Thoms erl book

LAB-1

Name:- Gondaliya Aksh K.
Roll no:- CE082
Batch:- B1

Aim: Understanding the Evolution and comparison of different architectures.

1. What is monolithic architecture? Mention the advantages and disadvantages.

Monolithic architecture is a software design model where all application components are built as a single unit.

Advantages:-
It is easy for development for smaller applications
We need to trace or debug only single units
Inter-component communication is faster because it occurs within a single process,

Disadvantages:-
It's difficult to scale individual components; if we need to modify one component still we need to scale whole application
As the code becomes more messier it neet to hadle become very complex
If we want to update only one component then it also affect to other functional components also

2. What is client server architecture? Mention the advantages and disadvantages.

In client server architecture client request service from server and server manage client requests and resources.

Advantages:-

- Data and resources are only managed at server(centralized).it os easy to maintenance,backup etc.
- Provide security at only server level control
- Centralized control improves data integrity by ensuring that all clients are using the same, up-to-date data.

Disadvantages:-

- If the server goes down, all clients lose access to the services and data

- The system relies heavily on the network connection. A slow or failed network can lead to delays or errors for clients.
- Initial setup for the server is expensive because for powerful server

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

The main difference is that **single-tier architecture** combines the presentation, application, and data layers on a single machine, while **two-tier architecture** separates the client (presentation layer) from the server (data layer). Single-tier is suitable for local, standalone applications, whereas two-tier is better for small to medium client-server systems where the client and server are on different machines, allowing for direct communication between them

Single-tier architecture

- Description: All components (user interface, business logic, and data) reside on the same machine. The user interacts directly with the database on their local system.
- Applications:
    - Desktop applications like Microsoft Office, where all functions and data are on the user's computer.
    - Simple database applications used by a single user for practice or local development.

Two-tier architecture

- Description: The presentation layer (client) and the data layer (server) are on separate machines, with the client sending requests directly to the server for data.
- Applications:
    - Small to medium-sized systems like a local inventory management or employee attendance system.

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

Multi-tier client-server architecture is a client-server model that physically separates an application into multiple layers, such as the presentation tier (user interface), business

logic and a data tier (data storage). This approach enhances scalability, security, and manageability by isolating functions and allowing individual tiers to be modified or scaled independently without affecting others.

Advantages:-

- Individual tiers can be scaled independently
- The separation of concerns allows for modifications to one tier without affecting the others, making the application easier to update and maintain.

Application:-

- Common in web applications where a browser (presentation tier) interacts with a web server (presentation tier) which communicates with an application server (logic tier) and a database server (data tier).
- Online stores use this to handle transactions, manage product information, and store customer data securely.
- Modern cloud platforms often use multi-tier architectures to provide scalable and reliable services to users.

5. What is distributed internet architecture? Compare and contrast with other architectures and identify the applications.

Distributed internet architecture spreads a system's load across multiple interconnected nodes, rather than relying on a single central server. This contrasts with centralized architectures, which have a single point of control, and client-server models, where clients access a central server for data. Distributed systems offer enhanced scalability, fault tolerance, and availability, and are used in applications like cloud computing and blockchain

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

A hybrid web services architecture

combines elements of traditional **Service-Oriented Architecture (SOA)** and contemporary **microservices architecture**, along with potentially other styles like monolithic or serverless . This approach aims to leverage the strengths of different architectures to suit specific business and technical needs within a single, unified system

Feature	Hybrid Web Services Architecture	SOAP-based Architecture	REST-based Architecture
Protocols Used	Both SOAP + REST	Only SOAP	Only REST
Message Format	JSON + XML (both possible)	XML only	JSON/XML
Security	Uses SSL + WS-Security (when SOAP used)	Strong WS-Security support	Basic HTTPS, OAuth
Flexibility	High (choose best type per module)	Low (strict rules)	High (simple and flexible)

Performance	Balanced	Slow (heavy XML)	Fast (lightweight)
Best for	Enterprise + modern apps mix	Enterprise-critical apps	Web/mobile applications

7. What is microservice architecture? Identify the need and applications.

Microservice architecture is a style of building applications as a suite of small, independent, and loosely coupled services, each focused on a specific business capability. This approach is needed to build large, complex applications that require rapid development, flexibility, and independent scaling of components

Need for microservice architecture

- Allows development teams to build, test, and deploy new features faster and more frequently.
- Each service can be scaled independently based on its specific needs
- he failure of one service is less likely to bring down the entire application, as other services remain functional.

Applications:-

- E-commerce: An e-commerce platform can be broken into services for user accounts, product catalogs, order processing, payment, and shipping.
- Streaming services: Companies like Netflix use a microservices architecture to manage their vast streaming platform, handling everything from user authentication to video playback and recommendations.

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

Feature	Client–Server Architecture	Service-Oriented Architecture (SOA)
Basic Definition	Two-tier model where client requests and server responds.	Multi-tier model where independent services communicate over a network.
Structure	Client ↔ Server	Service Consumer ↔ Service Provider (via Service Bus/ESB)
Components	1. Client 2. Server	1. Services 2. Service Contract 3. Service Broker 4. ESB
Communication Style	Usually tightly coupled	Loosely coupled
Interoperability	Limited (same technology or platform)	Very high (services can be in Java, .NET, Python, etc.)
Scalability	Moderate; scaling is harder	Highly scalable due to distributed services
Reusability	Low; components are designed for specific client	High; services are reusable across multiple applications
Data Format	Often custom or binary formats	XML, JSON, SOAP, REST-based formats
Deployment	Centralized server	Distributed services over a network

Flexibility	Low; changes affect entire system	High; services can be changed independently
Security	Depends on the server	Strong security standards (like WS-Security)
Best Used For	Small or medium applications	Large enterprise-level applications

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

Granularity and Reusability:

- **Client-Server:** Typically involves a "fat client" that handles much of the application logic and presentation, interacting with a server primarily for data storage and retrieval. Servers are often built for specific client applications and may not be easily reusable.
- **SOA:** Emphasizes building fine-grained, independent services that encapsulate specific business functionalities. These services are designed for enterprise-wide reuse by multiple applications and clients.

Coupling:

- **Client-Server:** Often exhibits tighter coupling between the client and server, as the server's design may be closely tied to the specific needs of its client applications.
- **SOA:** Promotes loose coupling between services and consumers. Services are independent and communicate through well-defined interfaces, allowing for greater flexibility and easier modification or replacement of individual services without impacting others.

Communication:

- **Client-Server:** Communication is typically a direct request-response interaction between a client and a specific server.

- **SOA:** Communication often involves standardized messaging protocols (e.g., SOAP, REST) and may utilize a service bus or other middleware to facilitate interactions between services, including orchestration and choreography.

Focus:

- **Client-Server:** Primarily focuses on separating user interface and presentation logic from data management and processing.
- **SOA:** Focuses on creating a collection of interoperable services that can be discovered, consumed, and composed to build complex business processes across an enterprise.

Technology Heterogeneity:

- **Client-Server:** While it can support different technologies, tight coupling can make integrating disparate systems more challenging.
- **SOA:** Is explicitly designed to handle heterogeneous environments, allowing services implemented in different technologies to communicate and interact seamlessly through standardized interfaces.

Scalability and Flexibility:

- **Client-Server:** Scalability often involves scaling the entire server or adding more dedicated servers for specific clients.
- **SOA:** Offers greater flexibility and scalability by allowing individual services to be scaled independently based on demand, and new services can be easily integrated into the existing architecture.

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

Client-Server Architecture	Service-Oriented Architecture (SOA)
1. Two-tier architecture (Client + Server).	Multi-tier architecture of loosely-coupled services.
2. Tight coupling between client and server.	Loose coupling between independent services.

3. Communication is usually direct and synchronous.	Communication through service interfaces; can be synchronous or asynchronous.
4. Designed for a single application environment.	Designed for integrating multiple heterogeneous applications.
5. Limited reusability — client depends on server logic.	High reusability — services can be reused across systems.
6. Hard to scale as client and server are inter-dependent.	Easy to scale as services run independently.
7. Typically uses proprietary protocols (e.g., TCP/IP, ODBC).	Uses open web standards (SOAP, REST, WSDL, XML).
8. Not suitable for enterprise-level integration.	Ideal for enterprise integration and interoperability.

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

Feature	Hybrid Web-Services Architecture	Service-Oriented Architecture (SOA)
Definition	Combines SOAP + REST web services in one system	Entire system designed as a set of reusable services
Communication Style	Mixed (SOAP for secure tasks, REST for lightweight tasks)	Mostly SOAP-based in traditional SOA
Flexibility	Very high (choose best protocol per module)	High but mostly depends on SOAP standards
Standards	Uses web protocols, no strict service contracts	Strong standards: WSDL, WS-Policy, UDDI
Coupling	Loosely coupled	Loosely coupled
Focus	Practical implementation mix for performance + security	Architectural model for enterprise-level services
Message Format	XML + JSON	Mostly XML

Scalability	Very high (REST services scale well)	High (SOA scales by adding services)
When Used	When both legacy SOAP and modern REST systems exist	For enterprise-level distributed service apps

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

SOAP Web Services	RESTful Web Services
SOAP = Protocol.	REST = Architectural style.
Uses XML only.	Uses JSON, XML, HTML, plain text.
Heavyweight due to envelope, header, strict rules.	Lightweight and simple.
Requires WSDL for service description.	Uses URL endpoints, no WSDL required.
Built-in WS-Security, supports encryption, signatures.	Basic security (HTTPS), OAuth, JWT.
Supports stateful and stateless operations.	Stateless by design.
Slower due to XML parsing and strict structure.	Faster and more efficient.
Higher reliability, ACID transactions support.	Not suitable for ACID transactions.
Best for enterprise-level apps (banking, payments).	Best for web/mobile apps & public APIs.