Full Stack Development with MERN

Project Documentation

1. Introduction

- **Project Title:** TuneTrail
- **Team Members:** Jaiaditya Mathur, Saket DB, Riya Autade, Savithri Nair

2. Project Overview

- **Purpose:** TuneTrail aims to revolutionize the music streaming experience by offering a comprehensive platform for music enthusiasts to discover, organize, and enjoy their favourite tunes effortlessly. It offers personalised song recommendation, user-centric playlist management, and a vast music library, ensuring users stay updated with the latest music trends.

- **Features:**
  - Search functionality for songs and artists
  - Create and manage playlists
  - User Profiles
  - Offline Listening

3. Architecture

- **Frontend:** Developed using React.js for its component-based architecture, ensuring modular and scalable UI development.
- **Backend:** Built on Node.js and Express.js, providing a robust and scalable server-side architecture. Implements APIs to handle operations for user data, playlists, and music metadata.
- **Database:** MongoDB used as the NoSQL database for storing user profiles, playlists, and music metadata. Utilizes Mongoose for seamless interaction with MongoDB.

4. Setup Instructions

- **Prerequisites:** Node.js, Express.js, React.js, MongoDB, Mongoose, HTML, CSS, JavaScript, Git and GitHub.

- **Installation:** Step-by-step guide to clone, install dependencies, and set up the environment variables:
    - Clone the repository: “git clone https://github.com/jai-mathur05/TuneTrail”
    - Navigate to the client directory: “cd tunetrail/client”
    - Install frontend dependencies: “npm install”
    - Navigate to the server directory: “cd ../server”
    - Install backend dependencies: “npm install”
    - Set up environment variables: Configure “.env” file for MongoDB URI and JWT secret key.

5. Folder Structure

- **Client:** Frontend Structure using React:
    - Public folder
      - *Audio folder*
        - Songs
      - Index file
      - Vite media
    - Src folder
      - *Admin folder*
        - Add songs
        - My songs
        - Navbar
        - Sidebar
        - Login
        - Signup
        - Home
        - User files
      - *Components folder*
        - Home
      - *User folder*
        - Favourites
        - Login
        - Signup
        - Songs
        - Playlists
        - Wishlist
        - Search bar
        - Sidebar
        - Home
        - Items
        - Navbar
      - *Assets folder*
        - Media
      - App files
      - Index files

    - Readme file
    - Vite plugin file
    - Index page
    - JSON files
    - ESLint files

- **Server:** Backend Structure using Node.js backend:

    - Database “db” folder
      - *Admin folder*
        - Add songs
        - Admin file
      - *User folder*
        - Playlist
        - Wishlist
        - User file
      - Configuration file

    - Uploads folder
      - Songs

    - JSON files
    - Server file
    - Env file
    - Git file

6. Running the Application

- Commands to start the frontend and backend servers locally:
    - **Frontend:** `npm start` in the client directory.
    - **Backend:** `npm start` in the server directory.

7. API Documentation

- Endpoints exposed by the backend with request methods and parameters:

    - **`GET /api/songs`**: Retrieves list of songs.
    - **`POST /api/playlists`**: Creates a new playlist.
    - **`PUT /api/playlists/:id`**: Updates an existing playlist.
    - **`DELETE /api/playlists/:id`**: Deletes a playlist by ID.

- Example response:

```
{ "title": "My Playlist",

  "songs": ["songId1", "songId2"]}
```

8. Authentication

- <u>Handling authentication and authorization in the project</u>:

  TuneTrail implements JWT (JSON Web Tokens) to ensure secure authentication and authorization across the platform.

- <u>Tokens, sessions, or any other methods used</u>:

  - Token Generation: Upon successful login, a JWT is generated and sent to the client. This token contains user information and an expiration time.

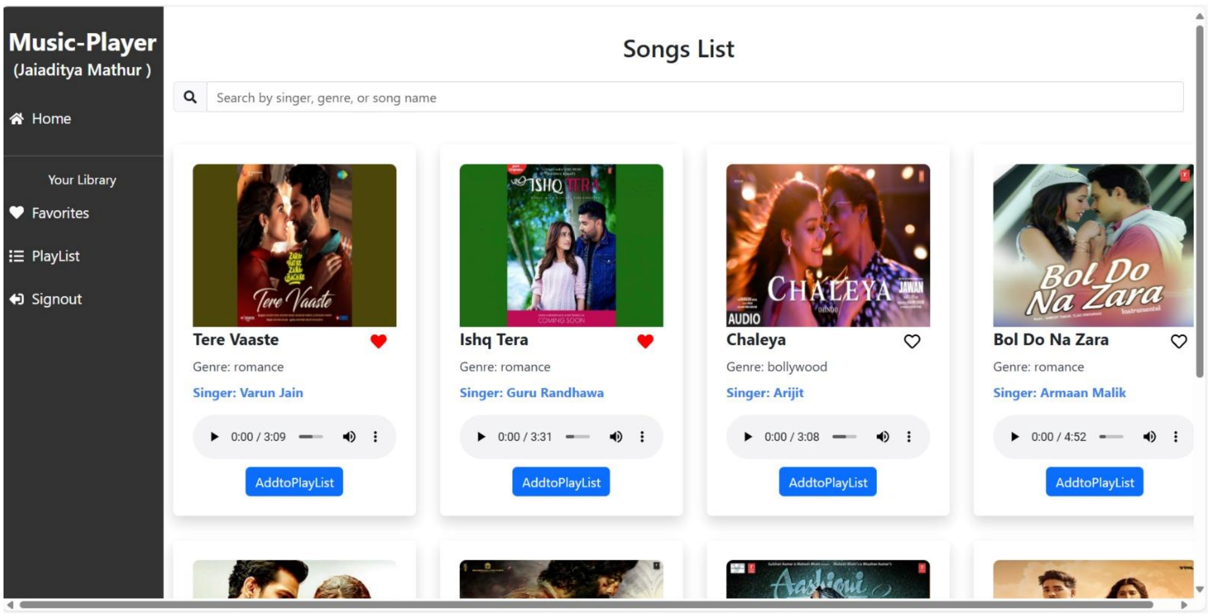
  - Token Verification: For protected routes, the server verifies the JWT to ensure the user is authenticated before granting access.

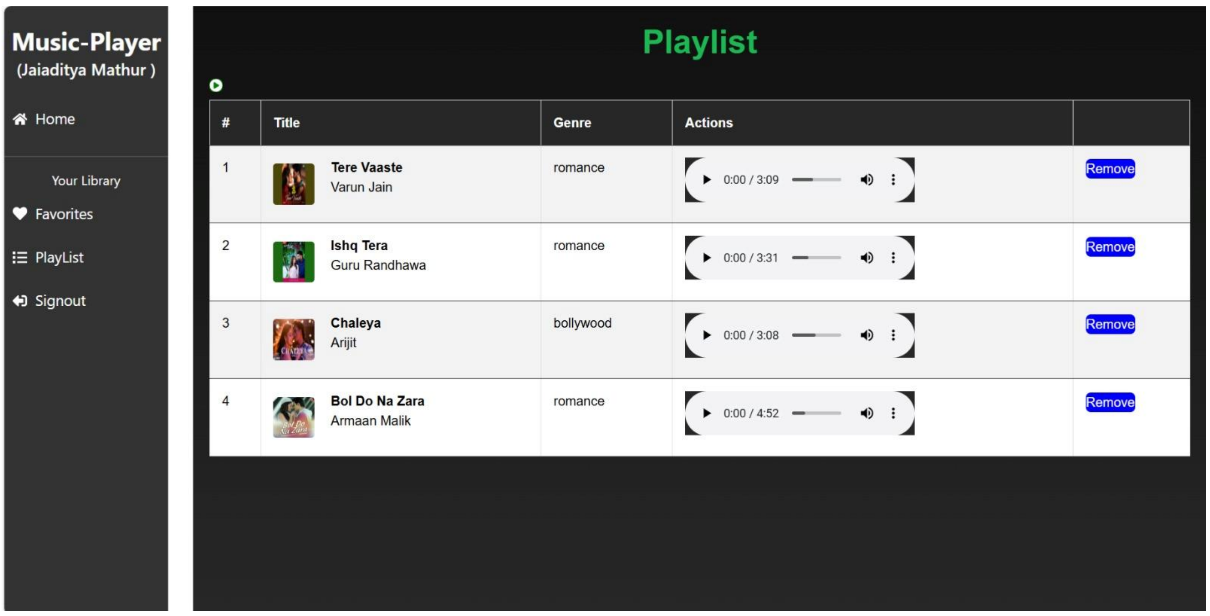
  - Token Storage: Tokens are securely stored in local storage on the client side and are included in the headers of API requests for authentication purposes.

  - Secure Data Handling: Sensitive user data is encrypted and securely managed to protect against unauthorized access.

9. User Interface

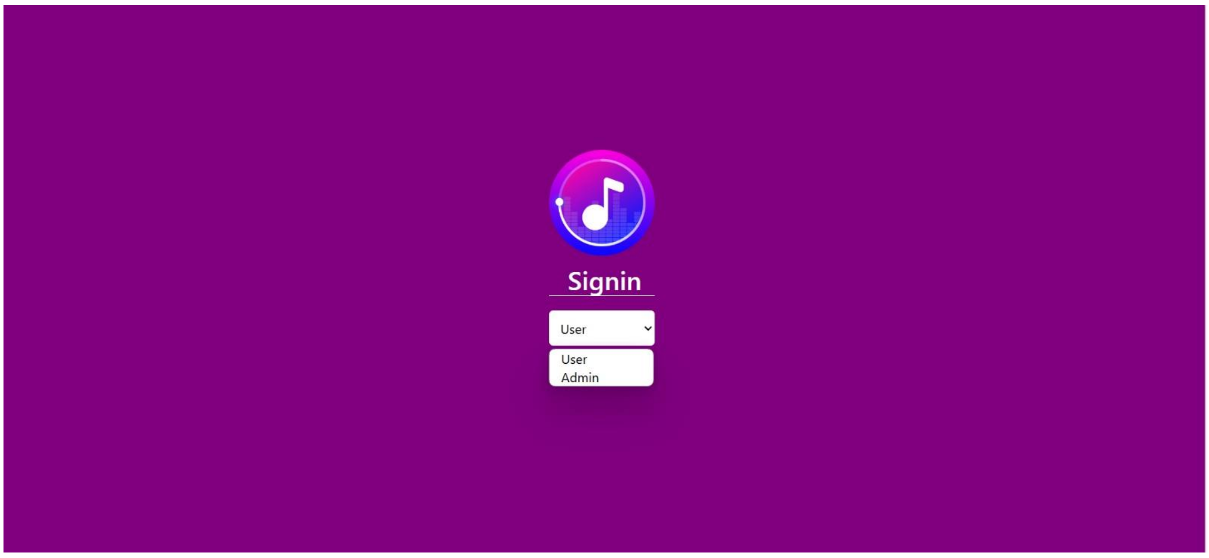
- <u>Screenshots showcasing different UI features</u>:





Playlist

#	Title	Genre	Actions	
1	Tere Vaaste Varun Jain	romance	0:00 / 3:09	Remove
2	Ishq Tera Guru Randhawa	romance	0:00 / 3:31	Remove
3	Chaleya Arijit	bollywood	0:00 / 3:08	Remove
4	Bol Do Na Zara Armaan Malik	romance	0:00 / 4:52	Remove



- **Homepage:** Displays tailored music recommendations based on user listening history and preferences, featuring trending songs and new releases.
- **Playlist Interface:** Allows users to create, edit, and organize playlists easily. Users can add or remove songs, reorder tracks, and share playlists with friends.
- **User Profile:** Users can update their profile details, such as username, profile picture, and bio.

10. Testing

- The testing strategy and tools used:

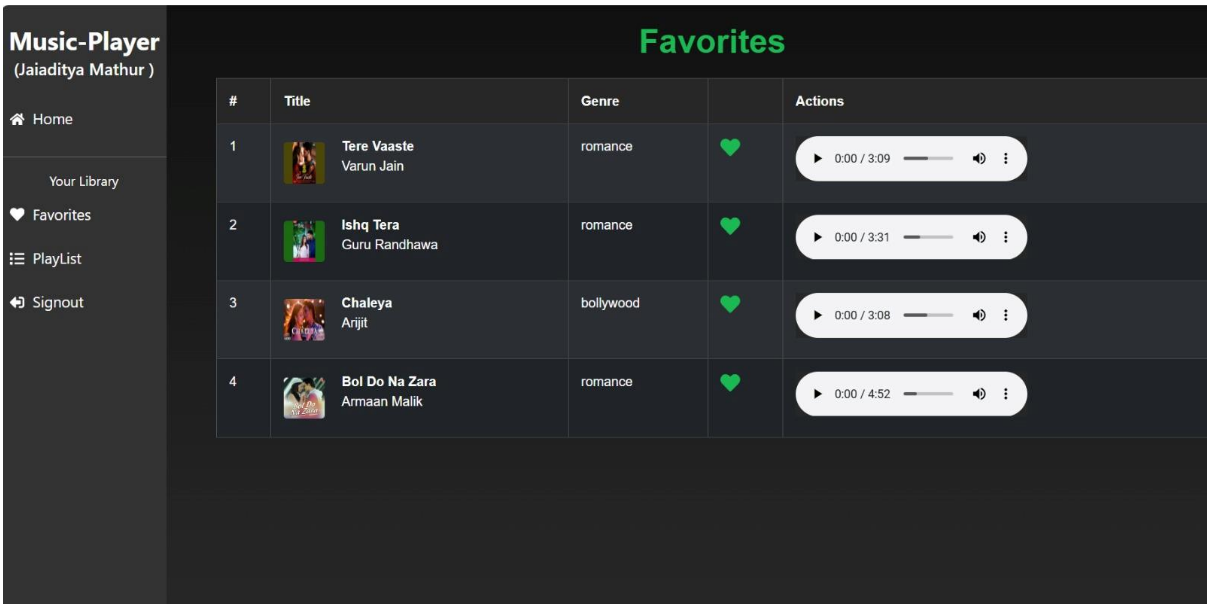
  TuneTrail adopts a comprehensive testing strategy to ensure the reliability and performance of the application.

- Unit Testing: To test individual React components, ensuring they function as expected in isolation.

- Integration Testing: To perform integration tests on backend APIs, verifying that different parts of the system work together correctly.

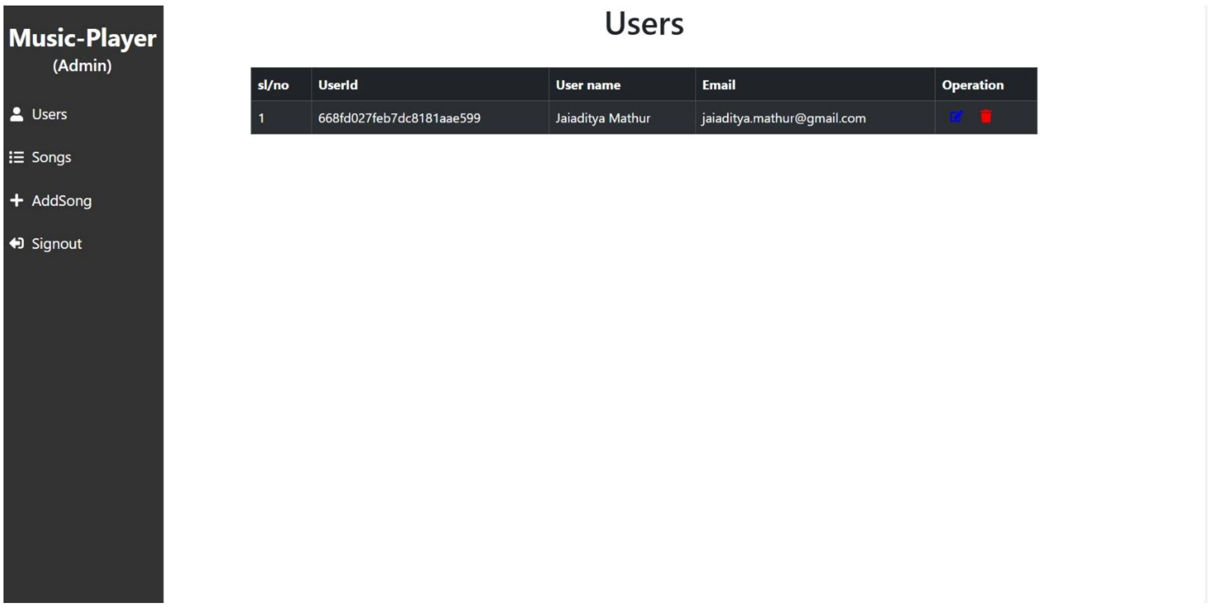
11. Screenshots or Demo

- Screenshots to showcase the application:



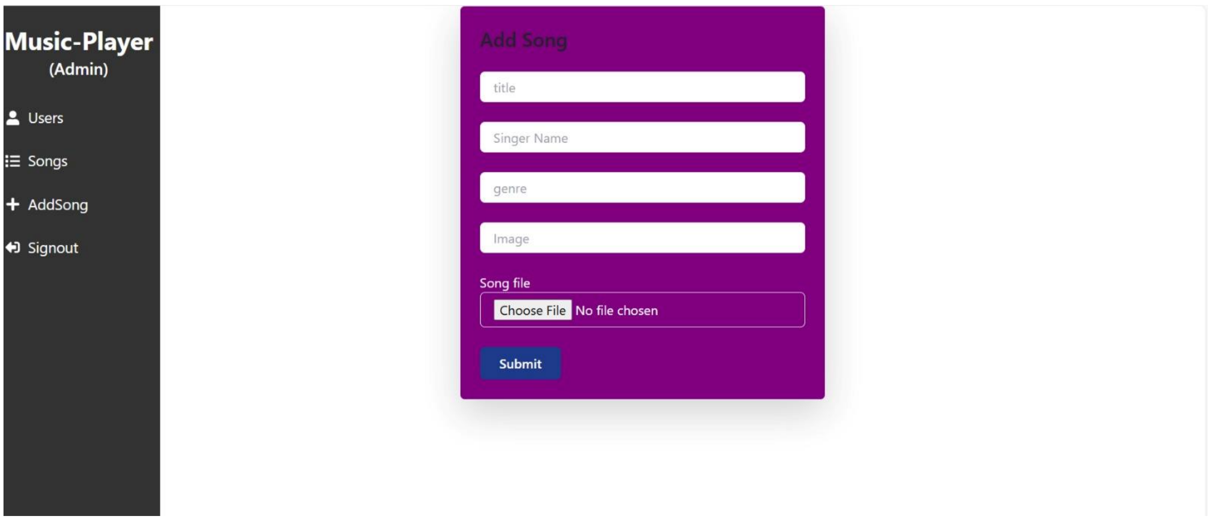
Favorites

#	Title	Genre		Actions
1	Tere Vaaste Varun Jain	romance	♥	0:00 / 3:09
2	Ishq Tera Guru Randhawa	romance	♥	0:00 / 3:31
3	Chaleya Arijit	bollywood	♥	0:00 / 3:08
4	Bol Do Na Zara Armaan Malik	romance	♥	0:00 / 4:52



Users

sl/no	UserId	User name	Email	Operation
1	668fd027feb7dc8181aae599	Jaiaditya Mathur	jaiaditya.mathur@gmail.com	



<u>Demo Link</u>:

12. Known Issues

- <u>Bugs or issues that users or developers should be aware of:</u>

  - Users may experience occasional delays in loading playlists due to high server load.

  - Search functionality might be slow under certain conditions.

  - Playback controls might be slow sometimes.

13. Future Enhancements

- <u>Potential future features or improvements that could be made to the project</u>:

  - Collaborative Playlist Features: Introducing features that allow users to collaborate on playlists with friends, share their favourite tracks, and follow other users' playlists.

  - Implementing Real-Time Chat: Adding real-time chat functionality to enable users to discuss music and share recommendations live.

  - Music Streaming Events: Hosting music streaming events where users can listen to and discuss music together in real-time.

  - Mood-based Playlists: Introducing mood-based playlists that adapt to the user's current mood or activity.

  - Podcast and Audiobook Integration: Adding support for streaming podcasts and audiobooks and giving personalized recommendations for the same.