LAIKIPIA UNIVERSITY

UNIVERSITY EXAMINATIONS

FIRST SEMESTER 2025/2026 ACADEMIC YEAR

FOURTH YEAR EXAMINATION FOR THE DEGREES OF BACHELOR OF SCIENCE IN COMPUTER SCIENCE AND ICT

COMP 411/BICT 317: DISTRIBUTED SYSTEMS

STREAM: R ***TIME: 2 HRS***

DAY: WEDNESDAY [8.30 – 10.30 A.M] ***DATE: 04/02/2026***

THIS QUESTION PAPER CONSISTS OF SEVEN (7) PAGES

PLEASE DO NOT OPEN UNTIL THE INVIGILATOR SAYS SO.

Answer Question 1 and any TWO other questions

QUESTION ONE (Compulsory) (30 Marks)

a) Compare and contrast the client-server and peer-to-peer architectural models, providing one example for each. [**2 Marks**]

b) What is the primary role of middleware in a distributed system? Provide one specific example. [**2 Marks**]

c) Briefly explain the fundamental difference between Message Passing and Remote Procedure Call (RPC). [**2 Marks**]

d) What is the primary purpose of a stub in the context of Remote Procedure Call (RPC)? [**1 Mark**]

e) What is the key problem that Logical Clocks (e.g., Lamport timestamps) solve, and what is their main limitation? **[2 Marks**]

f) Briefly explain the key conceptual difference between Cluster Computing and Grid Computing. [**2 Marks**]

g) A client makes an RPC (Remote Procedure Call) call to a server. Illustrate, with a sequence diagram or a step-by-step description, the sequence of events that occur from the call initiation to the return of the result, including the role of the client and server stubs [**3 Marks**]

h) Define a fail-silent (crash-stop) failure model. [**1 Mark]**

i) You are tasked with running a highly parallel, data-intensive scientific calculation. Justify whether you would choose a Cluster Computing or a Cloud Computing paradigm, giving two reasons for your choice. [**3 Marks**]

j) Design the core interface for a simple distributed, object-oriented "BankAccount" service using RMI principles. The interface should support at least three key methods. Justify your choice of methods. [**4 Marks**]

k) The Centralized, Distributed, and Token-Passing algorithms are three classic approaches to achieving mutual exclusion. Compare and contrast these three algorithms based on their message overhead, susceptibility to failure, and potential for creating a bottleneck. [**4 Marks**]

l) "Transparency in distributed systems is an ideal, not always a practical goal." Critically evaluate this statement. Discuss at least two types of transparency where the benefits might be outweighed by the performance or complexity costs in a large-scale system. [**4 Marks**]

QUESTION TWO (20 MARKS)

GlobalFinance is a multinational bank designing a new distributed database system, the "Transaction Hub," to process millions of international money transfers daily. The system must be highly available (24/7) and guarantee that transactions are never lost or duplicated, even in the event of server failures, network partitions, or data center outages. The bank has decided to use a Primary-Backup Replication strategy for critical data. However, the engineering team is debating the finer points of the implementation. They are particularly concerned about the trade-offs between consistency, availability, and performance during failure scenarios.

m) During a network partition, the Primary replica for an account database becomes isolated from its Backup replicas but remains accessible to some clients.

(i) Analyze the potential risks if the Primary continues to process withdrawal transactions. [**2 Marks**]

(ii) Evaluate the trade-offs of two different strategies the system could employ in this scenario: (i) continuing to operate, or (ii) automatically shutting down. [**2 Marks**]

(iii) Based on your analysis, which strategy would you recommend for GlobalFinance? Justify your choice with specific reference to the banking context. [**1 Mark**]

n) The team is considering switching from a Primary-Backup model to a Multi-Primary replication system to improve write performance.

(i) Explain one key advantage of a Multi-Primary system for a global bank. [**1 Mark**]

(ii) Synthesize your knowledge of distributed transactions to identify a significant consistency challenge that would be introduced by this change, using a concrete example of two concurrent transactions (e.g., transferring funds from the same account in two different geographic regions). [**3 Marks**]

(iii) Name a concurrency control technique that could be used to mitigate this challenge. [**1 Mark**]

o) "Ultimately, for a critical system like GlobalFinance's, strong consistency must be prioritized over low latency and high availability during partitions." Critically evaluate this statement by discussing the following:

(i) Explain what is meant by "strong consistency" in the context of the CAP theorem. [**2 Marks**]

(ii) Argue for the statement, explaining why strong consistency is non-negotiable for a financial institution. Use the risk of financial loss or regulatory penalties in your argument. [**3 Marks**]

(iii) Argue against the statement, explaining the real-world business impact (e.g., customer dissatisfaction, lost revenue) of choosing consistency over availability during a network partition. [**3 Marks**]

(iv) Provide a final, justified conclusion on whether you believe the engineers at GlobalFinance should configure their system to be a CP (Consistent/Partition-tolerant) or an AP (Available/Partition-tolerant) system according to the CAP theorem. [**2 Marks**]

QUESTION THREE (20 MARKS)

socialSphere is a massive social media platform with data centers worldwide. A user's "feed" is generated by aggregating posts from their friends, which can be located on any server. A critical feature is the "Like" counter on posts. The company is facing a problem: during viral events, millions of users concurrently like the same post (e.g., a celebrity's tweet or a breaking news story), resulting in inconsistent like counts across different users' views. Some users experience erratic jumps in the count, while others observe a decrease in the count. The engineering team is evaluating various consistency models and concurrency control mechanisms to address this issue without compromising the system's performance.

p) The team is debating the appropriate consistency model for the "Like" counter.

(i) Define Strong Consistency (Linearizability) and Eventual Consistency in the context of the Like counter. [**2 Marks**]

(ii) Synthesize an argument for using Eventual Consistency. Your argument should reference the CAP theorem and explain the benefits for system availability and user-perceived latency. [**3 Marks**]

(iii) Synthesize an argument for using Strong Consistency. Your argument should focus on the user experience and the perceived "correctness" of the platform, especially for a feature that implies a precise count. [**3 Marks**]

(iv) As a system architect, which model would you implement for the Like counter, and which would you implement for the actual list of users who liked a post? Justify this nuanced decision. [**2 Marks**]

q) To avoid the read-modify-write cycle entirely, a senior engineer proposes an alternative paradigm.

(i) Explain how the "Like" operation could be modeled not as an update to a counter, but as an append-only event (e.g., adding a "like event" to a log). [**1 Mark**]

(ii) Create a high-level design for this event-sourcing approach. Describe how the total Like count is derived from the event log. [**2 Marks**]

(iii) Identify one key advantage of this approach regarding concurrency and one challenge concerning query performance when generating the count. [**2 Marks**]

r) A product manager argues: "The user doesn't care if the count is off by a few for a split second; they just want the site to be fast. Therefore, we should always prioritize availability and latency over strong consistency for this feature." Critically evaluate this statement. In your response, discuss whether there is a point at which inconsistency becomes a business logic failure rather than just a temporary UI glitch, considering features like "Top Liked Posts of the Day" or paid promotions based on Like milestones. [**5 Marks**]

QUESTION FOUR (20 MARKS)

The highly anticipated Furya FC vs. Gorlin FC championship match led to a catastrophic failure of the official online ticketing portal, "TicketiGo." The system, designed to handle 10,000 concurrent users, was overwhelmed when over 500,000 fans attempted to purchase 50,000 available tickets the moment sales opened.

The Failure:

- The web servers became unresponsive within seconds, displaying HTTP 503 (Service Unavailable) errors.
- Many users saw a "Payment Successful" page but never received a ticket confirmation email, suggesting a failure after payment processing.
- The "Available Tickets" counter showed inconsistent numbers, sometimes going negative.
- Reports emerged of the same ticket ID being allocated to multiple users.

The public and media have condemned the system as a complete failure. You are a distributed systems consultant brought in to conduct a post-mortem and design a robust solution for the future.

a) Based on the described failure, analyze the likely architectural shortcomings of the "TicketiGo" system.

(i) Identify three specific points of failure in a typical 3-tier web architecture (Web Server, Application Server, Database) that could have caused the observed symptoms (e.g., 503 errors, inconsistent ticket count). [**3 Marks**]

(ii) For a re-design, propose and justify the use of a microservices architecture for the ticketing system. Specifically, suggest how you would decouple the core functions into separate, scalable services. [**4 Marks**]

(iii) Explain how this microservices design would help prevent a total system collapse, even if one service (like Payment Processing) becomes a bottleneck. [**3 Marks**]

b) A technical lead suggests using an "artificial delay" or a "virtual waiting room" that randomly admits users, arguing that "it's better to randomly fail fast for 90% of users than to have the entire system fail for everyone." Critically evaluate this strategy from both a technical and an ethical perspective. Is deliberately throttling user access a valid engineering solution for managing extreme load, or is it an unfair abdication of the responsibility to build a scalable system? Justify your opinion. [**5 Marks**]

s) The core problem is preventing the overselling of tickets (selling more than 50,000). The team suggests using a distributed queue (e.g., Apache Kafka or RabbitMQ). Design the ticket purchase flow using this queue. Explain how a user's request is processed and how the queue ensures that exactly 50,000 tickets are allocated, even under massive concurrency. [**5 Marks**]

QUESTION FIVE (20 MARKS)

M-Kopa Bank, a leading mobile-based bank in Kenya, is launching a new "Super Wallet" that will allow near-instant, low-cost micro-transactions across East Africa. To achieve the required scale, they are moving from a single, centralized ledger database to a sharded architecture, where customer accounts are distributed across multiple, independent database clusters (shards). The critical requirement is that any transaction moving funds between shards (e.g., a user on Shard A pays a merchant on Shard B) must be atomic and consistent. It must never happen that money is deducted from Shard A but not credited to Shard B, or vice versa. The bank cannot tolerate even a single transaction being lost or duplicated during a network partition or server failure.

t) The initial design employs a simple two-phase commit (2PC) protocol, coordinated by a "Transaction Manager," for cross-shard transactions.

(i) Describe the regular operation of 2PC for a cross-shard payment from Shard A to Shard B. [**3 Marks**]

(ii) Analyze a specific failure scenario: The Transaction Manager crashes after all shards have voted "YES" but before it sends the final "COMMIT" decision. Explain the problem this creates. What is the state of the transaction on each shard, and why is this a critical failure for the bank? [**4 Marks**]

(iii) Explain why the existence of this "blocking" problem in 2PC makes it unsuitable for a highly available system and necessitates a fault-tolerant consensus algorithm like Raft or Paxos for managing critical state. [**3 Marks**]

u) A junior engineer proposes: "To make it faster, we should run a separate Raft consensus cluster for each pair of shards, instead of one global cluster for all transactions."Critically evaluate this proposal.

(i) Identify one potential performance benefit of this decentralized approach. [**1 Mark**]

(ii) b) Identify two significant consistency, complexity, or operational drawbacks that make this a poor design choice for a financial system [**2 Marks**]

(iii) c) Justify why a single, strongly consistent global transaction log is preferable, despite being a potential bottleneck. [**2 Marks**]

v) During a major network outage that partitions the M-Kopa data center hosting the Raft leader from the rest of the clusters, the system must make a choice.

(i) According to the CAP theorem, what two properties can the system choose between during this partition? [**1 Mark**]

(ii) b) As the Site Reliability Engineer (SRE), you must choose one of the following actions. Justify your choice and explain the consequences.

* Option A: Keep the existing Raft leader running and accepting writes, sacrificing availability for the partitioned nodes.

* Option B: Initiate a new leader election within the majority partition, risking a "split-brain" scenario if the old leader is still active. [**3 Marks**]

(iii) c) For a mobile banking wallet, why is consistency almost always the correct choice to sacrifice during a partition? What is the business risk of choosing consistency instead?

[**1 Mark**]