Name :- Nisarg .K. Amlani
Roll :- CE001
ID :- 22ceueg082

Lab 1

Q1) What is monolithic architecture? Mention the advantages and disadvantages.

Ans) Monolithic Architecture is a design pattern that is combined of an application's component into a single inseparable unit .

Advantages

- **Simplicity** : Monolithic architecture is easy to develop, test, and deploy because it's built as a single unit with a single codebase.
- **Cost-Effective :** Monolithic architecture can be cost-effective to start with, requiring less investment in infrastructure and human resources.
- **Specialist knowledge**: As an application grows, development teams can specialize in one or two parts, allowing technology specialists to apply their knowledge.

Disadvantages

- **Scalability :** It's difficult to scale individual parts of a monolithic application because all components are bundled together.
- **Updates and maintenance :** Monolithic applications are more complex to update and maintain than microservices-based applications.
- **Single point of failure**: Monolithic architecture has a single point of failure, which can increase the risk of system-wide outages.

- **Continuous deployment**: Monolithic architecture can be challenging for continuous deployment because the entire application must be redeployed for any change.

Q2) What is client - server architecture ? mention advantages and disadvantages ?

Ans) The Client-server model is a distributed application structure that partitions tasks or workloads between the providers of a resource or service, called servers, and service requesters called clients

Advantages

- Centralized system with all data in a single place.
- Cost efficient requires less maintenance cost and Data recovery is possible.
- The capacity of the Client and Servers can be changed separately.

Disadvantages

- Clients are prone to viruses, Trojans, and worms if present in the Server or uploaded into the Server.
- Servers are prone to Denial of Service (DOS) attacks.
- Data packets may be spoofed or modified during transmission.

Q3) What is the difference between single-tier and two-tier client server architectures? Mention the applications of both.

Ans) **Single-tier client-server architecture,** also known as one-tier architecture, is a software architecture where all the components of an application are stored on a single device or shared storage device.

Typically used for small , simple application a prototype

A **two-tier client-server architecture** is an IT infrastructure model that separates application components into two layers: the client or front-end, and the server or back-end. The client interacts directly with the server through a protocol.

E.g Used for non-complex , non-time critical information system

Q4) What is multi-tier client server architecture? Mention the advantages and applications.

Ans) A software architecture that organizes application components into tiers, or layers, to provide specific functionality.

Advantages

- **Improved scalability :** Each tier can be scaled independently of the others, allowing you to meet changing performance demands.
- **Easy Maintenance :** Modifications to one tier often have minimal or no effect on other tiers.
- **Easier Operations :** In a large system, only a few people are expert at each tier's applications .

Application : provides a general framework to ensure decoupled and independently scalable application components can be separately developed, managed, and maintained

Q5) What is distributed internet architecture? Compare and contrast with other architecture and identify the applications.

Ans) A distributed internet architecture is a system where components are spread across multiple interconnected computers or servers.

Comparison with other Architectures

Centralized Architecture:

- **Single point of failure:** A single server handles all requests, making it vulnerable to failures.
- **Scalability limitations:** Difficulty in scaling to handle increased load.
- **Lower fault tolerance:** A failure in the central server can disrupt the entire system.

Client-Server Architecture:

- **Improved scalability:** Multiple servers can handle requests, increasing capacity.
- **Better fault tolerance:** Failure of one server can be mitigated by others.
- **Still relies on a central server:** A single point of failure remains, albeit less critical.

Distributed Architecture:

- **High scalability:** Effortlessly scales by adding more nodes.
- **Enhanced fault tolerance:** Failure of one node does not affect the overall system.
- **Complex management:** Requires sophisticated management and coordination.

Q6) What is hybrid web-services architecture? Identify the difference with other architectures and mention the applications.

Ans) Hybrid web-service architecture combines multiple architecture style in one. Often combining RESTful APIs , microservices and other Arch.

Difference with other architectures

- **Traditional Web Services (SOAP):** More complex and verbose, often requiring extensive configuration.

- **RESTful APIs:** Simpler and more lightweight, but may lack the rich feature set of SOAP.
- **Microservices:** Highly modular, but can be complex to manage and deploy.
- **Applications**
- **Enterprise Integration:** Integrating diverse systems and applications within an organization.
- **API-Driven Development:** Creating APIs for internal and external use

Q7) What is microservice architecture? Identify the need and applications.

Ans) A microservice architecture is a design pattern where a single application is broken down into smaller, independent services. Each service focuses on a specific business capability and communicates with others through well-defined APIs.

Need for Microservice Architecture:

- **Scalability:** Independent scaling of services based on their specific needs.
- **Flexibility**: Easier to adopt new technologies and frameworks for different services.
- **Resilience**: Isolated failures, minimizing impact on the overall system.

Applications of Microservice Architecture:

- **E-commerce:** Product catalog, order processing, payment processing, and shipping.
- **Financial Services:** Account management, trading, risk assessment, and fraud detection.
- **Social Media:** User profiles, news feeds, messaging, and notifications.

Q8) Identify and list the differences between Client-Server architecture and Service Oriented Architecture.

Ans)

Client-Server Architecture

- **Centralized Server:** A single server handles requests from multiple clients.
- **Direct Communication:** Clients communicate directly with the server.
- **Simple Structure:** Relatively straightforward architecture.
- **Limited Scalability:** Can be scaled by adding more servers, but it can be complex.
- **Tight Coupling:** Clients and server are tightly coupled, making changes to one often requiring changes to the other.

Service-Oriented Architecture (SOA)

- **Decentralized Services:** Multiple services, each with a specific function.
- **Loose Coupling:** Services communicate through well-defined interfaces, reducing dependencies.
- **Reusability:** Services can be reused across multiple applications.
- **Flexibility:** Easier to add, remove, or modify services without affecting the entire system.
- **Scalability:** Can be scaled by adding more instances of services or by using load balancing.
- **Complex Implementation:** Requires careful design and management.

Q9) Identify and list the differences between Distributed Internet architecture and Service Oriented Architecture.

Ans) Distributed Internet Architecture

- **Focus**: Network infrastructure and data distribution.
- **Granularity**: Components are typically smaller, focused on specific tasks.
- **Communication**: Relies on network protocols (HTTP, TCP/IP) for direct communication.
- **Coupling**: Components are loosely coupled, often using APIs or message queues.
- **Scalability**: Highly scalable, as components can be added or removed independently.
- **Fault Tolerance**: Built-in fault tolerance through redundancy, replication, and failover mechanisms.

Service-Oriented Architecture (SOA)

- **Focus**: Business processes and services.
- **Granularity**: Services are typically larger, encapsulating significant business capabilities.
- **Communication**: Often relies on a centralized Enterprise Service Bus (ESB) for message routing and transformation.
- **Coupling**: Services can be loosely coupled, but often rely on shared resources or infrastructure.
- **Scalability**: Can be scaled, but may have limitations.
- **Fault Tolerance**: Can be designed to be fault-tolerant, but often relies on the ESB for handling failures.

Q10) Identify and list the differences between Hybrid web-services architecture and Service Oriented Architecture.

Ans)

Hybrid Web Services Architecture

- **Combination of Styles:** Leverages a mix of architectural styles, such as SOAP, REST, and GraphQL.
- **Flexibility:** Can adapt to different use cases and evolving technologies.
- **Scalability:** Can scale both horizontally and vertically to accommodate varying workloads.
- **Performance:** Can optimize performance through caching, load balancing, and other techniques.
- **Security:** Implements robust security measures to protect sensitive data.

Service-Oriented Architecture (SOA)

- **Focus:** Business processes and services.
- **Granularity:** Services are typically larger, encapsulating significant business capabilities.
- **Communication:** Often relies on a centralized Enterprise Service Bus (ESB) for message routing and transformation.
- **Coupling:** Services can be loosely coupled, but often rely on shared resources or infrastructure.
- **Scalability:** Can be scaled, but may have limitations.
- **Fault Tolerance:** Can be designed to be fault-tolerant, but often relies on the ESB for handling failures.

Q11) Identify and list the primary differences between SOAP based and RESTful web services.

Ans)

SOAP (Simple Object Access Protocol)

- **XML-based:** Uses XML for both message format and protocol.
- **WSDL:** Requires a WSDL (Web Services Description Language) to define the service contract.
- **Stateful or Stateless:** Can be stateful or stateless.

- **Security:** Built-in security mechanisms like WS-Security.
- **Complexity:** More complex to implement and understand.
- **Performance:** Can be less performant due to XML parsing overhead.

REST (Representational State Transfer)

- **Resource-Based:** Treats data as resources identified by URIs.
- **Stateless:** Each request is independent, without server-side session state.
- **HTTP Methods:** Uses standard HTTP methods (GET, POST, PUT, DELETE) for operations.
- **JSON or XML:** Can use JSON or XML for data format.
- **Simplicity:** Simpler to implement and understand.
- **Performance:** Generally more performant due to lightweight protocols.