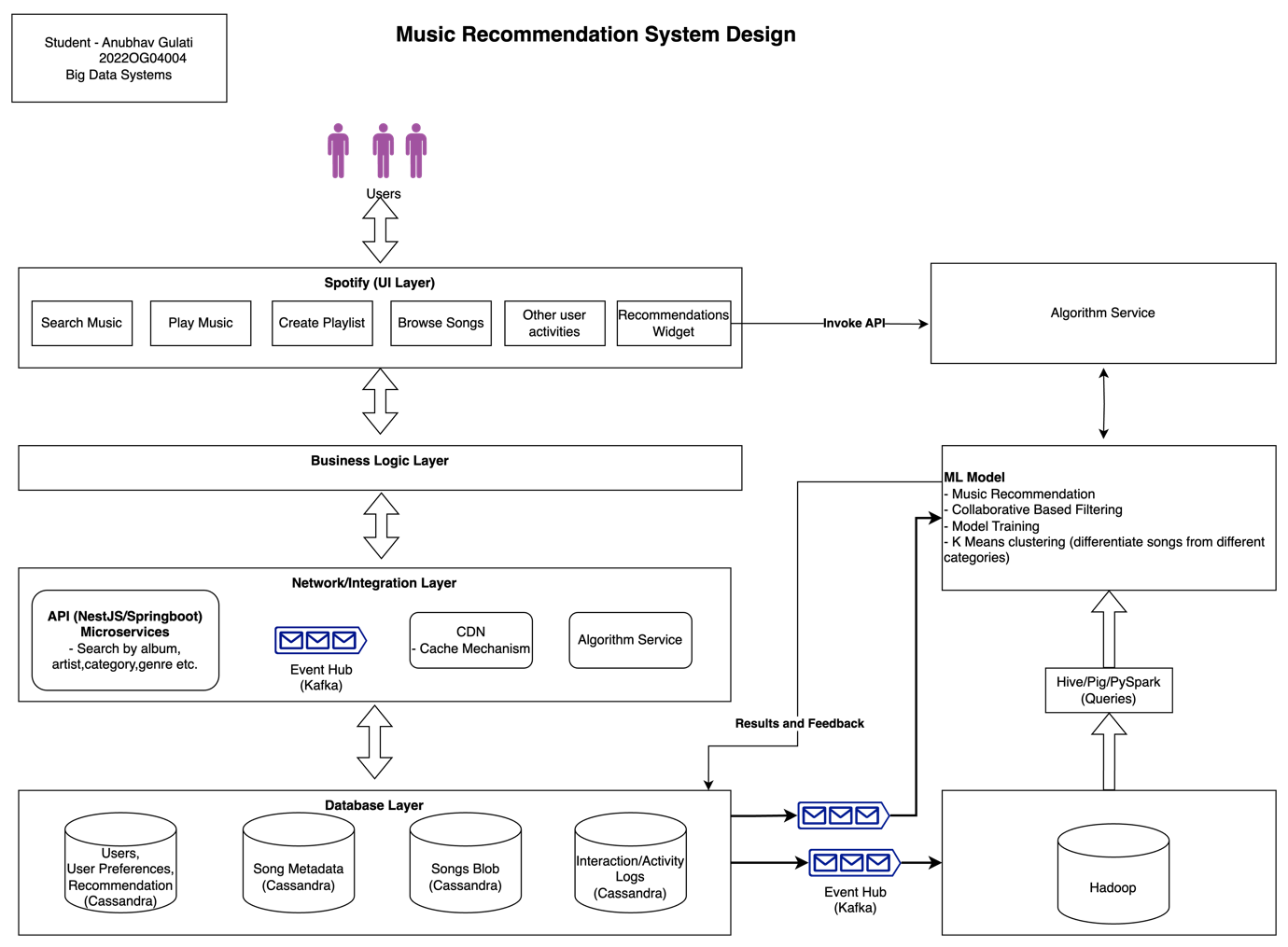
**BIG DATA SYSTEM**

**Anubhav Gulati – 2022OG04004**

**Requirements:**

1/ **Design:** Design the backend services, APIs, data models and the databases you will use for this. The design should support both realtime stream processing and batch analytics use cases. Explain the rationale for your design especially tradeoffs wrt CAP theorem, describe the tech choices for the databases and other components. You must use a NoSQL database as part of your design (MongoDB, Neo4j, DynamoDb, HBase, anything appropriate for your solution).



Music recommendation system design is going to support both the real time stream processing and batch analytics use cases.

**Different architectural layers are shown in the diagram, starting from top – user login.**

* UI Layer – User can login on spotify app and can do multiple operations like search music, play the music, create playlist, browsing of songs and all other user interactions.
  + Also it displays the recommendation songs on the recommendation widget.
* Business Layer – UI and any business logic should be separated. UI should not have any business logic as it can degrade the performances.
* Network/Integration Layer – Just for diagram have shown it in one, but can be considered as 2 different layers.
  + CDN will make sure to help us in latency issues. Information can be cached to improve the latency and system should play next song or search and display the songs fast. That’s one of the key goals.
  + This layer also has all other integration with other systems either by functions developed on nestjs/spring boot and kafka integration too.
  + API functions – all different sort of API’s need to build to query the song information by different songs metadata. This is also deployed on cloud as Serverless functions.
* Database layer – All the information whether its about user, user preferences, what user likes, what user does not like, songs metadata, songs in blob storage, user interaction history gets stored in cassandra.

**Trade off –**

* CAP theorem – Consistency, Availability, Partition Tolerance
  + distributed system can deliver only two of three desired characteristics
* There are several NOSQL Databases available. It’s purely depend on the use case what we need, what’s our focus on.
* Between Cassandra and Mongo DB
  + Mongo DB is CP system
  + Cassandara DB is AP system.
* In this case we are going with Cassandra DB, as it can easily store our data and availability is the most thing needed here rather than consistency.
* We are not doing intensive read writes, that’s another reason to go with Cassandra.

**Real Time Recommendation**

* Recommendation can be real time and batch based.
* Based on the user online search, all those key interactions will be logged and it will be fed to ML model, and based on the ML model design, collaborative based filtering, K means clustering, recommendation will be derived which will be shown to user on recommendation widget.
* Users action on recommendation songs whether user plays, or do not plays , all these interactions will be stored and feed to the ML Model to improve the user recommendations.
* Algorithm service – This service is used which will be invoked by UI layer to get the recommendation lists on UI. Behind the scene its invokes model and try to get the recommendations.

**Batch Recommendation**

* All day history and much more info data about the user, can be passed to big data system such as Hadoop via Kafka.
* Data analytics can be done on such large amount of data.

All images and this document is attached on github.

* Same ML model can be used for batch also and model will derive the recommendation which can be stored and based on this recommendation will be displayed to the user.

Even to store users info, its better to go with NoSQL rather than RDBMS database.

2/ **OLTP Queries:** Execute at least 3 key read/write queries against the NoSQL database as part of the solution with the optimizations and the consistency levels set in your queries.

**Solution notes:**Submission for this part MUST include a pdf of the queries and a video of their successful execution (you can use loom for the video). No more than 2 pages for the queries and no more than 1 minute for the loom video.