Quiz 2 – A

Q1) Please write your name and student ID [5 pts].

Q2) A hospital network collects data every few seconds from hundreds of IoT medical devices (ECG, oxygen monitors, IV pumps). Each device sends time-stamped readings to a central server for both real-time alerts (e.g., patient vitals out of range) and long-term analytics (e.g., trend detection). The system must store data for at least 90 days and allow reprocessing of old messages when models are updated. Which messaging system would you choose — RabbitMQ or Kafka — and why? [25 pts].

Kafka

- 90-day retention + replay: Kafka is a durable event log; you can keep data for months and reprocess it later for model retraining. RabbitMQ is a transient queue—once consumed/acked, messages are gone.
- High-throughput streaming: Handles continuous telemetry from hundreds (→ thousands) of devices via partitioned topics; scales horizontally.
- Real-time alerts supported: A low-latency consumer (or Kafka Streams/Flink) can read the live topic and trigger alerts immediately. You still get historical storage for audits and ML.
- Exactly-once/at-least-once processing options: Useful for reliable alerting and analytics pipelines.

Q3) MapReduce is designed to process massive datasets reliably across many machines. Explain the architectural features that ensure fault tolerance and enable parallel execution in the MapReduce framework.? [25 pts]

MapReduce achieves fault tolerance through task re-execution and data replication. If a node fails, its task is automatically reassigned to another node using copies of the input data stored in a distributed file system (e.g., HDFS).

It achieves parallel execution by dividing the dataset into independent splits that are processed simultaneously by multiple mappers and reducers. This allows large-scale data processing to be distributed across many machines while maintaining reliability and scalability.

Q4) Write down the name of your favorite book (I know, so many ☺, choose one). The following answers are also acceptable: *prefer not to share, I don't have any,* etc. [5 pts].

Q5) You're developing a large-scale e-commerce platform (like Amazon or Shopify). When users browse and shop, the system must keep track of each user's shopping cart in real time. Each cart is linked to a unique session ID, and users frequently add, remove, or update items. The data for one user's cart is independent of others, and the application only ever needs to fetch or update a single cart at a time.

The system must support: 1) Extremely fast reads and writes (to update the cart instantly), 2) Scalability for millions of users, and 3) Simplicity, since no complex relationships or queries are needed. Which NoSQL database type would be most appropriate, and why? [40 pts]

Speed: Key–value stores (like Redis, Amazon DynamoDB, or Riak) are optimized for constant-time lookups (O(1)), ideal for real-time cart updates.

Simplicity: Access patterns are predictable — all reads and writes are done by session_id, with no need for joins or secondary indexes.

Scalability: These databases easily scale horizontally by sharding keys across servers, enabling support for millions of concurrent users.

Flexibility: You can store the cart as a serialized JSON, making it easy to modify without schema changes.