SYMPHONY-MUSIC PLAYER

A PROJECT COMPONENT REPORT

Submitted by

POORNIMA S	**(Reg. No. 202104104)**
SIMRITI R	**(Reg. No. 202104151)**

for the Theory Cum Project Component

of

19CS694 – WEB USER INTERFACE DESIGN

during

VI Semester – 2023 – 2024

DEPARTMENTOFCOMPUTER SCIENCE ANDENGINEERING

MEPCO SCHLENK ENGINEERING COLLEGE, SIVAKASI

(An Autonomous Institution affiliated to Anna University Chennai)

April 2024

MEPCO SCHLENK ENGINEERING COLLEGE, SIVAKASI

(An Autonomous Institution affiliated to Anna University Chennai)

Department of Computer Science and Engineering

BONAFIDE CERTIFICATE

Certified that this project component report titled **SYMPHONY-MUSIC PLAYER** is the bonafide work of **POORNIMA S (Reg. No. 202104104),**and **SIMRITI R (Reg. No. 202104151)** who carried out this work under my guidance for the Theory cum Project Component course **"19CS694 – WEB USER INTERFACE DESIGN"** during the sixth semester.

Dr.S.KARKUZHALI, M.E., Ph.D.	**Dr. J. RAJA SEKAR,** M.E.,Ph.D.
Assistant Professor	Professor
Course Instructor	Head of the Department
Department of Computer Science & Engg.	Department of Computer Science & Engg.
Mepco Schlenk Engineering College	Mepco Schlenk Engineering College
Sivakasi.	Sivakasi.

Submitted for viva-Voce Examination held at **MEPCO SCHLENK ENGINEERING COLLEGE (Autonomous), SIVAKASI** on **………/……/…. 20..........**

Internal Examiner **External Examiner**

ABSTRACT

Nowadays, music plays a vital role among people. Most people prefer music while they are doing some work. Many music websites insist that users pay to access their pages to listen to music. However, our music website, Symphony, is completely user-friendly, allowing all music listeners to access it for free. Our website also provides users with more features, such as creating their own playlists to listen to songs according to their taste. They can search for a song by its name, singer name, or album name. Additionally, our music website has a review page where users can leave comments about the page and give ratings. Users can also download the songs they want for free. We have added playback and speed limit options as well. Users can relax their minds by listening to their favorite songs anytime and anywhere. This website is truly a user-friendly music player website.

ACKNOWLEDGEMENT

First and foremost, we thank the **LORD ALMIGHTY** for his abundant blessings that is showered upon our past, present and future successful endeavors.

We extend our sincere gratitude to our college management and Principal **Dr. S. Arivazhagan M.E., Ph.D.,** for providing sufficient working environment such as systems and library facilities. We also thank him very much for providing us with adequate lab facilities, which enable us to complete our project.

We would like to extend our heartfelt gratitude to **Dr. J. Raja Sekar M.E., Ph.D.,** Professor and Head, Department of Computer Science and Engineering, Mepco Schlenk Engineering College for giving me the golden opportunity to undertake a project of this nature and for his most valuable guidance given at every phase of our work.

We would also like to extend our gratitude and sincere thanks to **Dr.S.Karkuzhali M.E., Ph.D.,** Assistant Professor, Department of Computer Science and Engineering, Mepco Schlenk Engineering College for being our Project Mentor. He has put his valuable experience and expertise in directing, suggesting and supporting us throughout the Project to bring out the best.

Our sincere thanks to our revered **faculty members and lab technicians** for their help over this project work.

Last but not least, we extend our indebtedness towards out beloved family and our friends for their support which made the project a successful one.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 PERSCPECTIVE

Our platform offers two login options: Admin and User. Each login requires a username and password. New users can register using either their email or phone number. The Admin holds the authority to enrich the music library by adding songs across various genres and recommending tracks based on user preferences. Furthermore, the Admin can distinguish verified accounts with a blue tick and extend special privileges to premium users. We ensure that any advertisements displayed between songs are reasonable. Users have the freedom to curate their playlists and mark songs as favorites. They can explore songs by artists, genre, language, and movies. Users can upgrade to a premium account by subscribing for a specific duration. During this period, they enjoy the perks of downloading songs for offline listening without any interruptions. Furthermore, users can provide feedback by rating the website after each session or at their convenience. They can also submit queries or report issues encountered while using the platform, which will promptly be addressed by the Admin.

1.2 OBJECTIVE

To design a user-friendly music website addressing global music lovers' common issues. Offering intuitive navigation, extensive music libraries, personalized recommendations, and seamless offline listening, we aim to enhance the music experience for all users.

1.3 SCOPE

Nowadays most of the music websites don’t allow to skip to the particular part of the song and the songs are unable to play in the order .This aims in playing songs in order and skip to particular part of the song.

CHAPTER 2

REQUIREMENT DESCRIPTION

2.1 FUNCTIONAL REQUIREMENTS

- Playback Controls: Enable users to play, pause, skip, and adjust volume during song playback.
- Playlist Management: Allow users to create, edit, and delete playlists, as well as add or remove songs from them.
- Search Functionality: Provide users with the ability to search for songs by title, artist, album, genre, or other criteria.
- Offline Mode: Enable users to download songs for offline listening and manage offline content.
- User Authentication and Account Management: Allow users to create accounts, log in securely, and manage their profiles, including preferences and playlists.

2.2 NON-FUNCTIONAL REQUIREMENTS

- Performance: Ensure fast and responsive performance, with minimal loading times for song playback, search results, and playlist management.
- Usability: Design an intuitive and user-friendly interface that is easy to navigate, with clear controls and instructions for all functionalities.
- Security: Implement robust security measures to protect user data, including encryption of sensitive information such as login credentials and adherence to industry-standard security protocols.
- Reliability: Ensure the music player operates reliably without frequent crashes or errors, providing a consistent and stable user experience.
- Scalability: Design the music player to accommodate a growing number of users and a larger music library, with the ability to handle increased traffic and data storage requirements efficiently.

CHAPTER 3

SYSTEM DESIGN

3.1 ARCHITECTURE DESIGN

The architectural design explains about the flow of user and admin. User/admin needs to sign-up/sign-in to use the music player. Admin can add new songs, view their own and user's profile, delete users, and view the review and rating given by the users. Users can search songs by it title/singer name/album name. And they can create their own playlist to listen songs according to their music taste. Users can gave review about the page and also the rating. Suppose the user forget their password they can reset the password.

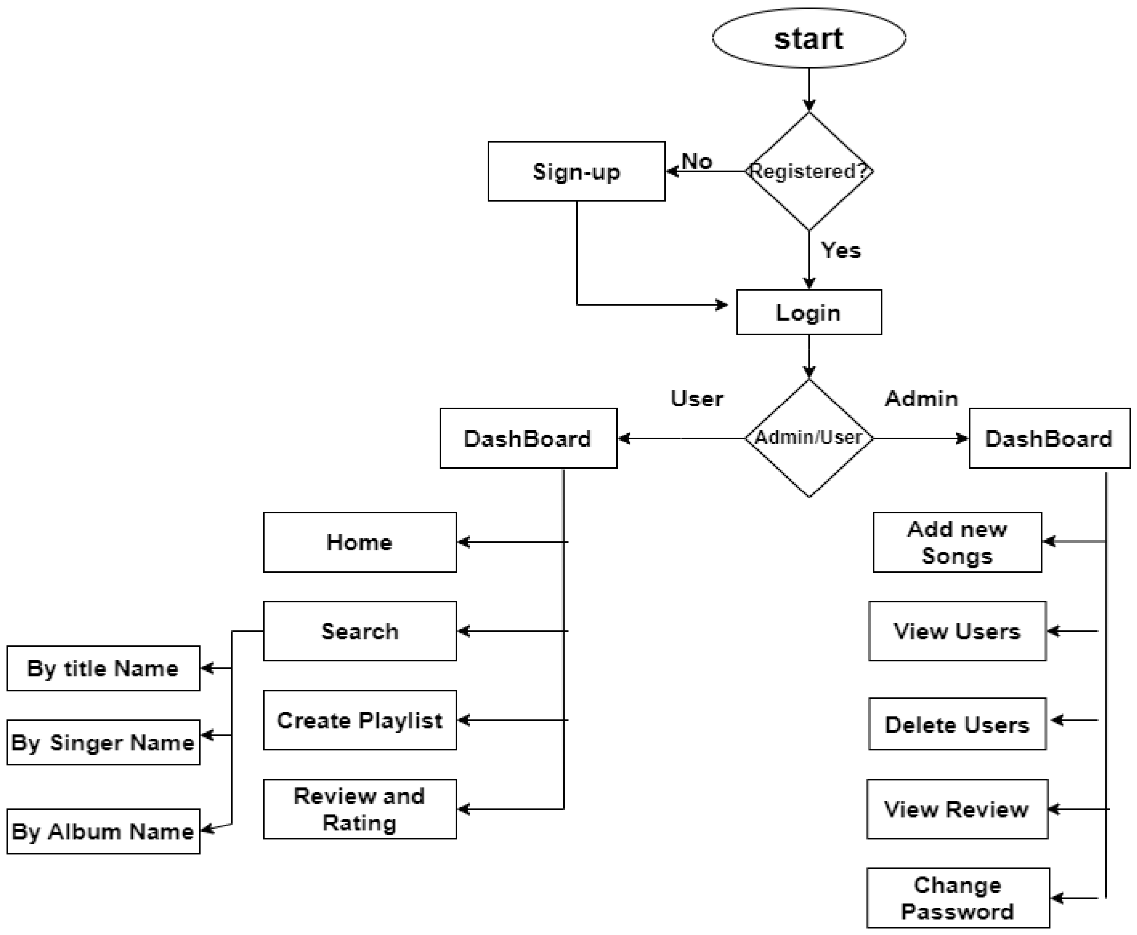


Figure 3.1: Architecture Diagram of Music Player Website

3.2 DESIGN COMPONENTS

3.2.1 Front End:

The music player uses Android Studios for developing interactive pages.

3.2.2 Back End:

Uses Firebase for back end to store data.

3.3 DATABASE DESCRIPTION

Listed below gives a description of database document schemas used for Music player

3.3.1 User Register

As shown in table 3.1, user register contains the details of the users.

Table 3.1: User Register

Attribute Name	Type	Constraint(s)	Description
Email	String	Required	Email id of the user
Phone number	Number	Required, phn no = 10	Phone number of the user
Password	String	Required, Password > 6	Password for the account

3.3.2 Admin/user Login

As shown in table 3.2, admin/user login contains the details of the users/admin.

Table 3.2: Admin/user Login

Attribute Name	Type	Constraint(s)	Description
Email	String	Required	Email id of the user
Password	String	Required	Password for the account

3.3.3 Add song

As shown in table 3.3, add song contains the details of the added song.

Table 3.3: Add song

Attribute Name	Type	Constraint(s)	Description
Song Name	String	Required	Name of the Song
Singer Name	String	Required	Name of the singer
Song URL	String	Required	URL of the song

3.3.4 Review

As shown in table 3.4, your review contains the details of the review given by the user.

Table 3.4: Review

Attribute Name	Type	Constraint(s)	Description
Email	String	Required	Email id of the user
Review	String	Required	Review for the page
Rating	Number	Required	Rating for the page

3.4 LOW LEVEL DESIGN

The following section illustrates the functionalities of the system.

3.4.1 Login

Table 3.5 shows the login details of the application.

Table 3.5 Login Details

Files used	Login.ts, login.html, login.css
Short Description	Allows the user/admin to login to the webpage
Arguments	Email, Password
Return	Success/Failure in login
Pre-Condition	The user/admin must have an account
Post-Condition	The home page will be displayed
Exception	Invalid Email and password
Actor	Admin and users

3.4.2 Sign up

Table 3.6 shows the sign up details of the application.

Table 3.6 Sign up Details

Files used	Signup.ts, signup.html, signup.css
Short Description	Allows the user to signup to the webpage
Arguments	Email, Phone number, Password
Return	Success/Failure in is signup
Pre-Condition	Anyone register in the application
Post-Condition	User registered successfully
Exception	Invalid email, username and password
Actor	Users

3.4.3 Search song

Table 3.7 shows the search song details of the application.

Table 3.7 Search song Details

Files used	Search.ts, Search.html, Search.css
Short Description	Allows the user to search the song by its title/singer name/album name
Arguments	Title/Singer name/Album name
Return	Success/Failure in searching the song
Pre-Condition	Know anything about the song
Post-Condition	Display the searched song
Exception	No matching music found.
Actor	Users

3.4.4 Review

Table 3.8 shows the review details of the application.

Table 3.8 Review Details

Files used	Review.ts, Review.html, Review.css
Short Description	Allows the user to gave review and rating about the songs
Arguments	Email, Review, Rating
Return	Review submitted successfully
Pre-Condition	Experience about the usage of page
Post-Condition	Successfully Submitted
Exception	Fill up the required fields
Actor	Users

3.4.5 Create playlist

Table 3.9 shows the create playlist details of the application.

Table 3.9 Create playlist Details

Files used	Playlist.ts, Playlist.html, Playlist.css
Short Description	Allows the user to create their own playlist
Arguments	Add, Remove
Return	Playlist Created
Pre-Condition	Empty playlist, songs yet to add
Post-Condition	Songs added to playlist
Exception	Can’t download from playlist
Actor	Users

3.4.6 Change password

Table 3.10 shows the Change password details of the application.

Table 3.10 Change password Details

Files used	change.ts, change.html, change.css
Short Description	Allows the user to change their password
Arguments	Email, old password, new password
Return	Success in updation of the password
Pre-Condition	The password must be available.
Post-Condition	Password changed
Exception	Account is unavailable
Actor	Users

3.4.7 View user's profile

Table 3.11 shows the View user's profile details of the application.

Table 3.11 View user's profile Details

Files used	view.ts, view.html, view.css
Short Description	Allows the admin to view the user's profile
Arguments	Username
Return	User's profile displayed successfully
Pre-Condition	User have an account
Post-Condition	Displayed User's Profile
Exception	User not available
Actor	Admin

3.4.8 Add song

Table 3.12 shows the Add song details of the application.

Table 3.12 Add song Details

Files used	Addsong.ts, Addsong.html, Addsong.css
Short Description	Allows the admin to Add songs to the website
Arguments	Song Name, Singer name, Song URL
Return	Success/Failure in is insertion of the song
Pre-Condition	The song must be unavailable.
Post-Condition	New song added successfully
Exception	Song already available
Actor	Admin

3.4.13 View review

Table 3.4 shows the View review details of the application.

Table 3.13 View review Details

Files used	viewreview.ts, viewreview.html, viewreview.css
Short Description	Allows the admin to view the user’s review
Arguments	Username
Return	User’s review displayed successfully
Pre-Condition	User already gave review
Post-Condition	Displayed User’s review
Exception	User not available
Actor	Admin

3.4.10 View song

Table 3.14 shows the View song details of the application.

Table 3.14 View song Details

Files used	viewsong.ts, viewsong.html, viewsong.css
Short Description	Allows the admin to the songs added
Arguments	Song name
Return	Song displayed
Pre-Condition	Admin added the song
Post-Condition	Displayed added songs list
Exception	Song not added
Actor	Admin

3.4.11 Delete user

Table 3.15 shows the Delete user details of the application.

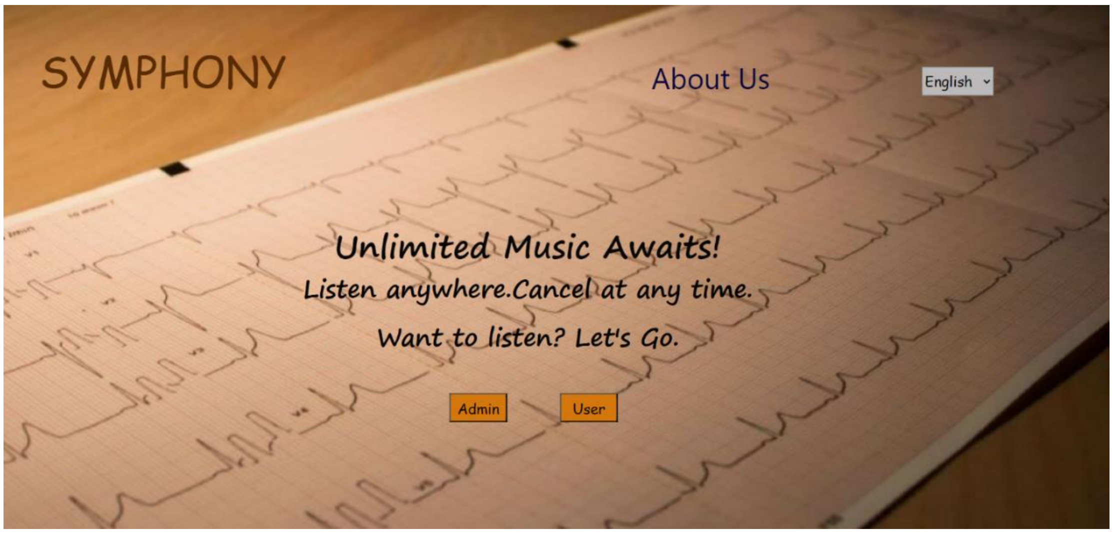
Table 3.15 Delete user Details

Files used	deleteuser.ts, deleteuser.html, deleteuser.css
Short Description	Allows the admin to delete user
Arguments	Username
Return	User deleted successfully
Pre-Condition	User already available
Post-Condition	Deleted the user
Exception	User not available
Actor	Admin

3.5 USER INTERFACE DESIGN

3.5.1 Main Activity

Figure 3.2 provide the interface for main activity.

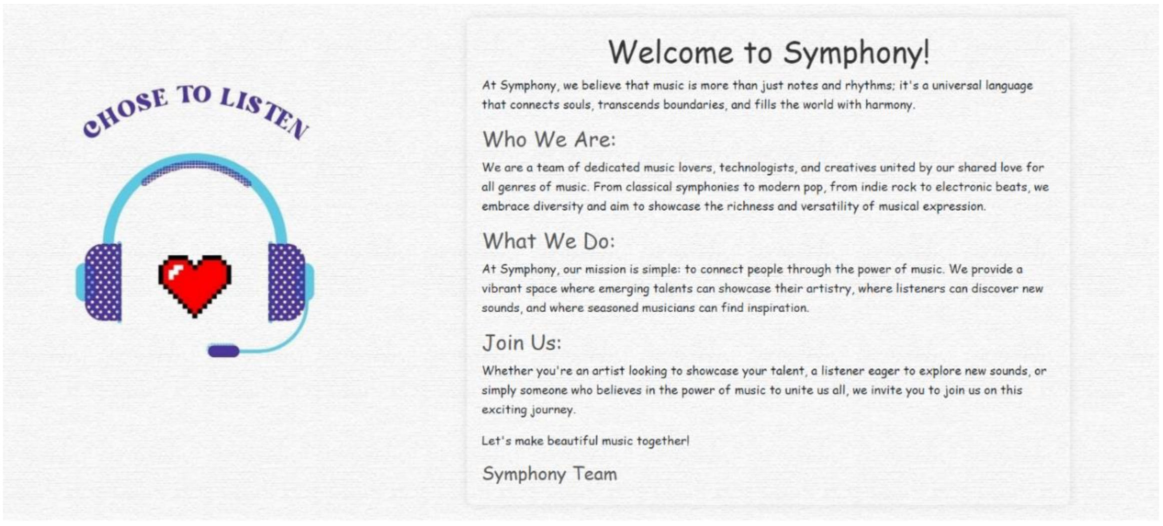


Unlimited Music Awaits!
Listen anywhere.Cancel at any time.
Want to listen? Let's Go.

Figure 3.2: Main layout of Music player

3.5.2 About us

Figure 3.3 provide the interface for About us.



Welcome to Symphony!

At Symphony, we believe that music is more than just notes and rhythms; it's a universal language that connects souls, transcends boundaries, and fills the world with harmony.

Who We Are:

We are a team of dedicated music lovers, technologists, and creatives united by our shared love for all genres of music. From classical symphonies to modern pop, from indie rock to electronic beats, we embrace diversity and aim to showcase the richness and versatility of musical expression.

What We Do:

At Symphony, our mission is simple: to connect people through the power of music. We provide a vibrant space where emerging talents can showcase their artistry, where listeners can discover new sounds, and where seasoned musicians can find inspiration.

Join Us:

Whether you're an artist looking to showcase your talent, a listener eager to explore new sounds, or simply someone who believes in the power of music to unite us all, we invite you to join us on this exciting journey.

Let's make beautiful music together!

Symphony Team

Figure 3.3: About us of Music player

3.5.3 User register

Figure 3.4 provide the interface for User register.

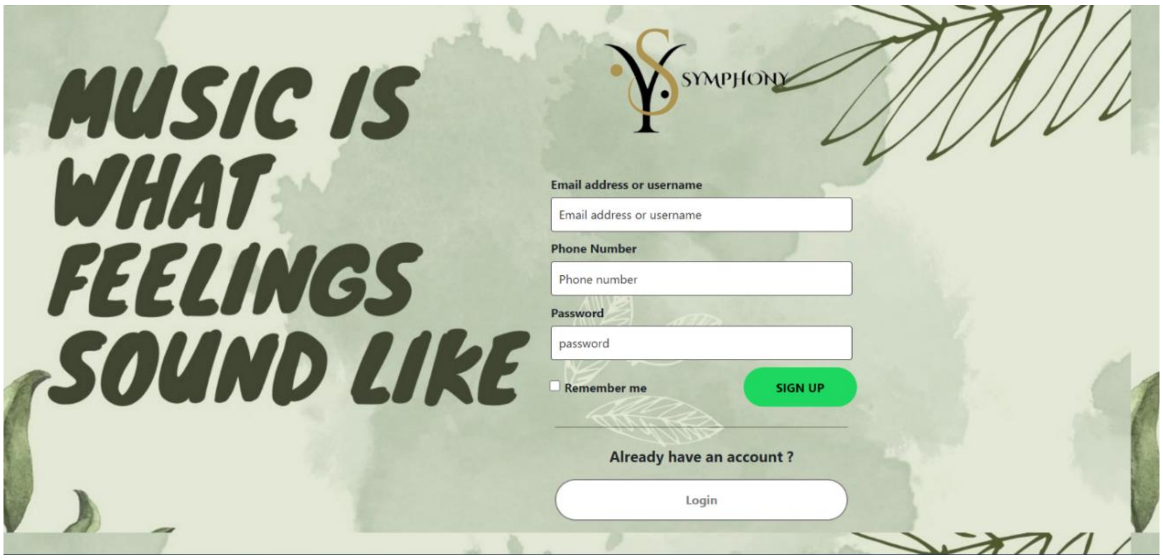

MUSIC IS WHAT FEELINGS SOUND LIKE
SYMPHONY
Email address or username
Email address or username
Phone Number
Phone number
Password
password
Remember me
SIGN UP
Already have an account ?
Login

Figure 3.4: User register of Music player

3.5.4 User login

Figure 3.5 provide the interface for User login.

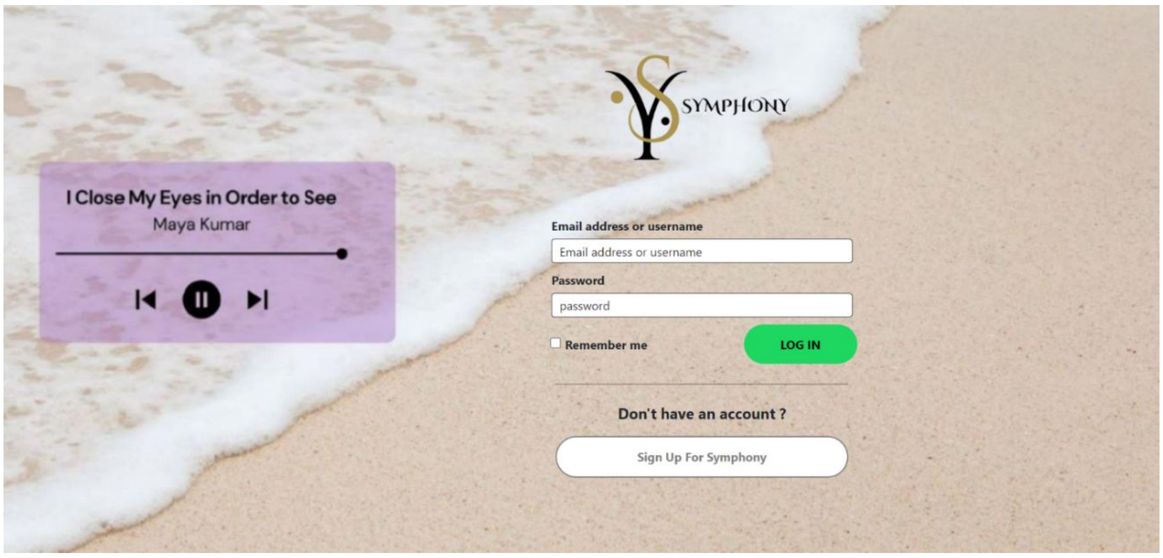


Figure 3.5: User login of Music player

3.5.5 Admin login

Figure 3.6 provide the interface for Admin login.

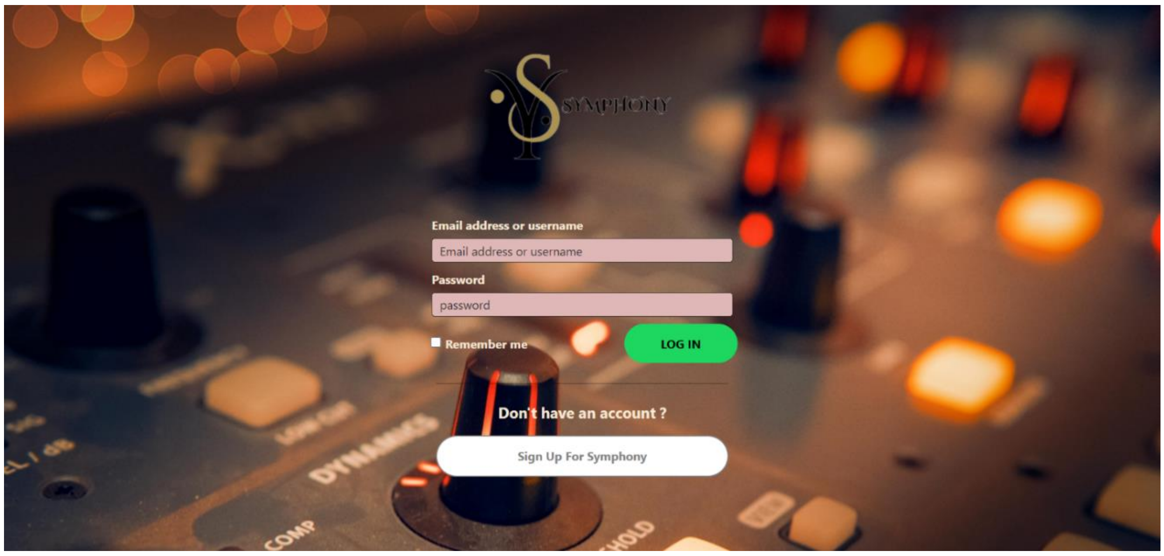
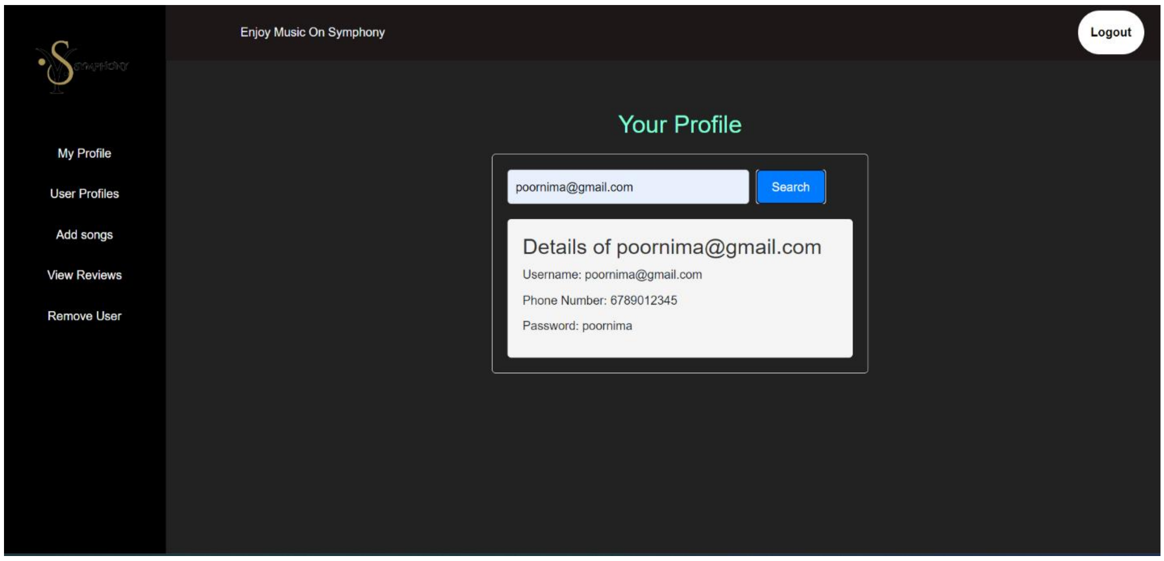


Figure 3.6: Admin login of Music player

3.5.6 Admin profile

Figure 3.7 provide the interface for Admin profile.



Details of poornima@gmail.com

Username: poornima@gmail.com

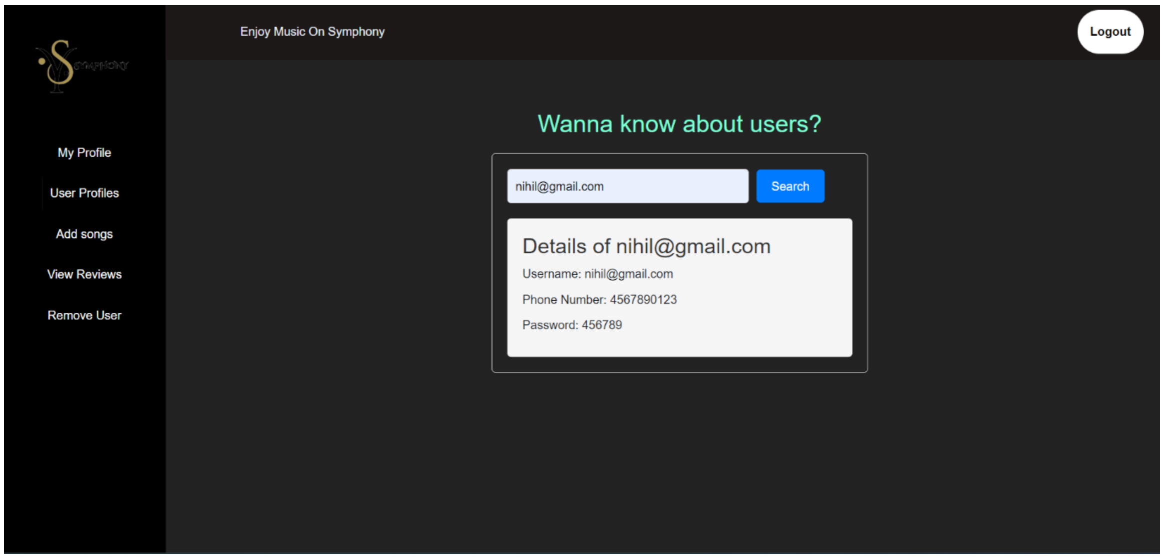
Phone Number: 6789012345

Password: poornima

Figure 3.7: Admin profile of Music player

3.5.7 User profile

Figure 3.8 provide the interface for User profile.



Details of nihil@gmail.com

Username: nihil@gmail.com

Phone Number: 4567890123

Password: 456789

Figure 3.8: User profile of Music player

3.5.8 Add song

Figure 3.9 provide the interface for add song.

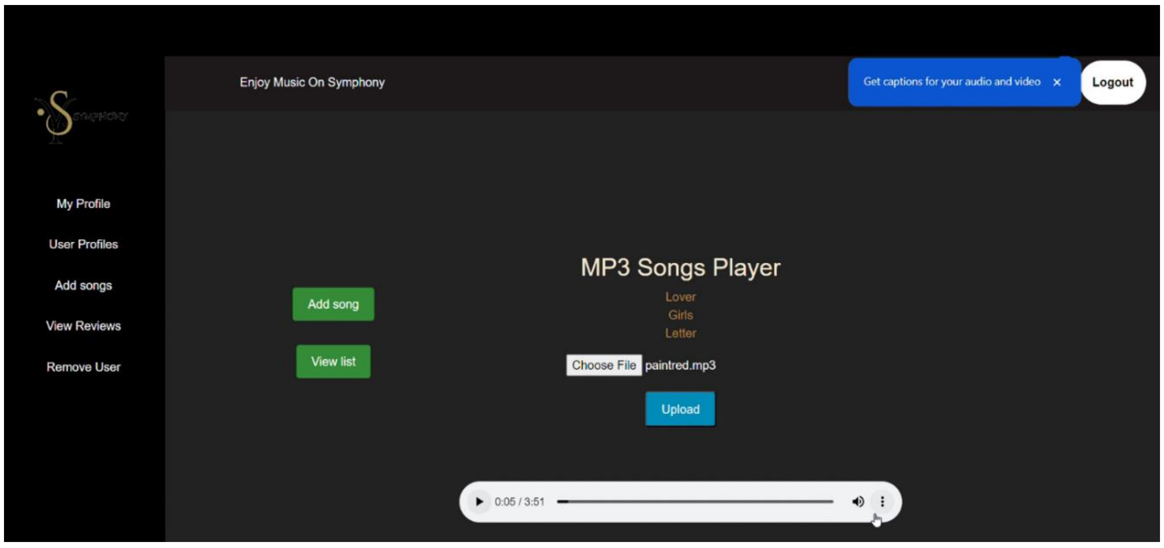
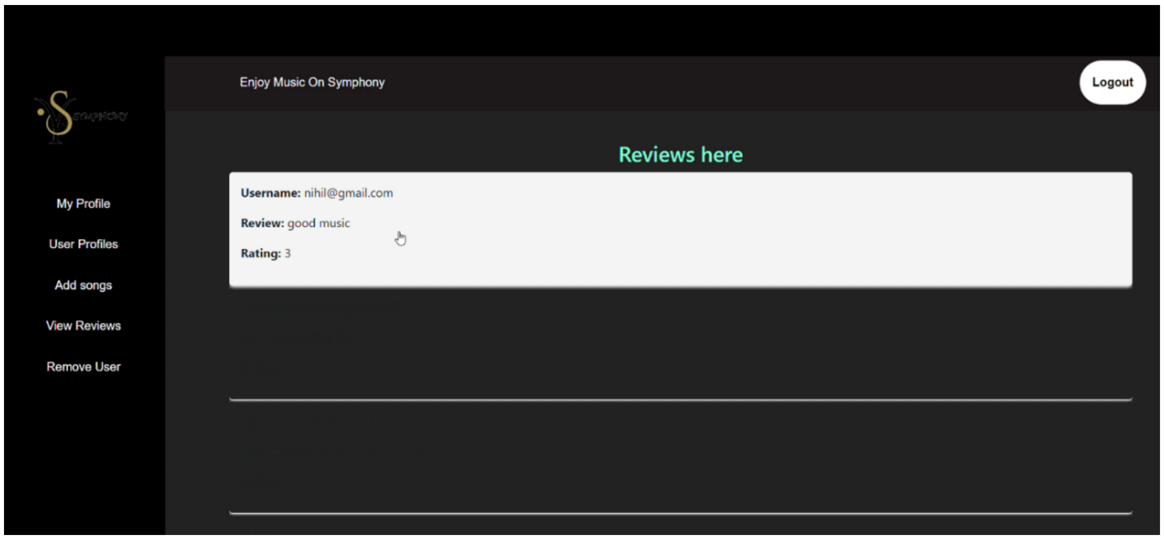


Figure 3.9: Add song of Music player

3.5.9 View review

Figure 3.10 provide the interface for View review.

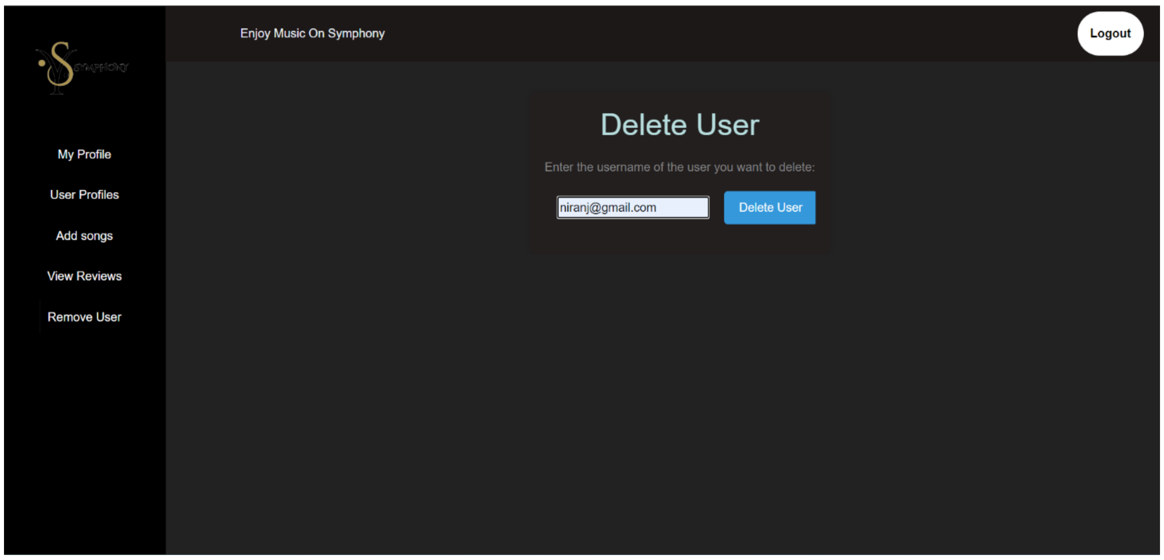


Username: nihil@gmail.com
Review: good music
Rating: 3

Figure 3.10: View review of Music player

3.5.10 Delete user

Figure 3.11 provide the interface for Delete user.



Enter the username of the user you want to delete:
niranj@gmail.com

Figure 3.11: Delete user of Music player

3.5.11 User Home

Figure 3.12 provide the interface for User Home.

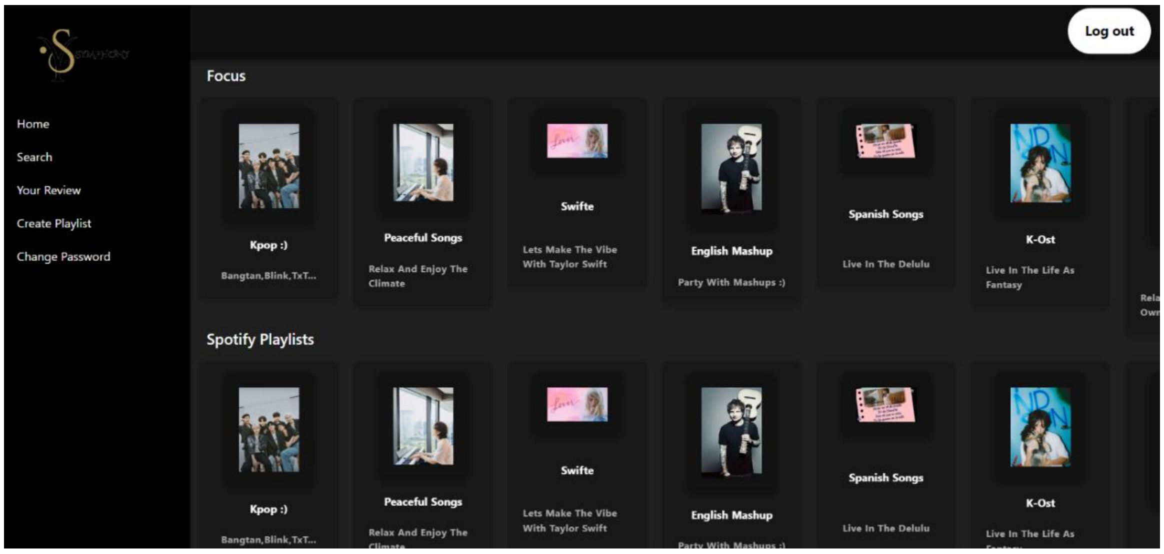
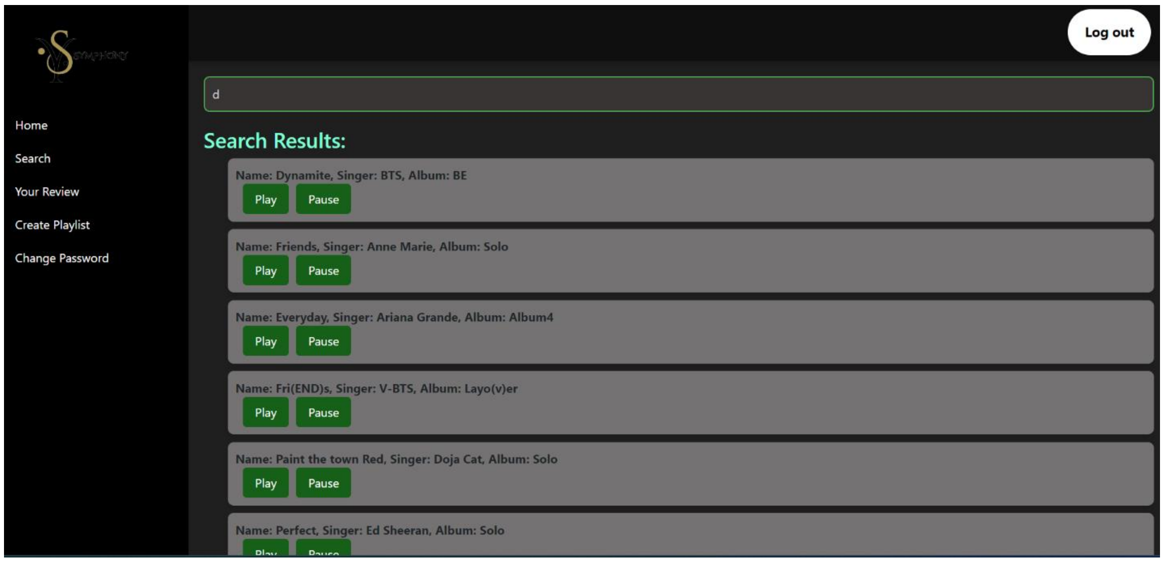


Figure 3.12: User home of Music player

3.5.12 Search page

Figure 3.13 provide the interface for Search page.



Search Results:

Name: Dynamite, Singer: BTS, Album: BE
Play Pause

Name: Friends, Singer: Anne Marie, Album: Solo
Play Pause

Name: Everyday, Singer: Ariana Grande, Album: Album4
Play Pause

Name: Fri(END)s, Singer: V-BTS, Album: Layo(v)er
Play Pause

Name: Paint the town Red, Singer: Doja Cat, Album: Solo
Play Pause

Name: Perfect, Singer: Ed Sheeran, Album: Solo

Figure 3.13: Search page of Music player

3.5.13 User review

Figure 3.14 provide the interface for User review.

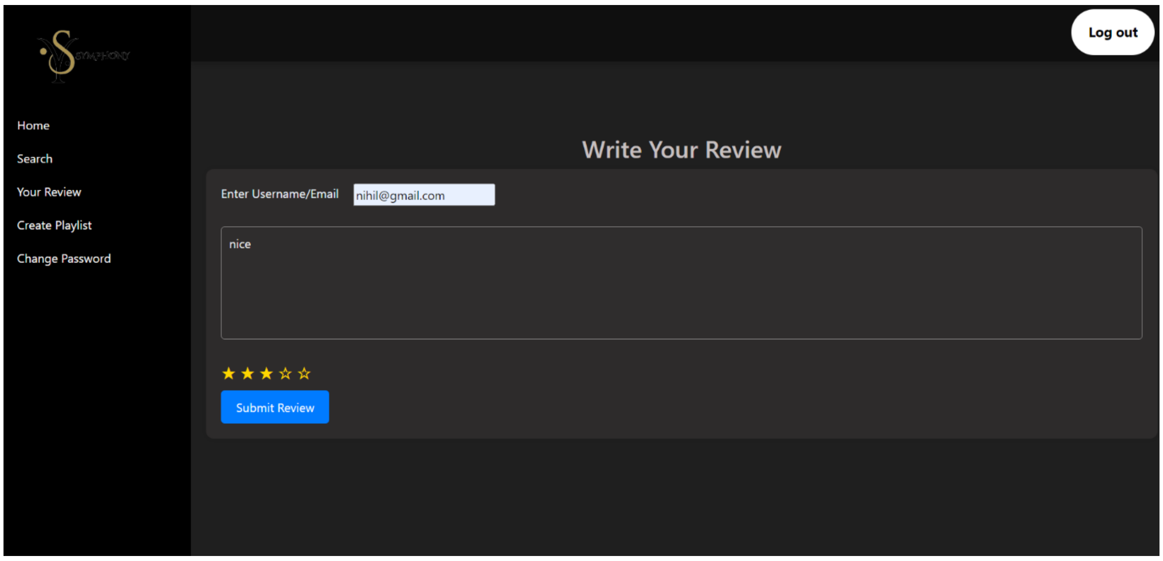


Figure 3.14: User review of Music player

3.5.14 Create playlist

Figure 3.15 provide the interface for Create playlist.

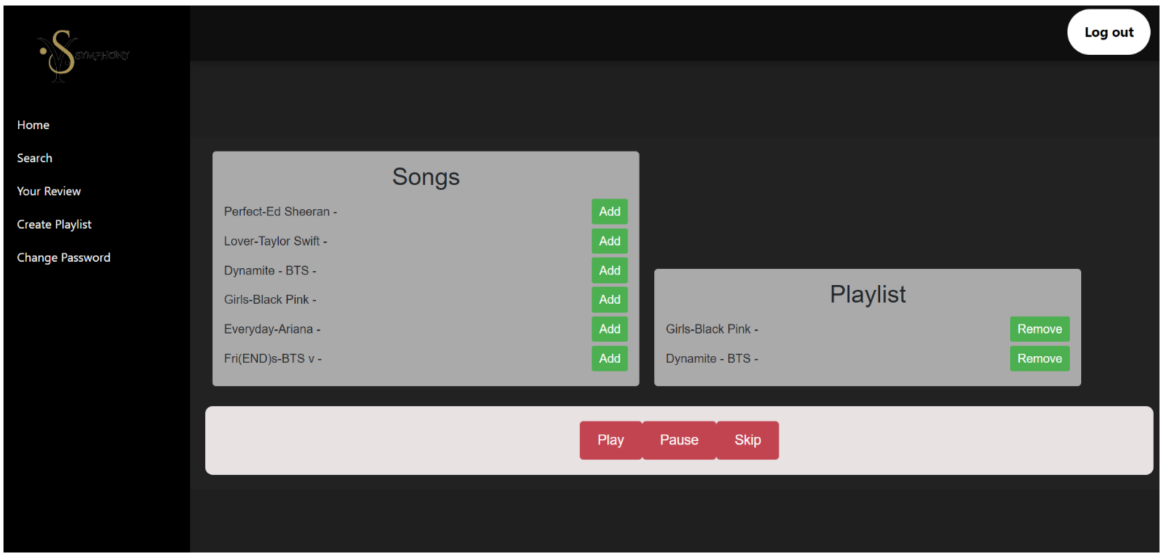
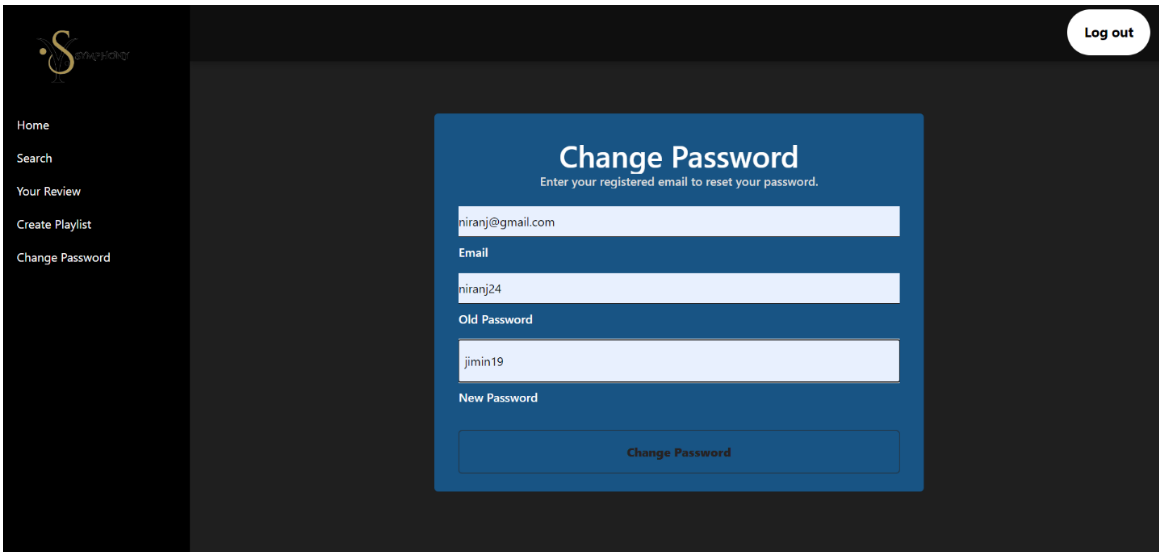


Figure 3.15: Create playlist of Music player

3.5.15 Change password

Figure 3.16 provide the interface for Change password.



Change Password

Enter your registered email to reset your password.

niranj@gmail.com

Email

niranj24

Old Password

jimin19

New Password

Change Password

Figure 3.16: Change password of Music player

3.5.16 Change password

Figure 3.17 provide the interface for download song.

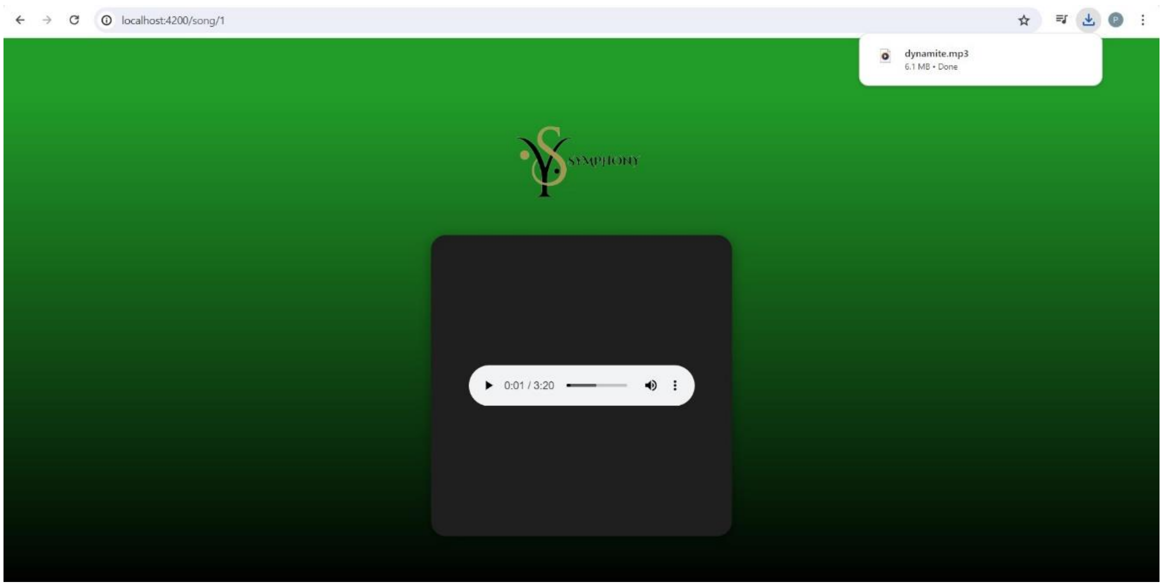


Figure 3.17: Downloads of Music player

CHAPTER 4

SYSTEM IMPLEMENTATION

4.1 LOGIN IMPLEMENTATION

The login credentials are obtained. If the credentials are OK, then the user is redirected to the homepage

```
GET email, password
   IF email, password valid
           RETURN homepage
   ELSE
           TOAST Invalid Credential
```

4.2SIGNUP IMPLEMENTATION

The form fields are obtained. If they are valid, then the user is added to the database.

```
GET requestFields
   IF requestFields valid
           RETURN added to db
   ELSE
           TOAST enter valid details
```

CHAPTER 5

RESULTS AND DISCUSSION

5.1 TEST CASES AND RESULTS

5.1.1 Test Cases and Results for Login function:

The Table 5.1, Table 5.2 shows that the possible test data for the both positive and negative test case given below, if the user is already having account then the output is true otherwise false.

Table 5.1: Positive Test Case and result for Login

Test Case ID	TC1
Test Case Description	It tests whether the given login details are valid or not
Test Data	kookie@gmail.com,kookie
Expected Output	TRUE
Result	PASS

Table 5.2: Negative Test Case and result for Login

Test Case ID	TC2
Test Case Description	It tests whether the given login details are valid or not
Test Data	Kookie,12345
Expected Output	FALSE
Result	PASS

5.1.2 Test Cases and Results for sign up function:

The Table 5.1, Table 5.2 shows that the possible test data for the both positive and negative test case given below, if the user is already having account then the output is false otherwise true.

Table 5.3: Positive Test Case and result for sign up

Test Case ID	TC3
Test Case Description	It tests whether the given sign up details are valid or not
Test Data	kookie@gmail.com,1234567898,kookie
Expected Output	TRUE
Result	PASS

Table 5.4: Negative Test Case and result for sign up

Test Case ID	TC4
Test Case Description	It tests whether the given sign up details are valid or not
Test Data	Kookie,23456765,12345
Expected Output	FALSE
Result	PASS

5.1.3 Test Cases and Results for search song function:

The Table 5.1, Table 5.2 shows that the possible test data for the both positive and negative test case given below.

Table 5.5: Positive Test Case and result for search song

Test Case ID	TC5
Test Case Description	It tests whether the song is available or not
Test Data	Euphoria,Jungkook,BE
Expected Output	TRUE
Result	PASS

Table 5.6: Negative Test Case and result for search song

Test Case ID	TC6
Test Case Description	It tests whether the song is available or not
Test Data	Friends
Expected Output	FALSE
Result	PASS

5.1.4 Test Cases and Results for create playlist function:

The Table 5.1, Table 5.2 shows that the possible test data for the both positive and negative test case given below.

Table 5.7: Positive Test Case and result for create playlist

Test Case ID	TC7
Test Case Description	It tests whether the playlist is created or not
Test Data	Dynamite, Girls, Summer-Added
Expected Output	TRUE
Result	PASS

Table 5.8: Negative Test Case and result for create playlist

Test Case ID	TC8
Test Case Description	It tests whether the playlist is created or not
Test Data	<<Empty>>
Expected Output	FALSE
Result	PASS

5.1.5 Test Cases and Results for change password function:

The Table 5.1, Table 5.2 shows that the possible test data for the both positive and negative test case given below.

Table 5.9: Positive Test Case and result for change password

Test Case ID	TC9
Test Case Description	It tests whether the password was changed or not
Test Data	kookie@gmail.com,kookie,13101995
Expected Output	TRUE
Result	PASS

Table 5.10: Negative Test Case and result for change password

Test Case ID	TC10
Test Case Description	It tests whether the password was changed or not
Test Data	kookie@gmail.com,kookie,1395
Expected Output	FALSE
Result	PASS

5.1.6 Test Cases and Results for add song function:

The Table 5.1, Table 5.2 shows that the possible test data for the both positive and negative test case given below.

Table 5.11: Positive Test Case and result for add song

Test Case ID	TC11
Test Case Description	It tests whether the song was added or not
Test Data	Every day, Ariana Grande, everyday.mp3
Expected Output	TRUE
Result	PASS

Table 5.12: Negative Test Case and result for add song

Test Case ID	TC12
Test Case Description	It tests whether the song was added or not
Test Data	<<empty>>
Expected Output	FALSE
Result	PASS

5.1.7 Test Cases and Results for delete user function:

The Table 5.1, Table 5.2 shows that the possible test data for the both positive and negative test case given below.

Table 5.13: Positive Test Case and result for delete user

Test Case ID	TC13
Test Case Description	It tests whether user was deleted or not
Test Data	kookie@gmail.com
Expected Output	TRUE
Result	PASS

Table 5.14: Negative Test Case and result for delete user

Test Case ID	TC14
Test Case Description	It tests whether user was deleted or not
Test Data	shalu@gmail.com
Expected Output	FALSE
Result	PASS

CHAPTER 6

CONCLUSION AND FUTURE ENHANCEMENT

In conclusion, our music player offers a seamless and enjoyable listening experience with intuitive controls, robust playlist management, and efficient search functionality. Emphasizing performance, usability, security, reliability, and scalability, it meets the diverse needs of music enthusiasts worldwide. Looking ahead, future enhancements could include implementing personalized recommendations through machine learning algorithms, enhancing social integration for sharing playlists, improving offline mode with automatic syncing and additional song information, integrating with smart devices for a seamless experience, enhancing accessibility features, introducing collaborative playlist creation, and exploring options for live streaming concerts or events within the app. These developments aim to further elevate the user experience and solidify the music player as a top choice for music lovers.

APPENDIX – A

SYSTEM REQUIREMENTS

HARDWARE REQUIREMENT:

Processor (CPU)	:	Modern multi-core processor
Memory (RAM)	:	At least 4GB of RAM
Storage	:	Sufficient storage space

SOFTWARE REQUIREMENT:

Operating System	:	Any
DBMS	:	Mongo DB
IDE used	:	Angular
Angular Version	:	17

APPENDIX – B

SOURCE CODE

About.css

```
body{
    background-image: url('/../assets/about.jpg');
    background-size: cover;
  }
  .acontain {
    max-width: 800px;
    margin: 20px auto; /* Center the container horizontally */
    padding: 20px;
    border-radius: 10px;
    box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
    transform: translateX(250px);
  }

  h1 {
    text-align: center;
    color: #333;
    font-family: 'Comic Sans MS', sans-serif;
  }

  h3, h4 {
    color: #555;
    font-family: 'Comic Sans MS', sans-serif;
  }

  p {
    line-height: 1.6;
    font-family: 'Comic Sans MS', sans-serif;
  }
```

About.html

```
<body >
    <div class="container">
      <div class="acontain">
      <h1>Welcome to Symphony!</h1>
      <p>At Symphony, we believe that music is more than just notes and rhythms;
it's a universal language that connects souls, transcends boundaries, and fills the world with
```

```
        harmony.</p>
            <h3>Who We Are:</h3>
            <p>We are a team of dedicated music lovers, technologists, and creatives united
by our shared love for all genres of music. From classical symphonies to modern pop, from
indie rock to electronic beats, we embrace diversity and aim to showcase the richness and
versatility of musical expression.</p>
            <h3>What We Do:</h3>
            <p>At Symphony, our mission is simple: to connect people through the power
of music. We provide a vibrant space where emerging talents can showcase their artistry,
where listeners can discover new sounds, and where seasoned musicians can find
inspiration.</p>
            <h3>Join Us:</h3>
            <p>Whether you're an artist looking to showcase your talent, a listener eager to
explore new sounds, or simply someone who believes in the power of music to unite us all,
we invite you to join us on this exciting journey.</p>
            <p>Let's make beautiful music together!</p>
            <h4>Symphony Team</h4>
            </div>
          </div>
        </body>
```

Addsong.css

```
.container {
    text-align: center;
    margin-top: 74px;
  }
  .switch-btn {
    background-color: #358c38;
    border: none;
    color: white;
    padding: 10px 20px;
    text-align: center;
    text-decoration: none;
    display: inline-block;
    font-size: 16px;
    margin: 4px 2px;
    cursor: pointer;
    border-radius: 4px;
    transform: translate(-460px,190px);
  }
  .switch-btn:hover{
    background-color: #1d571f;
  }
  #form-container {
    margin-top: 20px;
    border: aliceblue;
  }
  .form {
    display: flex;
    flex-direction: column;
    align-items: center;
    border: aliceblue;
  }
  .song-list {
    list-style-type: none;
    padding: 0;
  }
  .song-list li {
    cursor: pointer;
    color: rgb(175, 128, 70);
    text-decoration: solid;
  }
  .upload-input {
    margin-bottom: 10px;
    color:white;
  }
```

```
 .upload-button {
   background-color: #008CBA;
   color: white;
   padding: 10px 20px;
   text-align: center;
   text-decoration: none;
   display: inline-block;
   font-size: 16px;
   margin-top: 10px;
   cursor: pointer;
   border-radius: 4px;
 }
 #view-list ul {
   list-style-type: none;
   padding: 0;
 }
 #view-list li {
   margin-bottom: 5px;
   color: rgb(175, 128, 70);
   text-decoration: solid;
 }
 h2{
   color: antiquewhite;
 }
```

Addsong.html

```
<div class="container">
  <form id="add-song" class="form" #form="ngForm" (ngSubmit)="onSubmit(form)"
enctype="multipart/form-data">
     <h2>MP3 Songs Player</h2>
     <input type="file" class="upload-input" accept=".mp3" name="songlink"
id="songlink" (change)="onFileSelected($event)">
     <button class="upload-button" type="submit">Upload</button>
  </form>
</div>
```

Addsong.ts

```
import { Component } from '@angular/core';
import { Router } from '@angular/router';
import { NodeutilityService } from 'src/app/nodeutility.service';
@Component({
  selector: 'app-addsong',
  templateUrl: './addsong.component.html',
  styleUrls: ['./addsong.component.css']
})
export class AddsongComponent {
  constructor(private util:NodeutilityService,private router:Router) { }
  msg: string = '';
  selectedFile: File | undefined; // Define selectedFile property
  onFileSelected(event: any) {
    const file: File = event.target.files[0];
    this.selectedFile = file;
  }
  onSubmit(form: any) {
    // Ensure that the restaurantimage field is correctly populated
    if (!this.validateForm()) {
      return; // Stop form submission if validation fails
    }
    if (!this.selectedFile) {
      console.error('No file selected.');
      this.msg = 'Upload a song.';
    }

    const formData = new FormData();
    if (this.selectedFile) { // Add null check
      formData.append('songlink', this.selectedFile, this.selectedFile.name);
    console.log(formData); // Log the FormData object to verify file attachment
    this.util.up(formData).subscribe((data) => {
      if (data.status) {
        this.msg = data.message;
        alert(this.msg);
      } else {
        this.msg = data.message;
        alert(this.msg);
      }
    });
}
validateForm(): boolean {
  const fileInput = <HTMLInputElement>document.querySelector('#songlink');
  let song: File | null = null;
  if (fileInput.files && fileInput.files.length > 0) {
    song = fileInput.files[0]; // Get the first file selected (assuming single file upload)
  } else {
    console.error('No file selected.');
    this.msg = 'Upload a song.';
    return false;
  }
   if ( !song || !confirm) {
    this.msg = 'Please select a song.';
    return false;
  }
  return true; // Form is valid
}
}
```

Ahome.css

```
body {
    background-color:#222222;
    font-family: Arial, sans-serif;
```

```
margin: 0;
    padding: 0;
}
.container {
    display: flex;
    justify-content: center;
    align-items: flex-start;
    padding-top: 20px;
}
.sidebar {
    display: flex;
  flex-direction: column;
  justify-content: flex-start;
  align-items: center;
  background-color: black;
  height: 100vh;
    transform: translate(-541px,-19px);
}
.sidebar_items {
    margin-bottom: 10px;
}
.sidebar_items a {
    color: #ffffff;
    text-decoration: none;
    display: block;
    padding: 10px;
}
.sidebar_items a:hover {
    background-color: #444444;
}
.info {
    padding: 20px;
    position: absolute; /* Add position absolute */
```

```
right: 20px; /* Adjust the right distance */
    top: 20px; /* Adjust the top distance */
    width: calc(100% - 300px); /* Adjust the width based on sidebar width */
    background-color: #222222;
    border-radius: 8px;
    box-shadow: 0 0 10px #222222;
}
.alw{
    transform: translateX(-140px,150px);
}
h1 {
    text-align: center;
    color: #928c8c;
    font-family: 'Comic Sans MS', sans-serif;
  }
  h2{
    color: #c1b6b6;
    font-family: 'Comic Sans MS', sans-serif;
  }
  h4 {
    color: #e07dcb;
    font-family: 'Comic Sans MS', sans-serif;
  }
  p {
    line-height: 1.6;
    font-family: 'Comic Sans MS', sans-serif;
    color:beige
  }
```

Ahome.html

```
<body style="background-color: #222222;">
    <div class="container row">
```

```
  <div class="sidebar col-2">
        <div class="sidebar_items">
          <br>
          <img
            width="150"
            class="m-3"
            src="assets/logo.png"
            alt="spotify light logo"
          />
        </div>
        <br>
        <div class="sidebar_items"(click)="navigateToAprof()"><a >My Profile</a></div>
        <div class="sidebar_items"(click)="navigateToUprof()"><a >User
Profiles</a></div>
        <div class="sidebar_items" (click)="navigateToAddSong()"><a
(click)="navigateToAddSong()">Add songs</a></div>
        <div class="sidebar_items" (click)="navigateToViewReviews()"><a>View
Reviews</a></div>

        <div class="sidebar_items"(click)="navigateToRemove()"><a >Remove
User</a></div>
      </div>
      <div class="info col-14" name="in" >
        <app-top-nav></app-top-nav>
        <div *ngIf="!(pageVisibilityService.isAprofVisible | async)
                && !(pageVisibilityService.isUprofVisible | async)
                && !(pageVisibilityService.isAddSongVisible | async)
                && !(pageVisibilityService.isViewReviewsVisible | async)
                && !(pageVisibilityService.isviewlistVisible | async)
                && !(pageVisibilityService.isremoveVisible | async)
                && !(pageVisibilityService.isachangeVisible | async)">
        <div class="alw" style="margin-top: 150px;">
```

```
    <h1>Welcome to Symphony!</h1>
          <br>
          <br>
          <h2>At Symphony, we believe that music is more than just notes and rhythms; it's a universal language that connects souls, transcends boundaries, and fills the world with harmony.</h2>
          <h2>Let's make beautiful music together!</h2>
        <h1 style="color: #e07dcb;
        font-family: 'Comic Sans MS', sans-serif;transform:translate(150px);">Symphony Team</h1>
        </div>
        </div>
         <div *ngIf="pageVisibilityService.isAprofVisible | async">
           <app-aprof></app-aprof>
         </div>
         <div *ngIf="pageVisibilityService.isUprofVisible | async">
           <app-uprof></app-uprof>
         </div>
         <div *ngIf="pageVisibilityService.isAddSongVisible | async">
           <app-addsong></app-addsong>
         </div>
         <div *ngIf="pageVisibilityService.isViewReviewsVisible | async">
           <app-aview></app-aview>
           </div>
         <div *ngIf="pageVisibilityService.isviewlistVisible | async">
             <app-viewlist></app-viewlist>
         </div>
         <div *ngIf="pageVisibilityService.isremoveVisible | async" class="remove-container">
               <app-remove></app-remove>
               </div>
         <div *ngIf="pageVisibilityService.isachangeVisible | async">
                <app-achange></app-achange>
```

```
     </div>
   </div>
 </body>
```

Ahome.ts

```
import { Component } from '@angular/core';
import { PageVisibilityService } from 'src/app/services/PageVisibilityService';
@Component({
  selector: 'app-ahome',
  templateUrl: './ahome.component.html',
  styleUrls: ['./ahome.component.css']
})
export class AhomeComponent {
  isAprofVisible!: boolean;
  constructor(public pageVisibilityService: PageVisibilityService) {}
  navigateToAprof() {
    this.pageVisibilityService.showAprof();
  }
  navigateToUprof() {
    this.pageVisibilityService.showUprof();
  }
  navigateToAddSong() {
    this.pageVisibilityService.showAddSong();
  }
  navigateToViewReviews() {
    this.pageVisibilityService.showViewReviews();
  }
  navigateToViewlist() {
    this.pageVisibilityService.showAlist();
  }
  navigateToRemove() {
    this.pageVisibilityService.showRemove();
  }
```

```
 navigateToAChange() {
   this.pageVisibilityService.showAChange();
  }
}
```

Alog.css

```
.login_container{
   display: flex;
   flex-direction: column;
   justify-content: center;
   align-items: center;
   height: 730px;
   gap: 10px;
  }
  label,h5{
   color: antiquewhite;
  }
  input[type="text"],input[type="password"] {
   padding: 10px;
   border: none;
   border: 1px solid rgb(45, 44, 44);
   border-radius: 4px;
   width: 400px;
   height: 32px;
   background-color: rgb(223, 183, 183);
  }
  input::placeholder {
   color: rgb(45, 44, 44);
  }
  hr{
  width: 100%;
  border: 1px solid black;
  }
  #remember_me{
```

```
margin-left: 4px;
 height: 20px;
 background-color: #1ed760;
 width: 15px;
 }
 .login_btn {
   background-color: #1ed760;
   color: black;
   padding: 14px;
   border-radius: 28px;
   width: 150px;
   border: none;
   font-weight: bold;
   cursor: pointer;
 }
 .signup_btn{
 background-color: white;
 color: gray;
 padding: 14px;
 border-radius: 28px;
 border: 2px solid grey;
 font-weight: bold;
 cursor: pointer;
 }
 .action_buttons{
   display: flex;
   flex-direction: row;
   justify-content: space-between;
   align-items: center;
   width: 412px;
 }
```

```
img{
   transform:translateY(-50px);
  }
  body{
   background-image: url('/../assets/alogin.jpg');
  }
```

Alog.html

```
<body>
   <div class="container">
      <form #form="ngForm" (ngSubmit)="onSubmit(form)">
       <div class="login_container">
          <img src="/../assets/logo.png" width="300"/>
          <div>
             <label><strong>Email address or username</strong></label><br />
             <input
               ngModel #username="ngModel"
               type="text"
               placeholder="Email address or username"
               name="username"
            />
          </div>
          <div>
             <label><strong>Password</strong></label><br />
             <input


               type="password"
               placeholder="password"
               ngModel #pw="ngModel"
               name="pw"
            />
            <br />
          </div>
          <div class="action_buttons">
```

```
                [style.display]="'flex'"
                [style.justifycontent]="'flex-start'"
                [style.alignItems]="'stretch'"
                [style.gap]="'5px'"
              >
                <input type="checkbox" id="remember_me"/>
                <label for="remember_me">
                  <span><strong>Remember me</strong></span>
                </label>
              </span>
              <button class="login_btn">LOG IN</button>
            </div>
            <hr [style.width]="'30%'"/>
            <h5><strong>Don't have an account ?</strong></h5>
            <button              class="signup_btn"              [style.width]="'30%'"
onclick="window.location.href='asign';">Sign Up For Symphony</button>
          </div>
        </form>
      </div>
      </body>
```

Alog.ts

```
import { Component } from '@angular/core';
import { Router } from '@angular/router';
import { NodeutilityService } from 'src/app/nodeutility.service';
@Component({
  selector: 'app-alog',
  templateUrl: './alog.component.html',
  styleUrls: ['./alog.component.css']
})
export class AlogComponent {
  msg:string="";
  user1:string | null="";
  constructor(private util:NodeutilityService,private router:Router){
```

```
 onSubmit(form: any) {
  this.util.insert3(form.value.username, form.value.pw).subscribe((data) => {
      if (data.status){
        localStorage.setItem("user1",form.value.username);
        this.msg = data.message;
        alert(this.msg);
        this.router.navigate(['/ahome']);
      }
      else{
        this.msg = data.message;
        alert(this.msg);
      }
    });
 }
}
```

Aprof.css

```
.container {
  max-width: 500px;
  margin: 0 auto;
  padding: 20px;
  border: 1px solid #ccc;
  border-radius: 5px;
  margin-top:120px;
 }
  .search-container {
  margin-bottom: 20px;
 }
 input[type="text"] {
  width: 70%;
  padding: 10px;
  margin-right: 10px;
  border: 1px solid #ccc;
  border-radius: 5px;
 .search-button {
  padding: 10px 20px;
  background-color: #007bff;
  color: #fff;
  border: none;
  border-radius: 5px;
  cursor: pointer;
 }
 .search-button:hover {
  background-color: #0056b3;
 }
 .result-container {
  background-color: #f5f5f5;
  padding: 20px;
  border-radius: 5px;
 }
 .result-container h3 {
  margin-top: 0;
  color: #333;
 }
 .result-container p {
  margin-bottom: 10px;
 }
 h2{
  color:aquamarine;
  text-align: center;
  transform: translateY(100px);
 }
```

Aprof.html

```
<h2>Your Profile</h2>
<div class="container">
    <form #form="ngForm" (ngSubmit)="search(form)">
    <div class="search-container">
      <input         type="text"         name="searchName"         [(ngModel)]="searchName"
placeholder="Enter name">
      <button class="search-button" >Search</button>
    </div>
    <div class="result-container" *ngIf="selectedPerson">
      <h3>Details of {{ selectedPerson.name }}</h3>
      <p>Username: {{ selectedPerson.name }}</p>
      <p>Phone Number: {{ selectedPerson.phno }}</p>
      <p>Password: {{ selectedPerson.pw }}</p>
    </div>
</form>
  </div>
```

Aprof.ts

```
import { Component } from '@angular/core';
import { NodeutilityService } from 'src/app/nodeutility.service';
@Component({
  selector: 'app-aprof',
  templateUrl: './aprof.component.html',
  styleUrls: ['./aprof.component.css']
})
export class AprofComponent {
  searchName: string = '';
  selectedPerson: any;
  constructor(private util: NodeutilityService) {}
  search(form:any) {
    if (this.searchName.trim() !== '') {
      this.util.find2(this.searchName).subscribe((data) => {
        if (data.status) {
          this.selectedPerson = data.person;
          console.log("success");
        } else {
          this.selectedPerson = null;
      });
    }
  }
}
```

Assign.css

```
.login_container{
    display: flex;
    flex-direction: column;
    justify-content: center;
    align-items: center;
    height: 730px;
    gap: 10px;
    transform:translateX(160px);
  }
  input[type="text"],input[type="password"] {
    padding: 10px;
    border: none;
    border: 1px solid rgb(45, 44, 44);
    border-radius: 4px;
    width: 400px;
    height: 35p2;
  }
  input::placeholder {
    color: rgb(45, 44, 44);
  }
  hr{
  width: 100%;
  border: 1px solid black;
  }
  #remember_me{
  margin-left: 4px;
  height: 20px;
  background-color: #1ed760;
}
.login_btn {
  background-color: #1ed760;
  color: black;
  padding: 14px;
  border-radius: 28px;
  width: 150px;
  border: none;
  font-weight: bold;
  cursor: pointer;
}
.signup_btn{
background-color: white;
color: gray;
padding: 14px;
border-radius: 28px;
border: 2px solid grey;
font-weight: bold;
cursor: pointer;
}
.action_buttons{
  display: flex;
  flex-direction: row;
  justify-content: space-between;
  align-items: center;
  width: 412px;
}
img{
  transform:translateY(-30px);
}
body{
  background-image: url('/../assets/asign.jpg');
}
```

Assign.html

```
<body>
    <div class="container">
        <form [formGroup]="userForm" #form="ngForm" (ngSubmit)="onSubmit(form)">
         <div class="login_container">
            <img src="/../assets/logo.png" width="300"/>
            <div>
                <label><strong>Email address or username</strong></label><br />
                <input
                  [formControl]="usernameFormControl"
                  type="text"
                  placeholder="Email address or username"
                  name="username"
                  />
            </div>
            <div>
               <label><strong>Phone Number</strong></label><br />
               <input
                 [formControl]="phnoFormControl"
                 type="text"
                 placeholder="Phone number"
                 name="phno"/>
           </div>
            <div>
                <label><strong>Password</strong></label><br />
                <input
                  [formControl]="passwordFormControl"
                  type="password"
                  placeholder="password"
                  name="pw"/>
               <br />
            </div>
            <div class="action_buttons">
```

```
                [style.display]="'flex'"
                [style.justifycontent]="'flex-start'"
                [style.alignItems]="'stretch'"
                [style.gap]="'5px'">
                <input type="checkbox" id="remember_me"/>
                <label for="remember_me">
                  <span><strong>Remember me</strong></span>
                </label>
              </span>
              <button class="login_btn">SIGN UP</button>
            </div>
            <hr [style.width]="'30%'"/>
            <h5><strong>Already have an account ?</strong></h5>
            <button class="signup_btn" [style.width]="'30%'"
onclick="window.location.href='alog';">Login</button>
          </div>
        </form>
      </div>
    </body>
```

Assign.ts

```
import { Component } from '@angular/core';
import { FormControl, FormGroup, Validators } from '@angular/forms';
import { Router } from '@angular/router';
import { NodeutilityService } from 'src/app/nodeutility.service';
@Component({
  selector: 'app-asign',
  templateUrl: './asign.component.html',
  styleUrls: ['./asign.component.css']
})
export class AsignComponent {
  public usernameFormControl = new FormControl(null,[Validators.required,
Validators.email]);
  public phnoFormControl = new
```

```
 FormControl(null,[Validators.minLength(10),Validators.maxLength(10)]);
  public passwordFormControl = new FormControl(null,[Validators.minLength(6)]);
  public userForm! : FormGroup;
  constructor(private util:NodeutilityService,private router:Router){}
  msg:string='';
  username:string='';
  phno:string='';
  pw:string='';
  onSubmit(form: any) {
    if (this.userForm.valid) {
      const username = this.userForm.value.username;
      const phno = this.userForm.value.phno;
      const password = this.userForm.value.password;
      this.util.insert(username, phno, password).subscribe((data) => {
        if (data.status) {
          this.msg = data.message;
          alert("Registration Succesfull");
        }
      });
      this.router.navigate(['/alog']);
    } else {
      this.userForm.markAllAsTouched();
      alert("Please enter valid values"); // Mark the form controls as touched to display
validation errors
    }
  }
  ngOnInit(): void {
    this.userForm = new FormGroup({
      username: this.usernameFormControl,
      phno:this.phnoFormControl,
      password: this. passwordFormControl ,
    });
  }
```

```
    console.log(this.userForm.value);
  }
}
```

Home1.css

```
html {
    scroll-behavior: smooth;
  }
  h1 {
    font-family:Comic Sans MS;
    font-size:400%;
    color:rgb(11, 11, 11);
  }
  h2 {
    font-family:Comic Sans MS;
    color:rgb(7, 7, 7);
  }
  p {
    font-family:Comic Sans MS;
    color:rgb(5, 5, 5);
  }
  #section1 {
    height:800px;
    color:rgb(17, 16, 16);
    background-image: url('/../assets/home.jpg');
  }
  #section2 {
    height:800px;
    color:rgb(2, 2, 2);
    background-color:black;
    border: 5px outset gray;
    top:100px;left:70px;
  }
  #section3 {
```

```
height:800px;
    color:white;
    background-color:black;
    border: 5px outset gray;
  }

  #b1 {
    top:78%;
    left:40%;
    width:80px;
    height:40px;
    position: absolute;
    background: rgb(212, 120, 14);
    font-size:20px;
    font-family:Comic Sans MS;
    color:black;
  }

  #Language {
    top:5%;
    left:83%;
    width:100px;
    height:40px;
    position: absolute;
    background: rgb(187, 185, 185);
    font-size:20px;
    font-family:Comic Sans MS;
  }

  #b2 {
    top:78%;
    left:50%;
    width:80px;
    position: absolute;
    background: rgb(212, 120, 14);
    font-size:20px;
    font-family:Comic Sans MS;
    color:black;
  }

  body {
    color:rgb(14, 14, 14);
    background-color:black;
  }

  #txt {
    top:79%;
    left:35%;
    width:300px;
    height:39px;
    position: absolute;
    background: rgb(8, 8, 8);
    font-size:20px;
    font-family:Comic Sans MS;
  }



  #lftp {
    top:40%;
    left:10%;
    position: relative;
    font-size: 34px;
  }

  #rytp {
    top:40%;
```

```
position: relative;
    font-size: 34px;
  }

  #lfth {
     top:35%;
     left:10%;
     position: relative;
     font-size: 44px;
  }

  #ryth {
    top:35%;
    left:50%;
    position: relative;
    font-size: 44px;
  }

  nav {
    text-align:center;
  }

  nav li {
    display:inline;
    font-family:Comic Sans MS;
    font-size:28px;
    padding-right:40px;
    transform: translate(-50px);
  }

   .ab {
    font-size: 40px;
    color:rgb(5, 5, 62);
  transform:translate(900px,-10px);
 }
```

Home1.html

```
<!DOCTYPE html>
<html>
<head>
  <title>Symphony</title>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">
 </head>
  <body>
  <div class="main" id="section1">
   <h1 style="font-size: 60px;color:rgb(90, 44, 4);transform:translate(50px,55px);">SYMPHONY</h1>
   <div class="ab"><a (click)="getabout()">About Us</a></div>
   <select id="Language" style="transform:translateY(49px);">
    <option value="eng">English</option>
    <option value="es">Espanol</option>
    <option value="fr">French</option>
    <option value="gr">German</option>
   </select>
   <h2 style="text-align: center;font-family:Segoe Print;font-size:44px;transform: translateX(-40px);"><strong>Unlimited Music Awaits!</strong></h2>
   <p style="text-align: center;font-family:Segoe Print;font-size:34px;transform: translateX(-40px);"><strong>Listen anywhere.Cancel at any time.</strong></p>
   <p style="text-align: center;font-family:Segoe Print;font-size:34px;transform: translateX(-40px);"><strong>Want to listen? Let's Go.</strong></p>

   <button id="b1" style="transform:translate(5px,-30px);" onclick="window.location.href='alog';">Admin</button>
   <button id="b2" style="transform:translate(5px,-30px);" onclick="window.location.href='ulog';">User</button>
  </div>
```

Playsong.css

```
    display: flex;
    align-items: center;
    justify-content: center;
    background-color: #e9e2e3;
    padding: 20px;
    border-radius: 10px;
    box-shadow: 0 4px 6px rgba(0, 0, 0, 0.1);
  }

  .player-container p {
    font-size: 24px;
    margin-right: 20px;
    color: #662027;
  }

  .player-container button {
    font-size: 18px;
    padding: 12px 24px;
    background-color: #c24451;
    color: #fff;
    border: none;
    border-radius: 5px;
    cursor: pointer;
    transition: background-color 0.3s;
  }

  .player-container button:hover {
    background-color: #c82333;
  }
```

Playsong.html

```
<div class="player-container">
    <button (click)="play()">Play</button>
    <button (click)="skip()">Skip</button>
  </div>
```

Playsong.ts

```
import { Component } from '@angular/core';
import { PlaylistService } from 'src/app/services/playlist.service';

@Component({
  selector: 'app-play-song',
  templateUrl: './play-song.component.html',
  styleUrls: ['./play-song.component.css']
})
export class PlaySongComponent {
  currentSong!: { name: string; url: string };
  isPlaying: boolean = false;
  audioPlayer: HTMLAudioElement = new Audio();

  constructor(private playlistService: PlaylistService) {}

  ngOnInit(): void {
    this.currentSong = this.playlistService.playlist[0] || { name: '', url: '' }; // Initialize with
the first song or empty object
    this.audioPlayer.src = this.currentSong.url;
  }

  play() {
    this.audioPlayer.play();
    this.isPlaying = true;
  }

  pause() {
    this.audioPlayer.pause();
    this.isPlaying = false;
  }
  skip() {
    const currentIndex = this.playlistService.playlist.indexOf(this.currentSong);
    const nextIndex = (currentIndex + 1) % this.playlistService.playlist.length;
    this.currentSong = this.playlistService.playlist[nextIndex];
    this.audioPlayer.src = this.currentSong.url;
    this.play();
  }
}
```

Playlist.css

```
body {
  font-family: sans-serif; /* Choose a nice font family */
  margin: 0;
  padding: 0;
  background-color: #222222;
  margin-top:174px;
  background-size: auto;/* Light background */
}

.scrollable-container {
  overflow-y: auto; /* Enable scrolling for long playlists */
  max-height: 500px; /* Adjust height as needed */
  padding: 20px;
 /* Light border */
  border-radius: 5px; /* Rounded corners */
  margin: 0 auto; /* Center content horizontally */
}

/* Song List Styles */
.container,
.playlist-section {
  display: inline-block; /* Display both sections side-by-side */
  width: 45%; /* Adjust width as needed */
  padding: 15px;
  border-radius: 5px;
  background-color: #abaaaa; /* White background for content */
}

h2 {
  margin-bottom: 10px;
  text-align: center; /* Center headings */
}

ul {
  list-style: none;
  padding: 0;
  margin: 0;
}

li {
  display: flex; /* Arrange list items horizontally */
  justify-content: space-between; /* Space out content within list items */
  align-items: center; /* Vertically align content */
  margin-bottom: 5px;
}

/* Song Name Styling */
.song-name {
  flex: 1; /* Allow song name to fill available space */
  font-weight: bold;
  color: #333; /* Darker color for song names */
}

/* Button Styles */
button {
  background-color: #4CAF50; /* Green color for buttons */
  padding: 5px 10px;
  border: none;
  border-radius: 3px;
  cursor: pointer; /* Indicate clickable behavior */
}

button:hover {
  background-color: #3e8e41; /* Darker green on hover */
}

/* Play Song Component Styling */
app-play-song {
  display: block;
  margin-top: 20px;
  text-align: center; /* Center play song component */
}
```

Playlist.html

```
<body>
    <div class="scrollable-container">
    <div class="container ">
      <h2>Songs</h2>
      <ul>
        <li *ngFor="let song of playlistService.songs">
          {{ song.name }} -
          <button (click)="playlistService.addToPlaylist(song)">Add</button>
        </li>
      </ul>
    </div>

    <div class="playlist-section ">
      <h2>Playlist</h2>
      <ul>
        <li *ngFor="let song of playlistService.playlist; let i = index">
```

```
          <button (click)="playlistService.removeFromPlaylist(i)">Remove</button>
        </li>
      </ul>
    </div>

    <app-play-song></app-play-song>
  </div>
  </body>
```

Playlist.ts

```
import { PlaylistService } from 'src/app/services/playlist.service';

@Component({
  selector: 'app-playlist',
  templateUrl: './playlist.component.html',
  styleUrls: ['./playlist.component.css']
})
export class PlaylistComponent implements OnInit{
  constructor(public playlistService: PlaylistService) {}
  playSong(song: { name: string; url: string }) {
    this.playlistService.addToPlaylist(song);
  }

  ngOnInit(): void {}

}
```

Search.css

```
.search-container {
    padding: 20px;
      margin-top: 74px;
  }

  input[type="text"] {
    padding: 10px;
  border-radius: 8px;
  border: 2px solid #ddd;
  box-sizing: border-box;
  background-color: rgb(55, 52, 52);
  color: #ddd;
}

input[type="text"]:focus {
  outline: none;
  border-color: #4CAF50;
}

.results-container {
  margin-top: 20px;
}

.music-item {
  background-color: #747373;
  padding: 10px;
  border-radius: 8px;
  margin-bottom: 10px;
  box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1);
}

.music-item p {
  margin: 0;
}

.music-item button {
  padding: 8px 16px;
  background-color: #17601a;
  color: white;
  border: none;
  cursor: pointer;
  transition: background-color 0.3s;
  margin-left: 10px;
}

.music-item button:hover {
  background-color: #45a049;
}

.no-results {
  margin-top: 20px;
  color: #FF0000;
}
input ::placeholder{
  color: #f1f1f1;
}

h3{
  color: aquamarine;
}
```

Search.html

```
<div class="search-container">
  <input type="text" [(ngModel)]="searchQuery" (input)="search()" placeholder="Search">

  <div *ngIf="searchResults.length > 0" class="results-container">
    <h3>Search Results:</h3>
    <ul>
      <li *ngFor="let result of searchResults" class="music-item">
        <p><strong>Name: {{ result.name }}, Singer: {{ result.singer }}, Album: {{ result.album }}</strong></p>
        <button class="play-button" (click)="playMusic(result)">Play</button>
        <button class="pause-button" (click)="pauseMusic()">Pause</button>
    </ul>
  </div>

  <div *ngIf="searchResults.length === 0" class="no-results">
    <p>No matching music found.</p>
  </div>
 </div>
```

Search.ts

```
export class SearchComponent {
  searchQuery: string = '';
  searchResults: any[] = [];
  currentMusic: any = null;
  audioPlayer: any = new Audio();
  isPaused: boolean = false;

  constructor(private musicService: MusicService, private pageVisibilityService:
PageVisibilityService) { }

  search() {
    this.searchResults = this.musicService.searchMusic(this.searchQuery);
  }

  playMusic(music: any) {
    this.currentMusic = music;
    this.audioPlayer.src = this.currentMusic.src;
    if (!this.isPaused) {
      this.audioPlayer.play();
    }
    this.isPaused = false;
  }

  pauseMusic() {
    if (this.audioPlayer.paused) {
      this.isPaused = false;
    } else {
      this.audioPlayer.pause();
      this.isPaused = true;
    }
  }
}
```

Uchange.css

```
* {
    margin: 0;
    padding: 0;
    box-sizing: border-box;
    font-family: "segoe ui", verdana, helvetica, arial, sans-serif;
    font-size: 16px;
    transition: all 500ms ease; }

  body {
    -webkit-font-smoothing: antialiased;
    -moz-osx-font-smoothing: grayscale;
    text-rendering: optimizeLegibility;
     }

  .row {
    background-color: rgba(20, 120, 200, 0.6);
    color: #fff;
    text-align: center;
    padding: 2em 2em 0.5em;
    width: 90%;
    margin: 2em auto;
    margin-top: 144px;

    border-radius: 5px; }
    .row h1 {
```

```
.row .form-group {
 margin: 0.5em 0; }
 .row .form-group label {
   display: block;
   color: #fff;
   text-align: left;
   font-weight: 600; }
 .row .form-group input, .row .form-group button {
   display: block;
   padding: 0.5em 0;
   width: 100%;
   margin-top: 1em;
   margin-bottom: 0.5em;
   background-color: inherit;
   border: none;
   border-bottom: 1px solid #211f1f;
   color: #eee; }
   .row .form-group input:focus, .row .form-group button:focus {
     background-color: #fff;
     color: #000;
     border: none;
     padding: 1em 0.5em; animation: pulse 1s infinite ease;}

 .row .form-group .submit {
   border: 1px solid #2c2b2b;
   border-radius: 5px;
   outline: none;
   -moz-user-select: none;
   user-select: none;
   color: #282222;
   font-weight: 800;
   cursor: pointer;
   margin-top: 2em;
```

```
      .row .form-group button:hover, .row .form-group button:focus {
        background-color: #fff; }
      .row .form-group button.is-loading::after {
        animation: spinner 500ms infinite linear;
        content: "";
        position: absolute;
        margin-left: 2em;
        border: 2px solid #000;
        border-radius: 100%;
        border-right-color: transparent;
        border-left-color: transparent;
        height: 1em;
        width: 4%; }
  .row .footer h5 {
    margin-top: 1em; }
  .row .footer p {
    margin-top: 2em; }
    .row .footer p .symbols {
      color: #444; }
  .row .footer a {
    color: inherit;
    text-decoration: none; }

.information-text {
  color: #ddd; }

@media screen and (max-width: 320px) {
  .row {
    padding-left: 1em;
    padding-right: 1em; }
    .row h1 {
      font-size: 1.5em !important; } }
@media screen and (min-width: 900px) {
    width: 50%; } }
```

uchange.html

```
<form #form="ngForm" (ngSubmit)="onSubmit(form)">
    <div class="row">
        <h1>Change Password</h1>
        <h6 class="information-text">Enter your registered email to reset your password.</h6>
        <div class="form-group">
            <input type="email" name="email" id="user_email" ngModel #email="ngModel">
            <p><label for="username">Email</label></p>
            <input type="text" name="oldpass" id="oldpass" ngModel #oldpass="ngModel">
            <p><label for="oldpass">Old Password</label></p>
            <input type="text" name="newpass" id="newpass" ngModel #newpass="ngModel">
            <p><label for="newpass">New Password</label></p>
            <input type="submit" class="submit" value="Change Password">
        </div>
    </div>
</form>
```

Uchange.ts

```
@Component({
  selector: 'app-uchange',
  templateUrl: './uchange.component.html',
  styleUrls: ['./uchange.component.css']
})
export class UchangeComponent {
  ngOnInit(): void{};
  constructor(private util: NodeutilityService, private router: Router) {}
  msg: string = '';
  onSubmit(form: any) {
```

```
  this.util.update(form.value.email,                                              form.value.oldpass,
form.value.newpass).subscribe((data) => {
      if (data.status) {
        this.msg = data.message;
        alert("Updated");
        console.log(this.msg);

      }
      else{
        this.msg = data.message;
      }
      });
  }
```

Server code

```
const express = require('express');
const app = express();
const multer = require('multer');
const path = require('path');
const bodyParser=require('body-parser');
const cors = require('cors');
const fs= require('fs');
app.use(cors());
app.use(express.urlencoded({ extended: true }));
// MongoDB connection setup
const { MongoClient } = require('mongodb');
const url = 'mongodb://localhost:27017';
const client = new MongoClient(url);
const uploadDir = path.join(__dirname, 'symphony','src','assets','audio');

const urlencodedParser = bodyParser.urlencoded({ extended: false });
app.use(urlencodedParser);
// Connect to the MongoDB server
```

```
    try {
        await client.connect();
        console.log('Connected to MongoDB server');
    } catch (err) {
        console.error('Error connecting to MongoDB server', err);
        process.exit(1);
    }
}

const storage = multer.diskStorage({
  destination: function (req, file, cb) {
      if (!fs.existsSync(uploadDir)) {
          fs.mkdirSync(uploadDir, { recursive: true });
      }
      cb(null, uploadDir);
  },
  filename: function (req, file, cb) {
      cb(null, file.originalname);
  }
});

const upload = multer({ storage });

app.use(function (err, req, res, next) {
  if (err instanceof multer.MulterError) {
      // Multer error occurred
      res.status(400).send('File upload error: ' + err.message);
  } else {
      // Other errors
      res.status(500).send('Internal server error: ' + err.message);
  }
});
app.post('/up', upload.single('songlink'), async (req, res) => {
```

```
try {
    console.log("Request Query Parameters:", req.body);
    res.setHeader('content-type','application/json');
    res.setHeader("Access-Control-Allow-Origin","*");
    const db = client.db('Music1');
    const collection=db.collection('song');
    const songlink  = req.songlink;
    const uploadDir = path.join(__dirname, 'symphony', 'src', 'assets', 'audio');
    const songPath = path.join(uploadDir, req.file.filename);
    const existingUser = await collection.findOne({ songlink:songlink });
    console.log(existingUser);
    if (existingUser) {
      res.json({ status: false, message: 'Song already exists.' });
      return;
    }
    const result = await collection.insertOne({

      songlink: songPath, // Store image path in the database

    });
    res.json({ status: true, message: 'Added Successfully' });
  } catch (err) {
    console.error('Error:', err);
    res.json({ status: false, message: 'Insert Failed' });
  }
});

app.get('/insert', async function (req, res){
    try {
    res.setHeader('content-type','application/json')
    res.setHeader("Access-Control-Allow-Origin","*");
        const db = client.db('Music1');
        const result = await collection.insertOne(req.query);
        data={ status:true,message: "Inserted Successfully" };
    res.json(data);
    } catch (err) {
        console.error('Error ', err);
        data={ status:false,message: "Insert Failed" };
    res.json(data);
    }
});

app.get('/insert1', async function (req, res){
  try {
  res.setHeader('content-type','application/json')
  res.setHeader("Access-Control-Allow-Origin","*");
      const db = client.db('Music1');
      const collection=db.collection('usign');
      const result = await collection.insertOne(req.query);
      data={ status:true,message: "Inserted Successfully" };
  res.json(data);
  } catch (err) {
      console.error('Error ', err);
      data={ status:false,message: "Insert Failed" };
  res.json(data);
  }
});

app.get('/insert2', async function (req, res){
  try {
    console.log("Request Query Parameters:", req.query);
  res.setHeader('content-type','application/json')
  res.setHeader("Access-Control-Allow-Origin","*");
  const db = client.db('Music1');
  const collection=db.collection('usign');
  const user = await collection.findOne(doc);
  console.log(user);
  if(user!=null){
    data={ status:true,message: "Login Successful" };
    res.json(data);
  }

  }catch (err) {
      console.error('Error ', err);
      data={ status:false,message: "Login Failed" };
  res.json(data);
  }
});

app.get('/insert3', async function (req, res){
  try {
    console.log("Request Query Parameters:", req.query);
  res.setHeader('content-type','application/json')
  res.setHeader("Access-Control-Allow-Origin","*");
  const db = client.db('Music1');
  const collection=db.collection('asign');
  var doc={name:req.query.username,pw:req.query.pw};
  const user = await collection.findOne(doc);
  console.log(user);
  if(user!=null){
    data={ status:true,message: "Login Successful" };
    res.json(data);

  }
  }catch (err) {
      console.error('Error ', err);
      data={ status:false,message: "Login Failed" };
```

```
app.get('/insert4', async function (req, res) {
    try {
      console.log("Request Query Parameters:", req.query);
      res.setHeader('content-type', 'application/json');
      res.setHeader("Access-Control-Allow-Origin", "*");
      const db = client.db('Music1');
      const collection = db.collection('review');
      const                                   doc                                   =
{email:req.query.email,reviewText:req.query.reviewText,rating:req.query.rating };
      const result = await collection.insertOne(doc);
      console.log(result);
      if (result.insertedCount > 0) {
        res.json({ status: true, message: "Review inserted successfully" });
      } else {
        res.json({ status: false, message: "Failed to insert review" });
      }
    } catch (err) {
      console.error('Error ', err);
      res.json({ status: false, message: "An error occurred while processing your request" });
    }
  });

  app.get('/update', async function (req, res){
    try {
      console.log("Request Query Parameters:", req.query);
    res.setHeader('content-type','application/json')
    res.setHeader("Access-Control-Allow-Origin","*");
    const db = client.db('Music1');
    const collection=db.collection('usign');
    const email=req.query.email;
    const newpass=req.query.newpass;
    const oldpass=req.query.oldpass;
    const result = await collection.updateOne({name:email,pw:
oldpass},{$set:{pw:newpass}});

    if (result.modifiedCount > 0)
      data = { status: true, message: "Updated Successfully", noOfDoc: result.modifiedCount
};
    else
      data = { status: false, message: "No data found or old password is incorrect", noOfDoc:
result.modifiedCount };
    res.json(data);
    } catch (err) {
        console.error('Error ', err);
        data={ status:false,message: "update action failed" };
    res.json(data);
    }
  });
    app.get('/delete', async function (req, res){
    try {
      console.log("Request Query Parameters for delete:", req.query);
    res.setHeader('content-type','application/json')
    res.setHeader("Access-Control-Allow-Origin","*");
    const db = client.db('Music1');
    const collection=db.collection('usign');
    const name=req.query.name;


    const result = await collection.deleteOne({name:name});

    if (result.deletedCount > 0)
      data = { status: true, message: "Deleted Successfully", noOfDoc: result.deletedCount
};
    else
      data = { status: false, message: "No data found ", noOfDoc: result.deletedCount };
   res.json(data);
    } catch (err) {
        console.error('Error ', err);
        data={ status:false,message: "update action failed" };
    res.json(data);
    }
  });

  app.get('/findAll', async function (req, res){
    try {
    res.setHeader('content-type','application/json')
    res.setHeader("Access-Control-Allow-Origin","*");
    const db = client.db('Music1');
    const collection=db.collection('review');

    const result = await collection.find({},{_id:0,email:1,reviewText:1,rating:1}).toArray();
    data = { status: true, message: "Listed Successfully", list:result };
    res.json(data);
    } catch (err) {
        console.error('Error ', err);
        data={ status:false,message: "Action failed" };
    res.json(data);
    }
  });

  app.get('/find1', async function (req, res) {
    try {
      res.setHeader('content-type', 'application/json');
      res.setHeader("Access-Control-Allow-Origin", "*");
      const db = client.db('Music1');
      const collection = db.collection('usign');
      const searchName = req.query.searchName;
```

```
      // Use findOne instead of find to retrieve only one document
      const result = await collection.findOne({ name: searchName }, { _id: 0, name: 1, phno:
1, pw: 1 });

      if (result) {
        const data = { status: true, message: "Listed Successfully", person: result }; // Return
the matching document
        res.json(data);
      } else {
        const data = { status: false, message: "No matching record found" };
        res.json(data);
      }
    } catch (err) {
      console.error('Error ', err);
      res.status(500).json({ status: false, message: "Action failed" });
    }
  });

  app.get('/find2', async function (req, res) {
    try {
      res.setHeader('content-type', 'application/json');
      res.setHeader("Access-Control-Allow-Origin", "*");
      const db = client.db('Music1');
      const collection = db.collection('asign');
      const searchName = req.query.searchName;

      // Use findOne instead of find to retrieve only one document
      const result = await collection.findOne({ name: searchName }, { _id: 0, name: 1, phno:
1, pw: 1 });

      if (result) {
        const data = { status: true, message: "Listed Successfully", person: result }; // Return
```

```
the matching document
      res.json(data);
    } else {
      const data = { status: false, message: "No matching record found" };
      res.json(data);
    }
  }
app.get('/checkUnameAvailability', async (req, res) => {
  try {
    res.setHeader('content-type', 'application/json');
    res.setHeader("Access-Control-Allow-Origin", "*");
    const db = client.db('Music1');
    const collection = db.collection('usign');
    const username = req.query.username;

    const data = await collection.findOne({ name: username });
    if (data) {
      res.json({ available: false });
    } else {
      res.json({ available: true });
    }
  } catch (error) {
    console.error(error);
    res.status(500).send('Error checking email availability.');
  }
// Start the server
app.listen(5000, () => {
    console.log('Server running at http://localhost:5000');
  connect(); // Connect to MongoDB when the server starts
});
```

REFERENCES

- https://auth0.com/blog/building-an-audio-player-app-with-angular-and-rxjs/
- https://auth0.com/blog/building-an-audio-player-app-with-angular-and-rxjs/
- https://pantherax.com/how-to-create-a-music-player-app-with-angular-and-soundcloud-api/
- https://waltercode.com/creating-custom-music-player-desktop-app-with-angular-and-electronjs/