# Distributed Systems Group Assignment

## Title: Designing a Fault-Tolerant Distributed Messaging System

## Scenario:

Your team has been assigned the task of designing a fault-tolerant distributed messaging system where clients send messages to each other through a set of distributed servers. The system must support real-time message delivery, storage, and retrieval, ensuring high availability, fault tolerance, and consistency across multiple servers.

## Tasks:

Each team member will focus on one core aspect of the system, but the team must collaborate to ensure integration and seamless functionality.

### 1. Fault Tolerance (Member 1):

Objective: Ensure the system continues delivering and retrieving messages even in case of failures.

- Implement message redundancy mechanisms (e.g., message replication, erasure coding) across multiple servers.

- Design a failure detection system to identify when a messaging node becomes unavailable.

- Develop an automatic failover mechanism to redirect messages in case of server failures.

- Propose a message recovery mechanism for nodes that rejoin the system.

- Evaluate the impact of redundancy on system performance and storage overhead.

### 2. Data Replication and Consistency (Member 2):

Objective: Ensure messages are replicated across multiple servers while maintaining consistency and availability.

- Design a replication strategy (e.g., quorum-based, primary-backup, or sharding).

- Choose a consistency model (e.g., strong consistency, eventual consistency) and justify the trade-offs.

- Implement a deduplication mechanism to handle duplicate messages caused by retries or failovers.

- Optimize message retrieval performance while ensuring consistency across servers.

- Analyze how replication impacts latency and storage efficiency.

### 3. Time Synchronization (Member 3):

Objective: Ensure messages have accurate timestamps for correct ordering and debugging across distributed servers.

- Implement a time synchronization protocol (e.g., NTP, PTP) across all messaging servers.

- Analyze the impact of clock skew on message ordering and consistency.

- Develop a mechanism to reorder messages that arrive out of sequence due to network delays.

- Implement message timestamp correction techniques to ensure event accuracy.

- Evaluate the trade-offs between synchronization accuracy and system overhead.

### 4. Consensus and Agreement Algorithms (Member 4):

Objective: Ensure distributed messaging servers agree on message storage, indexing, and retrieval policies.

- Research and implement a consensus algorithm (e.g., Raft, Paxos) for distributed message consistency.

- Design a leader election mechanism to manage message coordination.

- Evaluate the performance of the consensus algorithm under high message transmission rates.

- Propose optimizations to reduce consensus overhead while ensuring consistency.

- Test the system under different failure scenarios (network partitions, node crashes).

## Deliverables:

1. Report:  
 - A detailed 10-12 page report explaining the design choices, message replication strategies, and system optimizations.

2. Implementation:  
 - A working prototype of the distributed messaging system using Python, Java, or Go.

3. Presentation:  
 - A 15-minute presentation explaining the architecture, implementation, and evaluation.

## Evaluation Criteria:

|  |  |  |
| --- | --- | --- |
| Criteria | Weight | Description |
| Fault Tolerance | 20% | Effectiveness of failure detection, message redundancy, and recovery mechanisms. |
| Data Replication | 20% | Efficiency of message replication, consistency guarantees, and retrieval speed. |
| Time Synchronization | 20% | Accuracy of timestamps and resolution of out-of-order messages. |
| Consensus & Agreement | 20% | Reliability and efficiency of the consensus algorithm for message coordination. |
| Overall Integration | 20% | Seamless operation of all components and scalability of the system. |

## Submission Guidelines:

- Submit the report, source code, and presentation slides in a single zip file.

- Include a README file with the names, registration numbers, and emails of the team members, along with instructions for running the prototype.