* Project title and description
* An overview of the challenges of this project (e.g. why is this project a good fit for Enterprise Software Development?)
* An overview of some non-functional requirements that are important to your project
* Some initial ideas that you already have on how to achieve this project (architecture, technology selection, etc.)

HeardIt

An application where you can upload music tracks. Users can create accounts, upload/delete tracks, search for tracks. Users can play the tracks

Account functionalities:

Create account

Username, password, profile picture, *? login using Google ?*

Login/ Logout

Upload/Delete music tracks own microservice?

General user

Search for tracks by name

Play tracks

**Project Title: Dynamic Music Discovery Platform**

**Objective:** To develop a scalable music discovery platform inspired by the dynamic content curation of Reddit and the diverse user-generated content of 4chan. The project aims to provide users with a space to explore, share, and discuss a wide range of music genres, emphasizing scalability.

**Learning Outcomes:**

1. **Scalable Architecture:** Design and implement a backend architecture that can handle a growing number of users, music uploads, and discussions without compromising performance.
2. **User-Generated Playlists:** Develop features that allow users to create and share playlists, encouraging a collaborative and diverse approach to music discovery.
3. **Community Engagement:** Implement interactive features such as upvoting, commenting, and sharing to foster community engagement and enhance the music discovery experience.
4. **Scalability Testing:** Conduct tests to evaluate the platform's performance and scalability under increasing user-generated content and concurrent user engagement.

**Non-Functional Requirements:**

1. **Scalability:** The platform should be capable of handling at least 1 million active users and a vast library of music tracks, ensuring a smooth and responsive experience for all users.
2. **User-Generated Playlists:** Users should be able to create, share, and collaborate on playlists with ease, promoting a diverse and user-driven music curation experience.
3. **Community Engagement:** Design features that encourage user interaction, such as comments, upvotes, and sharing, to create a vibrant music discovery community.
4. **Performance:** Music tracks and discussions should load within 2 seconds under regular usage conditions to maintain a seamless user experience.

**Challenges:**

1. **Scalable Media Storage:** Implement an efficient media storage solution to handle a large volume of music uploads while maintaining fast retrieval times.
2. **Playlist Collaboration:** Enable smooth and real-time collaboration on playlists, allowing multiple users to contribute and curate seamlessly.

**Quality Assurance:**

1. Conduct load testing to evaluate the platform's scalability and performance under various user-generated content scenarios.
2. Implement user feedback mechanisms to understand user preferences and continuously enhance the platform's music discovery features.
3. Monitor community engagement metrics to assess the success of interactive features and improve them based on user behavior.

**Conclusion:** The Dynamic Music Discovery Platform aims to provide users with a scalable and engaging space for music exploration. Success will be measured by the platform's scalability, the diversity and quality of user-generated playlists, and the effectiveness of community engagement features in fostering a collaborative music discovery experience.

User

**You**

ok but I dont have the music now. What external APIs can I use?

ChatGPT

**ChatGPT**

Certainly! There are several external APIs that you can leverage for your music discovery platform. Here are some popular ones:

1. **Spotify API:**
   * **Documentation:** Spotify for Developers
   * **Capabilities:** Access Spotify's extensive music catalog, retrieve track details, get recommendations, and create playlists.
2. **Apple Music API:**
   * **Documentation:** Apple Music API
   * **Capabilities:** Access the Apple Music catalog, fetch details about songs, albums, and playlists, and integrate Apple Music features.
3. **Deezer API:**
   * **Documentation:** Deezer API
   * **Capabilities:** Retrieve information about artists, albums, and tracks, and enable playlist management functionalities.
4. **Genius API:**
   * **Documentation:** Genius API
   * **Capabilities:** Retrieve lyrics and metadata for songs, making it a valuable addition for a music discovery platform.
5. **Last.fm API:**
   * **Documentation:** Last.fm API
   * **Capabilities:** Access data related to artists, albums, and tracks, and integrate features like scrobbling and user recommendations.
6. **Musixmatch API:**
   * **Documentation:** Musixmatch Developer Portal
   * **Capabilities:** Retrieve lyrics, track details, and get information about artists, making it useful for enhancing the content on your platform.
7. **SoundCloud API:**
   * **Documentation:** SoundCloud API
   * **Capabilities:** Access SoundCloud's vast collection of user-uploaded tracks, enabling a diverse range of music on your platform.
8. **YouTube Data API:**
   * **Documentation:** YouTube Data API
   * **Capabilities:** Fetch music-related content from YouTube, including video details, playlists, and channels.

Ensure that you review the terms of use and licensing agreements for each API to comply with their policies and restrictions. Additionally, consider the specific features and data each API provides to align with the goals of your music discovery platform.

Microservices IDEAS

1. **User Service:**
   * Manages user accounts, authentication, and user profiles.
   * Responsible for handling user-related operations, such as user registration, login, and profile updates.
2. **Media Management Service:**
   * Handles the storage and retrieval of media files (music tracks, album covers).
   * Integrates with a cloud-based storage system for efficient media handling.
3. **Content Aggregation Service:**
   * Gathers and aggregates content from various external APIs (e.g., Spotify, Apple Music) for a diverse music catalog.
   * Handles requests related to searching, filtering, and retrieving music content.
4. **Playlist Service:**
   * Manages user-generated playlists, including creation, modification, and deletion.
   * Enables collaborative playlist features, allowing multiple users to contribute to and edit playlists.
5. **Recommendation Service:**
   * Utilizes recommendation algorithms to suggest music tracks, albums, or playlists based on user preferences and behavior.
   * Collaborates with the Content Aggregation Service to fetch relevant content.
6. **Interaction Service:**
   * Handles user interactions within the platform, such as upvotes, comments, and sharing.
   * Facilitates real-time updates and notifications for user interactions.
7. **Community Service:**
   * Manages community-related features, including user following, follower feeds, and community engagement metrics.
   * Integrates with the Interaction Service for community-driven features.
8. **Moderation Service:**
   * Implements moderation tools to ensure content quality, user safety, and adherence to community guidelines.
   * Collaborates with the Content Aggregation Service to flag and manage inappropriate content.
9. **Authentication and Authorization Service:**
   * Centralizes authentication and authorization processes for all microservices.
   * Ensures secure access to various functionalities based on user roles and permissions.
10. **Analytics Service:**
    * Collects and analyzes platform usage data, user behavior, and engagement metrics.
    * Provides insights for continuous improvement and feature enhancements.
11. **Gateway Service:**
    * Serves as the entry point for external requests, handling API gateway functionalities.
    * Routes requests to the appropriate microservices and manages authentication.