Name: Atul Kumar Gour Section: KRG_3A

UID: 23BCS11181 SUBJECT: SD

Assignment 1

Q1. Explanation of Single Responsibility Principle (SRP) and Open–Closed Principle (OCP)

The Single Responsibility Principle (SRP) is one of the fundamental principles of object-oriented software design and is part of the SOLID design principles. SRP states that a class, module, or software component should have only one reason to change, meaning it should be responsible for only one specific functionality or concern. The principle emphasizes separation of concerns, where each responsibility within a system is isolated into a dedicated component. When a class handles multiple responsibilities, changes in one responsibility may unintentionally affect others, making the system harder to maintain, test, and scale. By following SRP, software systems achieve higher cohesion, reduced coupling, improved readability, and better maintainability.

For example, consider a class named StudentManager that handles student data storage, result calculation, and report generation. This design violates SRP because the class has multiple reasons to change, such as changes in database schema, grading logic, or reporting format. Applying SRP involves decomposing this class into multiple classes such as StudentRepository for database operations, ResultCalculator for computing marks and grades, and ReportGenerator for generating result reports. Each class now has a single responsibility, making the system easier to modify and extend.

The Open–Closed Principle (OCP) states that software entities such as classes, modules, and functions should be open for extension but closed for modification. This principle encourages designing systems in such a way that new functionality can be added without altering existing, tested code. The main motivation behind OCP is to minimize the risk of introducing defects when requirements change. OCP is typically implemented using abstraction, interfaces, inheritance, and polymorphism, allowing developers to extend behavior by adding new code rather than modifying existing implementations.

Q2. Violations of SRP and OCP Along with Their Fixes

Violations of the Single Responsibility Principle commonly occur when a single class is designed to handle multiple unrelated responsibilities. Such classes are often referred to as god classes. For example, a controller class that manages user input, business logic, database access, and logging violates SRP because it has multiple reasons to change. These violations make the system difficult to understand, debug, and maintain. Fixing SRP violations involves refactoring the code by identifying distinct responsibilities and separating them into independent classes or services. This refactoring leads to loosely coupled components with clearly defined responsibilities.

Violations of the Open–Closed Principle typically arise when systems rely heavily on conditional logic such as if-else or switch-case statements to determine behavior. When new requirements are introduced, developers are forced to modify existing code, increasing complexity and the likelihood of errors. For example, adding a new discount type in a billing system may require modifying multiple conditional blocks. The fix for OCP violations is to introduce abstraction through interfaces or abstract classes and apply polymorphism. Each new behavior is implemented as a new class, allowing the system to be extended without modifying existing code.

In practice, SRP and OCP violations often occur together. A class that violates SRP by handling multiple responsibilities is also more likely to violate OCP, as any extension requires modification of existing logic. Applying both principles together improves modularity, testability, scalability, and long-term maintainability of software systems.

Q3. High-Level Design (HLD) of an Online Examination System Applying SRP and OCP

An Online Examination System is a large-scale application designed to manage exam creation, scheduling, question delivery, evaluation, result processing, and reporting. Applying the Single Responsibility Principle at the architectural level ensures that each system component is responsible for a specific functional area. For example, an Authentication Service handles user login and authorization, an Exam Management Service manages exam scheduling and configuration, a Question Bank Service stores and retrieves questions, an Evaluation Service processes answers and calculates scores, and a Result Service generates and publishes results. Each service is independent and focused on a single responsibility.

he Open–Closed Principle is applied by designing the system using interfaces and extensible components. The evaluation mechanism can be implemented using an EvaluationStrategy interface, with different implementations for multiple-choice exams, descriptive exams, and coding-based exams. When a new evaluation technique such as AI-based assessment is required, it can be introduced as a new implementation without modifying existing evaluation logic. Similarly, notification functionality can be extended by implementing new notification services for email, SMS, or push notifications using a common NotificationService interface.

At a high level, the system follows a layered architecture consisting of a presentation layer for user interaction, a service layer implementing business logic using SRP-compliant services, and a data access layer responsible for persistence operations. Each layer depends on abstractions rather than concrete implementations, ensuring compliance with the Open–Closed Principle. This design results in a scalable, maintainable, and extensible Online Examination System suitable for evolving academic and enterprise requirements.