
  </form>
```

REGESTRATION PAGE

localhost/exdbms/register.php

```
<?php include('server.php') ?>

<form method="post" action="register.php">

        <?php include('errors.php'); ?>
        <div class="input-group">
          <label>Username</label>
          <input type="text" name="username" value="<?php echo $username; ?>">
        </div>
        <div class="input-group">
          <label>Email</label>
          <input type="email" name="email" value="<?php echo $email; ?>">
        </div>
        <div class="input-group">
          <label>Password</label>
          <input type="password" name="password_1">
        </div>
        <div class="input-group">
          <label>Confirm password</label>
          <input type="password" name="password_2">
        </div>
        <div class="input-group">
          <button type="submit" class="btn" name="reg_user">Register</button>
        </div>
        <p>
                Already a member? <a href="login.php">Sign in</a>
        </p>
  </form>
```

SERVER

localhost/exdbms/server.php

```
<?php

session_start();

// initializing variables

$username = "";

$email    = "";

$errors = array();
```

```
// connect to the database

$db = mysqli_connect('localhost', 'root', '', 'dbms_project');

if (isset($_POST['reg_user'])) {

  // receive all input values from the form

  $username = mysqli_real_escape_string($db, $_POST['username']);

  $email = mysqli_real_escape_string($db, $_POST['email']);

  $password_1 = mysqli_real_escape_string($db, $_POST['password_1']);

  $password_2 = mysqli_real_escape_string($db, $_POST['password_2']);

  if (count($errors) == 0) {

            $password = ($password_1);

            $query = "INSERT INTO users (username, email, password)

                              VALUES('$username', '$email', '$password')";

            mysqli_query($db, $query);

            $_SESSION['username'] = $username;

            $_SESSION['success'] = "You are now logged in";

            header('location: index.php');

  }

}

if (isset($_POST['login_user'])) {

  $username = mysqli_real_escape_string($db, $_POST['username']);

  $password = mysqli_real_escape_string($db, $_POST['password']);

}

?>
```

INDEX

localhost/exdbms/index.php

```
<?php
  session_start();
  if (!isset($_SESSION['username'])) {
        $_SESSION['msg'] = "You must log in first";
        header('location: login.php');
  }
  if (isset($_GET['logout'])) {
        session_destroy();
        unset($_SESSION['username']);
        header("location: login.php");
  }
<nav>
      <ul>
      <li class="brand"><img src="OIG.jpg" alt="Spotify">Musify</li>
      <div class="brand">
    <form action="searchbysongs2.php" method="GET">
      <input type="text" name="search" class="search-input" placeholder="Search Song">
      <button type="submit" class="search-button">Search</button>
    </form>
</div>
  <div class="brand">
  <form action="searchbyartist.php" method="GET">
      <input type="text" name="search" class="search-input" placeholder="Search by Artist">
      <button type="submit" class="search-button">Search</button>
    </form>
  </div>
  <div class="brand">
  <form action="searchbyalbum.php" method="GET">
      <input type="text" name="search" class="search-input" placeholder="Search by Album">
      <button type="submit" class="search-button">Search</button>
    </form>
  </div>
```

```
 <?php  if (isset($_SESSION['username'])) : ?>
      <p>Welcome <strong><?php echo $_SESSION['username']; ?></strong></p>
      <p> <a href="index.php?logout='1'" style="color: red;">logout</a> </p>
  <?php endif ?>
    </ul>
  </nav>
```

localhost/exdbms/searchbysong2.php

```
<?php
$username = "";
$email    = "";
$errors = array();
$db = mysqli_connect('localhost', 'root', '', 'dbms_project');
if ($db->connect_error) {
    die("Connection failed: " . $db->connect_error);
}
$searchQuery = $_GET['search'];
$sql = "SELECT * FROM track WHERE track_name LIKE '%$searchQuery%'";
$result = $db->query($sql);
$songs = array();
if ($result->num_rows > 0) {
    while ($row = $result->fetch_assoc()) {
        $song = array(
            'songName' => $row['track_name'],
            'filePath' => $row['track_path'],
            'coverPath' => $row['cover_path']
        );
        $songs[] = $song;
    }
}
else {
   echo "No results found.";
}
?>
```

localhost/exdbms/searchbyartist.php

```
<?php
$username = "";
$email    = "";
$errors = array();
$db = mysqli_connect('localhost', 'root', '', 'dbms_project');
if ($db->connect_error) {
   die("Connection failed: " . $db->connect_error);
}
$searchQuery = $_GET['search'];
$sql = "SELECT *
        FROM artist a
        JOIN album ab ON a.artist_id = ab.artid
        JOIN track t ON ab.album_id = t.albid
        WHERE a.artist_name LIKE '%$searchQuery%'";
$result = $db->query($sql);
$songs = array();
if ($result->num_rows > 0) {
   while ($row = $result->fetch_assoc()) {
        $song = array(
           'songName' => $row['track_name'],
           'filePath' => $row['track_path'],
           'coverPath' => $row['cover_path']
        );
        $songs[] = $song;
   }
}
else {
    echo "No results found.";
}
?>
```

localhost/exdbms/searchbyalbum.php

```
<?php
$username = "";
$email    = "";
$errors = array();
$db = mysqli_connect('localhost', 'root', '', 'dbms_project');
if ($db->connect_error) {
    die("Connection failed: " . $db->connect_error);
}
$searchQuery = $_GET['search'];
//$sql = "SELECT * FROM track WHERE track_name LIKE '%$searchQuery%'";
$sql = "SELECT *
        FROM track t
        JOIN album ab ON t.albid = ab.album_id
        JOIN artist a ON ab.artid = a.artist_id
        WHERE ab.album_name LIKE '%$searchQuery%'";
$result = $db->query($sql);
$songs = array();
if ($result->num_rows > 0) {
    while ($row = $result->fetch_assoc()) {
        $song = array(
            'songName' => $row['track_name'],
            'filePath' => $row['track_path'],
            'coverPath' => $row['cover_path']
        );
        $songs[] = $song;
    }
}
else {
    echo "No results found.";
}
?>
```

The working of the Musify website can be explained as follows:

- The website uses XAMP as the backend, which consists of Apache, MySQL, PHP, and Perl. Apache is the web server that handles the requests from the users. MySQL is the database management system that stores and retrieves the data for the website. PHP is the scripting language that processes the logic and functionality of the website. Perl is an optional component that can be used for additional features.
- The website has five tables in the database: USER, ARTIST, ALBUM, TRACK, and PLAYLIST. These tables store the information about the users, artists, albums, tracks, and playlists respectively. The tables are related to each other by foreign keys that reference the primary keys of other tables.
- The website has an index page that displays a list of trending songs that users can listen to. The index page also has a navigation bar with three search options for songs, artists, and albums. The users can enter their search queries in the respective search bars and get matching results from the database.
- The website also has pages for artists, albums, and tracks. These pages show the details of the selected artist, album, or track. The users can also play the songs from these pages using audio controls.
- The website allows users to create playlists of their favorite songs. The users can add or remove songs from their playlists using buttons on the track pages. The playlists are stored in the database along with the user id and playlist id.
- The website requires users to register and log in to use its features. The registration process involves entering a username, email, and password. The login process involves entering an email and password. The user credentials are stored in the USER table and verified by PHP scripts.
- The website uses SQL queries to interact with the database. SQL queries are statements that specify what data to select, insert, update, or delete from the database. PHP scripts execute SQL queries using functions such as mysqli_query() or PDO::query(). The results of SQL queries are returned as arrays or objects that can be displayed on the web pages using HTML and CSS.

4.5 Discussion of Results

Login Page

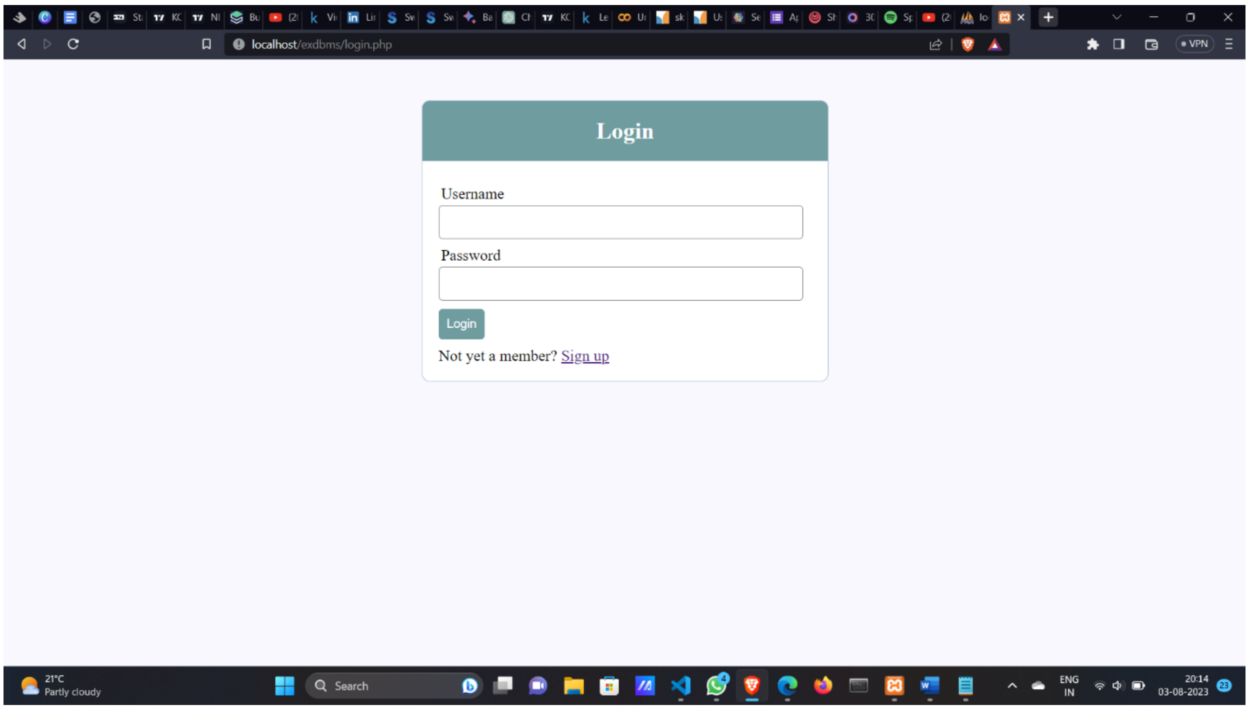


Figure 4.2 Login Page

If the user is registered and is active, he/she will be able to login using his/her username and password. After login is successful, he/she will be redirected to the Index Page.Else he can register himself by clicking on **Sign Up**

Registration Page

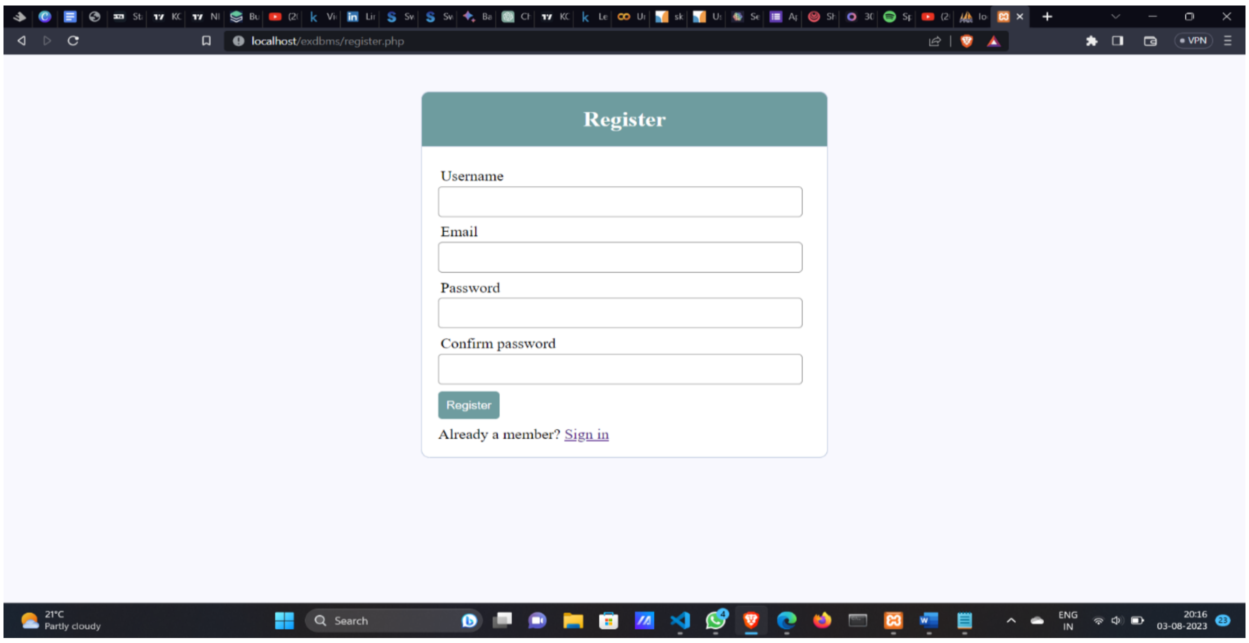
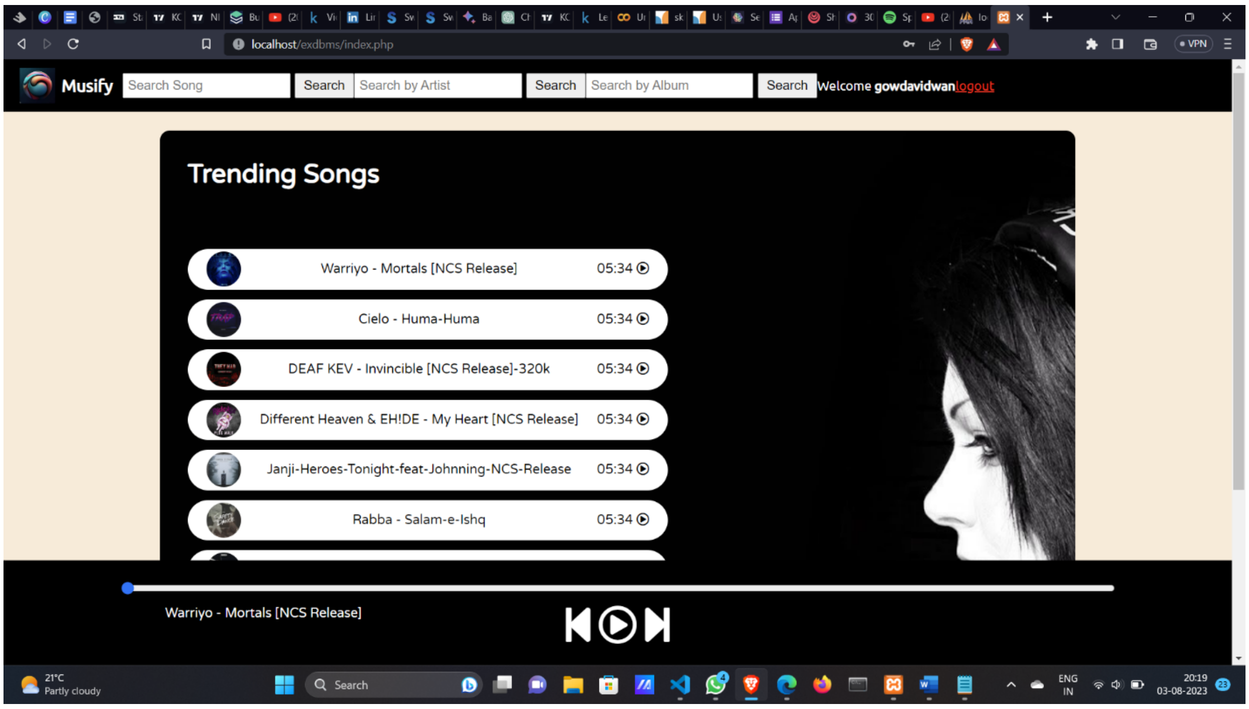


Figure 4.3 Registration Page

The registration page is the page where new users can create an account for the Musify website. The page has a form that asks for the user's desired username, email, and password. The page also has a button that submits the form and validates the user input. If the input is valid, the page redirects the user to the Index page. If the input is invalid, the page displays an error message and asks the user to try again.

INDEX PAGE



Warriyo - Mortals [NCS Release] 05:34
Cielo - Huma-Huma 05:34
DEAF KEV - Invincible [NCS Release]-320k 05:34
Different Heaven & EH!DE - My Heart [NCS Release] 05:34
Janji-Heroes-Tonight-feat-Johnning-NCS-Release 05:34
Rabba - Salam-e-Ishq 05:34

Warriyo - Mortals [NCS Release]

Figure 4.4 Index Page

The index page of Musify serves as the main landing page for users, offering a glimpse of the website's key features and content. It presents a curated list of trending songs, showcasing the hottest and most popular tracks at a glance. Users can easily explore these trending songs and play them with audio controls for an immersive music experience. The navigation bar on the index page provides three search bars, enabling users to search for songs, artists, and albums

Result for Search Song

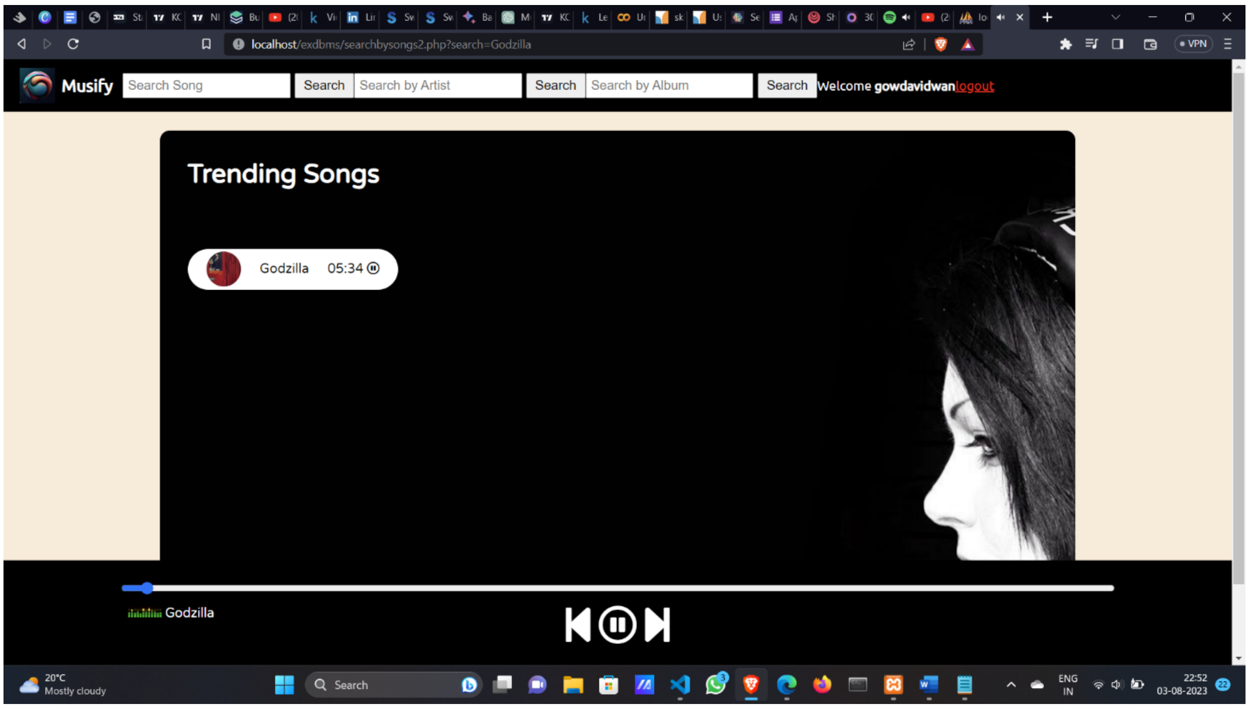


Figure 4.5 Search track using its name

The "Search by Song" feature in Musify allows users to find specific music tracks by entering the name of the name of the song they are looking for. By providing a "Search by Song" feature, Musify enables users to quickly find their favorite tracks and discover new music that aligns with their musical preferences, enhancing their overall experience on the platform.

RESULT FOR SEARCH BY ARTIST NAME

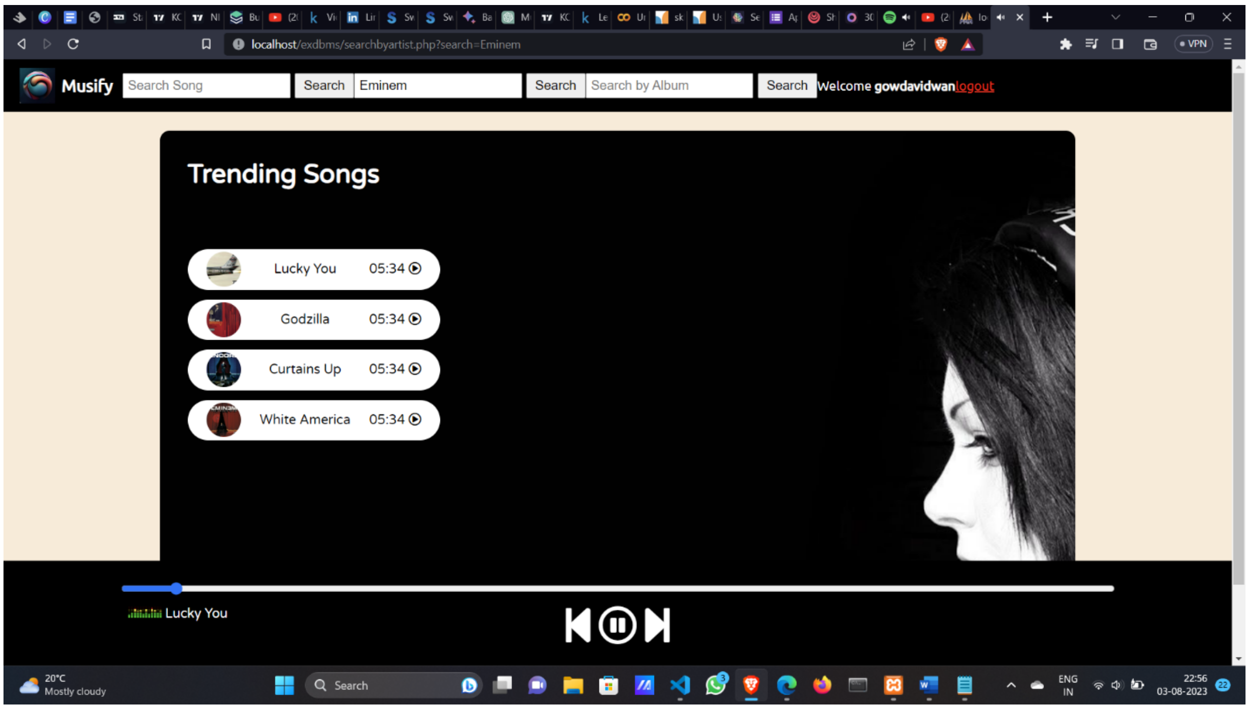


Figure 4.6 Search tracks using its artist name

The "Search by Artist" feature in Musify allows users to discover music by searching for specific artists or bands. The "Search by Artist" feature empowers users to discover and explore music from their favorite artists and find new musicians that align with their musical preferences, providing an enriching and personalized experience on the Musify platform.

RESULT FOR SEARCH BY ALBUM NAME

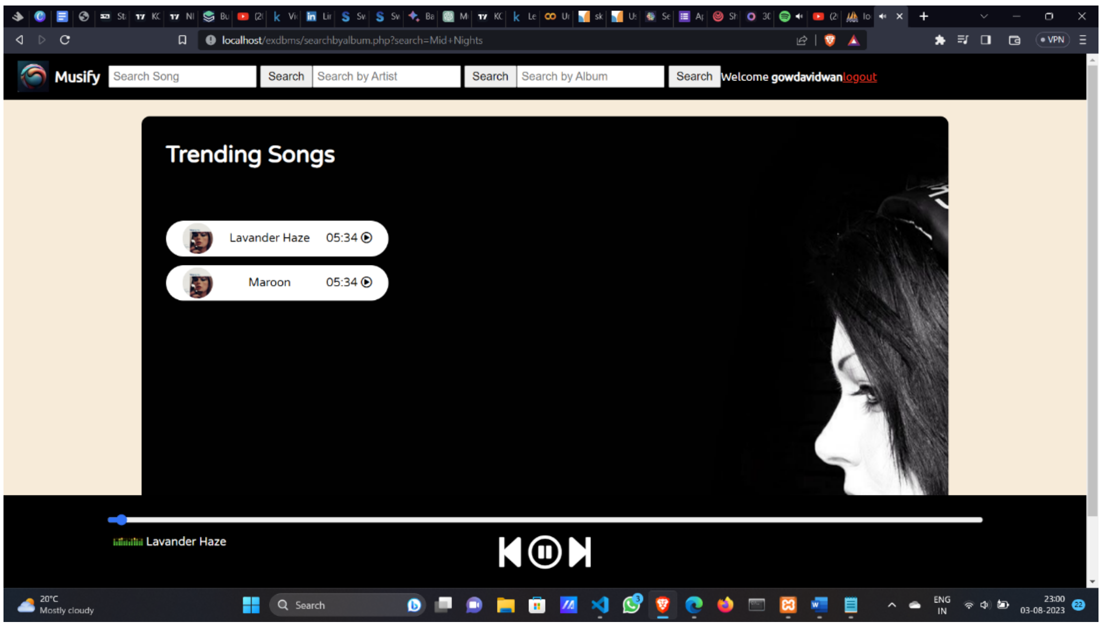
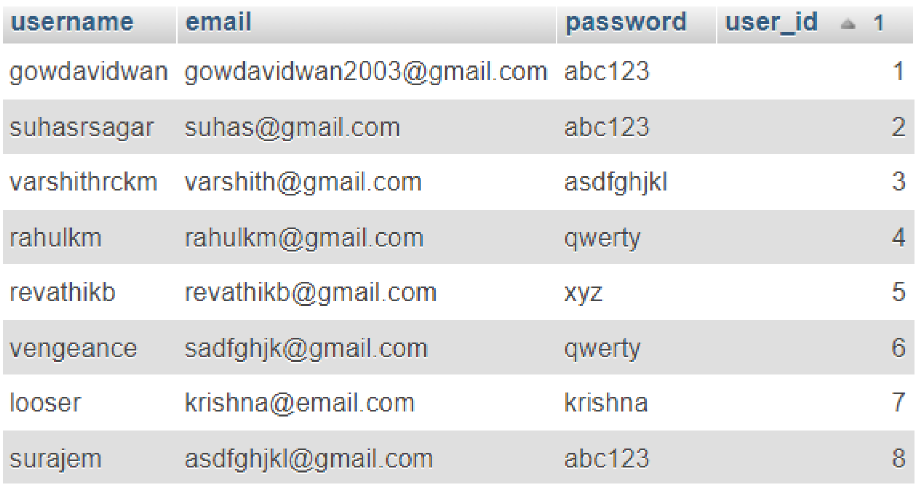


Figure 4.7 Search tracks using its album name

The "Search by Album" feature in Musify empowers users to discover music by searching for specific albums. The "Search by Album" feature enhances users' music discovery experience, enabling them to find and enjoy entire albums created by their favorite artists or explore new albums that align with their musical preferences on the Musify platform.

TABLES STORED IN DATABASE

Users table-

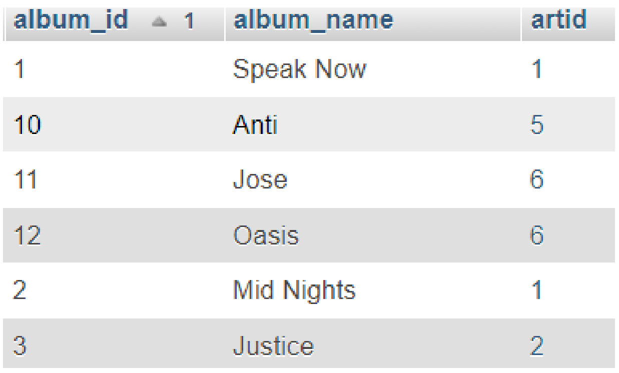


username	email	password	user_id ▲ 1
gowdavidwan	gowdavidwan2003@gmail.com	abc123	1
suhasrsagar	suhas@gmail.com	abc123	2
varshithrckm	varshith@gmail.com	asdfghjkl	3
rahulkm	rahulkm@gmail.com	qwerty	4
revathikb	revathikb@gmail.com	xyz	5
vengeance	sadfghjk@gmail.com	qwerty	6
looser	krishna@email.com	krishna	7
surajem	asdfghjkl@gmail.com	abc123	8

Figure 4.8 Users table

The "user" table in Musify stores user-related information, featuring columns for user_id (primary key), username, email, and password. It manages user registration and login data, ensuring secure access to the platform.

Album table-

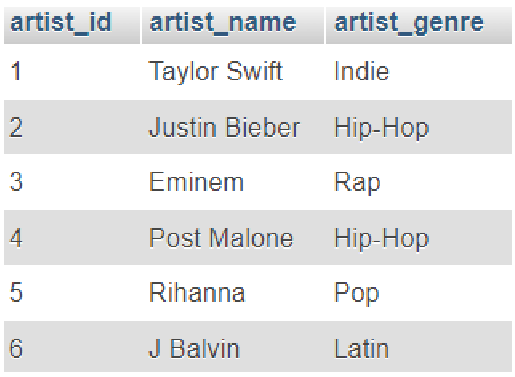


album_id ▲ 1	album_name	artid
1	Speak Now	1
10	Anti	5
11	Jose	6
12	Oasis	6
2	Mid Nights	1
3	Justice	2

Figure 4.9 Album Table

The "album" table in Musify contains information about music albums available on the platform. It includes columns for album_id (primary key), album_name, and artid (foreign key referencing artist_id in the "artist" table). This table facilitates album organization, linking albums to their respective artists,

Artist table-

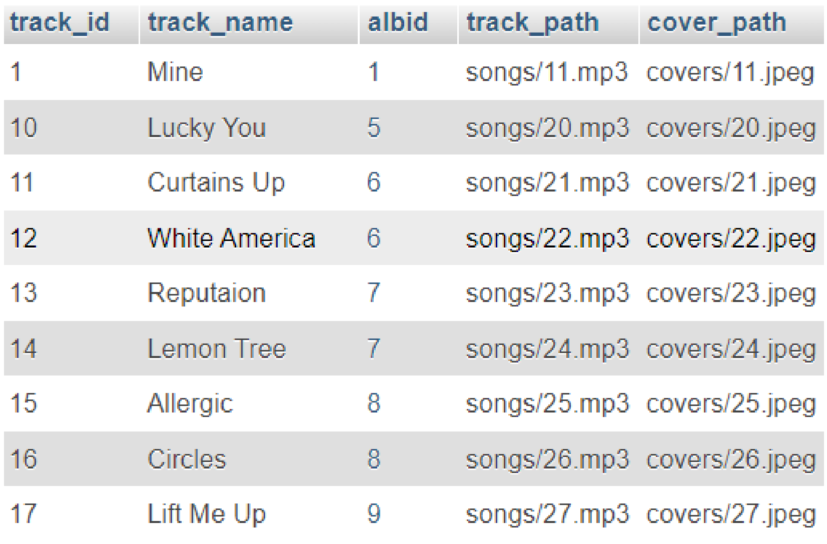


artist_id	artist_name	artist_genre
1	Taylor Swift	Indie
2	Justin Bieber	Hip-Hop
3	Eminem	Rap
4	Post Malone	Hip-Hop
5	Rihanna	Pop
6	J Balvin	Latin

Figure 4.10 Artist Table

The "artist" table in Musify stores details about music artists or bands featured on the platform. It consists of columns for artist_id (primary key), artist_name, and artist_genre. This table helps organize artist-related information, making it easy to associate artists with their albums and tracks

Track Table-



track_id	track_name	albid	track_path	cover_path
1	Mine	1	songs/11.mp3	covers/11.jpeg
10	Lucky You	5	songs/20.mp3	covers/20.jpeg
11	Curtains Up	6	songs/21.mp3	covers/21.jpeg
12	White America	6	songs/22.mp3	covers/22.jpeg
13	Reputaion	7	songs/23.mp3	covers/23.jpeg
14	Lemon Tree	7	songs/24.mp3	covers/24.jpeg
15	Allergic	8	songs/25.mp3	covers/25.jpeg
16	Circles	8	songs/26.mp3	covers/26.jpeg
17	Lift Me Up	9	songs/27.mp3	covers/27.jpeg

Figure 4.11 Track Table

The "track" table in Musify stores information about individual music tracks, with columns for track_id (primary key), track_name, track_path (audio file location), cover_path (cover image location), and albid (foreign key referencing album_id in the "album" table).

4.6 Application of project

The Musify DBMS project has several applications that can benefit both users and developers. Some of the key applications include:

1. Music Discovery and Enjoyment: Musify provides users with a platform to discover new music, explore artists' discographies, and enjoy their favorite tracks and albums.

2. Personalized Music Recommendations: The system can use user data and preferences to offer personalized music recommendations, enhancing the user experience and encouraging engagement.

3. User Management and Security: The project includes user registration and login functionalities, ensuring secure access to the platform and managing user data securely.

4. Dynamic Content Management: With a database-driven approach, Musify efficiently manages dynamic content like trending songs, search results, and artist/album details.

5. Audio Playback and Streaming: The application enables users to play music directly on the website, providing a seamless audio streaming experience.

6. Music Catalog Management: Musify allows easy addition, modification, and removal of tracks, artists, and albums from the music catalog through the database.

7. Search and Filtering Capabilities: The search functionality allows users to find specific songs, artists, and albums quickly, enhancing user satisfaction and interaction.

8. Data Analytics and Insights: Developers can use the database to gather valuable data about user behavior, music preferences, and trending songs, leading to data-driven insights and decision-making.

9. Developer Learning and Practice: As a DBMS mini-project, Musify serves as a learning tool for developers to practice database design, SQL queries, and PHP programming.

10. Integration with Front-end and UI: The database project complements the front-end development, enabling the integration of data with user interfaces and designing user-friendly interactions.

11. Scalability and Future Enhancements: As the project evolves, the database can accommodate new features, scalability, and future enhancements for a continuously improving music app.

Overall, the Musify DBMS project has practical applications in the field of music app development, user engagement, data management, and personalized music experiences, making it a valuable learning experience and a functional platform for music enthusiasts.

Chapter 5

CONCLUSION AND FUTURE ENHANCEMENT

5.1 Conclusion

The Musify DBMS project is a user-friendly music app that lets users discover, enjoy, and connect with their favorite music. It has features such as secure login, personalized recommendations, and convenient search functionalities for songs, artists, and albums.The project uses a MySQL database to store and manage music-related data, such as tracks, artists, and albums. It also uses stored procedures and SQL queries to optimize data retrieval and performance.The project uses PHP as a server-side scripting language to render dynamic content and integrate with the front-end user interface.As a full-stack developer, creating the Musify project has shown expertise in database design and management, web development, and database-driven applications. It has also been a valuable learning opportunity to enhance these skills.The Musify project is a successful example of combining web development, database management, and music expertise to create an appealing and useful music app.

5.2 Future Enhancements

Certainly! Here are some further enhancements you can consider implementing to improve and expand the Musify music app:

1. User Playlists: Allow users to create and manage their playlists, enabling them to curate and organize their favorite tracks for easy access.

2. Social Sharing: Implement social sharing features, allowing users to share their favorite songs, albums, and playlists with their friends on social media platforms.

3. User Reviews and Ratings: Enable users to leave reviews and ratings for songs, albums, and artists, fostering user engagement and community interaction.

4. User Activity Feed: Create a personalized activity feed for each user, showing their recent listens, liked tracks, and other music-related activities.

5. User Following and Followers: Introduce a "follow" system, allowing users to follow their favorite artists and receive updates on their latest releases and activities.

6. Related Artists and Recommendations: Enhance music recommendations by suggesting related artists and albums based on user listening history.

7. Song Lyrics: Include lyrics for songs, enabling users to sing along or understand the meaning behind the music.

8. User Comments on Songs: Allow users to comment on songs, fostering discussions and interactions around specific tracks.

Chapter 5

9. Offline Listening: Introduce offline listening capabilities, enabling users to download their favorite songs for offline playback.

10. Music Genres and Playlists: Implement genre-based playlists or radio stations, allowing users to explore music based on their preferred genres.

11. Music Charts and Top Lists: Display popular tracks, albums, and artists on Musify through charts and top lists, showcasing trending music.

12. Music Events and Concerts: Provide information about upcoming music events and concerts related to the artists on Musify.

13. User Profile Customization: Allow users to customize their profile pages, such as adding profile pictures and updating their music preferences.

14. Music Recommendations from Friends: Introduce a feature that suggests music based on the listening habits of users' friends or people they follow.

These enhancements can add significant value to the Musify music app, making it more engaging, personalized, and feature-rich for its users. As you implement these new features, continuously gather user feedback to iterate and improve the app further.

Chapter 6

REFERENCES

[1] CHAT GPT → https://chat.openai.com/

[2] MICROSOFT BING AI

[3] phpMyAdmin’s documentation

[4] https://open.spotify.com/

[5] https://dev.mysql.com/doc/

[6] https://www.mysqltutorial.org/