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

Assignment

Student Name: Ankit Kumar
UID: 23BCS13740
Branch: BE-CSE
Section/Group: KRG-2B
Semester: 6th
Date of Performance: 03/02/2026
Subject Name: System Design
Subject Code: 23CSH-314

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

1.Single Responsibility Principle (SRP)

Definition

The **Single Responsibility Principle (SRP)** states that:

A class should have only one responsibility or one reason to change.

This means a class should perform **only one specific task**.

Example to Explain SRP

```
class Invoice {
   void calculateTotal() {
      // calculate total amount
   }
}
class InvoicePrinter {
   void printInvoice() {
      // print invoice
   }
}
class InvoiceRepository {
   void saveInvoice() {
      // save invoice to database
   }
}
```

Explanation

In this example:

- Invoice is responsible for **calculation**
- InvoicePrinter is responsible for **printing**
- InvoiceRepository is responsible for **storing data**

Each class has **only one responsibility**, which follows the **Single Responsibility Principle**.

2.Open/Closed Principle (OCP)

Definition

The **Open/Closed Principle (OCP)** states that:

A software component should be open for extension but closed for modification.

This means new functionality should be added **without changing existing code**.

Example to Explain OCP

```
interface Shape {
   double calculateArea();
}
class Rectangle implements Shape {
   public double calculateArea() {
      return 10 * 5;
   }
}
class Circle implements Shape {
   public double calculateArea() {
      return 3.14 * 7 * 7;
   }
}
```

Explanation

In this example:

- Shape interface defines a common structure
- New shapes can be added by **creating new classes**
- Existing code does not need to be modified

Thus, the system is **open for extension** and **closed for modification**, which follows the **Open/Closed Principle**.

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

1) SRP – Single Responsibility Principle

Definition

Single Responsibility Principle (SRP) states that:

A class should have only one reason to change, meaning it should perform only one specific responsibility.

In simple words, **one class = one job**.

Explanation

If a class is handling **more than one responsibility**, then:

- Changes in one responsibility may affect others
- Code becomes hard to maintain
- Testing becomes difficult

SRP helps in:

- Better readability
- Easier maintenance
- Reduced side effects when code changes

Violation of SRP (Bad Design)

```
class Student {
   void addStudent() {
      // add student to database
   }

   void calculateResult() {
      // calculate marks and grades
   }

   void printReport() {
      // print student report
   }
}
```

Problem:

- One class is doing **three different jobs**
    - Database work
    - Business logic
    - Report generation

If report format changes → Student class changes
If database changes → Student class changes
Multiple reasons to change

SRP Followed (Good Design)

```
class Student {
   String name;
   int marks;
}


class StudentRepository {
   void addStudent(Student s) {
      // database logic
   }
}


class ResultCalculator {
   void calculateResult(Student s) {
      // business logic
   }
}
class ReportPrinter {
   void printReport(Student s) {
      // report logic
   }
}
```

Why this is good:

- Each class has **one responsibility**

- Changes in one part don't affect others
- Easy to test and maintain

2) OCP – Open/Closed Principle

Definition

Open/Closed Principle (OCP) states that:

Software entities should be open for extension but closed for modification.

This means:

- You should be able to **add new features**
- Without **changing existing code**

Explanation

When new requirements come:

- We should extend the system
- Not modify already tested and working code

OCP reduces the risk of:

- Introducing new bugs
- Breaking existing functionality

Violation of OCP (Bad Design)

```
class Payment {
   void pay(String type) {
      if (type.equals("CreditCard")) {
         // credit card payment
      } else if (type.equals("UPI")) {
         // UPI payment
      }
   }
}
```

Problem:

- Adding a new payment method (PayPal, NetBanking)
- Requires **modifying the existing class**
- Violates OCP

OCP Followed (Good Design)

```
interface Payment {
    void pay();
}

class CreditCardPayment implements Payment {
    public void pay() {
        // credit card payment logic
    }
}

class UPIPayment implements Payment {
    public void pay() {
        // UPI payment logic
    }
}
```

Now if a new payment method is needed:

```
class PayPalPayment implements Payment {
    public void pay() {
        // PayPal payment logic
    }
}
```

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

1. Introduction

An Online Examination System is a software application that allows students to take exams online, evaluators to assess performance, and administrators to manage exams, users, and results. The system is designed following Single Responsibility Principle (SRP) and Open/Closed Principle (OCP) to ensure scalability and maintainability.

2. Functional Requirements

Functional requirements describe what the system should do.

2.1 User Management

- The system shall allow students, teachers, and administrators to register and log in.
- The system shall authenticate users based on roles.
- The system shall allow administrators to manage user accounts.

2.2 Exam Management

- The system shall allow administrators to create, update, and delete exams.
- The system shall support different exam types (MCQ, descriptive).
- The system shall allow scheduling of exams with date and time.

2.3 Question Management

- The system shall allow teachers to add, update, and remove questions.
- The system shall support multiple question types.
- The system shall maintain a question bank.

2.4 Exam Participation

- The system shall allow students to attempt exams online.
- The system shall display questions one by one or all at once.
- The system shall submit answers automatically when time expires.

2.5 Evaluation and Result Processing

- The system shall automatically evaluate objective questions.
- The system shall allow manual evaluation for descriptive answers.
- The system shall generate results and scores for students.

2.6 Result Viewing and Reports

- The system shall allow students to view their results.
- The system shall allow administrators to generate exam reports.
- The system shall provide performance analysis.

3. Non-Functional Requirements

Non-functional requirements describe how the system should perform.

3.1 Performance

- The system should support multiple users simultaneously.
- Exam submission and result generation should be fast.

3.2 Scalability

- The system should handle an increase in number of users and exams.
- New exam types should be added without modifying existing modules (OCP).

3.3 Security

- The system should provide secure login and authentication.
- Exam data and results should be protected from unauthorized access.
- The system should prevent cheating using session and time controls.

3.4 Reliability

- The system should not lose data during power or network failure.
- Auto-save of answers should be supported.

3.5 Availability

- The system should be available during scheduled exam time.
- Minimal downtime during exams.

4. Design Principles Applied

- **SRP**:
  Separate modules for user management, exam management, evaluation, and reporting.
- **OCP**:
  New question types or exam formats can be added by extending the system without modifying existing components.