**1.** **Define Service Oriented Architecture. Provide some benefits of this approach.**

Service Oriented Architecture is a style of computer software design where services are providing to other components by using application components. Service Oriented Architecture provides many benefits in this approach are platform independence, easy testability, more reliable, service reusability (losing couple and reused in multiple applications), transparency (change its location at any time and consumers will still be able to locate the service), parallel development (more parallelism in development environment as SOA and reduce the time it takes to develop the software), high availability, and better scalability.

**2.** **What are some concerns that need to be dealt with when using Service Oriented Architectures?**

There are some concerns that needs to be dealt with when using Service Oriented Architectures such as increased overhead (using multiple services would overload the system with extra computation), high investment cost (costly in terms of human resources), development through technology, and complex service management (web service sends and receives messages and information frequently into million requests for a single application).

**3.** **What is a Data-Centric Service?**

Data-Centric Service handles persistent data which is included in the storage and retrieval of data, locking mechanisms, and transaction management. For example, developers write applications that can read and update entries in this data space.

**4.** **What is a Logic/Message-Centric Service?**

Logic/Message-Centric Service services encapsulate algorithms and formulas for complex business rules. It approaches takes many forms, from simple direct transmission to more complex message queue and transactional systems.

**5.** **What are the 4 most common layers of a Service Oriented Architecture and what is the purpose of each? What are some design issues for each layer?**

**Presentation Layer** - This layer contains the user changing business processes require adjustable, interoperable, and flexible user interfaces. One issue might often become so difficult to develop complex graphic interfaces, this task is handled often by various programmers.

**Service Layer** - This layer is considered as a bridge between higher and lower layers, characterized by the number of services carrying out an individual business function. There are some design issues are loosely coupled, design only for the service contract, sanctity of the API, idempotency, commutativity, and others are similar to Business Layer. There are limitations to the data service layer such as the common knowledge of all tables must have a primary key, but multi-column primary keys are not supported for the base tables. You should never read or write the same data while using JDBCQueryService and PhysicalDataContainer within the same transaction or else you will most likely have a corrupted data.

**Business/Enterprise Layer** - An information exchange flow between participants (individual users and business entities) and so on to achieve an enterprise goal. There are some design issues are authentication (helps avoid it if will be used only by a presentation layer or service layer on the same tier within a trusted boundary), authorization (helps avoid impersonation and delegation which might affect performance and scaling opportunities.), coupling and cohesion. A good weakness with the database would be once you select a database, you will be locked in with that specific database possibly forever and cannot change your mind in later future.

**Data/Information Layer**- Provides a uniform way of representing, accessing, managing, analyzing, integrating data and information across information sources. Potential issues of this data layer can include leaky abstraction, code duplication, and abstraction inversion.

There are some design issues are components are loosely coupled (it accesses another component does not require knowledge of the data structures, transaction management, and others.), configurable, interoperable, and location independent.

**6. What are crosscutting concerns? List 3 different ones and at least one suggestion for how to deal with each?**

Cross-cutting concerns are like an aspect of a program affecting other concerns in design and implementation. Cross-cutting concerns are parts of a program that requires to respond to or affect other parts of the system. One example of crosscutting concerns is logging, you should always log your works in all functions and modules although not too much. Another example of crosscutting is security, there is not much common business logic but it would be safer, in my opinion, to separate data connections for security checks. Last example of crosscutting is data integrity, the data integrity of a software is becoming more of an importance to shield from other nosy users out of business, you can do so by creating a full clone or snapshots in virtualization.

**7. What is Tight-Coupling? What is Loose-Coupling?**

Tight coupling means that one class is heavily dependent on another class. There is hard-coded dependency declared in methods. Loose coupling means one class is dependent on the interface rather than the class. For loose coupling, we pass the dependency externally at runtime rather hardcoded.

**8. What is Cohesion?**

Cohesion is one of the good design principles. Cohesion is a class that does this task and other relevant tasks but this only task very well in these fundamentals: reusability, maintainability, and simplicity. If cohesion goes higher, the more reusability, maintainability, simplicity, and the better it is.

**9. What is the difference between Authentication and Authorization?**

Authentication verifies that who you say you are while authorization decides whether you should have access to a resource. In a software engineering world, authentication would be login form, HTTP authentication, HTTP digest, X.509 certificates, and other custom authentication and authorization methods are access controls for URLs, secure objects and methods, and going through access controls lists (Networking ACL).