ASSIGNMENT 1

Student Name: Tanisha Kumari
UID: 23BCS12542
Branch: CSE
Section/Group: KRG_1B
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 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.

This means **one class = one responsibility**.

Why SRP is important

- Easier maintenance
- Better readability
- Reduced impact of changes
- Improved testability

Example (Violation)

```
class Report {
   void generateReport() { }
   void saveToFile() { }
   void printReport() { }
}
```

SRP-Compliant Version

```
class ReportGenerator {
   void generateReport() { }
}
```

```
class ReportSaver {
   void saveToFile() { }
}
class ReportPrinter {
   void printReport() { }
}
```

*Each class has **one responsibility***

2. Open–Closed Principle (OCP)

Definition

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.

Example (Violation)

```
class Discount {
   double calculate(String type) {
      if(type.equals("Student")) return 0.1;
      if(type.equals("Senior")) return 0.2;
      return 0;
   }
}
```

OCP-Compliant Version

```
interface Discount {
   double calculate();
}

class StudentDiscount implements Discount {
   public double calculate() { return 0.1; }
}

class SeniorDiscount implements Discount {
   public double calculate() { return 0.2; }
}
```

New discounts can be added without changing existing code

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

SRP Violation

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

- A class performs multiple tasks
- Example: Authentication + Logging + Database operations

Problems

- Difficult to maintain
- High coupling
- Hard to test

Fix

- Split responsibilities into separate classes

OCP Violation

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

Violation

- Use of if–else or switch for behavior changes
- Modifying existing code for new functionality

Problems

- Risk of introducing bugs
- Code becomes rigid

Fix

- Use interfaces, inheritance, and polymorphism

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

Functional Requirements:

- User should be able to register and login
- System should support different roles (Student, Admin)

- Admin should be able to create and schedule exams
- Admin should be able to add/manage questions
- Student should be able to attempt exams online
- System should evaluate answers automatically
- System should generate results
- System should notify users about exam and results

Non Functional Requirements:

A. Scalability
- System should handle multiple students simultaneously during exams

B. Security
- Secure authentication and authorization
- Exam data should be protected

C. Performance
- Low response time during exam submission

D. Reliability
- No data loss during exam submission

E. Maintainability
- Easy to add new exam or evaluation types

Core Entities:

- User
- Exam
- Question
- Answer
- Result

API Endpoints

Auth & User Service APIs
- POST /login
- POST /register
- GET /users/{id}

Exam Management Service APIs
- POST /exams
- GET /exams

Question Bank Service APIs

- POST /questions
- GET /questions/{examId}

Evaluation Service APIs

- POST /evaluate

E. Result Service APIs

- GET /results/{userId}

High Level Diagram :

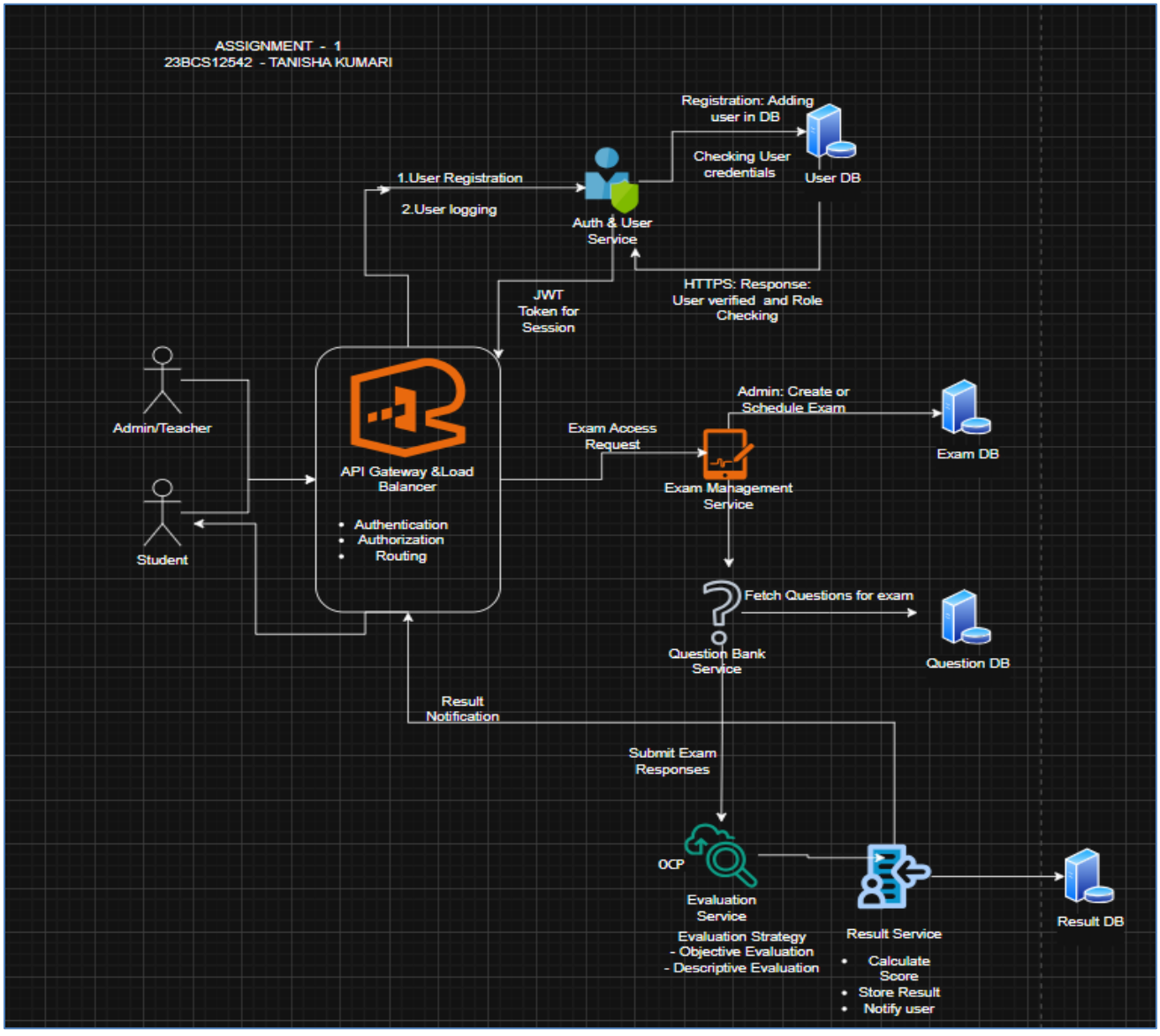

Figure 1: HLD of Online Examination System