# Unit 1 Questions

## Briefly explain Data management in Enterprise system and its components and best practices for the same.

Data management in an enterprise system refers to the process of organizing, storing, protecting, and maintaining data to ensure its accuracy, availability, and reliability. It involves the use of various components and best practices to ensure that data is effectively managed throughout its lifecycle.

Components of Data Management in Enterprise Systems:

1. Data Architecture: This involves the design and organization of data structures, data models, and data flows within an enterprise system.
2. Data Governance: This involves the policies, procedures, and processes for managing and protecting data within an enterprise system. It also includes the roles and responsibilities of stakeholders in managing data.
3. Data Quality Management: This involves ensuring that data is accurate, complete, and consistent, and conforms to defined business rules.
4. Master Data Management: This involves creating and maintaining a single, authoritative source of master data that is used throughout an enterprise system.
5. Metadata Management: This involves capturing and managing metadata, which provides context and meaning to data.

Best Practices for Data Management in Enterprise Systems:

1. Establish data governance policies and procedures that align with the organization's overall goals and objectives.
2. Use a data catalog or inventory to identify and manage data assets.
3. Implement data quality controls to ensure that data is accurate, complete, and consistent.
4. Implement data security and privacy controls to protect sensitive data.
5. Establish a master data management program to ensure that master data is consistent and accurate across the enterprise.
6. Use metadata management to provide context and meaning to data.
7. Implement a data retention policy to ensure that data is retained for the appropriate period of time.
8. Ensure that data management practices are regularly reviewed and updated to reflect changes in business needs and regulatory requirements.

## Describe in detail workflow in a business process and its types and advantages

Workflow in a business process refers to the sequence of tasks and activities that are performed to achieve a specific business objective. It involves the movement of information, documents, or tasks from one step to another in a systematic and structured manner. Workflows can be automated or manual, and can be categorized into several types, including sequential, parallel, and hybrid workflows.

Types of Workflows:

1. Sequential Workflow: In a sequential workflow, tasks and activities are completed in a linear or sequential manner, where the output of one task becomes the input of the next. This type of workflow is best suited for processes that have a clear and predefined sequence of steps.
2. Parallel Workflow: In a parallel workflow, multiple tasks and activities are performed simultaneously. This type of workflow is best suited for processes that require collaboration or involve multiple departments or individuals.
3. Hybrid Workflow: A hybrid workflow combines sequential and parallel workflows to achieve a specific business objective. This type of workflow is best suited for processes that have both sequential and parallel elements.

Advantages of Workflow in Business Processes:

1. Increased Efficiency: Workflow automates and streamlines processes, reducing the time and effort required to complete tasks and activities. This can lead to increased productivity and reduced costs.
2. Improved Collaboration: Workflows facilitate collaboration among team members and departments, allowing for better communication and coordination.
3. Better Visibility: Workflows provide real-time visibility into the status of tasks and activities, enabling managers to identify bottlenecks and areas for improvement.
4. Improved Compliance: Workflows can ensure that processes are compliant with industry standards and regulations, reducing the risk of penalties and fines.
5. Enhanced Customer Service: Workflows can improve customer service by providing faster response times and more accurate information.

In summary, workflows are an essential aspect of business processes, enabling organizations to automate and streamline tasks and activities, improve collaboration, increase efficiency, and enhance customer service. The choice of workflow type depends on the nature of the process and the business objectives that need to be achieved.

## Explain ERP in detail with help of a suitable diagram and its advantage and disadvantage

Enterprise Resource Planning (ERP) is a type of software application that integrates and manages all the core business processes and functions within an organization, such as finance, human resources, inventory management, supply chain management, and customer relationship management. The primary objective of an ERP system is to provide a centralized and real-time view of the organization's data and processes to facilitate decision-making and improve operational efficiency.

Diagram: Here is a high-level diagram that depicts the various modules and components of an ERP system:



Enterprise resource planning (ERP) refers to a type of software that organizations use to manage day-to-day business activities such as accounting, procurement, project management, risk management and compliance, and supply chain operations.

### Advantages of ERP:

1. Improved Efficiency: An ERP system can streamline and automate routine tasks, reducing manual effort and increasing operational efficiency.
2. Enhanced Collaboration: ERP enables better collaboration among departments and functions by providing a unified view of data and processes, improving communication and coordination.
3. Increased Visibility: ERP provides real-time access to data and metrics, enabling managers to make informed decisions based on accurate information.
4. Standardization: ERP promotes standardization of processes and procedures across departments, reducing variability and improving consistency.
5. Scalability: ERP is highly scalable, allowing organizations to add new modules and functionality as needed to support growth and expansion.

### Disadvantages of ERP:

1. High Cost: ERP systems can be expensive to implement, requiring significant investment in hardware, software, and customization.
2. Complexity: ERP systems can be complex and require significant effort to configure and customize to meet the organization's needs.
3. Resistance to Change: Employees may resist using a new system, which can impact user adoption and affect the overall success of the implementation.
4. Integration Challenges: Integrating an ERP system with other applications and systems can be challenging, requiring significant effort and resources.
5. Dependence on Vendor: Organizations may become dependent on the ERP vendor for support and upgrades, limiting flexibility and autonomy.

## Discuss Supply Chain Management (SCM) & its types of models.

Supply Chain Management (SCM) refers to the management of the flow of goods, services, information, and finances across the entire supply chain, from the suppliers to the end customers. The primary goal of SCM is to optimize the entire supply chain to achieve maximum efficiency and profitability while minimizing costs and risks.

There are several types of SCM models, including:

1. **Push Model**: In a push model, the production and delivery of goods and services are based on anticipated demand or forecasts. This model is commonly used in industries with stable demand patterns, such as grocery stores.
2. **Pull Model**: In a pull model, the production and delivery of goods and services are based on actual demand. This model is commonly used in industries with highly variable demand patterns, such as fashion.
3. **Lean Model**: In a lean model, the focus is on reducing waste and improving efficiency throughout the supply chain. This model is based on the principles of lean manufacturing and aims to minimize inventory, reduce lead times, and improve quality.
4. **Agile Model**: In an agile model, the focus is on quickly responding to changing customer demands and market conditions. This model requires a flexible and responsive supply chain that can adapt to changes in demand and supply.
5. **Resilient Model**: In a resilient model, the focus is on mitigating risks and disruptions in the supply chain. This model requires a robust and redundant supply chain that can quickly recover from disruptions such as natural disasters or supply chain disruptions.

### Advantages of SCM:

1. Improved Efficiency: SCM can improve the efficiency of the supply chain by reducing lead times, minimizing inventory, and improving quality.
2. Cost Savings: SCM can reduce costs by optimizing the use of resources and improving productivity.
3. Improved Customer Service: SCM can improve customer service by reducing lead times, improving product quality, and providing faster delivery times.
4. Improved Collaboration: SCM can improve collaboration among suppliers, manufacturers, distributors, and customers, improving communication and coordination.
5. Reduced Risk: SCM can reduce the risk of supply chain disruptions and improve the ability to respond to unforeseen events such as natural disasters or market fluctuations.

## Elaborate Customer Relationship Management (CRM) & Components and types.

Customer Relationship Management (CRM) is a business strategy that focuses on managing interactions with customers to improve customer satisfaction, loyalty, and retention. The primary goal of CRM is to create long-term relationships with customers and increase their lifetime value.

Components of CRM:

1. Customer Data Management: CRM requires a robust and centralized customer database that stores all customer-related data, including personal information, transaction history, purchase behavior, and contact details.
2. Sales Automation: CRM systems can automate sales-related tasks, including lead generation, lead qualification, opportunity tracking, and sales forecasting.
3. Marketing Automation: CRM systems can automate marketing-related tasks, including email marketing, social media marketing, and lead nurturing.
4. Customer Service and Support: CRM systems can manage customer inquiries, complaints, and support tickets, providing a consistent and high-quality customer experience.
5. Analytics and Reporting: CRM systems provide analytics and reporting tools that enable organizations to measure the effectiveness of their customer engagement strategies, identify trends and patterns, and make data-driven decisions.

Types of CRM:

1. Operational CRM: Operational CRM focuses on improving customer-facing processes and optimizing customer interactions, such as sales, marketing, and customer service. Operational CRM aims to improve efficiency, effectiveness, and customer satisfaction.
2. Analytical CRM: Analytical CRM focuses on analyzing customer data to gain insights into customer behavior, preferences, and needs. Analytical CRM uses data mining, predictive modeling, and other analytical techniques to identify patterns and trends and provide actionable insights for improving customer engagement strategies.
3. Collaborative CRM: Collaborative CRM focuses on improving collaboration and communication among internal teams and external partners to improve customer interactions. Collaborative CRM aims to provide a seamless and consistent customer experience across all touchpoints.

Advantages of CRM:

1. Improved Customer Experience: CRM systems can provide a personalized and seamless customer experience across all touchpoints, improving customer satisfaction and loyalty.
2. Increased Sales: CRM systems can improve sales effectiveness by automating sales-related tasks, tracking customer interactions, and providing insights into customer needs and preferences.
3. Improved Marketing ROI: CRM systems can improve marketing ROI by automating marketing-related tasks, tracking customer behavior, and providing insights into the effectiveness of marketing campaigns.
4. Improved Customer Retention: CRM systems can improve customer retention by providing a high-quality customer experience, addressing customer needs and concerns promptly, and providing personalized offers and promotions.
5. Improved Collaboration: CRM systems can improve collaboration and communication among internal teams and external partners, improving the efficiency and effectiveness of customer-facing processes.

## Illustrate Product Life Cycle & its stages along with its advantages and limitations

Product life cycle (PLC) is the progression of a product through its lifespan, from its introduction to the market until it is eventually withdrawn. The product life cycle is typically represented by a curve with four stages: Introduction, Growth, Maturity, and Decline.

1. Introduction Stage: In this stage, the product is introduced to the market. Sales are low, and the company may incur losses due to the high costs of developing and launching the product. Advertising and promotional activities are focused on creating awareness and generating interest among potential customers.
2. Growth Stage: In this stage, sales increase rapidly, and the product gains acceptance in the market. Competition may increase, and the company may need to invest in additional production capacity to meet the growing demand. Advertising and promotional activities are focused on building brand loyalty and increasing market share.
3. Maturity Stage: In this stage, sales growth slows down, and the product reaches its peak. The market becomes saturated, and competition becomes intense. The company may need to reduce prices, increase promotions, or introduce product improvements to maintain market share.
4. Decline Stage: In this stage, sales decline, and the product is eventually withdrawn from the market. The decline may be due to changes in customer preferences, technological advances, or the emergence of new products that offer better value.

Advantages of Product Life Cycle:

1. Helps companies to plan and allocate resources based on the stage of the product.
2. Provides insights into the sales and profitability of the product.
3. Allows companies to identify potential problems and develop strategies to overcome them.
4. Helps companies to understand the dynamics of the market and competition.

Limitations of Product Life Cycle:

1. Assumes that products follow a predictable and uniform life cycle, which may not be true in all cases.
2. Does not take into account the impact of external factors such as changes in technology, regulations, or economic conditions.
3. May not accurately reflect the sales and profitability of the product due to variations in product quality, price, and distribution.
4. May not consider the impact of marketing activities, such as advertising and promotion, on the life cycle of the product.

## Briefly explain Human Resource Management (HRM)

Human Resource Management (HRM) refers to the management of human capital within an organization. It involves recruiting, hiring, training, and managing employees to achieve organizational goals. HRM also includes developing and implementing policies and procedures that ensure the fair treatment of employees and compliance with legal requirements.

The primary functions of HRM include:

1. **Recruitment and selection**: Attracting and selecting the right people for the job.
2. **Training and development**: Providing employees with the necessary knowledge and skills to perform their job effectively.
3. **Performance management**: Evaluating and managing employee performance to achieve organizational goals.
4. **Compensation and benefits**: Determining and administering employee compensation and benefits packages.
5. **Employee relations**: Maintaining positive relationships between employees and management and addressing workplace issues and conflicts.
6. **Compliance with legal requirements**: Ensuring that the organization complies with employment laws and regulations.

HRM is essential for the success of any organization, as it helps to ensure that the right people are in the right positions and that they are equipped with the necessary skills and knowledge to perform their jobs effectively. It also helps to create a positive workplace culture that fosters employee engagement, retention, and productivity.

## What are General Ledger Systems?

A General Ledger System (GLS) is a financial accounting software system used by organizations to maintain and manage their financial transactions. It provides a centralized repository of all financial transactions, which can be used to generate financial reports and analyses financial performance.

The primary function of a GLS is to record and classify financial transactions into specific categories such as assets, liabilities, revenues, expenses, and equity. The GLS also provides a mechanism for creating and managing journal entries, which are used to adjust the financial records for any errors or omissions.

### The key features of a GLS include:

1. Chart of Accounts: A predefined list of accounts used to classify financial transactions.
2. Journal Entry: A mechanism to record financial transactions and adjust the financial records.
3. Trial Balance: A summary of all the account balances to ensure that the total debits and credits are equal.
4. Financial Reporting: A mechanism to generate financial statements such as income statements, balance sheets, and cash flow statements.
5. Audit Trail: A record of all financial transactions to track any changes made to the financial records.

### The benefits of using a GLS include:

1. Increased accuracy and efficiency: A GLS provide a centralized repository for financial transactions, which reduces the likelihood of errors and makes the process more efficient.
2. Better financial reporting: A GLS provides a mechanism for generating accurate and timely financial reports, which are essential for decision-making.
3. Improved compliance: A GLS ensures compliance with accounting standards and regulations, reducing the risk of penalties and legal issues.
4. Enhanced decision-making: A GLS provides accurate and timely financial information, which can be used to make informed decisions about the organization's financial performance.

# Unit 2 Questions

## Explain Distributivity and how to secure distributed enterprise?

Distributivity refers to the property of a system or process that is distributed across multiple locations or nodes. In a distributed system, the processing power, storage, and other resources are spread across multiple nodes, which enables better scalability, fault tolerance, and performance.

However, securing a distributed enterprise can be challenging due to the following reasons:

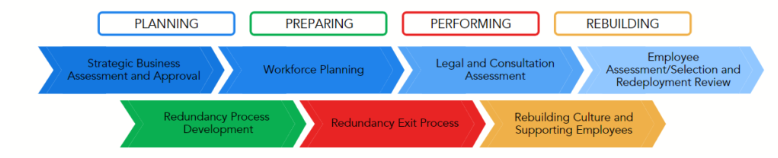
1. Increased attack surface: A distributed enterprise has a larger attack surface as it involves multiple nodes and communication channels, making it vulnerable to various cyber threats such as hacking, malware, and phishing attacks.
2. Complex network topology: A distributed enterprise involves a complex network topology, which can be difficult to manage and secure. This complexity makes it challenging to ensure proper authentication, access control, and data protection.
3. Data synchronization and consistency: In a distributed enterprise, data needs to be synchronized across multiple nodes to ensure consistency. However, this synchronization can be difficult to achieve and can result in data inconsistencies, which can compromise data integrity and security.

To secure a distributed enterprise, the following best practices can be followed:

1. Network segmentation: Segregate the network into smaller, more manageable segments to reduce the attack surface and limit the impact of any security breaches.
2. Authentication and access control: Implement strong authentication and access control mechanisms such as multi-factor authentication, role-based access control, and least privilege access to ensure that only authorized users can access sensitive data and resources.
3. Data encryption: Use strong encryption algorithms to protect sensitive data and ensure that data is protected both at rest and in transit.
4. Continuous monitoring and threat detection: Implement continuous monitoring and threat detection mechanisms to detect and respond to security threats in real-time.
5. Disaster recovery and business continuity planning: Develop and implement disaster recovery and business continuity plans to ensure that the organization can recover quickly in case of any security breaches or other disruptions.

## What is a redundancy process? What is the best redundancy process to follow?

A redundancy process is a process where an organization reduces its workforce by terminating the employment of one or more employees due to reasons such as financial constraints, changes in business strategy, or restructuring. Redundancy can be a difficult process for both the employees and the organization, and it is essential to follow a fair and transparent process to minimize the impact on the affected employees and maintain the organization's reputation.



### **Strategic Business Assessment and Approval**

1. Inspect the current state of your business / organisation.
2. Develop a map of your business goals and strategy to get a clear vantage point.
3. When working for a larger organisation, investigate if you are eligible to run the redundancy. If you belong to a smaller business, understand the impacts on your budget.
4. In some cases, running the redundancy requires other corporate members.

### **Workforce Planning**

* Analyse your talent against your current goals.
* Consider the future and the skills and experience you have to meet your future strategy.
* Identify the gaps in your workforce and assess the external market.
* Determine the solution that suits your needs and understand if redundancies are required.

### **Legal and Consultation Assessment**

* Assess legal requirements and ensure you know your obligations under the law.
* Assess Industrial Relations/Employee Relations Risks and Union consultation or collaboration requirements under any enterprise agreements.
* Ensure that the people involved sign confidentiality statements and understand the implications of breaching these.

### **Employee Assessment/Selection and Redeployment Review**

* Determine if you even need to run a selection process. If your redundancy program involves reducing the number of people performing the same roles, then the answer will be yes.
* Carry out a sound assessment process—include evidence requirements, relevant training for leaders and assessment procedures.

### **Redundancy Process Development**

* Develop a redundancy and exit process, considering the most dignified approach for impacted employees.
* Develop templates, tools, scripts and training that may be required to ensure consistency and professionalism of your program.
* Using scripts to support all redundancy meetings will ensure conversations stay factual, objective and free of bias.

### **Redundancy Exit Process**

Distribute communications to all relevant stakeholders in line with your communication plan. Consider the following:

* Appropriate Dates and Venue
* Communication to Employees
* Structure of Meetings
* Employee Support
* Exit Checklist
* Government Notification

### **Rebuilding Culture and Supporting Employees**

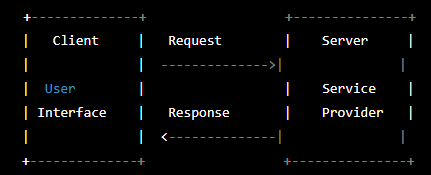
* Communicate to the workforce about the end of the process
* Acknowledge and honour those that left and create opportunities for people to talk
* Monitor individuals to ensure mental health is intact—be aware of survival guilt
* Coordinate employee assistance program (EAP) or mental health support. Initiate engagement activities over the following weeks

## Elaborate client server architecture with suitable diagram and its type

Client-server architecture is a computing model in which clients request services or resources from servers, which provide the requested services or resources. This architecture involves a distributed computing model where the processing power and resources are split between the client and server.

The client-server architecture consists of the following components:

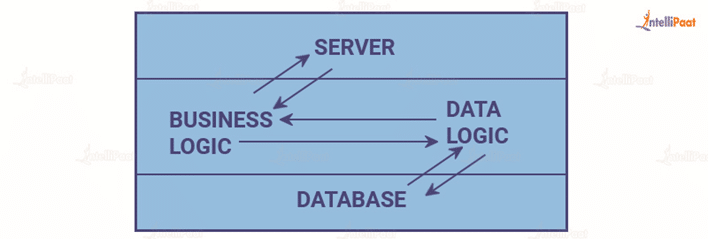
1. Client: A client is a device or software application that requests services or resources from the server. Clients can be desktop applications, mobile applications, web browsers, or any other device that can connect to a network.
2. Server: A server is a device or software application that provides services or resources to clients. Servers can be web servers, file servers, database servers, or any other device that can respond to client requests.
3. Network: The network connects clients and servers and enables communication between them. Networks can be local area networks (LANs), wide area networks (WANs), or the internet.
4. Protocol: A protocol is a set of rules that governs communication between clients and servers. The most common protocols used in client-server architecture are HTTP, TCP/IP, and SMTP.



### **Types of client server architecture**

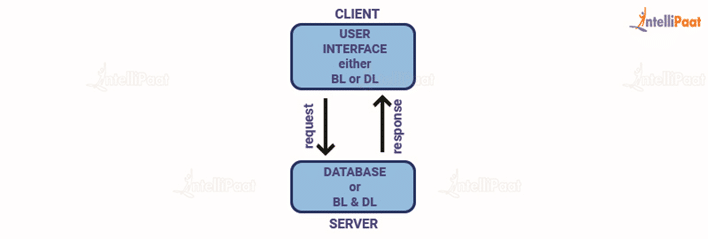
The functionality of client server architecture is in various tiers.

#### **1-tier architecture**



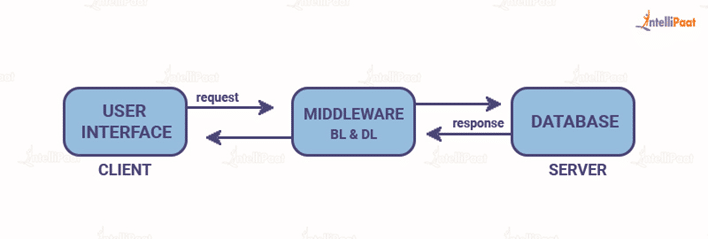
In this category of client server architecture, the architecture contains all kinds of settings, such as configuration setting and marketing logic, on a single device.  While the diversity of services offered by 1-tier architecture makes it one of the reliable sources, handling such an architecture is difficult. This is primarily due to the data variance.  It often results in replication of work. 1-tier architecture consists of several layers, such as presentation layer, business layer, and data layer, that are combined with the help of a unique software package. The data present in this layer is usually stored in local systems or on a shared drive.

#### **2-tier architecture**



his architecture has the best environment. In this architecture, the user interface is stored on the client’s side and the database is stored on the server, while database logic and business logic is maintained either on the client’s side or on the server’s side.  
  
The 2-tier architecture is faster in comparison to the 1-tier architecture; this is because the 2-tier architecture does not have any intermediary between the client and the server. It is often utilized to avoid confusion between clients. One of the popular examples of 2-tier architecture is the online ticket reservation system.

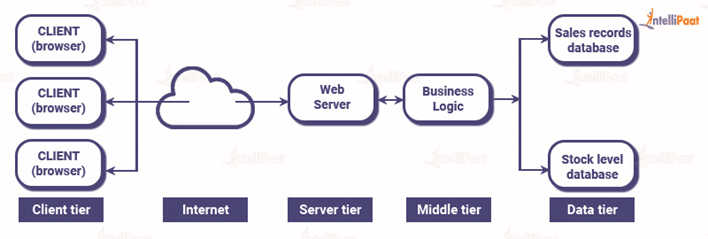
#### **3-tier architecture**



Unlike 2-tier architecture that has no intermediary, in 3-tier client server architecture, a middleware lies between the client and the server. If the client places a request to fetch specific information from the server, the request will first be received by the middleware. It will then be dispatched to the server for further actions. The same pattern will be followed when the server sends a response to the client. The framework of 3-tier architecture is categorized into three main layers, presentation layer, application layer, and database tier.

All three layers are controlled at different ends. While the presentation layer is controlled at the client’s device, the middleware and the server handle the application layer and the database tier respectively. Due to the presence of a third layer that provides data control, 3-tier architecture is more secure, has invisible database structure, and provides data integrity.

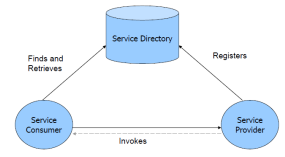
#### **N-tier architecture**



N-tier architecture is also called multi-tier architecture. It is the scaled form of the other three types of architecture. This architecture has a provision for locating each function as an isolated layer that includes presentation, application processing, and management of data functionalities.

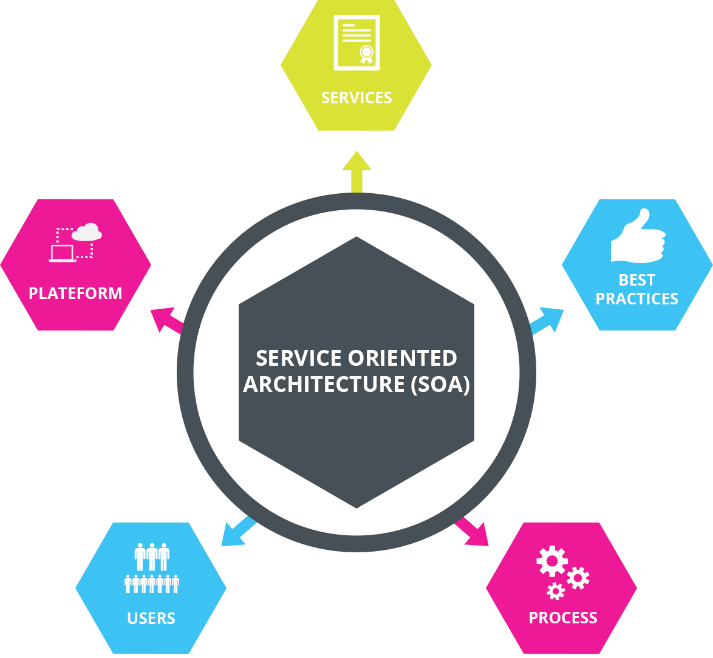
## Illustrate Service Oriented Architecture with help of suitable diagram

Service Oriented Architecture (SOA) is an architectural style that involves the creation of reusable services that can be accessed by different applications and systems. SOA enables businesses to build flexible and scalable systems that can adapt to changing business requirements.



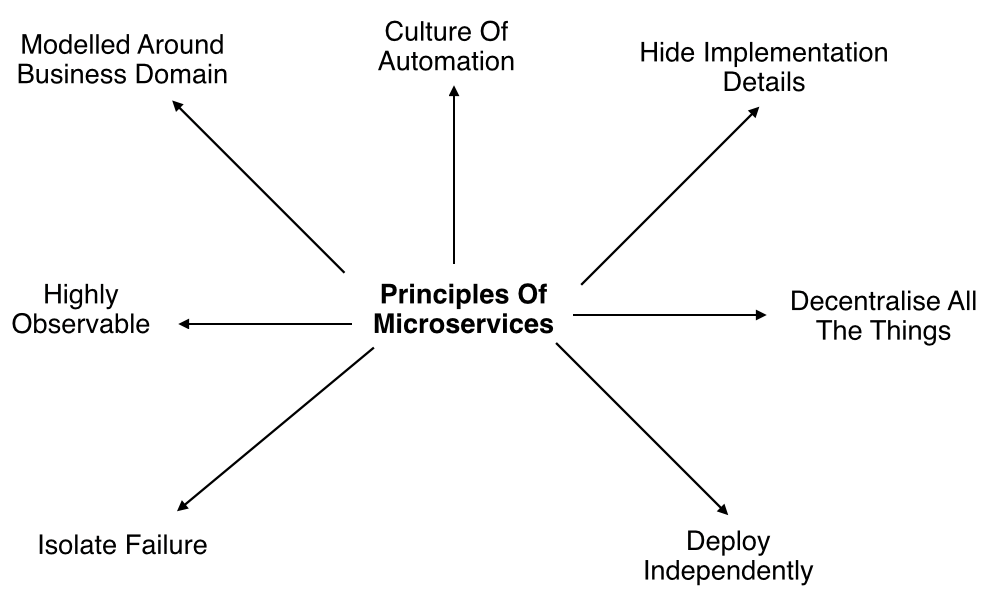
A SOA has three major parts; **service provider, service consumer, and service directory*. Service providers*** are the parties who build service and make available service. ***Service consumers*** are the clients who consume services. ***Service directory*** is the place where service providers register the services and consumer search for services. Service directory provide following services: 1. Scalability of services; can add services incrementally. 2. Decouples consumers from providers. 3. Allows for hot updates of services. 4. Provides a look-up service for consumers. 5. Allows consumers to choose between providers at runtime rather than hard-coding a single provider

There are three roles in each of the Service-Oriented Architecture building blocks: *service provider; service broker, service registry, service repository; and service requester/consumer*.



## Describe Micro- service principles and architecture.

### Microservice Priciple



Modelled around business domain: Micro-services architecture lets us separate system capability into different domains. Each domain will focus on one thing and its associated logic and can easily migrate independently to the next version, and also scale independently according to requirement.

* **Culture of Automation**: As we are building, testing, deploying, and monitoring each service separately and there is an increase in the number of deployment units compared to monolithic architecture we should follow the culture of automation by designing it for continuous integration and continuous delivery.

The smaller and compact codebases and their defined scope are generally turned out to quicker deployments, which also allow us to start to explore the benefits of Continuous Deployment and Continuous integration seamlessly.

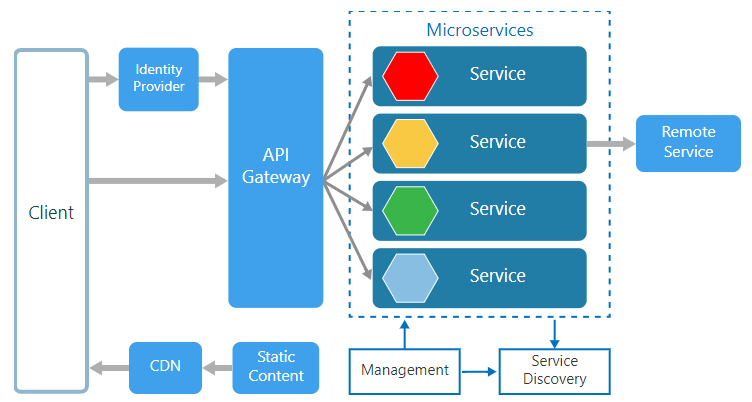
* **Hide implementation details**: [Micro-services](https://www.dotnettricks.com/learn/microservices) should be architected in such a way that they won’t expose the internal details; neither technical implementation nor the business rules that drive it. This will reduce the coupling and help to do changes and improvements without affecting the overall architecture.
* **Decentralization**: In traditional monolithic implementations, the software is designed to use a single database with different tables whereas micro-services are designed in such a way to manage their own database.
* **Deploy Independently**: To enjoy the complete benefits of the architecture, Micro-services should be independently deployable. If you are failing to do so, check for any coupling in the application and solve it.
* **Failure Isolation**: The impact of a failure is less in Micro-services architecture compares to the monolithic type as it will only affect that particular service and its association while other services can keep running. The associated services should handle such scenarios when the dependent is unresponsive or slow.

The larger or the enterprise applications may remain unaffected mostly by the failure of a single module and due to that, other parts of the application are running concurrently which enhances the availability of the feature to the customers most of the time.

* **Highly Observable**: The services should collect as much information to analyze what is happening within each of them like log events and stats.

### Microservice Architecture

An architectural style that structures an application as a collection of small self-contained processes, modelled around a business capability. They don’t share the data structure and will be communicating through APIs. While in a monolithic application all the components are in a single module, in Micro-services we can see all the components are divided into a separate module and communication happens with each other with the help of APIs. In Micro-services Architecture the data is federated where each Micro-services is responsible for its own data model and data.

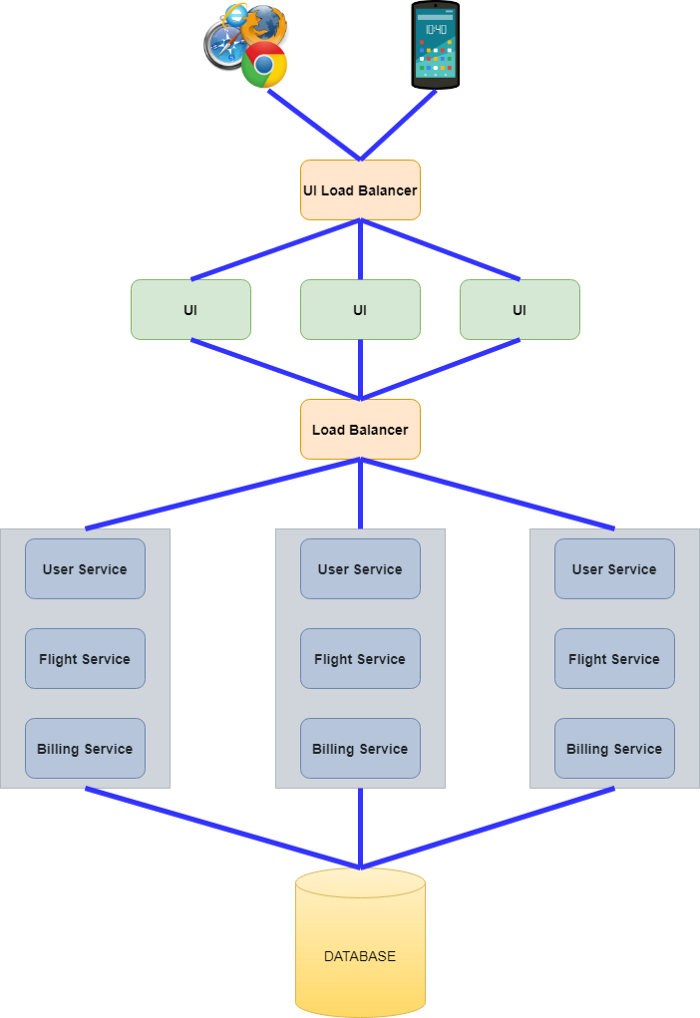


Being small in size, independent, and loosely coupled each service can be developed and deployed independently by a small team as each service is having its own codebase. Data and state persistence should be taken with each service as it lacks a separate data layer to handle it. The services only communicate with well-defined APIs hiding each service’s internal implementation from the other. Each service can use a different technology stack, language, libraries, or frameworks.

* **Management**: The Management takes care of the placement of services on nodes, checking for failures, and rebalancing services across nodes in case of any failures.
* **Service Discovery**: Maintains a list of services and the nodes where each service is located, and also enables the service to look up to find the endpoint for a particular service.
* **API Gateway**: The entry point for clients where all the calls from the client will be taken, analyze, and forward to appropriate services. In case some calls are needed from multiple services API Gateway will aggregate and will return the aggregated result.

## Sketch and elaborate monolithic architecture and state its advantages and disadvantages.

Monolithic Architecture is a traditional way of building applications. This software architecture principle has both advantages and disadvantages. On the one hand, it can bring delight. On the other hand, it can cause disappointment. Let’s review its place in software architecture.



Typical Monolithic Architecture is presented in the picture above. The key characteristic of it is the fact that everything ( User Service, Flight Service, and Billing Service) is located inside in a single deliverable (a jar file in case of [Spring Boot](https://spring.io/projects/spring-boot)). These services are tightly coupled ( despite the fact that they have totally different functions) and use the

same database.

# Advantages of Monolithic Architecture

* **Simplicity of development**. The monolithic approach is a standard way of building applications. No additional knowledge is required. All source code is located in one place which can be quickly understood.
* **Simplicity of debugging**. The debugging process is simple because all code is located in one place. You can easily follow the flow of a request and find an issue.
* **Simplicity of testing**. You test only one service without any dependencies. Everything is usually clear.
* **Simplicity of deployment.** Only one deployment unit (e.g. jar file) should be deployed. There are no dependencies. In cases when UI is packaged with backend code you do not have any breaking changes. Everything exists and changes in one place.
* **Simplicity of application evolution.** Basically, the application does not have any limitation from a business logic perspective. If you need some data for new feature, it is already there.
* **Cross-cutting concerns and customizations are used only once**. You should take care of cross-cutting concerns only once. For instance, security, logging, exception handling, monitoring, choosing and setting up [tomcat](http://tomcat.apache.org/) parameters, setup of data source connection pool, etc.
* **Simplicity in onboarding new team members**. The source code is located in one place. New team members can easily debug some functional flow and to get familiar with the application.
* **Low cost in the early stages of the application**. All source code is located in one place, packaged in a single deployment unit, and deployed. What can be easier? There is no overhead neither in infrastructure cost nor development cost.

# **Disadvantages of Monolithic Architecture**

* **Slow speed of development**. The simplest disadvantage relates to [CI/CD pipeline](https://datamify.com/java-static-code-analysis-overview/). Imagine the monolith that contains a lot of services. Each service in this monolith is covered with tests that are executed for each Pull Request. Even for a small change in a source code, you should wait a lot of time (e.g. 1 hour) for your pipeline to succeed
* **High code coupling**. Of course, you can keep a clear service structure inside your repository. However, as practice shows, eventually, you will end up with a spaghetti code in at least a few places. As a result, the system becomes harder to understand especially for new team members.
* **Code ownership cannot be used**. The system is growing. The logical step is to split responsibilities between several teams. E.g. one team can work on Flight Service, another — for Billing Service. However, there are no boundaries between those services. One team can affect another.
* **Testing becomes harder**. Even a small change can negatively affect the system. As a result, the regression for full monolithic service is required.
* **Performance issues**. Potentially, you can scale the whole monolithic service in cases of performance issues. But what to do with the database? The single database is used for all services. You can start to optimize your database queries or use read replicas. However, there is a limit to this type of optimization.
* **The cost of infrastructure**. In cases of performance issues, you should scale the whole monolithic service. It brings additional costs for application operability.
* **Lack of flexibility.** Using Monolithic Architecture you are tight to the technologies that are used inside your monolith. You cannot use other tools even if they are better for the problem at hand.
* **Problems with deployment.** Even a small change requires the redeployment of the whole monolith.

## Distinguish between Monolithic vs SOA vs Microservices

|  |  |  |  |
| --- | --- | --- | --- |
| Feature | Monolith | SOA | Microservices |
| Architecture | A single, self-contained application that includes all of the application's functionality. | A distributed architecture that uses standardized protocols to enable communication between services. | A distributed architecture that breaks an application down into small, independent services that communicate with each other using lightweight protocols. |
| Advantages | * + Simple to develop and deploy.   + Easy to maintain. | * + Reusability, as services can be shared between different applications and systems.   + Flexibility, as services can be modified and updated without impacting other parts of the system.   + Interoperability, as services can be accessed using different technologies and platforms.   + Scalability, as services can be scaled independently of each other. | - Scalability, as each service can be scaled independently of each other.  - Flexibility, as each service can be modified and updated without impacting other services.  - Resilience, as failures in one service do not necessarily impact the entire application.  - Easier to update and deploy, as changes can be made to individual services without re-deploying the entire application. |
| Disadvantages | - Limited scalability, as the entire application must be scaled as a single unit.  - Difficult to update and deploy, as any changes require re-deploying the entire application. | - Complex to design and implement, as the architecture requires careful planning and coordination.  - Potential for service dependency issues, as changes to one service can impact other services that depend on it. | - Complexity of managing multiple services, as the architecture requires careful coordination and monitoring.  - Potential for service dependency issues, as changes to one service can impact other services that depend on it  - Increased overhead, as there is additional network communication overhead between services. |

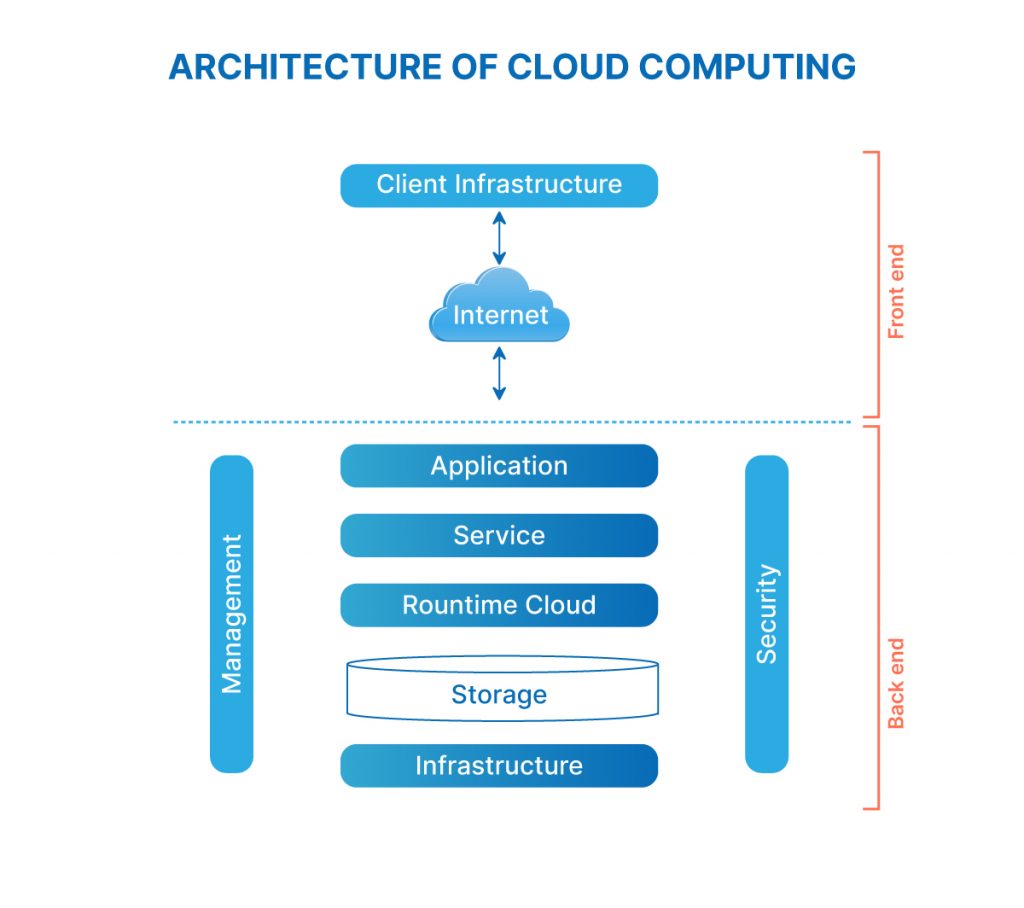
## Define cloud computing and explain its architecture

Cloud computing refers to the delivery of computing resources, including software, storage, and processing power, over the internet. Rather than maintaining their own computing infrastructure, users can access these resources on-demand from a cloud service provider, typically paying for only what they use.

Cloud architecture consists of two major parts:

* Front End
* Back End

Here’s the cloud computing architecture diagram:

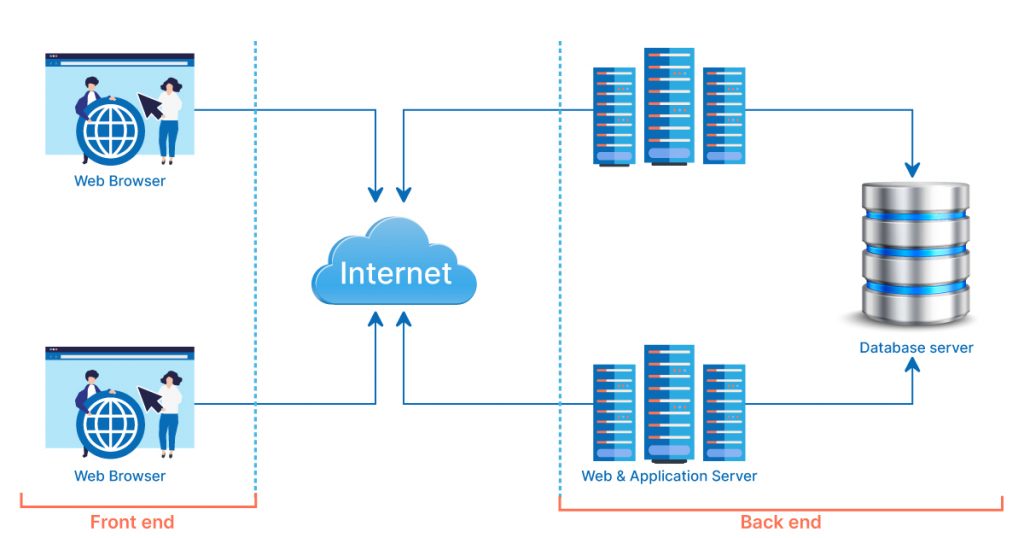


### **What is the Front End of cloud architecture?**

Everything with which the end user interacts is part of the front-end infrastructure. The user interface is the result of integrating various sub-components, such as browsers, tablets, mobile devices, etc. With the help of the front end, the end user can connect to the cloud computing infrastructure. **In short, “the front end is the end that the client interacts with.”**

### **What is the Back End of cloud architecture?**

The back end is everything the user does not usually see and everything that processes the data. The service provider uses the back end to manage all the resources required to provide cloud computing services, such as data storage, security mechanisms, virtual machines, deploying models, servers, traffic control mechanisms, and so on. **In short, “the back end is the end that service provider interacts with.”**



## **Components of cloud computing architecture**

There are various components of cloud architecture. Some of those components are:

1. **Client infrastructure:** The client infrastructure component is a frontend component that provides a graphical user interface (GUI) for users to interact with the cloud.
2. **Application:** An application is any platform, such as an app or software, provided by a company through which clients can access the cloud.
3. **Service:** A cloud service manages the type of service a client uses based on his needs. There are three types of services: [SaaS (software as a service)](https://www.naukri.com/learning/articles/introduction-to-infrastructure-as-a-service-iaas/), [PaaS (platform as a service)](https://www.naukri.com/learning/articles/paas-in-cloud-computing/), and [IaaS (infrastructure as a service)](https://www.naukri.com/learning/articles/introduction-to-infrastructure-as-a-service-iaas/).
4. **Runtime cloud:**The runtime cloud offers virtual machine implementation and runtime environment.
5. **Storage:**The cloud computing storage component provides storage space in the cloud for managing and storing data. Cloud storage is of three types: public cloud, private cloud, and hybrid cloud.
6. **Infrastructure:** The infrastructure component provides services on three levels: the host, the application, and the network. It includes the software and hardware components required to support the cloud computing model, such as storage network devices, servers, and other storage resources.
7. **Management:** The management component is in charge of managing backend components such as storage services, applications, runtime cloud infrastructure, and security issues, as well as establishing coordination.
8. **Security:** Security is the backend component of cloud computing that ensures data security in the cloud.
9. **Internet:** The internet is the medium through which the frontend and backend components communicate and interact.

# Unit 3 Questions

## Illustrate Layer Architecture & each layer in it. Support your with and example.

Layered architecture, also known as the n-tier architecture, is a software design pattern that separates the different components of an application into layers. Each layer is responsible for a specific set of functions and communicates with the layers above and below it using well-defined interfaces. This approach makes it easier to maintain, test, and scale the application.



* Presentation layer would be responsible for handling all user interface and browser communication logic.
* Business layer would be responsible for executing specific business rules associated with the request.
* The persistence layer, also called the data access layer, acts as a protective layer. It contains the code that's necessary to access the database layer. This layer also holds the set of codes that allow you to manipulate various aspects of the database.
* Database layer is responsible for handling data, databases.

For example, when a user submits a form in the web application***, the presentation layer*** would receive the form data and pass it to the business layer. ***The business layer*** would validate the data, perform any necessary business logic, and coordinate with the persistence layer to store or retrieve data. The business layer would then communicate with the presentation layer to display the results of the request to the user.

When a user requests data from the web application, such as a list of products or a customer record, the presentation layer would receive the request and pass it to the business layer. The business layer would coordinate with ***the persistence layer*** to retrieve the requested data from the database. The business layer would then communicate with the presentation layer to display the data to the user.

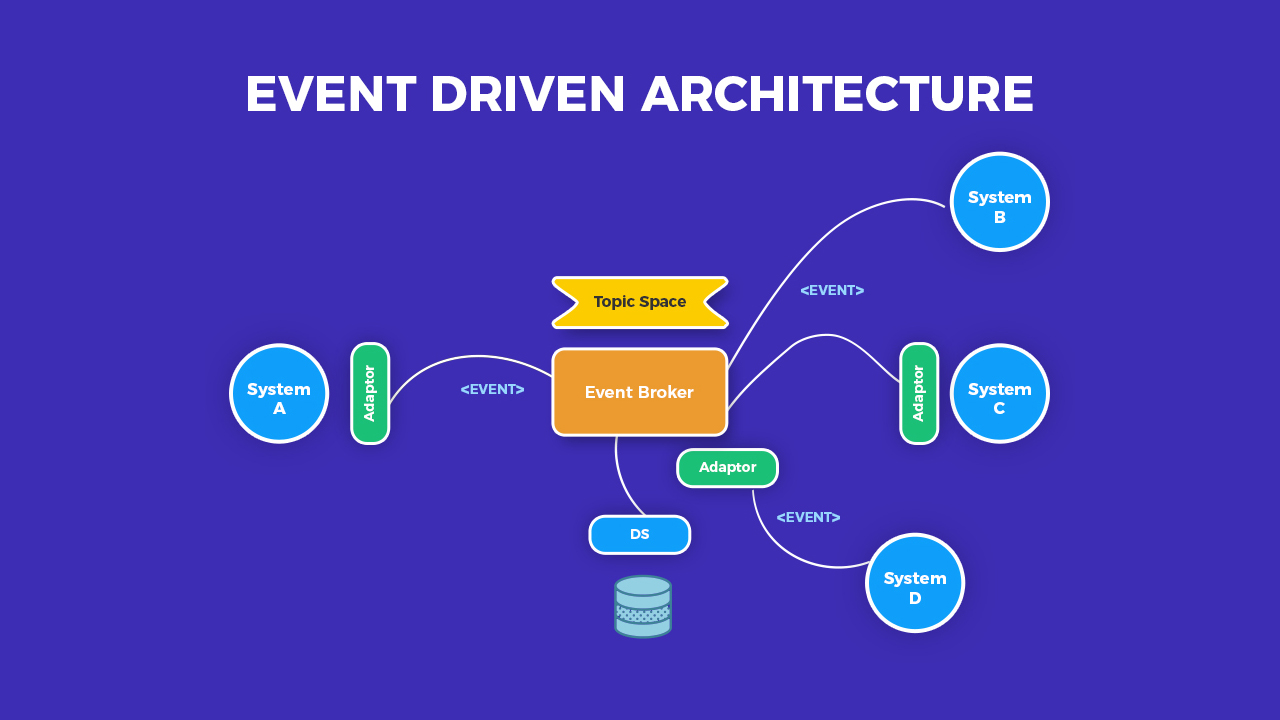
## Elaborate Event Driven Architecture with suitable diagram.

Event-driven architecture models your business systems as a flow of events — when an important business event happens, your systems are alerted to that change of state. A simple example could be a customer changing their address: once that state change is registered, your billing systems get notified of the new address. This is in stark contrast to traditional request-based architectures.

A major benefit of this architectural pattern is that it is both scalable and relatively easy to change. EDA’s inherently loosely coupled nature means that it’s relatively easy to make changes in one particular part of your systems, without breaking anything else.

A well-designed EDA will be based on events that are meaningful to the business. The events could be triggered by user activity, external inputs, such as sensor activity, or outputs from an analytics system. What’s important is the way you define those events, so that you’re capturing something important to your organization.

By basing your designs on these triggering events, you gain flexibility; you’re able to add new behaviours without having to redesign the entire system.



## Discuss the advantages and disadvantages of SOA and give your opinion

### Advantages of SOA:

1. Modularity: SOA allows for the creation of small, independent services that can be reused across multiple applications, reducing the need to develop and maintain duplicate functionality.
2. Flexibility: SOA's loosely coupled architecture allows services to be added, removed, and modified without impacting other services, making it easier to adapt to changing business requirements.
3. Scalability: SOA allows for the creation of scalable services that can handle large volumes of requests, ensuring that the application remains performant even under heavy loads.
4. Interoperability: SOA allows services to be developed using different programming languages, platforms, and technologies, enabling them to communicate with each other seamlessly.

### Disadvantages of SOA:

1. Complexity: SOA can be complex to design and implement due to its distributed nature, making it difficult to manage and troubleshoot.
2. Overhead: The additional layers of abstraction and communication involved in SOA can introduce overhead, slowing down the performance of the application.
3. Security: SOA's distributed nature can make it more difficult to ensure the security of the application, requiring additional measures to protect against attacks.
4. Integration: Integration with existing systems can be challenging, requiring additional effort to ensure that data is exchanged correctly and consistently.

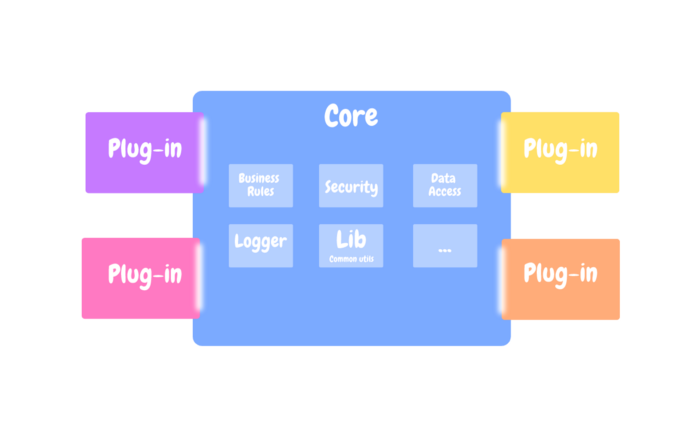
### My opinion:

SOA can be a powerful approach to developing complex, distributed systems that need to be highly scalable and flexible. However, it requires careful planning and design to be successful, and the added complexity and overhead can make it less suitable for smaller, less complex systems. In my opinion, SOA should be considered on a case-by-case basis, with careful consideration given to the specific requirements and constraints of the application being developed.

## Briefly explain Plug-in Architecture support your answer with a suitable diagram

A plug-in architecture is a software design pattern that allows applications to be extended with new functionality by loading and executing external code modules, called plug-ins or extensions. In this architecture, the application provides a framework that defines a set of interfaces or extension points, which the plug-ins can implement or extend to provide additional features.

Here is a diagram that illustrates the plug-in architecture:



### Advantages of Plug-in Architecture:

1. Extensibility: Plug-in architecture allows the application to be easily extended with new functionality without modifying the core code.
2. Modularization: Plug-in architecture promotes the modularization of code, making it easier to maintain and debug.
3. Flexibility: Plug-in architecture provides greater flexibility, as users can choose which plug-ins to install and use.
4. Reusability: Plug-in architecture allows for the reuse of code across multiple applications.

### Disadvantages of Plug-in Architecture:

1. Complexity: Plug-in architecture can introduce additional complexity, as the application must manage the loading and unloading of plug-ins.
2. Compatibility: Plug-in architecture can introduce compatibility issues, as different plug-ins may have conflicting dependencies or requirements.
3. Security: Plug-in architecture can introduce security risks, as plug-ins may be provided by third parties and may not be trustworthy.

## Describe mapping of database and steps for the same

Data mapping is crucial to the success of many data processes. One misstep in data mapping can ripple throughout your organization, leading to replicated errors, and ultimately, to inaccurate analysis.

Step 1: Define — Define the data to be moved, including the tables, the fields within each table, and the format of the field after it's moved. For data integrations, the frequency of data transfer is also defined.

Step 2: Map the Data — Match source fields to destination fields.

Step 3: Transformation — If a field requires transformation, the transformation formula or rule is coded.

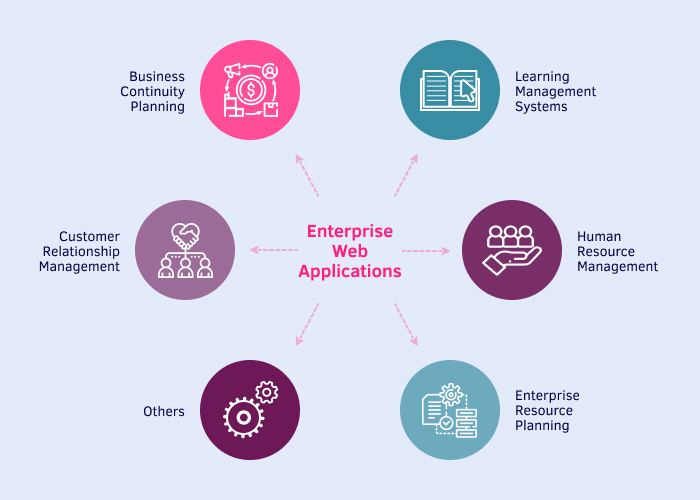
Step 4: Test — Using a test system and sample data from the source, run the transfer to see how it works and make adjustments as necessary.

Step 5: Deploy — Once it's determined that the data transformation is working as planned, schedule a migration or integration go-live event.

Step 6: Maintain and Update — For ongoing data integration, the data map is a living entity that will require updates and changes as new data sources are added, as data sources change, or as requirements at the destination change.

## Illustrate Web Presentation

Enterprise Web Applications allow you to manage and record the internal and external operations and processes of your company or organization. The main advantages are the important cost savings and time savings, by helping the modern company in their digital transformation path.



## Describe Concurrency and Concurrent organization

Concurrency refers to the ability of a system to execute multiple tasks or processes simultaneously. In a concurrent system, two or more tasks can be executed simultaneously, resulting in more efficient use of resources and improved performance.

A concurrent organization is a type of organization in which multiple teams or individuals work on different tasks or projects simultaneously. The goal of concurrent organization is to reduce the time needed to complete a project by overlapping different stages of the project. In a concurrent organization, tasks can be completed simultaneously, rather than in a sequential manner.

There are several advantages to concurrent organization:

1. Faster time to market: By overlapping different stages of a project, concurrent organization can reduce the time needed to complete a project and bring products to market faster.
2. Improved efficiency: Concurrent organization allows for more efficient use of resources and improves overall productivity.
3. Better risk management: By dividing a project into smaller tasks and executing them simultaneously, concurrent organization can reduce the impact of potential risks and improve risk management.
4. Increased innovation: Concurrent organization encourages collaboration and communication between teams, which can lead to increased innovation and creativity.

However, there are also some challenges associated with concurrent organization, such as:

1. Coordination: Coordinating multiple teams and ensuring that they are working towards the same goals can be challenging.
2. Communication: Effective communication is critical in a concurrent organization, as it is important to ensure that all teams are aware of each other's progress and any changes in project requirements.
3. Resource allocation: Proper resource allocation is crucial in a concurrent organization, as it is important to ensure that each team has the resources it needs to complete its tasks.

## Explain organizing digital logic in detail with suitable diagram.

Organizing digital logic refers to the process of structuring the hardware components of an enterprise system to ensure that they function together efficiently and effectively. This typically involves breaking down the system into smaller components or modules, and then organizing these modules in a logical and structured way.

One approach to organizing digital logic is through the use of a hierarchical design structure. In this approach, the system is broken down into smaller components or modules, each of which can be designed and tested separately before being integrated into the overall system. These modules can be organized into a hierarchy, with higher-level modules composed of lower-level modules.

In this example, the enterprise system is divided into two main components: hardware and software. The hardware component is further divided into three modules: CPU, RAM, and I/O. The software component is divided into two modules: the operating system and applications.

The CPU module is then broken down further into control unit and data paths, which are the two main sub-components of the CPU. This hierarchy allows for each module to be designed and tested separately before being integrated into the larger system. It also allows for easier debugging and maintenance of the system.

By organizing digital logic in this way, enterprise systems can be designed and implemented more efficiently and effectively. However, it is important to note that the specific organization of digital logic will vary depending on the specific needs and requirements of the enterprise system.

# Unit 4 Questions

## State the key elements of Enterprise Integration

Enterprise Integration refers to the process of connecting different systems and applications within an organization in order to improve communication, collaboration, and efficiency. The key elements of Enterprise Integration are:

1. Business Processes: The set of activities or tasks performed by an organization to achieve its business objectives. The integration of business processes is critical for streamlining operations, improving customer service, and reducing costs.
2. Data Integration: The process of combining data from multiple sources to provide a unified view of the data. Data integration involves mapping and transforming data between different systems and applications.
3. Application Integration: The process of connecting different applications and systems to enable them to work together seamlessly. Application integration involves connecting different applications through standard interfaces or middleware.
4. Infrastructure Integration: The process of connecting different infrastructure components such as servers, storage devices, and networks to enable seamless communication and collaboration.
5. Standards: Standards play a critical role in Enterprise Integration as they ensure that different systems and applications can communicate and work together seamlessly. Standards such as web services, REST, SOAP, and XML provide a common language for systems and applications to communicate.
6. Security: Security is an important element of Enterprise Integration as it involves protecting the confidentiality, integrity, and availability of information as it flows between different systems and applications.
7. Governance: Governance refers to the process of managing Enterprise Integration and ensuring that it aligns with business objectives. Governance involves defining policies and procedures, establishing roles and responsibilities, and monitoring and measuring performance.

## Classify the different types of integration styles.

There are several different types of integration styles used in Enterprise Integration, including:

1. Point-to-Point Integration: This style of integration involves connecting two systems directly, without any intermediary components. It is a simple and direct approach, but it can become complex and difficult to maintain as the number of systems increases.
2. Message-Oriented Middleware (MOM): This style of integration involves using a message broker or middleware to facilitate communication between systems. Messages are sent between systems through the middleware, which acts as an intermediary.
3. Service-Oriented Architecture (SOA): This style of integration involves building systems and applications as a collection of services that can be accessed and used by other systems and applications. Services are accessed through standard interfaces, making it easier to integrate different systems.
4. Event-Driven Architecture (EDA): This style of integration involves using events and messages to trigger actions and processes within systems. Events can be generated by different systems and applications, and they are used to trigger actions within other systems.
5. API-Led Integration: This style of integration involves building systems and applications as a set of APIs that can be accessed and used by other systems and applications. APIs are accessed through standard interfaces, making it easier to integrate different systems.
6. File-Based Integration: This style of integration involves exchanging files between systems using a shared file system or a file transfer protocol. This approach is simple and straightforward, but it can become complex and difficult to manage as the number of systems increases.

## Illustrate the idea of Supplier and exchange perspective in integrating the B2B exchange

B2B (business-to-business) exchanges are platforms that enable organizations to conduct business transactions and collaborate with their trading partners over the internet. The integration of B2B exchanges involves the collaboration and integration of suppliers and buyers through a common platform. The supplier and exchange perspectives are two key aspects of this integration process.

The supplier perspective involves the integration of suppliers into the B2B exchange. Suppliers can use the platform to sell their products or services, manage their inventory, receive orders, and process payments. The exchange provides a centralized location for suppliers to manage their transactions and interact with their customers.

The exchange perspective involves the integration of buyers into the B2B exchange. Buyers can use the platform to search for suppliers, place orders, manage their accounts, and process payments. The exchange provides a centralized location for buyers to manage their transactions and interact with their suppliers.

The following diagram illustrates the idea of supplier and exchange perspective in integrating the B2B exchange:



In this diagram, the B2B exchange serves as a platform that connects suppliers and buyers. Suppliers and buyers can interact with the exchange to manage their transactions and collaborate with their trading partners.

The integration of suppliers and buyers into the B2B exchange provides several benefits, including increased efficiency, reduced costs, improved collaboration, and enhanced visibility into supply chain operations. By leveraging a common platform, organizations can streamline their operations and improve their competitiveness in the marketplace.

## Describe the most common automated integration option orchestration. Write the pros and cons of using the full automation process for integration.

Orchestration is a common automated integration option that involves the use of a centralized system or tool to manage and coordinate the interactions between different applications, systems, and services. The orchestration process involves defining a series of steps or workflows that need to be executed to achieve a specific business outcome. The orchestration system is responsible for coordinating the execution of these steps across different systems and services.

One of the most common types of automated integration options is full automation, which involves automating the entire integration process from end to end. In a fully automated integration, the orchestration system is responsible for executing all the necessary steps to complete the integration, without requiring any manual intervention or oversight.

Pros of full automation:

1. Increased efficiency: Full automation can significantly improve the speed and efficiency of the integration process by eliminating the need for manual intervention or oversight.
2. Consistency: With full automation, the integration process is executed in a consistent and repeatable manner, reducing the risk of errors or inconsistencies.
3. Scalability: Full automation can be easily scaled to handle large volumes of data or transactions, making it ideal for enterprise-scale integrations.
4. Cost savings: Full automation can help organizations reduce their IT costs by reducing the need for manual labor and oversight.

Cons of full automation:

1. Complexity: Full automation can be complex to set up and maintain, requiring specialized expertise and resources.
2. Lack of flexibility: Full automation may not be suitable for all types of integrations, as it may lack the flexibility to accommodate unique or custom integration requirements.
3. Risk of errors: Full automation may increase the risk of errors or issues if the integration process is not properly designed or configured.
4. Limited human oversight: Full automation may result in limited human oversight, which can make it difficult to identify and resolve issues that may arise during the integration process.

In summary, while full automation can offer significant benefits in terms of efficiency, consistency, scalability, and cost savings, it may not be suitable for all types of integrations. Organizations need to carefully evaluate their integration requirements and consider the potential benefits and drawbacks of full automation before implementing this option.

## Explain the concept of Web Services. What are the characteristics of WSDL?

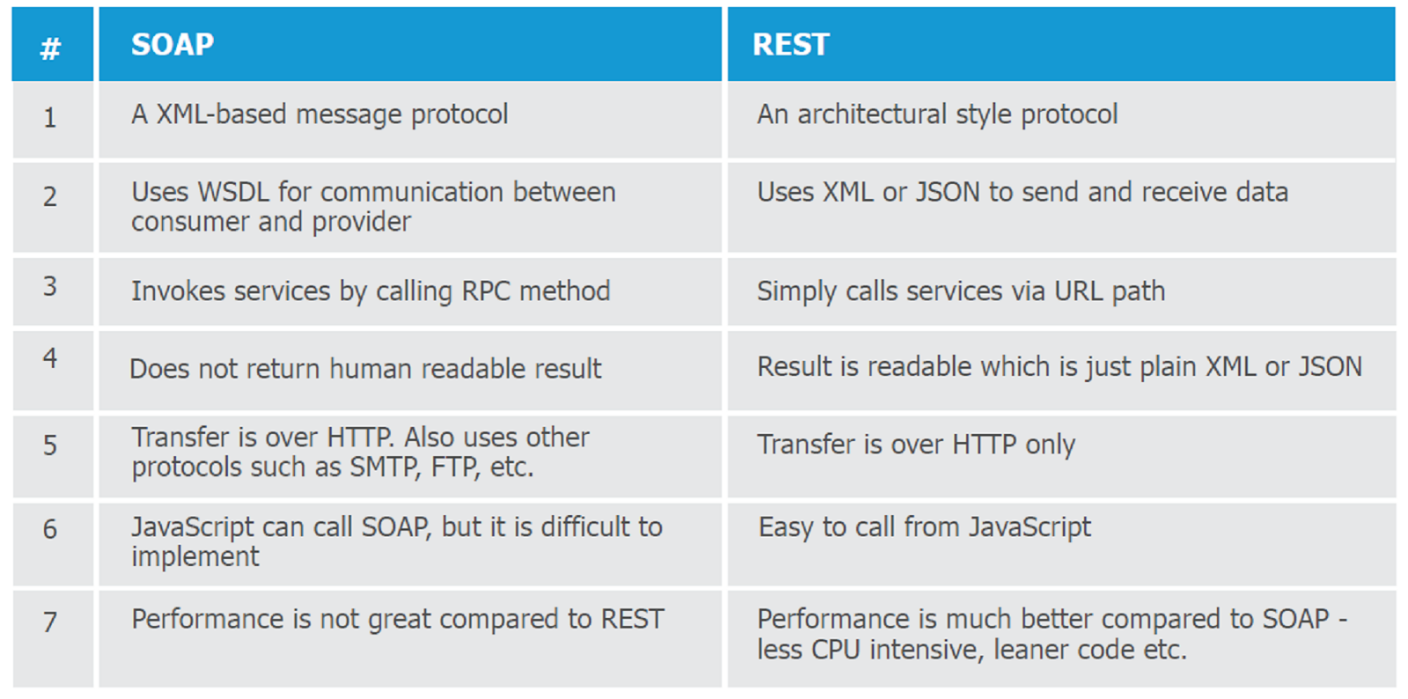
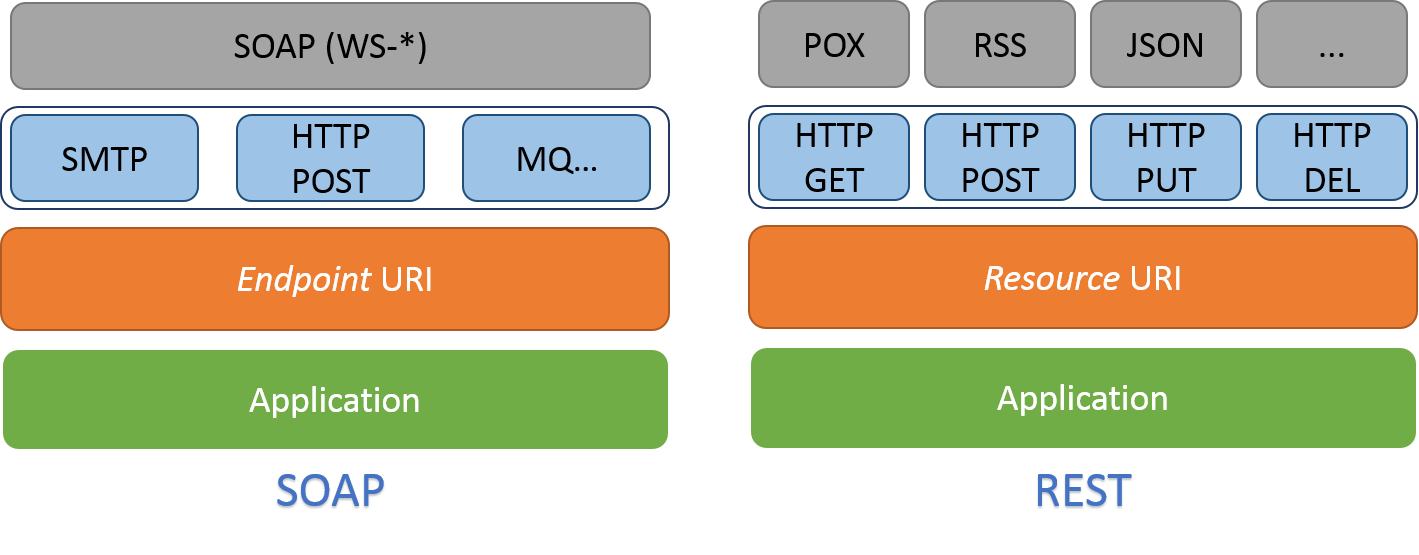
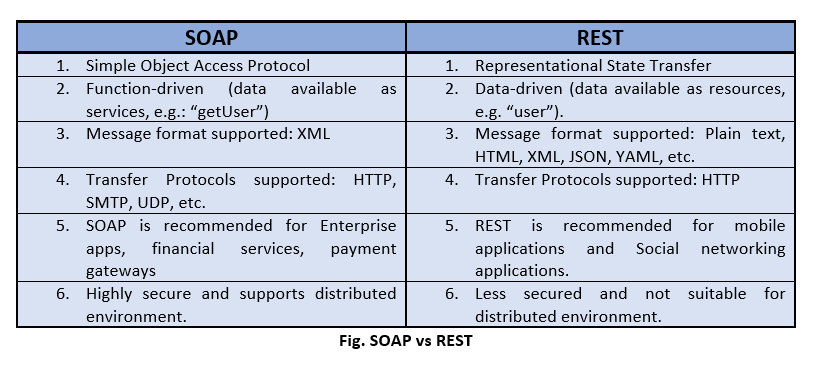
Web Services are a standardized way of communication between various applications running on different platforms and operating systems over the internet using open standards such as XML, SOAP, and WSDL.

Web Services Description Language (WSDL) is an XML-based language that is used to describe the functionality offered by a web service. It provides a contract for the client and server to communicate and understand the communication protocol. WSDL describes the following characteristics of a web service:

1. Interface: It describes the operations that can be performed on the web service, including their inputs, outputs, and fault messages.
2. Binding: It defines the protocol to be used for communication, such as HTTP, SOAP, or other custom protocols.
3. Endpoint: It specifies the network address of the web service where the requests are sent.
4. Message format: It defines the message format and structure that the web service expects and returns.
5. Service description: It provides a description of the web service, including its name, version, and other relevant metadata.

The characteristics of WSDL are essential for the creation, discovery, and integration of web services into enterprise systems. WSDL provides a common language for developers and businesses to interact and integrate their applications using open standards, which simplifies the process of building and maintaining complex enterprise systems.

## Differentiate between SOAP and REST.

## Describe the services provided by restful API servers.

Restful API servers provide a range of services that enable clients to interact with resources over the web using the REST architectural style. Some of the key services provided by RESTful API servers include:

1. Resource identification: RESTful APIs use URIs to identify resources, which can be any type of data, including documents, images, videos, or data records.
2. Resource manipulation: RESTful APIs provide a set of HTTP methods (GET, POST, PUT, DELETE) that allow clients to create, read, update, or delete resources. These methods are mapped to CRUD (Create, Read, Update, Delete) operations in a database.
3. Stateless communication: RESTful APIs are stateless, meaning that each request contains all the information necessary for the server to process the request. This enables the server to scale easily and handle large volumes of requests.
4. Representation of resources: RESTful APIs represent resources using a variety of formats, such as JSON or XML. This allows clients to choose the format that best suits their needs.
5. Caching: RESTful APIs support caching of responses, which can improve performance and reduce network traffic.
6. Security: RESTful APIs provide a range of security mechanisms, including authentication, authorization, and encryption, to ensure that resources are accessed only by authorized clients.
7. Discoverability: RESTful APIs provide a uniform interface that enables clients to discover available resources and their capabilities.

Overall, RESTful APIs provide a flexible, scalable, and efficient way for clients to interact with resources over the web, making them an essential component of modern web applications and services.

## Give two modern service integration techniques with their advantages and disadvantages.

Two modern service integration techniques are:

1. Event-Driven Architecture (EDA): EDA is a service integration technique where communication is based on the events. An event is a change in state or a trigger in a system. EDA follows a publish-subscribe model, where publishers send events to subscribers, and subscribers listen to specific events that they are interested in. EDA provides benefits such as scalability, flexibility, and responsiveness. However, implementing EDA requires careful planning, and it can be challenging to manage events in large-scale systems.

### Advantages:

* Scalability: EDA allows scaling of individual services without affecting other services in the system.
* Flexibility: EDA provides a flexible architecture that can handle various event sources and types.
* Responsiveness: EDA enables real-time processing of events, leading to quicker responses.

### Disadvantages:

* Complexity: EDA can be complex to design and implement.
* Event Management: Managing events in a large-scale system can be challenging.
* Debugging: Debugging issues in EDA can be challenging, as events can be asynchronous and distributed.

1. GraphQL: GraphQL is a query language and runtime for APIs that was developed by Facebook. It provides a flexible and efficient way to request and deliver data. In GraphQL, clients define the structure of the data they need, and the server delivers the exact data requested. GraphQL allows retrieving multiple resources in a single request, reducing the number of requests needed to the server. GraphQL provides benefits such as efficiency, flexibility, and improved developer experience. However, it requires a learning curve to master and can lead to over-fetching and under-fetching of data.

### Advantages:

* Efficiency: GraphQL allows retrieving multiple resources in a single request, leading to improved performance and reduced network traffic.
* Flexibility: GraphQL provides a flexible data model that allows developers to request only the data they need.
* Improved Developer Experience: GraphQL provides tools for client-side development, making it easier for developers to build and maintain applications.

### Disadvantages:

* Learning Curve: GraphQL requires a learning curve to master, especially for developers who are new to the technology.

Over-fetching and Under-fetching of Data: GraphQL can lead to over-fetching and under-fetching of data, as the client defines the structure of the data they need.

# Unit 5 Questions

## Express the idea of deployment of an application? Explain with example.

Deployment of an application refers to the process of making an application available to users or clients after its development. It involves setting up the necessary infrastructure and configurations to run the application in a production environment.

For example, let's consider the deployment of a web application that allows users to book movie tickets. After the development phase is completed, the application needs to be deployed to a web server so that users can access it through their web browsers.

The deployment process typically involves the following steps:

1. Infrastructure setup: This involves setting up the necessary hardware, software, and network components required to run the application. This includes servers, databases, load balancers, firewalls, and other related tools.
2. Configuration: Once the infrastructure is set up, the next step is to configure the application to run in the production environment. This includes setting up the web server, configuring the database, and configuring any other software components that the application requires.
3. Testing: Before making the application available to users, it is important to test it thoroughly to ensure that it is working correctly. This may involve testing the application's performance, security, and functionality.
4. Deployment: Once the application has been tested and is ready for production use, it can be deployed to the web server. This may involve copying the application files to the server, configuring the web server to run the application, and starting the application.
5. Maintenance: Once the application is deployed, it needs to be maintained and monitored to ensure that it continues to run smoothly. This may involve performing regular backups, applying security patches, and monitoring the application's performance.

In summary, deploying an application involves setting up the infrastructure, configuring the software components, testing the application, and making it available to users in a production environment.

## Compare Group Policy, PDQ Deploy and CITRIX with its pros and cons.

|  |  |  |  |
| --- | --- | --- | --- |
| **Feature** | **Group Policy** | **PDQ Deploy** | **CITRIX** |
| Definition | A feature in Microsoft Windows that allows administrators to manage and configure settings on multiple computers in an organization | A software deployment tool that simplifies the process of installing and updating software across multiple computers | A virtualization solution that allows users to access applications and desktops from anywhere |
| Deployment | Centralized control | Centralized control | Decentralized control |
| Targeted Devices | Windows OS devices | Windows OS devices | Various OS and devices |
| Scalability | Limited to Windows environment | Limited to Windows environment | Scalable to multiple devices |
| Security | Secure communication | Secure communication | Secure virtualization |
| Automation | Automatic application | Automatic application | Automatic deployment |
| Management | Limited to Windows environment | Limited to Windows environment | Management of virtualization |
| Cost | No additional cost | Paid software | Paid software |
| Pros | Secured way of deploying over a network | Applications features can be edited before deploying | Suitable for apps and mobile applications |
| Cons | Organizing and handling large number of GPOs is difficult | Features like automated Windows update repository are missing | It is a costly solution |

## How security can be achieved while doing enterprise deployment?

Enterprise deployment often involves installing software on multiple machines within an organization. This process needs to be done securely to ensure that sensitive data is not compromised. Here are some ways to achieve security during enterprise deployment:

1. **Use secure communication channels**: When deploying software, it is important to ensure that the communication channel between the deployment server and the client machines is secure. This can be done by using secure protocols such as HTTPS, SSH, or SSL.
2. **Authentication and Authorization**: Authentication is the process of verifying the identity of a user, while authorization is the process of granting or denying access to a particular resource. Strong authentication and authorization mechanisms should be put in place to prevent unauthorized access.
3. **Encryption**: Data should be encrypted both in transit and at rest. This prevents unauthorized access to data by hackers or other malicious actors.
4. **Access control**: Access control policies should be put in place to ensure that only authorized personnel have access to the deployment environment. This can be done by implementing role-based access control or other access control mechanisms.
5. **Testing and Auditing**: It is important to thoroughly test the deployment process and audit the deployed software for vulnerabilities. Regular security testing should be conducted to identify and fix security flaws.
6. **Patching and Updates**: Regular patching and updating of software is essential to ensure that vulnerabilities are addressed promptly.

By implementing these security measures, an organization can ensure that their enterprise deployment is secure and their sensitive data is protected.

## Explain the best known metric of network availability. Discuss the parallel transport availability.

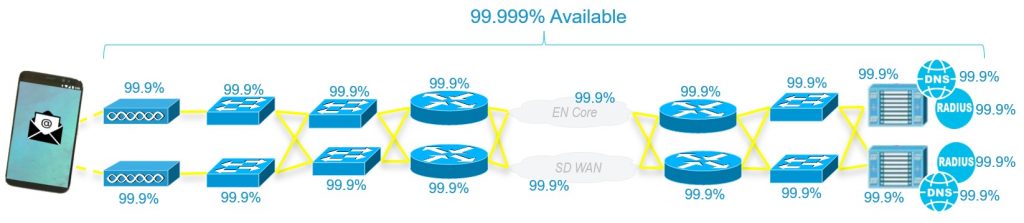
The best-known metric of network availability is known as “five nines”. What five nines means is that the end-user perceives that their application is available 99.999% of the time. This permits only 5.26 minutes of downtime a year. Depending on the application and network topology, this can be a very stringent standard.

Parallel transport availability is a metric that measures the ability of a network to maintain continuous communication between two or more points, even in the event of a failure in one or more network links. This is achieved by creating parallel paths between the points, such that if one path fails, the data can be rerouted through another path, thus ensuring uninterrupted communication.

Parallel transport availability is achieved through the use of redundancy and failover mechanisms in the network infrastructure. Redundancy refers to the provision of multiple paths between the points, while failover refers to the ability to automatically switch to an alternate path in the event of a failure in the primary path.

The parallel transport availability metric is usually expressed as a percentage, which represents the amount of time that the network is available for parallel transport between the points. For example, a network with a parallel transport availability of 99.999% (five nines) would be available for parallel transport for 99.999% of the time, or 5 minutes and 15.6 seconds of downtime per year.

Parallel transport availability is critical for enterprise systems that require high availability, such as financial transactions, healthcare systems, and emergency services. By ensuring continuous communication between the points, parallel transport availability helps to prevent data loss, reduce downtime, and minimize the impact of failures on the system.



## Give any key requirement to successful delivery. How it can transform the work environment?

Transparency is a key requirement for successful delivery in any organization. It involves clear communication of information, decisions, and actions to all stakeholders involved in the project. Transparency can transform the work environment in several ways:

1. **Trust**: Transparency builds trust between team members and stakeholders. When everyone has access to the same information, there are no hidden agendas or surprises. This can help to build trust and foster better working relationships.
2. **Collaboration**: Transparency encourages collaboration between team members. When everyone is aware of what is going on, they can provide feedback, suggest improvements, and work together to solve problems.
3. **Accountability**: Transparency helps to hold team members accountable for their actions. When everyone can see what is going on, it is easier to identify who is responsible for specific tasks or decisions.
4. **Continuous Improvement**: Transparency allows for continuous improvement. When everyone has access to the same information, it is easier to identify areas for improvement and make changes to the process.

Top of Form

# Unit 6 Questions

## Illustrate why are Enterprises moving towards cloud? Give any three reasons.

Enterprises are moving towards the cloud for several reasons, including:

1. Scalability: Cloud computing provides enterprises with the ability to scale their computing resources up or down as needed, without the need for significant upfront investment in infrastructure. This allows businesses to handle increased workloads during peak periods without having to worry about investing in additional hardware or software.
2. Cost savings: By using cloud computing services, enterprises can save money by avoiding the upfront costs associated with purchasing and maintaining hardware and software. They can also avoid the ongoing costs of maintaining and upgrading that infrastructure over time.
3. Flexibility: Cloud computing services provide enterprises with a high degree of flexibility, allowing them to quickly and easily adapt to changing business requirements. For example, if a business needs to scale up its computing resources in response to a sudden increase in demand, it can do so quickly and easily by simply increasing its usage of cloud services.

## Classify the types of Private Cloud. Give examples of private cloud providers.

Private clouds can be classified into three types:

1. **On-premise Private Cloud**: This type of private cloud is built within an organization's own data center or server room. It gives organizations full control over their infrastructure and data. However, it requires significant investment in hardware and software, as well as skilled IT staff to manage the infrastructure.

Examples of on-premise private cloud providers include OpenStack, VMware, and Microsoft Azure Stack.

1. **Hosted Private Cloud**: This type of private cloud is built and managed by a third-party service provider, but is dedicated to a single organization. The provider manages the hardware, software, and infrastructure, while the organization retains control over its data.

Examples of hosted private cloud providers include IBM Cloud, Rackspace, and CenturyLink.

1. **Virtual Private Cloud**: This type of private cloud is a virtualized environment within a public cloud infrastructure. It provides an isolated environment for an organization's data and applications, while still leveraging the scalability and flexibility of a public cloud.

Examples of virtual private cloud providers include Amazon Web Services (AWS), Google Cloud Platform, and Microsoft Azure.

## Public Cloud is an IT model where on-demand computing services and infrastructure are managed by a third-party provider". Justify the statement with relevant example.

Public cloud refers to a type of cloud computing in which resources, such as virtual machines, storage, and applications, are made available to the general public over the internet by a third-party provider, on a pay-per-use basis.

An example of a public cloud provider is Amazon Web Services (AWS). AWS offers a range of services such as Elastic Compute Cloud (EC2) for virtual machine instances, Simple Storage Service (S3) for object storage, and Lambda for serverless computing. Any individual or organization with an internet connection can access and utilize these services on a pay-per-use basis, without having to invest in expensive hardware or software infrastructure.

Another example of a public cloud provider is Microsoft Azure. Azure offers similar services to AWS and is widely used by businesses and individuals around the world.

Public cloud services offer several benefits, such as scalability, flexibility, and cost-effectiveness, making it an attractive option for businesses and individuals looking to leverage the benefits of cloud computing without investing in expensive infrastructure.

## Identify how public clouds can save money? Elaborate the different ways.

Public clouds can save money in several ways:

1. **Pay-as-you-go pricing model:** Public cloud providers typically use a pay-as-you-go pricing model, where customers only pay for the services and resources they use, rather than investing in expensive hardware upfront. This can help organizations reduce their capital expenditure (CapEx) and convert it into operational expenditure (OpEx).
2. **Economies of scale:** Public cloud providers benefit from economies of scale, as they can leverage their large infrastructure and resources to offer services at a lower cost. This can help organizations save money compared to building and maintaining their own infrastructure.
3. **Reduced maintenance and operational costs:** Public cloud providers handle the maintenance and operational costs associated with running a data center, which can be expensive for organizations. This can include costs related to power, cooling, physical security, and staffing.
4. **Increased flexibility and agility:** Public clouds offer a high degree of flexibility and agility, allowing organizations to quickly scale up or down their resources as needed. This can help organizations save money by avoiding the need to invest in excess capacity to handle occasional spikes in demand.

It is important to note that while public clouds can save money, organizations need to be careful in selecting the right services and resources that fit their needs to avoid unnecessary costs.

## Differentiate between hybrid cloud and multi cloud.

|  |  |  |
| --- | --- | --- |
|  | Hybrid-Cloud | Multi-Cloud |
| Definition | A cloud computing environment that combines both public and private clouds. | A cloud computing strategy that involves using multiple cloud services from different providers. |
| Components | Typically consists of at least one private cloud and one public cloud. | Utilizes multiple cloud services, which can be either public or private. |
| Management | Managed and controlled by a single organization. | Managed and controlled by multiple organizations. |
| Deployment | Requires a certain level of integration and coordination between the public and private cloud components. | Deploying different services or applications to different clouds based on their requirements. |
| Flexibility | Provides a level of flexibility to an organization, allowing it to take advantage of the strengths of both public and private clouds. | Offers greater flexibility, as it allows organizations to mix and match services from different providers to meet their specific needs. |
| Cost | Can be more cost-effective than a purely public cloud approach, but may require more upfront investment in private cloud infrastructure. | Can be more expensive than a hybrid cloud approach, as it may require additional management and integration costs. |
| Security and Control | Provides greater control and security over sensitive data and applications by keeping them within a private cloud environment. | Can offer greater security by spreading workloads across multiple providers, reducing the risk of a single point of failure. However, it may also present challenges in terms of maintaining consistent security across multiple clouds. |
| Complexity | Can be more complex to manage due to the need for integration and coordination between multiple clouds. | Can also be complex, as organizations need to manage and integrate services from multiple providers. |

## Identify the types of scaling in cloud computing.

There are two types of scaling in cloud computing:

1. **Vertical Scaling**: Also known as scaling up or instance resizing, this type of scaling involves increasing or decreasing the size of the instance (e.g. CPU, RAM, storage) in response to changes in demand. This is typically done manually and can lead to downtime during the resizing process.
2. **Horizontal Scaling**: Also known as scaling out or instance replication, this type of scaling involves adding or removing instances in response to changes in demand. This is typically done automatically by load balancers, and can be done without downtime if done properly.

## Explain the concept of cloud-based software testing.

Cloud-based testing means performing tests for a software application through resources found in the cloud. These tests can include the hardware, software and infrastructure of an application. QA teams rely on a cloud software testing strategy and these cloud based mobile testing solutions in order to verify a product’s security, functionality and usability before market launch.

Cloud testing concentrates on these core testing components to ensure full testing coverage:

1. **Application**: Covers cloud based testing software for functionality, data security, browser compatibility and end-to-end business workflows.
2. **Network**: Includes testing a variety of network bandwidths and protocols as well as successful transfer of data through networks.
3. **Infrastructure**: Focuses on testing for disaster recovery, secure connectivity, backups, and storage policies.

### How the Cloud Changes Testing

Cloud software testing is essential after migrating to the cloud. However, the cloud environment encourages QA teams to alter how they execute test cases so that they experience successful testing cycles.

1. **Functional Testing**: Cloud testing means validating the cloud service or SaaS functions, including the end-to-end functionality of an application.
2. **Integration Testing**: QA testers review SaaS based integration in the cloud as well as the application's integration between legacy systems.
3. **Security Testing**: Cloud-based mobile app testing solutions allow QA testers to execute test cases that focus on user privacy and security across a diverse range of the user. It also focuses on data integrity and protection during transit and rest periods, connectivity security, protection against cyberattacks, and the security of the software interface.
4. **Performance Testing**: Cloud based mobile app testing tools enable QA teams to leverage cloud for load testing, monitor application stability and execute performance testing in a scalable environment.

## Explain why is laaS important? Discuss the use cases of Infrastructure as a services

Infrastructure as a Service (IaaS) is an essential component of cloud computing that provides businesses with a virtual infrastructure that can be accessed and managed over the internet. IaaS is important for various reasons, including:

1. **Scalability**: IaaS allows businesses to scale up or down their computing infrastructure as their needs change without having to invest in physical hardware or worry about capacity constraints.
2. **Cost savings**: With IaaS, businesses can avoid the high costs associated with purchasing, maintaining, and upgrading physical hardware, as well as the expenses associated with data centre facilities and management.
3. **Flexibility**: IaaS enables businesses to choose the specific services they need and customize their infrastructure to meet their unique requirements.

Some of the key use cases for IaaS include:

1. **Web hosting**: IaaS can be used to host websites and web applications, providing businesses with a scalable and cost-effective solution for online operations.
2. **Data backup and recovery**: IaaS can be used to store data backups in the cloud, providing businesses with a secure and reliable backup solution that can be accessed from anywhere.
3. **Development and testing**: IaaS can be used to create virtual environments for software development and testing, allowing developers to easily provision resources and test their applications in a variety of scenarios.
4. **Big data analytics**: IaaS can be used to provide scalable computing resources for big data processing and analysis, allowing businesses to gain insights from large volumes of data in real-time.
5. **Disaster recovery**: IaaS can be used to create disaster recovery solutions, providing businesses with a secondary infrastructure that can be used in the event of an outage or other disaster.

## List the major public clouds. Also identify the pros and cons for any two public clouds

Major public clouds are:

1. Amazon Web Services (AWS)
2. Microsoft Azure
3. Google Cloud Platform (GCP)
4. IBM Cloud
5. Oracle Cloud Infrastructure (OCI)

Two public clouds with their pros and cons are:

1. Amazon Web Services (AWS)

* Pros:
  + Wide range of services and features
  + High availability and reliability
  + Good security and compliance
  + Flexible pricing and pay-as-you-go model
* Cons:
  + Complex pricing and documentation
  + Steep learning curve
  + Lack of transparency in billing
  + Limited customization options

1. Microsoft Azure

* Pros:
  + Strong hybrid cloud capabilities
  + Comprehensive security and compliance
  + Good support for enterprise applications
  + Large global network of data centers
* Cons:
  + Limited scalability options
  + Higher pricing for Windows-based applications
  + Complex interface and documentation
  + Limited support for open source technologies

# Unit 7 Questions

## Describe the Enterprise architecture frameworks? Explain with the help of a example

Enterprise Architecture Frameworks (EAFs) are a set of guidelines, principles, and best practices that help organizations to create a comprehensive blueprint for managing and aligning IT resources with business objectives. They provide a structured approach for designing, implementing, and managing enterprise architecture.

There are several EAFs available, but some of the most commonly used ones include:

1. The Open Group Architecture Framework (TOGAF)
2. Federal Enterprise Architecture Framework (FEAF)
3. Zachman Framework for Enterprise Architecture

Each of these frameworks has its own set of principles, components, and tools that help organizations to create their enterprise architecture. For instance, TOGAF provides a detailed methodology for developing enterprise architecture, while FEAF focuses on the development and implementation of IT systems within the federal government.

Let's take an example to understand how EAFs work. Suppose a large organization is looking to develop an enterprise architecture to support its growth and expansion plans. The organization chooses to use the TOGAF framework to develop its architecture.

The first step in the TOGAF framework is to establish a clear understanding of the organization's goals and objectives. This includes understanding the organization's business strategy, vision, and mission, as well as its current and future IT requirements.

The next step is to develop an architecture vision that outlines the organization's future-state architecture. This includes identifying the key components of the architecture, such as business processes, data, applications, and technology infrastructure.

Once the architecture vision is established, the organization can begin to develop a detailed architecture plan that outlines the steps required to implement the architecture. This includes identifying the key stakeholders, developing an implementation roadmap, and identifying the resources required to support the architecture.

Throughout the development process, the organization uses a range of TOGAF tools and techniques, such as architecture principles, architecture patterns, and architecture building blocks, to ensure that the architecture aligns with the organization's goals and objectives.

## "Enterprise architecture facilitates collaboration with project management. Justify the statement with proper explanation.

Enterprise architecture (EA) can help facilitate collaboration with project management by providing a common understanding of the organization's IT landscape, its goals and objectives, and the interrelationships between business processes, information, applications, and technology. This understanding can help align projects with the broader goals and objectives of the organization, enabling project teams to work more effectively with one another.

EA can provide a structured approach to project management by defining standard processes, methodologies, and tools for project initiation, planning, execution, and control. By providing a common language and set of processes, EA can help to reduce confusion and misunderstandings between project teams, enabling them to work more collaboratively and efficiently.

For example, if an organization wants to implement a new customer relationship management (CRM) system, the EA framework can help to identify the business processes that will be affected by the new system, the data flows between different systems, and the technical infrastructure needed to support the new system. This information can be used by project managers to develop a detailed project plan that takes into account the wider context of the organization and the interrelationships between different systems.

In this way, EA can help project managers to avoid potential conflicts and ensure that their projects align with the broader goals and objectives of the organization, leading to more successful outcomes.

## Outline the working of enterprise architecture models with diagram.

## Illustrate how can you make good enterprise architectural model? Take the use case to better illustrate the analogy.

To make a good enterprise architectural model, the following steps can be taken:

1. Define the scope and objectives: The first step is to define the scope of the architecture and the objectives that it aims to achieve.
2. Identify stakeholders: Identify all the stakeholders involved and their concerns, as it helps in prioritizing requirements.
3. Define architectural principles: Establishing architectural principles provides guidance for decision making and designing the architecture.
4. Develop architecture views: Develop various architecture views to depict different aspects of the enterprise, such as business, data, application, and technology architecture.
5. Develop architecture roadmap: Develop an architecture roadmap to depict the sequence in which architecture changes will be implemented.
6. Review and communicate the architecture: Regularly review and communicate the architecture to ensure that it aligns with the organizational objectives and stakeholders' requirements.

To better illustrate the analogy, consider a use case of a retail company that wants to implement a new e-commerce platform. The enterprise architectural model for this use case can be developed as follows:

1. Define the scope and objectives: The scope of the architecture is to design an e-commerce platform that enables the company to sell its products online and increase its customer base. The objective is to provide a seamless shopping experience to customers and increase the company's revenue.
2. Identify stakeholders: The stakeholders involved are the company's management, IT team, marketing team, and customers.
3. Define architectural principles: The architectural principles include security, scalability, usability, and maintainability.
4. Develop architecture views: Develop business, data, application, and technology architecture views to depict the various aspects of the enterprise.
5. Develop architecture roadmap: Develop an architecture roadmap to implement the e-commerce platform in phases. The first phase can be to design and develop the front-end website, followed by the integration of the backend systems.
6. Review and communicate the architecture: Regularly review and communicate the architecture to ensure that it aligns with the company's objectives and stakeholders' requirements. This ensures that the e-commerce platform meets the needs of the customers and generates the desired revenue for the company.

## Discuss the approach used by TOGAF to EAF's. List the benefits of using TOGAF.

TOGAF (The Open Group Architecture Framework) is an industry standard framework for enterprise architecture. It provides a common language, methodology, and tools for designing and managing enterprise architecture. TOGAF uses an iterative approach to develop the architecture, starting with a high-level view and then gradually refining it.

The approach used by TOGAF to EAF's (Enterprise Architecture Frameworks) is based on four key components:

1. Architecture Development Method (ADM): ADM is a step-by-step process for developing and implementing an enterprise architecture. It consists of nine phases, including planning, scoping, and defining the architecture.
2. Architecture Content Framework: This framework defines the various artifacts that are used to describe the enterprise architecture, including models, diagrams, and documents.
3. Architecture Capability Framework: This framework defines the people, processes, and tools required to develop and manage the enterprise architecture.
4. Architecture Governance Framework: This framework provides the structure for managing the enterprise architecture, including defining the roles and responsibilities of the various stakeholders.

The benefits of using TOGAF include:

1. Standardization: TOGAF provides a standard language and methodology for enterprise architecture, making it easier to communicate and collaborate across different departments and organizations.
2. Flexibility: TOGAF is designed to be flexible and adaptable to different organizations and industries, allowing for customization and optimization of the framework to meet specific needs.
3. Efficiency: The iterative approach of TOGAF helps to minimize rework and streamline the development process, resulting in faster and more efficient delivery of enterprise architecture.
4. Cost Savings: By providing a common framework for enterprise architecture, TOGAF can help to reduce duplication of effort and eliminate unnecessary spending on redundant or conflicting systems.

Overall, the use of TOGAF can help organizations to better align their business and IT strategies, improve communication and collaboration across departments, and optimize their enterprise architecture to support business objectives.

## Describe in briefly:

## Zachman Framework

The Zachman Framework is an enterprise architecture framework that provides a structured way of organizing and classifying architectural artifacts. It defines a matrix-like structure consisting of six perspectives and six levels of abstraction. The perspectives are Who, What, Where, When, Why, and How, while the levels of abstraction are Scope, Business Model, System Model, Technology Model, Detailed Representations, and Working System. Each cell in the matrix represents a specific architectural artifact that needs to be developed, managed, and maintained throughout the lifecycle of the enterprise architecture.

## Importance of enterprise architecture

Enterprise architecture is important because it provides a structured approach to managing the complexity of an organization's systems, processes, and technology infrastructure. It helps align business objectives with IT strategy, identifies areas for improvement and optimization, and supports decision-making processes. Enterprise architecture also facilitates communication and collaboration between different stakeholders, helps manage risk, and supports innovation and agility.

## Unified Architecture Framework

The Unified Architecture Framework (UAF) is an enterprise architecture framework that provides a comprehensive and integrated view of the architecture of a system or organization. It is based on the UML (Unified Modeling Language) standard and supports modeling of systems at different levels of abstraction, including operational, system, and software levels. UAF includes a set of viewpoints that represent different perspectives of the system, such as operational, system, software, hardware, and security. It also includes a set of architecture patterns and best practices that can be used to guide the development of architectures for complex systems. UAF provides a flexible and adaptable framework that can be used across different domains and industries.