**Software Development Life Cycle**

* Explain SDLC at a high level:
  + Software Development Life Cycle is a process used by software developers to organize the software development process. SDLC aims to structure the development process to be complete within the time and cost estimated. A typical SDLC consists of the following stages: planning, defining, designing, building, testing, and deployment.
* What is waterfall and why it is still relevant:
  + Waterfall is a software development model that breaks down the development process into phases. The Waterfall model consists of the following phases: requirements, design, implementation, verification and maintenance. This type of SDLC is recommended only when the final product is decided before the development process and no changes are allowed during the process.
* Explain Agile Model with a use case and the role of SCRUM in that:
  + Agile is a software development model that is more flexible than other traditional models. Instead of spending a large amount of time planning on a project, Agile instead relies on constant communication with customers to efficiently meet their requirements. For instance, when developing a project the software developer will meet with the customer to discuss the different ways they want to use the software application. Then the software developer takes a certain amount of time to develop the different use cases provided by the customer. Finally, the software developer returns to the customer to show the current state of the software application, and the customer reviews the application. If the customer approves, the changes are kept or re-develop according to their feedback. Scrum is a software development model that takes the main concepts of agile but is more suited for larger teams.
* Who is Scrum Master:
  + The scrum master is not a traditional project manager, they act ensures that the scrum framework is followed.
* Differentiate between Product/Sprint Backlog:
  + The product backlog is the breakdown of work to be done for the entire scrum team. The sprint backlog is the list of work the development team must address during the next sprint.
* What is Epic & Story:
  + Epic is a large task that probably will not be completed in a sprint. Story is a small task that can be completed in a sprint. Epic tasks are usually created at the start and later broken down into stories.
* What is called Velocity in SCRUM:
  + The total effort a team is capable of in a sprint. The velocity value is composed by evaluating how much a team completes in previous sprints. Velocity is used to understand how much work can be achieved in future sprints.
* Explain the SCRUM ceremonies:
  + SCRUM consists of four ceremonies which are sprint planning, daily scrum, sprint review, and sprint retrospective. Sprint planning is where the team meets and decides what they need to complete in the coming sprint. Daily SCRUM is a short meeting for the team to make sure they're all on the same page. Sprint review is a meeting where the team demos what they shipped in the sprint. Sprint retrospective is when the team reviews their work by identifying what they did well and what did not go as planned.
* What is grooming:
  + Grooming is the process where the backlog is reviewed/updated to make sure it contains all the appropriate items.
* How Jira board is effective in SCRUM:
  + Jira provides a real time view of the progress to all SCRUM roles, allowing an organization to verify a product is on track to meet release dates.
* Differentiate between SCRUM & Waterfall:
  + SCRUM is an agile based SDLC that puts tha main focus on the management and development of the project. SCRUM has a more flexible development process since changes to the requirements can happen during the development phase. Waterfall is a traditional SDLC where its approach is more linear and sequential. Waterfall is more restrictive during the development process since it only welcomes changes at the start and end of the SDLC.
* Explain the responsibilities of Product Owner:
  + Represents the voice of the customer, is responsible for delivering good business results, manages product backlog and maximizes the value that the team delivers.

**Client-Server**

* Tell us about the features of the client/server:
  + Data from the server can be accessed anywhere with network access, authentication is easier since all data is centralized, and transferring of data is easier through network connection.
* What is a Web server in a client server environment?:
  + Primary function of a web server is to store, process and deliver web pages to clients. The communication between client and server takes place using the hypertext transfer protocol.
* What is the role of the presentation layer:
  + The presentation layer acts as a translator between the application and the network. It translates data between the formats the network requires and the format for the computer client.
* They say this architecture is secure, how is it done in your opinion?:
  + Block connections from reaching the server by only authorizing some requests to go through. A login page would be a good example, to not let unauthorized computers from accessing server data without providing valid credentials.
* What is a Database Server in a client server environment?:
  + Database server is hardware/software responsible for database storage, access, and processing.
* What are Super servers in client server environments?:
  + Super server is a hardware/software that is responsible to start other servers when needed. Super server can be referred to as a service dispatcher.
* Explain 2-Tier and 3-Tier architecture:
  + 2-Tier Architecture, also known as open database connectivity, allows the client-side program to call the DBMS (Database Management System) directly. The 3-Tier Architecture is an extension of the 2-Tier Architecture. The 3-Tier Architecture has an application layer between the user and the DBMS. The application layer is responsible to handle the client's request to the DBMS system and return a response from the DBMS to the client. The added application layer allows rules to what data can be passed through from client to DBMS and back.
* What is a File server?:
  + File server is a central storage place for files that can be accessed to all authorized clients through a network connection.

**SOA & MicroServices**

* What are the main benefits of SOA?:
  + Services are reusable, easily maintained, reliable, and available across multiple clients.
* How can you achieve loose coupling in SOA:
  + Service Loose Coupling is a design principle that is applied to the services in order to ensure that the service contract is not tightly coupled to the service consumers and to the underlying service logic and implementation. This allows service contracts that could be freely evolved without affecting either the service consumers or the service implementation. The purpose of loose coupling is related to how related classes in OOP can be changed in a manner that such a change does not break the existing relationship.
* Are web services and SOA the same?:
  + Web services are software systems designed to support interoperable machine to machine interaction over a network. SOA based applications are distributed multi-tier applications that have presentation, business login and persistence layers. Web services are just software designed to be accessed through a network connection, while SOA is a method of how applications are designed.
* What is a reusable service?:
  + Service reusability principle is the concept that a service can be designed to be independent of any particular technology and be reused across a business.
* What are the disadvantages of SOA?:
  + High bandwidth server, extra overload and higher cost.
* What is ESB and where does it fit in?:
  + Enterprise Service Bus implements a communication system between mutually interacting software applications in a SOA. It represents a SOA for distributed computing and is a special variant of a more general client server model.
* 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.
* What is the most important skill needed to adopt SOA ?technical or cultural?:
  + Culture is the most important skill needed since SOA requires people to think of business and technology differently. Instead of thinking of technology first must first think about business functions and services. SOA will change businesses since organizations will change to service oriented.
* List down the advantages of Microservices Architecture.:
  + Faster to develop, easier to scale, isolated services have less chance of failing, easier to maintain.
* What are the best practices to design Microservices?:
  + Create separate data storage for each microservice program, do separate build for each microservice, deploy each microservice separately.
* How does Microservice Architecture work?:
  + Microservices is a software development technique that arranges an application as a collection of loosely coupled services. Microservices are completely isolated from each other and operate in their own environment. Individual applications communicate with each other through specific interfaces. There are various ways of isolating different services, for example container, virtual machines. Isolation is key in microservice architecture.
* What are the pros and cons of Microservice Architecture?:
  + Pros: improved scalability, better fault isolation, optimized scaling decisions, localized complexity, increased business agility, increased developer productivity. Cons: can be complex, requires careful planning, proper sizing is hard to calculate, little control over third party microservices, downstream dependencies are difficult to track.
* What is the difference between Monolithic, SOA and Microservices Architecture?:
  + Monolithic apps consist of interdependent, indivisible units and features very low development speed. SOA is broken into smaller, moderately coupled services, and features slow development. Microservices are very small, loosely coupled independent services and feature rapid continuous development.
* What are the challenges you face while working Microservice Architectures?:
  + Needs more collaboration, harder to test and monitor because of the complexity of the architecture, harder to maintain, does not function properly without the proper corporate culture, security issues.
* What are the characteristics of Microservices?:
  + Microservices is a software development technique that arranges an application as a collection of loosely grouped services. Some characteristics of microservices are: multiple individual deployed components encourage you to break you application into smaller components making is easier to apply future changes, decentralized makes future changes less likely to affect the entire application, service distribution a team can be in charge of different micro services instead of a team managing the entire service.

**HTTP Fundamentals**

* What are the basic Features of HTTP?:
  + HyperText Transfer Protocol is the protocol that allows web servers and browsers to exchange data over the web. The basic features of HTTP: media independent, connectionless, and stateless.
* What are request methods in HTTP?:
  + HTTP works as a request-response protocol between a client and server. There are several HTTP request methods: GET, POST, PUT, HEAD, DELETE, PATCH, OPTIONS.
* What are the differences between GET and POST methods?:
  + GET method is used to request data from a specified resource, and GET requests can be cached, ramin in the browser history, can be bookmarked, should never be used when dealing with sensitive data, have length restrictions, and are only used to request data. POST method is used to send data to a server to create/update a resource, and POST requests are never cached, do not remain in the browser history, cannot be bookmarked, and have no restrictions on data length.
* What is status code in HTTP?:
  + HTTP response status codes indicate whether a specific HTTP request has been successfully completed. There are five classes of responses: informational (100-199), successful (200-299), redirects (300-399), client errors (400-499), and server errors (500-599).
* What are the header fields in HTTP?:
  + HTTP headers let the client and the server pass additional information with an HTTP request or response. An HTTP header consists of its case-insensitive name followed by a colon then by its value.
* What is URI?:
  + Uniform Resource Identifier is a string of characters that unambiguously identifies a particular resource. The most common form of URI is the Uniform Resource Locator frequently referred to informally as a web address.
* 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. No matter if the method is called multiple times, the result of the method is the same.
* Explain HTTP Request & Response Messages:
  + HTTP request and responses use a generic message format of RFC 822 for transferring the required data. This generic message format consists of the following: a start line, zero or more header fields, an empty line, optional message body.
* What is Session State in HTTP?:
  + Session state is a method to keep track of a user session during a series of HTTP requests. Session state allows a developer to store data about a user as he/she navigates through a web application.
* What is HTTPS?:
  + HTTPS is a secure way to send data between a web server and a web browser.

**Introduction to API**

* Explain REST and RESTFUL?:
  + Representational State Transfer is a software architectural style that defines a set of constraints to be used for creating web services. Web services that conform to the REST architectural style, called RESTful web services, provide interoperability between computer systems on the internet.
* Mention what are the HTTP methods supported by REST?:
  + HTTP GET, POST, PUT, DELETE, and PATCH.
* Explain the architectural style for creating web API?:
  + APIs can be developed in any of the following styles: REST, GraphQL, Falcor, RPC, SOAP and gRPC.
* Explain the RESTFul Web Service?:
  + RESTful Web Services are basically REST Architecture based Web Services. In REST Architecture everything is a resource. RESTful web services are lightweight, highly scalable and maintainable and are very commonly used to create API for web-based applications.
* Explain what is a “Resource” in REST?:
  + The fundamental concept in any RESTful API is the resource. A resource is an object with a type, associated data, relationships to other resources, and a set of methods that operate on it. It is similar to an object instance in an object-oriented programming language, with the important difference that only a few standard methods are defined for the resource (corresponding to the standard HTTP methods) while an object instance typically has many methods.
* Which protocol is used by RESTful web services?:
  + RESTful web services use the HTTP protocol. The protocol serves as a medium of data communication between client and server. HTTP standard methods are used to access resources in RESTful web service architecture.
* 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.
* State the core components of an HTTP Request?:
  + An HTTP request has five major parts: verb, URI, HTTP version, request header, request body. Verb indicates the HTTP methods and URI is used to identify the resource on the server. The request header contains metadata for the HTTP request message as a key-value pair, and the request body contains the message content or the resource representation.
* State the core components of an HTTP response?:
  + An HTTP response has four major parts: status/response code, HTTP version, response header, response body. The status/response code indicates the server status for the requested resource. The response header contains metadata for the HTTP response message as key value pairs, and the response body contains a response message content or resource representation.
* What do you understand about payload in RESTFul web service?:
  + The request data which is present in the body 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. There is no limit of sending data as payload through the POST method but the only concern is that more data will consume more time and bandwidth.
* Explain the caching mechanism?:
  + Cache is a method for storing information so that it can be later accessed much more quickly. A caching mechanism helps improve delivery speed by storing a copy of the asset you requested and later accessing the cached copy instead of the original.
* List the main differences between SOAP and REST?:
  + REST has no official standard at all because it is an architectural style while has an official standard because it is a protocol. REST uses multiple standards like HTTP, JSON, URL, and XML while SOAP is largely based on HTTP and XML. REST deploys multiple standards so it takes fewer resources and bandwidth as compared to SOAP that uses XML for creation of payload and results in the large sized file.
* Enlist advantages and disadvantages of ‘Statelessness’.:
  + Advantages:
    - 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 and no session synchronized logic to handle at the server side.
    - As the service calls can be cached by the underlying application the statelessness constraint brings down the server’s response time which improves performance with regard to response time.
    - Seamless integration/implementation with HTTP protocols is possible as HTTP is itself a stateless protocol.
  + Disadvantages:
    - Stateless applications will not keep information about a particular user session.
    - Applications need to get extra information in each request and then interpret to get the client’s state in case the client interactions are to be taken care of.

**Object Oriented Programming Fundamentals**

* What is the main difference between a class and an object?:
* What is Encapsulation? Explain with a used case:
* What is Polymorphism? Explain with a used case:
* Explain Overriding & Overloading and its advantages:
* What is Inheritance and different types of inheritance? Explain with a used case:
* What is an abstract class?:
* What is an interface and how multiple inheritance is achieved with this:
* What are the access modifiers?:
* What are the various types of constructors?:
* What is ‘this’ pointer?:
* What is static and dynamic Binding?:
* How many instances can be created for an abstract class and why?:
* Which OOPS concept is used as a reuse mechanism and explain with a use case:
* Please identify one practical scenario for each pillar of OOPs.:

**Unit Testing & JUnit**

1. What is unit testing?
2. What is the difference between manual testing and automated testing?
3. Is it necessary to write the test case for every logic? If yes, why
4. What are the features of JUnit?
5. What are the important JUnit annotations? And its usage in coding
6. What does Assert class?
7. What is Code Coverage?
8. What are the best practices to perform Unit Testing?
9. What is Mocking?

**GIT**

1. What is GIT and its significance in SDLC
2. What is the difference between GIT and SVN?
3. What are the advantages of using GIT?
4. What is “Staging Area” or “Index” in GIT?
5. What is GIT stash?
6. What is the function of git clone?
7. How can you create a repository in Git?
8. What is the purpose of branching in GIT?
9. What is the difference between ‘git remote’ and ‘git clone’?
10. What is the function of ‘git diff ’ in git?
11. Explain what the commit message is?
12. Why is it advisable to create an additional commit rather than amending an existing commit?
13. What is Rebasing

**Maven Fundamentals**

1. Explain what is Maven? How does it work?
2. Explain what is POM and its significance
3. Explain what a Maven artifact is?
4. List out the dependency scope in Maven?
5. List out what are the build phases in Maven?
6. Mention the three build lifecycle of Maven?
7. List out what are the aspects does Maven Manages?
8. Explain what a Maven Repository is? What are their types?
9. Explain how you can exclude dependency?
10. For POM what are the minimum required elements?

**CI/CD**

1. What are the fundamental differences between DevOps & Agile?
2. What is the need for DevOps?
3. What are the advantages of DevOps?
4. Explain with a use case where DevOps can be used in industry/ real-life.
5. What are the success factors for Continuous Integration?
6. What are the differences between continuous integration, continuous delivery, and continuous deployment?
7. What role does the Quality Assurance (QA) team play in DevOps?
8. Describe an efficient workflow for continuous integration
9. What are the best practices for DevOps implementation?
10. How will you approach when a project needs to implement DevOps?