Frontend Development with React.js

Project Documentation – Rhythmic Tune

1. Introduction

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

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:

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

3. Architecture

- 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
- State Management:
  - Context API is used for global state (e.g., currently playing song,

playlists).

- Routing:

  React Router is used with routes such as:

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

4. Setup Instructions

- Prerequisites:
  - Node.js, npm, Git
- Installation:

  •git clone - https://github.com/24bca04-lqtm/Rhythmic-Tunes-.git

  - cd rhythmic-tune
  - npm install
  - npm start

5. Folder Structure

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

- Utilities:
  - Helper functions for API calls and reusable hooks.

6. Running the Application

- Frontend:
  - npm start

7. Component Documentation

- Key Components:
  - MusicPlayer – Handles audio controls, progress bar, volume.
  - Playlist – Stores and displays songs added by the user.
  - SearchBar – Allows searching for tracks/artists.
- Reusable Components:
  - SongCard – Displays song details consistently across pages.
  - Button – Custom reusable button component.

8. State Management

- Global State:
  - Current track, playlist data, theme (dark/light).
- Local State:
  - Input fields for search, toggle states for UI elements.

9. User Interface

- Demo Link-https://drive.google.com/file/d/1ihSCO2s3YjHqM1rJ5WoSaYqydMJxJIc/view?usp=drive_link

10. Styling

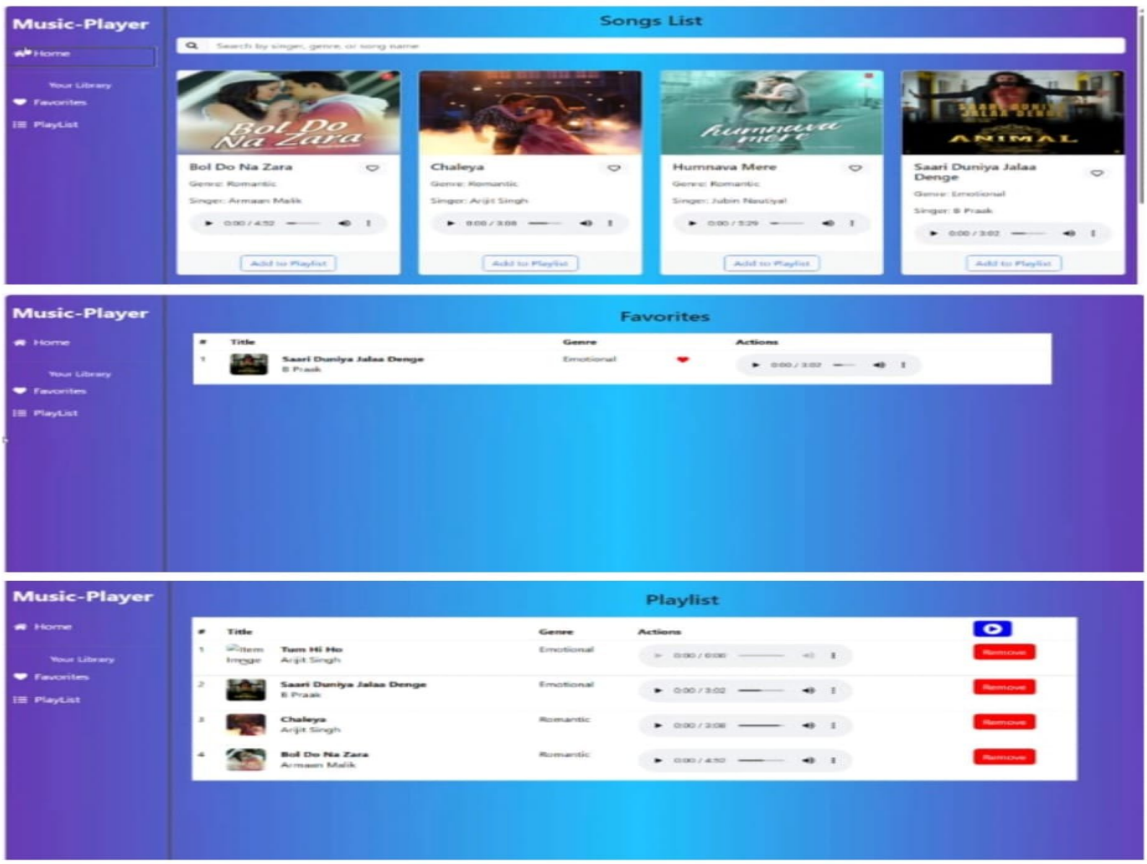
- CSS Frameworks/Libraries:
  - Tailwind CSS for styling
  - Styled-Components for scoped CSS.
- Theming:
  - Dark/Light theme toggle with persistent local storage.

11. Testing

- Testing Strategy:
  - Unit testing with Jest for core functions
  - Component testing with React Testing Library
  - Integration testing for player and playlist

- Code Coverage:
  - Measured using Jest coverage tools.

12. Screenshots or Demo



13. Known Issues

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

14. Future Enhancements

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