# Software Design Activity - Answers

1. Layered Architecture

Key Attributes: Divides the system into layers, each with specific responsibilities.

Common Uses: Enterprise applications, large-scale systems.

2. Client-Server Architecture

Key Attributes: Separates client (requester) and server (provider) roles.

Common Uses: Web applications, distributed systems.

3. Event-Driven Architecture

Key Attributes: Components communicate via events, allowing asynchronous processing.

Common Uses: Real-time systems, GUIs, IoT systems.

4. Microservices Architecture

Key Attributes: Decomposes applications into loosely coupled services.

Common Uses: Large-scale, scalable, and maintainable applications.

5. Pipe-and-Filter Architecture

Key Attributes: Data is processed in a series of filters connected by pipes.

Common Uses: Data transformation, compilers, stream processing.

Restaurant Ordering Application Design:

The application will consist of a web-based client for customers to browse the menu, place orders, and make payments, connected to a backend service managing orders, inventory, and kitchen workflow. The best architecture pattern would be the Microservices Architecture, as it allows independent scaling of order processing, inventory management, and payment handling services.

Class Diagram:

- Type: Static, Structural

- Purpose: Represents the structure of a system by showing its classes, attributes, and relationships.
- Main Components: Classes, attributes, methods, associations.

Object Diagram:
- Type: Static, Structural
- Purpose: Shows a complete or partial view of the structure of a modeled system at a specific time.
- Main Components: Objects (instances of classes), links between objects.

Deployment Diagram:
- Type: Static, Structural
- Purpose: Models the physical deployment of artifacts on nodes.
- Main Components: Nodes, artifacts, communication paths.

Sequence Diagram:
- Type: Dynamic, Behavioral
- Purpose: Models the sequence of interactions between objects in a time-ordered manner.
- Main Components: Actors, lifelines, messages.

Use Case Diagram:
- Type: Static, Behavioral
- Purpose: Describes the functional requirements of a system using actors and use cases.
- Main Components: Actors, use cases, system boundaries, relationships.

**Restaurant Ordering System Diagram:**

**[Diagram not available - please insert manually]**