Problem Statement

Imran Ahmed, the CTO of a rapidly growing social media tech startup, faces the daunting task of ensuring the smooth operation of his company's distributed services across multiple regions. This task becomes increasingly difficult without a centralized observability framework, resulting in a number of critical issues, including delayed issue resolution, a poor user experience, and slowed feature deployment.

The situation is further complicated by inefficient root cause analysis due to a lack of comprehensive monitoring and logging, limited visibility into system performance, and difficulty in scaling operations effectively. Estimating and commissioning infrastructure becomes even more difficult for not having proper system resource utilization metrics data.

Imran Ahmed needs help implementing a comprehensive, scalable, and observable solution that not only meets the current needs but is also adaptable to future demands, ensuring monitoring, enhancing security, and maintaining a high-quality SDLC process.

Now Participants can help Imran Ahmed's business by implementing:

Centralized Observability Framework: Create a scalable, cloud-agnostic system for comprehensive monitoring and logging.

Efficient Root Cause Analysis: Implement advanced algorithms or AI-driven tools to quickly identify and address the root causes of issues.

Performance Optimization: Develop real-time analytics tools to monitor system performance, predict bottlenecks, and suggest optimizations.

Scalability Solutions: Design a framework that easily scales with the growing demand, ensuring smooth operations even under heavy loads.

Developing a central visual monitor: Collect data and build a visual monitoring system so that tech/business people can see the overall technical load, need, and consumption.

Hackathon Problem

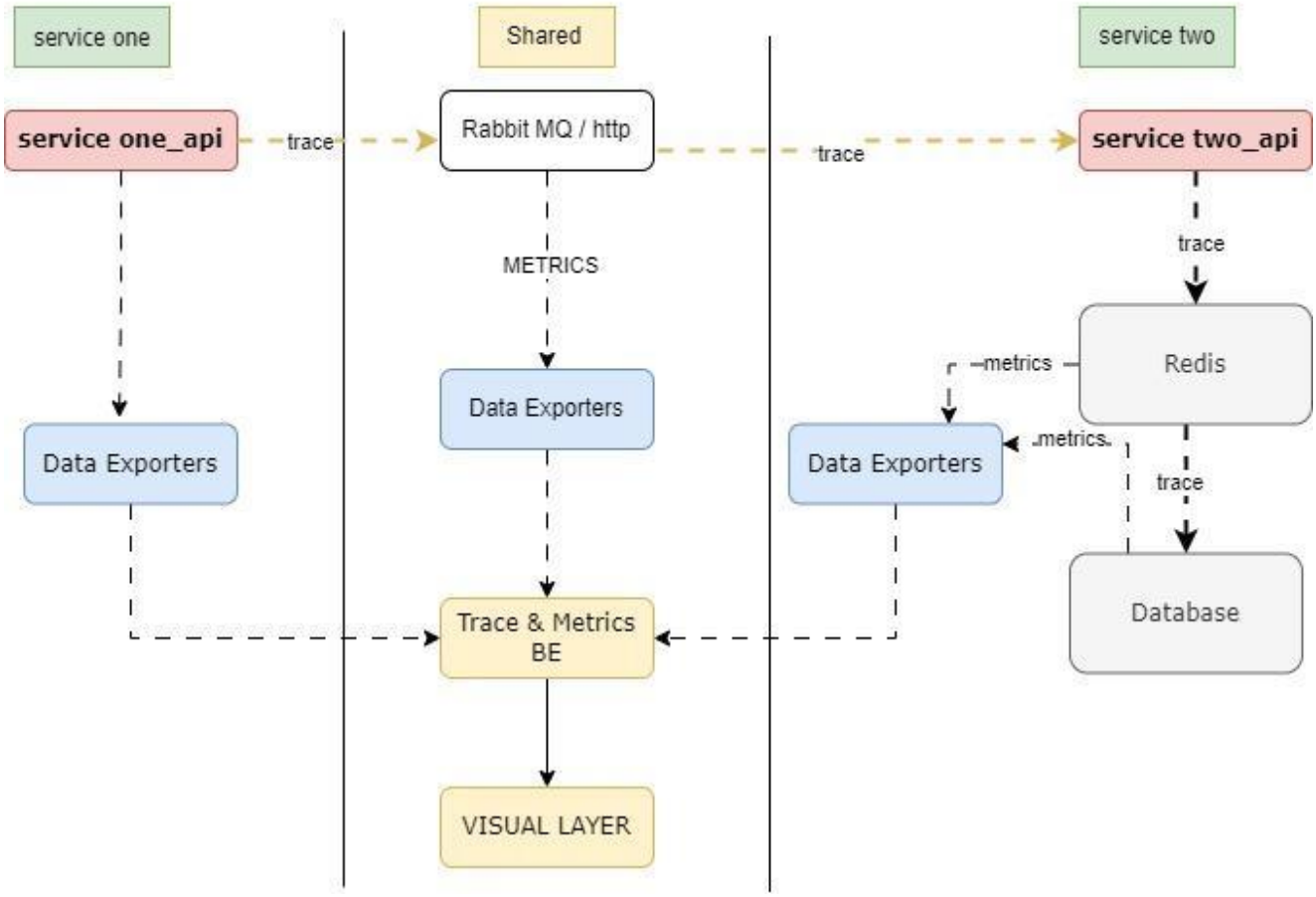

Figure: System Can Be Designed Like this

Service One Api:

1. Simple api to endpoint which can call internal business logic functions and validation functions. Design this api to create initial trace and spans (trace can be propagated within service one also).
2. Collect, save, and show service level trace/metrics/logs by applying observability tools, frameworks, and plugins (free to use any).

Service one api will be serving the end-user to Twitter feed {json payload of friends posts}, to get the feed service one first takes the request, checks the validity of the user, and passes the request to service two to get the Twitter feed data. Service one's challenge is to serve in high-traffic situations.

Service Two Api

1. Responsible for serving feed data, to get read data MongoDB can be used. /GET feed data will be getting data from the db layer.
2. Implement a redis cache layer to serve from the cache wherever possible.
3. Save user request data to postgres/mysql so that the system can track users' feeds. [i.e if the system wants to track user-wise feed]

While making these services find out where observability can help to solve the business problem and make the tech navigation easy. To track service time at each layer, collect data service wise any tools and plugins can be used, but keeping the observability convention in mind. Teams should focus on monitoring data that eventually helps to track latency, throughput, vulnerability, performance metrics, and root cause analysis via a visual monitoring system.

A team can also pick different user stories (service one and service two requirements) to implement the same!