Frontend Development with React.js Project Documentation

# Introduction

* + **Project Title :** Rhythmic Tunes: your melodic companion
  + **TEAM ID : NM2025TMID41963**
  + **TEAM LEADER: :** SANTHIYASRI R
  + **ROLE :** CODING AND DEVELOPMENT
  + **TEAM MEMBER: : SARUBALA B**
  + **ROLE :** UPLOADING
  + **TEAM MEMBER : SASIREKHA M**
  + **ROLE :** DEMO VIDEO
  + **TEAM MEMBER : SATHYA V**
  + **ROLE :** DOCUMENT CREATER

# Project Overview

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

# Architecture

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

# Setup Instructions

* + **Prerequisites**: The required software dependencies are Node.js and npm

,React.js.

* + **Installation**: Provide a step-by-step guide to Clone the repository from GitHub. Navigate to the client directory.
  + \* Run npm install to install dependencies.
  + \* Configure environment variables for any API keys

# Folder Structure

* + **Client**: The main React application is organized into folders such as components, pages, and assets for images and audio files.
  + **Utilities**: Helper functions for music playback and custom hooks for managing audio state are located in the utils folder

# Code:

* + project-name/
  + ├── public/
  + │ ├── index.html
  + │ └── favicon.ico
  + ├── src/
  + │ ├── assets/
  + │ │ ├── images/
  + │ │ └── styles/
  + │ ├── components/
  + │ │ ├── Header.js
  + │ │ ├── Footer.js
  + │ │ └── ...
  + │ ├── containers/
  + │ │ ├── App.js
  + │ │ └── ...
  + │ ├── actions/
  + │ │ ├── index.js
  + │ │ └── ...
  + │ ├── reducers/
  + │ │ ├── index.js
  + │ │ └── ...
  + │ ├── utils/
  + │ │ ├── api.js
  + │ │ └── ...
  + │ ├── index.js
  + │ └── setupTests.js
  + ├── tests/
  + │ ├── components/
  + │ │ ├── Header.test.js
  + │ │ └── ...
  + │ └── ...
  + ├── .git.ignore
  + ├── package. json
  + └── README.md

# Running the Application

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

# 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.

# 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

# User Interface

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

# 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

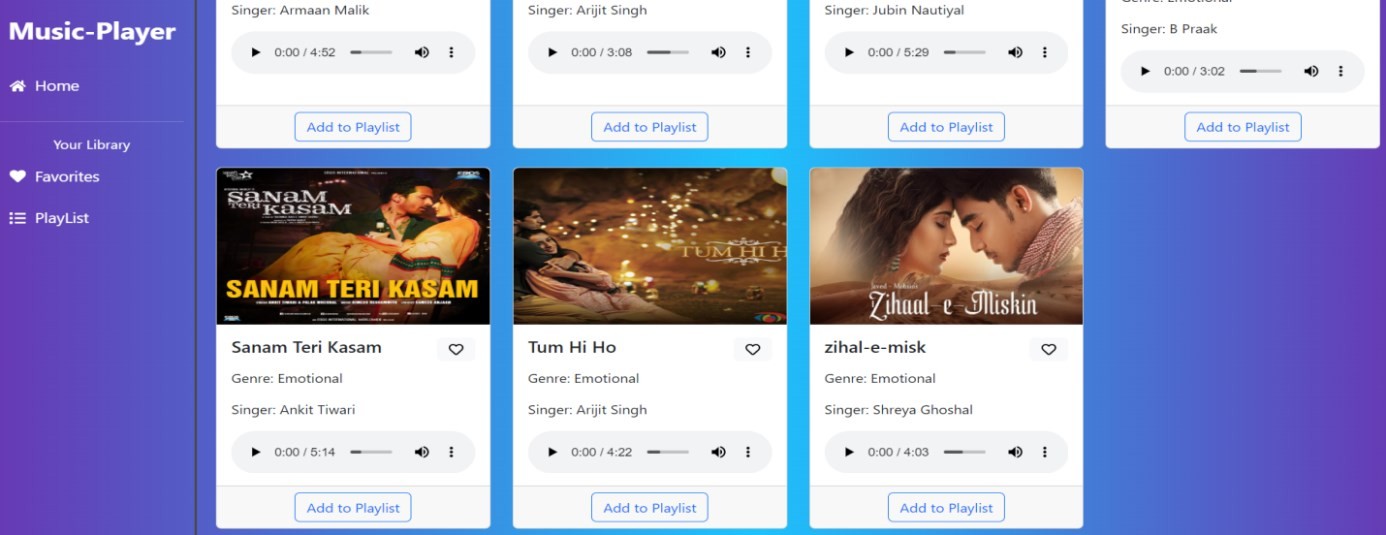
# Testing

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

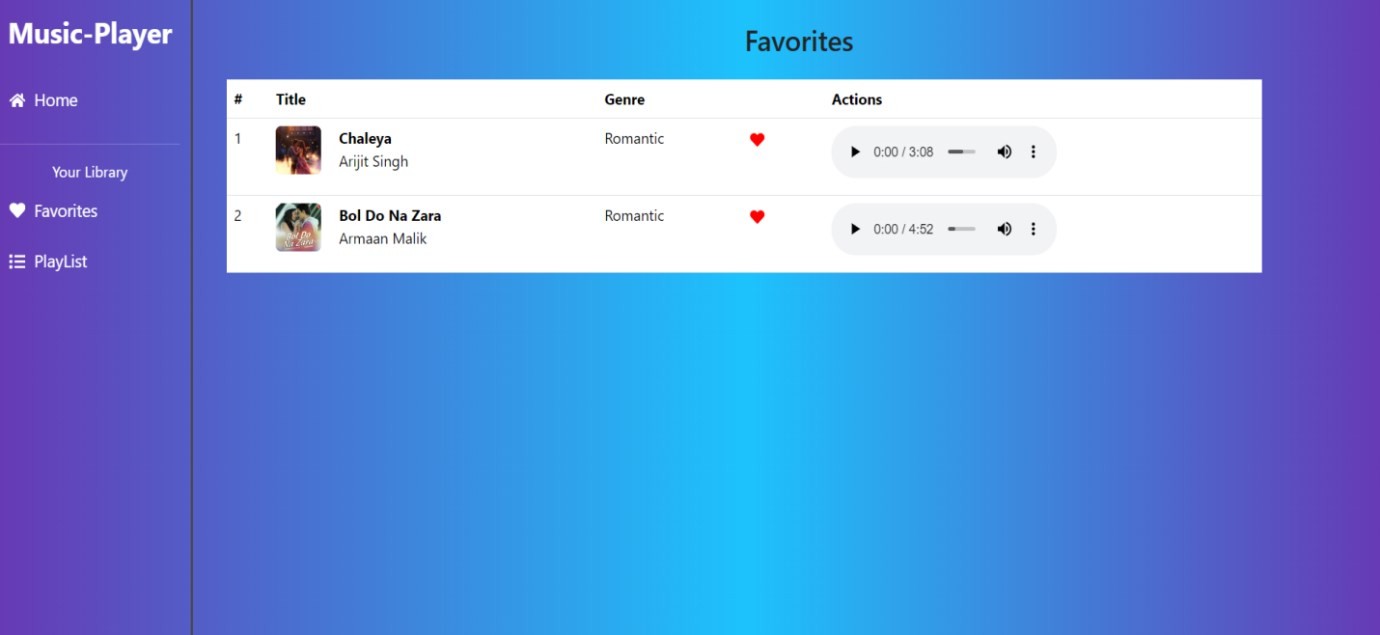
1. **Code Coverage**: Code coverage is tracked using a tool to ensure all key components are adequately tested.
2. **Screenshots or Demo:**

A screenshot of a playlist

AI-generated content may be incorrect.



**Favorites:**

****

1. **Known Issues**
   * Any known bugs or issues, such as occasional playback glitches on certain browsers, should be documented here

# Future Enhancements

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

# DEMOLINK:

* + https://drive.google.com/file/d/1ZLRccyfc5sx9d75jYc\_B7eH12Tcd7QPq/view?usp=drive\_link