SOFTWARE ENGINEERING
ASSIGNMENT

STUTI GARG
SE22UCSE263
CSE-4

TASK: Explain each of these architectural styles; Discuss the strengths and weaknesses; Explain suitable industry applications using these architectures.

(1) Monolithic Architecture

→ Monolithic architecture is a traditional software design where all components are unified into a single codebase. The user interface, business logic and data access layers are tightly coupled and deployed as a single unit. This makes development straightforward but can create challenges as the application grows in complexity.

Strengths:

(i) Simple to develop and test: Developers can build & run the entire application without worrying about distributed components.
(ii) Easier to deploy and manage initially: Since everything is in one packet, deployment processes are simpler.
(iii) Performance benefits due to fewer network calls: Communication b/w components happens in process, reducing latency.

Weaknesses:

(i) Hard to scale horizontally: Scaling requires replicating the entire system, even if one part needs more resources.
(ii) Difficult to maintain & update: Changes in one part of the system can impact others.
(iii) Single point of failure: A crash in one module can bring down the entire application.

Industry Applications

(i) Early-stage startups with simple applications: Startups often begin with a monolithic approach for faster time-to-market.
(ii) Legacy systems: Many older enterprise systems were built as monoliths and still operate that way.
(iii) Small business applications: Smaller systems with limited scope can benefit from the simplicity of a monolithic approach.

(2) Client-Server Architecture

→ Client server architecture divides applications into clients that request services and servers that fulfill those requests.

→ N-tiered architectures expand this concept by adding layers like presentation, application and data layers each handling specific responsibilities.

→ This promotes seperation of concerns and better manageability.

Strengths:

(i) Seperation of concerns : Different layers handle distinct tasks, making it easier to maintain.
(ii) Easier to scale specific layers : Each layer can be scaled independently, allowing resource optimization.
(iii) Centralized Security: Security policies can be managed at the server or database level, reducing vulnerabilities.

Weaknesses:

(i) Can become complex with multiple tiers: More layers implies more moving parts that can complicate development.
(ii) Network dependency can introduce latency: Communication between tiers adds overhead, especially in distributed environments.

Industry Applications:

(i) Web applications : Most modern web apps use a multi-tiered design for better structure and scalability.
(ii) Enterprise Resource Planning (ERP) systems: Large business applications often split logic across multiple layers.
(iii) Banking systems: Security and reliability make n-tier architectures ideal for handling sensitive transactions.

(3) Web-Based Architecture

→ Web-based architecture uses the internet to deliver applications to users through browsers. Servers handle business logic and data processing, while the browser renders the user interface. This approach makes applications accessible globally, without requiring installation.

Strengths:

(i) Platform independence: Users can access applications from any device with a browser.
(ii) Easy deployment & updates: Updates are applied to the server, instantly available to all users.
(iii) Global accessibility: Applications can reach users worldwide, enabling a larger audience.

Weaknesses:

(i) Performance relies on Internet connectivity: Users need a stable connection for smooth experience.
(ii) Security vulnerabilities: Web apps are exposed to online threats like DDoS attacks, requiring strong security measures.

Industry Applications:

(i) E-commerce platforms: Online stores like Amazon rely on web-based architecture to serve millions of users.
(ii) Social media networks: Platforms like facebook and Twitter connect people through web interfaces.
(iii) Content management systems: Tools like WordPress let users manage content directly from a browser.

(4) Cloud-Based Architecture

→ Cloud-based architecture uses remote servers hosted on the internet for data storage, computing and service delivery.

→ Applications can dynamically scale based on demand, and users can access them from anywhere.

Strengths:

(i) Scalability and Flexibility: Resources can be adjusted to handle varying workloads.

(ii) Cost efficiency with pay-as-you-go models: Reduces infra costs.

(iii) High availability & reliability: Cloud providers offer redundancy & failover mechanisms to ensure uptime.

Weaknesses:

(i) Dependency on Internet Connectivity: Without internet access, users can't interact with cloud-hosted services.

(ii) Potential vendor lock-in: Switching providers can be challenging due to proprietary services & data migration complexities.

Industry Applications:

(i) SaaS products: Tools like Google Workspace and Dropbox deliver functionality through the cloud.

(ii) Big data analytics: Platforms like AWS and Azure handle large-scale data processing.

(iii) Online storage services: Services like Google Drive and iCloud store user data in the cloud.

(5) Peer-to-Peer Architecture

→ Peer-to-Peer (P2P) architecture enables direct communication between nodes (peers) without relying on a central server.

→ Each peer can act as both a client and a server, sharing resources and workload.

Strengths:

(i) High fault tolerance: The network can keep functioning even if some nodes fail.

(ii) Resource efficiency: Peers share their resources, reducing reliance on centralized infrastructure.

Weaknesses:

(i) Difficult to secure and regulate: Decentralized systems can be harder to monitor and protect.

(ii) Inconsistent performance: Network performance can fluctuate based on peer availability & capacity.

Industry Applications:

(i) File sharing networks: Platforms like BitTorrent use P2P for fast, distributed file sharing.

(ii) Blockchain & Cryptocurrency systems: Cryptocurrencies like Bitcoin rely on P2P networks to validate transactions.

(iii) Decentralized applications (DApps): Apps like IPFS use P2P for distributed file storage.

(6) Service-Oriented Architecture

→ Service-oriented architecture structures software as a collection of services that communicate through standard protocols.

→ Each service performs a specific task and can be reused across different applications.

Strengths:

(i) Reusability of Services: Individual services can be reused across multiple applications, reducing development effort.
(ii) Platform Independence: Services can be developed in different technologies till they follow communication standards.
(iii) Scalability & Flexibility: Systems can evolve by adding services without disrupting the entire application.

Weaknesses:

(i) Complex service management: Managing many services (esp in distributed systems, is challenging.
(ii) Performance Overhead: Communication b/w services can introduce latency with complex workflows.

Industry Applications:

(i) Payment processing systems: Services like PayPal or Stripe offer reusable payment APIs.
(ii) Healthcare information systems: Different hospital systems can exchange patient data through standardized services.
(iii) Supply chain management: Businesses can integrate third-party logistics & inventory systems via services.

(1) Event-Driven Architecture

→ Event-driven architecture revolves around events triggering actions within a system.

→ Components produce and consume events asynchronously, enabling reactive, decoupled systems.

Strengths:

(i) High responsiveness : Systems can react to events in real time.

(ii) Loose Coupling : Components are independent, making the system more flexible and scalable.

Weaknesses:

(i) Complex debugging & monitoring : Asynchronous communication can make it harder to trace issues.

(ii) Event ordering challenges : Handling event sequences correctly can require careful design.

Industry Applications:

(i) E-commerce platforms : Real-time order tracking & inventory updates.

(ii) IOT systems : Sensor data triggering automated actions.

(iii) Fraud detection : Instant alerts for suspicious transactions.

(8) Microservices Architecture

→ Microservices break applications into small, independently deployable services.

→ Each service focuses on a specific functionality & communicates via API's.

Strengths:

(i) Scalability: Services can be scaled independently.

(ii) Resilience: Failures in one service doesn't necessarily affect other systems.

Weaknesses:

(i) Operational complexity: Managing many services can be challenging.

(ii) Network latency: Inter-service communication can slow down the system.

Industry Applications:

(i) Streaming platforms: Netflix uses microservices for content delivery.

(ii) E-commerce giants: Amazon's architecture supports massive scale.

(iii) Financial Services: Banks use micro-services for modular, scalable systems.