**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?:
  + A class is a template for objects. A class defines object properties including a valid range of values, and a default value. A class also describes object behavior. An object is a member or an “instance” of a class. An object has a state in which all of its properties have values that you either explicitly define or that are defined by default settings.
* What is Encapsulation? Explain with a used case:
  + Encapsulation is defined as the wrapping up of data under a single unit. It is the mechanism that binds together code and the data it manipulates. An example would be to declare private variables in a class and the only way to access these values of the variables is to declare public functions that return the value of the private variables. Only way to access the value of the private variables is through the public function.
* What is Polymorphism? Explain with a used case:
  + Polymorphism is the ability of an object to take many forms. The most common use of polymorphism in OOP occurs when a parent class reference is used to refer to a child class object (inheritance).
* Explain Overriding & Overloading and its advantages:
  + Overloading occurs when two or more methods in one class have the same method name but different parameters. Advantages of overloading: the function can perform different operations and hence eliminates the use of different functions name of the same kind of operations, program becomes easier to understand, and easy to maintain the code.
  + Overriding means having two methods with the same method name and parameters. One of the methods is in the parent class and the other is in the child class. Overriding allows a child class to provide a specific implementation of a method that is already provided by its parent class. Advantages of overriding: helps write generic code based on parent class or interface as object resolution happens in runtime, provides multiple implementation of same method and can invoke parent class overridden method using super keyword, defines what behavior a class can have and implementation of behavior has been take care by class which is going to implement.
* What is Inheritance and different types of inheritance? Explain with a used case:
  + Inheritance is the process of creating a new class (Derived Class) from an existing class (Base Class). Inheritance offers reusability of code, instead of developing new objects from scratch new code can be based on other classes and add only the new features needed. There are different types of inheritance: single, multi-level, hierarchical, hybrid, multipath, multiple. An example would be a class Teacher and a subclass Physics\_Teacher. The class Teacher can lay down the basic variables and features of a teacher while Physics\_Teacher can add to the Teacher class by inheriting from it.
* What is an abstract class?:
  + Abstract class is a restricted class that cannot be used to create objects, to access the abstract class it must be inherited from another class. The abstract class contains normal methods and abstract methods. The abstract methods can only be used in an abstract class and does not have a body. The body is provided by the subclass (inherited from).
* What is an interface and how multiple inheritance is achieved with this:
  + Inheritance can be defined as the process where one class acquires the properties of another class. With the use of inheritance the information is made manageable in a hierarchical order. Multiple inheritance is a feature of object oriented concepts where a class can inherit properties of more than one parent class.
* What are the access modifiers?:
  + The access modifiers specify the accessibility or scope of a field, method, constructor or class. There are four types of access modifiers: private, default, protected, and public.
* What are the various types of constructors?:
  + There are various types of constructors: default constructor, parameterized constructor, copy constructor, and destructor.
* What is ‘this’ pointer?:
  + This pointer is used to represent the address of an object inside a member function.
* What is static and dynamic Binding?:
  + Static binding is the binding which can be resolved at compile time. Dynamic binding is determining the method to invoke at runtime instead of compile time.
* How many instances can be created for an abstract class and why?:
  + No instances can be created for an abstract class, instead an instance is created for other classes that are extending from the abstract class.
* Which OOPS concept is used as a reuse mechanism and explain with a use case:
  + Inheritance is the OOPS concept that can be used as a reuse mechanism. An example would be a class Teacher and a subclass Physics\_Teacher. The class Teacher can lay down the basic variables and features of a teacher while Physics\_Teacher can add to the Teacher class by inheriting from it. The Teacher code can be reused for the class Physics\_Teacher with inheritance.
* Please identify one practical scenario for each pillar of OOPs.:
  + Abstraction: set up an abstract class to format classes that extend from it to a particular format.
  + Encapsulation: keep certain access from some variables, only can be accessed through a public function call.
  + Inheritance: reuse code when another class may have the same functions it can just be extended from base class.
  + Polymorphism: use when a certain task may be needed for different data types or objects.

**Unit Testing & JUnit**

* What is unit testing?
  + Unit testing is a way of testing a unit, the smallest piece of code that can be logically isolated in a system.
* What is the difference between manual testing and automated testing?
  + Manual testing is unit testing that is performed by people. Automated testing is unit testing that is performed without human intervention.
* Is it necessary to write the test case for every logic? If yes, why
  + Yes, the purpose of a test case is to ensure if different features within an application are working as expected. It helps validate the software is free of defects and if it is working as per the expectations of the end users.
* What are the features of JUnit?
  + JUnit is an open source unit testing tool used to test small or large units of code. JUnit provides the following features: fixtures, test suites, test runners, and JUnit classes.
* What are the important JUnit annotations? And its usage in coding
  + The test runner is used to execute the test cases.
    - @Test: this is the test method to run first unless otherwise specified.
    - @BeforeClass: this is runonce before any of the other test methods present in the class.
    - @Before: this is run before @Test.
    - @After: as the name suggests, this is run after the @Test.
    - @AfterClass: this is run one after all of the tests in the class have been run.
* What does Assert class?
  + Assert class provides a set of assertion methods useful for writing tests.
* What is Code Coverage?
  + Code coverage shows which lines of the code were (or were not) being executed by the tests.
* What are the best practices to perform Unit Testing?
  + Unit tests should be trustworthy: the test must fail if the code is broken and only if the code is broken, If it doesn’t, we cannot trust what the test results are telling us.
  + Unit tests should be maintainable and readable: when production code changes, tests often need to be updated, and possibly debugged as well. So it must be easy to read and understand the test, not only for whoever wrote it, but for other developers as well. Always organize and name you test for clarity and readability.
  + Unit tests should verify a single-use case: Good tests validate one thing and one thing only, which means that typically, they validate a single use-case. Tests that follow this best practice are simpler and more understandable, and that is good for maintainability and debugging. Tests that validate more than one thing can easily become complex and time-consuming to maintain.
* What is Mocking?
  + Mock testing is an approach to unit testing that lets you make assertions about how the code under test is interacting with other system modules. In mock testing, the dependencies are replaced with objects that simulate the behaviour of the real ones.

**GIT**

* What is GIT and its significance in SDLC
  + Git is a free and open source distributed version control system designed to handle everything from small to very large projects with speed and efficiency. Git is important for all code, files and assets that multiple team members will collaborate on. Using any version control software helps you keep track of changes and keep every team member working off the latest version.
* What is the difference between GIT and SVN?
  + Difference between Git and SVN version control systems is that Git is a distributed version control system, whereas SVN is a centralized version control system. Git uses multiple repositories including a centralized repository and server as well some local repositories. SVN does not have a centralized repository or server.
* What are the advantages of using GIT?
  + Advantages:
    - Distributed Model: Your work is your own, you can let others see only what is necessary. Not everything has to be public.
    - Branching and Merging are Easy
    - Workflow is Flexible: Compared to centralized VCS GIT has the qualities that allow you to choose your own workflow.
    - Data Integrity is Assured: Since GIT uses SHA1 trees, data corruption due to external reasons can be detected.
* What is “Staging Area” or “Index” in GIT?
  + The GIT Index is a staging area between your working directory and your repository. The index can be used to build up a set of changes that you want to commit together. When you create a commit what is committed is what is currently in the index, not what is in your working directory.
* What is GIT stash?
  + GIT stash temporarily shelves changes you’ve made to your working copy so you work on something else, and then come back and re-apply them later on.
* What is the function of git clone?
  + GIT close is primarily used to point to an existing repo and make a clone or copy of that repo in a new directory, at another location. The original repository can be located on the local filesystem or on remote machine accessible supported protocols.
* How can you create a repository in Git?
  + Directions:
    - Create a directory to contain the project
    - Go into the new directory
    - Run command *git init*
    - Make changes to directory
    - Run command *git add* to add the files
    - Run command *git commit*
* What is the purpose of branching in GIT?
  + Branches are a part of your everyday development process. GIT branches are effectively a pointer to a snapshot of your changes. When you want to add a new feature or fix you spawn a new branch to encapsulate your changes.
* What is the difference between ‘git remote’ and ‘git clone’?
  + *git remote* is used to refer to a remote repository or central repository. *git clone* is used to copy or clone a different repository.
* What is the function of ‘git diff ’ in git?
  + *git diff* is a multi-function GIT command which is used to compare changes committed in GIT.
* Explain what the commit message is?
  + A GIT commit is a change to code, a commit message is attached to that change. Accordingly when you write a commit message you are writing it as if it’s about to be applied, rather than about what you just did.
* Why is it advisable to create an additional commit rather than amending an existing commit?
  + The amend operation will destroy the state that was previously saved in a commit. If it’s just the commit message being changed then that’s not an issue, but if the contents are being amended then chances of eliminating something important increases. Abusing *git commit amend* can cause a small commit to grow and acquire unrelated changes.
* What is Rebasing
  + In GIT, the rebase command integrates changes from one branch into another. It is an alternative to the better known merge command.

**Maven Fundamentals**

* Explain what is Maven? How does it work?
  + Maven is a popular open source build tool for enterprise Java projects, designed to take much of the hard work out of the build process. Maven uses a declarative approach where the project structure and contents are described rather than the tasks based approach used in traditional make files.
* Explain what is POM and its significance
  + A Project Object Model is the fundamental unit of work in Maven, its an XML file that contains information about the project and configuration details used by Maven to build the project. It also contains default values for most projects.
* Explain what a Maven artifact is?
  + An artifact is a file that gets deployed to a Maven repository. Each artifact has a group ID, an artifact ID and a version string. A project’s dependencies are specified as artifacts.
* List out the dependency scope in Maven?
  + Maven dependency scope attribute is used to specify the visibility of dependency, relative to the different life cycle phases. Maven provides six scopes: compile, provided, runtime, test, system and import.
* List out what are the build phases in Maven?
  + A Maven phase represents a stage in the Maven build lifecycle.
    - Validate: check if all information necessary for the build is available
    - Compile: compile the source code
    - Test-Compile: compile the test source code
    - Test: run unit tests
    - Package: package compiled source code into the distributable format
    - Integration-Test: process and deploy the package if needed to run integration tests
    - Install: install the package to a local repository
    - Deploy: copy the package to the remote repository
* Mention the three build lifecycle of Maven?
  + There are built-in build lifecycles: default, clean and site. The default life cycle handles your project deployment, the clean life cycle handles project cleaning, white the site lifecycle handles the creation of your projects’ site documentation.
* List out what are the aspects does Maven Manages?
  + Maven handles following aspects:
    - Build
    - Documentation
    - Reporting
    - Dependencies
    - SCMs
    - Releases
    - Distribution
    - Mailing List
* Explain what a Maven Repository is? What are their types?
  + A Maven repository is a directory where all the project's jars, library jar, plugins or any other project specific artifacts are stored and can be used by Maven easily. Maven repositories are of three types: local, central, remote. Maven local repository is a folder location on your machine. Maven central repository is a repository provided by the Maven community. Maven remote repository is a developer’s own custom repository containing required libraries or other project jars.
* Explain how you can exclude dependency?
  + Open dependency POM and find the dependency you want to exclude. Surround the info of the dependency you want to exclude with an exclusions XML tag.
* For POM what are the minimum required elements?
  + groupId, artifactId and version

**CI/CD**

* What are the fundamental differences between DevOps & Agile?
  + Software development, testing, and deployment happen in both DevOps and Agile. However, Agile tends sto stop after these three states. DevOps includes operations which happen continually. In Agile separate people are responsible for developing, testing and deploying the software. In DevOps, the DevOps engineering roles are responsible for everything, deployment is operations, and operations is development.
* What is the need for DevOps?
  + Shorter development cycles and faster innovation, reduce implementation failure, reflections and recovery time. Better communication and cooperation.
* What are the advantages of DevOps?
  + Increase productivity of business and IT teams. Save costs on maintenance and upgrades, and eliminate unnecessary capital expenditure. Standardize processes for easy replication and faster delivery. Improve quality, reliability and reusability of all system components.
* Explain with a use case where DevOps can be used in industry/ real-life.
  + There are many industries that are using DevOps, for example: A peer to peer commerce website focused on vintage items and supplies. The e-commerce struggled with slow, painful site updates that frequently caused the site to go down and affected sales of millions of users. With the help of a new developer team the e-commerce migrated to a four hour full site deployments twice a week, a more agile approach. In the future they can average more deployments in a day with fewer disruptions.
* What are the success factors for Continuous Integration?
  + Smaller code changes, fault isolations, faster mean time to resolution, more test reliability, faster release rate, smaller backlog, customer satisfaction, and increased team transparency and accountability.
* What are the differences between continuous integration, continuous delivery, and continuous deployment?
  + Continuous integration is merging all code from all developers to one central branch of the repo many times a day trying to avoid conflicts in the code in the future. Continuous delivery is a software engineering practice in which teams develop, build, test and release software in short cycles. Continuous deployment is the process by which qualified changes in software code or architecture are deployed to production as soon as they are ready without human intervention.
* What role does the Quality Assurance (QA) team play in DevOps?
  + In the DevOps scenario, QA integrates into the testing and development process and takes the collaborative approach. Quality is ensured throughout the testing and delivery cycle and both the testing and development teams are responsible for it.
* Describe an efficient workflow for continuous integration
  + A typical continuous integration workflow:
    - A developer create a new branch of code in GitHub, makes changes in the code and commits them
    - When the developers pushes their work to GitHub, the code is build and is runned through an automated test suite
    - If any errors are detected the developer gets a notification
    - If no errors the code is pushed to the next pipeline allowing QA to test the changes in a production like environment
    - Once another develop has verified the changes the author can then merge the new branch of code into the master branch
* What are the best practices for DevOps implementation?
  + Have a centralized unit for DevOps, implement test automation, implement continuous deployment, keep all teams on the same page, exercise patience, implement automation in dashboards, monitor application’s performance, consider DevOps to be a cultural change.
* How will you approach when a project needs to implement DevOps?
  + Evaluate the need to implement DevOps practice, break the organizations & encourage collaboration, put customer/end-user satisfaction at the center, don’t jump start instead start small and then scale up, automate wherever possible, select tools that are compatible with each other and define performance review for team and an individual.