***Microservices, queues - how do you orchestrate? Is this a monolithic app or components, or in a platform like Databricks?***

Totally agree—this setup should lean towards microservices instead of a monolithic app. We’ll break down the components like the Agentic LLM, Drift Detection, and Data Quality modules into separate microservices. This will make scaling easier and keep the system flexible. Kubernetes seems like a good fit for orchestrating everything, especially if we go big with Databricks for large-scale processing. But, if things get too complex, we could start simpler and bring in Kubernetes later.

***Currently uses Pandas, but should you use Spark DataFrame API to get scale?***

Exactly—moving from Pandas to Spark DataFrame API is the way to go for handling bigger datasets and distributed processing. Spark will give us the scalability we need as the system grows, especially for drift detection and validation tasks.

***Model versioning, automated retuning***

Couldn’t agree more—model versioning and automated retuning are key. Let’s bring in MLflow to handle model versions in our registry. This way, we can keep track of different model versions, whether for drift detection, validation, or synthetic data generation. We should also set up pipelines for automatic retuning whenever there’s significant drift or new data. But, if model updates are rare early on, we might focus on versioning first and roll out full automation as we grow.

***How can you make data drift and other operations on worker nodes of a Spark cluster?***

Spot on. We need to run drift detection and similar tasks across Spark worker nodes to tap into Spark’s parallel processing power. If the current drift detection load is light, we could start on a single node or a smaller setup and scale up to a full Spark cluster as data volume increases, that's a great point.

***Storing results in a Hive-compatible format means can read from directory instead of table management through the app itself, or is it better to have RDBMS?***

Using a Hive-compatible format is a solid choice. We should adjust the design to store results this way, streamlining data access and big data tool integration.

### **Additional Considerations:**

**How can we design a feedback loop that not only updates the Knowledge Base but also continuously enhances the performance of the entire system?**

**What can we do to ensure the system remains user-friendly and adaptable across different teams or organizations?**

**Is there a way to make the system’s decision-making process more transparent to end-users?** **Why were certain decisions made by the Agentic LLM ?**

**how do we ensure the system scales effectively as data volumes grow?**