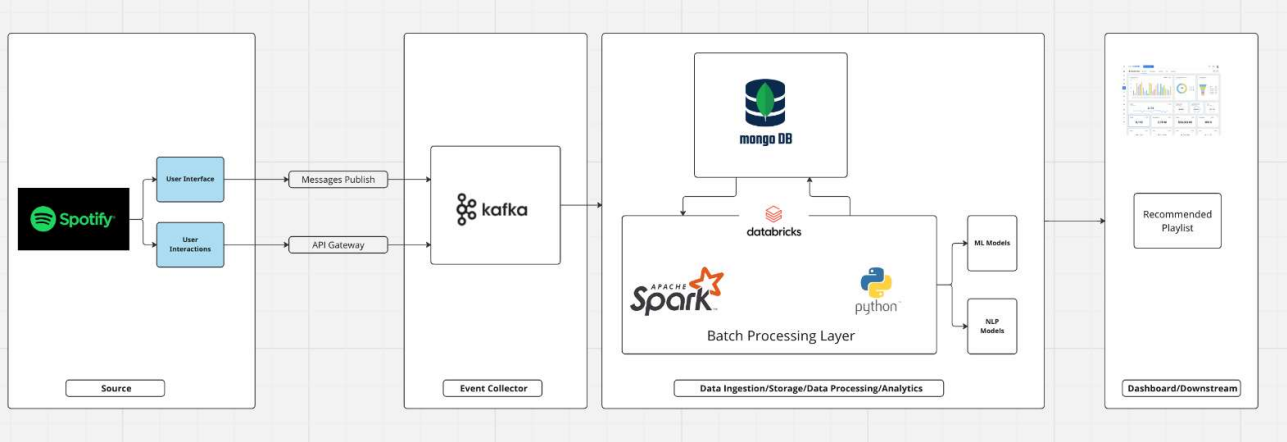
Name – Ajay Kumar
BITS ID – 2022OG04046

BDS Assignment – Sem 3

Option 1: Spotify Recommendations

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

1/ **Design:**



Below are some reasons for choosing different tech explained in the architecture:
In the context of Spotify's architecture, **MongoDB** can be a suitable choice as a NoSQL database for the following reasons:

1. **Scalability**: MongoDB is horizontally scalable, which means it can handle a large amount of data by spreading it across many servers. This is especially useful for a service like Spotify, which has a large, constantly growing dataset.
2. **Document-oriented**: MongoDB is a document-oriented database that can store, retrieve, and manage semi-structured data. This is beneficial for storing Spotify's data, which could include a variety of information about songs, albums, artists, playlists, user data, etc.
3. **Flexibility**: MongoDB does not require a fixed schema, which offers the flexibility to change the data model without downtime, an important factor for a dynamic and evolving platform like Spotify.
4. **Performance**: MongoDB can provide high performance for read and write operations. Indexing, replication, sharding, and in-memory storage engine are some of the features that enhance the performance of MongoDB.

In terms of the CAP theorem, MongoDB prioritizes Consistency and Partition tolerance (CP). This means that in the event of a network partition, MongoDB will ensure that all available data is consistent, even if it has to refuse some writes or reads to achieve it. This is suitable for a use case like Spotify, where it is critical that all users see the same data (e.g., song/playlist information, user-generated content), even if it may be slightly less available under certain conditions.

Apache Kafka is a distributed event streaming platform capable of handling trillions of events a day. It's used for real-time data pipelines, streaming analytics, data integration, and mission-critical applications. Here's why Kafka is an excellent choice for Spotify's architecture:

1. **Real-time Streaming**: Kafka provides real-time streaming capabilities which are crucial for Spotify's features like real-time music recommendations, active playlists, and activity feeds.
2. **Scalability**: Kafka is designed to handle data streams from many sources and deliver them to multiple consumers. It can easily scale up to handle Spotify's massive data streams.
3. **Durability and Reliability**: Kafka stores streams of records in a fault-tolerant way and provides strong durability guarantees. It's designed to work in distributed environments and tolerate failures, which is vital for a service like Spotify that needs to be always available.
4. **Integration**: Kafka can be integrated with various data processing tools like Apache Spark for real-time analytics, and storage systems like Hadoop HDFS or NoSQL databases. This flexibility is beneficial for Spotify's complex data architecture.
5. **Decoupling of Data Streams**: Kafka decouples data streams, meaning that the producers and consumers of data do not need to know about each other. This allows Spotify to add new systems (like a new recommendation engine) that consume data without affecting the systems that produce data.
6. **Order Preservation**: Kafka preserves the order of messages within a partition, which can be useful in certain situations where the sequence of events matters.

   Considering these factors, Kafka serves as an excellent backbone for handling Spotify's real-time data streaming needs. It provides a scalable and reliable solution to manage the high volume of data that Spotify generates and consumes.

Apache Spark is a powerful, open-source processing engine for big data built around speed, ease of use, and sophisticated analytics. It can be very beneficial for a data-rich application like Spotify. Here are some reasons why Spotify might use Spark:

1. **Processing Speed**: Spark performs processing tasks up to 100 times faster than MapReduce in memory, or 10 times faster on disk. Spotify has large amounts of data being generated every day, and Spark can help process this data more quickly.
2. **Ease of Use**: Spark comes with user-friendly APIs for Scala (Spark is built on Scala), Java, Python, and Spark SQL. This can make it easier for developers to work with.
3. **Advanced Analytics**: Spark not only supports 'Map' and 'Reduce' operations, but also supports SQL queries, Streaming data, Machine Learning (ML), and Graph algorithms. This makes it a great tool for analyzing Spotify's user data to make song recommendations, analyze listening trends, etc.

4. **Real-Time Processing**: Spark can process real-time data through Spark Streaming. This is useful for Spotify to provide real-time analytics, create dynamic playlists, or even suggest songs based on a user's current activity.
5. **Machine Learning**: Spark's MLlib is a machine learning library that provides various algorithms for classification, regression, clustering, and collaborative filtering, as well as model evaluation and data preparation. These tools can be used to provide personalized content to Spotify users.
6. **Integration**: Spark can be easily integrated with Hadoop and can process data stored in Hadoop Distributed File System (HDFS), HBase, or any other data source supported by Hadoop.

2/ **OLTP Queries:**

BDS_Assignment_MongoDB_Query_2022OG04046_AjayKumar - https://www.loom.com/share/92a863f0b6fb4ddbb62ba9b9e5075176

Below are the Queries that were used to run over the MongoDB.

Database Name created = **d_assignment**

Collection created = **c_assignment**

Source File Name - **train.csv (Dataset:** https://www.kaggle.com/datasets/thedevastator/spotify-tracks-genre-dataset)

|Step 1 - will be to import the source train.csv in to the MongoDB |

---

1. After import of csv filefind total number of records:

```
db.c_assignment.countDocuments()
```

2. To Print all the fields names

```
var doc = db.c_assignment.findOne();
for (var key in doc) {print(key)};
```

3. Creating index for optimisation

```
db.c_assignment.createIndex({"artists":1})
```

4. Find Operation with Projections

```
db.c_assignment.find({"artists":"Dan Berk"},{"track_name":"solo","popularity":52})
```

5. Update in MongoDB

```
db.c_assignment.updateOne({"track_id":"4ptDJbJl35d7gQfeNteBwp"},{$set:{"popularity":10}})
```

6. Delete in MongoDB

```
db.c_assignment.deleteOne({"track_id":"4ptDJbJl35d7gQfeNteBwp"})

db.c_assignment.deleteMany({"artists":"Jason Mraz"})
```

3/ **OLAP Analytics Queries:**

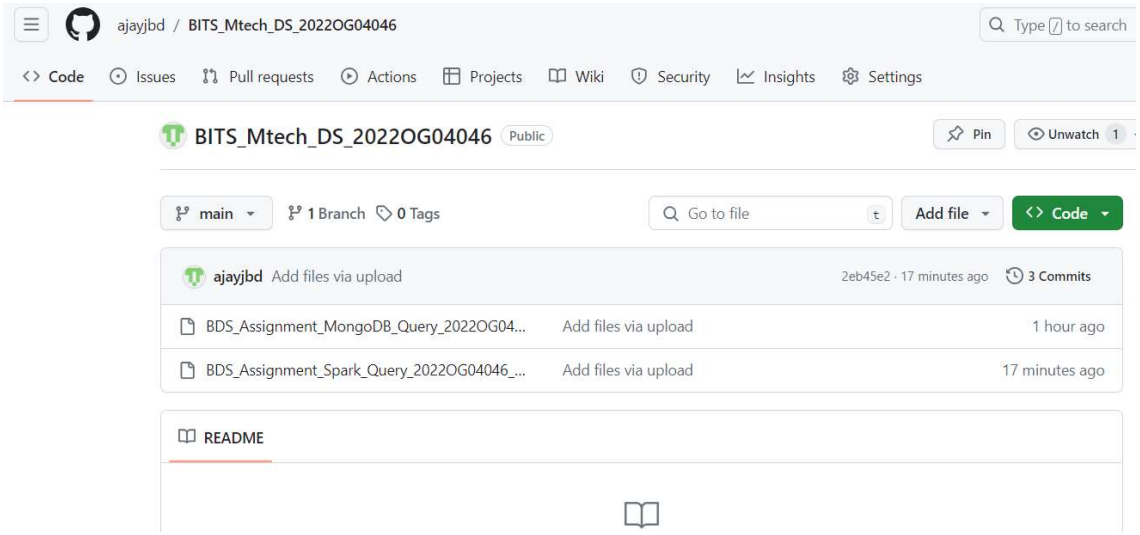
BDS_Assignment_Spark_Query_2022OG04046_AjayKumar - https://www.loom.com/share/2b26a0651d9840c4b187eb754506930d

Will share separate Databricks pdf file that has queries and the output of the spark queries for OLAP Analytics.

Both Looms videos are also uploaded over the Github as well

https://github.com/ajayjbd/BITS_Mtech_DS_2022OG04046

Below is the snippet.



Name – Ajay Kumar
BITS ID – 2022OG04046