**FRONTEND DEVELOPMENT WITH REACT.JS**

1. **INTRODUCTION**

**Project Title**

**-** Rythmic Tunes Your Melodic Companion(Music Streaming App)

**Team Members:**

* Deepika A – Team Leader
* Priyadharshini B – Frontend development, UI/UX designing.
* Shanmugam – Database and server-side logic management.
* Karthikeyan – Quality assurance testing.

1. **PROJECT OVERVIEW:**

**Purpose:**

RythmicTunes is a dynamic web application designed to provide users with an immersive music experience. It offers a wide range of features including song playback, personalized playlists, and a favorites list to enhance user engagement with music.

**Features:**

Song Playback: Play a variety of songs directly on the platform.

Favorites List: Allows users to save and access their favorite tracks easily.

Personalized Playlists: Users can create, customize, and manage their own playlists.

1. **Architecture:**

**Component Structure:** The application is structured with a clear hierarchy of React components. The main components include:

* SongList: Displays a list of available songs.
* Playlists: Manages and displays user playlists.
* Favorites: Shows the user's favorite songs.
* These components interact to provide a seamless user experience.

**State Management:** We utilize React's Context API for efficient state management, allowing for easy prop drilling and maintaining a clean architecture.

**Routing:** React Router is employed for navigation, providing a smooth, single-page application feel.

1. **Setup Instructions:**

**Pre-requisites:**

Node.js, VScode.

**Installation:**

1. Install the Node.js, VScode to your local machine.
2. Navigate to the project directory in your terminal.
3. Run npm install npx to install all necessary dependencies.
4. Start the development server with npm start.
5. **Folder Structure:**

**Client:**

* Components: Contains all React components.
* Pages: Includes different pages of the application.
* Assets: Stores static files like images and stylesheets.

**Utilities:** Custom hooks and utility functions are located in the hooks directory to manage state and logic.

1. **Running the Application:**

To run the frontend server locally, use the command npm start in the client directory.

Commands used:

* cd db
* npm install npx
* npx json-server –watch db.json
* npm i
* npm run dev

1. **Component Documentation:**

**Key Components:**

* SongList: Renders a list of songs and handles playback.
* Playlists: Manages playlist creation, deletion, and display.
* Favorites: Displays and manages the user's favorite songs.

**Reusable Components:** Components like SongCard and Button are designed to be reusable across the application.

1. **State Management:**

**Global State:** Context API is used to manage global state, such as the currently playing song and user preferences.

**Local State:** Local state within components is managed using the useState hook for simplicity and efficiency.

1. **User Interface:**

The UI is designed to be intuitive, with easy navigation and clear visual cues for playback and playlist management.

1. **CSS:**

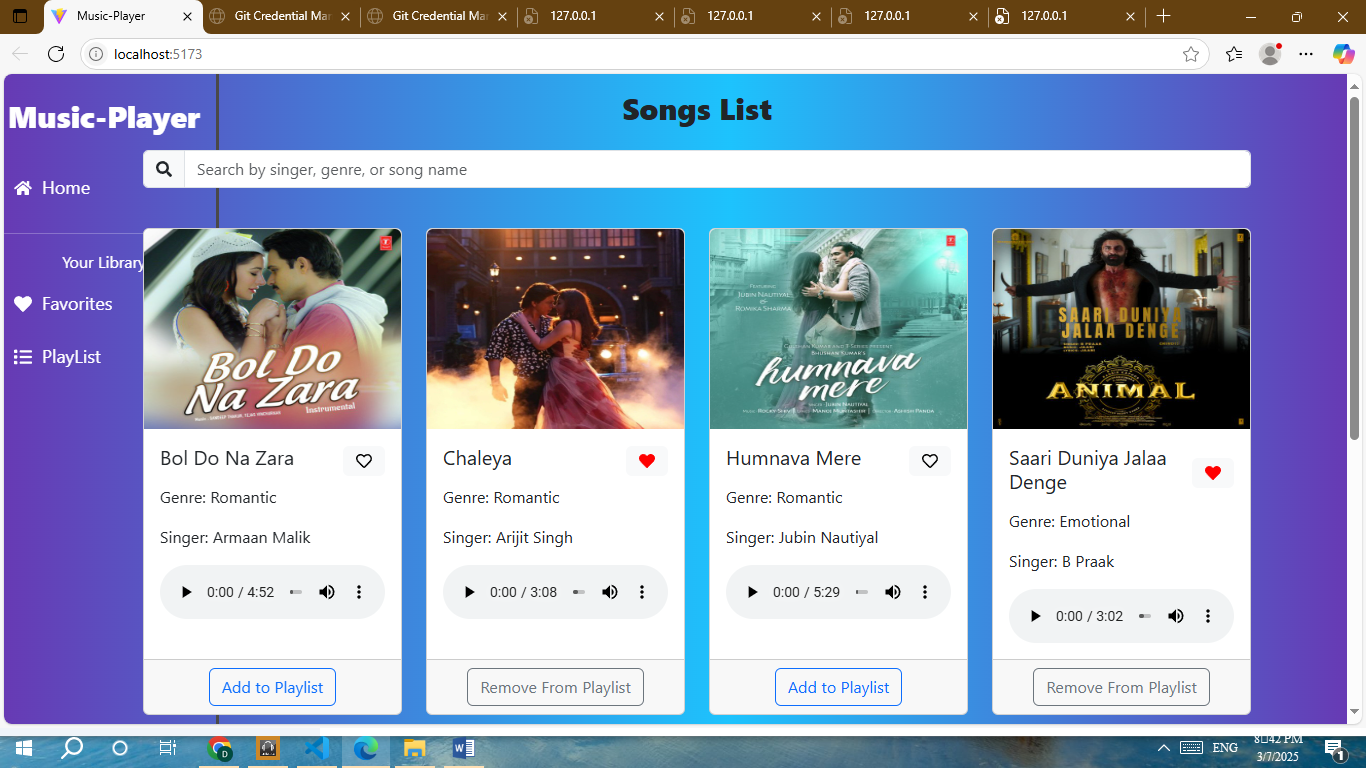
**CSS Frameworks/Libraries:** We use Material-UI for consistent and responsive styling.

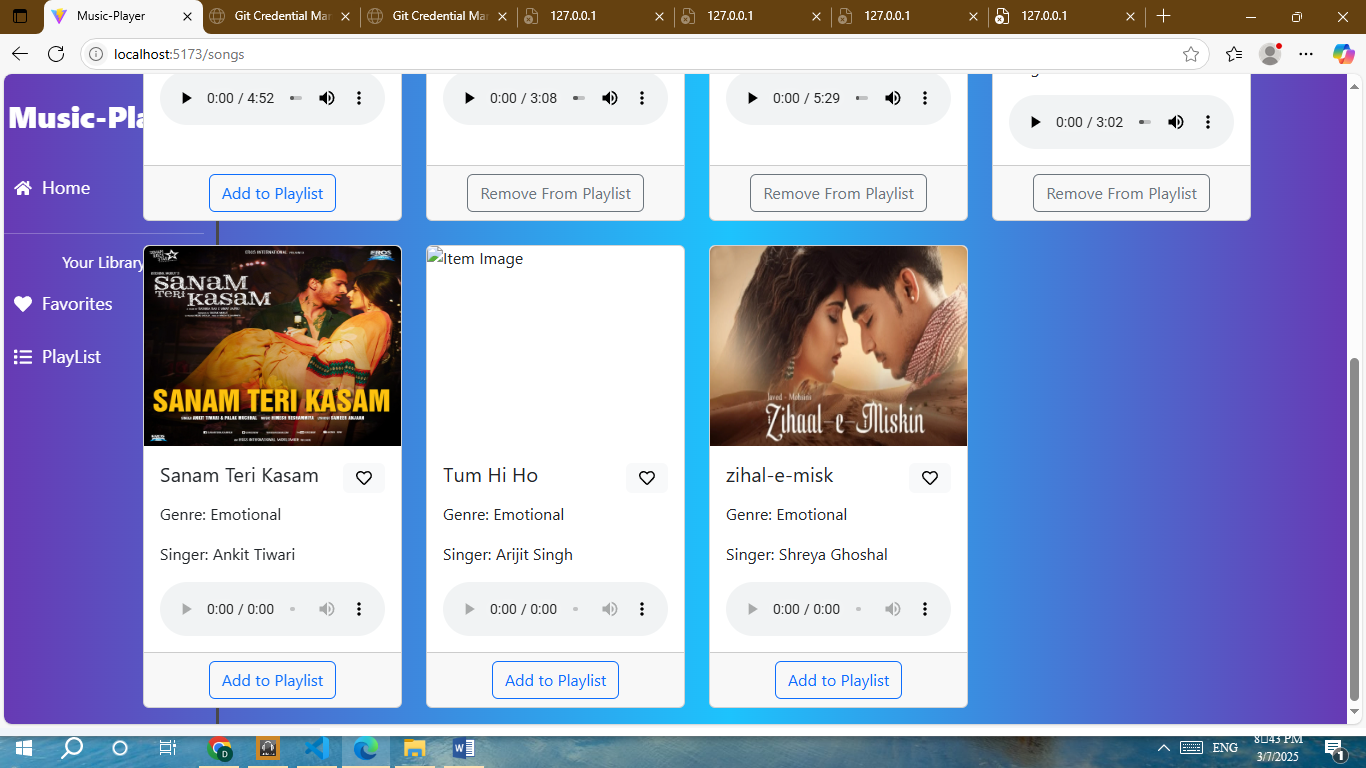
**Theming:** Custom themes are implemented to match the application's branding.

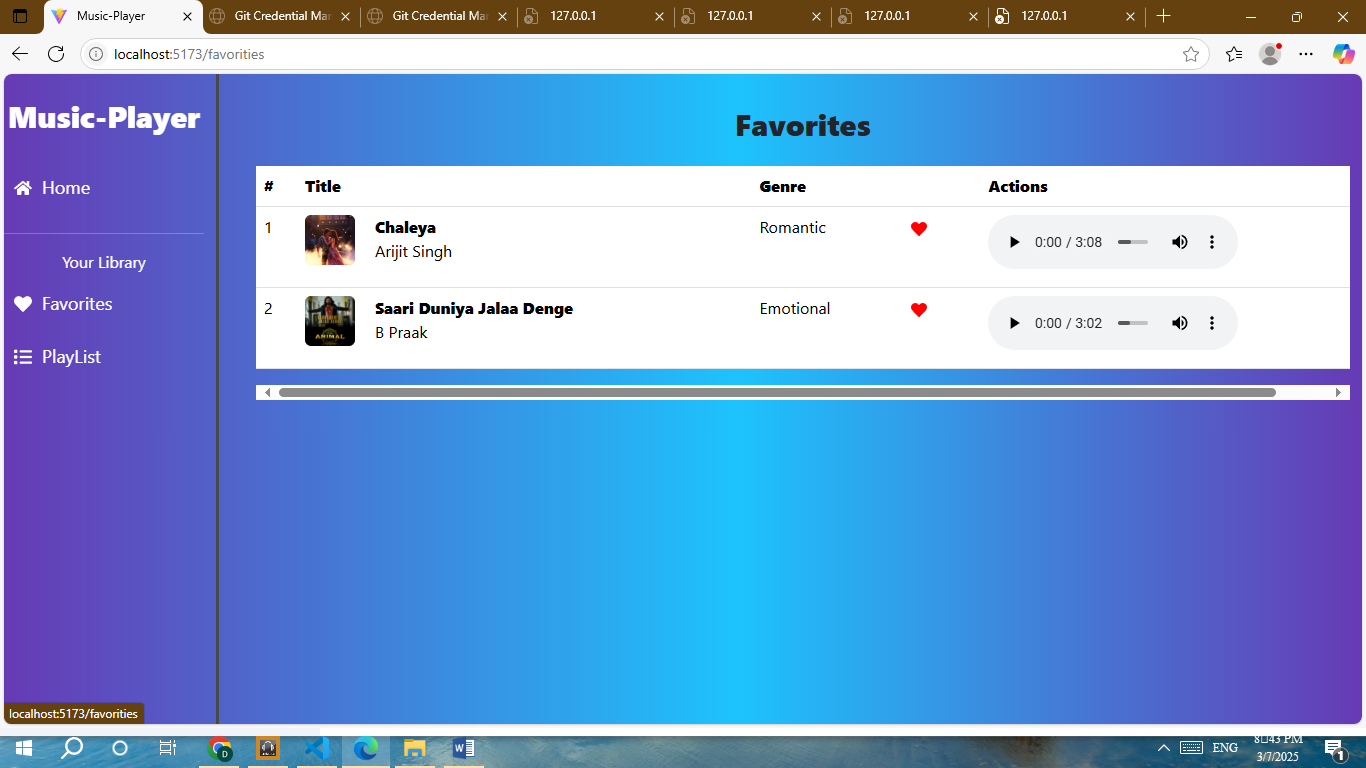
1. **Testing:**

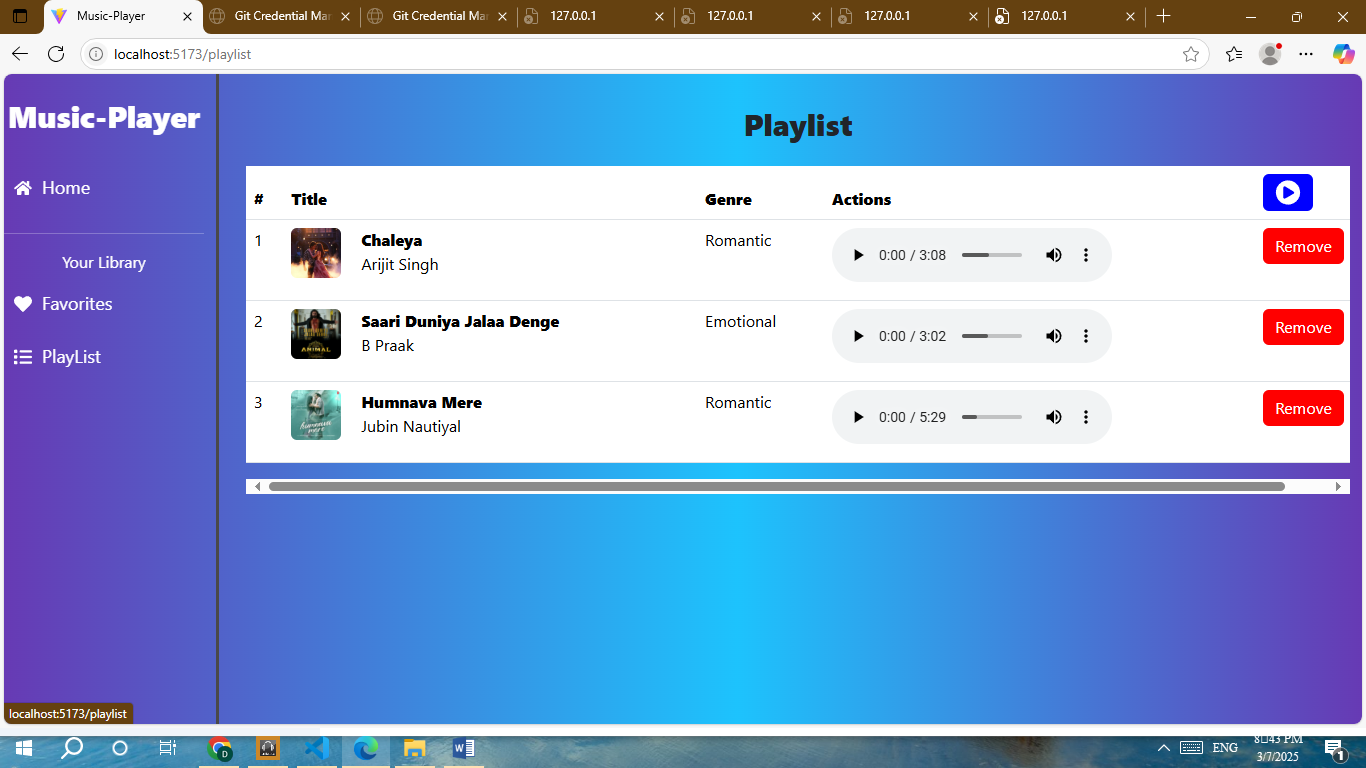
**Testing Strategy:** Jest and React Testing Library are used for unit and integration testing to ensure reliability.

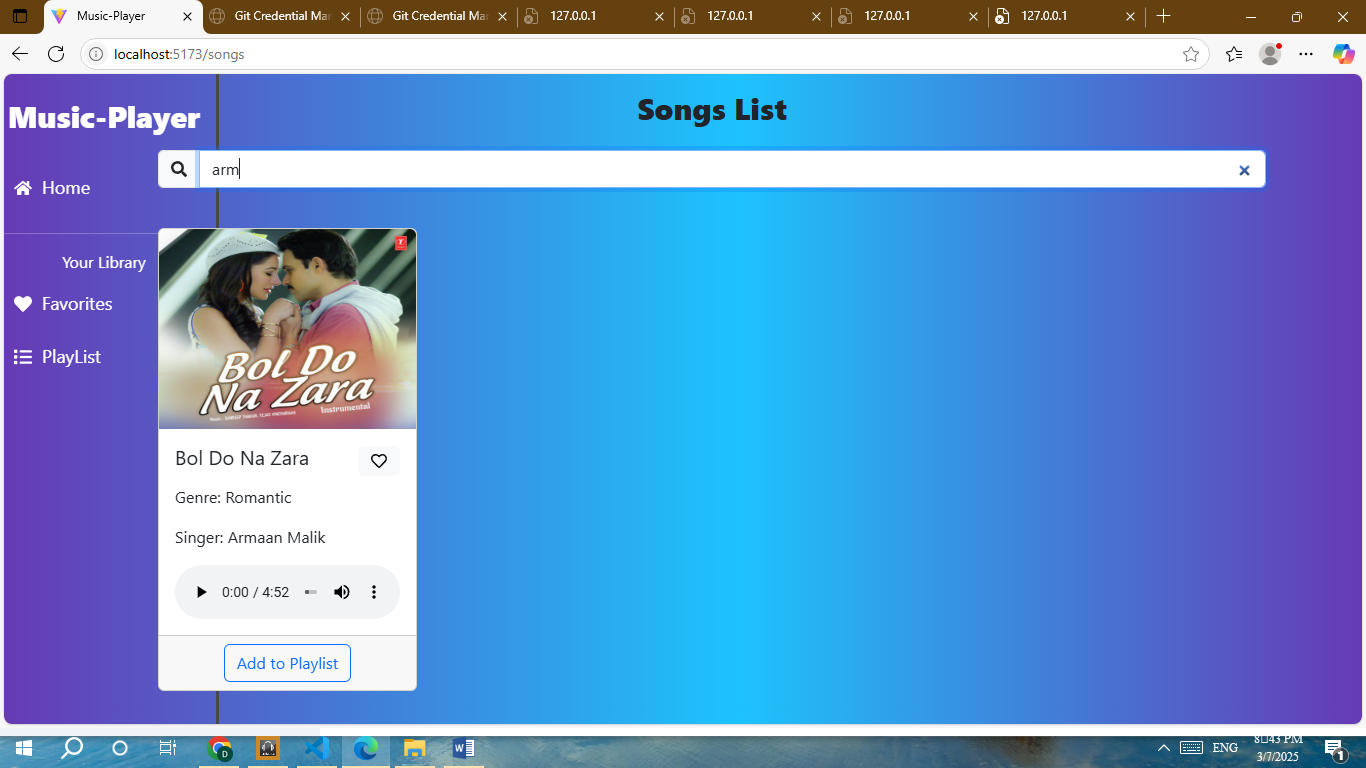
**Code Coverage:** We aim for high test coverage to maintain code quality.

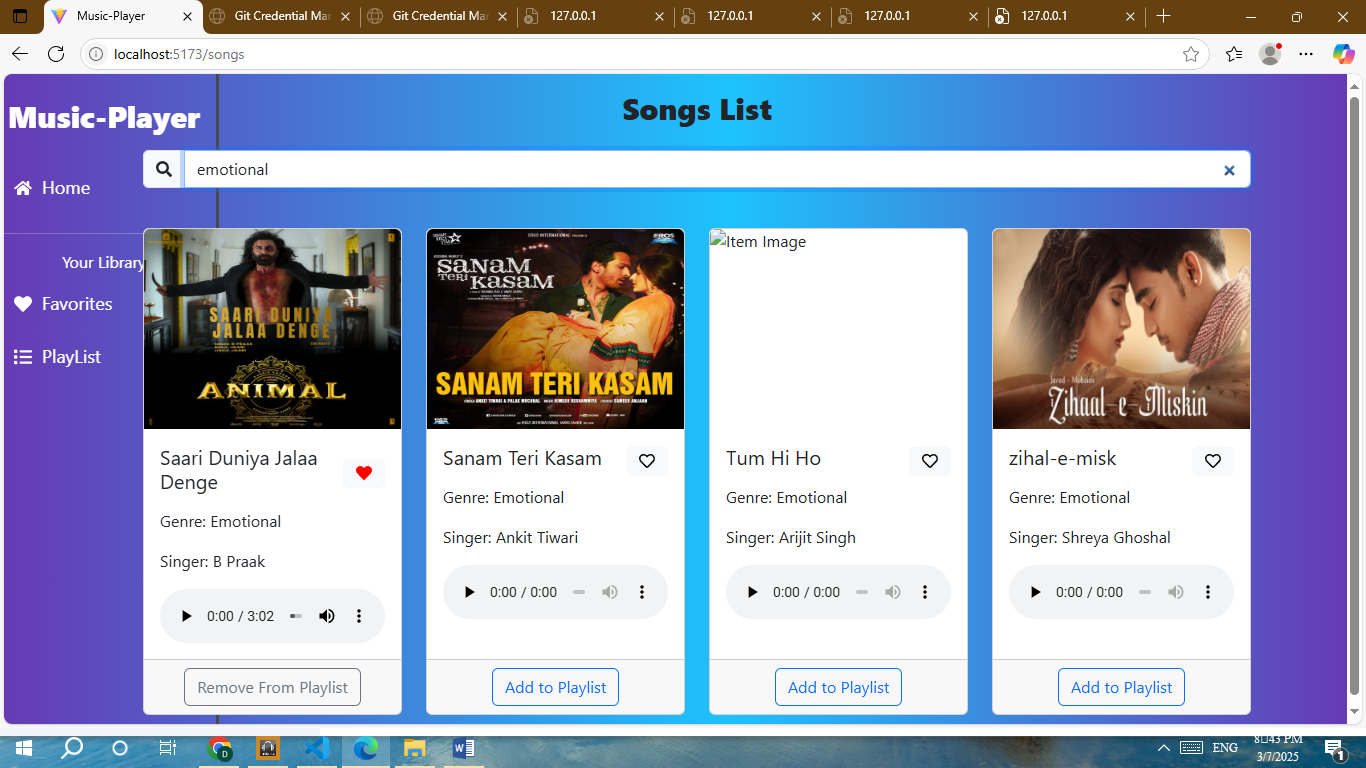
1. **Screenshots or Demo:**
2. ****

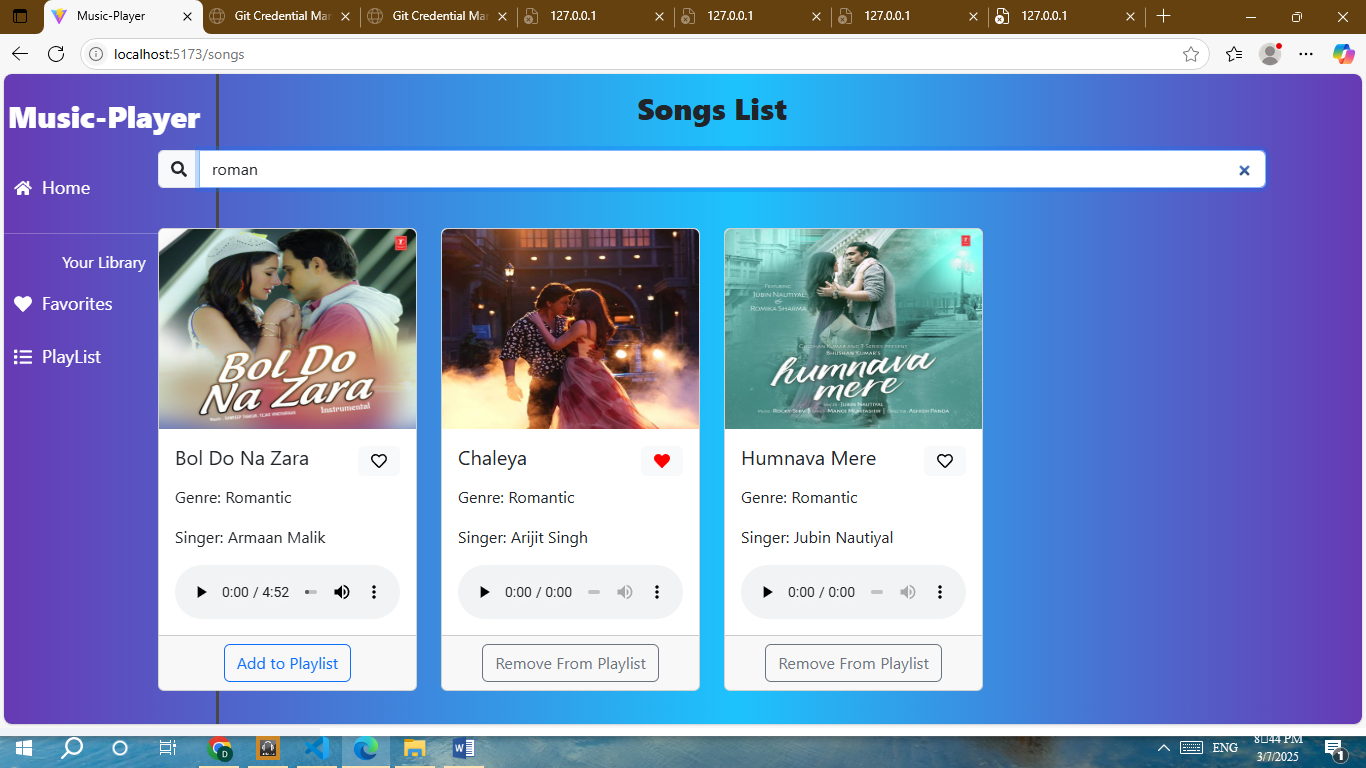
****











1. **Known Issues:**

Currently, there are no known major issues.

1. **Future Enhancements:**

Planned enhancements include adding user authentication, improving playlist management, and integrating with more music sources.