HARMORA(Node)

Introduction

Welcome to Harmora, a simple and efficient music streaming application built with Node.js. This platform is designed for music lovers who want a seamless experience while discovering, organizing, and playing their favorite songs.

Unlike traditional web applications that rely on heavy front-end frameworks, Harmora uses Node.js with HTML and CSS to create a lightweight and dynamic music streaming experience.

With Harmora, users can browse songs, create playlists, search for music by artist or genre, and listen to their favorite tracks anytime—all while enjoying a clean and user-friendly interface

Scenario-Based Introduction

It's an early morning, and you're preparing for a busy day at the office. You open Harmora on your phone, searching for a playlist that helps you focus and sets the right tone for productivity. Within moments, you find a curated list of ambient, instrumental tracks that allow you to concentrate, keeping your mind clear and sharp as you dive into work.

Later, during a critical meeting, you need to stay calm and composed. Harmora helps by providing a collection of soothing, classical music, ensuring you remain collected and poised. The music works as a subtle backdrop, enhancing your ability to engage in the conversation without distraction.

Whether it's to enhance focus, maintain composure, or elevate your work environment, Harmora ensures that you have the right music to optimize every professional moment.

Target Audience:-

Music Streaming is designed for a diverse audience, including:

- **Music Enthusiasts -** Anyone who enjoys discovering and listening to music anyone who enjoys discovering and listening to music.
- **Curators -** Users who like organizing personalized playlists.
- **Casual Listeners -** People looking for an easy-to-use streaming platform.

Project Goals and Objectives:-

The primary goal of Music Streaming is to provide a seamless platform for music enthusiasts, enjoying, and sharing diverse musical experiences. Our objectives include:

User-Friendly Interface: Develop an intuitive interface that allows users to effortlessly explore, save, and share their favorite music tracks and playlists.

Comprehensive Music Streaming: Provide robust features for organizing and managing music content, including advanced search options for easy discovery.

Modern Tech Stack: Harness cutting-edge web development technologies, such as Node.js to ensure an efficient and enjoyable user experience while navigating and interacting with the music streaming application.

Key Features:-

- **Music Listing** – Display all available songs with metadata.
- **Playlist Management** – Create, update, delete, and play playlists.
- **Music Playback** – Play, pause, and control volume.
- **Search Functionality** – Search by song title, artist, or genre.
- **Offline Mode** – Allow users to download songs for offline listening.
- **User Authentication** – Login, register, and manage user accounts.

PRE-QUISTES:-

Here are the key prerequisites for developing a frontend application using Node.js

- Node.js and npm:

Node.js is a powerful JavaScript runtime environment that allows you to run JavaScript code on the local environment. It provides a scalable and efficient platform for building network applications.

Install Node.js on your development machine, as they are required to run JavaScript on the server-side.

- Download: https://nodejs.org/en/download/
- Installation instructions: https://nodejs.org/en/download/package-manager

HTML, CSS, and JavaScript: Basic knowledge of HTML for creating the structure of your app, CSS for styling, for client-side interactivity is essential.

- Version Control: Use Git for version control, enabling collaboration and tracking changes throughout the development process. Platforms like GitHub or Bitbucket can host your repository.

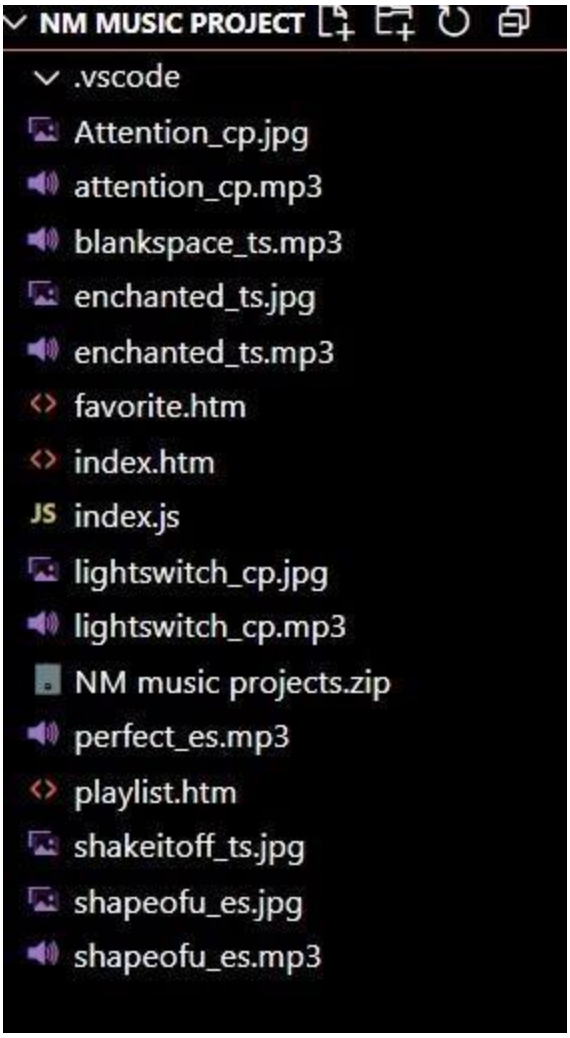
- Git: Download and installation instructions can be found at:

  https://git-scm.com/downloads

- Development Environment: Choose a code editor or Integrated Development Environment (IDE) that suits your preferences, such as Visual Studio Code, Sublime Text, or WebStorm.

  - Visual Studio Code: Download from https://code.visualstudio.com/download

Project Structure:



The project structure may vary depending on the specific library, framework, programming language, or development approach used. It's essential to organize the files and directories in a logical and consistent manner to improve code maintainability and collaboration among developers.

PROJECT FLOW:-

Project demo:

Before starting to work on this project let's see the demo

Demo Link:

https://drive.google.com/drive/folders/1BE_rUNirRFDZqXrKcoayetcDLQt7Hksh

Use the code in:

https://github.com/Krishasooyaaa/Music-streaming-app.git

Milestone 1: Project Setup and Configuration:

1. Install required tools and software:

● Installation of required tools:

1. Open the project folder and install the necessary tools.

In this project, we use:

• Node.js (JavaScript runtime)

• Express.js (Web framework for serving pages)

• EJS (For rendering HTML dynamically)

• Bootstrap (For styling)

• Axios (For API requests)

• Nodemon (For live server reload during development)

- For further reference, use the following resources

  https://youtu.be/NqANV4wXhx4?si=u7RgiYkRI5Jx8_lx

  https://youtu.be/-Y_sgknf57U?si=99UQ6vKwfJWHsqUy

  https://youtu.be/ENrzD9HAZK4?si=6cPFH5DAXg1jmLoo

Milestone 2 : Project Development

1. Setup Node.js Application

- Create a new project folder
- Install required libraries
- Set up routing
- Use static HTML, CSS, and Node.js

2. Fetching songs

```
<script>
    document.addEventListener("DOMContentLoaded", function () {
        const favoriteIcons = document.querySelectorAll(".favorite");

        // Load saved favorites from localStorage
        let favorites = JSON.parse(localStorage.getItem("favorites")) || [];
        let playlists = JSON.parse(localStorage.getItem("playlists")) || [];

        function updateFavoriteIcons() {
            favoriteIcons.forEach(icon => {
                const songCard = icon.closest(".song-card");
                const songTitle = songCard.querySelector("h3").innerText;

                if (favorites.some(song => song.title === songTitle)) {
                    icon.classList.add("liked");
                    icon.classList.replace("fa-regular", "fa-solid");
                } else {
                    icon.classList.remove("liked");
                    icon.classList.replace("fa-solid", "fa-regular");
                }
            });
        }

        updateFavoriteIcons(); // Ensure correct heart styles on page load

        favoriteIcons.forEach(icon => {
            icon.addEventListener("click", function () {
                const songCard = this.closest(".song-card");
                const songTitle = songCard.querySelector("h3").innerText;
                const songImage = songCard.querySelector("img").src;
                const songAudio = songCard.querySelector("audio source").src;
```

```
            if (this.classList.contains("liked")) {
                this.classList.replace("fa-solid", "fa-regular");
                favorites = favorites.filter(song => song.title !== songTitle);
            } else {
                this.classList.replace("fa-regular", "fa-solid");
                favorites.push({ title: songTitle, image: songImage, audio: songAudio });
            }

            this.classList.toggle("liked");
            localStorage.setItem("favorites", JSON.stringify(favorites));
        });
    });

    // Add to Playlist functionality
    const addToPlaylistButtons = document.querySelectorAll(".add-to-playlist");
    addToPlaylistButtons.forEach(button => {
        button.addEventListener("click", function () {
            const songCard = this.closest(".song-card");
            const songTitle = songCard.querySelector("h3").innerText;
            const songImage = songCard.querySelector("img").src;
            const songAudio = songCard.querySelector("audio source").src;

            const song = { title: songTitle, image: songImage, audio: songAudio };

            // Add song to playlist if not already added
            if (!playlists.some(s => s.title === songTitle)) {
                playlists.push(song);
                localStorage.setItem("playlists", JSON.stringify(playlists));
                localStorage.setItem("playlists", JSON.stringify(playlists));
                alert(`${songTitle} added to Playlist!`);
            } else {
                alert(`${songTitle} is already in the playlist.`);
            }
        });
    });
});

// Add to Playlist functionality
ocument.querySelectorAll(".add-to-playlist").forEach(button => {
  button.addEventListener("click", function () {
    const songCard = this.closest(".song-card");
    const songTitle = songCard.querySelector("h3").innerText;
    const songImage = songCard.querySelector("img").src;
    const songAudio = songCard.querySelector("audio source").src;

    let playlist = JSON.parse(localStorage.getItem("playlist")) || [];

    // Check if the song is already in the playlist
    if (!playlist.some(song => song.title === songTitle)) {
        playlist.push({ title: songTitle, image: songImage, audio: songAudio });
        localStorage.setItem("playlist", JSON.stringify(playlist));
        alert(`${songTitle} added to your playlist!`);
    } else {
        alert(`${songTitle} is already in your playlist.`);
    }
  });
```

```
    });
});


    // Search functionality
    document.getElementById("searchInput").addEventListener("input", function () {
        const searchTerm = this.value.toLowerCase();
        const songs = document.querySelectorAll(".song-card");
        let found = false;

        songs.forEach(song => {
            const title = song.getAttribute("data-song").toLowerCase();
            const singer = song.getAttribute("data-singer").toLowerCase();

            if (title.includes(searchTerm) || singer.includes(searchTerm)) {
                song.style.display = "block";
                found = true;
            } else {
                song.style.display = "none";
            }
        });
        document.querySelector(".no-results").style.display = found ? "none" : "block";
    });
</script>
```

CODE DESCRIPTION

1. **Add to Playlist Functionality**

- This segment allows users to add songs to a playlist and store it in localStorage.
- It selects all elements with the .add-to-playlist class.
- When a button is clicked, it retrieves song details (title, image, audio) from the closest .song-card.
- It checks if the song is already in the playlist using .some().
- If the song is not in the playlist, it adds the song details to an array and updates localStorage.
- If the song is already in the playlist, it alerts the user.

2. **Search Functionality**

- This part enables a search feature for songs.
- It listens for input in the search bar (#searchInput).
- It retrieves the search term and converts it to lowercase.
- It loops through all song elements (.song-card), checking if either the song title or singer name contains the search term.
- If a match is found, the song is displayed; otherwise, it is hidden.
- If no results are found, a no-results message is displayed.

3. **Favorite Songs Feature**

- This part allows users to mark songs as favorites and save them.
- When the page loads, it retrieves favorites from localStorage.
- It updates the UI to show liked songs with a solid heart (fa-solid).
- Clicking on a favorite button toggles the song's favorite status.
- If already liked, it removes the song from favorites.
- If not liked, it adds it to the favorites list.
- It updates localStorage accordingly.

4. **Playlist Management**

- This section ensures songs are correctly added to the playlist.
- It loops through .add-to-playlist buttons and adds a click event listener.
- On clicking, it fetches song details and checks if the song is already in localStorage.
- If not present, it adds the song and updates localStorage.

FRONT-END CODE FOR DISPLAYING SONGS

```
<div class="song-container" id="songContainer">
    <div class="song-card" data-song="Attention" data-singer="Charlie puth" data-mp3="attention_cp.mp3">
        <i class="fa-regular fa-heart favorite"></i>
        <img src="attention_cp.jpg" alt="Song Cover">
        <h3>Attention</h3>
        <p>Genre: Pop music</p>
        <p>Singer: Charlie Puth</p>
        <audio controls>
            <source src="attention_cp.mp3" type="audio/mp3">
            Your browser does not support the audio element.
        </audio>
        <button class="add-to-playlist">Add to Playlist</button>
    </div>

    <div class="song-card" data-song="Enchanted" data-singer="Taylor Swift" data-mp3="enchanted_ts.mp3">
        <i class="fa-regular fa-heart favorite"></i>
        <img src="enchanted_ts.jpg" alt="Song Cover">
        <h3>Enchanted</h3>
        <p>Genre: Romantic</p>
        <p>Singer: Taylor Swift</p>
        <audio controls>
            <source src="enchanted_ts.mp3" type="audio/mp3">
            Your browser does not support the audio element.
        </audio>
        <button class="add-to-playlist">Add to Playlist</button>
    </div>
```

CODE DESCRIPTION

1. **Container (div class="song-container")**

   This acts as the main wrapper for the song list.

2. **Song Card (div class="song-card")**

   Each song is represented as a separate div with attributes:

   data-song: Stores the song title.

   data-singer: Stores the artist's name.

   data-mp3: Stores the filename of the corresponding MP3 file.

3. **Favorite Icon (i class="fa-regular fa-heart favorite")**

   Represents a favorite (heart) icon, likely used for liking the song.

4. **Song Cover (img src="...")**

   Displays the cover image for the song.

5. **Song Title (h3)**

   Displays the name of the song.

6. **Genre & Singer Information (p**)

   Displays the genre and singer's name.

7. **Audio Player (audio controls)**

   Contains an tag with a element that dynamically loads the MP3 file.

8. "**Add to Playlist" Button (button class="add-to-playlist")**

   Allows users to add the song to their playlist.

Example Songs in the Code:

1. "Attention" by Charlie Puth

   Genre: Pop

   Cover Image: attention_cp.jpg

MP3 File: attention_cp.mp3

2. "Enchanted" by Taylor Swift

   Genre: Romantic

   Cover Image: enchanted_ts.jpg

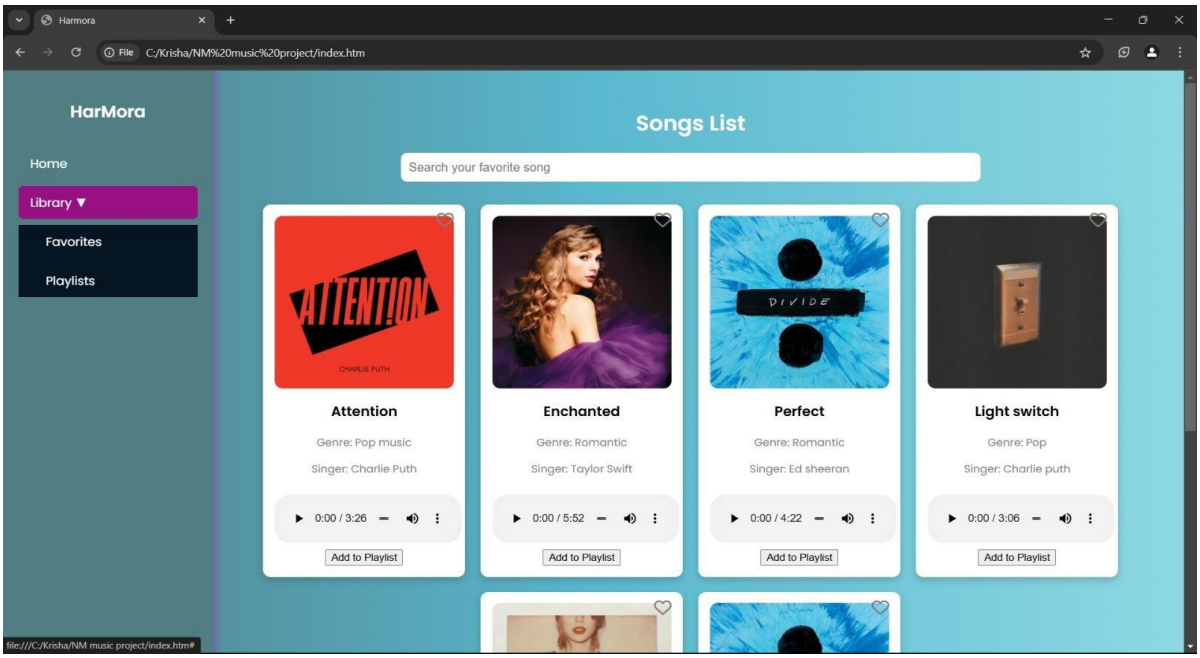
   MP3 File: enchanted_ts.mp3

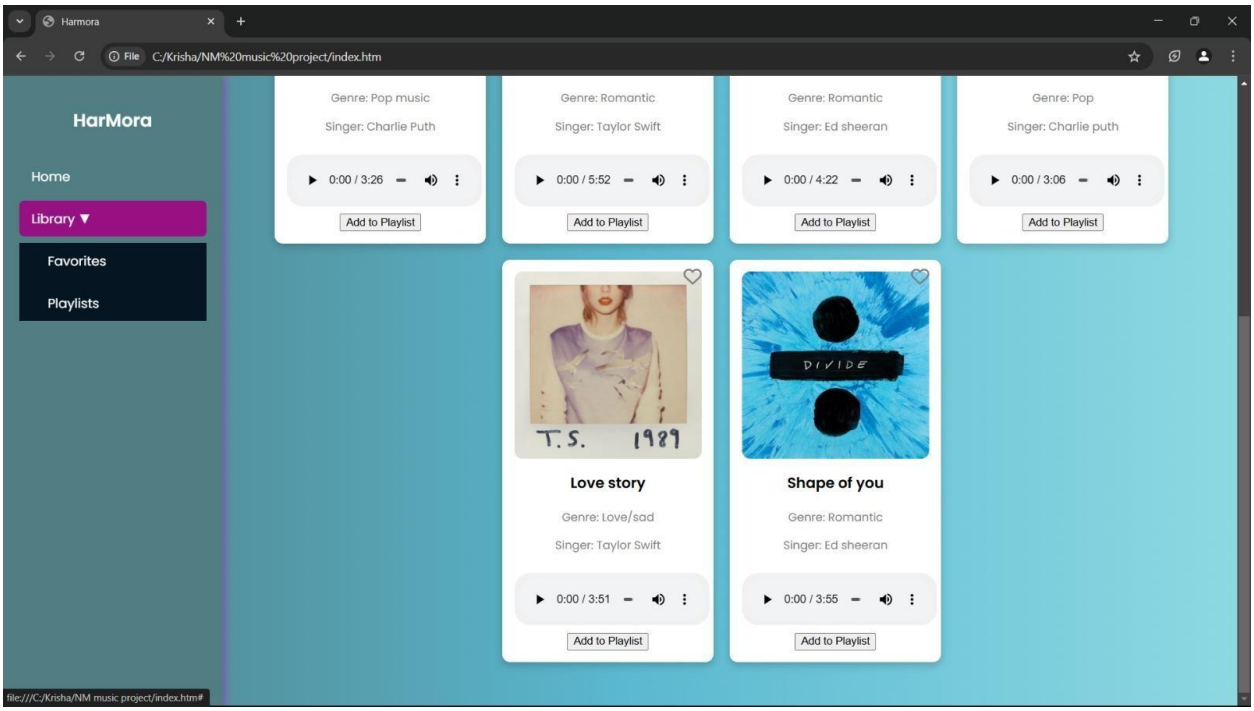
Project Execution:

After completing the code, run the Node.js application using vs code

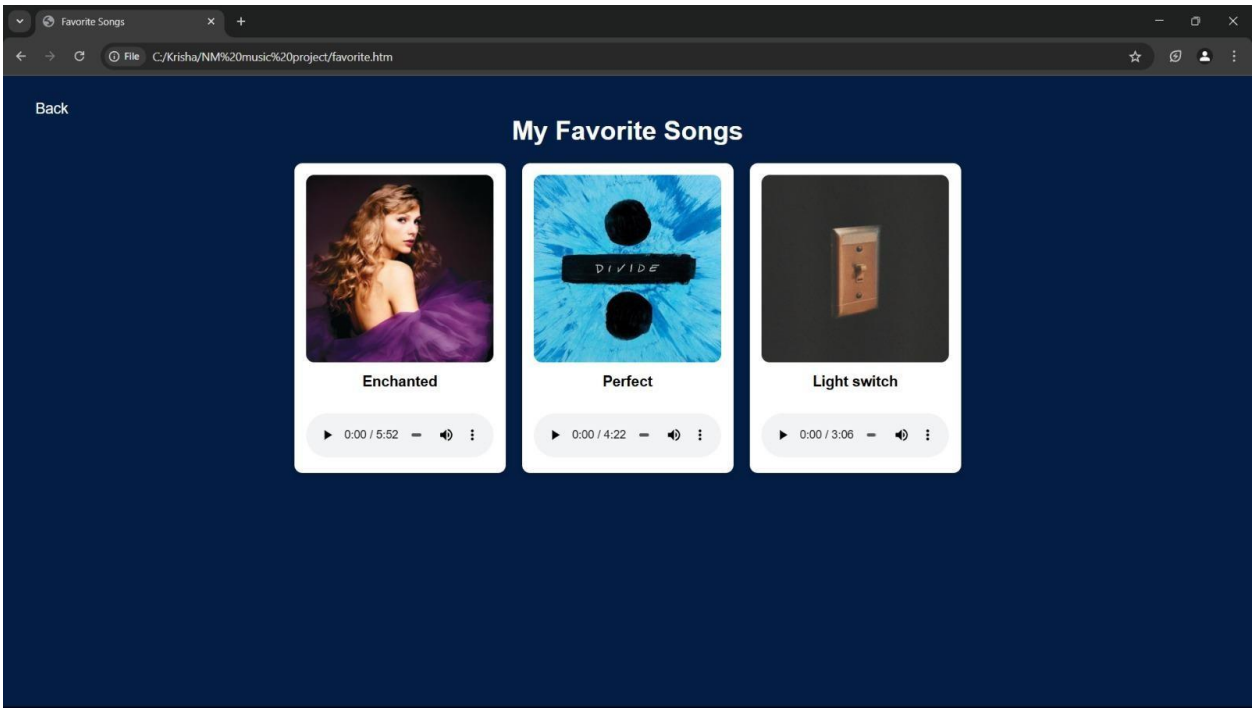
After that launch the Harmora

Here are some of the screenshots of the application.

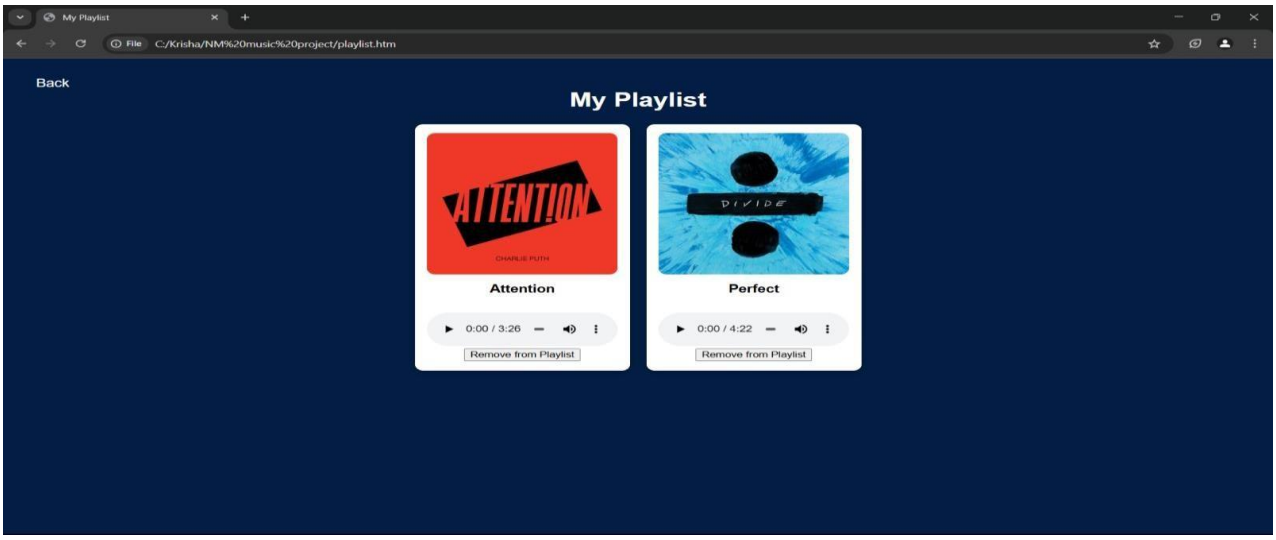




FAVORITES



PLAYLIST



PROJECT DEMO LINK

https://drive.google.com/drive/folders/1BJgnEpilG4Gw02z1PMTUm2C9rbKshkZk