

# Full Stack Development with MERN

## Project Documentation format

### 1. Introduction

**Project Title:** TuneTrails - Music Streaming App

**Team Members:**

- Viraj Mane – Worked on Backend and Video Demonstration
- Darpan Nemade – Worked on Frontend and Database Connectivity
- Aniket Shankarwar – Worked on UI/UX design and User/Admin Authentication
- Prathamesh Chaudhari– Worked on Project Documentation and Database Connectivity

### 2. Project Overview

**Purpose:** TuneTrails is a full-stack music streaming web application developed using the MERN stack. It provides users with a platform to explore, search, and stream music online. Admins can manage content by uploading albums and songs. The app offers smooth audio playback, playlist management, and secure authentication to ensure a seamless user experience.

**Features:**

- JWT-based authentication and role-based authorization
- Admin and user registration/login system
- Album listing with embedded thumbnail and description
- Song streaming with playback controls (play, pause, volume, time)
- Search functionality to filter songs by title
- Playlist management with user-based tracking
- Secure, token-based access for protected routes
- Cloudinary integration for media (songs and images) hosting
- Responsive and modern UI using React.js

### 3. Architecture

**Frontend (React):** Built using React with Axios for API calls and Context API for managing global authentication state. The application follows a modular component-based structure, includes responsive layouts, and dynamically renders data such as albums, songs, and user playlists fetched via APIs.

**Backend (Node.js + Express.js):** RESTful API built with Express. Handles JWT-based authentication and authorization, admin and user role distinction, file upload with Cloudinary, and CRUD operations for albums and songs.

**Database (MongoDB):** MongoDB stores user, album, and song data. Mongoose is used to define schemas for users, albums, and songs. Each song is associated with an album, and user preferences like playlists are stored securely.

### 4. Setup Instructions

#### Prerequisites:

- Node.js (v18+)
- MongoDB (local or Atlas)
- Git
- npm

#### Installation:

```
git clone https://github.com/ItzVirAj/TuneTrails---Music-Streaming-Web-App.git
cd TuneTrails---Music-Streaming-Web-App
```

Create .env file in both /backend folders:

```
DB_NAME=<DBNAME>
```

In /backend index.js replace

```
CLOUDINARY_CLOUD_NAME=<Cloudinary Name>
CLOUDINARY_API_KEY=<API Key>
CLOUDINARY_API_SECRET=<API Secret>
```

Install dependencies:

```
cd frontend
```

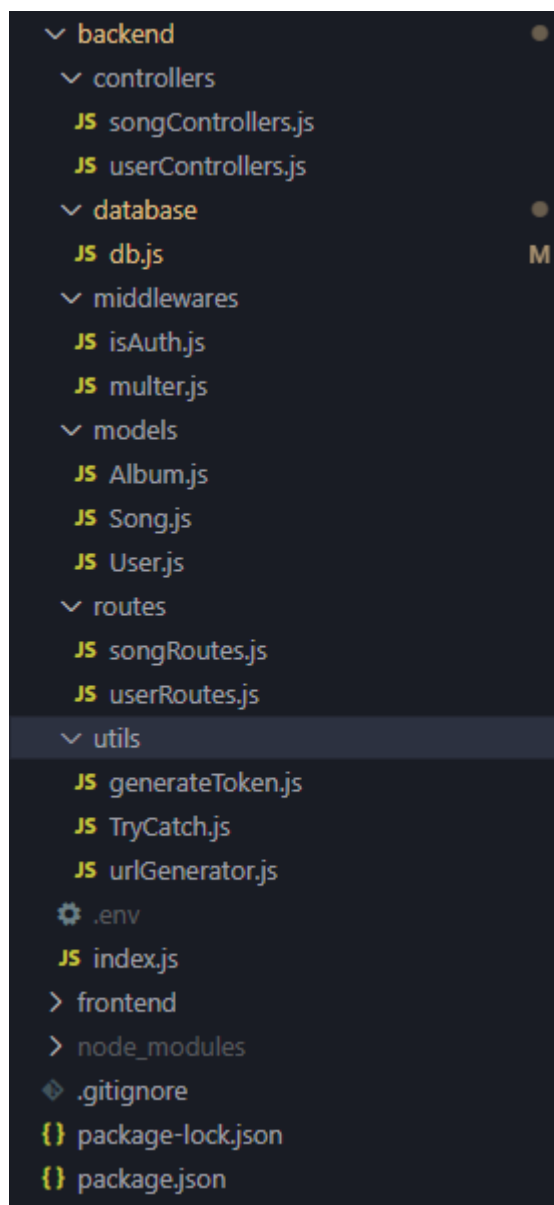
```
npm install
```

```
cd ../backend
```

```
npm install
```

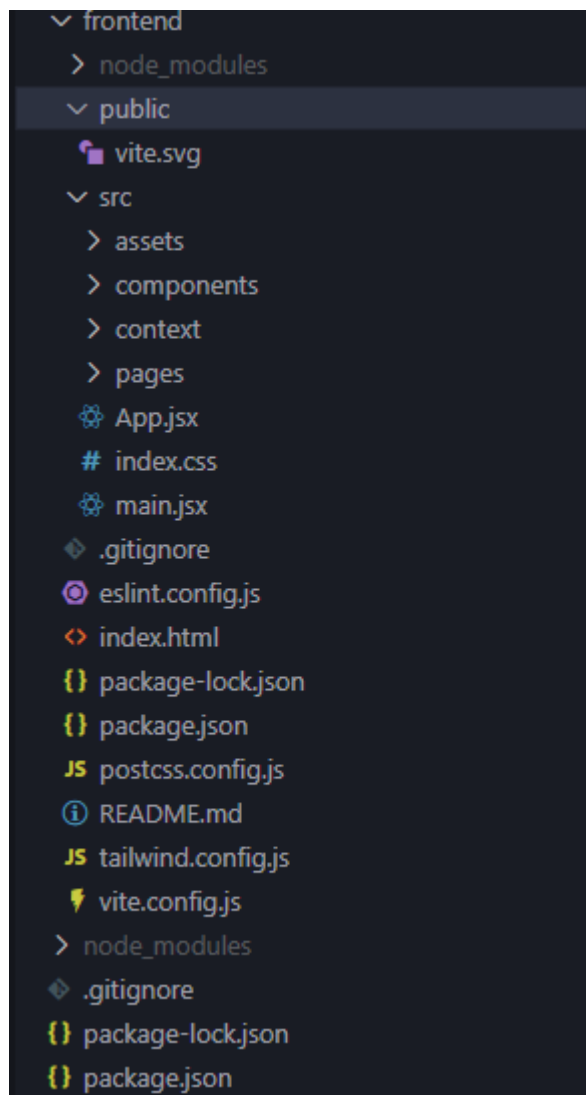
## 5. Folder Structure

Backend Folder Structure -



(package-lock.json , package.json for backend are in root folder)

## Frontend Folder Structure -



## 6. Running the Application

Provide commands to start the frontend and backend servers locally.

### Frontend:

```
cd tunetrails
```

```
cd frontend
```

```
npm run dev
```

### Backend:

```
cd tunetrails
```

```
cd backend
```

```
npm run dev
```

## 7. API Documentation

Endpoint	Method	Description	Auth Required
/api/user/register	POST	Register new user	No
/api/user/login	POST	Login user	No
/api/admin/login	POST	Login admin	No
/api/album	POST	Add new album	Yes(admin)
/api/album	GET	Get All album	No
/api/song	POST	Add new Songs	Yes(admin)
/api/song	GET	Get All Song	No
/api/song/:id	GET	Get Single Song	No
/api/song/album/:id	GET	Get songs by album	No
/api/song/:id	DELETE	Delete song	Yes(admin)

### Example response

```
{  
  "_id": "123",  
  "title": "Night Sky",  
  "singer": "Alan Walker",  
  "audio": {  
    "url": "https://cloudinary.com/audio.mp3"  
  },  
  "album": "6421abc"  
}
```

## 8. Authentication

**Method:** JWT stored in cookies or headers.

### Process:

- On login/register, a token is issued.
- Axios attaches the token for protected routes.
- Backend verifies token and allows access based on role (admin/user).

**Libraries** Used: jsonwebtoken, bcryptjs, cookie-parser

## 9. User Interface

The UI offers a smooth, responsive experience with a modern music dashboard. Users can browse through albums, play music via embedded audio controls, and manage their playlists. Admins can upload new albums and songs via the admin panel. A clean login/register flow ensures user access control.

## 10. Testing

**Tools Used:** Postman, browser testing

**Strategy:**

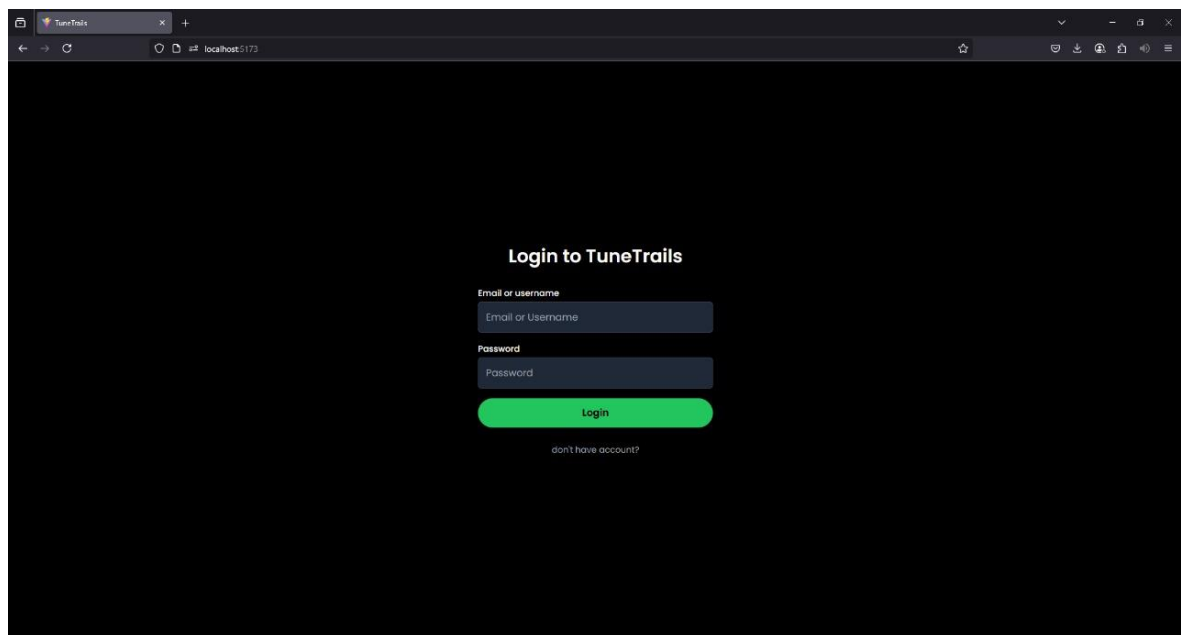
- Valid and invalid API requests tested
- JWT validation and protected route testing
- UI tested across Chrome, Edge

## 11. Screenshots or Demo

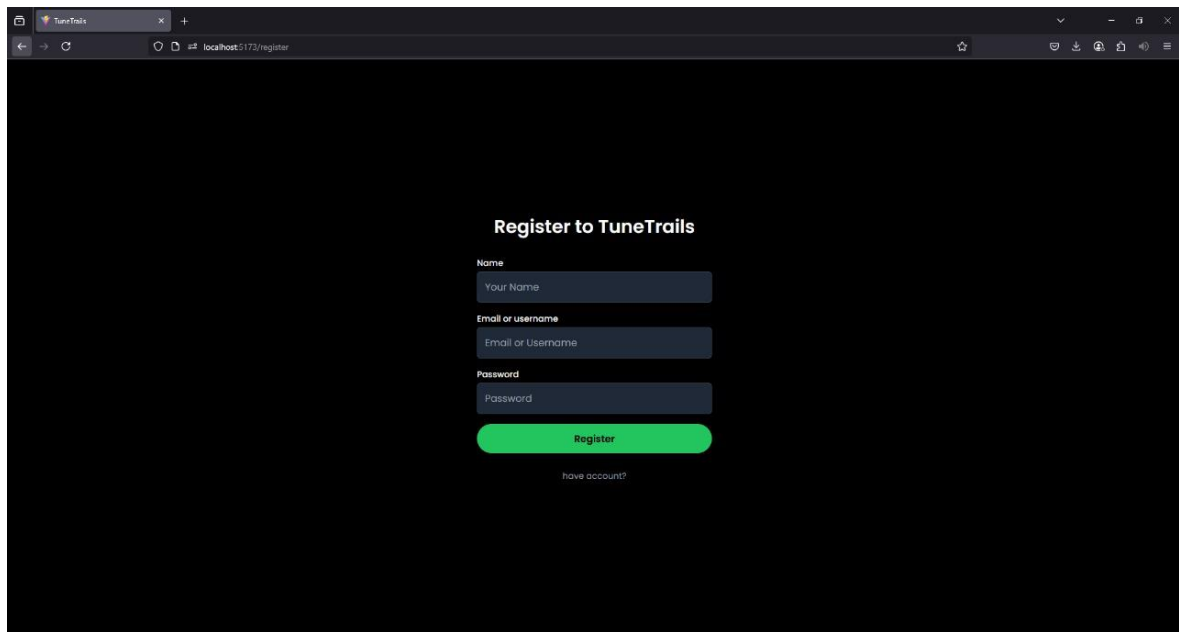
- GitHub Repo: [TuneTrails-MusicApp](#)
- TuneTrails Demo Video: [Video Demonstration](#)

**Sample Screenshots:**

Login Page

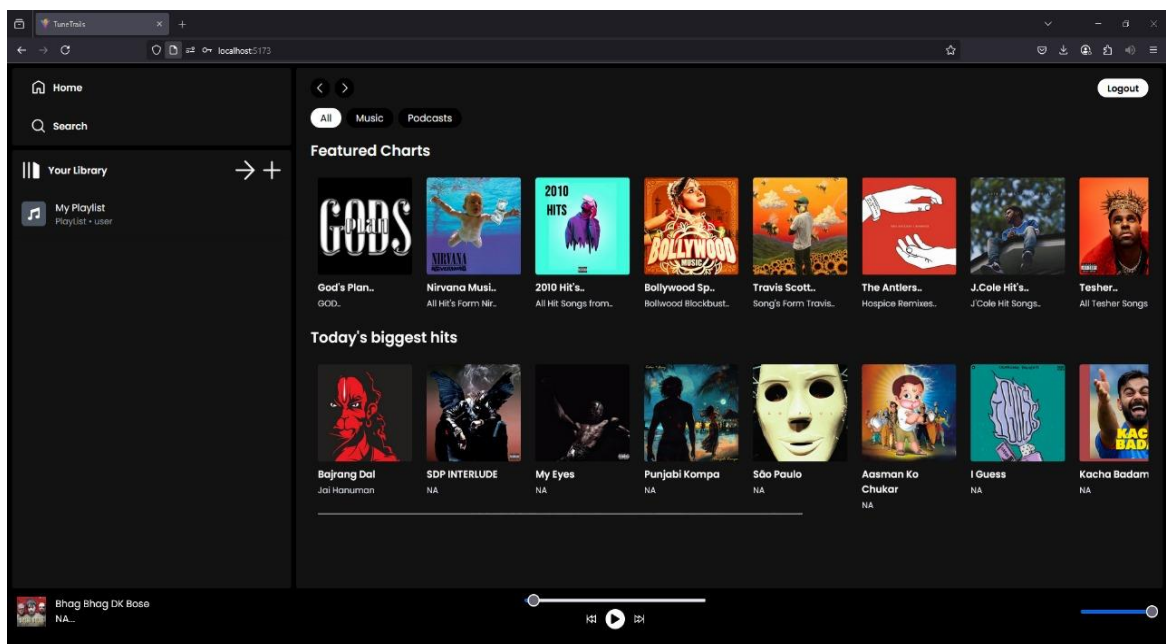


## Register Page



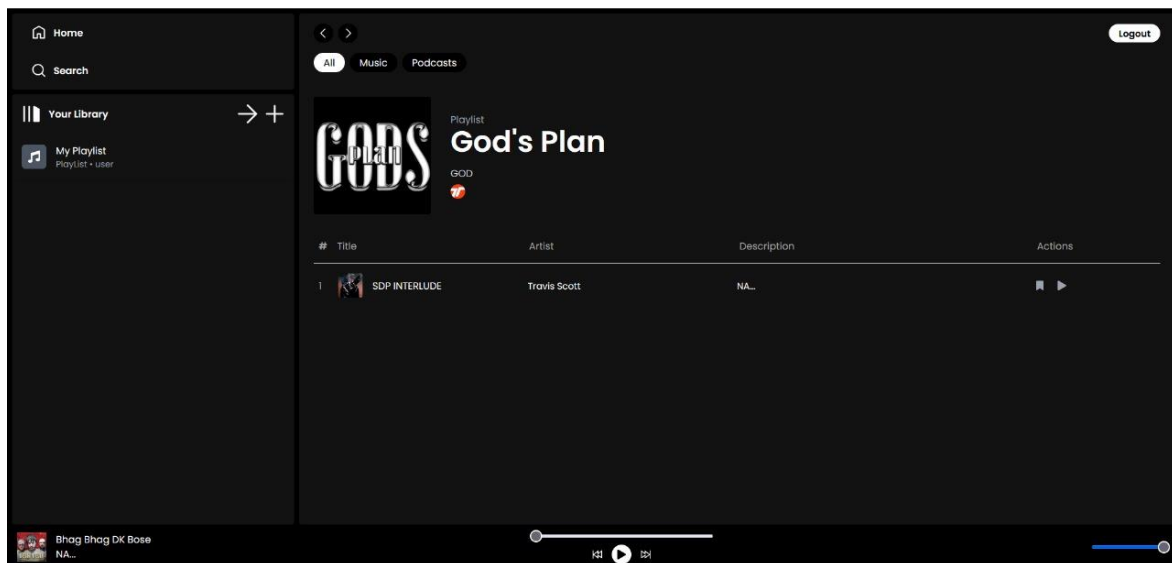
A screenshot of a web browser showing the registration page for TuneTrails. The browser's address bar displays 'localhost:5173/register'. The page has a dark background with a white registration form in the center. The form is titled 'Register to TuneTrails' and includes three input fields: 'Name' (placeholder: 'Your Name'), 'Email or username' (placeholder: 'Email or Username'), and 'Password' (placeholder: 'Password'). Below these fields is a green 'Register' button. At the bottom of the form, there is a link that says 'have account?'.

## User home page

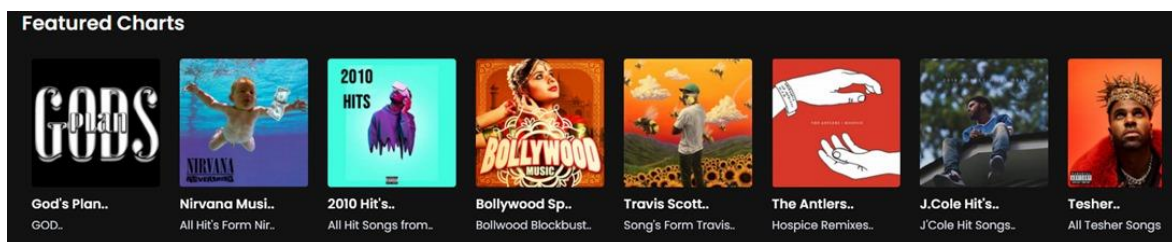




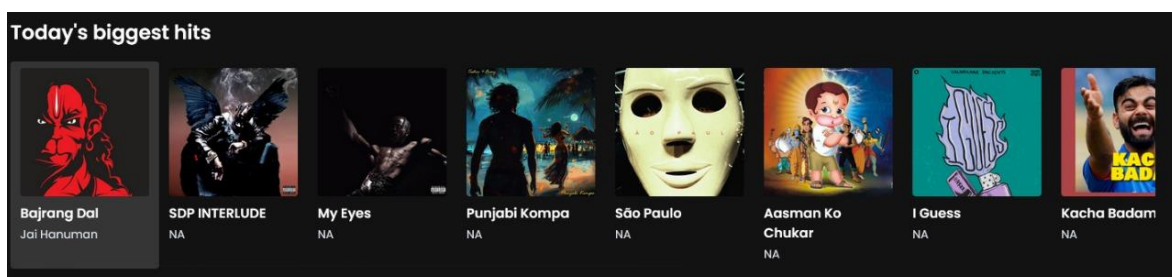
## Album page



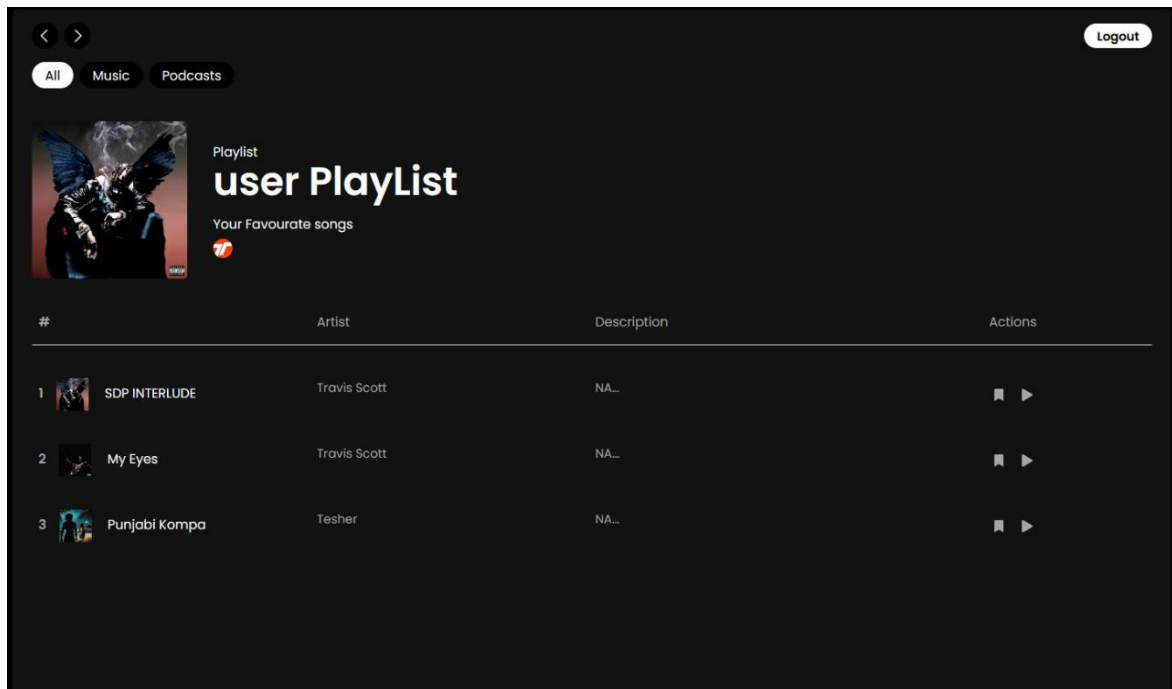
## Featured Albums



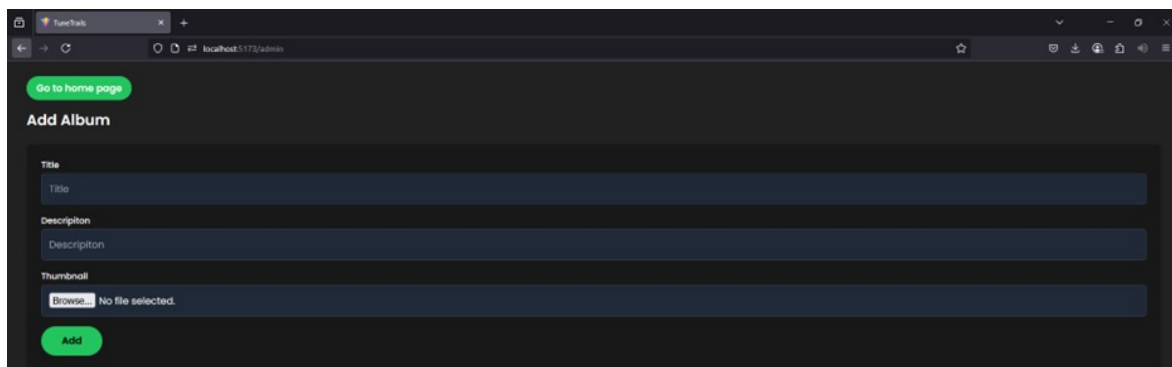
## Biggest Hits Section

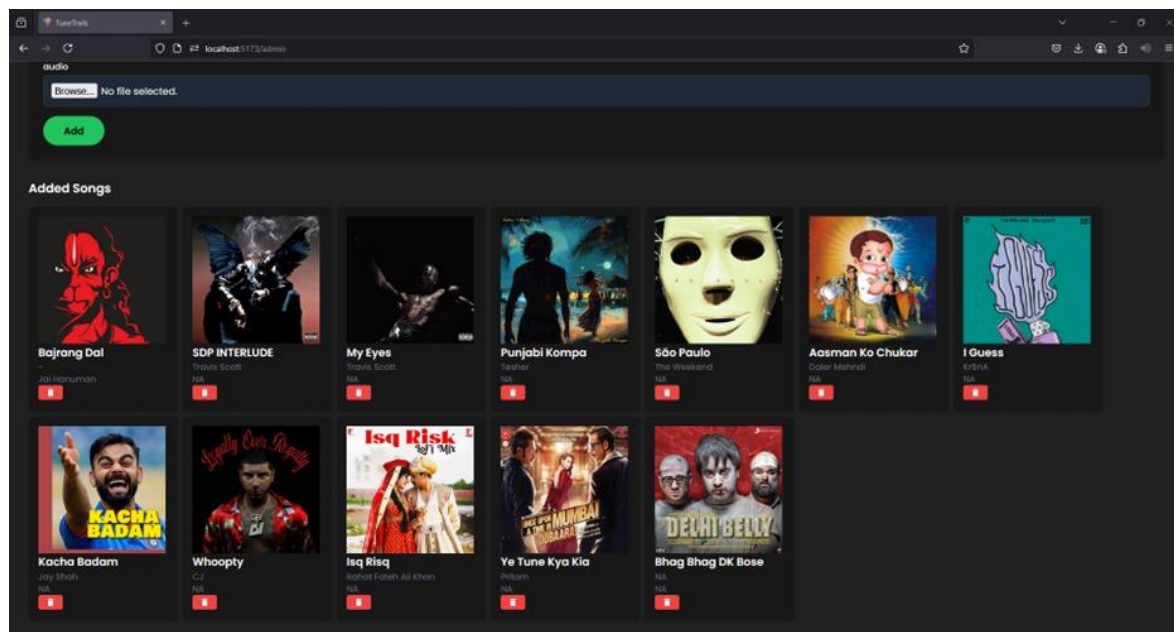


## Payment successful page



## Admin add songs/albums page





## 12. Known Issues

- Search suggestions/autocomplete yet to be implemented
- No pagination on albums or songs
- No Light mode UI

## 13. Future Enhancements

- Like/Favorite functionality for songs
- Playlist creation/edit support
- Music genre filtering
- Playback history and analytics
- Responsive mobile layout improvements
- Admin analytics dashboard