

Submitted by:

Soundarya Vaishnavi S

Rasila Banu M

Sriharini G

Sandhiya R

Shri Shankarlal Sundarbai Shasun Jain College for Women

“Where every beat tells a story.”

(Your Music Companion)

**Project Documentation**

*Rhythmic tunes (Your Music Companion)*

1. **Introduction**

**• Project Title**: Rhythmic tunes (Your Music Companion)

• Team ID: NM2025TMID37943

• **Team Leader**: Soundarya Vaishnavi S -[soundarya.vaishnavi2006@gmail.com](mailto:soundarya.vaishnavi2006@gmail.com)

**• Team Members**:

– Rasila Banu M – [banurasilabanu@gmail.com](mailto:banurasilabanu@gmail.com)

* Sriharini G – [sriharini1349@gmail.com](mailto:sriharini1349@gmail.com)
* Sandhiya R – [sandhiya442006@gmail.com](mailto:sandhiya442006@gmail.com)

**2. Project Introduction:**

Your Melody Companion is a front-end music streaming web application built with React.js, Vite, CSS, and Bootstrap.  
It allows users to stream songs, search by genre/singer/title, add to favourites, and build playlists with a clean and intuitive interface.

• It is a revolutionary music streaming app designed to elevate your listening experience.

• It offers an intuitive interface, powerful search capabilities and a vast library of track across all genres.

• It immerses users in an unparalleled musical journey effortlessly explore collection of songs, playlists with dynamic search bringing you the latest and most beloved tunes across the world.

• Embraces a diverse audience vibrant community by a shared love of music

**3.Architecture**

**Component Structure:**

App.jsx – Root component, sets up routing.

Navbar / Sidebar – Navigation between sections (Songs, Favorites, Playlist).

Songs Component – Displays all available tracks from db.json.

Favorites Component – Renders songs saved as favorites.

Playlist Component – Displays songs added to user’s playlist.

Player Component – Audio player with play/pause functionality.

SearchBar Component – Filters songs by singer, genre, or title.

**State Management:**

Local State: Managed using useState for search input, currently playing track, and form handling.

Data State: Synchronized with db.json using Axios and useEffect.

Global Sharing: Props drilling is used for passing state between components (React Context/Redux not implemented).

**Routing:**

**/ Songs Component**

**/favorities → Favorites Component**

**/playlist → Playlist Component**

**4. Why This Project?**

Music lovers face challenges such as:

* Difficulty managing large collections of songs
* Limited playlist customization in simple apps
* Complex user interfaces that reduce engagement

Rhythmic Tunes solves these by offering a lightweight, user-friendly music companion with playlists, favorites, and smooth playback.

**5.Objectives of the Project**

* **User-Friendly Interface**: Allow users to explore, save, and share music effortlessly.
* **Comprehensive Music Streaming**: Organize and manage songs with advanced search options.
* **Modern Tech Stack**: Use React.js, Vite, and Tailwind CSS for fast and efficient performance.

**. SCOPE OF THE PROJECT**

**What the Project Covers**

* Songs library with metadata (title, singer, genre, image, audio link)
* Search functionality (by song, genre, or singer)
* Playlist creation and management
* Favorites (wishlist) feature
* Audio player with one-track-at-a-time control

**6.Setup Instructions**

**Pre-requisites**

1. Clone the repository

git clone <repository-url>

cd rhythmic-tunes

2.Install Node.js & npm: <https://nodejs.org/en/download/>

3.Install dependencies

npm install

4.Start the JSON server:

json-server --watch db.json --port 3000

5. Access the app at [http://localhost:3000](http://localhost:3000).

**7. SYSTEM REQUIREMENTS**

**Hardware Requirements**

* **Processor**: Intel i3 or higher (i5 recommended)
* **RAM**: 4GB minimum (8GB recommended)
* **Storage**: 10GB free
* **Display**: 1366x768 or above

**Software Requirements**

* **OS**: Windows 10/11, macOS, Linux
* **Languages**: JavaScript (ES6+), JSX, HTML, CSS
* **Framework**: React.js (v18.2.0)
* **Styling**: Tailwind CSS, Bootstrap
* **Build Tool**: Vite
* **Version Control**: Git & GitHub
* **Package Manager**: Node.js + npm
* **Browser**: Chrome / Firefox / Edge

**8. CODE IMPLEMENTATION**

**Folder Structure:**

rhythmic-tunes/

│

├── public/

├── src/

│ ├── components/

│ ├── pages/

│ ├── App.jsx

│ ├── main.jsx

│

├── db.json

├── index.html

├── package.json

├── vite.config.js

└── README.md

**9.** **METHODOLOGY / PROJECT DEVELOPMENT**

**Planning and Design Steps Followed**

1. Requirement Analysis
2. UI/UX Design
3. Technology Selection (React, Tailwind, Bootstrap, Vite)
4. Database Setup (db.json with json-server)
5. Component Development (Songs, Playlist, Favorites)
6. Integration & Testing

**Tools and Technologies Used**

* **React.js**
* **Tailwind CSS** & **Bootstrap**
* **Axios** (API calls)
* **React Router DOM** (Routing)
* **json-server** (Mock Backend)
* **Vite** (Fast bundler)
* **GitHub** (Version control)

**10. PROJECT WORKFLOW**

1. Launch the app → Load songs from db.json
2. Search or browse songs
3. Add/remove songs to Favorites
4. Add/remove songs to Playlist
5. Play/Pause music (only one song plays at a time)

**11.** **Component Documentation**

**Key Components:**

Songs Component:

Displays list of songs with audio player.

Favorites Component:

Manages user’s favorite songs.

Playlist Component:

Allows adding/removing songs from playlist.

SearchBar Component:

Filters songs by title, genre, or singer.

Player Component:

Ensures only one track plays at a time.

**12.** **State Management**

**Global State:**

Simulated with \*json-server\* (db.json) acting as a backend.

Axios fetches:

\* /items → All songs

\* /favorities → Favorites

\* /playlist → Playlists

**Local State:**

Use State hooks for search term, current playing track, UI toggles.

Use Effect ensures real-time sync with db.json.

**13. FEATURES IMPLEMENTED**

* **Songs List Page**: Displays all songs with image, title, genre, singer.
* **Audio Player**: Embedded <audio> tag with play/pause controls.
* **Favorites (Wishlist)**: Add/remove songs with heart button.
* **Playlist Management**: Create custom playlists with add/remove buttons.
* **Search Functionality**: Search by title, singer, or genre.
* **Responsive UI**: Built with Bootstrap grid + Tailwind CSS.

*Screenshots placeholders: Home Page, Favorites, Playlist, Now Playing*

**10. TESTING & OUTPUT**

**Testing the System**

* **Song Fetching**: Verified songs load correctly.
* **Favorites**: Added/removed songs update instantly.
* **Playlist**: Items persist until removed.
* **Search**: Accurate filtering by singer, genre, or name.
* **Audio Playback**: Only one song plays at a time.
* **Responsiveness**: Works on desktop & mobile.

**Output Description**

* **Songs Page** → Displays library of songs.
* **Favorites Page** → User’s saved favorites.
* **Playlist Page** → Custom user playlists.
* **Now Playing** → Audio controls for current track.

**12. Screenshots & Demo Video Link**

Figure 1: Output opening Page

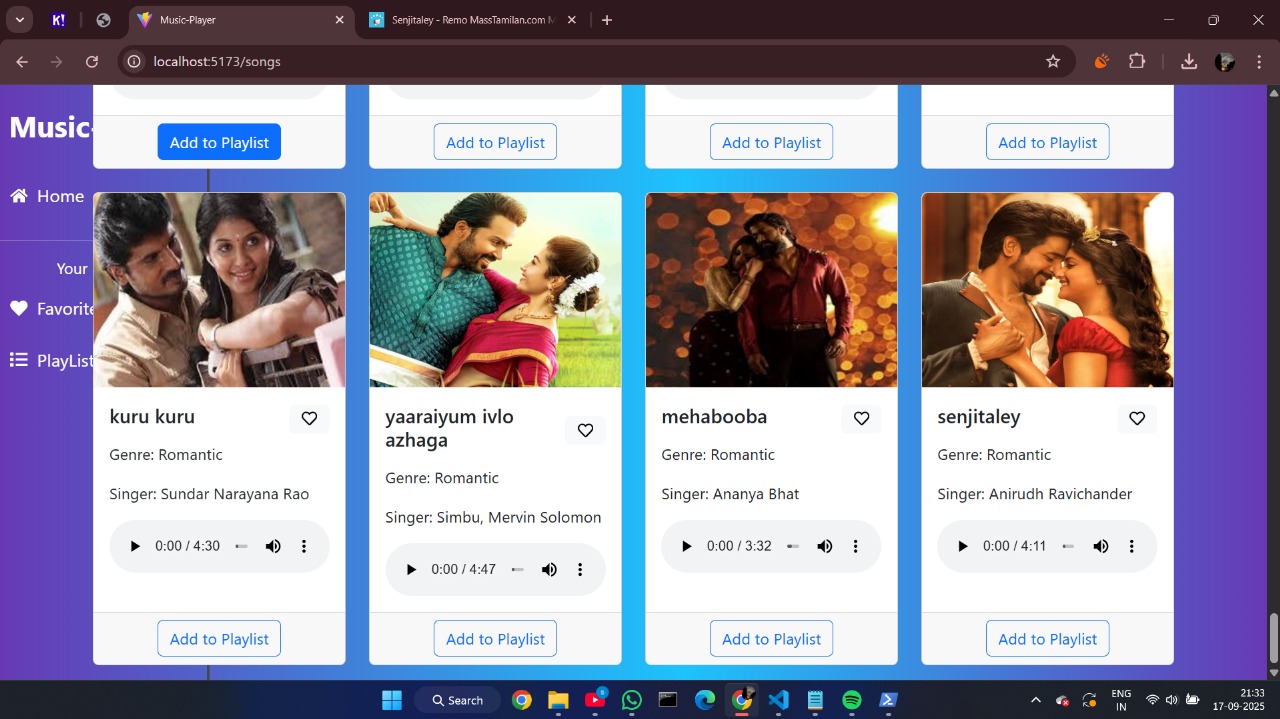


Figure 2: Home page

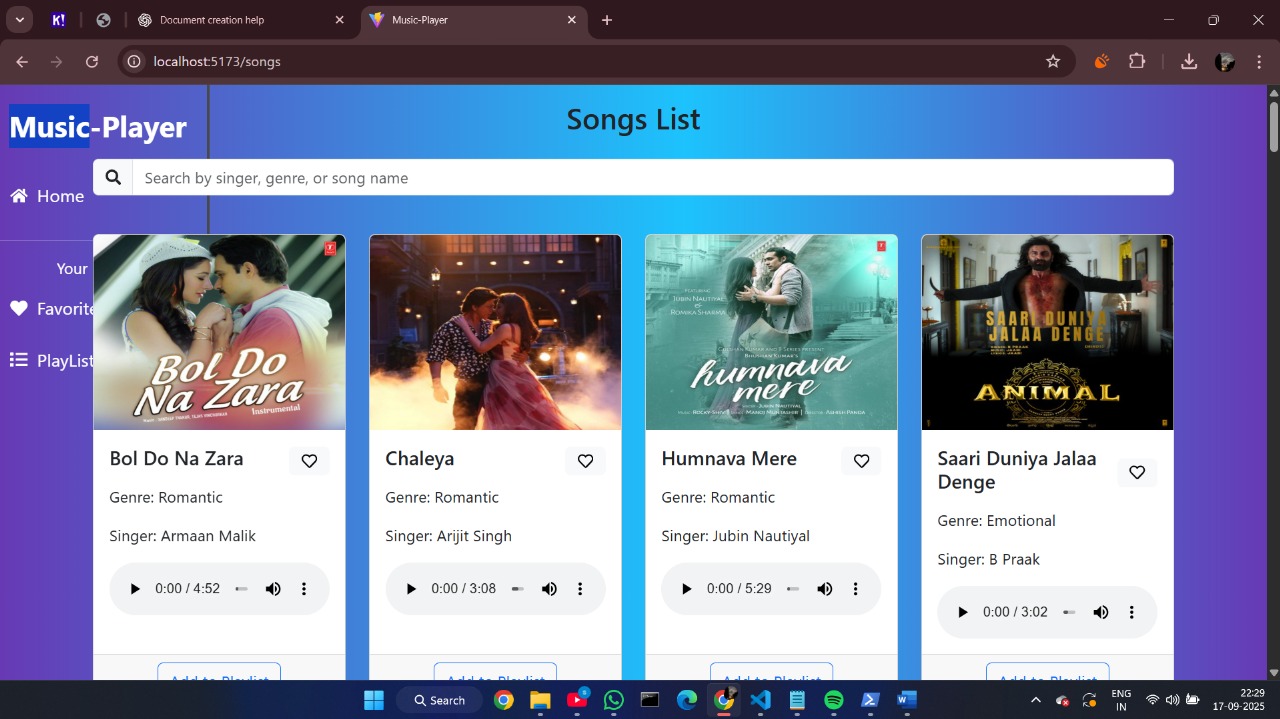


Figure 3: Favorites page

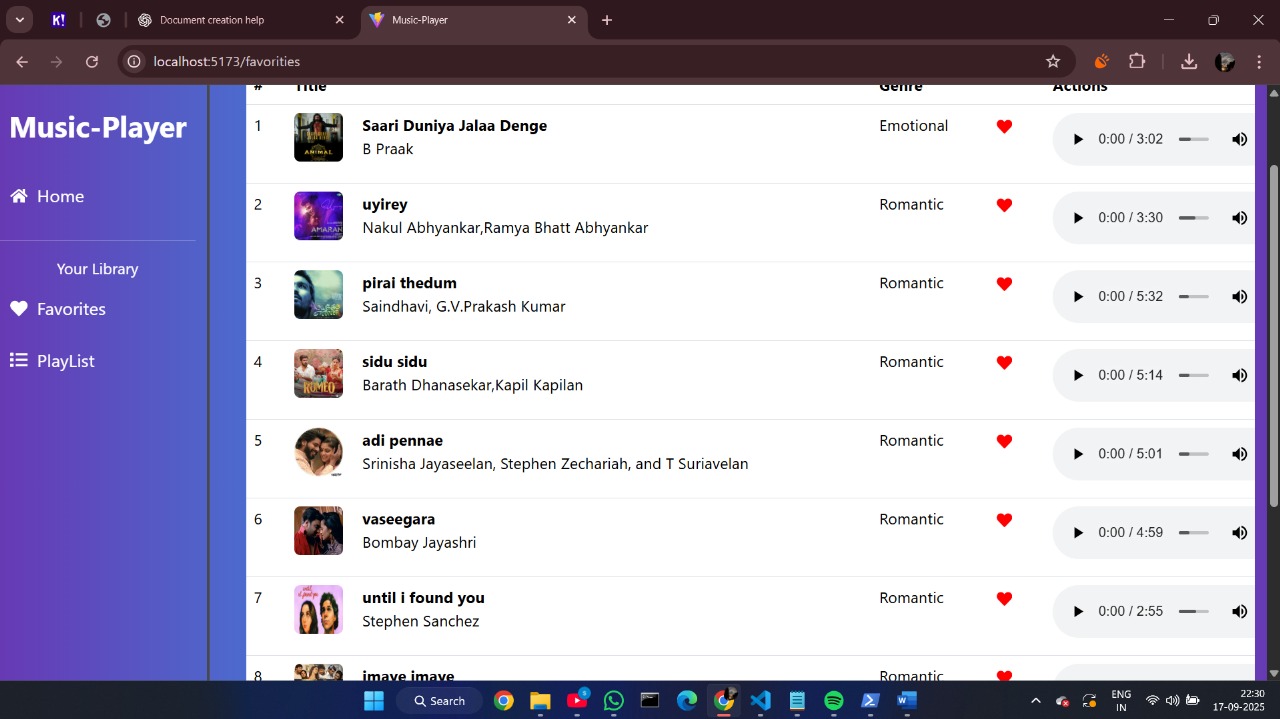


Figure 4: Playlist Page

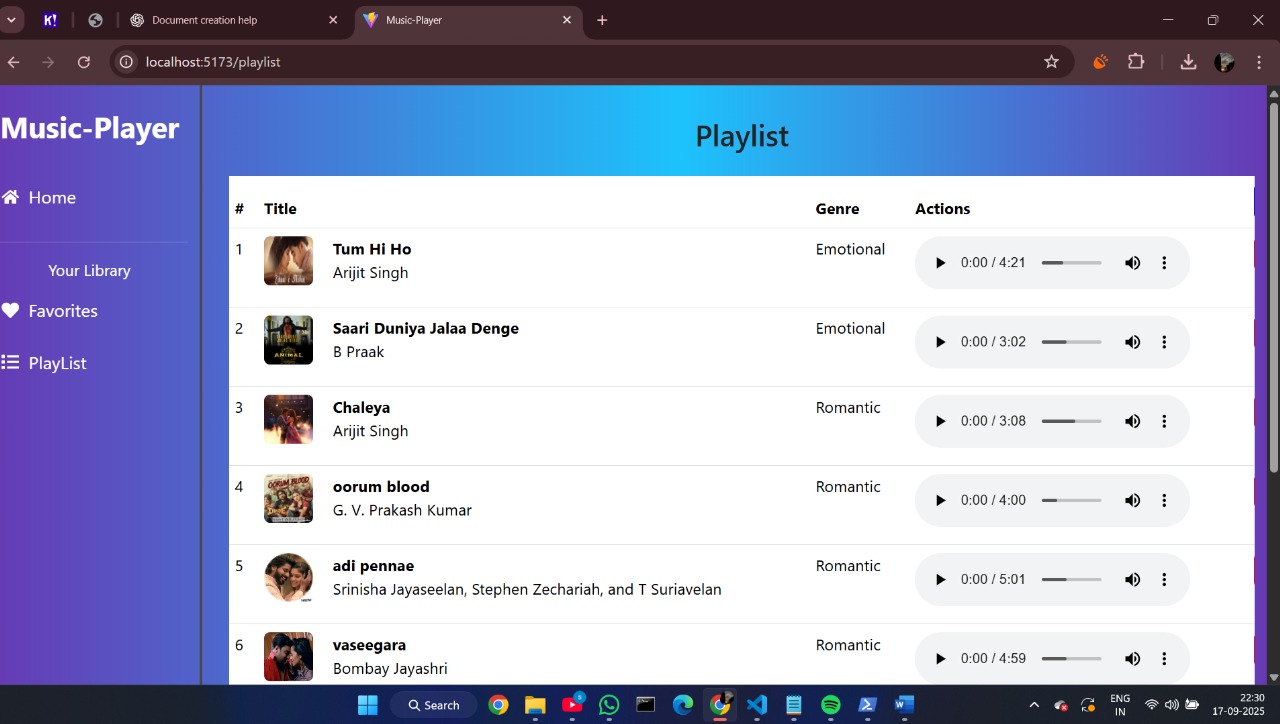
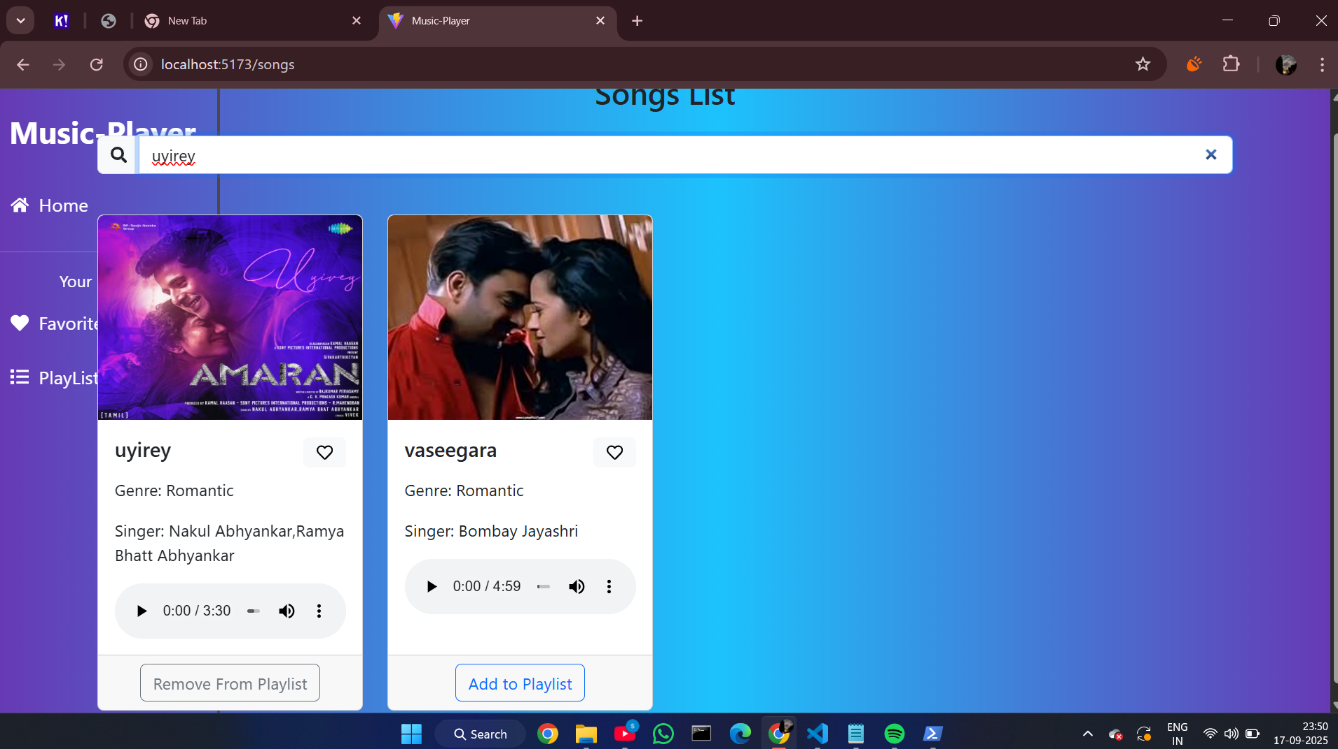


Figure 5: Searched Songs Display Page



Demo Video Link:

<https://drive.google.com/file/d/1fLtz-tCmyAdP0rFVnWrxR8OZ4OqTo-Hk/view?usp=drivesdk>

**13. Known Issues**

\* No authentication system (multi-user not supported).

\* No real backend (data lost on refresh if db.json is reset).

\* No offline mode or downloads.

\* Limited scalability for large music libraries.

**14. Future Enhancements**

\* Backend integration with permanent database (MongoDB, Firebase, etc.).

\* User authentication and profiles.

\* Real-time multi-user playlist sharing.

\* AI-powered song recommendations.

\* Mobile application version.