PROJECT DOCUMENTATION
RHYTHMIC TUNES: YOUR MELODIC COMPANION

INTRODUCTION

- **Project Title: RHYTHMIC TUNES: YOUR MELODIC COMPANION**
- **Team ID**: **NM2025TMID37392**
- **Team Leader: SHAMINI K** (shamini03k@gmail.com)
- **Team Members:**
  - **DEEPIKA M** (deepikamurugesan3007@gmail.com)
  - **KEERTHIKA B** (bkeerthikakeerthika6@gmail.com)

PROJECT OVERVIEW

PURPOSE:

The purpose of Rhythmic Tunes: Your Melodic Companion is to redefine the way individuals interact with music by creating a platform that goes beyond simple listening. Music is universal, and it plays an essential role in human life, whether it is to uplift moods, relieve stress, increase productivity, or provide relaxation. This project aims to develop a dynamic, intelligent, and user-friendly music application that not only delivers songs but also becomes a personalized melodic partner in the daily lives of its users.

Unlike traditional music platforms that focus mainly on streaming, Rhythmic Tunes emphasizes rhythm-based experiences and personalized companionship. The application leverages smart algorithms and user preferences to recommend songs, playlists, and rhythms that align with the listener's emotional state, activity, or environment. For example, whether someone is working, exercising, meditating, or relaxing, Rhythmic Tunes will adapt and provide suitable melodies that enhance those moments.

FEATURES:

- Song Listings: Display a comprehensive list of available songs with details such as title, artist, genre, and release date.
- Playlist Creation: Empower users to create personalized playlists, adding and organizing songs based on their preferences.
- Playback Control: Implement seamless playback control features, allowing users to play, pause, skip, and adjust volume during music playback.
- Offline Listening: Allow users to download songs for offline listening, enhancing the app's accessibility and convenience.
- Search Functionality: Implement a robust search feature for users to easily find specific songs, artists, or albums within the app.

SETUP INSTRUCTIONS

PREREQUISITES:

- **NODE.JS :**

  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 and npm 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/

- **REACT.JS :**

  React.js is a popular JavaScript library for building user interfaces. It enables developers to create interactive and reusable UI components, making it easier to build dynamic and responsive web applications.

  Install React.js, a JavaScript library for building user interfaces.

  - Create a new React app: npm create vite@latest Enter and then type project-name and select preferred frameworks and then enter
  - Navigate to the project directory: cd project-name npm install
  - Running the React App: With the React app created, you can now start the development server and see your React application in action.
  - Start the development server: npm run devThis command launches the development server, and you can access your React app at http://localhost:5173 in your web browser.

- **HTML, CSS, AND JAVASCRIPT:**

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

- **VERSION CONTROL:**

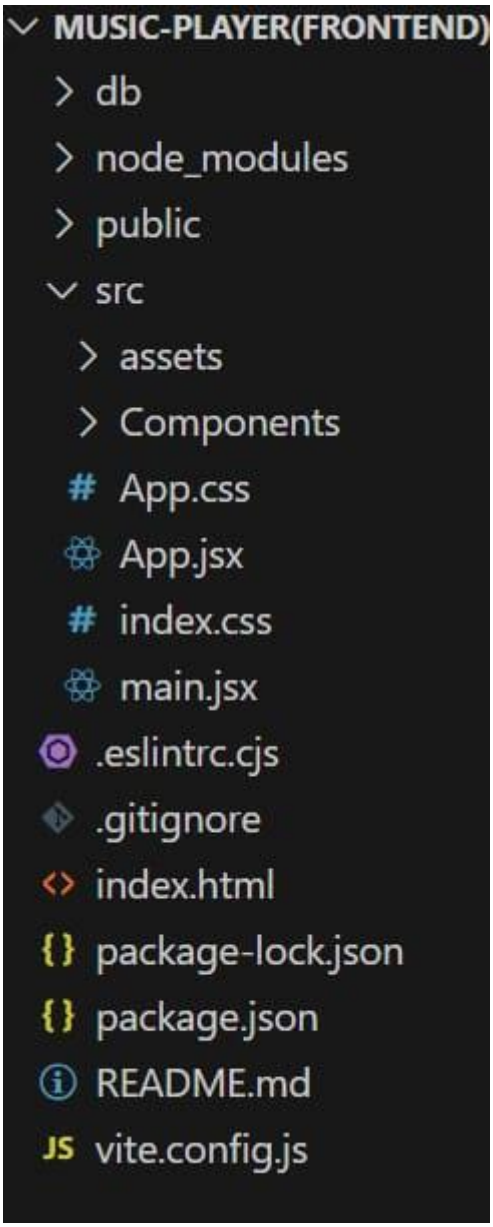
  Use Git for version control, enabling collaboration and tracking changes throughout the development process. Platforms like GitHub or Bitbucket can host your repository.

- **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
  - Sublime Text: Download from https://www.sublimetext.com/download
  - WebStorm: Download from https://www.jetbrains.com/webstorm/download

FOLDER 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.

app/app.component.css, src/app/app.component: These files are part of the main App Component, which serves as the root component for the React app.The component handles the overall layout and includes the router outlet for loading different components based on the current route.

CODE

```
import React, { useState, useEffect } from 'react';
import { Button, Form, InputGroup } from 'react-bootstrap';
import { FaHeart, FaRegHeart, FaSearch } from 'react-icons/fa';
import axios from 'axios';
import './sidebar.css'

function Songs() {
  const [items, setItems] = useState([]);
  const [wishlist, setWishlist] = useState([]);
  const [playlist, setPlaylist] = useState([]);
  const [currentlyPlaying, setCurrentlyPlaying] = useState(null);
  const [searchTerm, setSearchTerm] = useState('');

  useEffect(() => {
    // Fetch all items
    axios.get('http://localhost:3000/items')
      .then(response => setItems(response.data))
      .catch(error => console.error('Error fetching items: ', error));

  // Fetch favorities items
      axios.get('http://localhost:3000/favorities')
      .then(response => setWishlist(response.data))
      .catch(error => {
        console.error('Error fetching Favvorities:', error);
        // Initialize wishlist as an empty array in case of an error
        setWishlist([]);
      });

    // Fetch playlist items
    axios.get('http://localhost:3000/playlist')
      .then(response => setPlaylist(response.data))
      .catch(error => {
        console.error('Error fetching playlist: ', error);
        // Initialize playlist as an empty array in case of an error
        setPlaylist([]);
      });
      // Function to handle audio play
    const handleAudioPlay = (itemId, audioElement) => {
        if (currentlyPlaying && currentlyPlaying !== audioElement) {
          currentlyPlaying.pause(); // Pause the currently playing audio
        }
        setCurrentlyPlaying(audioElement); // Update the currently playing audio

      };
```

```
    // Event listener to handle audio play
    const handlePlay = (itemId, audioElement) => {
      audioElement.addEventListener('play', () => {
        handleAudioPlay(itemId, audioElement);
      });
    };

    // Add event listeners for each audio element
    items.forEach((item) => {
      const audioElement = document.getElementById(`audio-${item.id}`);
      if (audioElement) {
        handlePlay(item.id, audioElement);
      }
    });

    // Cleanup event listeners
    return () => {
      items.forEach((item) => {
        const audioElement = document.getElementById(`audio-${item.id}`);
        if (audioElement) {
          audioElement.removeEventListener('play', () => handleAudioPlay(item.id, audioElement));
        }
      });
    };
}, [items,currently Playing, searchTerm]);


const addToWishlist = async (itemId) => {
  try {
    const selectedItem = items.find((item) => item.id === itemId);
    if (!selectedItem) {
      throw new Error('Selected item not found');
    }
    const { title, imgUrl, genre, songUrl, singer, id: itemId2 } = selectedItem;
    await axios.post('http://localhost:3000/favorities', { itemId: itemId2, title, imgUrl, genre, songUrl, singer });
    const response = await axios.get('http://localhost:3000/favorities');
    setWishlist(response.data);
  } catch (error) {
    console.error('Error adding item to wishlist: ', error);
  }
};
```

```
return (
  <div style={{display:"flex", justifyContent:"flex-end"}}>
  <div className="songs-container" style={{width:"1300px"}}>
    <div className="container mx-auto p-3">
      <h2 className="text-3xl font-semibold mb-4 text-center">Songs List</h2>
      <InputGroup className="mb-3">
        <InputGroup.Text id="search-icon">
          <FaSearch />
        </InputGroup.Text>
        <Form.Control
          type="search"
          placeholder="Search by singer, genre, or song name"
          value={searchTerm}
          onChange={(e) => setSearchTerm(e.target.value)}
          className="search-input"
        />
      </InputGroup>
      <br />
      <div className="row row-cols-1 row-cols-md-2 row-cols-lg-3 row-cols-xl-4 g-4">
        {filteredItems.map((item, index) => (
          <div key={item.id} className="col">
            <div className="card h-100">
              <img
                src={item.imgUrl}
                alt="Item Image"
                className="card-img-top rounded-top"
                style={{ height: '200px', width: '100%' }}
              />
              <div className="card-body">
                <div className="d-flex justify-content-between align-items-center mb-2">
                  <h5 className="card-title">{item.title}</h5>
                  {isItemInWishlist(item.id) ? (
                    <Button
                      variant="light"
                      onClick={() => removeFromWishlist(item.id)}
                    >
                      <FaHeart color="red" />
                    </Button>
                  ) : (
                    <Button
                      variant="light"
                      onClick={() => addToWishlist(item.id)}
                    >
```

```
                <FaRegHeart color="black" />
              </Button>
            )}
          </div>
          <p className="card-text">Genre: {item.genre}</p>
          <p className="card-text">Singer: {item.singer}</p>
          <audio controls className="w-100" id={`audio-${item.id}`} >
            <source src={item.songUrl} />
          </audio>
        </div>
        <div className="card-footer d-flex justify-content-center">
          {isItemInPlaylist(item.id) ? (
            <Button
              variant="outline-secondary"
              onClick={() => removeFromPlaylist(item.id)}
            >
              Remove From Playlist
            </Button>
          ) : (
            <Button
              variant="outline-primary"
              onClick={() => addToPlaylist(item.id)}
            >
              Add to Playlist
            </Button>
          )}
        </div>
      </div>
    </div>
  ))}
 </div>
 </div>
 </div>
 </div>
 );
}

export default Songs;
```

TESTING

Tools:

- React js
- React router dom
- React icons
- Bootstrap/tailwind css
- Axios

Project Demo link:

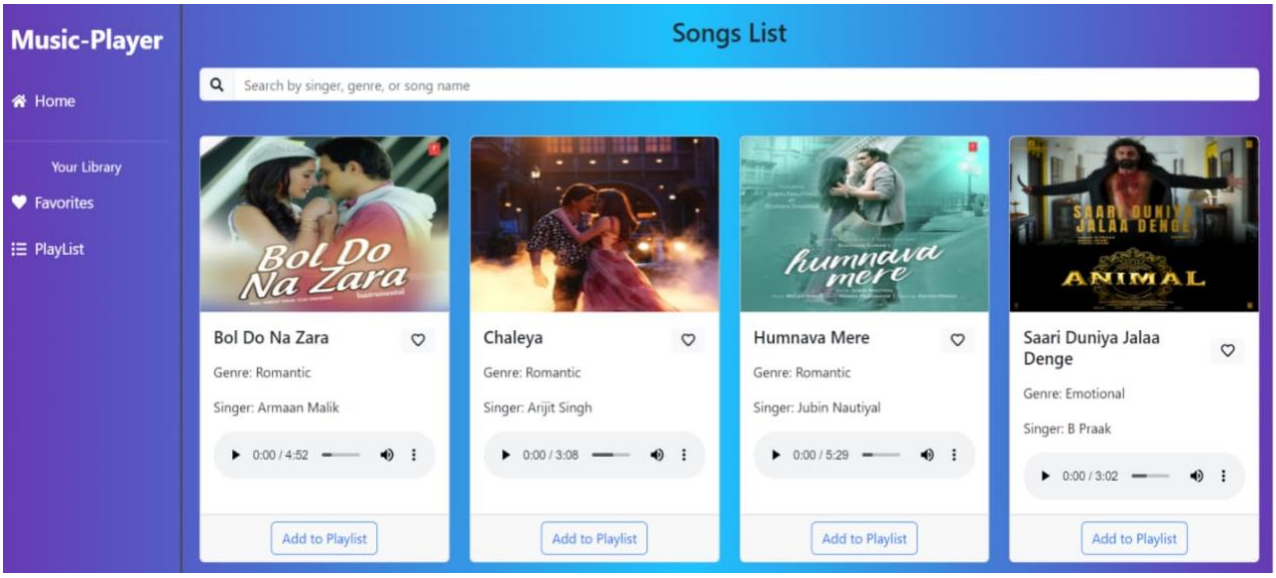
- **https://drive.google.com/file/d/11XgRsJP0h1s1vP_xPjgEkKw_lLKBQkW6/view?usp=drivesdk**

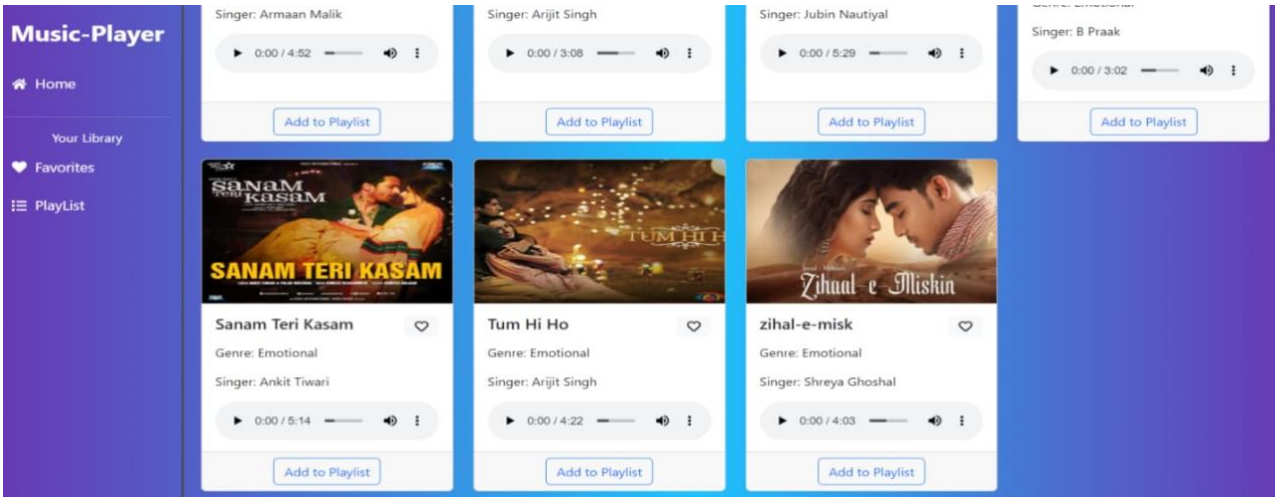
SCREENSHOTS

PROJECT EXECUTION:

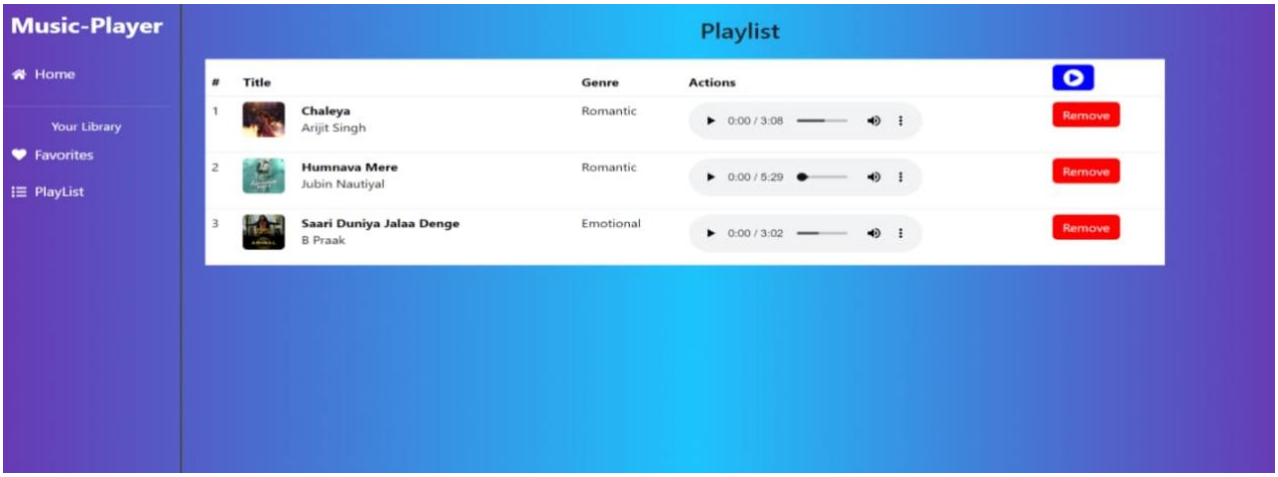
After completing the code, run the react application by using the command "npm Start" or "npm run dev" if you are using vite.js And the Open new Terminal type this command "json-server –watch ./db/db.json" to Start the Json server too.After that launch the Rythimic Tunes. Here are some of the screenshots of the application.

- **HOME PAGE:**

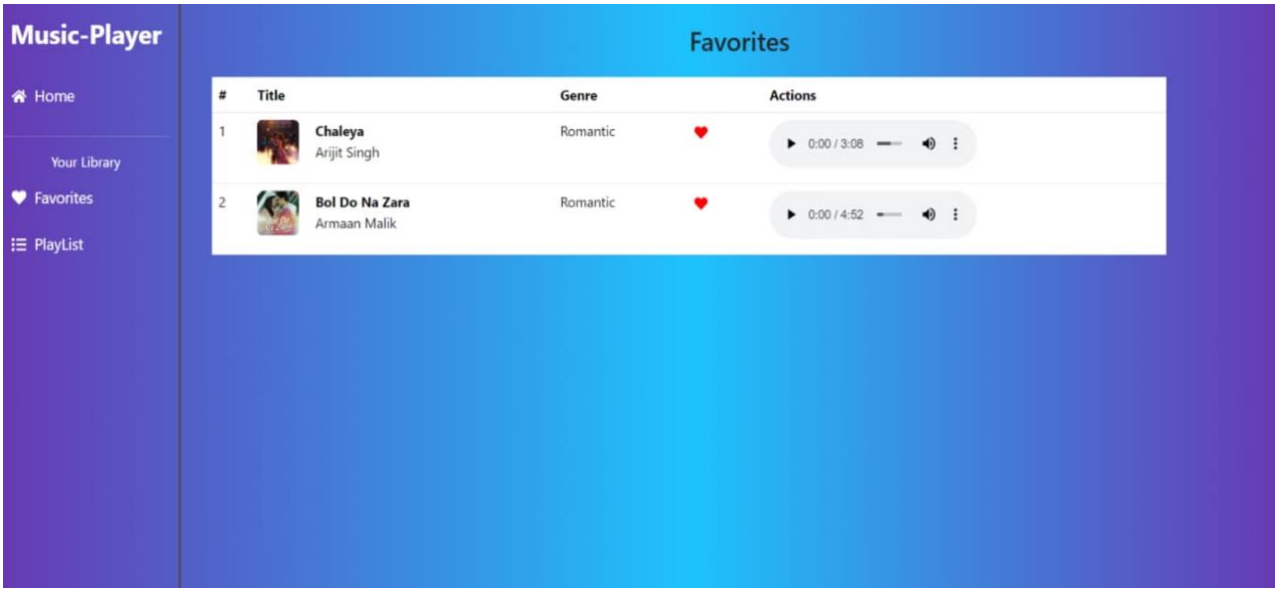




- **PLAYLIST:**



- **FAVORITES:**



FUTURE ENHANCMENTS

- AI-Driven Emotion Recognition: Automatically adapt playlists based on user mood.
- Smart Device & Wearable Integration: Sync music with workouts, daily activities, and IoT devices.
- VR/AR Music Experiences: Immersive virtual concerts and interactive rhythm visualization.
- Collaborative Playlists & Social Sharing: Create group playlists and host live listening sessions.
- Wellness Tools: Guided meditation, focus tracks, and sleep soundscapes.

KNOWN ISSUES

- Occasional app crashes during playlist creation.
- Limited offline playback support.
- Minor delays in loading large music libraries.
- Compatibility issues on older devices.