CA-S

Assignment -2

One-Page Draft Scenario Combining Microservices Architecture, Event-Driven Architecture, SOLID, DRY, and KISS Principles

Scenario: E-Commerce Platform – "ShopSwift":

"ShopSwift" is a modern, cloud-based e-commerce platform designed to handle the sale of electronics, fashion, and home essentials. To ensure high scalability, flexibility, and maintainability, the platform is built using a **Microservices Architecture**, powered by an **Event-Driven Architecture (EDA)**. It also strictly adheres to software engineering best practices like **SOLID principles**, and design philosophies like **DRY** (Don't Repeat Yourself) and **KISS** (Keep It Simple, Stupid).

Microservices Architecture in Practice:
The system is broken down into loosely coupled services, each responsible for a specific business capability:

- **User Service**: Manages user registration, login/authentication, and profile management.
- **Product Catalog Service**: Handles product listings, descriptions, categories, and search functionality.
- **Order Service**: Takes care of order placement, tracking, and maintaining order history.
- **Payment Service**: Processes customer payments and generates invoices.
- **Inventory Service**: Updates product stock levels in real-time.
- **Notification Service**: Sends email or SMS alerts for order-related events such as confirmation, dispatch, and delivery.

Each of these services operates independently, making it easier to scale and deploy without affecting others.

Event-Driven Architecture in Action:
In "ShopSwift", services communicate through events instead of direct calls, enabling asynchronous workflows and reducing tight coupling:

- When a user places an order, the **Order Service** emits an OrderPlaced event.
- The **Inventory Service** subscribes to this event and updates the stock count.
- The **Payment Service** listens and begins processing the payment.
- Once the payment is successful, a PaymentConfirmed event is published.
- The **Notification Service** responds to these events and sends appropriate messages to the user.

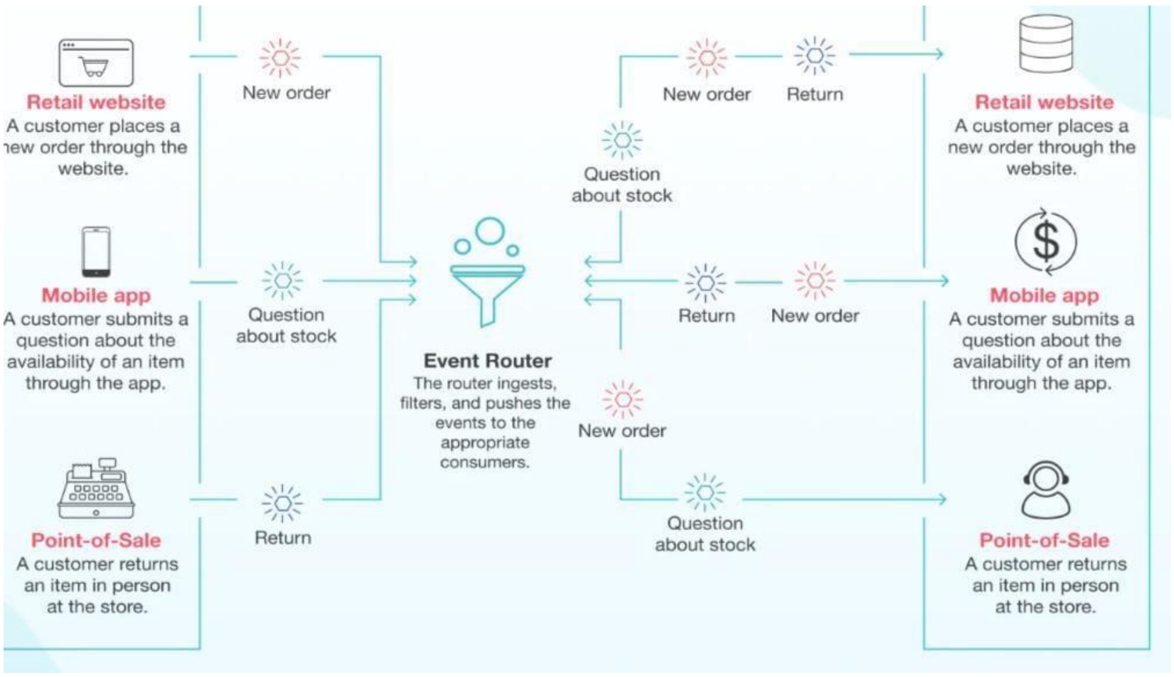
This model ensures better performance under load and easier integration of new features.

Applying SOLID Principles:

- **S (Single Responsibility)**: Each service handles one function only, e.g., the **Payment Service** is solely focused on transactions.
- **O (Open/Closed)**: Services are built to allow new features through events (e.g., adding a **Loyalty Service** that listens for OrderCompleted) without changing existing code.
- **L (Liskov Substitution)**: Interfaces like NotificationSender can easily switch between email or SMS without breaking anything.
- **I (Interface Segregation)**: Services only expose what is necessary. For example, a PaymentService interface may only offer pay() to general clients, while admin clients may use additional features.
- **D (Dependency Inversion)**: High-level services depend on abstractions like INotifier, allowing flexibility in swapping implementations without rewriting the core logic.

Following DRY and KISS Principles:

- **DRY (Don't Repeat Yourself)**:
  - All services use a shared **logging system**.
  - Common functionality like **input validation** and **error handling** is extracted into reusable libraries.
  - API contracts are defined centrally using OpenAPI to avoid duplication.
- **KISS (Keep It Simple, Stupid)**:
  - Each service does only what it's supposed to do—nothing more.
  - Events follow a uniform, easy-to-understand schema.
  - Services avoid shared databases or complex interdependencies, communicating strictly through events.



Retail website
A customer places a new order through the website.
Mobile app
A customer submits a question about the availability of an item through the app.
Point-of-Sale
A customer returns an item in person at the store.
New order
Question about stock
Return
Event Router
The router ingests, filters, and pushes the events to the appropriate consumers.
New order
Return
Question about stock
Return
New order
New order
Question about stock
Retail website
A customer places a new order through the website.
Mobile app
A customer submits a question about the availability of an item through the app.
Point-of-Sale
A customer returns an item in person at the store.

Figure 1: Microservices Design Principal (source: simform)

Name: Anil Gupta
Reg. No.: 2141007073