NJC LABS ACCELERATOR PROGRAM

SESSION 2 NOTES

Client-Server

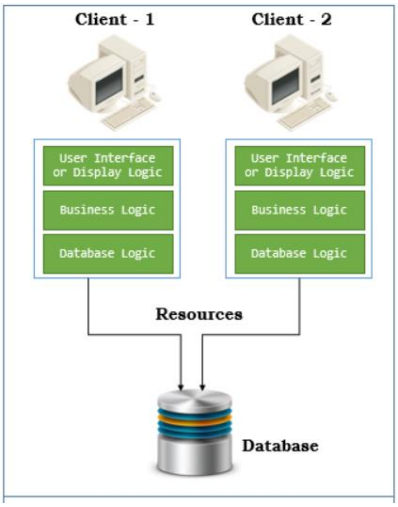
1. **Tell us about the features of client/server.**
   In a client-sever architecture, a client requests a server for a resource and the server provides the necessary resources. A server might serve several clients at a time while a client can only communicate with one server at a time.
   Features:
   - A server is a machine or group of machines where the application/business logic, data or both reside the same server.
   - A client-server architecture works with a system of request and response. A client sends a request to the server for an information and the server responds to the client with the required information.
   - The client and server share a common communication protocol to be followed to interact with each other.
   - A server can only accommodate a certain number of requests from clients, so it prioritises the requests.
2. **What is a Web server in a client server environment?**
   A web server holds several web pages or websites in a client/server environment. A client sends an internet request to a web server through a communication protocol like HTTP request and the web server reads the request to respond with the appropriate web page or webpage component files.
3. **What is the role of the presentation layer?**
   The presentation layer is responsible to provide the formatted representation of the data provided by the server.
4. **They say this architecture is secure, how is it done in your opinion?**
   The client/server architecture has three main components – client, server and the network. The client/server architecture can be secured at any of the three components. At the client side, a secure system can be ensured by secure software packages, password management, user identity and gateway security. At server side, the architecture can be secured by locating the servers in secure, access-controlled environments. Server security is the controlling of access. There are other client-server strategies that can be following such as endpoint security, firewalls, anti-virus software and anti-spyware software.
5. **What is a Database Server in a client server environment?**
   Database server is the term used to refer a back end system of a database application using a client-server architecture where all the data of the system resides, and tasks like data analysis, storage, data manipulation, archiving and other data-related tasks are performed.
6. **What are Super servers in client server environments?**
   In a large system, there are billions of clients that try to communicate with the server which might need millions of servers. To accommodate several servers, client-server architectures have super servers that can serve many clients simultaneously within a single process. Instead of separate server processes for each connection it uses threads of a single process and pools

the database cache buffers for use by all connections. The super servers start the other servers when necessary and do not use a lot of resources when in idle state.

7. Explain 2-Tier and 3-Tier architecture

Two-tier architecture is based on client server architecture. The direct communication happens between the client and server with no other layers in between. The two-tier architecture is divided into two parts:

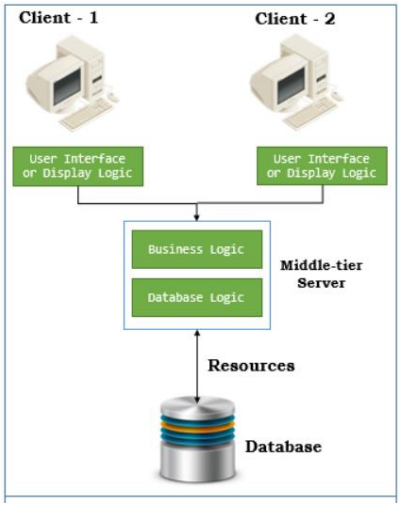
1. Client tier – It holds the code to save data from the data from database server. It is responsible to display logic, hold business logic and database logic.
2. Data tier – This is the database server which holds all the data.



2-tier Architecture

Three-tier architecture typically comprise a presentation tier, a business or data access tier, and a data tier. Three layers in the three-tier architecture are as follows:

1. Client layer – This is like the presentation layer which holds the code to display the data received from the server. This layer is responsible to send a request to the server and receives a static or dynamic content as response.
2. Business layer – This holds the business data and business logic like validation, calculation, processing, and data insertions.
3. Data layer – This layer is like the external resource like the database that accesses the data, inserts, updates, deletes data from data source.



3-tier Architecture

8. **What is a File server?**
   A file server is a central server in a computer network that provides file systems or at least parts of a file system to connected clients. File servers therefore offer users a central storage place for files on internal data media, which is accessible to all authorized clients. File servers are connected to internet and configured such that users can access files from local network as well as remote access.

SOA & MicroServices

1. **What are the main benefits of SOA?**
   Service Oriented Architecture is a combination of enterprise service bus(ESB) and application services which orchestrates the ESB to consume the services and provide other components by application components.
   Benefits:
   - Standardized Service Contract
   - Loose coupling
   - Abstraction
   - Reusability
   - Autonomy
   - Discoverability
   - Composability

2. **How can you achieve loose coupling in SOA?**
   Loosely coupling in SOA is how the services are implemented without affecting other services and applications. All the services and components in the server are implemented independently for the client application to request through an internet communication protocol and consume it without being affected by any other client request, service, or component.

3. **Are web services and SOA the same?**
   Service Oriented Architecture is a composition of independent services which encapsulate business functionality and expose it as a service. Web services are the most common way of realizing a SOA, however, they are not the same all the time. Web services are software systems designed to support interoperable machine-to-machine interaction over a network. An implementation of web service does not indicate the realisation of SOA in the system.

4. **What is a reusable service?**
   A reusable service is a functionality built independent of any specific project, consumer need, business or technology alongside a core service logic. These service components can be shared with other projects, developers or systems to be used in their systems without repeating the same work and instead focus on tending to other important tasks like designing an architecture, building unique or work intensive services. Reusable services allow flexibility and more productivity during the development of an application. For example, calculating an interest amount on a principle amount of money can be a reusable service in a financial management system.

5. **What are the disadvantages of SOA?**

- SOA may not always be the best architectural option as the optimal utilization of its services need an overhead development and design attempts that incur more costs.
- In view of the benefits that can be achieved by SOA, higher volume of service consumers may be expected which raises the problem of changes management and versioning control problems.
- SOA needs a very good infrastructure system to serve the quantity of consumers and the maintain the system, which again may incur lot of cost.
- To send and receive so many web pages, services and information from the server, the architecture may often need a high-speed server with a lot of bandwidth.
- Also, in SOA all inputs are validated prior to sending to the server. It involves using many services, thus overloading system with extra computation.

6. **What is ESB and where does it fit in?**
   An Enterprise Service Bus (ESB) is a communication system between mutually communication software applications or services in a SOA. In SOA, no component is necessarily only a client or a server. The motivation for the development of the architecture was to find a standard, structured, and general purpose concept for describing implementation of loosely coupled software components (called services) that are expected to be independently deployed, running, heterogeneous, and disparate within a network.

7. **In SOA do we need to build a system from scratch?**
   It is not necessary to build a system from scratch in SOA. It can utilise reusable service or micro-services or be created by exposing functions from legacy code.

8. **What is the most important skill needed to adopt SOA? technical or cultural?**
   It is necessary to have enough technical skills to be able to build a system using SOA as well as be able to use the reusable services to be consumed by systems built on other platforms or programming languages. Also, it is necessary for a team to have the cultural skills to adopt the features of SOA, especially the reusable services. Not a lot of developers or coders welcome the idea of using reusable service or expose a functionality of a legacy system.

9. **List down the advantages of Microservices Architecture.**
   Benefits of Microservices Architecture are:
   - Agility Microservices provide more agility and makes it easier to pivot through segments of the application quickly.
   - Small focused teams – Small focused teams can work on specific unique crucial services.
   - Small code base – IT allows to easily replicate, remove, scale or manage the application without impacting the entire application. If an error occurs only a small piece of code is affected.
   - Mix of Technologies - Microservices can be created on or consumed by an application built on any platform, language, architecture or infrastructure.
   - Fault Isolation - With a small code base and several microservices, it is easy to detect what exactly caused a fault in a system.

10. **What are the best practices to design Microservices?**

The following are the best practices to be followed while designing a Microservices architecture-based application:

- Domain Driven Design – Split a complex large problem into small sized, autonomous, independent microservices modelled around business domain.
- Separate Data storage – Data should be made private to other microservices. An access to any data should only through APIs.
- Build separate teams for different microservices - Teams should be divided based on microservices with one team working on one microservice.
- Automate enough for independent deployment - Nicely designed micro-services should be able to be deployed independently. And, build and release automation would enhance the deployment process thereby leading to quicker releases and shorter overall lead time.
- Failure isolation: Microservices-based architecture should consider adopting isolation of failure with independent microservices. Architecture principles and design patterns such as some of the following would help achieve the same:
    - Circuit-breaker design pattern
    - Asynchronous communication
    - Loose coupling
    - Event-driven architecture
    - Stateless design
    - Self-contained services
    - Timeouts

11. How does Microservice Architecture work?

The whole idea of microservices architecture is that it is easy to build and maintain an application seamlessly when broken down into smaller pieces. A client sends a request through API Gateway for a specific microservice in the independent microservices pool. These services can be consumed by any other service or be updated, maintained or deleted without disrupting the entire application.

12. What are the pros and cons of Microservice Architecture?

Pros:

Microservices Architecture are very popular due to the benefits that come with it. It is easy to develop, deploy and maintain each microservice on a different platform, using different programming languages and developer tools. Microservices are independent of each other and interact through APIS and communication protocols. As each microservice can be developed, maintained, updated, or deleted independently without affecting the application entirely, it is also easy to scale them separately. This leads to better agility. Also, the isolated nature provides better fault tolerance.

Cons:

As there is a lot of interaction within the components of the architecture, communication may be difficult until automated and advanced technologies are used. This leads to poor performance and a lot of overhead. Having to maintain a network leads to other issues. What we gain on the simplicity of single responsibility microservices, is lost on the complexity of the network. Due to the mix of technologies, collaboration may be a little difficult between teams.

13. **What is the difference between Monolithic, SOA and Microservices Architecture?**

    A Monolithic architecture consists of a database, client-side user interface and a server-side application. All the components are unified into a single component and all its functions are managed in one place. This approach is widely accepted by small teams for smaller projects.

    Service Oriented Architecture (SOA) is an architectural style that consists of discreet and loosely coupled components that utilise the services offered by other components through a common communication protocol. The components are integrated seamlessly and can be reused.

    Microservices architecture is a style of architecture where a large, complex system is split into smaller microservices that are developed and deployed independently. These services can be updated, consumed or deleted without impacting other services or the entire application. The small nature of services makes it easy for fault detection and correction as well as better for scaling.

14. **What are the challenges you face while working Microservice Architectures?**
    - As the number of microservices built increases, managing them gets more challenging. A good managing plan for services must be decided before building or while building the services.
    - Monitoring and tracking services or requests maybe difficult with so many services and messages. However, if they are not monitored or diagnosed correctly, it may be difficult to detect the cause of fault in case of occurrence.
    - It is necessary to embrace DevOp culture to iteratively release microservices and not delay any releases as all services are independent and loosely coupled.
    - Robust resiliency testing is key to successful fault tolerance.
    - Testing is much more complex in microservices due to different services, integration, and interdependencies. It may be very difficult to test all the services due to the asynchronous aspect.
    - While it may be counter-intuitive, expecting failure scenarios and building a robust set of microservices is imperative to a successful implementation.
15. **What are the characteristics of Microservices?**

    The following are the characteristics of Microservices:
    - *Componentization via Services* - A component is a unit of software that is independently replaceable and upgradeable. The primary way of componentizing a Microservices architecture is by services, as they are independently deployable.
    - *Organized around business capabilities* – Unlike functional based, microservices approach to division of large complex project is different, splitting up into services organized around business capability. This ensures joint collaboration of team members from different business areas.
    - *Products not Projects* – The microservice approach believes in quality product by during the final release and not very focused on a strict timeline to finishing an application as a project.
    - *Decentralized Data Management* – Microservices prefer letting services maintain their own database, either instances of the same database or a different one.
    - *Design for failure* - A consequence of using services as components, is that applications need to be designed so that they can tolerate the failure of services.