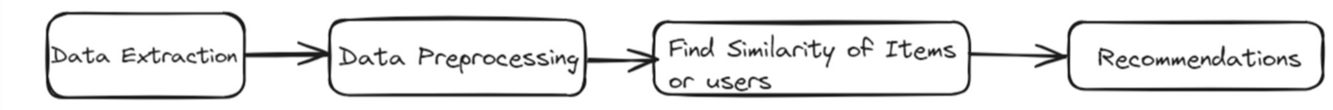
BDA Assignment – Part 1

Group Members:
1. Rishabh Mishra (2022OG04039)
2. Nirdosh Mishra (2022OG04021)

Introduction : Spotify Recommendations

Spotify recommendations are personalized song and playlist suggestions provided by the music streaming service, Spotify. These recommendations are generated using advanced algorithms and machine learning techniques to analyze user behaviour, preferences, and listening history. Spotify employs a combination of collaborative filtering, content-based filtering, and other recommendation methods to offer users a curated selection of music that aligns with their tastes.

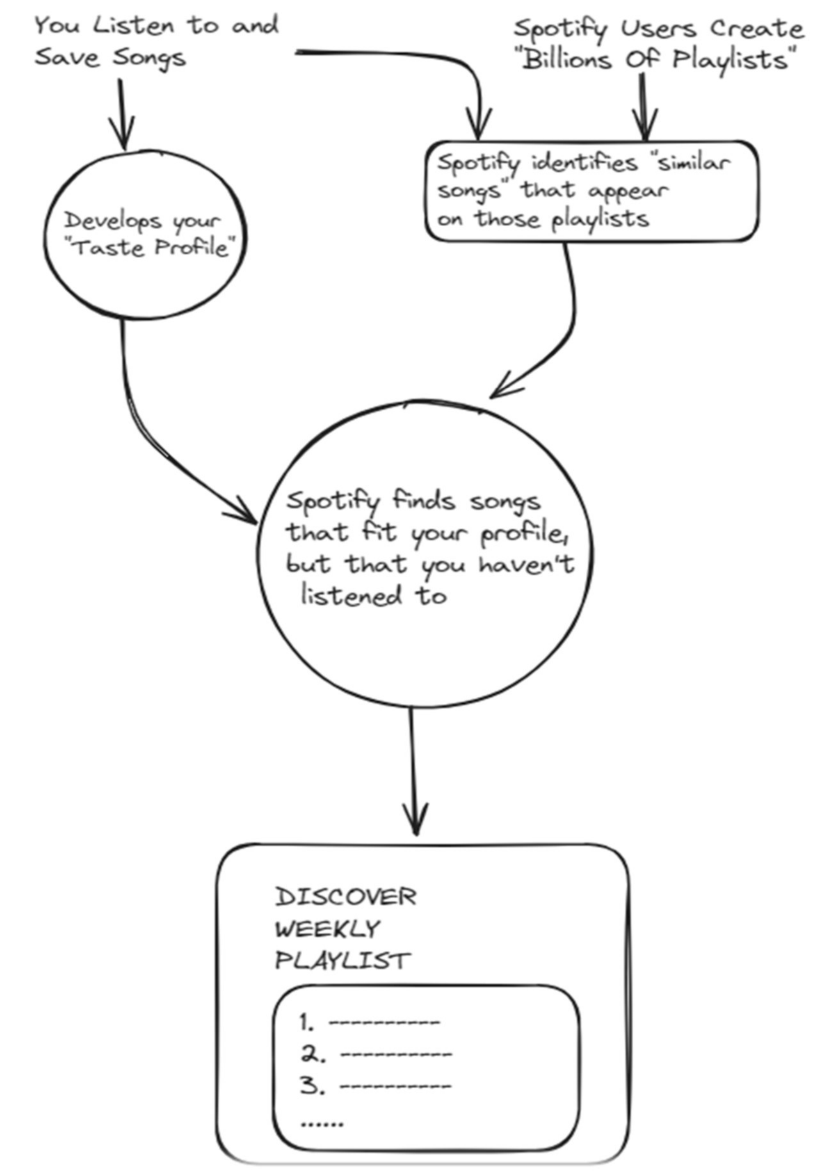
Design/Solution



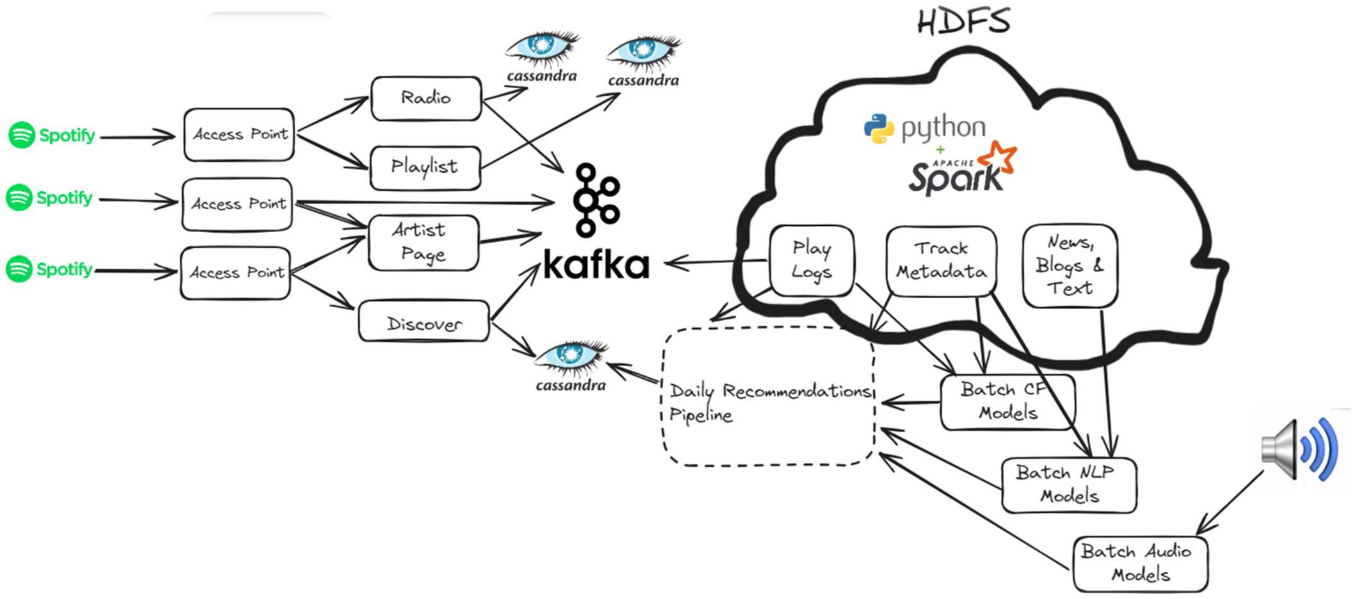
Dataset: https://www.kaggle.com/datasets/thedevastator/spotify-tracks-genre-dataset

We have thought to design recommendation systems using the above paradigms in the diagram which includes:
i) Data Extraction: Data extracted from the given link
ii) Data Preprocessing: This step we are processing the data to multiple systems to have recommendations to user.
iii)Find Similarity of Items or Users: This step in recommendation system is to identify the taste of the user.
iv) Recommendations: Finally recommending the user of their favourite genre type.

Data Flow


You Listen to and Save Songs
Spotify Users Create "Billions Of Playlists"
Develops your "Taste Profile"
Spotify identifies "similar songs" that appear on those playlists
Spotify finds songs that fit your profile, but that you haven't listened to
DISCOVER WEEKLY PLAYLIST
1. ----------
2. ----------
3. ----------
......

Architecture:

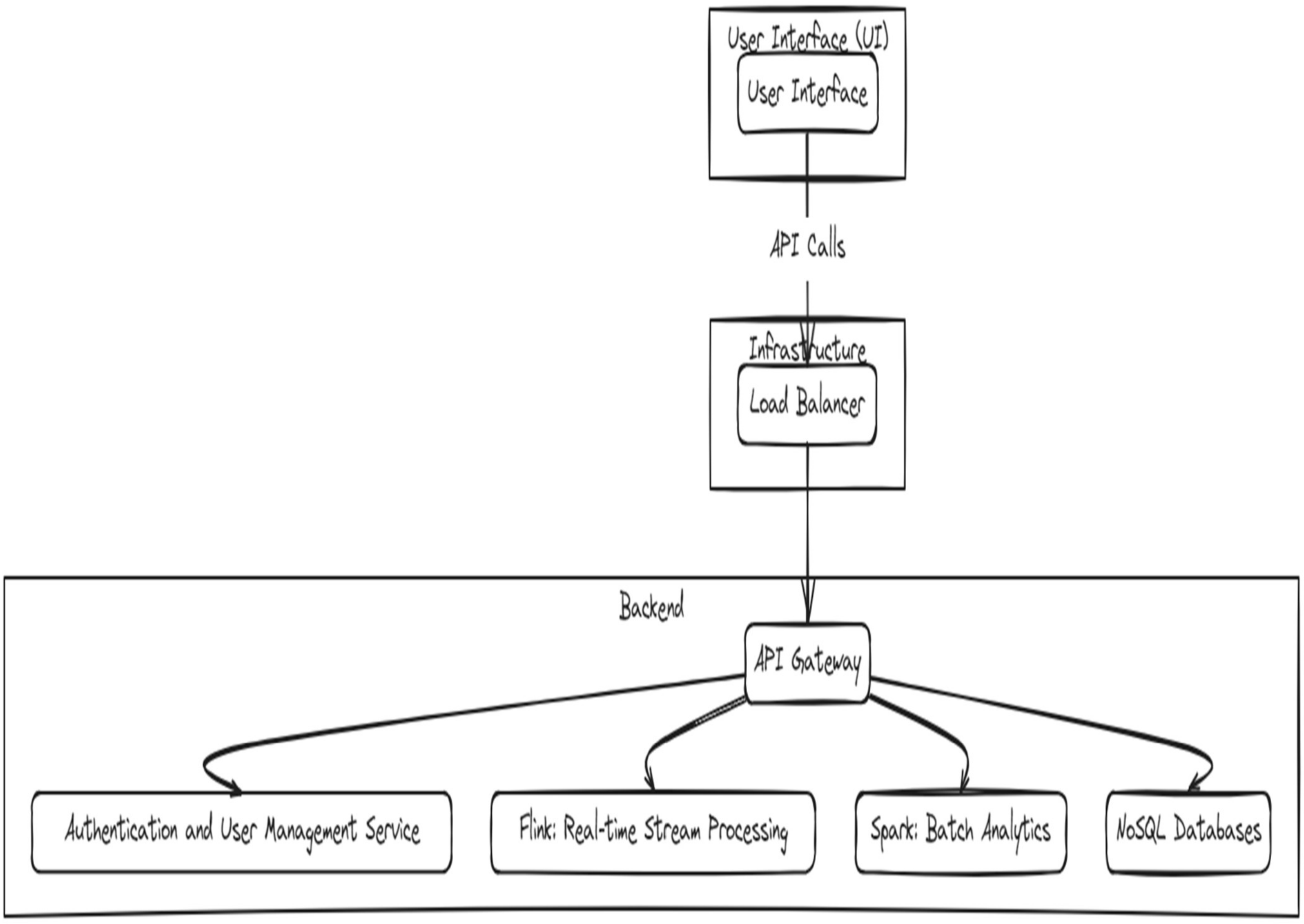


1. **User Interface (UI):**
    - Represents the user-facing application where users interact with the recommendation system.
2. **Frontend:**
    - Manages user interactions and communicates with the backend services via APIs.
3. **Load Balancer:**
    - Distributes incoming requests across multiple instances of backend services for scalability.
4. **API Gateway(Access Points):**
    - Serves as a single entry point for various services, routing requests to the appropriate backend service.
5. **Authentication and User Management Service:**
    - Handles user authentication and manages user profiles.
6. **Real-time Stream Processing (Apache Kafka):**
    - Consumes real-time events, updating user profiles and recommendation models in near real-time.
7. **Batch Analytics (Apache Spark):**
    - Performs batch processing on the dataset for periodic updates to recommendation models.
8. **NoSQL Databases:**
    - **Cassandra (Wide-column Store):**
        - Stores user profiles, preferences, and interaction history for quick access.
9. **Recommendation Engine:**
    - Utilizes collaborative filtering and track features to generate personalized recommendations.

Rationale and Tradeoffs:

- **CAP Theorem Tradeoffs:**
    - **Consistency:** Prioritizing eventual consistency to ensure low-latency updates for user profiles and recommendation models.
    - **Availability:** Ensuring high availability to provide a seamless user experience.
    - **Partition Tolerance:** Designing for partition tolerance to handle network failures.
- **Database Choices:**
    - Cassandra for scalability and quick access to user profiles.
- **Stream Processing:**
    - Apache Kafka chosen for its low-latency, event-driven architecture, suitable for real-time updates.
- **Batch Processing:**
    - Apache Spark selected for its ability to process large-scale batch analytics efficiently.

FLOW OF RECOMMENDATION ENGINE:


User Interface (UI)
User Interface
API Calls
Infrastructure
Load Balancer
Backend
API Gateway
Authentication and User Management Service
Flink: Real-time Stream Processing
Spark: Batch Analytics
NoSQL Databases