# 2nd ROUND OF FEEDBACK

***-Lewis Injai***

**DESIGN FINDINGS OPINIONS**

- Limitation on data storage used in the design. System is both heavy and write heavy. Read in fetching data for users/getting budgets and write in getting transactions from banks.
- Good proposition on the GDPR enforced around the data received from the bank. First measure of security in my opinion is whitelisting IP's from the RPC's from the bank.
- Before developing ML tools, find best way to maintain efficiency and increased performance on data retention.
- As stated earlier there is no much focus on the business logic and its operation. The design is high level enough hence incase of optimizing a particular service, the design presents a gray area.

**LOOM FEEDBACK**

*You can find the loom recordings in the links given below*
*Link 1, Link 2, Link 3*

Data Persistence

Considering the possibility of having multiple databases for read and write functions regarding the actions taken by each service.

Transactions data from banks and fintechs is write heavy and there should be a database for recording those transactions.
This will be the parent database and other database will share data from the parent.

To enable synchronization between the databases, write and read heavy i recommend a microservice built for the same functions.

Basically, it will act as a conduit enabling a message pipeline to persist data from the the parent to the child(read) databases.

Service Communication

Decoupling services as much as possible will be a key factor in efficiency improvement on the system. The services as stated in my design is basically each block acting as a microservice and communication between the microservices is a messaging system.

I prefer a publish-subscribe messaging model so that incase a service goes down, data is still retained in the queue and once up the subscriber continuous from where it started. RabbitMq seems to be the most widely accepted messaging queue.

From my design, the users interaction is guided by calls to different microservices hence the link between two services will be more of an asynchronous calls made between microservices.

Its paramount to state for efficiency and performance, the number of rpc's should be reduced as much as possible.