# Frontend Development with React.js

## Project Documentation

## Introduction

Your go-to music app for unlimited streaming, personalized playlists, and discovering new tunes. Enjoy music anywhere, anytime, with seamless access to all your favorite tracks!

## Project Title:

Rhythmic Tunes

## Team ID:

SWTID1741244291150149

## Team Members:

- Team Leader: Vijayalakshmi R  
- Team Member: Sharmila M  
- Team Member: Subetha J  
- Team Member: Swetha A  
- Team Member: Uma S

## Project Overview

## Purpose:

Rhythmic Tunes is a music streaming application designed to provide a seamless, personalized listening experience. It aims to simplify music discovery, enhance playback quality, and improve user engagement.

## Features:

- Music search & discovery   
- High-quality streaming with minimal buffering   
- Personalized playlists and recommendations   
- Playlist management &favoriting  
- Mobile-friendly, intuitive UI/UX

## Architecture

## Component Structure:

- Modular React components for scalability   
- Separation of concerns for UI, logic, and data handling

## State Management:

- Redux for global state management   
- Context API for lightweight state handling

## Routing:

- React Router for navigation between pages

## Setup Instructions

## Prerequisites:

- Node.js (latest stable version)   
- npm or yarn   
- React.js

## Installation:

1. Clone the repository   
2. Run `npm install` to install dependencies   
3. Configure environment variables (if needed)

## Folder Structure

* src/(Main source folder)
* components/ (Reusable UI components)
* pages/(Main application pages)
* assets/ (Images, icons, and other static files)
* utils/ (Helper functions and custom hooks)

## Running the Application

To start the frontend server:  
```  
npm start  
```

## Component Documentation

## Key Components:

* SearchBar:Allows users to search for songs, albums, and artists.
* Player: Handles audio playback, including play, pause, and skip.
* PlaylistManager:Manages user playlists and favorited songs.

## State Management

Global State: Redux manages user sessions, playback, and music library.   
Local State: Component-level state for UI elements (e.g., modal visibility).

## User Interface

Responsive design using Tailwind CSS &BootstrapDark mode and custom themes

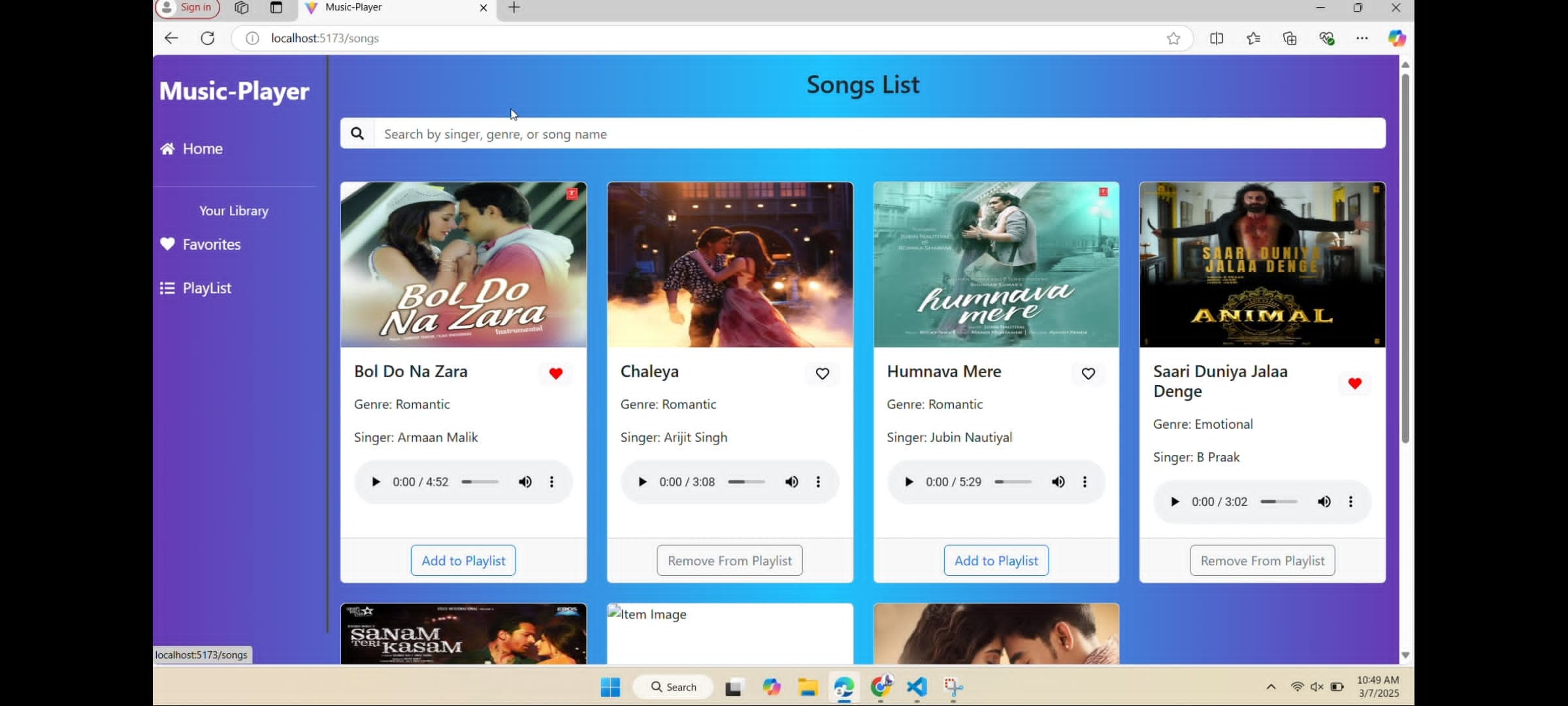
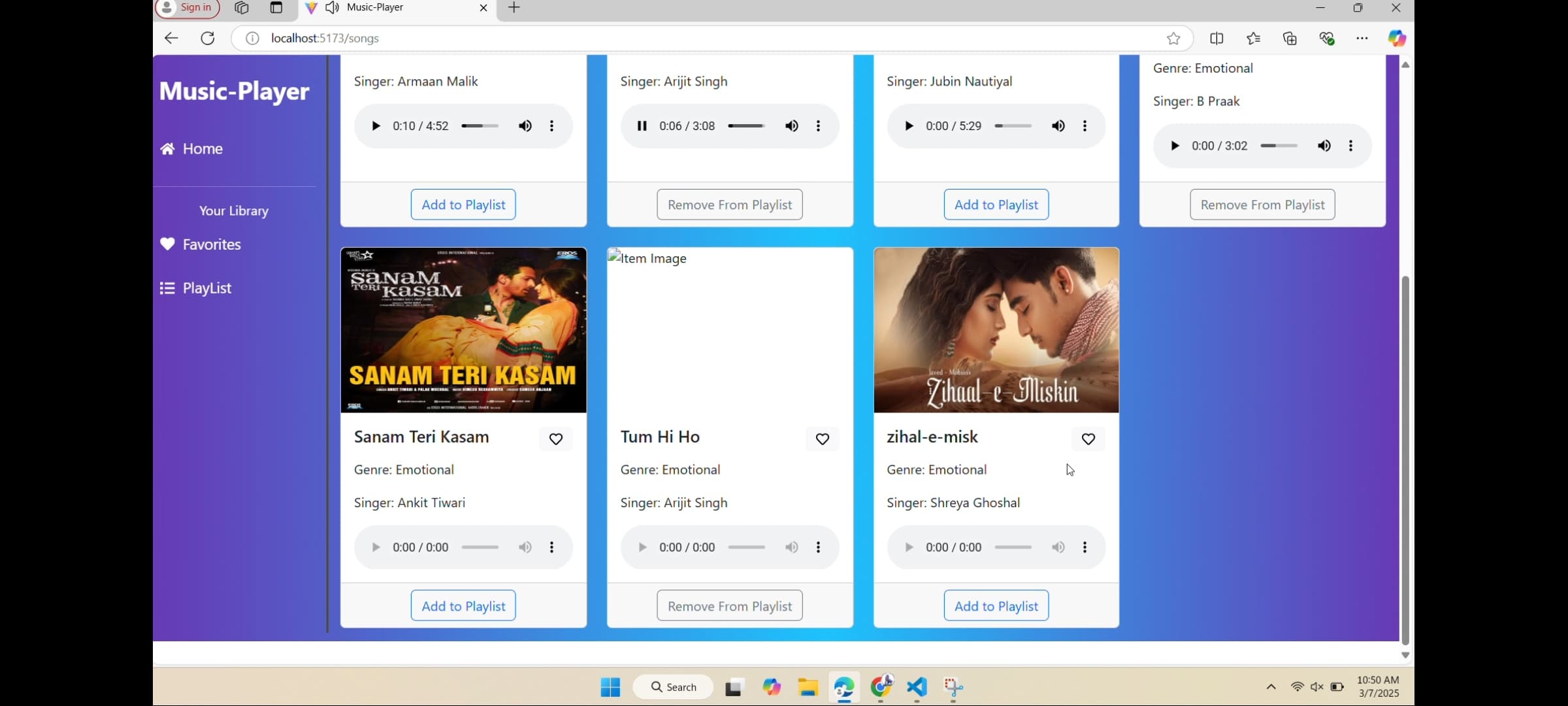
## Styling

- CSS-in-JS using Styled Components   
- Global styles managed via CSS modules

## Testing

- Jest & React Testing Library for unit and integration testing   
- Cypress for end-to-end testing

## Screenshots or Demo



## 

## 

## Known Issues

* No bugs founded till now

## Future Enhancements

- AI-powered recommendation system   
- Offline music playback feature   
- Social sharing and collaboration playlists