SPOTIFY MUSIC RECOMMENDATION SYSTEM

Submitted By-

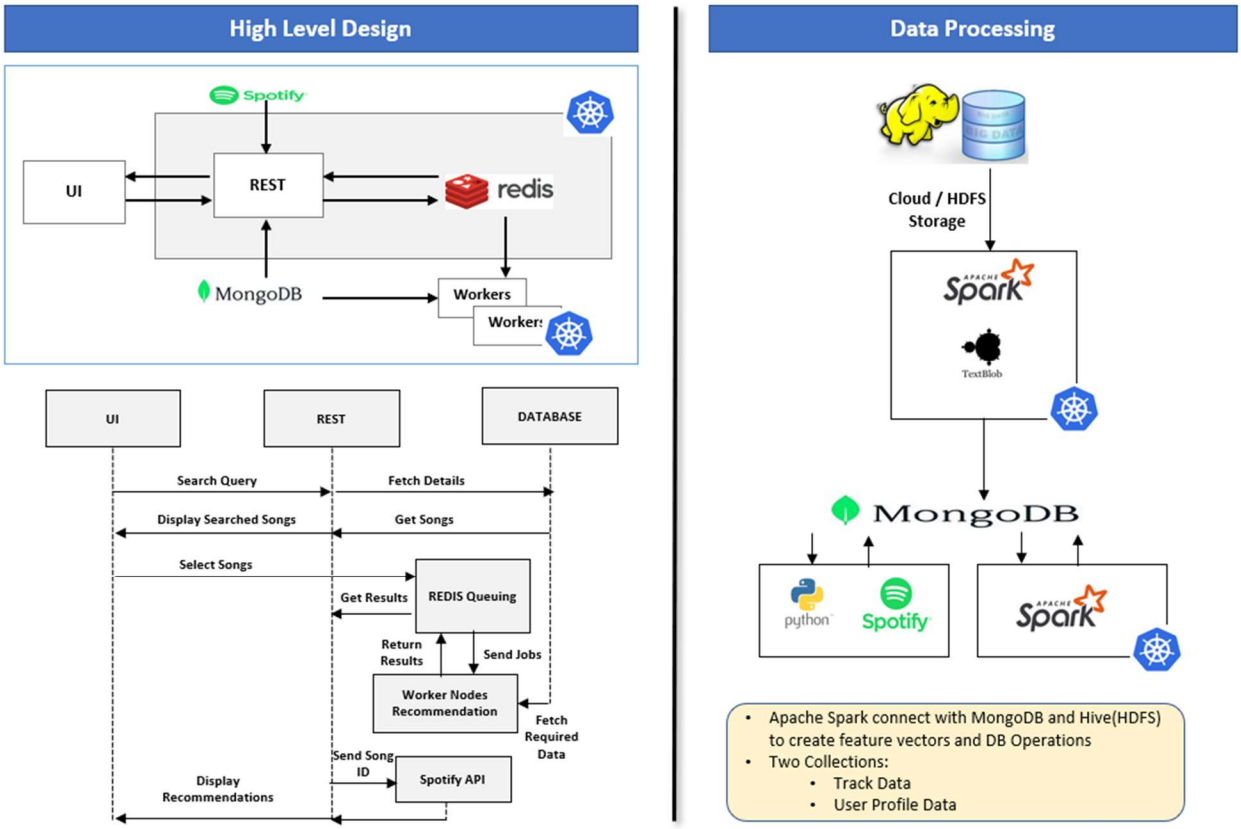
BALRAM (2022OG04028)

VIKALP SUHAG (2022OG04050)

Problem: Spotify Music Recommendation Platform

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.

Architecture:



Technology Stack:

- Azure Databricks and PySpark – Data Processing and analytics
- MongoDB - Database
- Hive HDFS - Database
- Redis - Messaging
- Flask REST server, JS - User interface
- Kubernetes/Azure Function App – Deployment
- Textblob - Sentiment Analysis

Design Components:

For the backend services, APIs, data models, and databases for the Spotify Music Recommendation Platform, we can consider the following design:

- **Backend Services (Worker Nodes):**
  A recommendation engine service that handles user requests and provides music recommendations based on user preferences and track characteristics. It should have below 2 Service – a.) An authentication service for user authentication and authorization.
  b.)A user profile service to manage user preferences, history, and feedback.

- **REST APIs:**
  Recommendation API: Accepts user input (e.g., song name, artist, genre) and returns recommended tracks or genres based on user preferences and track characteristics.
  User Profile API: Allows users to manage their profiles, preferences, and feedback.

- **Data Models:**
  User Profile: Stores user information, preferences, history, and feedback.
  Track Data: Stores audio features and metadata for each track, including genre information.

- **Databases:**
  NoSQL Database (MongoDB): Stores user profiles, preferences, and feedback. Provides high scalability and flexibility for handling large volumes of user data.
  HDFS Hive Database: Stores track data, including audio features and metadata. Offers structured data storage and efficient querying capabilities.

- **Real-time Stream Processing:**

Redis: store frequently accessed data, such as user profiles, playlists, or recently played songs. Caching can significantly improve the performance and responsiveness of the application.
Apache Spark Streaming: Processes and analyzes the streaming data to update user profiles and generate real-time recommendations.

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

Rationale and Trade-offs:

- **Why MongoDb as NoSQL Database?-** Our main reason for choosing MongoDb over other NoSQL data was- Its Cloud-Native Capabilities: MongoDB provides native integration with various cloud platforms, including Azure (As we are going to use Azure Databricks environment to do the data processing) and other reason were definitely scalability and flexibility in handling large volumes of user data, providing fast read and write operations. However, it sacrifices some level of data consistency compared to traditional relational databases.
- **Why Redis not Kafka for Real Time Data Streaming ?:** However we could have use both as these serves bit different purposes but for our use case we found Redis is able to do most of the work. In a Spotify-like application, Redis can be used for caching frequently accessed data, managing user sessions, and storing real-time analytics.
- **Why Apache Spark?** It provides efficient processing of streaming data and real-time recommendation generation and again Azure Databricks Native API package for Spark (PySpark) is very easy to integrate. Also Apache Spark offers a powerful batch processing environment for performing analytics queries on the dataset.