Assignment Name: Layered Architecture Model of an online Judge

Answer: An online judge is a system that automatically evaluates and grades code submissions based on pre-defined test cases. It is commonly used in programming competitions and practice platforms. The layered architecture of an online judge can be divided into multiple functional layers that interact with each other, each serving a distinct purpose.

Layered Architecture Model of an Online Judge:

1. User interface Layer (Presentation Layer)

   → A Web-based user interface (UI)

   → APIs for user communication

   → User account management (registration, login, profile)

   → Dashboard for contests, problem and submissions

2. Application Layer (Business Logic Layer):

→ Submission handler

→ Problem manager

→ Contest manager

→ Grading engine

→ Notification service

3. Execution Layer (Service Layer):

→ Code execution engine

→ Compilation and interpretation services

→ Test case execution

→ Resource management

→ Time and memory limit checking

4. Data Layer (Persistence Layer):

→ Database for storing user data, submissions, problem statements and results.

→ Data management (CRUD operations)

→ Query optimization and indexing.

5. Security Layer:

→ User authentication and authorization

→ Sandboxing for secure code execution.

→ Code analysis to detect malicious submissions

→ HTTPS/SSL encryption for secure communication.

OJ Architecture

```
        ┌─────────────────────────┐
        │   Presentation Layer    │
        └─────────────────────────┘
                    │
                    ▼
        ┌─────────────────────────┐
        │          API            │
        └─────────────────────────┘
           ↙                    ↘
┌──────────────┐  ┌──────────────┐  ┌──────────────┐
│  Submission  │←─│code Execution│→ │  Submission  │
│   Database   │  │    Engine    │  │    Queue     │
└──────────────┘  ├──────────────┤  └──────────────┘
                  │   Sandbox    │
                  ├──────────────┤
                  │  Compiler/   │
                  │ Interpreter  │
                  └──────────────┘
```