Group Project Title:

Design and Evaluation of a Scalable Distributed System with Process Management, Naming, and Data Consistency Mechanisms

Project Objectives:

Students will:

1. **Apply** key concepts of distributed processes, naming systems, and data consistency mechanisms in a prototype system.
2. **Analyze** the behavior and trade-offs of different design approaches in real or simulated environments.
3. **Evaluate and recommend** solutions based on system performance, scalability, and correctness.

Project Description:

Your team is tasked with designing and simulating a distributed system that manages:

- **Processes** (e.g., threads, virtualized services, or mobile code),
- **Naming and resource resolution** (flat vs. structured vs. attribute-based),
- **Replicated data with consistency models** (sequential, eventual, or client-centric).

You will develop a conceptual architecture, simulate core components using any preferred programming language (e.g., Python/Java), and analyze your design's trade-offs in scalability, fault tolerance, and consistency.

Suggested Deliverables:

1. System Design Document (A4)

- Describe system architecture: process roles, communication methods, and naming structure.
- Explain your choices for name resolution and data replication strategy.
- Use diagrams to visualize process/naming/data interactions.

2. Prototype or Simulation (A3)

- Demonstrate basic process handling (e.g., multithreading or migration).
- Simulate naming and lookup resolution across nodes.
- Simulate data replication with configurable consistency models (e.g., eventual vs. sequential).

3. Analysis Report (C4)

- Analyze trade-offs: structured vs. flat naming, thread vs. process-based design, strong vs. eventual consistency.
- Evaluate system performance under varied scenarios (e.g., node failure, high concurrency).
- Use charts/tables to compare latency, consistency violations, lookup success rates, etc.

4. Presentation & Demonstration

- Present system design, simulation results, and key takeaways.
- Each team member must explain a distinct component (e.g., process model, naming, replication logic).

Roles Suggestion (for 5 students):

1. **Process & Thread Model Developer**
2. **Naming System Architect**
3. **Consistency & Replication Engineer**
4. **System Integrator & Tester**
5. **Report Analyst & Presenter**

Taxonomy Mapping

- **A3 (Application):** Implementation of system features (threads, naming, replication)
- **C4 (Analysis):** Design decisions and trade-off reasoning
- **A4 (Analysis):** Teamwork integration and analytical clarity communicated through slide or video presentation.

Group Project Rubric (50 Marks)

Component	Marks	Criteria
System Design Document	10	Clear architectural explanation, appropriate use of diagrams, and rationale for design decisions. Must include process structure, naming approach, and consistency strategy.
Prototype / Simulation	15	Working simulation or partial implementation with at least: multithreaded behavior, simulated name resolution, and replicated data with consistency behavior.
Analysis Report	10	Well-structured analysis of trade-offs (e.g., process vs. thread, consistency models), includes graphs/tables, and demonstrates understanding of theoretical underpinnings.
Presentation &	5	Clear division of roles, logical presentation flow, working

Component	Marks	Criteria
Demo		demo or mock outputs, Q&A readiness.
Code Quality & Documentation	5	Modular, readable code with inline comments and a brief README explaining how to run/test the simulation.
Team Collaboration	5	Peer evaluation or teacher observation of teamwork dynamics and individual contributions.

Technologies & Tools Recommendation

Aspect	Suggested Tools / Technologies
Simulation Language	**Python** (easy thread/process modules), **Java** (strong object-oriented modeling), or **Node.js** for async sim.
Thread & Process Model	`threading` and `multiprocessing` (Python); `ExecutorService` in Java
Naming Simulation	Implement flat and structured naming manually; optionally simulate DNS using Python `dict` or Java `Map`
Consistency Simulation	Custom logic to simulate sequential, eventual, or client-centric consistency (see below)
Visualization (Optional)	**Graphviz**, **Matplotlib**, **Mermaid.js**, or **Draw.io** for architectural diagrams and event timelines
Code Sharing & Versioning	GitHub or GitLab
Collaboration	Google Docs, Trello, Slack or Discord

Example Features to Simulate

Process/Thread Management

- Simulate 2–3 processes or services that communicate or perform tasks (e.g., sleep/wake).
- Include multithreading to show parallel execution and shared data access.

```
# Python threading example
from threading import Thread, Lock
import time

shared_data = 0
lock = Lock()

def update_data(name):
```

```
    global shared_data
    for _ in range(5):
        time.sleep(1)
        with lock:
            shared_data += 1
            print(f"{name} updated shared_data to {shared_data}")

Thread(target=update_data, args=("P1",)).start()
Thread(target=update_data, args=("P2",)).start()
```

Naming Model

- Implement flat naming using a dictionary/hash table.
- Simulate structured naming like DNS using hierarchical keys.

```
# Example: DNS-like resolution
name_table = {
    "service1.university.edu": "192.168.1.10:5000",
    "service2.university.edu": "192.168.1.11:5001"
}
```

Replication & Consistency Models

- Simulate **Sequential Consistency** by keeping a central log of operations and ensuring all nodes apply updates in the same order.
- For **Eventual Consistency**, introduce delays and allow nodes to catch up.
- For **Client-Centric Consistency**, each client tracks its last seen version and applies updates accordingly.

```
# Eventual consistency (simplified)
replicas = {"A": 1, "B": 1}

def update_replica(replica_id, value):
    time.sleep(2 if replica_id == "B" else 0)  # Simulate delay
    replicas[replica_id] = value
    print(f"Replica {replica_id} updated to {value}")

# Simulate write
Thread(target=update_replica, args=("A", 2)).start()
Thread(target=update_replica, args=("B", 2)).start()
```

Optional Enhancements for Bonus Mark

- Simulate **process migration** with state transfer between threads or hosts.
- Introduce **DHT-like name resolution** with hashed node identifiers.
- Compare multiple consistency models with performance metrics (latency, stale reads).

Assessment Rubrics:

1) Report: CLO1, C4, MQF2 (10%)

A	B	C	D	E	F	G	H
Taxonomy Level	**Criteria**	**Weight (Marks)**	**Performance Indicators**	**Exceeds Expectations (100%)**	**Meets Expectations (75%)**	**Needs Improvement (50%)**	**Likert Scale**
C4: Analysis	Design Justification & Trade-Offs	10	Explains and supports design decisions with context and reasoning.	Detailed analysis and strong justification with references.	Provides justification with some reasoning.	Superficial or unclear reasoning behind design choices.	1 - Poor, 2 - Fair, 3 - Good, 4 - Very Good, 5 - Excellent
C4: Analysis	Comparative Evaluation	5	Analyzes and compares alternatives with supporting evidence.	Clear, in-depth comparison using quantitative and qualitative criteria.	Basic comparison of alternatives with some clarity.	Poor or missing comparison of alternatives.	1 - Poor, 2 - Fair, 3 - Good, 4 - Very Good, 5 - Excellent
C4: Analysis	Interpretation of Simulation Results	5	Uses simulation data to provide insightful interpretations.	Provides deep insights from accurate, relevant results.	Interprets results with some insight.	Little or unclear interpretation of results.	1 - Poor, 2 - Fair, 3 - Good, 4 - Very Good, 5 - Excellent

2) Report: CLO2, A3, MQF3d (20%)

A	B	C	D	E	F	G	H
Taxonomy Level	**Criteria**	**Weight (Marks)**	**Performance Indicators**	**Exceeds Expectations (100%)**	**Meets Expectations (75%)**	**Needs Improvement (50%)**	**Likert Scale**
A3: Application	Functional Prototype / Simulation	10	Implements distributed features with processes/threads, naming and replication.	All components implemented with advanced logic, fully working simulation.	Most components implemented, simulation runs with minor bugs.	Partial implementation or simulations that do not fully run.	1 - Poor, 2 - Fair, 3 - Good, 4 - Very Good, 5 - Excellent
A3: Application	Use of Concepts in Code	5	Applies correct concepts in simulation or pseudocode.	Concepts clearly and effectively applied across codebase.	Basic application of core concepts in code.	Some attempt to apply concepts but with gaps or errors.	1 - Poor, 2 - Fair, 3 - Good, 4 - Very Good, 5 - Excellent
A3: Application	System Architecture Design	5	Well-structured diagram or textual description of system design.	Design is modular, well-annotated with diagrams.	Good structure with some design rationale and diagrams.	Design lacks clarity or justification.	1 - Poor, 2 - Fair, 3 - Good, 4 - Very Good, 5 - Excellent

3) Presentation: CLO3, A4, MQF3f (10%)

A	B	C	D	E	F	G	H
Taxonomy Level	**Criteria**	**Weight (Marks)**	**Performance Indicators**	**Exceeds Expectations (100%)**	**Meets Expectations (75%)**	**Needs Improvement (50%)**	**Likert Scale**
A4: Analysis / Presentation	Clarity and Structure of Slide/Video Presentation	3	Presentation is clear, logically structured, and easy to follow with appropriate pacing.	Well-organized and professional presentation with smooth transitions, timing, and layout.	Mostly clear and structured with minor issues in pacing or flow.	Disorganized or unclear presentation; difficult to follow.	1 - Poor, 2 - Fair, 3 - Good, 4 - Very Good, 5 - Excellent
A4: Analysis / Presentation	Integration of Team Member Contributions	4	Each team member's contribution is clearly integrated and represented; the system is presented as a cohesive whole.	Teamwork is seamless; all components and members are well-represented and connected.	Most contributions are well integrated; minor imbalance or disconnect between parts.	Teamwork integration is weak or uneven; some parts poorly connected or unexplained.	1 - Poor, 2 - Fair, 3 - Good, 4 - Very Good, 5 - Excellent
A4: Analysis / Presentation	Analytical Explanation of System Design and Results	3	System behavior, performance, and trade-offs are explained analytically using visuals or spoken narration.	Excellent use of data/visuals; deep insights and analysis of system results are clearly communicated.	Good interpretation of system behavior with adequate visuals and discussion.	Superficial or unclear analysis; limited use of visuals or evidence.	1 - Poor, 2 - Fair, 3 - Good, 4 - Very Good, 5 - Excellent