Assignment 1, set-2

Student Name: Khushi Khemka **UID: 23BCS10652**
Branch: BE-CSE **Section/Group: Krg_3A**
Semester: 6th **Subject Name: System Design**
Subject Code: 23CSH-314

Question1: Explain SRP and OCP in detail with proper examples.

Answer1: SOLID is a set of five important principles of object-oriented programming that are used to design software in a clean, maintainable, and scalable way. These principles help developers write code that is easy to understand, modify, and extend. The word SOLID is an acronym where each letter represents a specific design principle.
In SOLID, the letter 'S' stands for Single Responsibility Principle (SRP), which states that a class should have only one responsibility or one reason to change. The letter 'O' stands for Open–Closed Principle (OCP), which states that software entities should be open for extension but closed for modification.

- **S – Single Responsibility Principle (SRP)**
  The **Single Responsibility Principle (SRP)** states that a class should have **only one responsibility or one reason to change**. This means a class should focus on performing a single task. SRP helps in keeping the code simple, understandable, and easy to maintain.

Example and Code:

A Student class that manages student data, calculates marks, and generates reports violates SRP because it performs multiple tasks.

```
class Student {
public:
    void addStudent() { }
    void calculateMarks() { }
    void generateReport() { }
};
```

To follow SRP, these responsibilities should be separated into different classes.

```
class Student {
public:
    void addStudent() { }
};

class MarksCalculator {
public:
    void calculateMarks() { }
};

class ReportGenerator {
public:
    void generateReport() { }
};
```

Advantages of Single Responsibility Principle (SRP)

1. It improves code readability and maintainability because each class has a clear and focused purpose.
2. It makes testing and debugging easier, as changes in one responsibility do not affect others.

Disadvantages of Single Responsibility Principle (SRP)

1. It may lead to an increase in the number of classes, which can make the project structure complex.
2. It is sometimes difficult to identify a single responsibility, especially in large or real-world systems.

- **O – Open–Closed Principle (OCP)**

  The Open–Closed Principle (OCP) states that software entities like classes, modules, and functions should be open for extension but closed for modification. This means you can add new functionality without changing existing code, which reduces the risk of introducing bugs. OCP is usually achieved using inheritance, polymorphism, or interfaces.

  Example and Code:
  A class calculating areas of shapes can violate OCP if we need to modify it every time a new shape is added.

```
// Violation of OCP
class Shape {
public:
    double area(string shapeType) {
        if(shapeType == "Circle") return 3.14 * 5 * 5;
        else if(shapeType == "Rectangle") return 10 * 5;
        return 0;
    }
};
```

To follow OCP, we use inheritance and polymorphism:

```
class Shape {
public:
    virtual double area() = 0; // open for extension
};

class Circle : public Shape {
public:
    double area() { return 3.14 * 5 * 5; }
};

class Rectangle : public Shape {
public:
    double area() { return 10 * 5; }
};
```

Now, new shapes can be added by creating new classes, without modifying existing ones.

Advantages of Open–Closed Principle (OCP)

1. Easy to extend functionality without touching existing code.
2. Reduces the risk of bugs because existing tested code remains unchanged.

Disadvantages of Open–Closed Principle(OCP)

1. May require more classes and interfaces, increasing system complexity.
2. Can be hard to design initially because you need to anticipate future extensions.

Question2: Discuss in detail about the vioations in SRP and OCP along with their fixes.

Answer2:

- **Violations in Single Responsibility Principle (SRP) and Their Fixes**

Violation of SRP

A class violates SRP when it handles more than one responsibility. This makes the class hard to maintain, test, and modify. For example:

```
class Student {
public:
    void addStudent() { }
    void calculateMarks() { }
    void generateReport() { }
};
```

Problem:
Here, the Student class has **three responsibilities**:

1. Managing student data
2. Calculating marks
3. Generating reports

Any change in marks calculation or report format requires modifying this class, which can unintentionally break other functionalities.

To follow SRP, these responsibilities should be separated into different classes.

```
class Student {
public:
    void addStudent() { }
};

class MarksCalculator {
public:
    void calculateMarks() { }
};

class ReportGenerator {
public:
    void generateReport() { }
};
```

Result:

- Each class has a **single responsibility**.
- Changes in marks or report do **not affect student data management**.

- **Violations in Open–Closed Principle (OCP) and Their Fixes**

Violation of OCP
A class violates OCP when **new functionality requires modifying existing code**, rather than extending it. For example:

```
// Violation of OCP
class Shape {
public:
    double area(string shapeType) {
        if(shapeType == "Circle") return 3.14 * 5 * 5;
        else if(shapeType == "Rectangle") return 10 * 5;
        return 0;
    }
};
```

Problem:

- Adding a new shape like Triangle requires modifying the existing Shape class.
- Modifying existing code may introduce bugs in previously working functionality.
- The code is not scalable.

To follow OCP, we use inheritance and polymorphism:

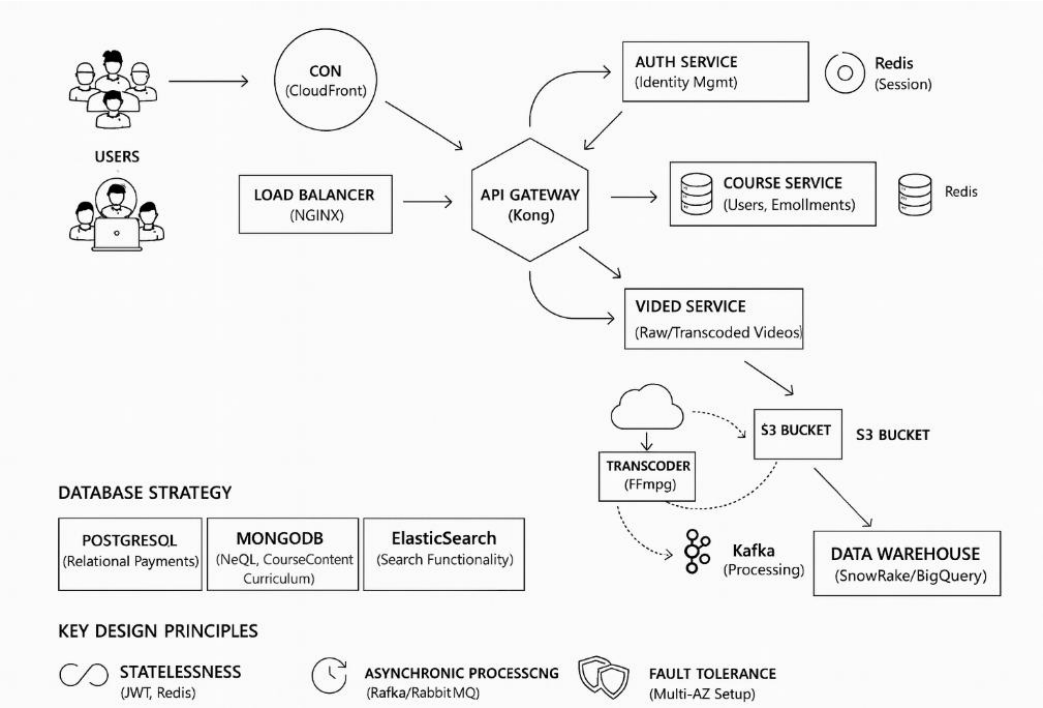
```
class Shape {
public:
    virtual double area() = 0; // open for extension
};

class Circle : public Shape {
public:
    double area() { return 3.14 * 5 * 5; }
};

class Rectangle : public Shape {
public:
    double area() { return 10 * 5; }
};
```

Result:

- The Shape class is closed for modification but open for extension.
- Adding new shapes does not break existing code.

Question 3: Design HLD for an Online Examination System applying these principles.

Answer 3: HLD of Online Examination System:



Component Description and Justification

1. Users
Description:
Users include **students, instructors, and administrators** who interact with the system via web or mobile interfaces.
Why used:
They are the primary actors who register, log in, attempt exams, submit answers, and view results.

2. CDN (Content Delivery Network)
Description:
A CDN distributes static content such as images, scripts, exam instructions, and video content to users from nearby servers.
Why used:

- Reduces latency
- Improves performance during peak exam times
- Handles high traffic efficiently

3. Load Balancer (NGINX)
Description:
The load balancer distributes incoming user requests across multiple backend servers.
Why used:

- Prevents server overload
- Improves availability and fault tolerance
- Ensures smooth operation during simultaneous exams

4. API Gateway
Description:
Acts as a single entry point for all client requests and routes them to appropriate backend services.
Why used:

- Centralized request handling
- Security enforcement (authentication, rate limiting)
- Simplifies microservice communication

5. Authentication Service
Description:
Manages user authentication, authorization, and identity verification.
Why used:
- Ensures only authorized users access exams
- Prevents impersonation and fraud
- Manages login, logout, and token validation

6. Redis (Session Management)
Description:
An in-memory data store used to manage active user sessions and authentication tokens.
Why used:
- Fast session access
- Supports stateless architecture
- Reduces database load

7. Course / Exam Service
Description:
Handles exam-related operations such as exam creation, question delivery, submissions, and evaluation.
Why used:
- Encapsulates core exam logic
- Allows independent scaling
- Improves maintainability

8. Video Service
Description:
Manages recorded or live proctoring videos and instructional content.
Why used:
- Supports online invigilation
- Stores and streams exam-related videos
- Enhances exam integrity

9. Transcoder (FFmpeg)
Description:
Converts raw video into optimized formats suitable for streaming.
Why used:
- Reduces bandwidth usage
- Ensures device compatibility
- Improves video playback quality

10. S3 Bucket (Object Storage)
Description:
Stores large objects such as exam recordings, question papers, and result files.
Why used:
- Scalable and durable storage
- Cost-effective for large files
- Supports secure access

11. Kafka (Message Processing)
Description:
A distributed message queue used for asynchronous processing.
Why used:

- Handles event-based processing (submissions, logs, alerts)
- Improves system reliability
- Enables real-time analytics

12. Data Warehouse
Description:
Stores processed data for reporting and analytics.
Why used:

- Exam performance analysis
- Result statistics generation
- Decision-making support

13. PostgreSQL (Relational Database)
Description:
Stores structured data such as user details, exam schedules, and results.
Why used:

- Strong data consistency
- ACID compliance
- Suitable for transactional data

14. MongoDB (NoSQL Database)
Description:
Stores unstructured or semi-structured data such as question banks and exam patterns.
Why used:

- Flexible schema
- Easy scaling
- Faster access for document-based data

15. Elasticsearch
Description:
Provides full-text search functionality.
Why used:

- Fast question search
- Efficient filtering and indexing
- Enhances user experience

The proposed High-Level Design ensures **scalability, security, reliability, and performance**, making the Online Examination System suitable for large-scale and real-time academic environments.