UNIVERSITY OF SRI JAYEWARDENEPURA
Faculty of Technology
Department of Information and Communication Technology

ITC 3013 - Service Oriented Architecture & Web Services Assignment 01

Name: G.H.S. De Silva
Index: ICT/21/823

1. **Compare and contrast the Layered Pattern and the Client-Server Architecture in terms of their structure and ideal use cases.**

Aspect	Layered Pattern	Client-Server Architecture
Structure	Organized into horizontal layers, each with specific responsibilities *e.g., presentation* *business logic* *data access*	Divides the system into two main components: clients and servers. Clients request services, and servers provide them.
Ideal Use cases	Suitable for systems requiring separation of concerns and clear modularization e.g., enterprise applications web apps	Ideal for distributed applications where clients need access to shared resources or services (e.g., web services, database access).
Communication	Typically involves interaction between adjacent layers.	Communication occurs between clients and servers, often over a network.
Scalability	Can be challenging to scale as changes in one layer may affect others.	Scalability can be achieved by adding more servers or clients.
Performance	Performance can degrade if there are too many layers.	Performance depends on the network and server capacity.

2. **Discuss the strategic importance of performance, scalability, and security in the context of software architectural patterns. Provide examples of patterns that specifically address these issues.**

- **Performance:** Crucial for both user satisfaction and system efficiency. Different patterns that focus on performance are:
    - Caching Pattern: Increases efficiency by saving data accessed regularly.
    - Asynchronous Communication Pattern: Improves speed by separating message transmission from handling.

- **Scalability:** Guarantees the capability of the system to manage a higher workload. Patterns that focus on scalability encompass:
    - Microservices Architecture: Enables services to be scaled independently.
    - Load Balancing Pattern: Involves spreading incoming requests among several servers.

- **Security:** Involves preventing unauthorized access and data breaches. Security patterns involve addressing various security concerns.
  - Service Gateway Pattern: Offers a sole access point with security measures.
  - SSL Pattern: Guarantees secure transmission across networks

3. **Explain how the concept of high availability is implemented in the Master-Slave Pattern and give a real-world example of its application.**

   High availability in the master-slave pattern ensures continuous operation and minimal downtime of a system. It involves replicating data from a primary (master) server to one or more secondary (slave) servers. If the master server fails, a slave server can be promoted to master, ensuring data accessibility and system functionality.

   Real-World Example

   MySQL Replication

   In MySQL databases, master-slave replication is commonly used for high availability.

   - **Master Server:** Handles all write operations (INSERT, UPDATE, DELETE) and records them in a binary log.
   - **Slave Servers:** Continuously read the binary log from the master and execute the recorded SQL statements to replicate the master's data.
   - **Failover:** If the master server goes down, one of the slaves is promoted to become the new master. This can be automated with tools like MySQL Failover Manager.
   - **Load Balancing:** Read operations can be distributed across multiple slaves to balance the load and improve performance.

4. **In what ways does the Microservices Architecture pattern adhere to the Single Responsibility Principle, and what are the implications of this adherence for system maintainability and scalability?**

- **Adherence to SRP**
  - **Service Decomposition:** Involves designing each microservice to manage a particular business function while following the Single Responsibility Principle. This implies that each service is accountable for a specific aspect of the application's operation.
  - **Development and deployment autonomy**: Microservices allow for separate development, testing, and deployment to avoid impacting other services.

- **Implications for the Maintainability and Scalability of Systems:**
    - **Maintainability:** Each service's sole responsibility makes it simpler to comprehend, test, and uphold. Issues can be identified and resolved independently without impacting the rest of the system.
    - **Scalability:** Services can be independently scaled according to their unique load needs. For example, if the payment processing service faces increased traffic, it can be expanded independently from other services.

5. **Describe the historical evolution of API architectural styles, from RPC to GraphQL, highlighting the main advantages and limitations of each style.**

Style	Advantages	Limitations
RPC	Simple and efficient for small-scale applications.	Tight coupling and poor scalability.
SOAP	Robust and standardized, with built-in error handling.	Verbose and complex, leading to performance overhead.
REST	Lightweight, scalable, and stateless, with wide adoption.	Can be less suitable for complex transactions.
GraphQL	Flexible querying, reduces over-fetching and under-fetching.	Can be complex to implement and manage.

6. **Analyze the pros and cons of Monolithic Architecture compared to Microservices Architecture in terms of fault tolerance, deployment, and response time.**

Aspect	Monolithic Architecture	Microservices Architecture
Fault Tolerance	A failure in one component can affect the entire system.	Isolated failures, as each service operates independently.
Deployment	Simple to deploy as a single unit.	Complex deployment due to managing multiple services.
Response Time	Generally faster within a single application context.	Potential overhead due to inter-service communication.

7. What is a Web Service and how does it support interoperability between different software applications?

A web service is a uniform method of combining web applications using open standards over an internet protocol backbone. By using XML for all communication, web services enable applications from different sources to interact without the need for time-consuming custom coding, allowing them to communicate independently of any specific operating system or programming language.

- Interoperability refers to the ability of different systems or applications to communicate and work together effectively.
- **Standardized protocols**, such as HTTP, SOAP, and REST, are employed by web services to facilitate communication among various systems.
- **XML and JSON** are commonly used to exchange data, regardless of the programming language being used, to allow for easy understanding and processing by different systems.
- Web services commonly utilize **WSDL or Swagger** to define available operations and data formats, facilitating integration between systems.
- **Decoupling:** Web services create a level of separation that disconnects the client from the server. This separation enables various systems to develop autonomously while still being able to communicate.

8. Explain the key differences between Type I and Type II web services.

Aspect	**Type I Web Services**	**Type II Web Services**
Protocol	Uses SOAP protocol.	Uses HTTP protocol.
Message Format	XML-based messages.	Can use JSON, XML, or other formats.
Complexity	More complex with extensive standards.	Simpler and more flexible.
State Management	Typically stateless but can be stateful	Stateless
Security	Built-in WS-Security for message-level security.	Relies on HTTPS for transport-level security.
Transport	Can operate over various protocols (HTTP, SMTP).	Operates exclusively over HTTP/HTTPS.
Tooling Support	Mature and widely supported in enterprise environments.	Lightweight and easily integrated with web technologies.

9. What role does the Service Registry play in SOA

A service registry in Service-Oriented Architecture (SOA) functions as a centralized directory where services are listed and can be located by other system components.

- **Service Discovery:** Enables clients to dynamically locate and invoke services without hardcoding their endpoints.
- **Decoupling:** Facilitates loose coupling of services, allowing clients to find services at runtime instead of during the design phase.
- **Load Balancing:** Provides details about multiple service instances, helping clients to evenly distribute requests.
- **Version Management:** Manages different versions of services, ensuring clients can access compatible versions.
- **Metadata Repository:** Maintains information about services, including contracts, policies, and documentation, which supports service governance and management.

10. How does SOA provide a cost-effective solution for enterprises with existing investments in legacy systems?

SOA provides a structure for combining different systems, such as older systems, into a cohesive environment focused on services.

Cost-Effective Solutions

- **Incremental Integration:** Allows organizations to gradually incorporate legacy systems without completely replacing the entire system. Services can be built based on current features, aiding in gradual transition and upgrading.
- **Utilizing current investments**: Enables older systems to make their functions available as services, allowing for the reutilization of existing business logic and data without the necessity to overhaul or substitute the entire system.
- **Interoperability:** Achieved by employing open standards such as XML, SOAP, and WSDL to guarantee that old and new systems are compatible, reducing the necessity for custom integration code.
- **Cutting Down Maintenance Expenses**: Converting old system functions into services makes maintenance easier, as these services can be modified or replaced without affecting the applications that rely on them.
- **Flexibility and Scalability:** The modular design of SOA enables businesses to grow and adjust their IT setup according to changing business demands, without the need for major investments in new systems.