**day27\_107856406\_dsdipt\_sudipto\_14august2025**

**Employee Code:** 107856406

**Login ID:** dsdipt

**Email :** dsdipt@amazon.com

**Name:** Sudipto Das

**Date:** 14 August 2025 (Day 27)

### ***Task 00: MVC & Large-Scale System Architecture***

**MVC Framework (Model–View–Controller)**

MVC is an architectural pattern that separates an application into **three interconnected components**:

* + **Model** → Manages data, business logic, and rules.
  + **View** → Handles the presentation layer (UI).
  + **Controller** → Receives user input, processes it, and updates the Model or View.

The idea is to keep the UI (View) and business logic (Model) separate, with the Controller acting as a mediator.

**Key Points:**

* **Loose coupling** between data, UI, and control flow.
* **Reusability** → Model can work with different Views.
* **Maintainability** → Easier to update UI or logic without affecting the other.
* **Flow:** User action → Controller → Model update → View refresh.
* Commonly used in frameworks like Spring MVC, ASP.NET MVC, Django (MTV pattern), Rails.

**When to Use:**

* When you want a **clear separation** between UI and business logic.
* In **web applications** where the same data may be displayed in multiple formats.
* When you need **scalable and maintainable** application structure.

**Large-Scale Systems – Architectural Patterns**

Large-scale systems need **architectural patterns** that focus on scalability, maintainability, and reliability. Common patterns include:

**Microservices Architecture**

* + Breaks an application into small, independently deployable services.
  + Each service has its own database and can scale independently.

**Event-Driven Architecture**

* + Components communicate through events (Pub/Sub, message queues).
  + Allows asynchronous processing and decoupled services.

**Layered (N-Tier) Architecture**

* + Organizes the system into layers (presentation, business logic, data access, etc.).
  + Promotes modularity and separation of concerns.

**Service-Oriented Architecture (SOA)**

* + Similar to microservices but often uses heavier protocols (SOAP, ESB).

**CQRS (Command Query Responsibility Segregation)**

* + Separates read and write operations into different models for performance and scalability.

**Key Points:**

* Large-scale systems often combine **multiple architectural patterns**.
* They prioritize **horizontal scaling** (more servers) over vertical scaling (bigger server).
* Reliability and fault tolerance are critical — patterns like **Circuit Breaker** and **Bulkhead** are often used.
* Observability (logging, metrics, tracing) is essential.

**When to Use:**

* When building **enterprise-grade or high-traffic** applications.
* When the system must handle **millions of requests** and scale across multiple regions.
* When services need to be **maintained and deployed independently** without downtime.

### ***Task 01: MCQs***

1. **In which of the following mechanisms, types of all variables and expressions are fixed at compilation time.**  
    a) Strong Typing  
    b) Weak Typing  
    **c) Static Binding/ early binding** d) Dynamic Binding/ late binding
2. **In which of the following mechanisms, types of all variables and expressions are not known until runtime**  
    a) Strong Typing  
    b) Weak Typing  
    c) Static Binding/ early binding  
    **d) Dynamic Binding/ late binding**
3. **Which of the following statements about Persistence is correct?**  
    a) It is the enforcement of the class of an object, such that objects of different types may not be interchanged, or at the most they may be interchanged only in very restricted ways.  
    **b) It is the property of an object through which its existence transcends time and/or space.** c) It is the property that distinguishes an active object from one that is not active.  
    d) All of the mentioned
4. **What is that concept in type theory in which a single name may denote objects of many different classes that are related by some common super class referred to \_\_\_\_\_\_**  
    a) Monomorphism  
    b) Type Checking  
    **c) Polymorphism** d) Generalization
5. **Which of the following patterns is used to create a single instance of a class and provide a global point of access to it?**  
    a) Factory Pattern  
    **b) Singleton Pattern** c) Builder Pattern  
    d) Prototype Pattern
6. **The Adapter pattern is a type of \_\_\_\_\_\_ pattern.**  
    a) Creational  
    **b) Structural** c) Behavioral  
    d) Concurrency
7. **Which design pattern defines a one-to-many dependency between objects so that when one object changes state, all its dependents are notified and updated automatically?**  
    a) Strategy Pattern  
    b) Command Pattern  
    **c) Observer Pattern** d) Mediator Pattern
8. **In which pattern does a class represent the functionality of another class, providing a simplified interface to a complex subsystem?**  
    a) Decorator Pattern  
    **b) Facade Pattern** c) Proxy Pattern  
    d) Composite Pattern
9. **The Model-View-Controller (MVC) is an example of a \_\_\_\_\_\_ pattern.**  
    a) Creational  
    b) Structural  
    c) Behavioral  
    **d) Architectural**