Assignment 2:

Principles in Practice - Draft a one-page scenario where you apply Microservices Architecture and Event-Driven Architecture to a hypothetical e-commerce platform. Outline how SOLID principles could enhance the design. Use bullet points to indicate how DRY and KISS principles can be observed in this context.

Microservice Architecture:

A microservice architecture is a type of application architecture where the application is developed as a collection of services.

Where do you apply Microservices Architecture to a hypothetical e-commerce platform

- Managing product information, including products, categories and attributes.
- Handles order processing, including order creation and status updates.
- Manages user authentication, authorisation and profile information.
- Provides personalized product recommendations based on user behaviour and preferences.

Event-Drive Architecture:

Event-Drive architecture is a software design pattern that enables an organization to detect "events" or important business moments and act on them in real-time or near real time.

Where do you apply Event-Drive Architecture to a hypothetical e-commerce platform

- Generated when a user successfully places an order. This event triggers downstream processes like inventory deduction, order confirmation mail, and payment processing.
- Indicates the outcome of a payment transaction. On successful payment, it triggers the fulfilment process on failure, it initiates a retry mechanism.
- Notifies the order management system and users about shipping status changes, such as shipped, out for delivery, or delivered.

Solid principles:

Single Responsibility Principle:

Each microservice has a single responsibility, ensuring it is focused and easy to maintain.

Open/Closed Principle:

Services are open for extension (adding new features) but closed for modification (existing functionalities remain intact).

Interface Segregation Principle:

Service interfaces are customized to the needs of their clients, avoiding unnecessary dependencies.

How DRY and KISS principles can be observed in this context

DRY (Don’t repeat yourself**):**

Shared components like authentication and database connections are centralized to avoid duplication and inconsistences across service.

KISS(Keep it simple stupid**):**

Service is designed to be simple and focused on there core functionalities, avoiding unnecessary complexities that could delay development and maintenance.