

Final Report

Tune Trails (Music Streaming App)

1. INTRODUCTION

1.1 Project Overview

TuneTrail is a MERN stack-powered music streaming app that delivers a seamless experience for discovering, streaming, and managing music. Users can explore curated playlists, enjoy offline listening. The app also features artist profiles with their exclusive playlists, creating a dynamic platform for both music lovers and independent creators.

1.2 Purpose

TuneTrail is a music streaming app built with the MERN stack, designed to bring you high-quality, legal music in a smooth and enjoyable way. Whether you're vibing to curated playlists or listening offline, TuneTrail makes it easy. By focusing on licensed tracks and verified artist profiles, it not only gives users a great experience but also supports the musicians behind the music.

2. IDEATION PHASE

2.1 Problem Statement

Many music lovers, especially students, struggle to enjoy seamless music streaming due to expensive premium plans, ad-heavy free versions, and lack of personalized recommendations. There is a need for an affordable, user-friendly music streaming platform that offers personalized experiences without constant interruptions.

2.2 Empathy Map Canvas

Think & Feel:

Wants an uninterrupted, personalized music experience that fits their mood and taste.

See:

Cluttered music apps, excessive ads, and limited skips in free versions.

Hear:

Friends talking about better premium apps or complaining about poor recommendations.

Say & Do:

Expresses frustration with constant ads, skips songs quickly, and searches for playlists manually.

Pain:

Ads breaking the vibe, poor song recommendations, and limited control in the free version.

Gain:

Desires an intuitive, ad-light platform with personalized playlists and smooth playback.



2.3 Brainstorming

Each team member suggested ideas ranging from secure user authentication to personalized music recommendations, and we prioritized them using an idea priority matrix to focus on features with the highest user impact and feasibility.

Viraj Handled backend authentication using JWT, structured the MongoDB schema, and created admin-specific routes.	Prathamesh Designed and implemented the UI/UX for login screens and responsive layout, including trending music carousel.
Aniket Built core music recommendation features based on genre and tags, and enabled tracking of user favorites.	Darpan Implemented visual themes (dark/light mode), interactive UI animations, and a consistent color scheme.

3. REQUIREMENT ANALYSIS

3.1 Customer Journey map

Awareness: Learn about the music app through social media, app stores, or word-of-mouth.

Consideration: Explores features like playlists, trending songs, and artist pages.

Onboarding: Signs up, sets music preferences, and customizes their experience.

Experience: Streams songs, creates playlists, explores recommendations.

Feedback: Likes/dislikes tracks, rates the app, and shares songs with friends.

3.2 Solution Requirement

User Side:

Browse songs, stream music, create/manage playlists, mark favorites, set preferences.

Admin Side:

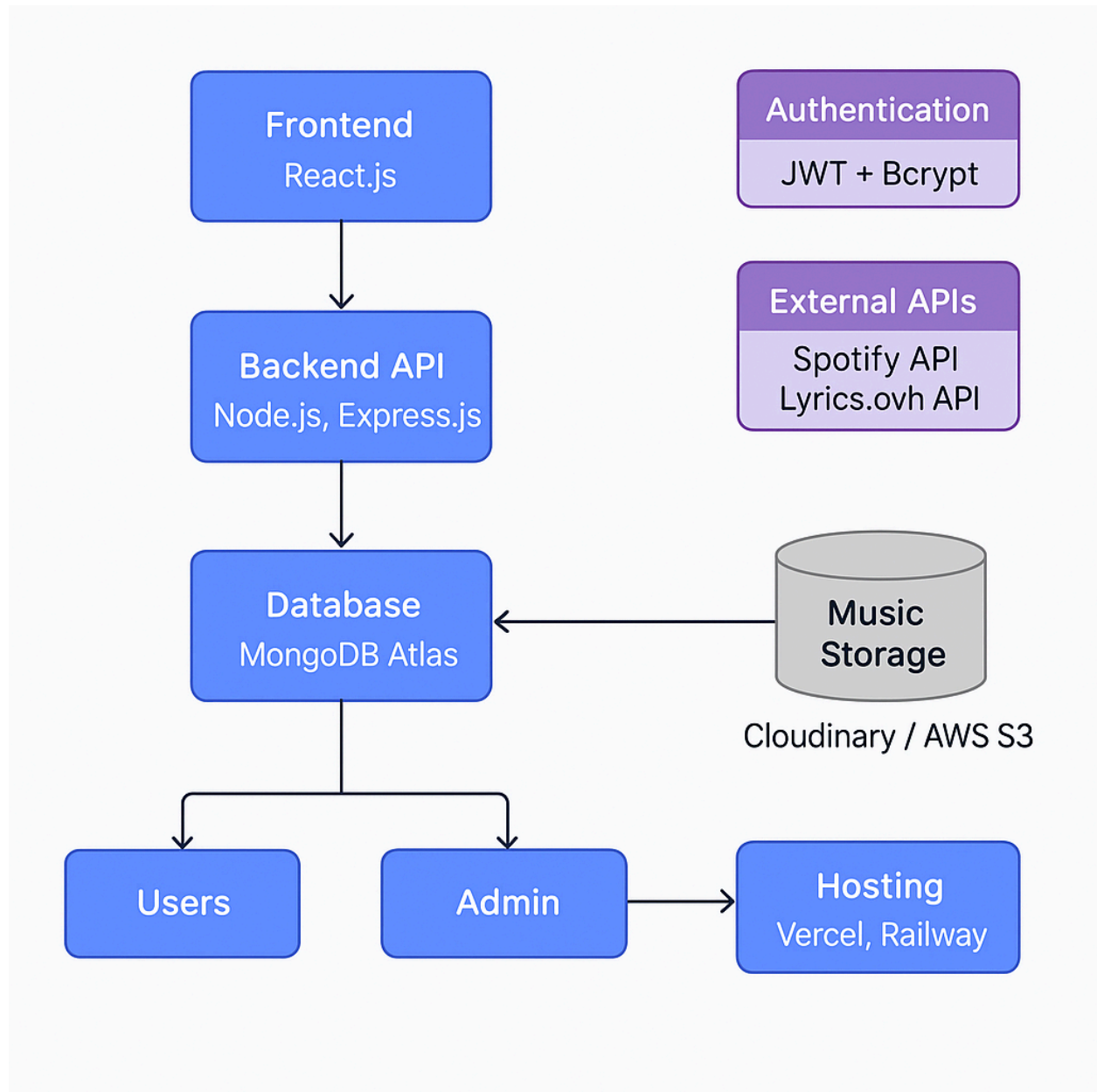
Upload/manage music library, monitor user activity, manage subscriptions and analytics.

System:

Secure authentication, fast streaming, user data encryption, scalable backend, real-time recommendation engine.

3.3 Data Flow Diagram

A DFD was created to illustrate the interaction between users, the music player UI, backend services, and the music database.



3.4 Technology Stack

- **Frontend**

Technology: HTML, SCSS, JavaScript, React.js

The user interface is responsive and optimized for mobile and desktop views.

- **Backend**

Technology: Node.js, Express.js

Manages APIs for registration, login, music browsing, playlist handling, and admin operations.

- **Authentication**

Technology: JWT, bcrypt

Secures user sessions and protects sensitive credentials.

- **Database**

Technology: MongoDB (NoSQL), hosted on MongoDB Atlas

Stores user data, song metadata, playlists, favorites, and admin settings.

- **Music Storage**

Technology: AWS S3 or Cloudinary (planned)

Stores and streams high-quality audio files and album artworks.

- **External APIs (Planned)**

Spotify API / Lyrics.ovh API

For lyrics fetching, song previews, and artist metadata.

- **Hosting**

Frontend: Vercel

Backend: Railway

4. PROJECT DESIGN

4.1 Problem Solution Fit

Solution for Tune Trails (Music Streaming App)

Customer Segment(s) Music lovers, college students, working professionals, casual listeners, playlist creators, families who stream music online.		Customer Limitations Budget-conscious users, basic smartphones, average internet connectivity, non-tech-savvy people, limited storage for offline music.	Available Solutions Spotify, Gaana, JioSaavn, Apple Music (Popular but have ads/fees, limited personalization). Local apps (limited music library, outdated UI).
Problems / Pains Difficult UI in existing platforms, frequent ads, limited skips, lack of personalized recommendations, hard to discover new music, limited offline downloads.		Problem Root / Cause Centralized platforms are profit-driven, not customer-first. Lack of focus on smart music discovery and seamless user experience.	Behavior Frequently search for new songs/playlists, compare app features, share music with friends, use free versions to avoid fees, skip songs often.
Triggers to Act New album releases, trending playlists, friend recommendations, social media buzz, artist promotions.	Emotions Before: Frustration, boredom, confusion. After: Satisfaction, relaxation, excitement.	Your Solution A MERN-based, ad-supported music streaming platform with no hidden fees, smart personalized recommendations, seamless and intuitive UI, and easy offline access.	Channels of Behavior Online: Mobile apps, websites, social media. Offline: Word of mouth, posters, campus events.

4.2 Proposed Solution

A MERN stack-based web platform with separate flows for users and admins, offering smooth navigation, personalized music streaming, playlist management, and efficient content administration.

4.3 Solution Architecture

Consists of:

- **Frontend (ReactJS):**
User-friendly interface with routing, real-time playback controls, and state management for playlists and preferences.
- **Backend (NodeJS + ExpressJS):**
Handles core logic, user authentication, music streaming APIs, and recommendation services.
- **Database (MongoDB):**
Stores user data, music tracks, playlists, playback history, and admin content metadata.

5. PROJECT PLANNING & SCHEDULING

5.1 Project Planning

- **Week 1:** Ideation and Problem Statement Finalization
Brainstorm features, define user/admin roles, finalize tech stack.
- **Week 2:** Frontend–Backend Setup
Initialize React frontend, set up Node.js + Express backend, connect MongoDB.
- **Week 3:** Implement Authentication & Music Streaming Flow
User login/signup, secure authentication, implement music browsing and playback.
- **Week 4:** Admin Features & UI Enhancements
Admin dashboard, music upload, analytics, responsive UI/UX design.
- **Week 5:** Testing, Debugging & Final Deployment
Conduct end-to-end testing, resolve bugs, deploy to production (e.g., Render/Netlify + MongoDB Atlas).

6. FUNCTIONAL AND PERFORMANCE TESTING

6.1 Performance Testing

Tools: Browser DevTools, Postman

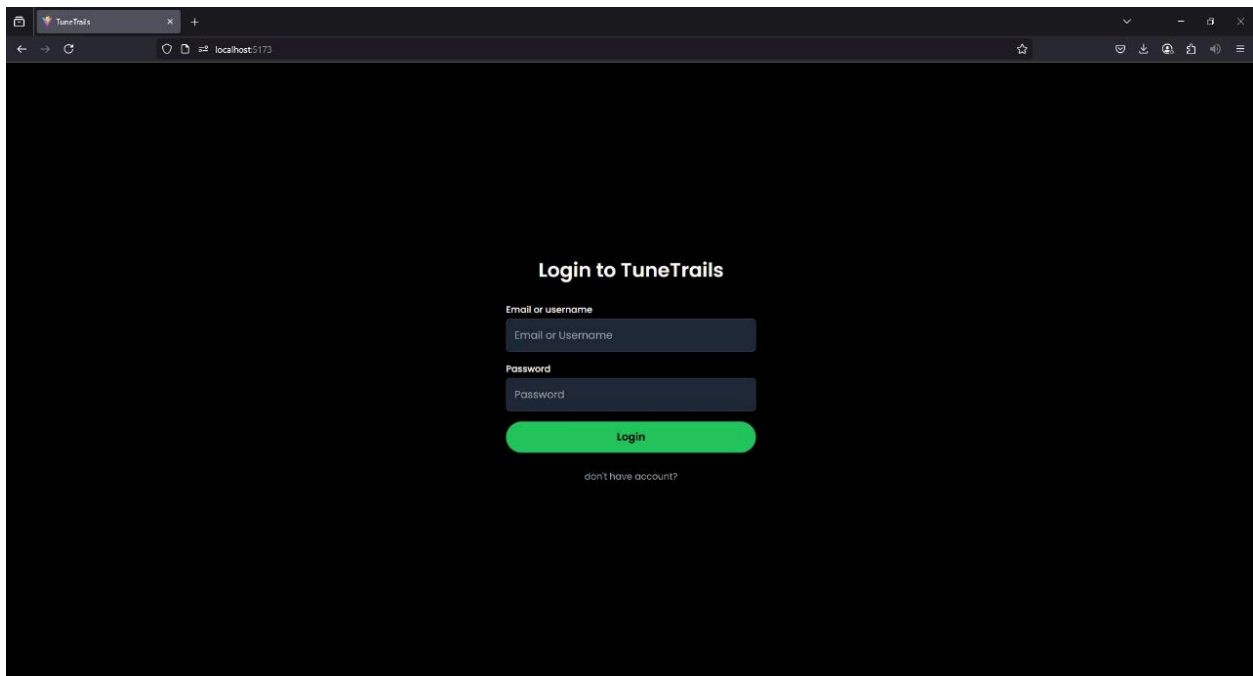
Focus: API response time, login speed, load handling

Result: The app remains stable under load and exhibits optimized response times for a seamless user experience.

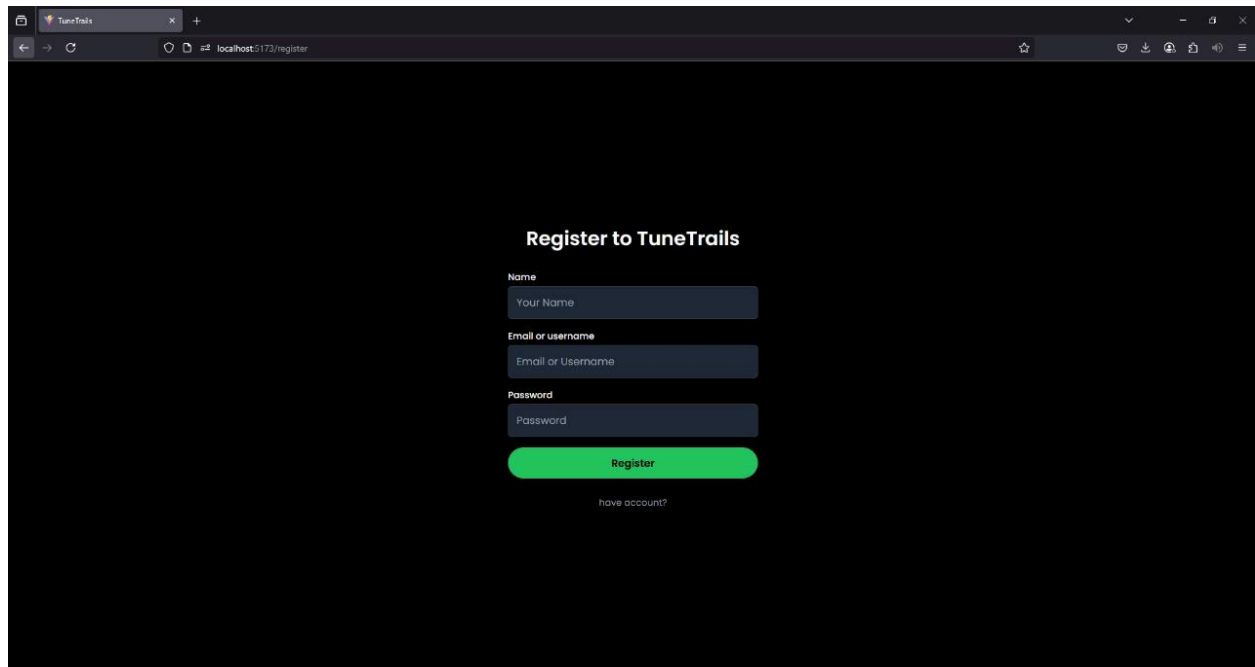
7. RESULTS

7.1 Output Screenshots

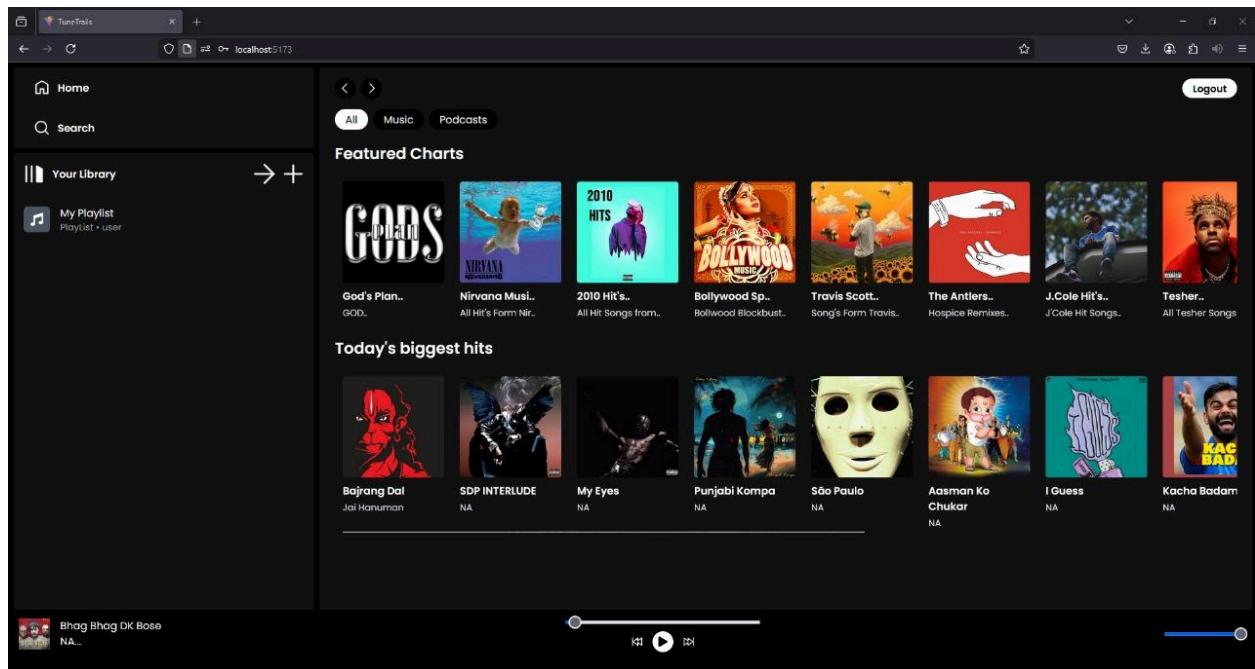
Login Page



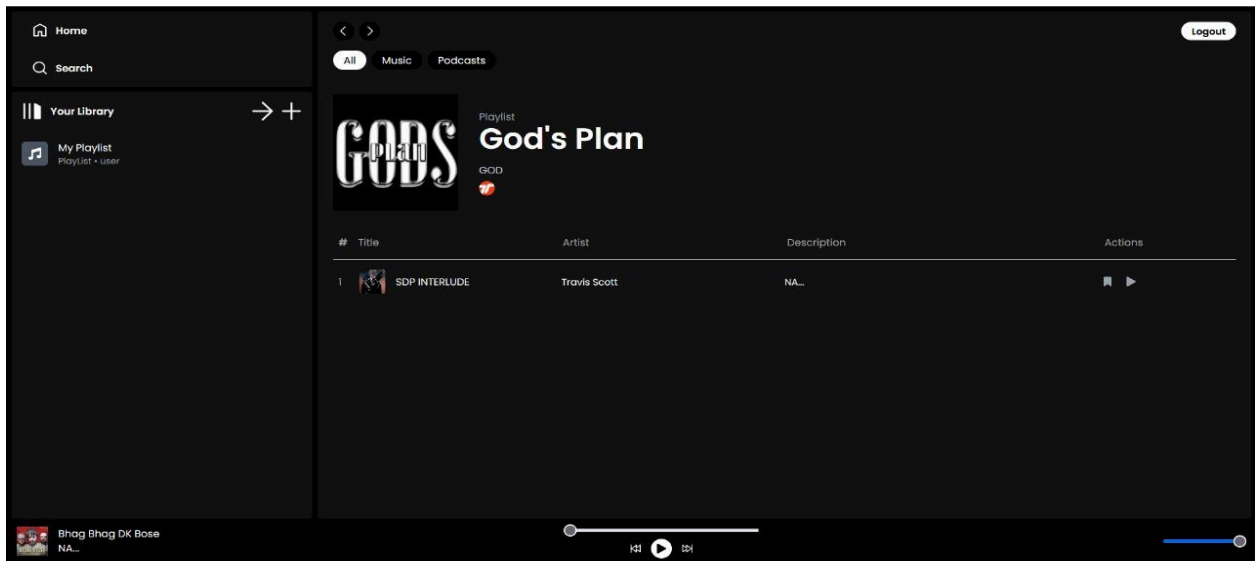
Register Page



User Home Page



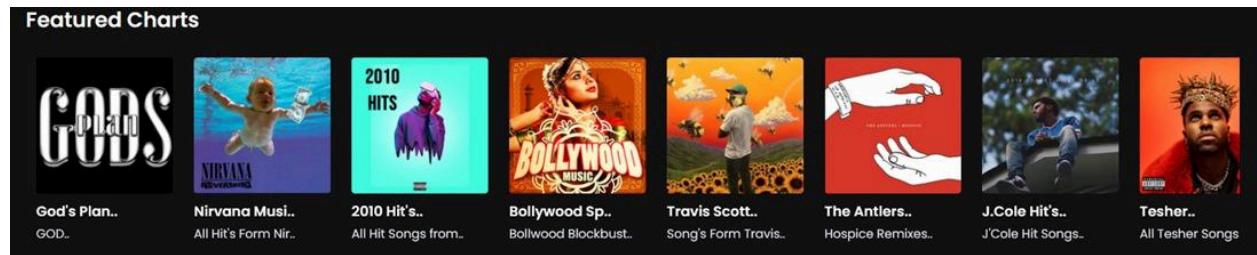
Album Page



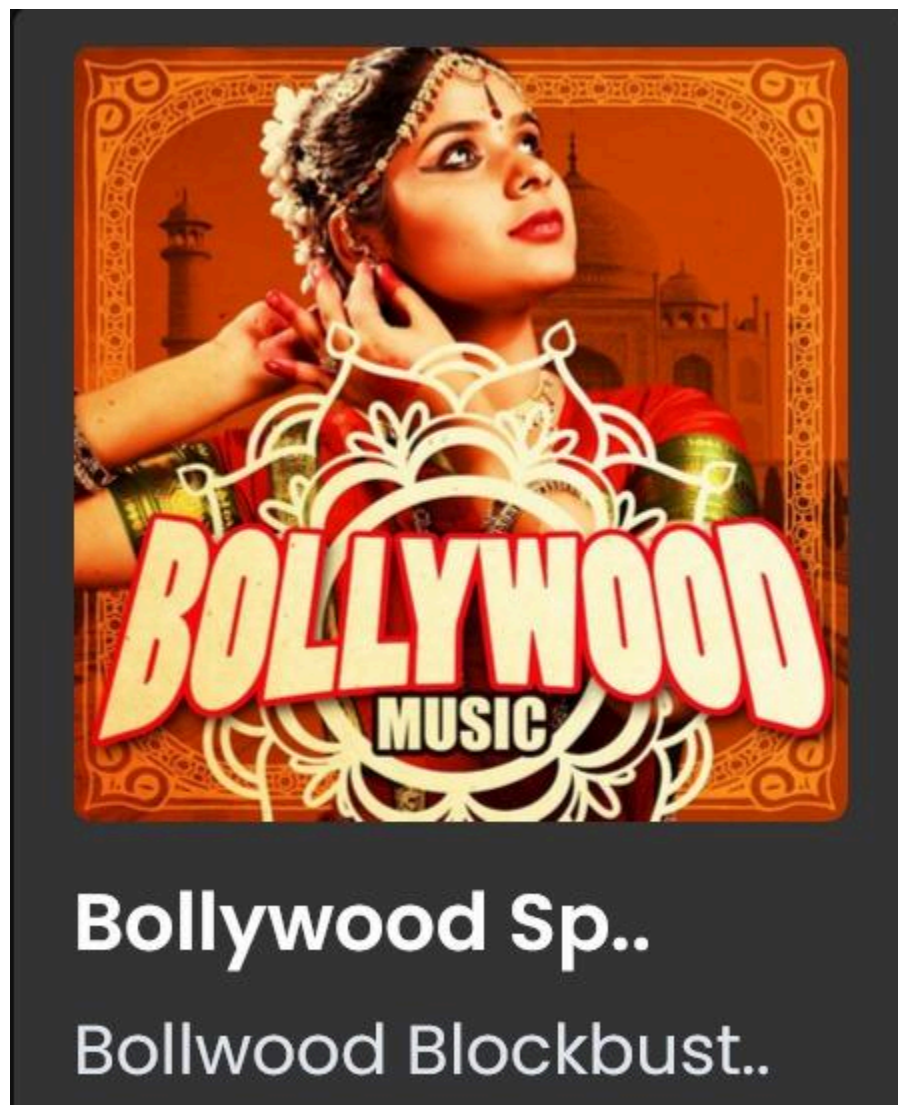
Song Cover(Song_Name, Artist_Name)



Featured Album


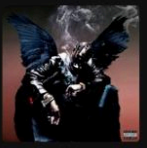
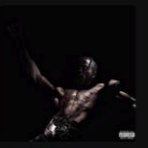







Album Cover





Today's Hit collection




Today's biggest hits


 Bajrang Dal Jai Hanuman	 SDP INTERLUDE NA	 My Eyes NA	 Punjabi Kompa NA	 São Paulo NA	 Aasman Ko Chukar NA	 I Guess NA	 Kacha Badam NA
--	---	---	---	---	--	---	---


Sidebar

 **Home**

 **Search**

 **Your Library**  

 **My Playlist**
PlayList • user

 **Bhag Bhag DK Bose**
NA...


Personal Playlist

<

>

Logout


AllMusicPodcasts












Playlist

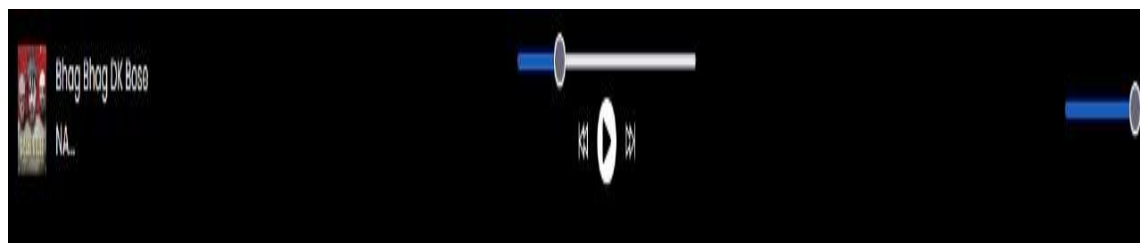
user PlayList

Your Favourite songs

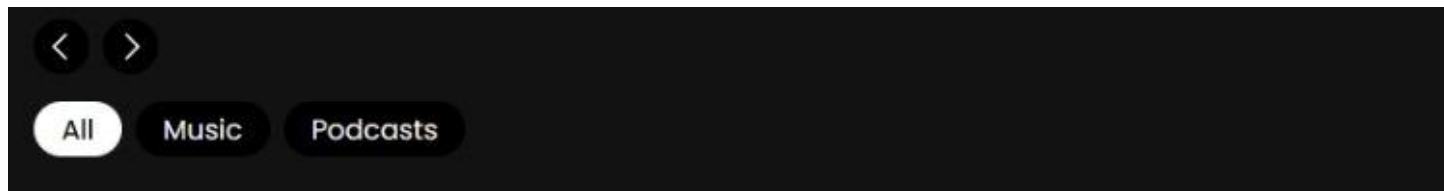


#		Artist	Description	Actions	
1		SDP INTERLUDE	Travis Scott	NA...	 
2		My Eyes	Travis Scott	NA...	 
3		Punjabi Kompa	Tesher	NA...	 

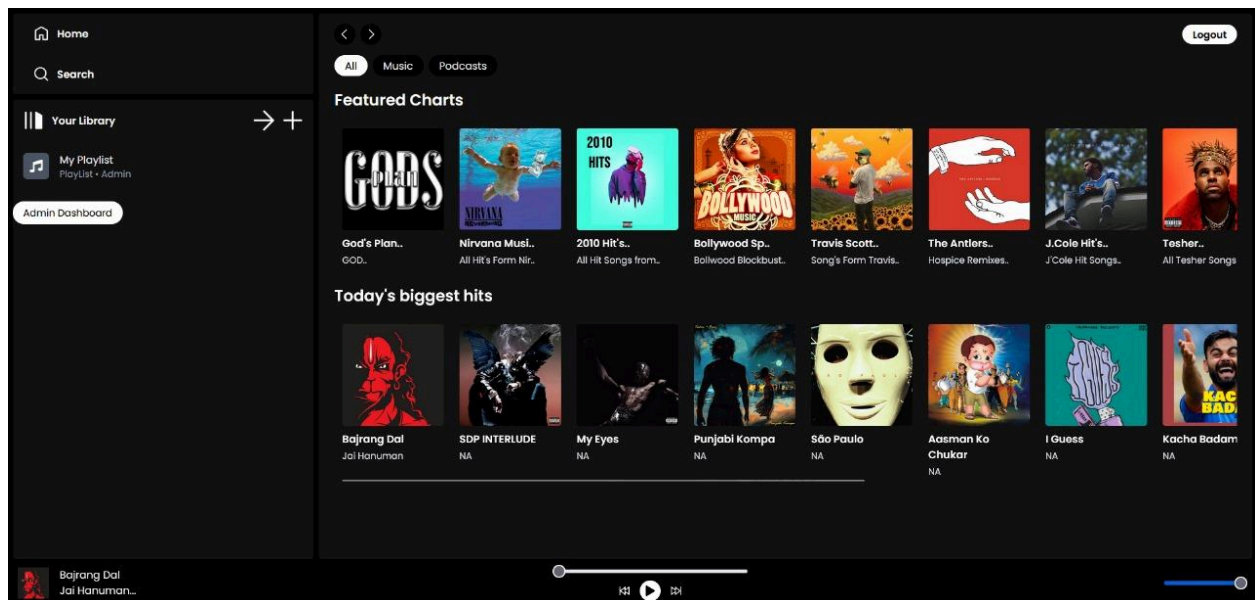
Music Bar



Upper Navigation Bar



Home Page(Admin)



Add, Delete the song(Admin)

A screenshot of a web browser window showing a form titled "Add Album". The form has three input fields: "Title", "Description", and "Thumbnail". The "Thumbnail" field has a "Browse..." button and the text "No file selected." below it. There is a green "Add" button at the bottom of the form. The browser's address bar shows "localhost:3173/admin".

8. ADVANTAGES & DISADVANTAGES

Advantages

1. **Scalable:** MongoDB, Express, React, and Node.js are great for handling large data and traffic.
2. **Real-time Features:** Can implement low-latency music streaming with WebSockets.
3. **User Customization:** Personalized playlists, recommendations, and user accounts are easy to integrate.
4. **Cross-platform:** Easily extendable to mobile with React Native.
5. **Fast Development:** Pre-built tools and strong community support speed up development.

Disadvantages

1. **Performance:** MongoDB might struggle with complex queries as data grows.
2. **Streaming Complexity:** Efficient audio streaming and low-latency playback are challenging.

3. **Security Risks:** Needs strong DRM and user data protection.
4. **Scaling Infrastructure:** As the user base grows, scaling the app's backend can be tricky.
5. **Monetization:** Converting free users to paid subscribers is challenging.

9. CONCLUSION

Tune Trails successfully delivers a seamless and efficient music streaming experience using the MERN stack. With robust features and an intuitive design, it addresses common user frustrations and modernizes the music listening process.


10. FUTURE SCOPE

Tune Trails successfully delivers a seamless and efficient music streaming experience using the MERN stack. With robust features and an intuitive design, it addresses common user frustrations and modernizes the music listening process. Key improvements include:

- Mobile App Version (React Native): Expanding access to iOS and Android devices.
- Payment Gateway Integration: Enabling smooth and secure premium subscriptions.
- Social Login and User Reviews: Simplifying sign-ups and allowing users to share feedback.
- Enhanced Admin Analytics: Providing detailed graphs and reports for better decision-making.

11. APPENDIX

Source Code:

 MERN_Stack_VideoReview - Made with Clipchamp_1745054748080.mp4

GitHub & Project Demo Link:

<https://github.com/ItzVirAj/TuneTrails-MusicApp>

Team Members:

- Viraj Mane - 22BCE11473
- Prathamesh Chaudhari - 22BCE10926
- Aniket Shankarwar - 22BCG10061
- Darpan Nemade - 22BAI10197