

Project Documentation

1. Introduction

- o Project Title : Rhythmic Tunes: your melodic companion
- o TEAM ID : NMNM2025TMID48641
- o TEAM LEADER: : VIDHYA D
MAIL ID: kumarpvk12@gmail.com
- o ROLE : CODING AND DEVELOPMENT
- o TEAM MEMBER : VIKASHINI D ; MAIL ID: vikashini220307@gmail.com
- o ROLE : CODING AND DEVELOPMENT
- o TEAM MEMBER : YUVAMALA M
MAIL ID : yuvamala025@gmail.com
- o ROLE : DEMO VIDEO
- o TEAM MEMBER : SUMITHRA M ; MAIL ID : sumithra952007@gmail.com
- o ROLE : DOCUMENT CREATER

2. Project Overview

- o Purpose: The purpose of the Rhythmic Tunes Music Player is to provide a user-friendly application for playing music..
- o Features: Key features include: music playback, playlist creation, and search functionality.

3. Architecture

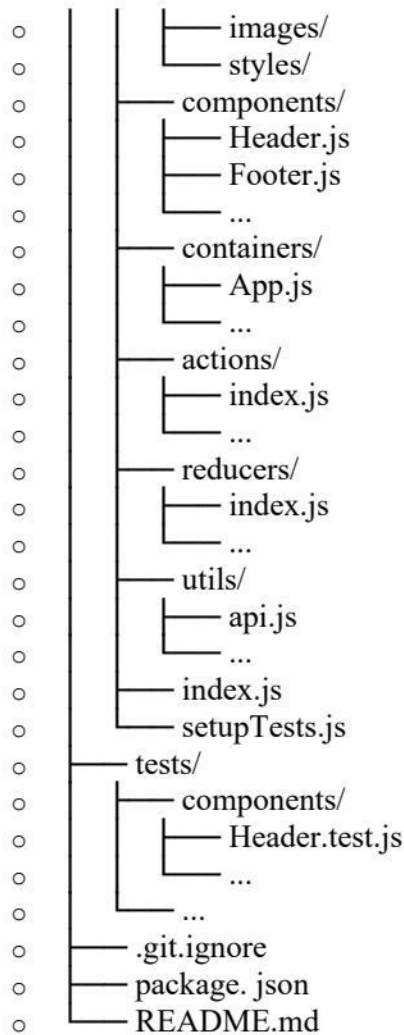
- o Component Structure: Key features include: music playback, playlist creation, and search functionality.
- o State Management: The application uses the Context API for global state management to handle the current song, playback status, and user-created playlists.
- o Routing: React Router is used to manage navigation between different views, such as the home page and the playlist page

4. Setup Instructions

- o Prerequisites: The required software dependencies are Node.js and npm ,React.js.
- o Installation: Provide a step-by-step guide to Clone the repository from GitHub. Navigate to the client directory.
- o * Run npm install to install dependencies.
- o * Configure environment variables for any API keys

5. Folder Structure

- o Client: The main React application is organized into folders such as components, pages, and assets for images and audio files.
- o Utilities: Helper functions for music playback and custom hooks for managing audio state are located in the utils folder
- o Code:
 - o project-name/
 - o public/
 - o index.html
 - o favicon.ico
 - o src/
 - o assets/



6. Running the Application

- Provide commands to start the frontend server locally.
 - **Frontend:** npm start in the client directory.

7. Component Documentation

- **Key Components:** Manages music playback controls. It receives a song prop.
- * Playlist Component: Displays the list of songs and manages playlist creation.
- * Search Component: Handles searching for songs
- **Reusable Components:** Button Component: A configurable button component used throughout the application.
- * Song Card Component: A component for displaying individual songs in lists.

8. State Management

- **Global State:** The Context API is used to manage the global state, ensuring that the current song and playback status are accessible to all components..
- **Local State:** se use State hooks to manage their own local state, such as form inputs or UI toggles

9. User Interface

- Provide screenshots or GIFs showcasing different UI features, such as pages, forms, or interactions.

-

10. Styling

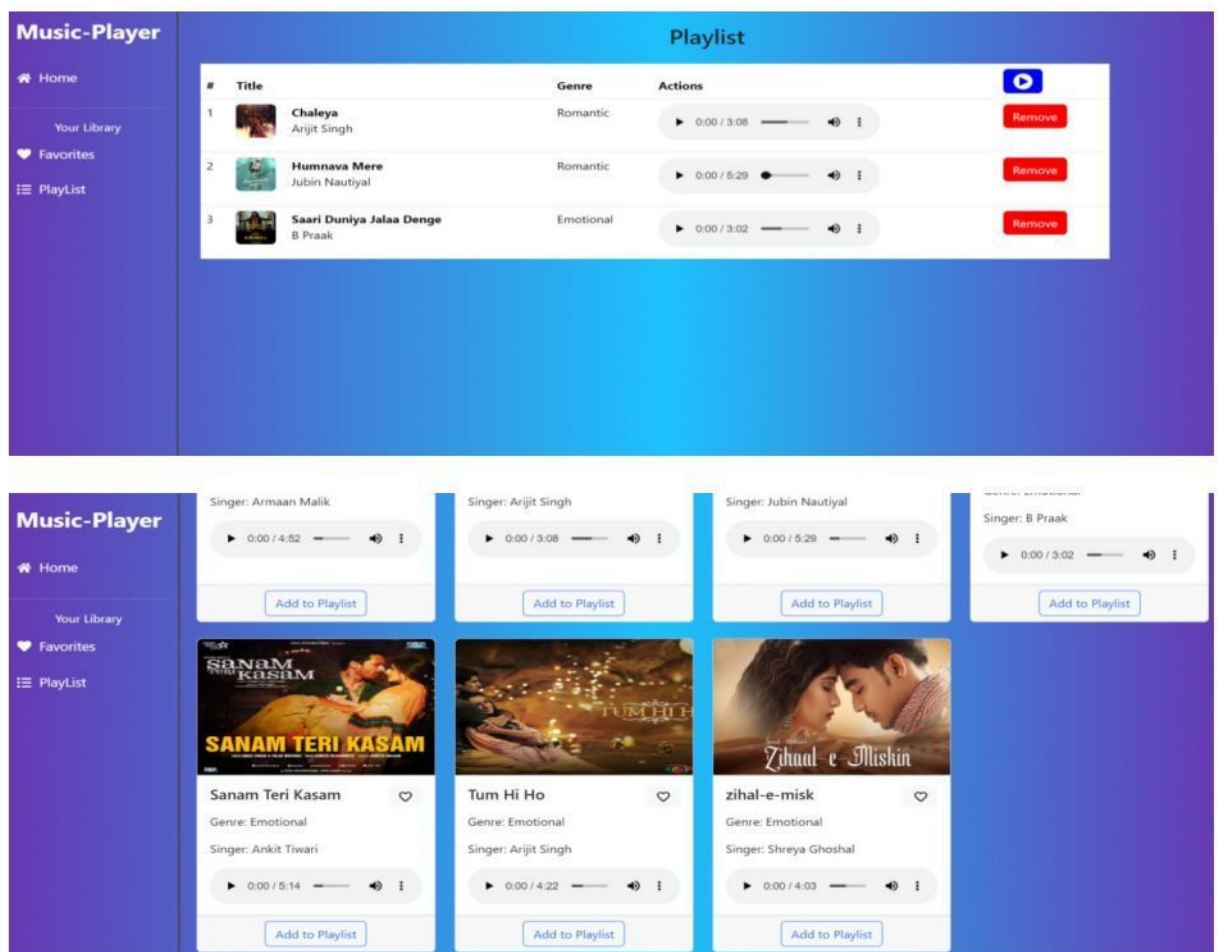
- **CSS Frameworks/Libraries:** Describe any CSS frameworks, libraries, or pre-processors (e.g., Sass, Styled-Components) used.
- **Theming:** A custom design system with light and dark themes is implemented

11. Testing

- **Testing Strategy:** The application uses Jest and React Testing Library for unit and integration testing of components.

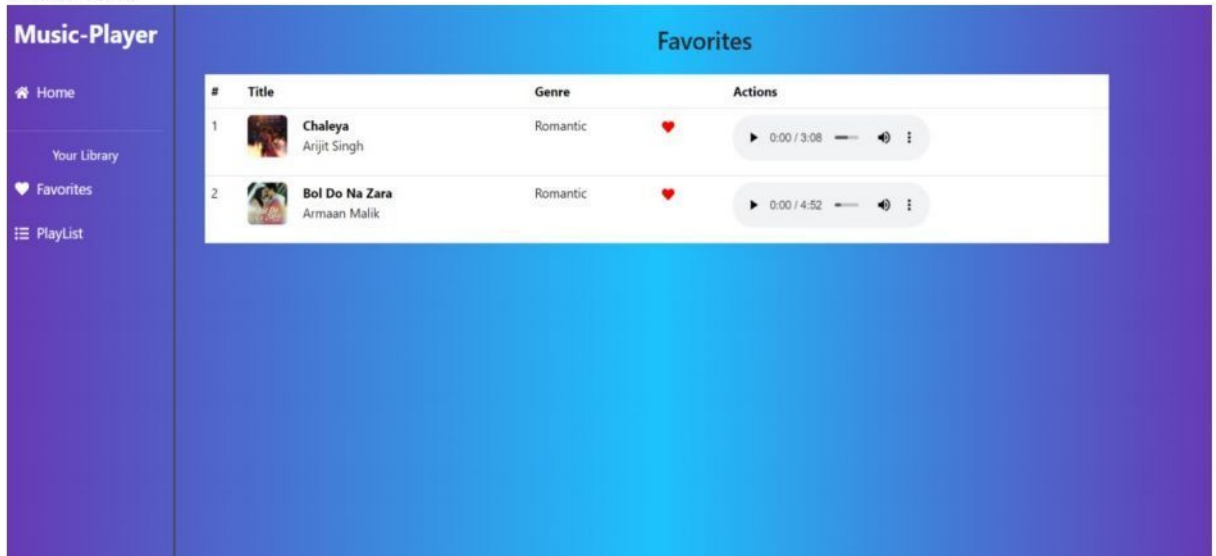
12. **Code Coverage:** Code coverage is tracked using a tool to ensure all key components are adequately tested.

13. Screenshots or Demo:



-

.favorties



13. Known Issues

- Any known bugs or issues, such as occasional playback glitches on certain browsers, should be documented here

14. Future Enhancements

- Potential future features could include a user authentication system, enhanced search functionality, and adding more animations to the user interface.

15. DEMOLINK:

- https://drive.google.com/file/d/1DozFUlo3wyvhVFyyd-E3alVq5Dy_BK54/view?usp=sharing