DEPARTMENT OF
COMPUTER SCIENCE & ENGINEERING
Discover. Learn. Empower.

Assignment No. 1

Student Name: Om Nayak
UID: 23BCS12368
Branch: BE-CSE
Section/Group: KRG-3A
Semester: 6th
Submission date: 04/02/2026
Subject Name: System Design
Subject Code: 23CSH-314

Q1. Explain SRP and OCP in detail with proper examples.

Single Responsibility Principle (SRP)

The Single Responsibility Principle states that a class should have only one reason to change. A responsibility refers to a specific functionality or behavior that may change over time. SRP reduces coupling and increases maintainability.

SRP Violation Example:

```
class StudentService {
    void registerStudent() {}
    void calculateResult() {}
    void saveToDatabase() {}
}
```

This class violates SRP because registration logic, business logic, and persistence logic are tightly coupled.

SRP-Compliant Design:

```
class StudentRegistration {}
class ResultCalculator {}
class StudentRepository {}
```

Each class now has a single responsibility.

Open Closed Principle (OCP)

The Open Closed Principle states that software entities should be open for extension but closed for modification. Existing code should not be altered when adding new features.

OCP Violation Example:

```
if(type.equals("MCQ")) {}
else if(type.equals("CODING")) {}
```

OCP-Compliant Design:

```
interface ExamType {
    void evaluate();
}
```

```
class MCQExam implements ExamType {}
class CodingExam implements ExamType {}
```

Q2. Discuss in detail about the violations in SRP and OCP along with their fixes.

SRP Violations

SRP violations commonly occur when a class grows over time and starts handling multiple unrelated responsibilities. Such classes are often called *God Classes*. These classes are difficult to maintain, test, and extend.

Violation Example:

```
class ExamManager {
    void createExam() {}
    void evaluateExam() {}
    void generateReport() {}
    void storeExam() {}
}
```

Problems Identified:

- Business logic and persistence logic are mixed
- Any change in reporting affects exam logic
- Testing becomes complex due to tight coupling
- High risk of regression bugs

SRP Fix (Separation of Concerns):

```
class ExamCreator {}
class ExamEvaluator {}
class ReportGenerator {}
class ExamRepository {}
```

Each class now has a single reason to change, improving maintainability.

OCP Violations

OCP violations occur when developers use conditional logic to handle variations. This forces modification of existing code whenever a new feature is added.

Violation Example:

```
class QuestionEvaluator {
    int   evaluate(String   type)  {
        if(type.equals("MCQ")) return 1;
        if(type.equals("CODING")) return 2;
        return 0;
    }
}
```

Problems Identified:

- Adding a new question type requires modifying existing code
- High risk of breaking working logic
- Violates extensibility

OCP Fix Using Polymorphism:

```
interface Question {
    int evaluate();
}

class MCQQuestion implements Question {
    public int evaluate() { return 1; }
}

class CodingQuestion implements Question {
    public int evaluate() { return 2; }
}
```

New question types can now be added without changing existing classes.

Q3. Design an HLD for an Online Examination System applying these principles.

The Online Examination System is designed using SRP and OCP to ensure scalability, fault isolation, and extensibility. The system follows a microservices-based and event-driven architecture.

Application of SRP

- User Service – authentication and user management
- Exam Engine – exam lifecycle handling
- Question Engine – pluggable question evaluation
- Search Service – fast retrieval

- Result Service – result storage
- Grading Worker – asynchronous evaluation

Application of OCP

- Question types follow a common interface
- New question types can be added without modifying core logic
- Event-driven grading supports extensibility

High Level Design (HLD)

The system includes Client Layer, API Gateway, Core Services, Event Pipeline, Databases, and Cache. Kafka enables asynchronous processing, Redis supports caching, and CDC synchronizes data with ElasticSearch.

Note: The complete HLD diagram has been created using draw.io and is uploaded along with this assignment.

HLD OUTPUT