

AI-Powered Music Recommendation System

**Internship - Low Level Design**

Build an AI-powered recommendation system for a music streaming platform

### MASTERS OF COMPUTER APPLICATION

## AIML

**SUBMITTED BY:**

|  |  |
| --- | --- |
| **NAME** | **SAP ID** |
| Akash Panwar | 500106837 |
| Omji Shukla | 500100963 |
| Tanishka Chandola | 500104885 |
| Abhishek Kumar | 500106851 |
| Deepesh Singh | 500106971 |

**GUIDED BY: Dr. Kavitha**

**Table of Contents**

|  |  |  |
| --- | --- | --- |
| **S.no** | **Topics** | **Page No.** |
| **1** | **Introduction** | 3 |
|  | Scope of the document |  |
|  | Intended audience |  |
|  | System overview |  |
| **2** | **Low Level System Design** | 4 |
|  | Sequence Diagram |  |
|  | Navigation Flow |  |
|  | Defaults and Attributes |  |
|  | Components Design Implementation |  |
|  | Configurations/Settings |  |
|  | Interfaces to other components |  |
| **3** | **Key notes** | 7 |
| **4** | **Reference** | 8 |

**Introduction**

**Scope of the document**

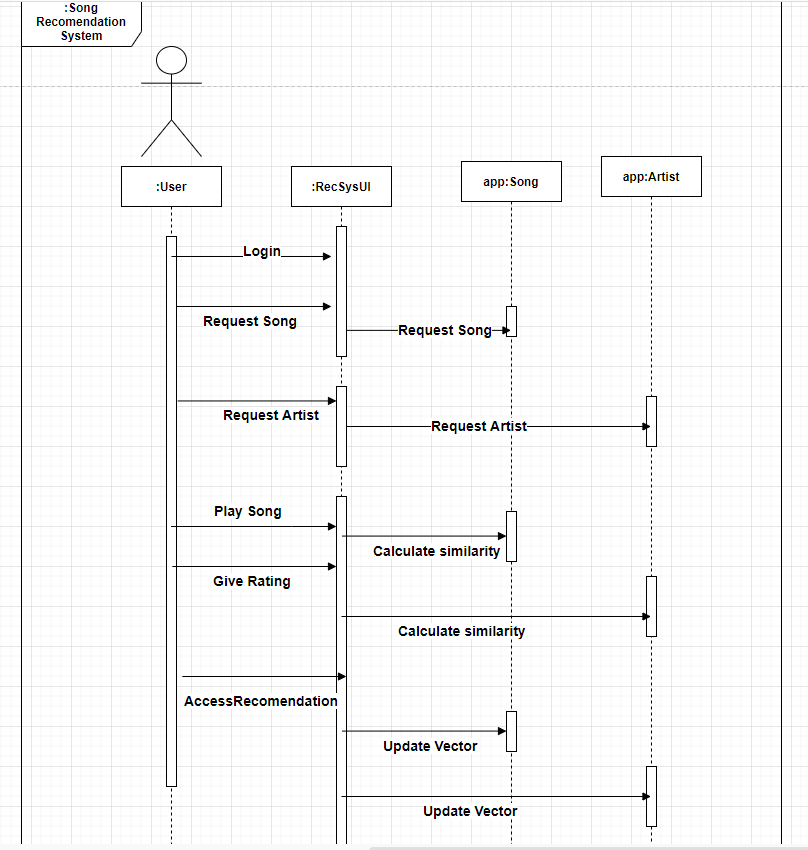
The goal of this material is to offer a thorough how-to manual for creating and deploying an AI-powered recommendation system for a streaming music service. It will address the ensuing topics:  
**Goals:** To provide specifics about the technologies, algorithms, and building techniques utilized to create the recommendation system.   
Contents: Detailed descriptions of the procedures for collecting data, designing features, building models, deploying them, and evaluating users will be provided.   
**Exclusions:** The finer points of configuring backend infrastructure or implementing intricate code will not be covered in this documentation.   
Time restrictions may prevent some technical explanations from going as far as they should, but every attempt will be taken to provide readers a comprehensive grasp of the project.

**Intended audience**  
This documentation's target audience consists of the following:   
**Developers**: The individuals responsible for creating and implementing the recommendation system.   
Project managers: Who need a summary of the technical information and the project's status.   
**Stakeholders:** People who are interested in knowing the technical details of the recommendation system, such as investors or executives.

**System overview**  
The recommendation engine offers tailored music recommendations based on user preferences and behavior in an effort to improve the user experience on the music streaming platform. The main objective of the system is to increase user pleasure and engagement by providing personalized music recommendations.   
Functionality: To provide tailored recommendations, it examines user activities such as playlists, likes, dislikes, and song plays.   
Architecture: The system processes user input and provides recommendations in real-time by using machine learning algorithms and data analytics techniques.   
Important benefits: The system's main benefits are content discovery, scalability, adaptability, personalized recommendations, and improved user experience.   
Technologies used: Flask/Django for backend APIs, TensorFlow/PyTorch for machine learning, SQL for database administration, and Apache Spark for processing massive amounts of data.

**Low Level System Design**

**Sequence Diagram**



**Navigation Flow**  
The user can receive tailored music recommendations from a dedicated section on the homepage.   
This section allows users to find music based on their activity and tastes.   
**Functionality of Search:**   
Users can look for particular musicians, albums, or songs using the search feature in the navigation bar.   
When users type keywords into the search bar, pertinent search results are returned.   
**Filters for genre and mood:**   
Recommendations can be filtered by the user according to preferred genres or moods.   
You can choose your favorite genres or moods from the dropdown menu or navigation sidebar, and the recommendations will be more tailored to your preferences.

**Defaults and Attributes**

**Attributes:**  
**Features of the Song:**   
When displaying song attributes in search results and suggestion lists, include the title, artist, album, duration, and release year.   
Give listeners the ability to filter and arrange songs according to criteria like genre, popularity, mood, or release date.   
**Features of the Playlist:**   
Add details about the playlist, such as the title, artist, number of songs, description, and cover image.   
After a playlist is created, allow users to change or alter its properties.

**Default Recommendations:**

Display default recommendations for new users based on popular genres or trending playlists.

Personalize recommendations based on user interactions once sufficient data is available.

**Components Design Implementation**

1. User Interface Components:

1. **Recommendations Section**:

* Create suggestion cards with the artist, title, and artwork of the song along with interactive features like "like," "add to playlist," and "play."
* Use dynamic loading to adjust suggestions according to user choices and interactions.

1. **Search Functionality**:

* Include auto-suggestion in the design of the search bar to help consumers narrow down their search terms.
* Put into practice search result pages that show relevant playlists, artists, albums, and songs.

2. Backend Components:

1. **Data Preprocessing**:
   * Design data preprocessing pipelines to clean, transform, and aggregate raw user interaction data.
   * Implement feature engineering techniques to extract relevant features for model training.
2. **Model Training**:
   * Design model training pipelines to train recommendation models using preprocessed data.
   * Implement machine learning algorithms such as collaborative filtering, matrix factorization, or neural networks.
3. **Recommendation Engine**:
   * Design recommendation engines capable of generating personalized recommendations based on user preferences and activity.
   * Implement algorithms for ranking and filtering recommendations to improve relevance and diversity.

3. Database Components:

1. **Recommendation Models**:
   * Design storage mechanisms for trained recommendation models, including parameters, weights, and configurations.
   * Implement versioning and model management strategies to track model performance and updates.

**Configurations/Settings**

**1.Model Configurations:**

**Selection of Algorithms:**   
Permit administrators to select from a variety of recommendation algorithms (such as content-based filtering, collaborative filtering, and hybrid models) in accordance with user preferences and system performance.   
**Hyper-parameters:**   
To optimize model performance, give recommendation models tunable hyperparameters such embedding dimensions, regularization terms, and learning rates.   
**Options for Feature Engineering:**   
Give administrators the ability to set up feature engineering methods like binning, normalization, and feature scaling that are used to prepare user interaction data.

**2. Preferences of the User: Genre and Mood:**   
Give customers the option to select their favorite moods and genres to receive tailored recommendations.   
Give users the ability to dynamically modify their preferences in response to changing interests and tastes.

**3.Options for Search and Filtering:   
Lookup Tables:**   
Permit users to narrow down search results by album, genre, artist, release date, and popularity.   
Offer sophisticated filtering choices to further hone search results.

**Content Limitations:**   
Use parental controls or user preferences to filter out explicit content or particular genres when implementing content limitations.

**Interfaces to other components**

**Content Metadata APIs: Third-Party APIs**   
Interfaces with other APIs (Spotify) to retrieve extra metadata, artist details, and song recommendations to enhance the content database of the recommendation algorithm.

**APIs for geolocation:**   
provides location-based suggestions, event recommendations, and geolocation-specific personalized information by integrating with geolocation APIs.

**Key notes**

**Tailored Suggestions:** The recommendation engine seeks to offer users tailored suggestions according to their listening habits, interests, and engagement. By providing personalized content, this increases user happiness and engagement.   
  
**Machine Learning Algorithms:** The recommendation system makes use of data analytics techniques and machine learning algorithms to examine user interactions, create personalized playlists, and promote new music according to user preferences.   
  
**Content Discovery:** In addition to suggesting well-known media, the system also emphasizes obscure artists, musical genres, and song titles to assist users in finding new material that suits their preferences.

**Scalability and Performance:** The recommendation system is built to process large volumes of user data and provide recommendations with minimal latency in real time. Scalability is essential to handle the increasing volume of data and user base.  
  
**Evolution and Adaptability:** The recommendation system is designed to continuously learn from and adjust to shifting consumer preferences and industry trends. This guarantees the recommendations' continued relevance and worth.

**Reference:**

1. Yoon, J., Lee, J., Kim, Y., & Kim, H. (2019). Deep Learning Based Music Recommendation System Using Triplet Loss. *IEEE Access, 7*, 39568-39578.
2. He, X., Liao, L., Zhang, H., Nie, L., Hu, X., & Chua, T. S. (2017). Neural Collaborative Filtering for Personalized Music Recommendation. *Proceedings of the 26th International Conference on World Wide Web*, 131-140.
3. Li, C., Liu, Y., Wu, J., Li, Y., & Wang, J. (2020). Content-based Music Recommendation via Hybrid Neural Networks with Feature Projection. *IEEE Transactions on Multimedia, 22(8)*, 2111-2123.
4. Liu, Z., Zhang, J., Zhao, Y., Zhang, Y., & Ma, S. (2018). Hybrid Music Recommendation via Latent Factor and Discriminative Feature Learning. *Information Sciences, 433*, 188-202.
5. Flask Documentation. (n.d.). Retrieved from https://flask.palletsprojects.com/
6. Django Documentation. (n.d.). Retrieved from [https://www.djangoproject.com/](https://www.djangoproject.com/" \t "_new)
7. TensorFlow Documentation. (n.d.). Retrieved from [https://www.tensorflow.org/](https://www.tensorflow.org/" \t "_new)
8. PyTorch Documentation. (n.d.). Retrieved from [https://pytorch.org/](https://pytorch.org/" \t "_new)
9. Redis Documentation. (n.d.). Retrieved from https://redis.io/documentation
10. pytest Documentation. (n.d.). Retrieved from https://docs.pytest.org/en/latest/
11. Last.fm API Documentation. (n.d.). Retrieved from https://www.last.fm/api
12. MusicBrainz API Documentation. (n.d.). Retrieved from https://musicbrainz.org/doc/Development/XML\_Web\_Service/Version\_2