

Frontend Development with React.js

Project Documentation – Rhythmic Tune

1. Introduction

- o Project Title: Rhythmic Tune
- o Team Members:
 - Gopika .L(Code execution)
 - Kaviya .M(Demo video)
 - Kavipriya .P(Demo video)
 - Monika .S(Documentation)

2. Project Overview

Purpose:

Rhythmic Tune is a music-based web application that allows users to explore, play, and organize their favorite tunes. It provides a smooth, interactive, and visually appealing interface for discovering songs and creating playlists.

Features:

- o Music player with play, pause, next, and previous controls
- o Playlist creation and management
- o Search functionality for tracks/artists
- o Responsive UI for mobile and desktop
- o Dark/Light mode support

3. Architecture

- o Component Structure:
 - App.js – Root component
 - Navbar.js – Navigation bar
 - MusicPlayer.js – Core music player controls
 - Playlist.js – Playlist management
 - SearchBar.js – Search functionality
 - SongCard.js – Individual song display
- o State Management:
 - Context API is used for global state (e.g., currently playing song,

playlists).

- o Routing:

React Router is used with routes such as:

- Home (trending tunes)
- playlist – User playlists
- search – Search result

4. Setup Instructions

- o Prerequisites:

- Node.js, npm, Git

- o Installation:

- git clone <https://github.com/your-repo/rhythmic-tune.git>
- cd rhythmic-tune
- npm install
- npm start

5. Folder Structure

- o Client:

- src/
- components/
- Navbar.js
- MusicPlayer.js
- Playlist.js
- SongCard.js
- pages/
- Home.js
- Search.js
- Playlist.js
- assets/
- images/
- icons/
- utils/
- helpers.js

- o Utilities:

- Helper functions for API calls and reusable hooks.

6. Running the Application

- o Frontend:

- npm start

7. Component Documentation

- o Key Components:

- MusicPlayer – Handles audio controls, progress bar, volume.
- Playlist – Stores and displays songs added by the user.
- SearchBar – Allows searching for tracks/artists.

- o Reusable Components:

- SongCard – Displays song details consistently across pages.
- Button – Custom reusable button component.

8. State Management

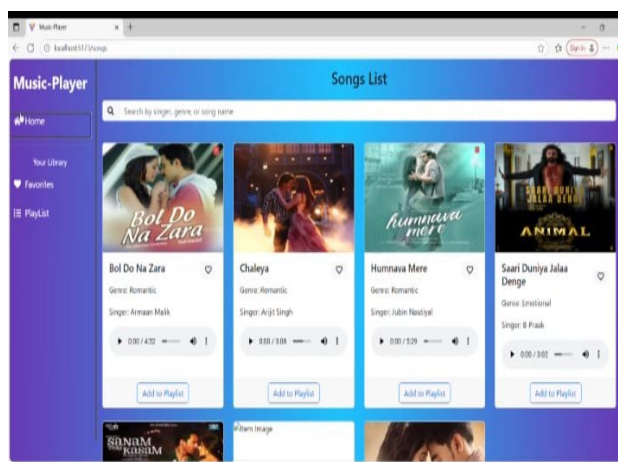
- o Global State:

- Current track, playlist data, theme (dark/light).

- o Local State:

- Input fields for search, toggle states for UI elements.

9. User Interface



10. Styling

- o CSS Frameworks/Libraries:

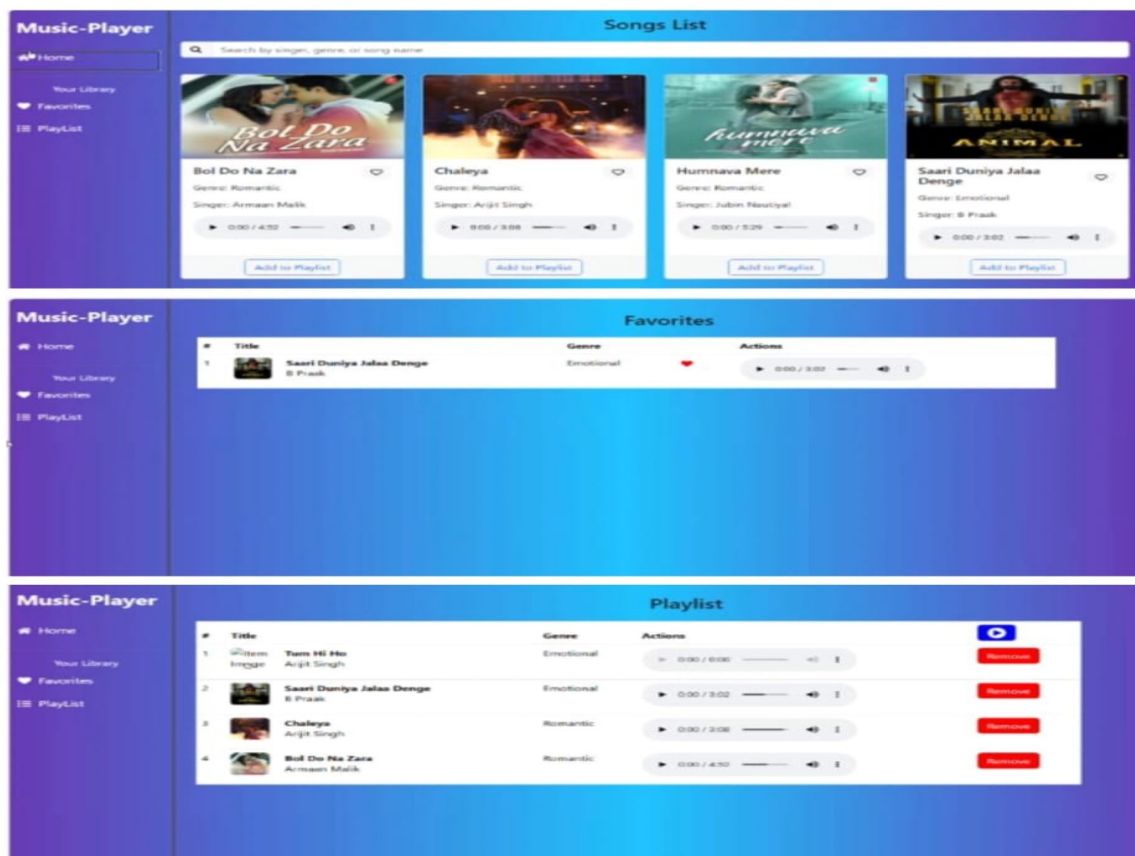
- Tailwind CSS for styling
- Styled-Components for scoped CSS.

- o Theming:
 - Dark/Light theme toggle with persistent local storage.

11. Testing

- o Testing Strategy:
 - Unit testing with Jest for core functions
 - Component testing with React Testing Library
 - Integration testing for player and playlist
- o Code Coverage:
 - Measured using Jest coverage tools.

12. Screenshots or Demo



13. Known Issues

- o Limited offline support.
- o Audio may lag on very low-end devices.
- o Currently supports only basic playlist features (no sharing)

14. Future Enhancements

- o Add user authentication for personalized playlists.

- o Support for offline playback.
- o Integration with third-party music APIs (Spotify, SoundCloud).
- o Advanced audio visualizations and animations.
- o Social features – share playlists with friends.