Big Data Systems (S1-23_DSEOGZG522) Assignment

****Spotify Recommendations ****

Group Submission

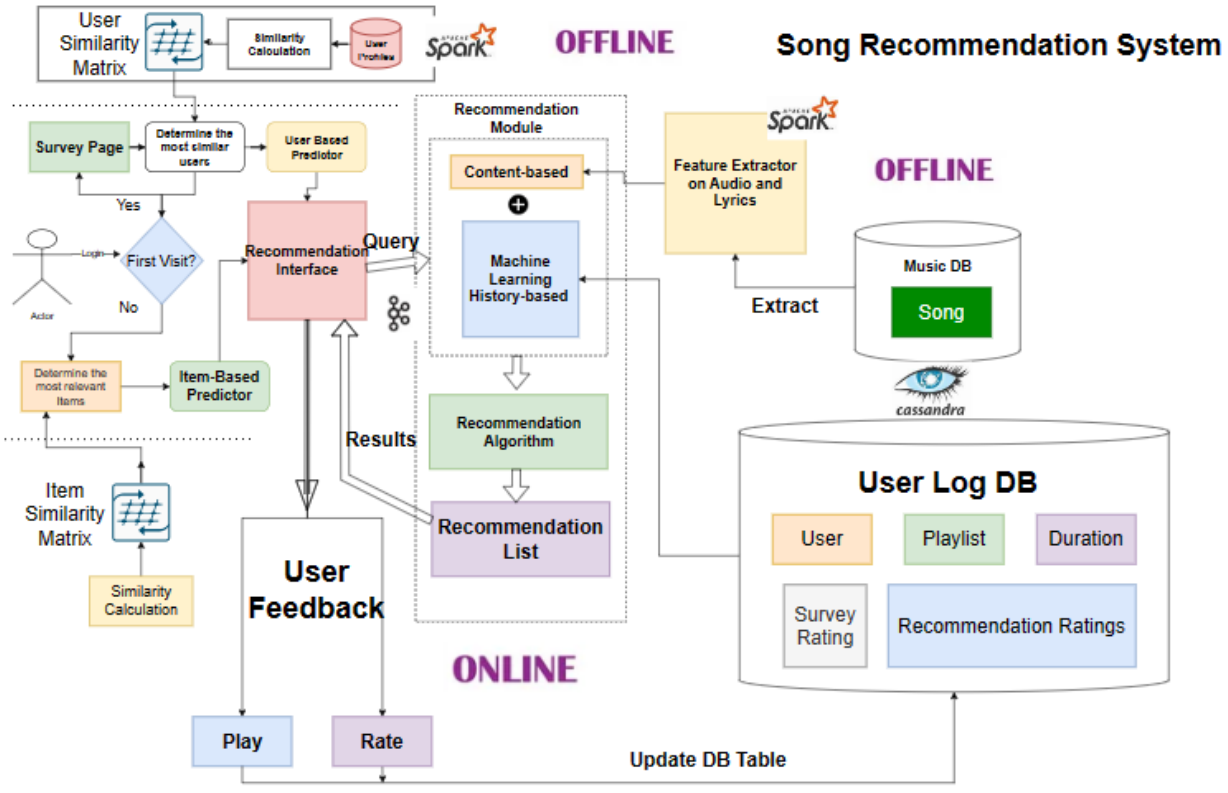
GAGAN GUPTA (2022OG04044) & VINAY SINGLA (2022OG04045)

Problem Definition

With multiple audio features such as danceability, energy, and valence, this dataset can be used to build a music recommendation system. By analyzing the preferences of users for certain genres and characteristics of tracks, the system can suggest similar tracks or even recommend new genres that users might enjoy.

Tools and technologies used in System Design:

- *Apache Spark*
- *Apache Spark SQL*
- *Kafka*
- *Cassandra*
- *Web Sockets, Spring Boot, Web Application Stacks (Bootstrap, Charts, etc.)*



Architecture Components:

To support both real-time stream processing and batch analytics use cases, we can design the following APIs, data models, and databases:

Backend APIs:

1. **Recommendation module** that will provide Recommendation list based on Recommendation Algorithm which intern uses Content-based filtering as well User history based machine learning. It takes inputs from user and recommended tracks or genres based on user preferences and track characteristics.
2. **User profile service** will be consumed to do similarity calculations for user similarity matrix to provide recommendations for new users. Also Allows users to manage their profiles, preferences, and feedbacks.

Data Models:

1. **Track/Song Data**: Stores audio features and metadata for each track, including genre information
2. **User Profile**: Stores user information, preferences, history, and feedback.

.

Databases/Storage:

1. **Cassandra** - Fast Distributed NoSQL DB for storing user data. NoSQL DB that can be scaled on many remote servers. It allows Spotify to manage scaling datasets such as user metadata with ease. It's open-source nature also reduces costs, as options like **MongoDB** may not scale as easily or cost more to use in enterprise.

2. **Amazon S3** - Static File Storage storing licensed Spotify songs to store the music it uses or outsource the hard work of setting up remote storage to Amazon and use their remote storage solution S3. S3 might be a good option due to its high availability and high fault tolerance when compared to other providers like Google Cloud and Azure.

Real-time Stream Processing:

1. **Apache Kafka** - Distributed Real-Time Computation for Search and Recommendation engine results. It optimized for real-time analytics. store frequently accessed data, such as user profiles, playlists, or recently played songs. Caching can significantly improve the performance and responsiveness of the application.
2. **Apache Spark Streaming**: Processes and analyses the streaming data to update user profiles and generate real-time recommendations.

Batch Analytics:

1. **Hadoop or Apache Spark**: Provides a big data batch processing environment for performing analytics queries on the dataset.
2. **Spark SQL**: Executes Map-Reduce jobs or SQL-like queries to perform analytical operations on the Spotify dataset.

Trade-offs in the CAP theorem:

Why Cassendra as NoSQL Database?

Cassandra: Sacrifices immediate consistency for high availability and partition tolerance. It favours availability and partition tolerance, allowing operations to continue despite network partitions, with the trade-off of eventual consistency.

MongoDB: Prioritizes consistency over immediate availability. It ensures strong consistency within a single replica set but might restrict availability in the case of network partitions to maintain consistency.

Why Kafka for Real Time Data Streaming ?

1. **Distributed Streaming Platform:** Kafka is designed for handling real-time data streams. It's excellent for handling events or data in motion.
2. **Message Broker:** It acts as a distributed messaging system, enabling communication between different parts of an application or between different applications.
3. **Scalability and Fault Tolerance:** Kafka's distributed nature allows for high scalability and fault tolerance, crucial for handling large volumes of data and ensuring reliability.

Why Spark SQL vs Hive?

Spark SQL often offers better performance due to its in-memory processing and distributed computing capabilities, making it preferable for real-time analytics and iterative algorithms. Hive, with its batch-oriented nature, might excel in processing large-scale data with less emphasis on real-time performance.

GitHub URL for all Documents and Videos:

https://github.com/sinvin-dse/spotify