Project Management Practices:

1. **Modularity and Organization:**
   - The code is organized into functions and endpoints, promoting modularity.
   - Each endpoint seems to handle a specific feature or functionality, contributing to a clear project structure.
2. **Error Handling:**
   - Proper error handling is implemented using try-except blocks in critical sections of the code.
   - The application returns appropriate HTTP status codes and error messages to the client.
3. **Documentation:**
   - Inline comments are used to explain complex sections of code or provide context.
   - Descriptive function and variable names contribute to code readability.
4. **Session Management:**
   - The use of the `session` object for storing the current user's information indicates a stateful approach, which is essential for maintaining user sessions.
5. **Database Abstraction:**
   - The code abstracts database operations, making use of functions to connect to the MongoDB database, interact with collections, and perform queries.

Codebase Management Practices:

1. **Version Control:**
   - using a version control system (e.g., Git) is crucial for codebase management. Commits and branches help track changes and collaborate effectively.
2. **Code Readability:**
   - The code follows PEP 8 conventions for Python, contributing to code consistency and readability.
   - Proper indentation and spacing enhance code readability.
3. **Separation of Concerns:**
   - Functionality is separated into different routes/endpoints, and the logic is distributed across multiple functions, promoting the separation of concerns.
4. **Environment Configuration:**

- The application have a configuration for running on a specific host and port, which is essential for deployment flexibility.

5. **Security Considerations:**
    - While the codebase does handle user authentication,we made sure to follow best practices for securing sensitive information, such as secret keys and user credentials.
6. **Testing:**
    - Unit test here.
7. **Dependency Management:**
    - Flask==2.0.1
    - certifi==2021.5.30
    - requests==2.26.0
    - pymongo==3.12.0
    - bson==0.5.10

REQUIREMENTS SPESIFICATONS:

1. **User Stories in Code Comments:**
    - The code includes comments that describe the purpose and functionality of certain blocks of code. This is a good practice to provide context and explanations for future developers.
2. **Descriptive Endpoint Names:**
    - Endpoint names such as `/register`, `/login`, `/recommend_song` indicate the functionality they serve. Descriptive endpoint names help in understanding the purpose of each API.
3. **Consistent Variable Naming:**
    - Variable names are relatively consistent and descriptive. For example, `username`, `user_document`, `liked_songs`. This contributes to code readability and understanding.
4. **Error Handling:**
    - The code includes error handling for potential exceptions, providing informative error messages. This is crucial for debugging and understanding the cause of issues.
5. **Input Validation:**
    - There are checks for missing parameters in query parameters and form data. For example, in `/get_higher_rated_genre` and `/get_users_liked_songs`, the code checks if the `username` parameter is present.
6. **Usage of Session for User Authentication:**
    - Session usage (`session.get('username')`) indicates an attempt to manage user authentication and maintain user state across requests.
7. **RESTful Endpoint Design:**
    - The endpoints are designed following RESTful principles, with HTTP methods (`GET`, `POST`) corresponding to the intended actions.
8. **Separation of Concerns:**

- The code separate concerns by having distinct endpoints for different functionalities, such as authentication, recommendation, search, and user-related activities.

9. **Structured JSON Responses:**
    - The API responses are structured in JSON format, making it easy for clients to parse and interpret the data.
10. **Documentation in Comments:**
    - There are comments that explain the purpose of specific code blocks, which can serve as a form of documentation for developers who review or maintain the code.