WaveBeats(Node)

1. Introduction

- **Project Title**: Music Streaming Application

  Project Name: WaveBeats

  Team Leaders – Lokeshwari M (mlokeshwari2021@gmail.com)

  Team Members

  1. Mahalakshmi (mahalakshminagendran583@gmail.com)
  2. Meenalakshmi (meena11oct@gmail.com)
  3. Meenakshi (meenakshisrinivasan137@gmail.com)

2. Project Overview

- **Purpose**:

  WaveBeats is a web-based music streaming platform designed to provide users with seamless access to a vast collection of music. The platform allows users to browse, search, and play their favorite tracks with an intuitive and user-friendly interface.

  WaveBeats enhances user engagement through personalized recommendations, curated playlists, and social sharing features. The platform leverages AI-driven algorithms to suggest songs based on listening habits, ensuring a tailored experience for every user. With high-quality audio streaming, offline listening capabilities, and cross-device compatibility, WaveBeats offers a flexible and immersive music experience. Furthermore, it supports independent artists by providing a space to showcase their work and connect with a global audience. Whether users are exploring new genres or enjoying their favorite tunes, WaveBeats delivers a dynamic and interactive way to experience music.

  - **Scenario-Based Introduction**

  Your day is filled with different moments, and each one deserves the right soundtrack. Wave Beats offers a seamless way to discover music that matches your mood and activity. Whether you're commuting, working, or unwinding, the right song is always just a tap away.

  Enjoy a peaceful commute with soothing instrumentals or power through workouts with high-energy beats. Set the perfect ambiance for work or relaxation with curated playlists designed to enhance focus and calmness.

With a vast library of genres and moods, Wave Beats ensures that every moment is accompanied by the right sound.

Beyond listening, Wave Beats helps you discover new artists and tracks tailored to your preferences. Personalized recommendations make it easy to explore fresh music that aligns with your taste, keeping your playlists dynamic and exciting. You can also create and share custom playlists, making every experience uniquely yours.

No matter where life takes you, Wave Beats provides a seamless and intuitive listening experience. Whether you need motivation, relaxation, or celebration, Wave Beats ensures you always have the perfect music to match your journey.

- **Features**:

  User-Friendly Interface: A simple and intuitive UI for easy navigation.

  Music Playback: Users can play, pause, and skip songs.

  Search Functionality: Users can search for songs by title, artist, or genre.

  Playlist Management: Users can create, edit, and manage playlists.

  Responsive Design: Optimized for various screen sizes .

  Custom Audio Player: Built using HTML5 and JavaScript for smooth playback

3. Architecture

- **Component Structure**: WaveBeats use the architecture follows a traditional Node.js + HTML + CSS approach. The main components are:

  App Component: Manages state and renders the main sections.

  Song List Component: Displays the list of songs.

  Song Card Component: Represents each song with an image, title, and audio controls.

  Favourites Component: Displays a list of favourite songs.

  INCLUDES:

  i. Frontend (HTML, CSS, JavaScript) – Handles UI and user interactions.
  ii. Backend (Node.js, Express) – Manages API requests, authentication, and database operations.

- **State Management**: The state is managed using JavaScript (localStorage, sessionStorage, or API calls to the backend).
- **Routing**: Implemented via Express.js for handling API routes.Frontend navigation is managed using standard HTML file linking and JavaScript

4. Setup Instructions

Pre-requisites:

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.

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

Install required tools and software:

- Installation of required tools:
  Open the project folder and install the necessary tools. In this project, we use:

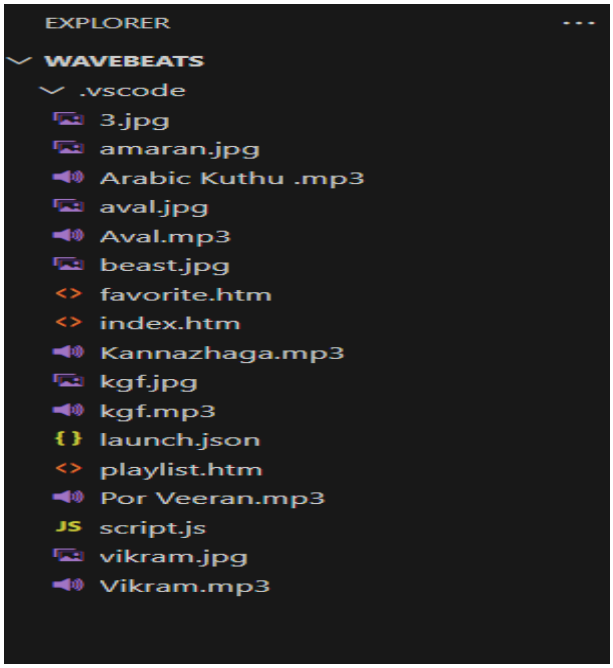
• Node.js (JavaScript runtime)
• Express.js (Web framework for serving pages)
• EJS (For rendering HTML dynamically)
• Bootstrap (For styling) • Axios (For API requests)
• Nodemon (For live server reload during development)
• For further reference, use the following resources
https://www.youtube.com/watch?v=NqANV4wXhx4
https://www.youtube.com/watch?v=-Y_sgknf57U
https://www.youtube.com/watch?v=ENrzD9HAZK4

Project Development

Setup Node.js Application

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

5. Project Folder



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/file/d/1-GxQyXf3M3vHPyi7-aBhpP5b9nOiCwmq/view?usp=sharing

Use this code:

https://github.com/21lokeshwari/WaveBeats_SWTID1741163327150524

6. Folder Structure

Utilities: - JavaScript Functions: Helper functions for audio playback, playlist management, and local storage handling.

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

- **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

- **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.

- **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.

- **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.

7.Running the Application

Frontend:

Run the following command to start the frontend locally:

npm start

Or simply open index.html in a browser.

8. Component Documentation

a. **Key Components**:

Audio Player: Custom HTML5 audio player with play, pause, skip, and volume controls.

Search Bar: JavaScript-powered search functionality to filter songs.

Playlist Manager: Allows users to create and save playlists.

Favorites : Allows users to mark/unmark songs as favorites .

b. **Reusable Components**:

Button Components: Used for playback controls.

Song Cards: Display song details (title, artist).

9. State Management

a. **Global State**:

Handled using local Storage to store user preferences and playlists.

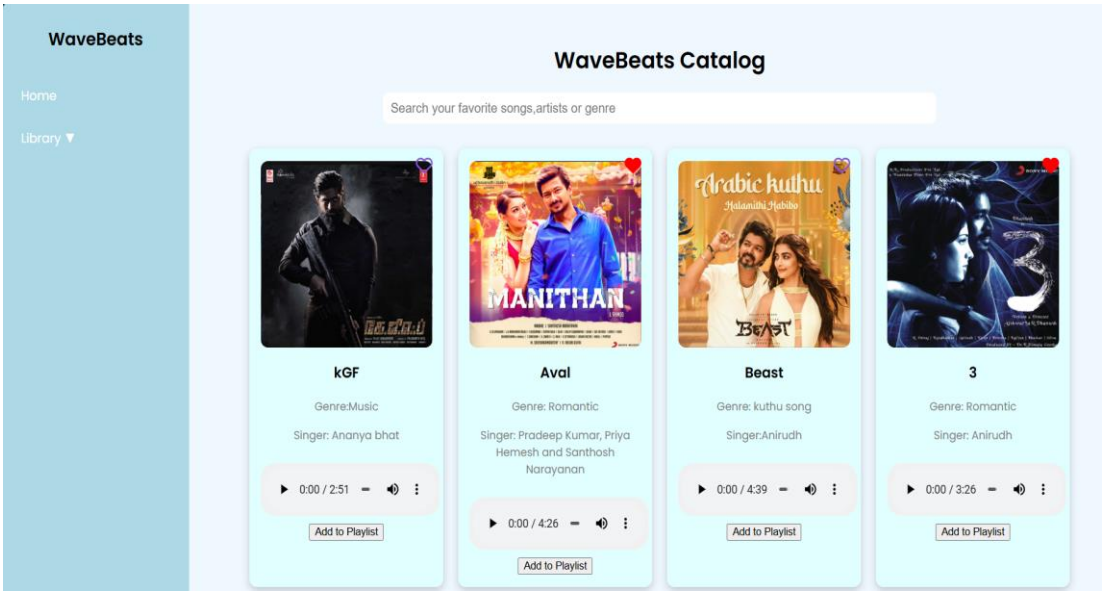
API calls fetch music data from the backend.

b. **Local State**:

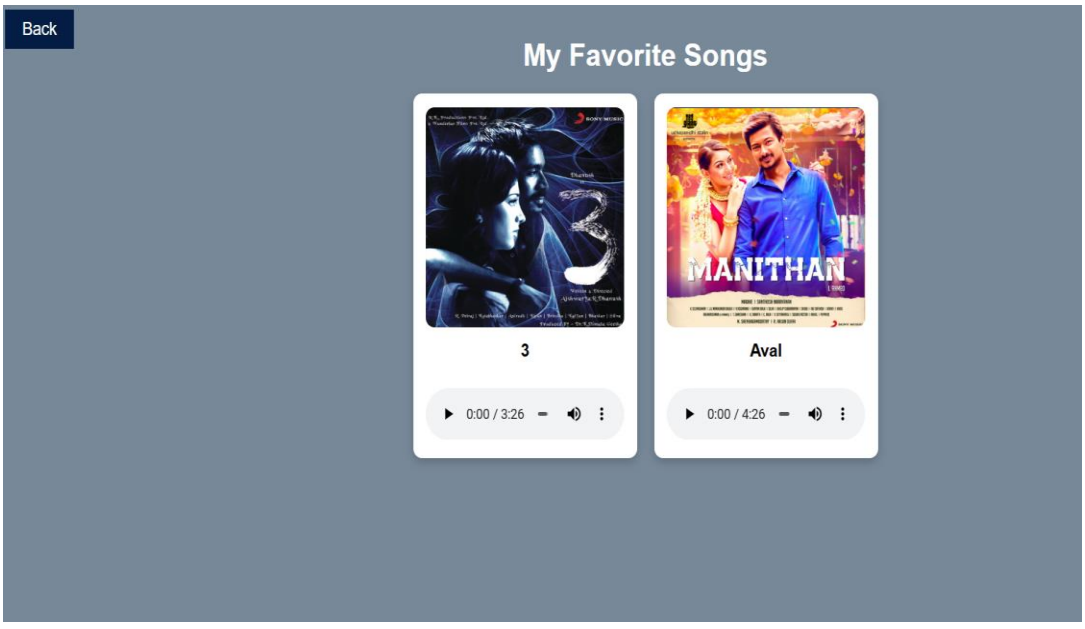
JavaScript variables manage audio playback, track progress, and UI updates.

10. User Interface

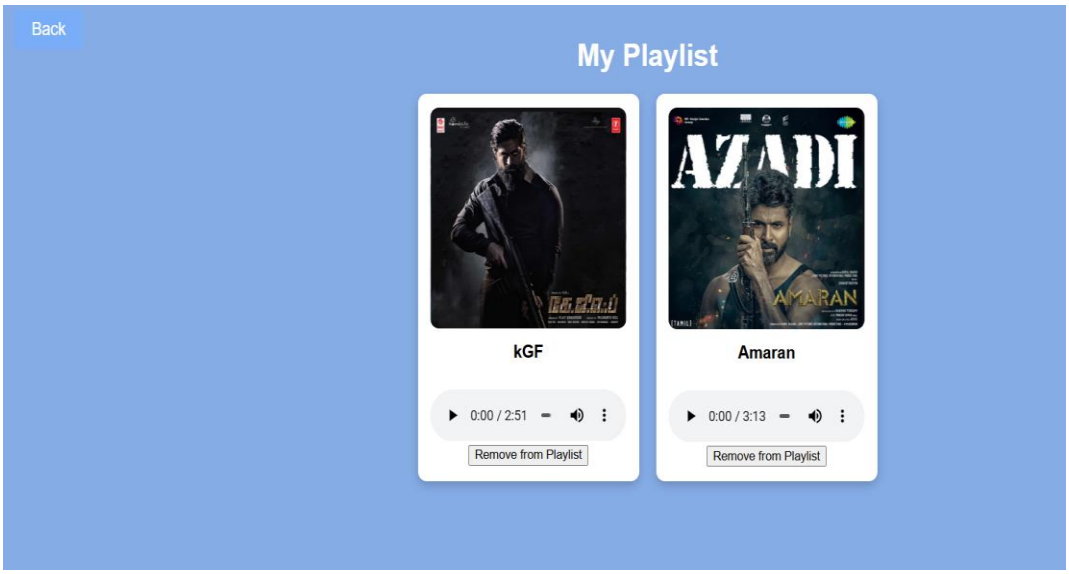
HOME



FAVORITE



PLAYLIST



11.Styling

- **CSS Frameworks/Libraries**: Pure CSS with some custom animations using CSS3.
- **Theming**: Light-Blue mixed with green theme color background

11. **Testing**

- **Testing Strategy**:

Unit Testing: Test JavaScript functions (e.g., playlist management).

Integration Testing: Ensure frontend and backend API communication works.

Manual UI Testing: Check responsiveness and functionality across devices.

- **Code Coverage**

Here I added the Coding Screenshots which we have done the projects

FRONT-END CODE FOR DISPLAYING SONGS

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>WaveBeats</title>
    <link rel="stylesheet" href="https://fonts.googleapis.com/css2?family=Poppins:wght@300;400;600&display=swap">
    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.4.2/css/all.min.css">
    <style>
        /* General Styles */
        body {
            font-family: 'Poppins', sans-serif;
            margin: 0;
            padding: 0;
            background: aliceblue;
            color: black;
        }

        /* Sidebar */
        .sidebar {
            width: 230px;
            height: 100vh;
            position: fixed;
            background-color: lightblue;
            padding: 10px;
            box-shadow: 1px 0 rpx rgba(0,0,0.1,); green, blue, alpha)
        }

        .sidebar h2 {
            font-size: 22px;
            margin-bottom: 30px;
            font-weight: 600;
            text-align: center;
        }

        .sidebar a {
            display: block;
```

Ln 1, Col 1 Spaces: 4 UT

```
<html lang="en">
<head>
    <style>
        .sidebar a {
            display: block;
            color: white;
            text-decoration: none;
            padding: 10px 15px;
            margin-bottom: 8px;
            border-radius: 6px;
            transition: 0.3s;
        }

        .sidebar a:hover {
            background-color: blueviolet;
        }

        /* Dropdown for Library */
        .dropdown {
            position: relative;
        }

        .dropdown-content {
            display: none;
            background-color: black;
            padding-left: 20px;
        }

        .dropdown:hover .dropdown-content {
            display: block;
        }

        /* Main Content */
        .main-content {
            margin-left: 250px;
            padding: 30px;
        }
```

```
<head>
    <style>
        .dropdown-content {
            padding-left: 20px;
        }

        .dropdown:hover .dropdown-content {
            display: block;
        }

        /* Main Content */
        .main-content {
            margin-left: 250px;
            padding: 30px;
        }

        h1 {
            text-align: center;
            font-size: 28px;
            font-weight: bold;
        }

        /* Search Bar */
        .search-bar {
            display: flex;
            justify-content: center;
            margin-bottom: 30px;
        }

        .search-bar input {
            width: 60%;
            padding: 10px;
            font-size: 16px;
            border-radius: 8px;
            border: none;
```

```
<head>
    <style>
        .search-bar input {
            border: none;
            outline: none;
        }

        /* Song Cards */
        .song-container {
            display: flex;
            flex-wrap: wrap;
            justify-content: center;
            gap: 20px;
        }

        .song-card {
            background: lightcyan;
            color: black;
            border-radius: 10px;
            padding: 15px;
            width: 230px;
            text-align: center;
            box-shadow: 0px 4px 10px rgba(0, 0, 0, 0.2);
            position: relative;
        }

        .song-card img {
            width: 100%;
            border-radius: 10px;
        }

        .song-card h3 {
            font-size: 18px;
            margin-top: 10px;
        }
```

```
<html lang="en">
<head>
    <style>
        .song-card h3 {
            font-size: 18px;
            margin-top: 10px;
        }

        .song-card p {
            font-size: 14px;
            color: gray;
        }

        .song-card audio {
            width: 100%;
            margin-top: 10px;
            border-radius: 20px;
            background: #f1f1f1;
            padding: 5px;
        }

        /* Favorite Icon */
        .favorite {
    position: absolute;
    top: 10px;
    right: 15px;
    font-size: 22px;
    cursor: pointer;
    color: rgb(139, 88, 209);
}


        .favorite.liked {
            color: red;
        }

        /* No Results Message */
```

```
<html lang="en">
<head>
    <style>
        .favorite.liked {
            color: red;
        }

        /* No Results Message */
        .no-results {
            display: none;
            text-align: center;
            font-size: 18px;
            color: lightgray;
            margin-top: 20px;
        }
    </style>
</head>
<body>

    <!-- Sidebar -->
    <div class="sidebar">
        <h2> WaveBeats</h2>
        <a href="index.htm">Home</a>
        <div class="dropdown">
            <a href="#">Library ▼</a>
            <div class="dropdown-content">
                <a href="favorite.htm">Favorites</a>
                <a href="playlist.htm">Playlists</a>
            </div>
        </div>
    </div>

    <!-- Main Content -->
    <div class="main-content">
        <h1>WaveBeats Catalog</h1>
```

```
lang="en">
>
div class="main-content">

    <!-- Search Bar -->
    <div class="search-bar">
        <input type="text" placeholder="Search your favorite songs,artists or genre" id="searchInput">
    </div>

    <!-- No Results Message -->
    <p class="no-results">No songs found.</p>

    <!-- Songs -->
    <div class="song-container" id="songContainer">
        <div class="song-card" data-song="kGF" data-singer="Ananya bhat" data-mp3="kgf.mp3">
            <i class="fa-regular fa-heart favorite"></i>
            <img src="kgf.jpg" alt="Song Cover">
            <h3>kGF</h3>
            <p>Genre:Music</p>
            <p>Singer: Ananya bhat</p>
            <audio controls>
                <source src="kgf.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="Aval" data-singer="Pradeep Kumar, Priya Hemesh and Santhosh Narayanan." data-mp3="Aval.mp3">
            <i class="fa-regular fa-heart favorite"></i>
            <img src="aval.jpg" alt="Song Cover">
            <h3>Aval</h3>
            <p>Genre: Romantic</p>
            <p>Singer: Pradeep Kumar, Priya Hemesh and Santhosh Narayanan</p>
            <audio controls>
                <source src="Aval.mp3" type="audio/mp3">
                Your browser does not support the audio element.
```

```
 lang="en">
>
div class="main-content">
   <div class="song-container" id="songContainer">
      <div class="song-card" data-song="Aval" data-singer="Pradeep Kumar, Priya Hemesh and Santhosh Narayanan." data-mp3="Aval.mp3">
            <source src="Aval.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="Beast" data-singer="Anirudh" data-mp3="Arabic Kuthu.mp3">
        <i class="fa-regular fa-heart favorite"></i>
        <img src="beast.jpg" alt="Song Cover">
        <h3>Beast</h3>
        <p>Genre: kuthu song</p>
        <p>Singer:Anirudh</p>
        <audio controls>
            <source src="Arabic Kuthu .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="3" data-singer="Anirudh" data-mp3="kannazhaga.mp3">
        <i class="fa-regular fa-heart favorite"></i>
        <img src="3.jpg" alt="Song Cover">
        <h3>3</h3>
        <p>Genre: Romantic</p>
        <p>Singer: Anirudh</p>
        <audio controls>
            <source src="kannazhaga.mp3" type="audio/mp3">
            Your browser does not support the audio element.
        </audio>
        <button class="add-to-playlist">Add to Playlist</button>
      </div>
```

Ln 177, Col 35 Spaces: 4 UTF-8 CRL

```
  lang="en">
 >
 div class="main-content">
    <div class="song-container" id="songContainer">
        <div class="song-card" data-song="3" data-singer="Anirudh" data-mp3="kannazhaga.mp3">
            <button class="add-to-playlist">Add to Playlist</button>
        </div>

        <div class="song-card" data-song="Vikram" data-singer="Anirudh" data-mp3="vikram.mp3">
            <i class="fa-regular fa-heart favorite"></i>
            <img src="vikram.jpg" alt="Song Cover">
            <h3>Vikram</h3>
            <p>Genre: Love/sad</p>
            <p>Singer: Anirudh</p>
            <audio controls>
                <source src="vikram.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="Amaran" data-singer="Arivu" data-mp3=" Por Veeran.mp3">
            <i class="fa-regular fa-heart favorite"></i>
            <img src="amaran.jpg" alt="Song Cover">
            <h3> Amaran</h3>
            <p>Genre:protest music</p>
            <p>Singer: Arivu</p>
            <audio controls>
                <source src=" Por Veeran.mp3" type="audio/mp3">
                Your browser does not support the audio element.
            </audio>
            <button class="add-to-playlist">Add to Playlist</button>
        </div>
    </div>
 /div>
```

```
 lang="en">
>
div class="main-content">
   <div class="song-container" id="songContainer">
   </div>
/div>

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;
```

Ln 177, Col 35 Spaces

```
 lang="en">
>
script>
   document.addEventListener("DOMContentLoaded", function () {
       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 };
```

```
<html lang="en">
<body>
    <script>
document.querySelectorAll(".add-to-playlist").forEach(button => {
    button.addEventListener("click", function () {
        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";
                }
```

```
<html lang="en">
<body>
    <script>
document.querySelectorAll(".add-to-playlist").forEach(button => {
    button.addEventListener("click", function () {
        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";
                }
```

CODE DESCRIPTION

Container (div class="song-container")

This acts as the main wrapper for the song list.

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.

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

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

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

Displays the cover image for the song..

Song Title (h3)

Displays the name of the song.

Genre & Singer Information (p)

Displays the genre and singer's name.

Audio Player (audio controls)

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

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

Allows users to add the song to their playlist.

Example Songs in the Code:

1. "KGF" by Ananya Bhat

   Genre: Music

   Cover Image: kgf.jpg

   MP3 File: kgf.mp3

2. "Aval" by Pradeep Kumar , Priya Hemesh and Santhosh Narayanan

   Genre: Romantic

   Cover Image aval.jpg

MP3 File: Aval.mp3

12.**Screenshots or Demo**

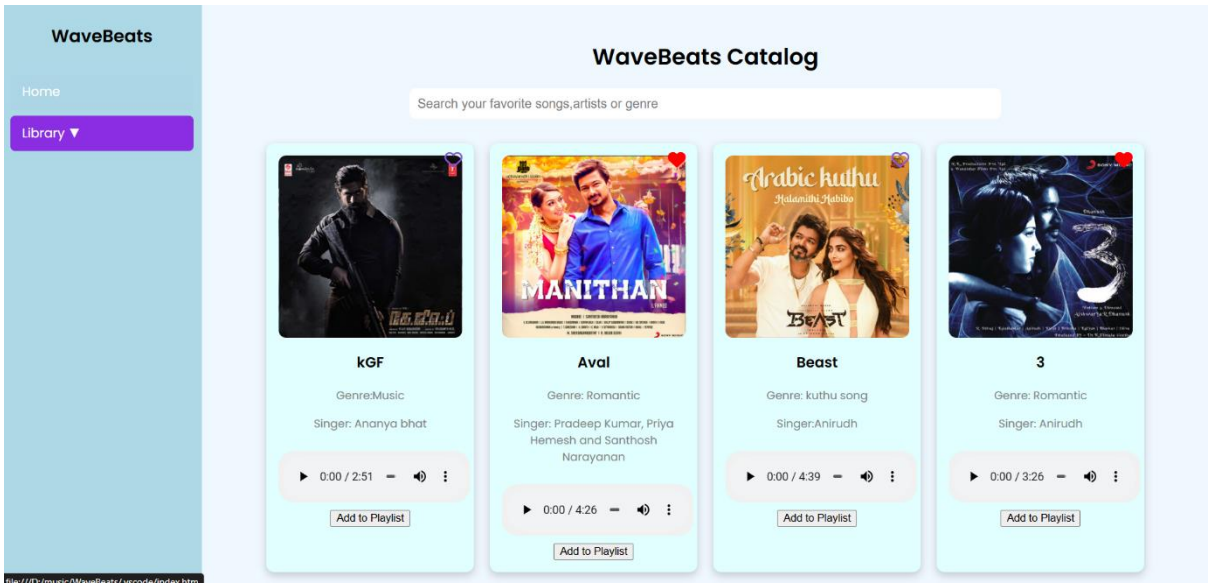
- **Screenshots**

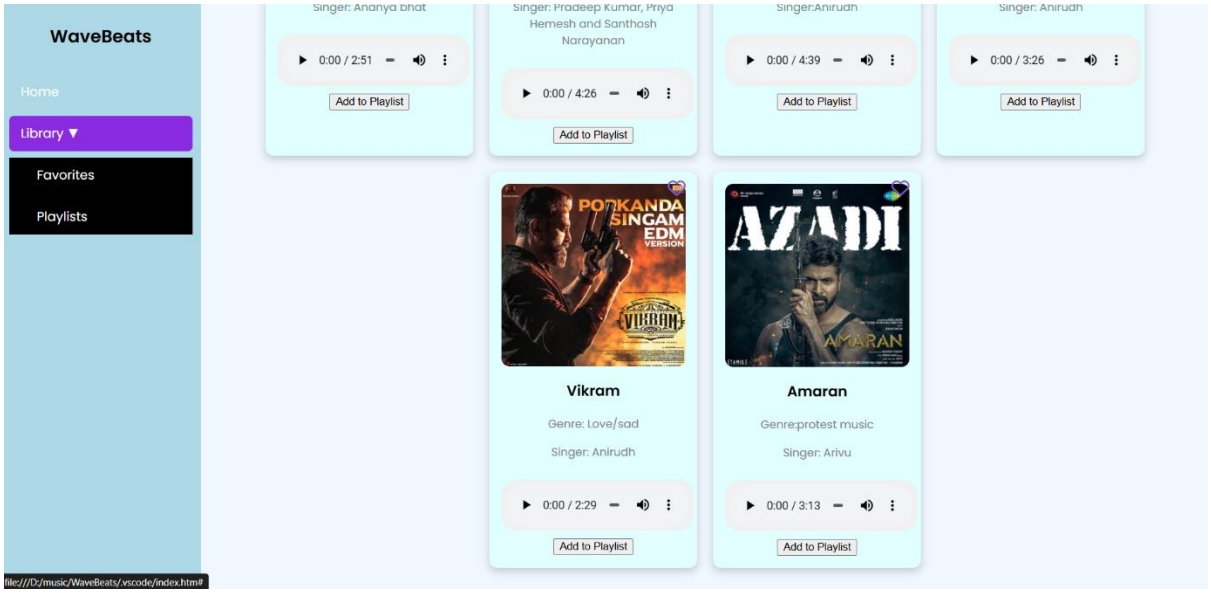
  After completing the code, run the Node.js application using vs code

  After that launch the WaveBeats

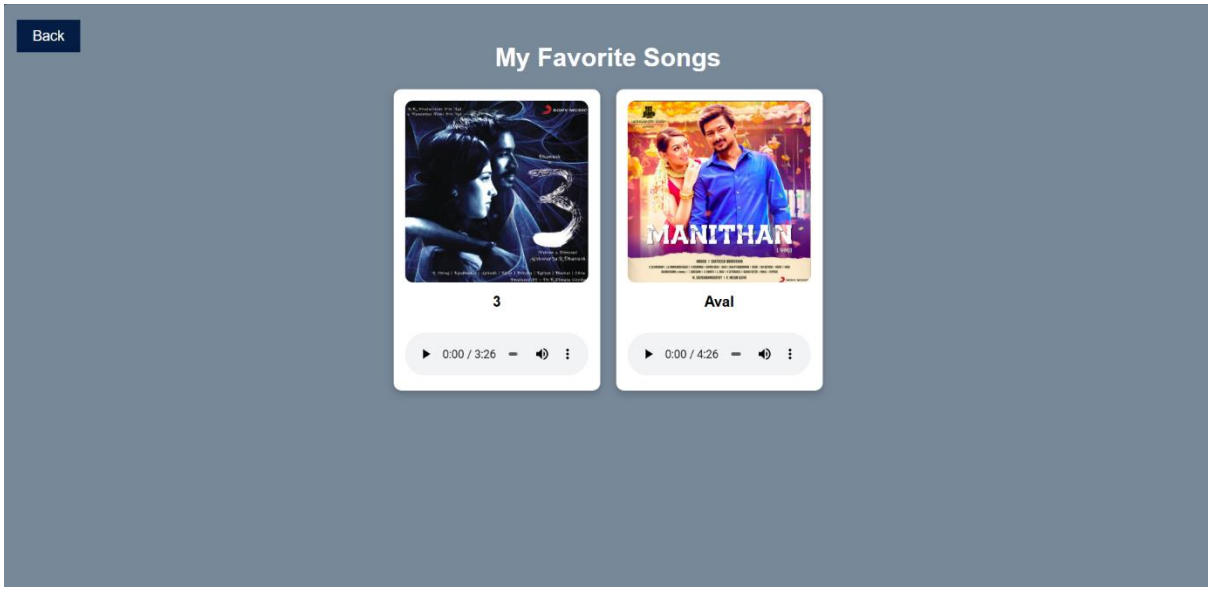
  Here are some of the screenshots of the application.

HERO COMPONENT

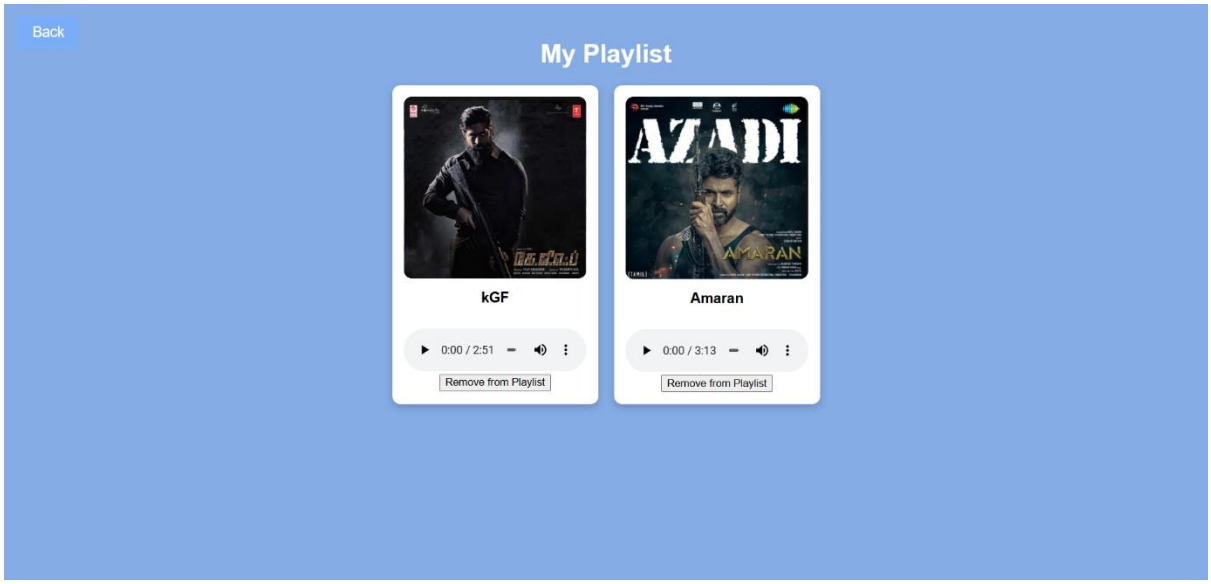




FAVORITES SONGS



PALYLISTS SONGS



- **Project Demo Link**

https://drive.google.com/file/d/1-GxQyXf3M3vHPyi7-aBhpP5b9nOiCwmq/view?usp=sharing

13.**Known Issues**

- Cross-browser compatibility issues on older browsers.
- Performance optimization needed for large song libraries.
- If there are too many songs, searching might be slow or unresponsive.

14. Future Enhancements

- User Authentication: Allow users to create accounts and save preferences.
- Social Sharing: Share playlists on social media.
- Offline Mode: Cache songs for offline playback.
- Advanced Music Recommendations: AI-powered song suggestions.