1. What are the main benefits of SOA?

Greater business agility; faster time to market: The efficiency of assembling applications from reusable service interfaces, rather than rewriting and reintegrating with every new development project, enables developers to build applications much more quickly in response to new business opportunities.

1. How can you achieve loose coupling in SOA

The concept of [loose coupling](https://en.wikipedia.org/wiki/Loose_coupling) within [SOA](https://en.wikipedia.org/wiki/Service-oriented_architecture) is directly influenced by the object-oriented design paradigm,[[4]](https://en.wikipedia.org/wiki/Service_loose_coupling_principle#cite_note-Bernhard-4) whereby the objective is to reduce coupling between classes in order to foster an environment where both the classes, although somehow related to each other, can be changed in a manner that such a change does not break the existing relationship, which is necessary for the working of a software program.

1. Are web services and SOA the same?

By definition, a Web service uses the web to communicate its business question. SOA, on the other hand, is the architectural framework that enables a series of those Web services to occur.

1. What is a reusable service

The service reusability principle is a design principle, applied within the service-orientation design paradigm, to create services that can be reused across a business. These reusable services are designed so that their solution logic is independent of any particular business process or technology.

1. What are the disadvantages of SOA

**High Bandwidth Server** – As therefore net service sends and receives messages and knowledge often times so it simply reaches high requests per day. So it involves a high-speed server with plenty of information measure to run an internet service.

**Extra Overload** – In SOA, all inputs square measures its validity before it’s sent to the service. If you are victimization multiple services then it’ll overload your system with further computation.

**High Cost**– It is expensive in terms of human resource, development, and technology.

1. What is ESB and where does it fit in?

An Enterprise Service Bus (ESB) is fundamentally an architecture. It is a set of rules and principles for integrating numerous applications together over a bus-like infrastructure. ESB products enable users to build this type of architecture, but vary in the way that they do it and the capabilities that they offer. An enterprise service bus (ESB) is a middleware tool used to distribute work among connected components of an application. ESBs are designed to provide a uniform means of moving work, offering applications the ability to connect to the bus and subscribe to messages based on simple structural and business policy rules.

1. In SOA do we need to build a system from scratch?

No. If you need to integrate or make an existing system as a business service, you just need to create loosely coupled wrappers which will wrap your custom systems and expose the systems functionality in a generic fashion to the external world.

1. What is the most important skill needed to adopt SOA ?technical or cultural?

Surely cultural. SOA does require people to think of business and technology differently.

1. List down the advantages of Microservices Architecture.

* Easier to Build and Maintain Apps. The key principle of microservices is simplicity.
* Organized Around Business Capabilities.
* Improved Productivity and Speed.
* Flexibility in Using Technologies and Scalability.
* Autonomous, Cross-functional Teams.

1. What are the best practices to design Microservices?

Design Patterns of Microservices

* 1. [Aggregator](https://www.edureka.co/blog/microservices-design-patterns#Aggregator)
  2. [API Gateway](https://www.edureka.co/blog/microservices-design-patterns#APIGateway)
  3. [Chained or Chain of Responsibility](https://www.edureka.co/blog/microservices-design-patterns#Chained)
  4. [Asynchronous Messaging](https://www.edureka.co/blog/microservices-design-patterns#AsynchronousMessaging)
  5. [Database or Shared Data](https://www.edureka.co/blog/microservices-design-patterns#Database)
  6. [Event Sourcing](https://www.edureka.co/blog/microservices-design-patterns#EventSourcing)
  7. [Branch](https://www.edureka.co/blog/microservices-design-patterns#Branch)
  8. [Command Query Responsibility Segregator](https://www.edureka.co/blog/microservices-design-patterns#CQRS)
  9. [Circuit Breaker](https://www.edureka.co/blog/microservices-design-patterns#CircuitBreaker)
  10. [Decomposition](https://www.edureka.co/blog/microservices-design-patterns#Decomposition)

1. How does microservice architecture works ?

Microservices are a popular software design architecture that breaks apart monolithic systems. Applications are built as collections of loosely coupled services. Each microservice is responsible for a single feature. They interact with each other through communication protocols such as HTTP and TCP.

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

Pros:

Better scalability

Faster development cycle (easier deployment and debugging)

Cons:

Poorer performance, as microservices need to communicate (network latency, message processing, etc.)

Harder to maintain the network (has less fault tolerance, needs more load balancing, etc.)

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

Monolithic architecture is a concept when complete software is developed as a single unit. All functionality, features and modules are developed and integrate and deployed in the form of a single entity. A single platform containing user interface, business logic and database layer on the same page. Monolithic application has a single shared database for each functionality and every feature combined in a single program as a unified model.

A monolithic architecture is comfortable and easily adopted by small teams and small projects. A lot of startups and small projects are used to developed using monolithic architecture. Its good for a project when the modules are interdependent and interconnected.

Service-oriented architecture some time is also known as **centralized oriented architecture**, is architecture when multiple, services which are also said as user agents are used or created interaction with a centralized system to make our traditional monolithic less burden and loosely coupled. A service-oriented architecture is specifically designed to communicate interoperable systems. Each service provides functionality on abstraction level, consider as a black box, self-contained to avoid the overhead of new development each time.

Service are abstract and can be developed on any technology. The communication between them can be done using a central point of connectivity called **Enterprise service bus.**ESB is taking care of all service and helping them to interact with each other.

Microservice architecture also considers as decentralised architecture, is much similar to SOA but not connected with any central service. Microservice architecture is an architecture which is build using autonomous services which communicate with each other using rest. Like **SOA**microservice not build on such service which can use again any other project as well instead its focus is to make each functionality separate as a service which should be micro and a self-contained, application, business, infrastructure and enterprise layer. Each service unit has its database.

All those microservice then join into UI microservice to show as a working application. From UI we call those microservices through rest calls.

1. What are the challenges you face while working Microservice Architectures?

Managing Microservices. As the number of microservices increases, managing them gets more challenging.

Monitoring.

Embracing DevOps Culture.

Fault Tolerance.

Testing.

Cyclic Dependencies.

1. What are the characteristics of Microservices?

Characteristics of a Microservice Architecture

* Componentization via Services. ...
* Organized around Business Capabilities. ...
* Products not Projects. ...
* Smart endpoints and dumb pipes. ...
* Decentralized Governance. ...
* Decentralized Data Management. ...
* Infrastructure Automation. ...
* Design for failure.

1. What are the basic Features of HTTP?

HTTP is connectionless

HTTP is stateless

HTTP is media independent

1. What are request methods in HTTP?

The primary or most-commonly-used HTTP verbs (or methods, as they are properly called) are POST, GET, PUT, PATCH, and DELETE. These correspond to create, read, update, and delete (or CRUD) operations, respectively.

1. What are the differences between GET and POST methods?

Both GET and POST method is used to transfer data from client to server in HTTP protocol but Main difference between POST and GET method is that GET carries request parameter appended in URL string while POST carries request parameter in message body which makes it more secure way of transferring data from client to server in http protocol.

1. What is status code in HTTP?

Status codes are issued by a server in response to a client's request made to the server.

1. What are the header fields in HTTP?

HTTP header fields are components of the header section of request and response messages in the Hypertext Transfer Protocol (HTTP). They define the operating parameters of an HTTP transaction.

1. What are Idempotent methods and why do we call them?

An idempotent HTTP method is a HTTP method that can be called many times without different outcomes. It would not matter if the method is called only once, or ten times over. The result should be the same. Again, this only applies to the result, not the resource itself.

1. Explain HTTP Request & Response Messages

HTTP works as a request-response protocol between a client and server. Example: A client (browser) sends an HTTP request to the server; then the server returns a response to the client. The response contains status information about the request and may also contain the requested content.

1. What is Session State in HTTP?

A session is a way to store information (in variables) to be used across multiple pages. Unlike a cookie, the information is not stored on the users computer.

1. What is HTTPS?

Hypertext Transfer Protocol Secure (HTTPS) is an extension of the Hypertext Transfer Protocol (HTTP). It is used for secure communication over a computer network, and is widely used on the Internet.

1. What is URI?

URI stands for Uniform Resource Identifier. URI provides a method for resources to be accessed by other systems over the World Wide Web or across a network.

1. Explain REST and RESTFUL?

Representational state transfer (REST) is a style of software architecture. As described in a dissertation by Roy Fielding, REST is an "architectural style" that basically exploits the existing technology and protocols of the Web. RESTful is typically used to refer to web services implementing such an architecture.

1. Mention what are the HTTP methods supported by REST?

HTTP methods supported by REST are:

* GET: It requests a resource at the request URL. It should not contain a request body as it will be discarded. Maybe it can be cached locally or on the server.
* POST: It submits information to the service for processing; it should typically return the modified or new resource
* PUT: At the request URL it update the resource
* DELETE: At the request URL it removes the resource
* OPTIONS: It indicates which techniques are supported
* HEAD: About the request URL it returns meta information

1. Explain the architectural style for creating web API?

REST architecture can be used for creating web API. REST stands for REpresentational State Transfer and API stands for Application Program Interface. REST is a software architectural style that defines the set of rules to be used for creating web services. Web services which follow the REST architectural style are known as RESTful web services.

1. Explain the RESTFul Web Service ?

**Restful Web Services** is a lightweight, maintainable, and scalable service that is built on the REST architecture. Restful Web Service, expose API from your application in a secure, uniform, stateless manner to the calling client. The calling client can perform predefined operations using the Restful service. The underlying protocol for REST is HTTP. REST stands for REpresentational State Transfer.

1. Explain what is a “Resource” in REST?

A resource in REST is a similar Object in Object Oriented Programming or is like an Entity in a Database. Once a resource is identified then its representation is to be decided using a standard format so that the server can send the resource in the above said format and client can understand the same format.

1. Which protocol is used by RESTful web services?

RESTful web services make use of HTTP protocol as a medium of communication between client and server.

1. What is messaging in RESTful web services?

RESTful Web Services make use of HTTP protocols as a medium of communication between client and server. A client sends a message in form of a HTTP Request and the server responds in the form of an HTTP Response. This technique is termed as Messaging.

1. State the core components of an HTTP Request?

The core components of an HTTP Request are:

HTTP version: Indicates version

Request Body: Represents message content

Request Header: Contains metadata, such as cache settings and client type, for the HTTP request message

URI: Identifies the resource on the server

Verb: Indicates HTTP methods such as GET, POST, and PUT

1. State the core components of an HTTP response?

The core components of an HTTP Response are:

HTTP version:  Indicates the present version of HTTP

Response Body: Represents the response message content

Response Header: Consists of metadata, like content length and server length, for the HTTP response message

Status/Response Code: Indicates the server status for the requested source.

1. What do you understand about payload in RESTFul web service?

The request data which is present in the body part of every HTTP message is referred to as 'Payload'. In Restful web service, the payload can only be passed to the recipient through the POST method.

1. Explain the caching mechanism?

Caching is a technique to speed up data lookups (data reading). Instead of reading the data directly from it source, which could be a database or another remote system, the data is read directly from a cache on the computer that needs the data.

1. List the main differences between SOAP and REST?

* SOAP stands for Simple Object Access Protocol whereas REST stands for Representational State Transfer.
* SOAP is a protocol whereas REST is an architectural pattern.
* SOAP uses service interfaces to expose its functionality to client applications while REST uses Uniform Service locators to access to the components on the hardware device.
* SOAP needs more bandwidth for its usage whereas REST doesn’t need much bandwidth.
* SOAP only works with XML formats whereas REST work with plain text, XML, HTML and JSON.
* SOAP cannot make use of REST whereas REST can make use of SOAP.

1. Enlist advantages and disadvantages of ‘Statelessness’.

The following are some advantages of statelessness:

* As the server does not need to manage any session, deploying the services to any number of servers is possible, and so scalability will never be a problem
* No states equals less complexity; no session (state) synchronize logic to handle at the server side
* As the service calls (requests) can be cached by the underlying application, the statelessness constraint brings down the server's response time, that is, it improves performance with regard to response time
* Seamless integration/implementation with HTTP protocols is possible as HTTP is itself a stateless protocol

A disadvantage of statelessness is that it may be necessary to include additional information in every request, and this extra information will need to be interpreted by the server.