C#.NET, ASP.NET, MVC, .NET Core

1. **What is the difference between .NET Framework and .NET Core?**
    - .NET Framework is used to build Windows applications, while .NET Core is cross-platform and used to build modern cloud-based applications.
2. **Explain the lifecycle of an ASP.NET MVC request.**
    - The lifecycle includes receiving the request, routing, creating a controller instance, invoking an action method, and returning the response to the client.
3. **What is Dependency Injection in .NET Core?**
    - Dependency Injection (DI) is a design pattern that allows classes to declare their dependencies, which are then provided by a service container.
4. **What is LINQ and why is it useful?**
    - LINQ (Language Integrated Query) is a set of methods that provide a query capability over collections, reducing boilerplate code and increasing readability.
5. **Explain Model Binding in ASP.NET MVC.**
    - Model Binding in MVC automatically maps form fields from an HTTP request to corresponding model properties.
6. **What are Action Filters in ASP.NET MVC?**
    - Action Filters allow code to run before or after an action method executes, commonly used for logging, authentication, or handling exceptions.
7. **Explain the Repository pattern and why it's used.**
    - The Repository pattern provides an abstraction for data access, separating business logic from data access logic.
8. **How would you implement Dependency Injection in ASP.NET Core?**
    - Register services in `Startup.cs` under the `ConfigureServices` method and inject them into the required classes via constructor injection.
9. **What are the main differences between synchronous and asynchronous programming in C#?**
    - Synchronous code blocks the thread until the operation is complete, while asynchronous code allows other operations to run while waiting for the task to complete.
10. **How can you handle exceptions globally in ASP.NET Core?**
    - Use middleware or custom exception filters to handle exceptions across the entire application.

MS SQL

11. **What are the different types of joins in SQL?**
    - Inner Join, Left Join, Right Join, Full Outer Join, and Cross Join.
12. **Explain the difference between a clustered and non-clustered index.**
    - A clustered index sorts the data rows in the table based on the key values, while a non-clustered index creates a separate structure that references the table's data.
13. **What is a stored procedure, and why would you use one?**
    - A stored procedure is a precompiled collection of SQL statements that can be executed as a single unit, improving performance and reusability.
14. **How do you optimize a slow-running SQL query?**
    - Use indexes, avoid SELECT *, ensure proper joins, and check query execution plans for inefficiencies.
15. **What is a primary key and how does it differ from a unique key?**
    - A primary key uniquely identifies each record in a table and cannot contain null values, while a unique key also enforces uniqueness but can contain a single null value.

16. **How would you implement database normalization?**
    - Break tables into smaller, more manageable pieces, ensuring that each table contains only related data, to reduce redundancy.
17. **What is a trigger in SQL, and how does it work?**
    - A trigger is a set of instructions that automatically executes in response to certain events on a table or view.
18. **Explain the ACID properties in a database context.**
    - ACID stands for Atomicity, Consistency, Isolation, and Durability, which are guarantees that database transactions will be processed reliably.
19. **What is a foreign key constraint in SQL?**
    - A foreign key is a field in a table that uniquely identifies a row in another table, enforcing referential integrity between the two tables.
20. **What is the use of the `ROW_NUMBER()` function in SQL?**
    - The `ROW_NUMBER()` function assigns a unique sequential number to rows within a result set, often used for pagination.

HTML, CSS, JavaScript, Bootstrap, jQuery

21. **What is the difference between block-level and inline-level elements in HTML?**
    - Block-level elements take up the entire width of the container, while inline elements take only as much width as necessary.
22. **Explain CSS specificity and how it works.**
    - CSS specificity determines which CSS rule will be applied by assigning weights to different types of selectors (IDs, classes, elements).
23. **What are media queries in CSS?**
    - Media queries allow you to apply styles based on the device's properties like width, height, orientation, etc.
24. **What is jQuery, and why is it used?**
    - jQuery is a lightweight JavaScript library that simplifies HTML DOM manipulation, event handling, and AJAX interactions.
25. **How can you use JavaScript to prevent the default action of an element?**
    - You can use `event.preventDefault()` to stop the default action of an element.
26. **What is Bootstrap and what are its key components?**
    - Bootstrap is a front-end framework for responsive web design. Key components include the grid system, forms, buttons, and navigation bars.
27. **How do you perform AJAX requests using jQuery?** *$.ajax({ url: 'example.com', method: 'GET', success: function(data) { console.log(data); } });*
28. **What is the box model in CSS?**
    - The box model consists of content, padding, border, and margin, controlling the size and spacing of elements.
29. **How do you create a modal in Bootstrap?** *<button type="button" class="btn btn-primary" data-toggle="modal" data-target="#myModal">Open Modal</button>*
30. **What is event delegation in JavaScript?**
    - Event delegation allows you to attach a single event listener to a parent element that will handle events for all of its child elements.

Problem Solving and Analytical Skills

31. **Explain a time when you faced a complex problem and how you solved it.**

- Demonstrate a situation from a previous project where you used analytical skills to identify the root cause and implement an effective solution.

32. **How do you debug an issue in an application?**
    - Use tools like logs, debuggers, and exception handling to identify the cause and fix it.
33. **What approach do you take to database design?**
    - Identify entities, relationships, and constraints, then normalize the data to reduce redundancy.
34. **How do you handle performance issues in application development?**
    - Use profiling tools to identify bottlenecks, optimize code, implement caching, and consider database indexing.
35. **Explain a situation where you had to optimize an existing solution.**
    - Provide a real-world example where you improved the performance, scalability, or reliability of an application.
36. **What is the difference between `var`, `let`, and `const` in JavaScript?**
    - `var` has function scope, while `let` and `const` have block scope. `const` is read-only and cannot be reassigned.
37. **How would you troubleshoot a cross-browser compatibility issue?**
    - Use browser-specific developer tools, validate CSS/HTML, and test across multiple browsers using tools like BrowserStack.
38. **What is the purpose of design patterns in software development?**
    - Design patterns provide reusable solutions to common problems, improving code maintainability and flexibility.
39. **Explain how you would optimize database queries for large datasets.**
    - Index tables, use partitioning, optimize SQL queries, and avoid unnecessary joins or subqueries.
40. **How do you manage security vulnerabilities in an application?**
    - Implement input validation, use encryption for sensitive data, and follow secure coding practices.

SSIS, Agile, Azure DevOps, Git

41. **What is SSIS and how is it used in ETL processes?**
    - SSIS (SQL Server Integration Services) is used to extract, transform, and load data from multiple sources to destinations.
42. **How do you handle version control in Git?**
    - Use branching, commits, merges, and pull requests to manage different versions of the code.
43. **What are the key ceremonies in Agile methodology?**
    - Sprint Planning, Daily Standups, Sprint Reviews, and Retrospectives are core ceremonies in Agile.
44. **What is the purpose of Continuous Integration (CI) in Azure DevOps?**
    - CI ensures that code changes are automatically tested and integrated, reducing integration issues.
45. **What is the role of a Product Owner in Agile?**
    - The Product Owner manages the product backlog and ensures the team delivers value according to business priorities.
46. **How do you create a pull request in Git?** *git checkout -b new-branch; git push origin new-branch; open a pull request on GitHub.*
47. **How would you create a pipeline in Azure DevOps?**
    - Use the Pipelines service in Azure DevOps, define build and release pipelines using YAML or the graphical interface.
48. **What is Continuous Delivery (CD) in DevOps?**

- CD automates the release process so that code changes are automatically deployed to production environments.

49. **What is the difference between Git and GitHub?**
    - Git is a version control system, while GitHub is a cloud-based platform that hosts Git repositories.
50. **What are the benefits of using Agile over traditional waterfall methods?**
    - Agile allows for faster iterations, continuous feedback, and adaptability to changing requirements.

Communication Skills

51. **How do you explain technical concepts to non-technical stakeholders?**
    - Use simple language, avoid jargon, and focus on how the technical solution meets business needs.
52. **How would you handle a situation where a client doesn't understand the requirements clearly?**
    - Ask clarifying questions, restate their requirements in simple terms, and confirm their understanding before proceeding.
53. **Describe a situation where you had to collaborate with a difficult team member.**
    - Discuss how you maintained professionalism, addressed the issue, and found common ground to work together.
54. **How do you handle feedback from clients?**
    - Listen actively, acknowledge their concerns, and work towards implementing their feedback in a constructive manner.
55. **How do you keep stakeholders informed about project progress?**
    - Use regular status updates, meetings, and reports to ensure transparency and manage expectations.

Object-Oriented Programming (OOP) and 2/3-Tier Architecture

56. **What are the four pillars of OOP?**
    - Encapsulation, Abstraction, Inheritance, and Polymorphism.
57. **Explain the concept of Polymorphism with an example.** *public class Animal { public virtual void Speak() { Console.WriteLine("Animal sound"); } } public class Dog : Animal { public override void Speak() { Console.WriteLine("Bark"); } }*
58. **What is Encapsulation and why is it important?**
    - Encapsulation hides the internal implementation of an object, exposing only what is necessary and protecting the object's integrity.
59. **How does Inheritance work in C#?**
    - Inheritance allows one class to inherit fields and methods from another class, enabling code reuse.
60. **What is Abstraction in OOP?**
    - Abstraction simplifies complex reality by modeling classes based on essential properties and behaviors, hiding the complexity from the user.
61. **What is the difference between an interface and an abstract class?**
    - An interface only defines signatures of methods, while an abstract class can provide default behavior along with method signatures.
62. **How do you implement 2-tier and 3-tier architecture?**
    - 2-tier architecture has a client and a server, while 3-tier architecture divides the application into Presentation, Business Logic, and Data Access layers.
63. **What are design patterns? Name a few commonly used ones.**
    - Design patterns are reusable solutions to common problems in software design. Common patterns include Singleton, Factory, and Repository.
64. **What is SOLID principle in OOP?**

- SOLID is an acronym for five design principles that help make software designs more understandable, flexible, and maintainable.

65. **How do you implement Dependency Injection in 3-tier architecture?**
    - Use constructor injection to inject dependencies like repositories or services into the business logic and presentation layers.

Insurance Domain Knowledge

66. **What is underwriting in the Insurance domain?**
    - Underwriting is the process by which an insurer evaluates the risk of insuring a client and decides the premium to charge.
67. **What are the main types of insurance policies?**
    - Life insurance, health insurance, property insurance, and liability insurance are the main types.
68. **What is a premium in insurance terms?**
    - A premium is the amount of money that an individual or business must pay for an insurance policy.
69. **Explain the concept of risk pooling in insurance.**
    - Risk pooling involves spreading risks among a large group of policyholders, allowing the insurer to absorb the cost of claims.
70. **What is a claim in insurance?**
    - A claim is a formal request made by the insured to the insurer for compensation of a covered loss or policy event.
71. **What is reinsurance and why is it important?**
    - Reinsurance is insurance for insurance companies, allowing them to mitigate risk by passing on some of the risk to other insurers.
72. **What are the key components of an insurance policy?**
    - The key components are the policyholder, the insurer, the premium, the coverage, and the term of the policy.
73. **What are deductibles in insurance?**
    - A deductible is the amount the insured must pay out-of-pocket before the insurer pays for the remaining covered loss.
74. **How does the concept of moral hazard apply to insurance?**
    - Moral hazard refers to the increased likelihood of risk-taking by the insured because they are protected by the insurance policy.
75. **What is an actuary and what role do they play in the insurance domain?**
    - An actuary is a professional who uses mathematical models to assess and manage financial risks, particularly in the context of insurance and pensions.

Advanced Topics

76. **How do you manage state in an ASP.NET Core application?**
    - Use session, cookies, or TempData to store state across multiple requests in an ASP.NET Core application.
77. **What is middleware in ASP.NET Core?**
    - Middleware is software that processes requests and responses in the HTTP pipeline.
78. **Explain what a `DbContext` is in Entity Framework Core.**
    - `DbContext` is the primary class for interacting with the database using Entity Framework Core, responsible for querying and saving data.
79. **How do you create an API in ASP.NET Core?**

- Use the `Controller` class, define routes using attributes, and return JSON or XML data in response to HTTP requests.

80. **What is AutoMapper and why would you use it in an ASP.NET Core application?**
    - AutoMapper is a library that automatically maps objects of one type to another, simplifying object transformation in the business layer.

Code-based Questions

81. **How do you handle null values in LINQ?** *var result = myList?.Where(x => x != null);*
82. **Write a LINQ query to select all customers with orders over $100.** *var customers = dbContext.Customers.Where(c => c.Orders.Any(o => o.Amount > 100));*
83. **How do you create a foreign key relationship in Entity Framework Core?** *modelBuilder.Entity<Order>().HasOne(o => o.Customer).WithMany(c => c.Orders).HasForeignKey(o => o.CustomerId);*
84. **Write a C# program to find the sum of all even numbers in a list.** *var evenSum = numbers.Where(n => n % 2 == 0).Sum();*
85. **Write an asynchronous method to read from a file in C#.** *public async Task<string> ReadFileAsync(string path) { using (var reader = new StreamReader(path)) { return await reader.ReadToEndAsync(); } }*

Miscellaneous

86. **What is version control, and why is it important in software development?**
    - Version control is the practice of managing changes to source code over time, allowing teams to collaborate and revert changes if necessary.
87. **What is the difference between `GET` and `POST` in HTTP?**
    - `GET` requests data from a server, while `POST` submits data to be processed to a server.
88. **Explain the differences between `IEnumerable` and `IQueryable`.**
    - `IEnumerable` is used for in-memory collection iteration, while `IQueryable` is used for querying databases with deferred execution.
89. **What is the purpose of caching in web applications?**
    - Caching improves performance by storing frequently accessed data, reducing the load on the server.
90. **Explain the difference between optimistic and pessimistic concurrency control in SQL.**
    - Optimistic concurrency control assumes no conflict will occur, whereas pessimistic control locks records to prevent conflicts.

SSIS and Azure DevOps

91. **What is the purpose of control flow in SSIS?**
    - Control flow defines the order in which tasks are executed in an SSIS package.
92. **How do you deploy an SSIS package?**
    - Use the SQL Server Data Tools (SSDT) to deploy SSIS packages to the Integration Services server or file system.
93. **What is the difference between Continuous Integration and Continuous Delivery?**
    - Continuous Integration ensures that code changes are automatically tested and integrated, while Continuous Delivery ensures that code can be deployed to production at any time.
94. **Explain the concept of a "sprint" in Agile methodology.**
    - A sprint is a time-boxed period, typically 2-4 weeks, during which the Agile team completes a set of tasks or stories from the product backlog.
95. **What are the benefits of using containers in application deployment?**

- Containers provide consistent environments across development, testing, and production, reducing dependency conflicts.

Git and Agile

96. **How do you resolve merge conflicts in Git?** *git merge feature-branch; git status; open the conflicted files, resolve conflicts, then commit.*
97. **Explain the difference between a `git fetch` and `git pull`.**
    - `git fetch` retrieves the latest changes from the remote without merging, while `git pull` fetches and merges the changes.
98. **What are user stories in Agile?**
    - User stories are short descriptions of functionality from the perspective of the user, helping the team understand and implement requirements.
99. **How do you revert a commit in Git?** *git revert <commit-hash>* undoes the changes of a specific commit by creating a new commit with the inverse changes.
100. **What is a burn-down chart in Agile?** - A burn-down chart tracks the amount of work remaining in a sprint, helping the team monitor progress.