Read the following scenario and answer the question.

A university management system provides a single "Student Services" interface to external applications. Internally, this interface coordinates multiple complex subsystems, including course registration, fee payment, transcript generation, and library access. External clients should not directly interact with these subsystems, nor understand their internal workflows. The goal is to simplify usage while reducing coupling between clients and the underlying system components.

Q1. Which design pattern will be most applicable in this scenario? Justify your answer.

Read the following scenario and answer the question.

A real-time collaboration platform includes multiple UI components such as a chat panel, notification system, user list, and video controller. These components must react to each other's state changes (e.g., muting a user updates notifications and UI status). Direct communication between components has become hard to maintain. The system requires a centralized coordination mechanism, so components interact indirectly and remain loosely coupled.

Q2. Write a pseudocode for the most suitable design pattern for the given scenario.

Read the following code and answer the question.

```
class Vehicle {
    public void startEngine() {
        System.out.println("Starting engine");
    }

    public void fly() {
        System.out.println("Flying");
    }
}

class Car extends Vehicle {

    public void operate(String mode) {
        if (mode.equals("START")) {
            startEngine();
            System.out.println("Car is moving");
        } else if (mode.equals("STOP")) {
            System.out.println("Stopping car");
            System.out.println("Car stopped");
        } else if (mode.equals("PARK")) {
            System.out.println("Stopping car");
            System.out.println("Car parked");
        }
    }
}
```

Q3. Identify the 3 code smells present in this code. Justify your answers with proper explanations.

Read the following code and answer the question.

An e-commerce platform must independently scale its user management, payment processing, inventory, and recommendation services while allowing different teams to deploy updates without impacting the entire system. The application is expected to handle high traffic, frequent feature releases, and integration with multiple third-party services.

Q4. Which architectural design between MVC and microservice architecture is better for the given scenario?

Q5. Differentiate between Layer and client-server architecture.

Read the following code and answer the question.

A smart city traffic management system continuously receives events from road sensors, traffic lights, and surveillance cameras. Each event, such as congestion detection or accident alerts, must trigger independent actions like rerouting traffic, notifying authorities, and updating public dashboards. The system must process events asynchronously, remain loosely coupled, and scale dynamically as new event producers and consumers are added.

Q6. Design the system architecture for the given application following 1 or more architectural patterns.

Q7. Draw an Architectural Context Diagram for the given scenario.