1.) General Project Questions

- What is the primary goal of the project?

The primary goal of your project is to create a user-friendly music rating platform where users can discover, rate, and share their favorite songs and albums. Whether they're music enthusiasts, casual listeners, or aspiring critics, your app aims to enhance their music experience.

- Who is the target audience?

Your target audience should be members of the 16 – 35 years old demographics.

- What platforms will the application be available on (e.g., web, mobile)?

The application should be accessible via the web and via mobile app. Consider creating an HTML-based application that would be accessible by both interfaces and will render appropriately.

- How will users authenticate or sign up for the application?

Users can sign up using various methods:

- ***Email/Password**: Traditional sign-up with email and password.*
- ***Social Logins**: Integration with platforms like Google, Facebook, or Apple.*

- What integrations are necessary (e.g., Spotify Web API)?

Consider the following integrations:

- ***Spotify Web API**: Consider integrating with Spotify to access their vast music catalog, user playlists, and recommendations.*
- ***Last.fm API**: For artist information, album details, and user scrobbles.*
- ***Lyrics APIs**: To display lyrics alongside songs.*
- ***Social Sharing APIs**: Enable users to share their favorite tracks on social media.*

- What are the key features that differentiate this application from other music list applications?

The following features should be implemented:

- ***User-Generated Playlists**: Allow users to create and share playlists.*
- ***Rating System**: Users can rate songs (e.g., stars, thumbs up/down).*
- ***Comments and Reviews**: Let users write reviews and comment on songs.*
- ***Real-Time Lyrics**: Display lyrics synchronized with the music.*
- ***Collaborative Playlists**: Users can collaborate on playlists with friends.*
- ***Artist Profiles**: Detailed info about artists, discography, and related artists.*
- ***Notifications**: Notify users about new releases, playlist updates, or recommendations.*

- How will user data be stored and managed?

Use a secure backend (server-side) to manage user data:

- ***Database**: Store user profiles, ratings, playlists, and song metadata.*
- ***Authentication Tokens**: Securely manage user sessions.*

Follow best practices for data privacy and compliance (GDPR, CCPA, etc.).

2.) Functional Requirements

- How will users add songs to their list?

Users can add songs to their lists in several ways:

- ***Search and Add**: A search bar where users can look up songs by title, artist, or album. Once they find a song, they can add it to their personal list.*
- ***Discover and Add**: When users discover new songs (e.g., through recommendations or browsing), they can add them directly.*
- ***Import from Existing Playlists**: If users connect their Spotify or Apple Music accounts, they can import existing playlists.*

- What features will be available for rating songs?

Users can rate songs using various methods:

- ***Star Ratings**: Classic 5-star system.*
- ***Thumbs Up/Down**: Simple and quick feedback.*
- ***Emoji Reactions**: Let users express their feelings (e.g., heart, fire, smiley).*
- ***Written Reviews**: Allow users to write detailed reviews or comments.*

- How can users manage and organize their song lists?

Users can organize their song lists by:

- ***Playlists**: Users can create custom playlists (more on this below).*
- ***Favorites**: A default list for songs they love.*
- ***Recently Played**: Automatically updated with recently listened songs.*
- ***Tags or Labels**: Users can tag songs (e.g., "Chill," "Workout," "Road Trip").*

- How will users interact with friends within the application?

Social features enhance the experience:

- ***Follow Friends**: Users can follow their friends' profiles.*
- ***Activity Feed**: See what friends are listening to, their ratings, and playlists.*
- ***Collaborative Playlists**: Create shared playlists with friends.*
- ***Recommendations from Friends**: Get song recommendations based on friends' preferences.*

- What features will be available for song discovery?

Help users discover new music:

- ***Personalized Recommendations**: Based on their listening history and ratings.*
- ***Genre-Based Playlists**: Curated playlists for specific genres.*
- ***Trending Songs**: What's popular right now.*
- ***Artist Radio**: Continuous play based on an artist or song.*

- ***Discovery Mode**: Shuffle songs from genres they haven't explored much.*

- Will users be able to create custom playlists?

Yes, users can create custom playlists:

- ***New Playlist**: Name it, add songs, and set privacy (public, private, or collaborative).*
- ***Auto-Generated Playlists**:*
    - ***Top Rated**: Auto-populated with their highest-rated songs.*
    - ***Recently Added**: Songs added in the last X days.*
    - ***Mood Playlists**: E.g., "Happy Vibes," "Late Night Chill."*

- How will search functionality be implemented?

Implement a robust search feature:

- ***Autocomplete Suggestions**: As users type, show relevant songs, artists, and albums.*
- ***Advanced Filters**: Filter by genre, release year, artist, etc.*
- ***Voice Search**: For hands-free interaction.*

3.) Non-functional Requirements

- What are the performance expectations for the application?

Ensuring quick response times is crucial for user satisfaction. Studies show that even minor delays (a few seconds) lead most users to navigate away. Aim for:

- *Under 2 seconds: As the new normal for responses.*
- *Every 1-second improvement: Retailers see 2-3% higher conversions.*

Your app should handle peak traffic without performance degradation.

- What are the security requirements, especially regarding user data?

***Authentication and Authorization**:*

- *Confirm users' identities before granting access.*
- *Use multi-factor authentication for added security.*

***Input Validation and Sanitization**:*

- *Validate and sanitize user inputs to prevent attacks like SQL injection or cross-site scripting (XSS).*

***Data Encryption**:*

- *Encrypt sensitive data (user profiles, ratings) during transmission.*

***Privacy Compliance**:*

- *Adhere to regulations (GDPR, CCPA) regarding user data protection.*

- What should the user interface look like?

Consider a clean, modern design:

- ***Home Screen**: Sections for personalized playlists, new releases, and popular songs.*
- ***Bold Typography**: Enhance readability.*
- ***Vibrant Colors**: Create visual appeal.*
- ***Responsive Design**: Adapt to various devices (web, mobile).*

- How should the application handle scalability?

- ***Vertical Scaling (Scaling Up)**:*
  - *Upgrade existing resources (CPU, memory, storage) to meet increased demand.*
- ***Horizontal Scaling (Scaling Out)**:*
  - *Add more instances (e.g., virtual machines) to distribute load.*
- ***Cloud Services**:*
  - *Leverage cloud resources for elasticity (provisioning and releasing as needed).*

- What is the expected response time for various features?

See 3.1.

- What are the availability and reliability expectations?

Aim for high availability (minimal downtime) and reliability (consistent performance). Implement redundancy (backup servers, failover mechanisms).

- What are the accessibility requirements for users with disabilities?

Ensure your app is usable by everyone:

- ***WCAG Guidelines**: Follow Web Content Accessibility Guidelines.*
- ***Screen Readers**: Support screen readers for visually impaired users.*
- ***Keyboard Navigation**: Enable keyboard-based interactions.*
- ***Contrast and Font Sizes**: Optimize for readability.*

4.) Constraints and Assumptions

- What constraints are there on the technologies that can be used?

When building a music rating app, you'll want to consider the following constraints:

- ***Platform Compatibility**: Each platform (Web, mobile app) has its own development requirements and limitations.*
- ***Third-Party Integrations**: Consider the APIs and services you'll use. For instance, if you're integrating with Spotify (or any other music service), you'll need to adhere to their guidelines and limitations.*
- ***Scalability**: Choose technologies that allow your app to scale efficiently as your user base grows.*

- ***Security**: Ensure your chosen stack supports robust security practices, especially when handling user data.*

- What assumptions are made about the users' devices or environments?

Assume that users will have varying devices (smartphones, tablets, desktops) and operating systems (iOS, Android, Windows, macOS).

Consider factors like screen sizes, network connectivity (3G/4G/5G, Wi-Fi), and device capabilities (e.g., GPS, camera).

- Are there any limitations imposed by the Spotify Web API?

The Spotify Web API has rate limits to maintain reliability and responsible usage:

- ***Rate Limit**: The number of calls your app can make within a rolling 30-second window determines the rate limit. Exceeding this limit results in 429 error responses.*
- ***Custom Limits**: Some endpoints (e.g., playlist image uploads) may have custom rate limits.*
- ***Auto Scheduling**: Implement backoff-retry strategies to handle rate limits gracefully.*
- ***Batch APIs**: Use batch APIs (e.g., Get Multiple Albums) to reduce the number of requests.*

- What is the project timeline?

The timeline is fixed until the end of the semester. No flexibility here.

- What is the budget for the project?

There is no budget for the project. You have your teams and the individual skills as resources.

5.) Risks and Dependencies

- What are the potential risks related to the use of the Spotify Web API?

Spotify Web API Risks:

- ***Inaccurate Data**: Some third-party websites that rely on Spotify API data (e.g., stats tracking sites) may provide inconsistent or inaccurate information. Be aware of potential discrepancies.*
- ***API Outages**: While rare, outages can impact apps relying on the Spotify Web API. Monitoring the Web API status page helps stay informed.*
- ***Rate Limits**: The API has rate limits; exceeding them can lead to temporary bans. Ensure your app handles rate limits gracefully.*

- What dependencies does the project have on external systems or services?

Your music rating app likely depends on external systems or services:

- ***Spotify API**: The core dependency for music data.*
- ***Authentication Providers**: OAuth providers for user sign-in.*
- ***Cloud Services**: If you use cloud storage or databases.*
- ***Analytics Tools**: For monitoring app usage.*

- What are the risks associated with data security and privacy?

Data Security and Privacy Risks:

- ***User Data Exposure**: Ensure proper encryption and secure transmission of user data (e.g., profiles, ratings).*
- ***Privacy Violations**: Mishandling user data can lead to legal and reputational issues.*
- ***Third-Party Integrations**: Verify that external services (like analytics tools) comply with privacy regulations.*

- What are the potential risks related to project timelines or team resources?

Timelines will not change in this project. You have until the last week of classes (week before finals week). Regarding resources, it is possible that your team might face issues with team member's performance. You'll need to deal with these issues as they come up.

- What are the risks of feature creep or scope changes?

Because you are not dealing with an external client, this will not be an issue. However, you'll need to be careful with what you commit to doing.

- How will changes in Spotify's API policies or availability affect the project?

Keep an eye on Spotify's API documentation and announcements. Changes in policies or availability could affect your app's functionality.

Plan for adaptability: If Spotify alters its API, be prepared to adjust your app accordingly.