4. Creating ASP.NET core MVC applications

1. Create a Controller named Product with actions List, Details, and Create.

2. Design a Razor view to display a list of products using tag helpers.

3. Implement model binding to accept product details from a form and display them.

5. Working with Database

1. Connect to a database using ADO.NET and fetch a list of products.

2. Implement Entity Framework Core to map the Product class to a database table.

6. State Management on ASP.NET Core Application

1. Store and retrieve user data using Session State.

2. Implement TempData to display one-time messages after performing actions like "Product Added Successfully".

7. Client-side Development in ASP.NET Core

1. Implement a form to capture user input and validate the input using JQuery before submitting.

8. Securing in ASP.NET Core Application

1. Set up ASP.NET Core Identity to register and authenticate users.

9. Hosting and Deploying ASP.NET Core Application

1. Deploy your ASP.NET Core application to a local IIS server.

***Do all from unit 2 to 9 .but I will provide last set practical to all.[red selected questions]***

All others:

Unit 1: Language Preliminaries (8Hrs.) Theory:

* Overview of the .NET framework: Introduction to the .NET framework and its components.(v.v.i)
* Compilation and execution of .NET applications: Understanding the compilation process and how .NET applications are executed.
* Concepts like constructors, properties, arrays, strings, indexers, and inheritance. .(v.v.i)
* The use of keywords like "base" for calling base class members, method hiding, and method overriding. .(v.v.i)
* Applying polymorphism for code extensibility. .(v.v.i)
* Working with abstract classes, sealed classes, interfaces, delegates, and events. .(v.v.i)
* Understanding partial classes and their usage. .(v.v.i)
* Introduction to collections and generics. .(v.v.i)
* File input/output operations. .(v.v.i)
* Handling exceptions using try statements. .(v.v.i)

Practical:

* Implementing classes with constructors, properties, and methods. .(v.v.i)
* Creating and using arrays and strings.
* Implementing inheritance and polymorphism in class hierarchies. .(v.v.i)
* Working with abstract classes, interfaces, and sealed classes. .(v.v.i)
* Implementing delegates and events for event-driven programming. .(v.v.i)
* Utilizing collections and generics for data storage and retrieval.
* Reading and writing data to files. .(v.v.i)
* Handling exceptions using try-catch blocks. .(v.v.i)

Unit 2: Introduction to ASP.NET (3 Hrs.) Theory:

* Overview of .NET and ASP.NET frameworks: Understanding the different frameworks available within the .NET ecosystem, including .NET Core, Mono, ASP.NET Web Forms, ASP.NET MVC, ASP.NET Web API, and ASP.NET Core.
* .NET architecture and design principles: Exploring the architecture and design principles underlying the .NET framework. .(v.v.i)
* Compilation and execution of .NET applications: Understanding the Common Language Infrastructure (CLI), Microsoft Intermediate Language (MSIL), and Common Language Runtime (CLR). .(v.v.i)
* Introduction to .NET Core: Overview of .NET Core and its features.
* Working with .NET CLI: Building, running, testing, and deploying .NET Core applications. .(v.v.i)

Practical:

* Creating a simple ASP.NET application using ASP.NET Core. .(v.v.i)
* Setting up the development environment and project structure. .(v.v.i)
* Building and running the ASP.NET application using .NET CLI. .(v.v.i)
* Testing and deploying the ASP.NET Core application.

Unit 3: HTTP and ASP.NET Core (3 Hrs.) Theory:

* Understanding the HTTP protocol and its request and response message format. .(v.v.i)
* Common web application architectures and patterns.
* Overview of the Model-View-Controller (MVC) pattern.
* ASP.NET Core architecture overview, including projects and conventions.
* Differences between ASP.NET and ASP.NET MVC. .(v.v.i)

Practical:

* Building a basic ASP.NET Core application using the MVC pattern. .(v.v.i)
* Handling HTTP requests and generating appropriate responses. .(v.v.i)
* Implementing controllers, views, and models.
* Configuring routing for URL mapping.
* Understanding the project structure and conventions in ASP.NET Core.

1. Q: What is the .NET framework? Describe its components and their roles.
2. Q: Explain the compilation process of .NET applications.
3. Q: What are constructors in C#? How are they used in object initialization?
4. Q: Define properties in C# and explain their significance in encapsulation.
5. Q: What are arrays in C#? How are they declared and accessed?
6. Q: Describe the concept of strings in C#. How are they different from other data types?
7. Q: What are indexers in C#? How do they enable array-like access to objects?
8. Q: Explain the concept of inheritance in object-oriented programming. How does it facilitate code reuse?
9. Q: How is the "base" keyword used in C#? Provide an example.
10. Q: Discuss method hiding and method overriding in C#. What is the difference between the two?
11. Q: How is polymorphism applied in C# to achieve code extensibility?
12. Q: Describe abstract classes and sealed classes in C#. What are their purposes and limitations?
13. Q: What is an interface in C#? How is it different from an abstract class?
14. Q: Define delegates and events in C#. How are they used for event-driven programming?
15. Q: Explain the concept of partial classes and their usage in C#.
16. Q: Describe collections in C#. What are some commonly used collection types?
17. Q: What are generics in C#? How do they enable type-safe and reusable code?
18. Q: Discuss file input/output operations in C#. How can files be read from and written to?
19. Q: How are exceptions handled in C#? Explain the purpose and usage of try statements.
20. Q: What are attributes in C#? Describe attribute classes, named and positional attribute parameters, attribute targets, and specifying multiple attributes.

These questions cover the various topics mentioned in Unit 1, providing a comprehensive understanding of the language preliminaries in the .NET framework.

1. Q: Write a C# program to demonstrate the use of constructors to initialize object properties.
2. Q: Create a class representing a student with properties like name, age, and grade. Implement a method to display the student's details.
3. Q: Write a program that uses arrays to store and retrieve a list of names entered by the user.
4. Q: Implement a C# program to reverse a given string using string manipulation techniques.
5. Q: Create a class representing a book with properties like title, author, and price. Implement an indexer to access the book properties using an index.
6. Q: Develop a C# program to demonstrate inheritance by creating a base class "Shape" and derived classes like "Circle" and "Rectangle" that inherit from it.
7. Q: Implement a C# program to demonstrate the use of the "base" keyword to call base class members from derived classes.
8. Q: Create a base class "Vehicle" with a method "Drive" and a derived class "Car" that hides the "Drive" method. Write a program to demonstrate method hiding in action. THIS ONE DONE
9. Q: Develop a C# program that showcases method overriding by creating a base class "Animal" with a method "MakeSound" and derived classes like "Cat" and "Dog" that override the "MakeSound" method. THIS ONE DONE
10. Q: Create an abstract class "Shape" with an abstract method "CalculateArea". Implement derived classes like "Circle" and "Rectangle" that provide their own implementations of the "CalculateArea" method. THIS ONE DONE
11. Q: Design a sealed class "Employee" with properties like name and salary. Write a program to demonstrate the restriction of inheriting from a sealed class. THIS ONE DONE
12. Q: Implement an interface "ILogger" with a method "Log" that takes a string parameter. Create a class "FileLogger" that implements the "ILogger" interface and writes the log message to a file. THIS ONE
13. Q: Create a delegate and an event in C# to handle a button click event. Write a program to demonstrate the usage of the delegate and event in a GUI application. THIS ONE
14. Q: Develop a C# program to demonstrate the usage of partial classes by splitting the implementation of a class across multiple files. THIS ONE
15. Q: Write a C# program to create a generic class "Stack<T>" that implements a stack data structure. Test the stack by pushing and popping elements of different data types. THIS ONE DONE
16. Q: Implement a C# program that reads data from a file and writes it to another file using file input/output operations. THIS ONE
17. Q: Write a program that demonstrates exception handling in C# by handling a divide-by-zero exception using a try-catch block. THIS ONE DONE
18. Q: Develop a C# program that uses try-finally and try-catch-finally blocks to handle exceptions and ensure cleanup of resources. THIS ONE
19. Q: Create a custom attribute class "CustomAttribute" and apply it to a method in a C# program. Retrieve and display the attribute information at runtime. THIS ONE
20. Q: Write a C# program that demonstrates the usage of named and positional attribute parameters by applying attributes with different parameter values to a class. THIS ONE

Overview of .NET and ASP.NET frameworks: Understanding the different frameworks available within the .NET ecosystem, including .NET Core, Mono, ASP.NET Web Forms, ASP.NET MVC, ASP.NET Web API, and ASP.NET Core. .NET architecture and design principles: Exploring the architecture and design principles underlying the .NET framework. Compilation and execution of .NET applications: Understanding the Common Language Infrastructure (CLI), Microsoft Intermediate Language (MSIL), and Common Language Runtime (CLR). Introduction to .NET Core: Overview of .NET Core and its features. Working with .NET CLI: Building, running, testing, and deploying .NET Core applications.MAKE THEORY QUESTIONS FROM THIS SECTION

1. Q: What is the .NET framework, and what are its key components?
2. Q: Explain the differences between the different frameworks within the .NET ecosystem, such as .NET Core, Mono, ASP.NET Web Forms, ASP.NET MVC, ASP.NET Web API, and ASP.NET Core.
3. Q: What are the architecture and design principles that underlie the .NET framework?
4. Q: Describe the compilation process of .NET applications and the role of the Common Language Infrastructure (CLI).
5. Q: What is Microsoft Intermediate Language (MSIL), and how does it relate to the execution of .NET applications?
6. Q: Explain the role of the Common Language Runtime (CLR) in executing .NET applications.
7. Q: What is .NET Core, and what are some of its key features and advantages over other frameworks?
8. Q: Describe the concept of the .NET Command-Line Interface (CLI) and its role in building, running, testing, and deploying .NET Core applications.

Q: Create a .NET Core console application that displays "Hello, World!" as output.

Q: Develop an ASP.NET Web Forms application that includes a registration form with validation for name, email, and password fields.

Q: Build an ASP.NET MVC application that allows users to create, read, update, and delete records from a database.

Q: Implement an ASP.NET Web API that provides endpoints for retrieving and manipulating data from a database.

Q: Create an ASP.NET Core application that utilizes authentication and authorization to restrict access to certain pages or API endpoints.

Q: Develop a .NET Core console application that reads data from a CSV file and stores it in a database.

Q: Build an ASP.NET MVC application that integrates with a third-party API, such as a weather API, to display real-time data.

Q: Implement a .NET Core console application that performs file input/output operations, such as copying, moving, or deleting files.

Q: Develop an ASP.NET Web Forms application that utilizes master pages for consistent layout and design across multiple pages.

Q: Create an ASP.NET Core application that uses dependency injection to manage the instantiation and lifecycle of services.

Q: Build an ASP.NET Web API that includes unit tests to ensure the correctness of the API endpoints.

Q: Implement a .NET Core console application that handles exceptions using try-catch blocks and provides appropriate error messages to the user.

Q: Develop an ASP.NET MVC application that utilizes partial views to modularize the user interface and improve code reusability.

Q: Create an ASP.NET Core application that deploys to a cloud hosting platform, such as Azure or AWS.

Q: Build a .NET Core console application that utilizes command-line arguments and options for customizing program behavior.

Q: Implement an ASP.NET Web Forms application that uses AJAX to perform asynchronous operations and update specific parts of the page without refreshing.

Q: Develop an ASP.NET Core application that includes logging functionality to record application events and errors.

Q: Create a .NET Core console application that generates PDF reports from data and saves them to the file system.

Q: Build an ASP.NET MVC application that implements caching mechanisms to improve performance and reduce database queries.

Q: Implement an ASP.NET Web API that supports versioning to allow backward compatibility and API evolution.

* Understanding the HTTP protocol and its request and response message format.
* Common web application architectures and patterns.
* Overview of the Model-View-Controller (MVC) pattern.
* ASP.NET Core architecture overview, including projects and conventions.
* Differences between ASP.NET and ASP.NET MVC.
* Q: What is the HTTP protocol? Explain its role in web communication.
* Q: Describe the structure and format of an HTTP request message.
* Q: Explain the components and format of an HTTP response message.
* Q: Discuss the different web application architectures and their characteristics.
* Q: What is the Model-View-Controller (MVC) pattern? How does it facilitate the development of web applications?
* Q: Describe the components and responsibilities of the Model, View, and Controller in the MVC pattern.
* Q: Explain the advantages of using the MVC pattern in web application development.
* Q: Provide an overview of the architecture of ASP.NET Core and its key components.
* Q: Describe the concept of projects in ASP.NET Core and how they are organized.
* Q: Discuss the conventions followed in ASP.NET Core for file and folder structures, routing, and naming conventions.
* Q: What are the main differences between ASP.NET and ASP.NET MVC?
* Q: Explain the key features and advantages of ASP.NET MVC over traditional ASP.NET Web Forms.
* Q: Describe the role of the Model, View, and Controller in ASP.NET MVC.
* Q: Discuss how ASP.NET Core enhances the performance and scalability of web applications compared to previous versions of ASP.NET.
* Q: Explain how routing works in ASP.NET Core and how it maps incoming requests to the appropriate controller actions.
* Q: Build a C# console application that makes an HTTP GET request to a specified URL and displays the response content.
* Q: Develop an ASP.NET Core web application that handles form submissions and displays a success message to the user.
* Q: Implement a C# program that parses an HTTP response message and extracts specific information, such as headers or body content.
* Q: Build an ASP.NET Core MVC application that allows users to create, read, update, and delete records in a database using the Entity Framework.
* Q: Create a C# program that consumes a public API, such as a weather API, and displays the retrieved data in a user-friendly format.
* Q: Develop an ASP.NET Core web application that uses a custom middleware to log each incoming request and response.
* Q: Implement a C# program that performs file uploads to a web server using the HTTP protocol.
* Q: Build an ASP.NET Core MVC application that uses dependency injection to inject services into controller actions.
* Q: Create a C# program that simulates a RESTful API by handling different HTTP methods (GET, POST, PUT, DELETE) for a specific resource.
* Q: Develop an ASP.NET Core web application that utilizes authentication and authorization to restrict access to certain pages or API endpoints.
* Q: Implement a C# program that performs HTTP redirects by sending appropriate response headers.
* Q: Build an ASP.NET Core MVC application that utilizes view models to separate the presentation logic from the data models.
* Q: Create a C# program that generates PDF documents dynamically based on user input and serves them as HTTP responses.
* Q: Develop an ASP.NET Core web application that implements caching mechanisms to improve performance and reduce database queries.
* Q: Implement a C# program that interacts with a RESTful API by sending HTTP requests and processing the responses.
* Q: Build an ASP.NET Core MVC application that utilizes partial views to modularize the user interface and improve code reusability.
* Q: Create a C# program that consumes a web service using SOAP (Simple Object Access Protocol) and retrieves data from it.
* Q: Develop an ASP.NET Core web application that uses Razor Pages to handle user requests and display dynamic content.
* Q: Implement a C# program that performs server-side validation of form inputs and provides appropriate error messages to the user.
* Q: Build an ASP.NET Core MVC application that integrates with a third-party API, such as a payment gateway, to process online payments.