ASSIGNMENT- 1

Student Name: Arshpreet Singh	**UID: 23BCS10502**
Branch: BE-CSE	**Section/Group: KRG 1-B**
Semester: 6th	**Date of Performance: 01-02-2026**
Subject Name: System Design	**Subject Code: 23CSH-314**

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

Ans :-

1. **Single Responsibility Principle (SRP)**
   SRP states that a class should have only one reason to change. In simple words: *A class should do only one job.*
   Each class/module should handle one responsibility and one business logic.

Why SRP is Important?

1. Easier to understand
2. Easier to test
3. Easier to maintain
4. Less bugs
5. Better reusability

Drawbacks:-

One class is doing:

- Data handling
- Report generation
- Email service

Example (Violation):

```
class Student {
public:
    void addStudent() {
        // add student to database
    }
    void generateReport() {
        // generate PDF report
    }
    void sendEmail() {
        // send email to student
    }
};
```

SRP correct Design

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

class ReportService {
public:
    void generateReport();
};

class EmailService {
public:
    void sendEmail();
};
```

Now,

1. Each class has **one responsibility**
2. Changes in email won't affect reports

2. Open/Closed Principle (OCP)

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

We should be able to:

- Add new features
- Without changing existing tested code

Why OCP is Important?

- Prevents breaking existing logic
- Encourages scalable systems
- Supports plug-and-play architecture

Example (Violation):

```
class Payment {
public:
   void pay(string type) {
      if(type == "UPI") {
         // upi logic
      } else if(type == "CARD") {
         // card logic
      }
   }
};
```

OCP Correct Design (Using Polymorphism)

```
class Payment {
public:
    virtual void pay() = 0;
};

class UpiPayment : public Payment {
public:
    void pay() override { }
};

class CardPayment : public Payment {
public:
    void pay() override { }
};
```

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

Ans.

Software design principles are introduced to improve the quality, maintainability, scalability, and reliability of software systems. Among them, Single Responsibility Principle (SRP) and Open/Closed Principle (OCP) play a crucial role. However, these principles are often violated in real-world systems, leading to poor design. This section discusses the violations, their causes, consequences, and fixes in detail.

Part A: Violations of Single Responsibility Principle (SRP)

1. Meaning of SRP Violation

A violation of SRP occurs when:

A single class or module is responsible for more than one functionality or business concern.

In such cases, the class has multiple reasons to change, which directly contradicts the core idea of SRP.

Common Causes of SRP Violation

1. Lack of proper design planning
   Developers start coding without analysing responsibilities.
2. Procedural thinking in OOP systems
   Mixing logic like database, UI, and business logic in one class.
3. Time constraints
   For quick delivery, everything is placed in one class.
4. Overloaded manager/controller classes
   God classes that handle everything.

Characteristics / Symptoms of SRP Violation

A class violating SRP usually shows:

- Very large size (too many methods)
- Unrelated methods in same class
- Difficult to understand
- Hard to test
- Changes in one feature break others
- High coupling, low cohesion

Such classes are often called:

God Classes / Blob Classes

Consequences of SRP Violation

1. Poor Maintainability
   Any small change requires understanding the whole class.
2. Low Reusability
   Cannot reuse logic independently.
3. High Risk of Bugs
   Modifying one responsibility may break another.
4. Difficult Testing
   Unit testing becomes complex.
5. Tight Coupling
   Many components become dependent on one class.

Fix for SRP Violations

The solution is:

Separation of Concerns

This means:

- Identify different responsibilities.
- Move each responsibility into a separate class/module.

Techniques Used:

- Layered Architecture
- Service classes
- Repository pattern
- MVC (Model-View-Controller)

Part B: Violations of Open/Closed Principle (OCP)

1. Meaning of OCP Violation

An OCP violation occurs when:

Existing source code must be modified to add new functionality.

This means the system is not extensible, and every change risks breaking working code.

Common Causes of OCP Violation

1. Excessive use of conditional statements
   if-else, switch-case for different behaviors.
2. No abstraction
   Concrete classes used directly instead of interfaces.
3. Poor use of polymorphism
   Behavior controlled by flags or type variables.
4. Hard-coded logic
   Business rules written directly inside methods.

Characteristics / Symptoms of OCP Violation

A system violating OCP usually shows:

- Many if-else conditions

- Frequent modification of same files
- Ripple effect (one change affects many modules)
- Lack of interfaces or abstract classes

Consequences of OCP Violation

1. High Regression Risk
   New features may break existing functionality.
2. Low Scalability
   Difficult to add new requirements.
3. Increased Testing Cost
   Entire system must be retested.
4. Code Fragility
   System becomes unstable over time.
5. Violation of Reusability
   Logic is tightly bound and not pluggable.

Fix for OCP Violations

The solution is:

Abstraction and Polymorphism

Key Techniques:

- Interfaces
- Abstract classes
- Strategy pattern
- Factory pattern
- Dependency Inversion
- Plugin-based architecture

Comparative Analysis (SRP vs OCP Violations)

Aspect	SRP Violation	OCP Violation
Core problem	Too many responsibilities	Not extensible
Main cause	Poor separation of concerns	No abstraction
Impact	Poor maintainability	Poor scalability
Main symptom	God classes	if-else chains
Primary fix	Split classes	Use interfaces
Design level	Structural issue	Behavioral issue

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

Ans. On Draw.io File