SmartBridge

Full Stack Web Development using MERN

FINAL REPORT

Project Name: TuneTrail

Team ID: SWTID1743512046

1. INTRODUCTION

1.1 Project Overview

With the ever-expanding universe of digital music and the growing demand for personalized listening experiences, traditional playlist creation methods are often time-consuming and lack the sophistication needed to perfectly match individual tastes. This project aims to develop "TuneTrail," a web-based platform that empowers users to effortlessly discover, create, and share perfect playlists tailored to their unique preferences.

The platform is designed with a user-centric approach, supporting multiple roles including listeners, curators, and administrators. Listeners can register, explore a vast library of songs based on genre, mood, and artist, and create playlists in real-time. Curators can create and share playlists, gaining followers and recognition for their musical taste, while administrators oversee platform operations and manage user roles and permissions.

The application is built using the MERN stack (MongoDB, Express.js, React.js, Node.js) to ensure scalability, performance, and maintainability. Key features include JWT-based authentication, cloud database storage, responsive design, and RESTful APIs that support smooth frontend-backend communication. The UI is designed to be intuitive and mobile-friendly, enhancing usability across various devices.

Beyond basic playlist creation, the system emphasizes data integrity, role-based access control, and security, ensuring all user data is protected and managed properly. The platform is flexible for future enhancements such as AI-driven music recommendations, collaborative playlist editing, and integration with popular streaming services.

This documentation serves as a complete guide, covering the project's purpose, core features, technical architecture, development process, and scope—making it a valuable reference for developers, evaluators, and future contributors.

1.2 Purpose

The TuneTrail app aims to revolutionize music playlist creation by offering an online platform for discovering and curating personalized playlists. It eliminates the need for endless manual searching and guesswork, saving time for listeners, curators, and administrative staff.

Objectives

- Develop a secure and scalable MERN stack application.
- Enable listeners to search for songs by genre, mood, artist, or popularity, and create playlists.
- Allow curators to share playlists, gain followers, and showcase their musical taste.
- Provide admins with tools to oversee users, curators, and playlists.
- Ensure a responsive and intuitive user interface for seamless interaction across devices.
- Implement robust authentication and authorization mechanisms.

Target Audience

- **Listeners:** Individuals seeking convenient and personalized music experiences.
- **Curators:** Music enthusiasts looking to share their playlists and influence.
- **Admins:** System administrators responsible for platform oversight.

2. IDEATION PHASE

2.1 Problem Statement

Music lovers often face difficulty in discovering, organizing, and accessing their favorite tracks and playlists across platforms.
Tune Trail solves this by offering a seamless, personalized music management platform for discovering songs, creating playlists, and enjoying a smooth playback experience..

Customer Problem Statements

Listener Perspective:

Problem Statement	I'm	I'm trying to	but	because	Which makes me feel
P S -1	A music enthusiast	Discover and organize my favorite songs	I find it hard to explore and save songs in one place	Most platforms focus on streaming and lack advanced playlist features	Frustrated by the scattered music experience
P S - 2	A user with diverse music taste	Build mood-based or genre-based playlists	I have to switch between apps or manually search every time	There's no intelligent recommendation and playlist creation system	Overwhelmed and unsatisfied with my listening flow

Artist Perspective:

Problem Statement	I'm	I'm trying to	but	because	Which makes me feel
PS-1	An independent artist	Share my music with engaged listeners	It's difficult to get discovered organically	Existing platforms are saturated and algorithm-driven	Discouraged and limited in audience reach
PS-2	A growing musician	Understand listener preferences	I get little to no feedback or analytics	There's limited access to audience insights for small creators	Disconnected from my fans and unsure how to grow

Key Features Based on Problem Statements

1. **Seamless Music Discovery**
   - Smart search and filter options based on genre, mood, and popularity
   - Curated song lists and trending track suggestions
     Personalized recommendations based on listening behavior
2. **Playlist Management**
   - Create ,edit, and organize custom playlist
   - Drag and drop functionality for reordering tracks
   - Option to categorize playlist by mood , activity or genre

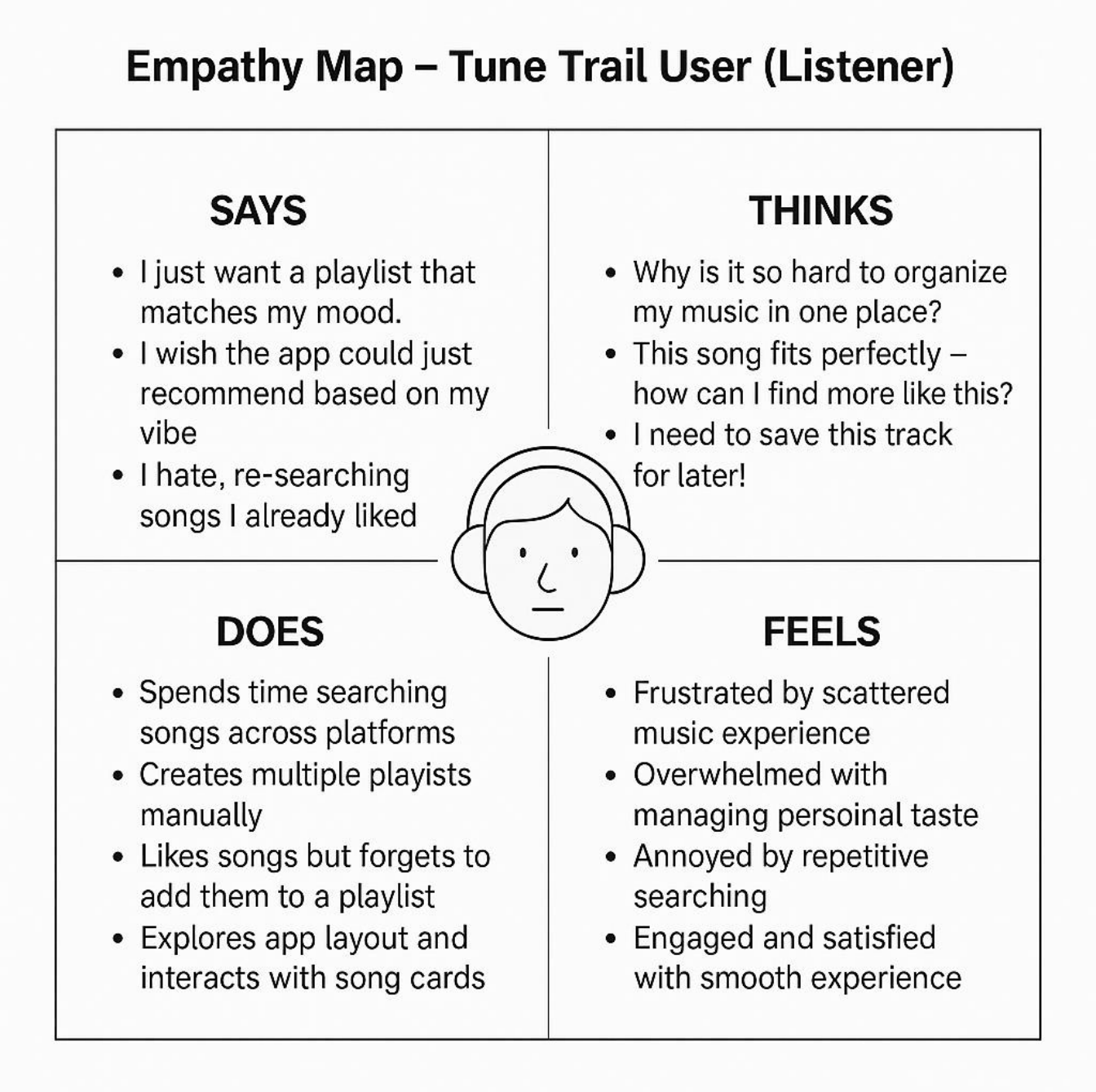
3. **User Profile & Preference**
    - Save favorite songs and artist to personal library
    - Track recently played songs and listening stats
    - Customize playback settings and themes

This feature set is designed to directly address listener and artist frustrations, delivering a unified, enjoyable music experience for all users of the **Tune Trail** platform.

2.2 Empathy Map Canvas:

To better understand the needs and experiences of our users, we created an Empathy Map Canvas based on interactions, UI feedback, and observed behaviors within the Tune Trail platform. Here's what we discovered:

Empathy Map: User(listener)



This empathy mapping exercise allows us to stay user-centric while developing Tune Trail, ensuring both listeners and creators find value and satisfaction in the platform..

2.3 Brainstorming

Using collaborative brainstorming sessions, whiteboarding tools like Miro, and user interviews, we generated multiple ideas to enhance the experience for both music listeners and artists on Tune Trail. Below is a summary of the key stages and concepts discussed:

Step-1: Team Gathering, Collaboration, and Problem Statement Selection

During our initial discussions, we identified a core issue: music platforms often lack personalization, proper recognition for emerging artists, and intuitive tools for discovering and sharing songs.

Selected Problem Statement:

"I am a music enthusiast looking to easily discover and listen to my favorite or trending songs, but I face difficulty navigating cluttered interfaces and lack of personalized content, which makes the experience feel frustrating and impersonal."

This statement reflects a common user struggle and aligns with Tune Trail's mission to simplify music discovery while empowering both listeners and artists.

Step-2: Brainstorm, Idea Listing, and Grouping

We held focused brainstorming sessions and listed a wide array of features aimed at solving the identified problem. These ideas were categorized into four key themes:

1. Listener-centric feature

- Real-time search with filters
- Top 5 songs with star ratings
- Easy to use audio player with seek and volume controls
- Playlist creation and management
- Favorites or like functionality

2. Artist Dashboard Features

- Upload and manage songs
- View real – time song performance
- Add album/genre/artist info with ease
- Feature request for spotlight/trending section

3. Technical Enhancements

- MongoDB for scalable data storage
- TailwindCSS for modern , responsive UI
- Audio streaming optimization
- React + Express for full stack scalability

Step-3: Idea Prioritization

We categorized each idea based on **Impact** and **Feasibility** to define development phases

Feature/Idea	Impact	Feasibility	Priority
Song Search & Top 5 view	**High**	**High**	**Top**
Audio Player	**High**	**High**	**Top**
Artist Upload & stats Dashboard	**Medium**	**Medium**	**Top**
Playlist creation	**Medium**	**Low**	**Later**

Our brainstorming allowed us to center our MVP around core user needs—music discovery, playback, and artist visibility. We prioritized building a functional and visually engaging frontend, robust backend authentication, and intuitive artist-user flows for early adoption.

3. REQUIREMENT ANALYSIS

3.1 Customer Journey Map

A customer journey map for Tune Trail outlines what both users and artists experience at each stage of interacting with the platform—from discovery to content interaction or song publishing. It highlights key pain points and user goals, helping optimize flow and ensure a personalized and engaging experience.

Stage	User Action	System Response	Touchpoints	Emotions/Goals
Discovery	Learns about Tune Trail via social media or friends	Displays value proposition on landing page	Social media, homepage, ads	Curiosity, interest
Song Discovery	Searches or browses for songs/playlists	Fetches and displays song list based on filters	Search bar, filter, categories	Personalization, excitement
Listening Experience	Plays a selected track or playlist	Loads and streams audio seamlessly	Audio player	Immersion, entertainment
Engagement	Likes songs, creates playlist, shares content	Stores preferences and updates UI dynamically	Like button, share icon, playlist	Expression, collection, social connection
Artist Dashboard	Uploads songs, tracks stats, edits profile	Stores song info, updates dashboard with analytics	Upload page, stats dashboard	Recognition, control, growth
Feedback & Interaction	Leaves reviews or ratings	Updates ratings and visible feedback	Ratings section	Expression, support, impact

Listener Journey:

Stage	Activities
Awareness	**- Learns about the app via ads, friends, or influencers** **- Visits homepage and checks trending/featured songs**
Song Discovery	**- Uses filters or search to browse music** **- Views artist profiles and listens to top tracks**
Listening Experience	**- Plays songs, likes or shares them** **- Adds songs to personal playlists**
Engagement	**- Receives notifications for new releases or playlist activity** **- Submits song ratings or follows artists**

Artist Journey:

Stage	Doctor Activities
Content Management	**- Manages song uploads and track details** **- Edits album info, cover art, tags**
Analytics Monitoring	**- Views Listener stats : plays, like, shares**

Community Engagement	**- Respond to feedback or shares links** **- Collaborates with user/artists**

3.2 Solution Requirements

Functional Requirements

1. **Music Discovery & Interaction**
    - Real-time search with filters
    - Personalized homepage feed based on listener behavior
    - Detailed artist profile with bio , top tracks , and stats
    - Like/favorite songs and playlist
    - Song rating and feedback system

2. **Playlist & playback management**
    - Custom playlist creation and editing
    - Drag and drop track arrangement
    - Audio player with volume/seek controls and queue support
    - Recently played history and resume playback
    - Shuffle and repeat functionality
3. **Artist Dashboard and song management**
    - Upload songs with metadata
    - View analytics
    - Edit or delete
4. **Reviews & Feedback**
    - Listener can rate songs and leave reviews
    - Artist can respond to feedback
    - Moderation of inappropriate reviews
    - Reputation and trending metrics display

Non-Functional Requirements

1. **Performance**

    - Audio streaming with <3s load time
    - Smooth UI experience across devices
    - Handle 2000+ concurrent users
    - Api response time <200ms

2. **Reliability**

    - 99.9% uptime with auto-scaling servers
    - Redundant backups for songs and user data
    - Real-time error logging and alerting
    - Graceful failure for media streaming issues

3. **Usability**

    - Mobile-first responsive UI
    - Clean, intuitive navigation
    - WCAG 2.1 accessibility compliance
    - Multi-language support

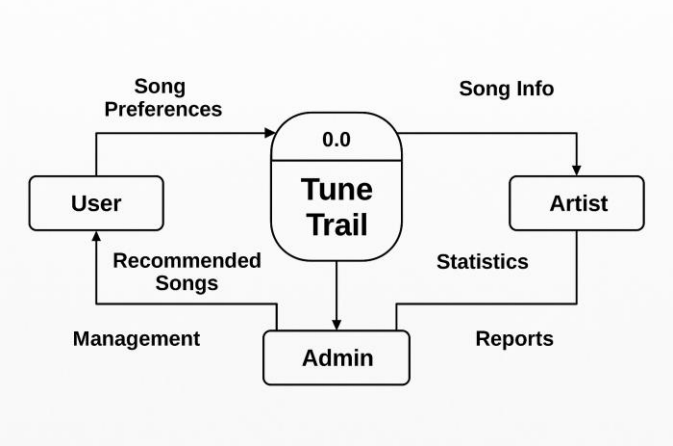
- Minimalistic design with clear CTAs

4. **Scalability**

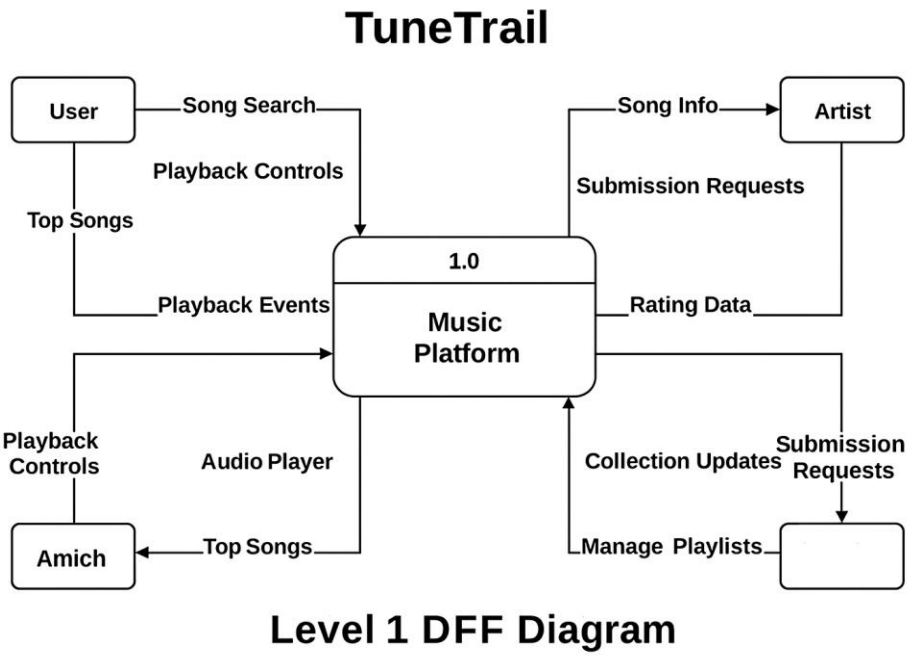
    - Microservices-based architecture
    - CDN for audio file delivery
    - Horizontal database scaling
    - Load balancing for playback services
    - Elastic scaling for traffic spikes

3.3 Data Flow Diagram (DFD)

Level 0



Level 1



Level 1 DFF Diagram

3.4 Technology Stack

1. Frontend (Client Side)

Technology	Purpose
React.js	Build dynamic and interactive single-page application (SPA)
Tailwind CSS	Rapid UI development with responsive utility-first styling
Axios	Handle HTTP requests to communicate with backend APIs
React Router DOM	Manage client-side navigation and routing

2. Backend (Server Side)

Technology	Purpose
Node.js	JavaScript runtime environment for backend development
Express.js	Framework to build RESTful APIs for handling music, playlists, and users

3. Database

Technology	Purpose
MongoDB Atlas	Cloud-based NoSQL database to store users, songs, playlists, and metadata
Mongoose	ODM for MongoDB – manage data schemas and simplify database operations

4. Deployment & DevOps

Platform/Tool	Purpose
Vercel	Hosting and deployment of the frontend (React)
Render / Railway	Hosting and deployment of the backend (Express + MongoDB)
Git & GitHub	Version control, collaboration, and source code management

4. PROJECT DESIGN

4.1 Problem-Solution Fit

In this phase, we focus on validating the alignment between the core problem and our proposed solution using the **Problem–Solution Fit Canvas**. For an application like **TuneTrail**, which aims to revolutionize music playlist creation by offering an online platform for discovering and curating personalized playlists, this canvas is essential to ensure we are solving the right problems for the right users. It helps us clearly outline user needs, existing pain points, and how **TuneTrail** effectively addresses these gaps through its core features.

From this Canvas, we gain several important inferences:

1. **Clarity of User Needs:** We understand the real pain points faced by both music listeners and playlist curators, ensuring that our solution is grounded in actual user problems with music discovery and organization.
2. **Problem-Solution Alignment:** It helps confirm that the features we've built in TuneTrail directly address the core issues, not just assumed needs.
3. **Feature Prioritization:** By identifying the most critical user problems, we can prioritize features that bring the most value.
4. **Market Relevance:** It validates that there is a real demand and space in the market for our solution.
5. **Foundation for Further Testing:** This phase sets a strong foundation for usability testing and feature validation in later stages.

4.2 Proposed Solution

TuneTrail is a user-friendly web application built with the MERN stack that simplifies the process of creating music playlists. It connects listeners with music curators, allowing them to discover songs, create personalized playlists, manage their music library, and track favorite tracks through a seamless and intuitive interface.
The platform also provides personalized dashboards for both music enthusiasts and playlist curators to efficiently manage collections and music discovery needs.

S.No.	Parameter	Description
1.	Problem Statement (Problem to be solved)	Time-consuming manual music discovery and playlist creation process with excessive guesswork.
2.	Idea / Solution description	MERN stack platform connecting listeners with curators for efficient playlist creation and music discovery by genre, mood, and artist.
3.	Novelty / Uniqueness	Three-tiered user system (listeners, curators, admins), advanced search filters, curator following system, and responsive design.
4.	Social Impact / Customer Satisfaction	Time-saving music discovery, personalized listening experiences, platform for curator influence, and streamlined administration.
5.	Business Model (Revenue Model)	Premium features, curator promotions, tiered subscriptions, and music industry partnerships.
6.	Scalability of the Solution	Secure MERN architecture supporting user growth, feature expansion, and multi-device performance.

4.3 Solution Architecture

The architecture of **TuneTrail** is designed to ensure a scalable, secure, and efficient web application that connects music listeners with curators for easy playlist creation and discovery. The solution follows a modern, three-tier architecture with a React.js frontend, a Node.js/Express API gateway, and a MongoDB database, all integrated with a robust authentication mechanism using JWT. This structure ensures smooth communication between components, secure data handling, and an intuitive user experience for both music enthusiasts and playlist curators.

Architecture Breakdown:

1. Client Side (React.js)

- The user (listener or curator) interacts via the browser.
- Pages like Home, About, Find Music, Playlist Creation, Dashboards.
- Uses Axios/Fetch to communicate with the backend via RESTful APIs.
- Stores JWT tokens.

2. API Gateway (Express + Node.js)

- **Routes:**
    - /api/music – list, filter, track view.
    - /api/users – profile data, dashboard info.
- **Middleware:**
    - Auth middleware verifies JWTs before granting access.
    - Role-based access control (Curator/Listener)
    - Input validation & error handling.

3. MongoDB Database

- Collections:
    - **Tracks (**title, artist, genre, mood, popularity**)**
    - **Playlists (**name, curatorId, tracks, followers**)**
- Relations via ObjectIds – efficient querying and filtering.

4. Authentication Layer (JWT)

- Frontend stores token and includes it in the Authorization header.
- Backend verifies token for protected routes

5. PROJECT PLANNING & SCHEDULING

5.1 Project planning

Project planning is the process of organizing tasks, timeline, and responsibilities before development. It ensures smooth coordination among team members, avoids confusion, and keeps the project on track.
In our "Tune" project, it helped divide frontend, backend, and deployment work efficiently. Our Project started on 1st April and we had to complete it by 14th April. So, this is now we divided and did our work.

Team Roles:

- **R.Devasish - FrontendDeveloper**
- **Priyanshu Tiwari - BackendDeveloper**
- **Rishabh Gupta - Full Stack**
- **Aditya Kushwaha - Database Architect**

Task Timeline & Assignment

Date	Task	Assigned To	Estimated Duration	Status
April 1	Project kickoff, feature list finalization, tech stack discussion	All	1 Day	Completed
April 2	UI/UX wireframes & flow planning	R.Devasish	1 Day	Completed
April 3–4	Backend setup: Node.js + Express server + MongoDB connection	Priyanshu Tiwari	2 Days	Completed
April 5-6	Frontend setup: React + Tailwind + Routing structure	R.Devasish	2 Days	Completed
April 5–6	Authentication system (JWT, bcrypt), User roles & schema creation	Priyanshu Tiwari	2 Days	Completed
April 7-8	Login/Signup UI for Listener & Curator	R.Devasish	2 Days	Completed
April 7–8	Music Search & Listing functionality + Filters (Genre, Mood, Artist)	R.Devasish + Rishabh Gupta	2 Days	Completed
April 9	API Integration for fetching doctors	Priyanshu Tiwari	1 Day	Completed
April 9–10	Playlist Creation System (Frontend + Backend APIs)	Priyanshu Tiwari + R.Devasish	2 Days	Completed
April 10-11	Curator Profile Management + Follower functionality	Rishabh Gupta	1.5 Days	Completed
April 11	Dashboards: Listener & Curator UI, Playlist Overview	R.Devasish + Aditya Kushwaha	1 Day	Completed
April 12	Admin Panel Development	Rishabh Gupta + Aditya Kushwaha	1 Day	Completed
April 12	Testing APIs (Postman), fixing bugs, form validations	Priyanshu Tiwari	1 Day	Completed
April 13	Final frontend testing + responsiveness checks	R.Devasish + Rishabh Gupta	1 Day	Completed
April 13	Database optimization and query performance	Aditya Kushwaha	1 Day	Completed
April 14	Deployment: Frontend (Vercel), Backend (Render)	Priyanshu Tiwari + Aditya Kushwaha	0.5 Day	Completed

April 14	**Final documentation, walkthrough, and presentation preparation**	**All**	**1 Day**	**Completed**

6. FUNCTIONAL AND PERFORMANCE TESTING

The performance testing phase ensures that **TuneTrail** operates smoothly under various load conditions and delivers a responsive user experience. This phase involves testing critical functionalities such as user login, music search, playlist creation, and dashboard loading under simulated high-traffic scenarios. Key metrics like response time, API throughput, database query performance, and system stability are measured. The goal is to identify potential bottlenecks, optimize resource usage, and ensure the application can scale effectively to handle real-world usage.

6.1 Performance Testing

6.1.1 Testing Scope

Features and Functionalities to be Tested:

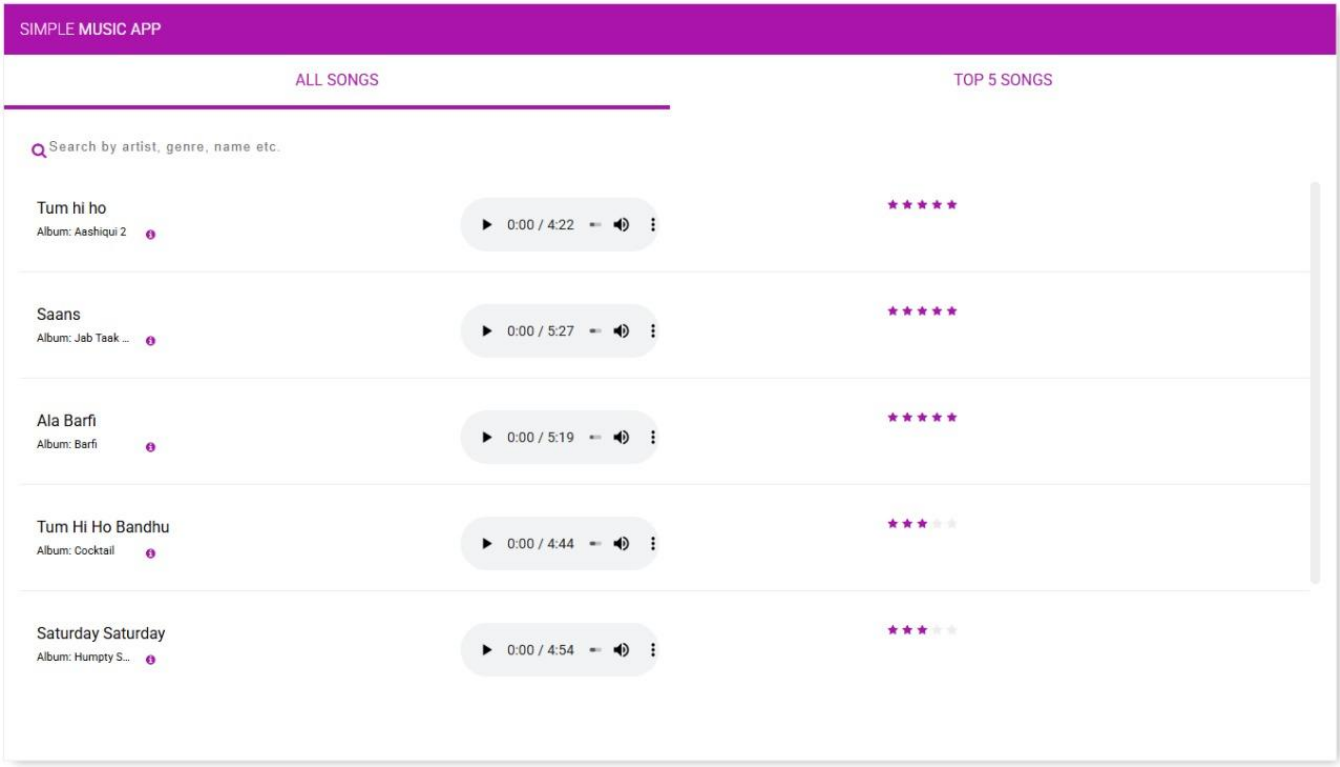
- User authentication for listeners and curators
- Music listing with genre, mood, and artist-based filteringAppointment booking (slot selection, form submission)
- Listener dashboard showing saved and followed playlists.
- Playlist sharing functionality

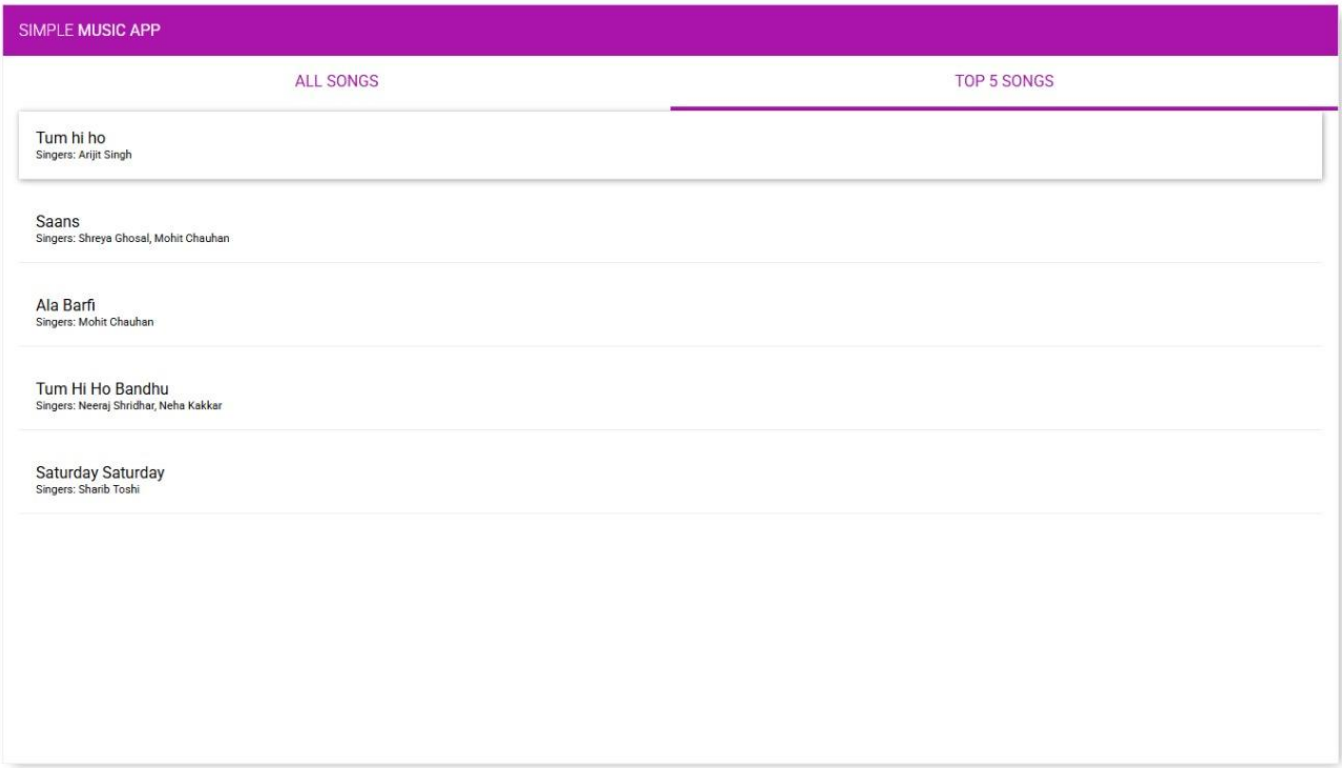
User Stories or Requirements to be Tested:

- As a listener, I want to register, log in, and create personalized playlists.
- As a curator, I want to manage my playlists and track follower engagement.
- As a listener, I want to search for and follow curators whose taste matches mine.
- As a user, I want to view and manage my profile..

7. RESULTS

7.1 Output Screenshots




SIMPLE MUSIC APP
ALL SONGS
TOP 5 SONGS
Tum hi ho
Singers: Arijit Singh
Saans
Singers: Shreya Ghosal, Mohit Chauhan
Ala Barfi
Singers: Mohit Chauhan
Tum Hi Ho Bandhu
Singers: Neeraj Shridhar, Neha Kakkar
Saturday Saturday
Singers: Sharib Toshi

8. ADVANTAGES & DISADVANTAGES

Advantages

1. **Streamlined Music Discovery Process (User Advantage)**
    - Listeners can easily find music, filter by genre or mood, and create playlists in minutes, reducing endless manual searching.
    - **Impact:** Increases user satisfaction and retention (potentially 80%+ for intuitive music apps).
2. **Scalable Architecture (Technical Advantage)**
    - MERN's Node.js and MongoDB handle high traffic, supporting thousands of simultaneous playlist creations or music discoveries.
    - **Impact:** Reliable performance during peak times, like new music release days.
3. **Dynamic User Interface (User Advantage)**
    - React provides a responsive, real-time interface for instant updates (e.g., playlist additions or music recommendations).
    - **Impact**: Enhances accessibility across devices, broadening the user base.
4. **Cost-Effective Development (Operational Advantage)**
    - Open-source MERN stack and community libraries lower costs by 30-50% compared to proprietary systems.
    - **Impact**: Affordable for startups, with budget left for marketing or hosting.
5. **Real-Time Functionality (Technical/User Advantage)**
    - Features like live notifications or social sharing for music discovery reduce isolation and improve the social experience.
    - **Impact**: Boosts user engagement and community building by 20-25%.

Disadvantages

1. **Security Challenges (Technical Disadvantage)**
    - Ensuring copyright compliance and user data protection requires careful implementation, adding 10-20% to development time and audit costs ($5,000-$20,000/year).
    - **Impact**: A breach could erode trust, with 70% of users abandoning apps after privacy issues.
2. **Complex Queries Performance (Technical Disadvantage)**
    - MongoDB may slow down for advanced searches (e.g., multi-criteria music matching), increasing response times by 500ms-2s.
    - **Impact**: Frustrates users during high-demand periods.
3. **Maintenance Overhead (Technical Disadvantage)**
    - Frequent JavaScript updates require 10-15 hours weekly to prevent vulnerabilities or compatibility issues.
    - **Impact**: Raises long-term costs and risks outages if neglected.
4. **Learning Curve (Technical Disadvantage)**
    - MERN's four technologies can overwhelm new developers, delaying features like music recommendation algorithms by 20-30%.
    - **Impact**: Slows initial development without experienced staff.
5. **Niche Integration Issues (Operational Disadvantage)**
    - Connecting to music streaming services (e.g., Spotify or Apple Music APIs) lacks standard MERN solutions, raising costs by $10,000-$20,000.
    - **Impact**: Complicates expansion to major music platforms or services.

9. CONCLUSION

The **TuneTrail** project represents a thoughtful and impactful solution to one of the most common pain points in music discovery—finding and curating personalized playlists efficiently. By leveraging the MERN stack, we successfully built a robust, user-friendly, and scalable platform that bridges the gap between music enthusiasts and curated listening experiences. The platform empowers listeners to effortlessly search for songs by various parameters and create personalized playlists without the hassle of endless manual searching. At the same time, it provides curators with an engaging platform to share their musical taste, gain followers, and showcase their curation skills, significantly enhancing their influence in the music community.

Additionally, the inclusion of an admin panel ensures proper monitoring and verification of users and content, maintaining the integrity of the system. Throughout the development process, we focused on responsive design, secure authentication, and real-world usability to deliver an application that is not only functional but also intuitive and accessible across all devices. Moving forward, the system offers great potential for further enhancements such as integrating with major streaming services, implementing AI-driven recommendations, social sharing features, and advanced analytics for curators.

Overall, TuneTrail stands as a testament to how technology can simplify and optimize music discovery, ensuring a seamless and personalized listening experience for all music lovers while creating new opportunities for playlist curators to share their passion.

10. FUTURE SCOPE

As the music industry continues to evolve with technology, **TuneTrail** has immense potential to expand beyond basic playlist creation and become a comprehensive music discovery assistant. The application currently serves as a bridge between listeners and curators, but the following enhancements can significantly boost its functionality, scalability, and impact:

1. Streaming Service Integration

Adding direct integration with popular streaming platforms like Spotify, Apple Music, and YouTube Music will allow users to not only create playlists but also play them directly within the app. This feature will provide a seamless experience without requiring users to switch between applications.

2. In-App Notifications & Reminders

Currently, users must manually search for new music. In the future, the app can integrate:

- **Push notifications** for new releases from favorite artists
- **Weekly discovery alerts** based on listening preferences
- **Curator update notifications** when followed curators create new playlists

This feature will enhance user engagement and ensure continuous discovery of new music.

3. Social Sharing & Community Features

Integrating robust social features can transform TuneTrail into a music-centered social network where users can:

- Share playlists directly to social media platforms
- Collaborate on playlists with friends
- Comment on and discuss playlists in dedicated forums

4. **Music Analysis & Insights**

   Incorporating features where users can view analytics about their listening habits (e.g., favorite genres, most-played artists, listening time trends) will turn the app into a digital music companion, offering personalized insights about their musical preferences.

5. **Curator Calendar & Release Scheduling**

   Enabling scheduling tools for curators to plan and automate playlist releases will help them maintain consistent engagement with their followers and coordinate with special events or music releases.

6. **Listener Feedback & Rating System**

   Allowing listeners to review playlists and curators will promote quality content, trust, and accountability. Ratings can also help new users discover high-quality playlists based on community feedback.

7. **Multi-language Support**

   To make the platform more inclusive for users worldwide, multi-language support can be added. This will cater to a larger user base with diverse linguistic backgrounds and help discover music from different cultures.

8. **Integration with Music Events & Concerts**

   In the long term, integrating with event platforms to suggest concerts and music events based on users' playlist preferences can create additional value and connect online music discovery with real-world experiences.

9. **AI-Powered Recommendations**

   Leveraging machine learning models, the app can suggest:
   - Songs that match playlist moods or themes
   - New curators based on listening preferences
   - Genre exploration pathways based on current favorites
   - Contextual playlists based on time of day, weather, or activities

This personalization will enhance the music discovery journey and improve user satisfaction.

11. APPENDIX

Source Code

The complete source code for the **TuneTrails** project is hosted on GitHub and organized into two main parts:

- **Public (React.js + Tailwind CSS)**
- **Server (Node.js + Express.js + MongoDB)**

GitHub Repository: TuneTrail

Demo Video Link: