Computer Science and Engineering Department

NATIONAL INSTITUTE OF TECHNOLOGY

DELHI

Session (2024-2026)

Project Synopsis

"Advanced Databases"

(CSBM 503)

Submitted To:
Dr. Parnika
Computer Science and
Engineering Department

Submitted By:
Joshua Joy
(242211009)
Manchuri Reddy Dhamarakesh
(242211011)
Palaji Prudhvi Raj
(242211014)
MTech (1st semester)
Computer Science & Engg.
(Analytics)

1. **Title:**
Multi-Database Distributed Inventory Management System for Real-Time Stock Updates

2. **Objective:**
To develop a robust and scalable inventory management system that effectively manages inventory data across multiple geographically dispersed locations, ensuring real-time updates, data consistency, and efficient access for stakeholders**.**

3. **Purpose:**
To address the challenges of managing inventory in a distributed environment, where traditional centralized systems are insufficient for real-time stock updates and efficient information sharing. To enable businesses to optimize their inventory management processes, reduce stockouts, improve order fulfillment, and gain valuable insights into inventory trends and performance.

4. **Methodology:**
   - Multi-Database Architecture: Employ a hybrid approach using a central MySQL database for core inventory data and MongoDB for distributed, location-specific inventory updates.
   - Data Synchronization: Leverage Apache Kafka as a message queue to facilitate real-time data synchronization between MySQL and MongoDB, ensuring data consistency across locations.
   - API Integration: Develop RESTful APIs for managing inventory data, allowing integration with warehouse management systems, point-of-sale systems, and other applications.

5. **Inventory Management Features:**
   - Product Management: Create, update, and manage product information.
   - Stock Management: Track stock levels, receive real-time updates, and manage stock adjustments.
   - Location Management: Define locations, assign stock to locations, and track location-specific stock levels.
   - Order Management: Process orders, track order fulfillment, and manage stock allocation.
   - User Interface (Optional): A user-friendly interface (web-based or mobile) for visualizing inventory data, managing inventory operations, and generating reports.

6. Technology Used:

Central Database: MySQL Distributed
Database: MongoDB Data
Synchronization: Apache Kafka API
Development: Spring Boot (Java)
Development Tools: IDE (e.g., IntelliJ IDEA), Git for version control

7. System Requirements:

Hardware: Server/system with sufficient RAM and storage capacity for databases and application server.

Software:

- ✓ MySQL database server
- ✓ MongoDB database server
- ✓ Apache Kafka messaging queue
- ✓ Java Development Kit (JDK)
- ✓ Spring Boot framework Development tools (IDE, Git)

8. Summary:

- The proposed system provides a comprehensive and flexible approach to managing distributed inventory, enabling businesses to optimize stock levels, streamline operations, and gain valuable insights into inventory performance.
- The system's modular design allows for easy scalability and adaptation to changing business requirements.
- Future development can focus on adding advanced features like inventory analytics, automated ordering, and integration with third-party applications.

9. References:

- ✓ **Apache Kafka for Real-Time Data Synchronization:** Apache Kafka is widely used for real-time data synchronization in distributed environments. It acts as a message broker, ensuring efficient data flow between systems like MySQL and MongoDB. **Source:** Confluent
- ✓ **Real-Time Inventory Management Benefits:** Real-time inventory systems help businesses maintain optimal stock levels, improve customer satisfaction, and reduce costs through automated, data-driven decisions. **Source:** Visiwise Blog
- ✓ **Scalability and Data Consistency:** Redis Enterprise provides an example of how real-time systems can scale and maintain consistency across distributed databases, making it a good reference for ensuring data reliability and fast access. **Source:** Redis Enterprise