Project Title	Music Streaming Platform
Technologies	MERN
Domain	Entertainment
Project Level	Hard

Table

Table of Contents

1. Problem Statement:

Design a Music Streaming Application using the MERN stack

1.1. Overview of MERN stack and Music Streaming Application

Full-stack web applications can be launched more quickly and easily with the JavaScript stack MERN. The MERN Stack is comprised of the four technologies MongoDB, Express, React, and Node.js. The goal is to simplify and streamline the development process.

Each of these four potent technologies provides a complete environment for developers to operate in and makes a substantial contribution to the creation of web apps.

A music streaming is an application in which a user can listen to the music anytime. The music will be updated by the admin in the application. The user will be able to listen that music directly from the web app.

Note: **Spotify** is one of the most well-known software you may use as a reference to get a sense of the music streaming service.

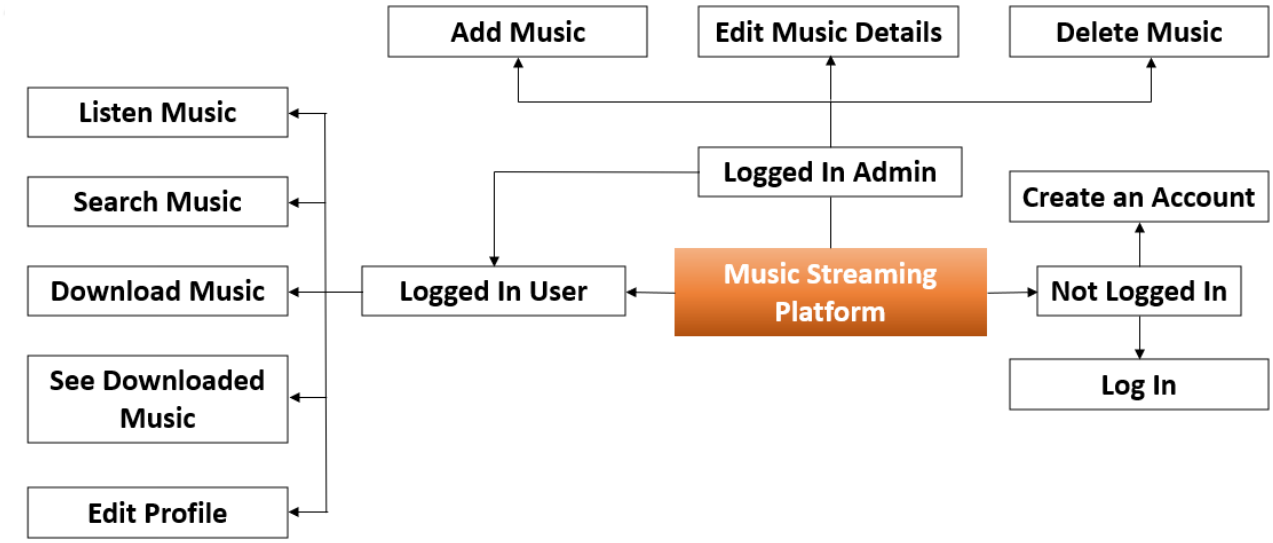
1.1.1 The Music Streaming App should possess these features

These are the list of all the expected features which should be available in your streaming website-

a. A landing page where the user can listen to music will be reached by the user.
b. A list of the music that is available for the user to listen to should be provided.
c. When a user is not signed into his account, he should not be able to listen to or explore the music.
d. A user can listen to, search for, and download music after logging into his account.

e. Without accessing the system's file explorer, a user ought to be able to see all of the music they've downloaded and delete it directly from the application.
f. The provided search box can be used to find the song.
g. A user should be able to filter his music based on the singer's name, the genre, the language, and many other factors.
h. The user should be able to see the music that is currently playing and access information about it, such as the singer's name, the opportunity to view the lyrics, the song's category, etc., on a card.
i. A user should have complete control over his player, allowing him to play the next or previous song, adjust the volume, fast-forward a song while listening, and other functions.

Low Level user flow



1.2. Project Objective

• **Music Streaming App:** Everyone enjoys listening to music these days while working out, driving, relaxing, etc., as seen by the expansion of music streaming services like Spotify and others.

• **User-friendliness:** To ensure that everyone can use the platform without difficulty, the user interface needs to be basic and easy to use.

1.3. Scope of The Project

1. Due to the increased demand for and use of music streaming services, these platforms are growing rapidly.

2. There is a sizable audience that will use the platform to listen to the music.

1.4. Functional and Non-Functional Requirements: -

1.4.1. Functional Requirements for User

1. **User Registration:** Users must be able to register for the application using their email, username, and password. Users must be able to log in right away if they already have an account or register themselves when they first launch the software.

2. **Browse and Listen Song:** The user ought to be able to search the music and play it as needed.

3. **Search and Sort:** The person who is logged in can sort the song list or search for a particular song by name or category.

5. **Dashboard:** The user should be able to view his profile information and downloads.

1.4.2. Functional Requirements for Admin

1. **Admin Registration:** The administrator needs to be able to access his account using his credentials.

2. **Dashboard:** The administrator's dashboard allows him to view every single registered user, song, and their details.

3. **CRUD on User:** From the dashboard, the administrator can add, modify, delete, or update a user.

4. **CRUD on Music:** From the dashboard, the administrator can add, modify, delete, or update any song at any moment.

1.4.3. Non-Functional Requirements

1. **Privacy:** Only users who are logged in can listen to and browse songs on the platform. Music and registered users should only be able to use CRUD operations by the admin.

2. **Robustness:** A backup of the user's settings should be kept on a remote server in case the user's system breaks, protecting his data.

3. **Performance:** The software must load the music quickly and be lightweight.

1.5. Use Case Table

Authentication System	Register, Login, Logout	User and Admin
Search Form	Search Music	User and Admin
User Dashboard	Display User’s details and downloaded music	User
Admin Dashboard	Display all users and music	Admin

Table 1. Use Case

2. Project Evaluation Metrics:

2.1. Code:

- You are supposed to write code in a modular fashion
- Safe: It can be used without causing harm.
- Testable: It can be tested at the code level.
- Maintainable: It can be maintained, even as your codebase grows.
- Portable: It works the same in every environment (operating system).
- You have to maintain your code on GitHub.
- You have to keep your GitHub repo public so that anyone can check your code.
- Proper readme file you have to maintain for any project development.
- You should include the basic workflow and execution of the entire project in the readme file on GitHub.
- Follow the coding standards.

2.2. Database:

MongoDB is a source-available cross-platform document-oriented database program. Classified as a NoSQL database program, MongoDB uses JSON-like documents with optional schemas.

2.3. API Details or User Interface:

You have to expose your complete solution as an API or try to create a user interface for your model testing. Anything will be fine for us.

2.4. Deployment:

Implementation of reverse proxy, load balancing, and security group is mandatory for deployed applications.

2.5. Solutions Design:

You have to submit complete solution design strategies in High-level Document (HLD), Low-level Document (LLD), and Wireframe documents**.**

2.6. System Architecture:

You have to submit a system architecture design in your wireframe document and architecture document.

2.7. Optimization of solutions:

Try to optimize your solution on code level, architecture level, and mention all of these things in your final submission.

Mention your test cases for your project.

3. Submission requirements:

3.1. High-level Document:

You have to create a high-level document design for your project. You can reference the HLD form below the link.

Sample link: HLD Document Link

3.2. Low-level document:

You have to create a Low-level document design for your project; you can refer to the LLD from the link below.

Sample link: LLD Document Link

3.3. Architecture:

You have to create an Architecture document design for your project; you can refer to the Architecture from the link below.

Sample link: Architecture sample link

3.4. Wireframe:

You have to create a Wireframe document design for your project; refer to the Wireframe from the link below.

Demo link: Wireframe Document Link

3.5. Project code:

You have to submit your code to the GitHub repo in your dashboard when the final submission of your project.

Demo link: Project code sample link

3.6. Detail project report:

You have to create a detailed project report and submit that document as per the given sample.

Demo link: DPR sample link

3.7. Project demo video:

You have to record a project demo video for at least 5 Minutes and submit that link as per the given demo.

3.8. The project LinkedIn a post:

You have to post your project details on LinkedIn and submit that post link in your dashboard in your respective field.