Rhythmic Tunes – Your Melodic Companion

Introduction

- Project Title: Rhythmic Tunes – Your Melodic Companion
- Team ID: NM2025TMID30129
- Team Leader: SHARUMATHI P – 202400821@sigc.edu
- Team Members:

GEEVIHA N - 202400365@sigc.edu
RUBA E - 202400161@sigc.edu
THULASI A - 202400896@sigc.edu

Project Overview

- **Discoverability:**
  Users can search for songs by title, singer, or genre using a powerful real-time search bar.

- **Interactive Music Experience**:
  Users can listen to music directly using in-built audio players with progress bars and volume control.

- **Favorites System:**
  Songs can be added or removed from the favorites list with a single click, making it easy to revisit beloved tracks.

- **Custom Playlists**:
  Users can add/remove songs to/from playlists, enabling them to organize their listening sessions based on mood or theme.

- **Categorization by Genre and Singer**:
  Songs are displayed with genre and singer information, allowing users to explore based on their musical preferences.

Features:

1.Audio Playback

- Built-in audio player for each song card
- Controls: Play, Pause, Seek, Volume Adjust

2. Favorites Management

- Add or remove any song to/from Favorites with a heart icon
- Visually indicates favorited songs
- View all favorites in a dedicated Favorites page

3. Playlist Management

- Add songs to a custom playlist
- Remove songs from playlist
- Playlist view accessible from the sidebar
- Playlist status changes dynamically (buttons update based on current list)

4. Real-Time Search

- Search bar to filter songs by:
  - Singer name
  - Genre
  - Song title

5. Song Metadata Display

Each song card shows:

- Title
- Genre (e.g., Romantic, Emotional)
- Singer name
- Album artwork/poster

6. Responsive UI

- Built using React + Bootstrap, ensuring:
  - Smooth layout
  - Responsive across screen sizes
  - Visually appealing with modern styling (gradient backgrounds, card layouts)

7. Organized Navigation

- Sidebar with clearly labeled sections:
  - Home – All songs
  - Your Library – (Optional future feature)
  - Favorites – Filtered view of favorited songs
  - Playlist – User-curated list of selected songs

8. Scalable Architecture (Under the Hood)

Although not visible in UI, based on the stack, your backend likely supports:

- User management (planned or optional)
- API routes for managing songs, playlists, favorites
- MongoDB storage for persistent data

9. Local Hosting and Development Friendly

- Runs on localhost:5173 for frontend (likely using Vite)
- Backend probably runs on localhost:3000 or similar
- Easy to set up and test during development

Architecture

1.Frontend Architecture

Framework: React.js
Styling: Bootstrap, CSS, and possibly Material UI

Components:

- Search Bar: Filters songs in real-time by name, singer, or genre.

- Song Card Component: Displays each song's title, genre, singer, audio controls, and action buttons (Add to Playlist, Favorite).
- Navigation Sidebar: Allows users to switch between Home, Favorites, and Playlist.
- **Pages:**
    - HomePage.js: Shows all songs
    - FavoritesPage.js: Shows favorited songs
    - PlaylistPage.js: Shows playlist songs

2. Backend Architecture

Framework: Node.js + Express.js
Responsibilities:

- Handle API requests (GET, POST, DELETE)
- Manage song data (CRUD operations)
- Store and serve playlist/favorites data
- Authenticate users (if implemented)

3. Database

Collections:

- songs – stores all song metadata (title, genre, singer, duration, URL, etc.)
- users (optional) – stores user info, playlists, and favorites
- favorites – links user ID to favorite songs
- playlists – links user ID to selected playlist songs

4. API Architecture

Feature	HTTP Method	Endpoint	Description
Get all songs	GET	/api/songs	Fetch list of songs
Add to playlist	POST	/api/playlist/add	Add song to playlist
Remove from playlist	DELETE	/api/playlist/remove/:id	Remove song from playlist
Mark as favorite	POST	/api/favorites/add	Add song to favorites
Remove favorite	DELETE	/api/favorites/remove	Remove song from favorites

Setup Instructions

Prerequisites

- Node.js
- MongoDB
- Git
- React.js
- Visual Studio Code

Installation Steps

Step 1: Download Node.js LTS version

Step 2: Open Windows PowerShell As Administrator

Step 3: Type set-executionPolicy unrestricted and press enter.Next type Y and press enter

Step 4: Now open VS code and open the code folder from the extracted folder**.**

Step 5: Now open a new terminal

Step 6: Type npm install in your terminal and press Enter and wait till all the dependencies gets download

Step 7: After all the downloads finishes, now type npm run dev and press Enter

Step 8: Now your application will be opened in your browser with url **http://localhost:5173**

Step 9: Create The JSON Server

- Split the terminal
- Change the directory from code to db using the command cd db
- Run the command npm i -g jsonserver , it will globally install the json server
- After install the server then run the following command json-server --watch db.json --port 3000

Folder Structure

Music-Player/

→code

→db

*db.json

→node modules

→public

*Songs

*vite

→src

*Assets

React.svg

*Components

Favorites.jsx

Playlist.jsx

Search.jsx

Sidebar.css

Sidebar.jsx

Songs.jsx

Uhome.css

Uhome.jsx

Uitem.jsx

Unavbar.jsx

Wishlist.jsx

*App.css

*App.jsx

*Index.css

*main.jsx

→.eslintrc

→.gitignore

→index.html

→package.json

→package-lock.json

→README.md

→vite.config.js

Running the Application

Frontend

npm install

npm run dev

Backend

cd code

npm i -g jsonserver

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

Access Application

http://localhost:5173/

API Documentation

User APIs:

- POST /api/user/register — Register new user
- POST /api/user/login — Login user

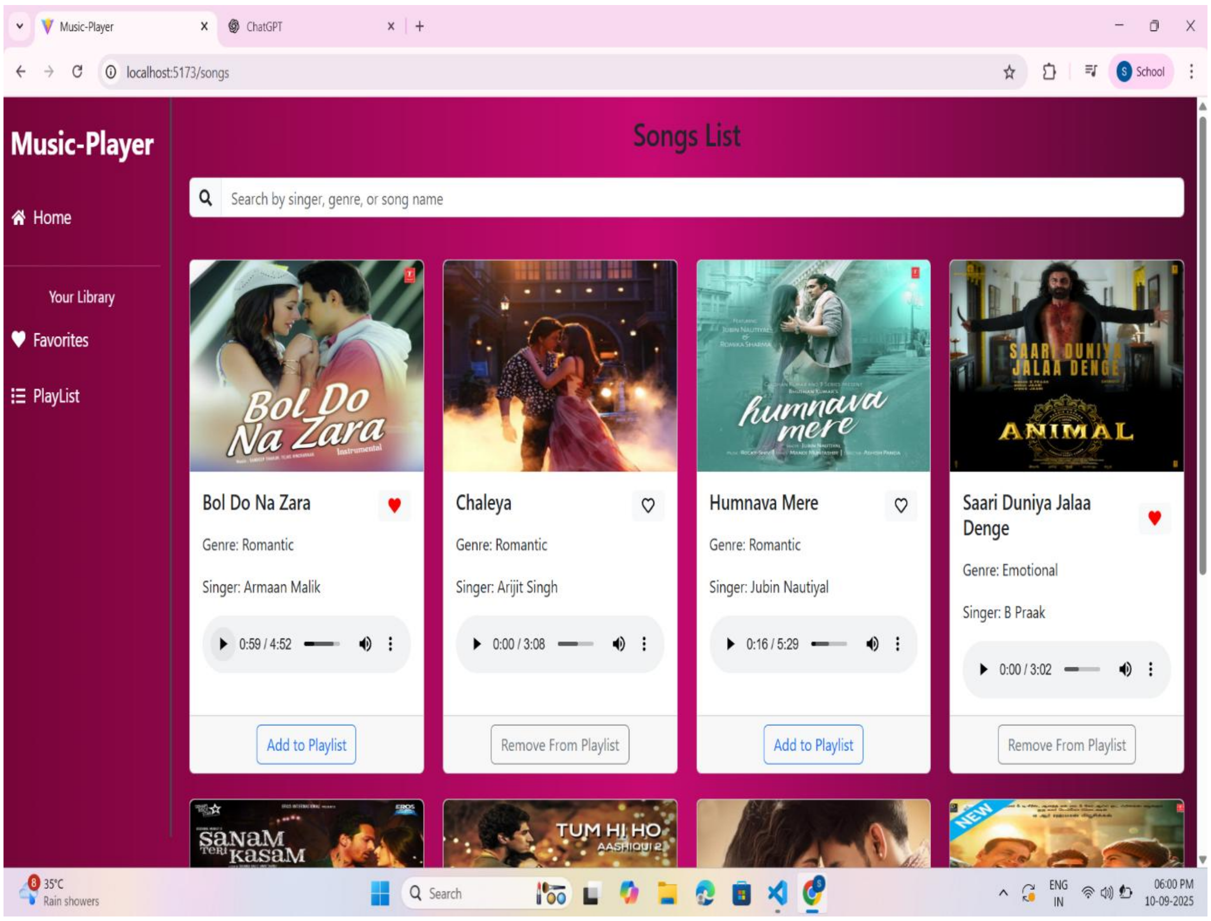
Songs APIs:

- GET /api/songs — Get all songs
- POST /api/songs/add — Add new song
- PUT /api/songs/favorite/:id — Toggle favorite
- POST /api/songs/playlist — Add to playlist

Authentication

- JWT-based authentication for secure user sessions
- Middleware functions protect private routes (playlist/favorites)

User Interface



Bol Do Na Zara
Genre: Romantic
Singer: Armaan Malik
0:59 / 4:52
Add to Playlist

Chaleya
Genre: Romantic
Singer: Arijit Singh
0:00 / 3:08
Remove From Playlist

Humnava Mere
Genre: Romantic
Singer: Jubin Nautiyal
0:16 / 5:29
Add to Playlist

Saari Duniya Jalaa Denge
Genre: Emotional
Singer: B Praak
0:00 / 3:02
Remove From Playlist

Home Page (Song List)

- Route: songs
- Features:
    - Displays all songs in a card/grid layout
    - Each card includes:
        - Song title, genre, and singer
        - Album art
        - Play/pause audio player
        - Add to Playlist button
        - Favorite (heart) toggle
    - Real-time audio control with progress and volume

Search Bar

- Positioned at the top of the song list
- Functionality**:** Search songs by:
    - Song title
    - Singer name
    - Genre

Sidebar Navigation

- Present on the left side of all pages
- Contains links to:
    - Home – List all songs
    - Your Library – Personalized user content (future scope or logged-in user)
    - Favorites – Filter and view marked favorite songs
    - Playlist – View and manage songs added to the playlist

Favorites Page

- Displays only songs marked as favorite
- Uses same card layout as the Home page
- Allows toggling favorite status directly

Playlist Page

- Shows songs added by the user to their playlist
- Functionality**:**
    - Remove song from playlist
    - Play songs directly from playlist

Responsiveness

- Designed with mobile-first approach using Bootstrap grid
- Fully responsive on:
    - Desktops
    - Tablets
    - Smartphones

Visual Preview

- Home Page with Song Cards
- Favorite Songs View
- Playlist View
- Responsive Search and Sidebar

Testing

Types of Testing Performed

- Manual Testing:
  Primary method for validating functionality, layout, and user interaction.

- API Testing:
  Ensured that all backend endpoints return correct data and error messages.

- UI Testing:
  Verified layout, responsiveness, and user interactions.

- Functional Testing:
  Tested each feature to ensure it behaves as expected

Tools Used

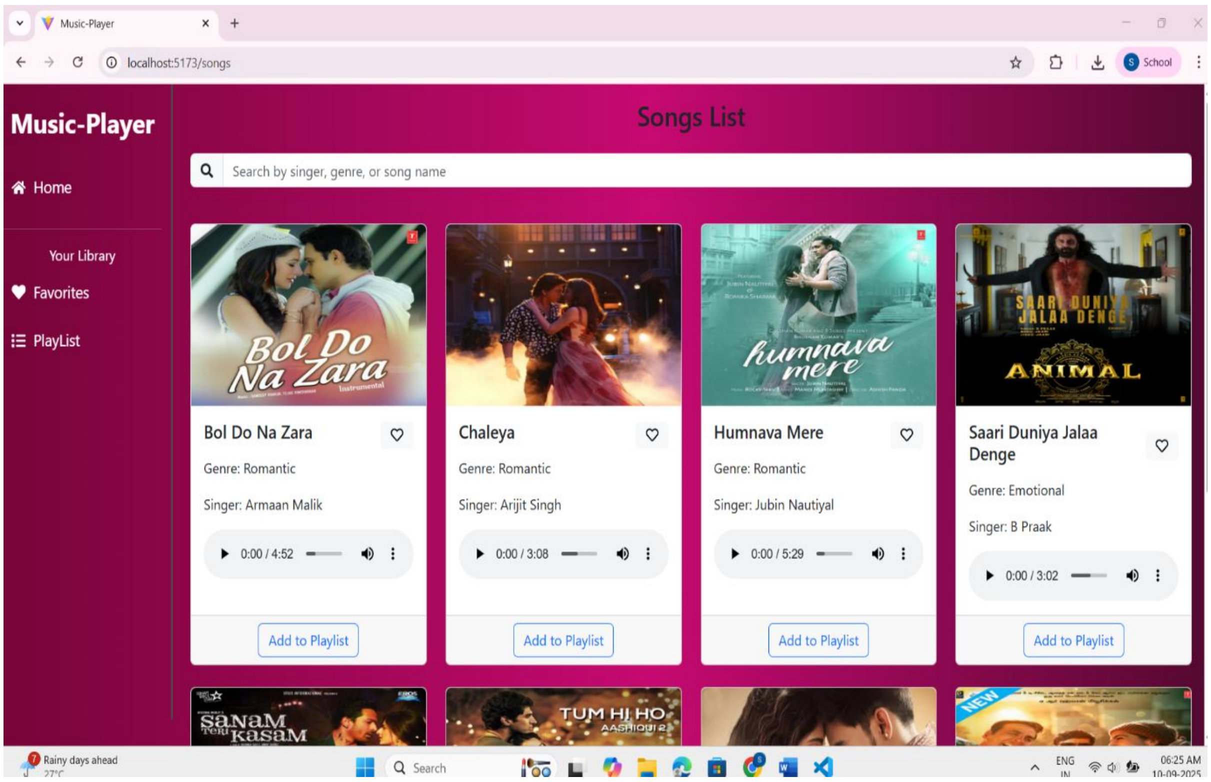
- Chrome DevTools:

  Debugging and responsive testing

- React Developer Tools:

  Debugging frontend state

Screenshots or Demo



Songs List

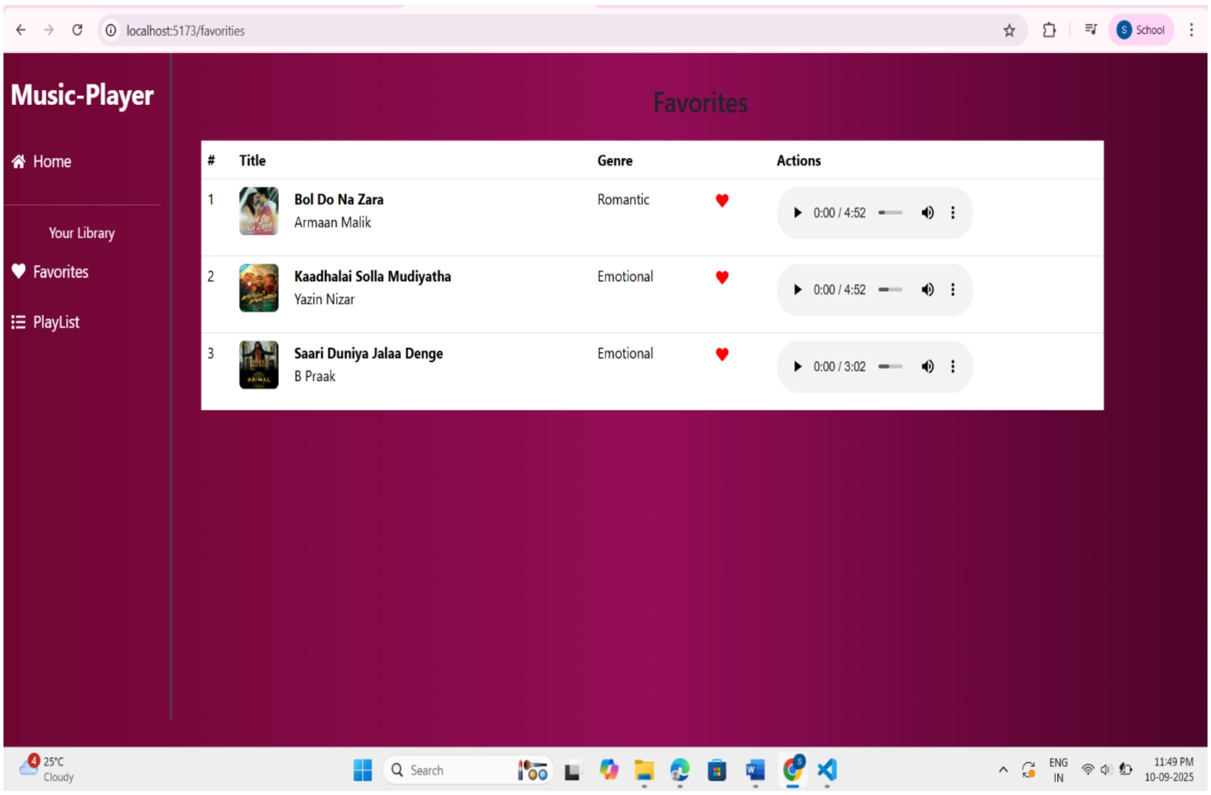
Search by singer, genre, or song name

Bol Do Na Zara
Genre: Romantic
Singer: Armaan Malik
0:00 / 4:52

Chaleya
Genre: Romantic
Singer: Arijit Singh
0:00 / 3:08

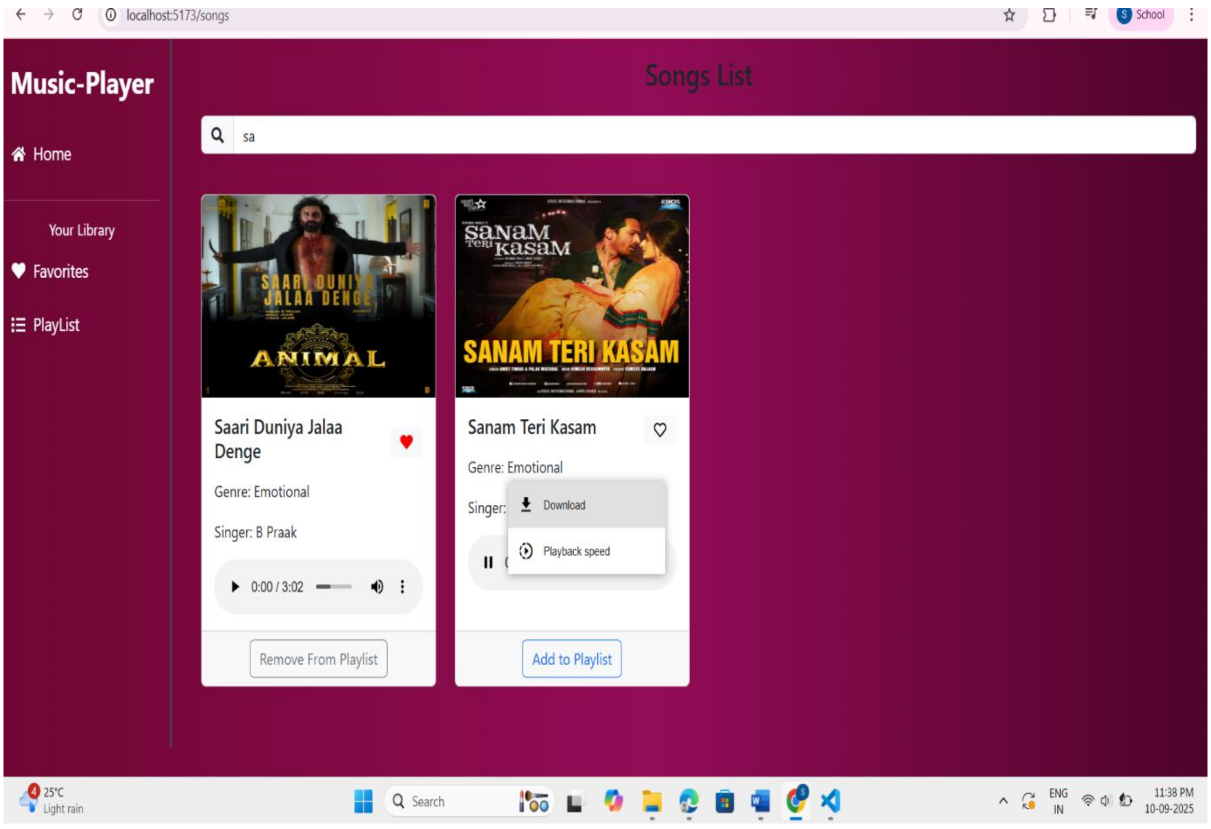
Humnava Mere
Genre: Romantic
Singer: Jubin Nautiyal
0:00 / 5:29

Saari Duniya Jalaa Denge
Genre: Emotional
Singer: B Praak
0:00 / 3:02



Favorites

#	Title	Genre		Actions
1	Bol Do Na Zara Armaan Malik	Romantic	♥	0:00 / 4:52
2	Kaadhalai Solla Mudiyatha Yazin Nizar	Emotional	♥	0:00 / 4:52
3	Saari Duniya Jalaa Denge B Praak	Emotional	♥	0:00 / 3:02



Known Issues

- Playlist changes not persistent without login (if authentication is disabled)
- Volume control may not sync across multiple song cards
- Mobile view requires further responsiveness enhancements

Future Enhancements

- Implement user login and profile management
- Add music categories and sort/filter options
- Integrate audio visualization
- Enable cloud audio storage or streaming from external APIs
- Support for creating multiple custom playlists

Conclusion

- The Music-Player Web Application successfully delivers a seamless and interactive music streaming experience through a modern, responsive web interface.
- Built with the MERN stack (MongoDB, Express.js, React.js, Node.js), the project demonstrates strong integration between frontend components and backend APIs.
- Key features like audio playback, dynamic song listing, playlist management, and favorites enhance the overall user experience, while JWT-based authentication ensures secure and personalized access to user data.
- The application's clean UI, intuitive controls, and robust functionality align well with current user expectations for music platforms.
- This project not only showcases practical implementation of full-stack web development but also lays a solid foundation for future enhancements such as user accounts, multi-playlists, music recommendations, and real-time streaming.
- Overall, the Music-Player project is a functional and scalable prototype that reflects both strong technical skills and a user-centered design approach.