SIT314/SIT729 – Week 8 Group Activity  
Designing Microservices

short line

# Overview

# In this activity, you will design **`An automatic purchasing, stock management and delivery system for supermarkets`** using an event-driven and microservice approach. This system needs to allow people to purchase products from a physical supermarket, manage the inventory, manage staff scheduling and manage restocking.

# Tasks

The aim of the task is to identify the microservices needed to implement this system. Answer the following to design your system.

1. What physical sensor and actuator infrastructure is needed in the supermarket?

-RFID Tags and Readers: Attached to products and shopping carts for real-time inventory tracking and customer purchases.

-Weight Sensors: Placed on shelves to monitor stock levels and detect when items are removed or replenished.

-Barcode/QR Code Scanners: Used at self-checkout counters and for stock intake verification.

-Motion Sensors: To detect customer presence in aisles and adjust staff deployment or lighting.

-Cameras with Computer Vision: For theft detection, customer flow monitoring, and shelf stock image analysis.

-Temperature & Humidity Sensors: For perishable goods like dairy, meat, or frozen items.

-Smart Shelves: Equipped with sensors to detect product type, stock quantity, and shelf conditions.

-Robotic Arms or Conveyor Belts: For automatic product sorting, packing, or delivery in warehouse or backend storage.

1. What Edge Computing infrastructure is needed in the supermarket?

-Local Edge Servers: Installed in the store to collect, process, and temporarily store data from physical devices before syncing to the cloud.

-Edge Gateways: Used to connect and filter data from multiple IoT sensors, handling protocol translation and security.

-AI/ML Inference Units: On-site processors that run lightweight ML models for tasks such as image recognition (e.g., checking empty shelves or misaligned products).

-Data Caching & Aggregation Modules: To enable fast access to sensor data without latency from cloud calls.

-Secure Network Routers & Firewalls: To isolate edge network and ensure secure device communication.

-Redundancy Systems: Backup power and mirrored systems to ensure fault tolerance during local outages.

1. What are the microservices needed for the system?

-Customer Purchase Service – Handles transactions, shopping cart data, and checkout processes.

-Inventory Management Service – Tracks stock levels, shelf status, and product availability.

-Restocking Service – Automates ordering from suppliers and updates when stock is replenished.

-Staff Scheduling Service – Manages rosters, shift changes, and allocation based on store traffic.

-Delivery and Logistics Service – Coordinates internal and external deliveries.

-Alert and Notification Service – Sends alerts for low stock, anomalies, or staff notifications.

-Product Catalog Service – Stores product metadata, pricing, expiry, and promotions.

-Sensor Data Aggregator – Ingests and preprocesses real-time sensor streams from edge devices.

-Billing and Payment Service – Processes transactions securely and handles refunds.

-Analytics and Reporting Service – Generates reports on sales trends, staff efficiency, and customer behavior.

1. Thinking about 3 distinct microservices, sketch the pseudocode for the event processing loop.

1. Inventory Management Service  
while True:

event = listen\_for("StockUpdateEvent")

if event.type == "PRODUCT\_REMOVED":

update\_inventory(event.product\_id, -1)

if inventory[event.product\_id] < threshold:

trigger\_event("RestockNeededEvent", event.product\_id)

Customer Purchase Service

while True:

event = listen\_for("CheckoutEvent")

if validate\_cart(event.cart\_id):

process\_payment(event.user\_id, event.total)

trigger\_event("StockUpdateEvent", event.cart\_items)

trigger\_event("ReceiptEvent", event.transaction\_id)

Staff Scheduling Service  
while True:

event = listen\_for("TrafficUpdateEvent")

if event.traffic\_level > high\_threshold:

assign\_extra\_staff(department=event.department)

elif event.traffic\_level < low\_threshold:

reduce\_staff(department=event.department)

1. Explore which microservices are critical to the performance of the application.

-Inventory Management Service: Core to stock awareness and critical for avoiding out-of-stock scenarios.

-Customer Purchase Service: Directly affects user experience and transaction completion.

-Sensor Data Aggregator: Must run in real time to ensure all downstream services (inventory, alerts) get accurate data.

-Restocking Service: Ensures seamless shelf replenishment to prevent loss of sales.

Alert and Notification Service: Crucial for operational awareness (e.g., perishable stock expiring soon).

1. What microservices would you choose to scale, and what would be the triggers to scaling?

1. Customer Purchase Service

Trigger: High checkout concurrency, holiday rush.

Scaling: Horizontal scaling using container orchestration (e.g., Kubernetes).

2. Sensor Data Aggregator

Trigger: Increased number of IoT devices or sensors being added.

Scaling: Streamlined with edge processing and asynchronous message queues (e.g., Kafka, MQTT).

3. Analytics and Reporting Service

Trigger: Demand for real-time insights across multiple branches.

Scaling: Migrate to cloud-based big data platforms for distributed processing.