Dot Net Syllabus

The **.NET MVC (Model-View-Controller)** framework is a popular design pattern used in web application development. It is a part of the **ASP.NET** framework and follows the MVC architecture to separate concerns and improve code maintainability.

.NET MVC Syllabus

Module 1: Introduction to .NET and MVC

- **Overview of .NET Framework**
  - History and evolution of .NET.
  - Understanding .NET Core vs. .NET Framework.
  - Key components: Common Language Runtime (CLR), Base Class Library (BCL).
- **Introduction to MVC Architecture**
  - Overview of MVC design pattern.
  - Benefits of MVC: Separation of concerns, testability, maintainability.
  - Understanding the role of Model, View, and Controller.
- **Setting up .NET MVC Development Environment**
  - Installing Visual Studio or Visual Studio Code.
  - Creating a simple MVC project in Visual Studio.
  - Overview of the solution structure in an MVC project.

Module 2: ASP.NET MVC Basics

- **Understanding MVC Components**
  - **Model**: Represents the data and business logic.
  - **View**: User interface (UI) rendering.
  - **Controller**: Handles user input and updates the model and view.
- **Routing in ASP.NET MVC**
  - URL patterns and how they map to controllers and actions.
  - Configuring routing in `RouteConfig.cs`.
  - Understanding default routes and custom routes.
- **Creating Controllers**
  - Syntax for creating controllers.
  - Action methods and action results.
  - Returning views from controllers.
  - Controller lifecycle.
- **Creating Views**
  - Using Razor syntax for views.
  - Understanding **@Model** and **@Html** helpers.
  - Creating and rendering views from controllers.
  - View types: strongly typed and dynamically typed views.

Module 3: Working with Models

- **Introduction to Models in MVC**
  - Models as classes that define the data structure.
  - Creating a model class in MVC.
  - Connecting models with databases using **Entity Framework**.
- **Model Binding**
  - Passing data from views to controllers using form submission.
  - Binding form data to model properties.
  - Validating models with data annotations.

- **Data Annotations**
  - Introduction to data validation using attributes (e.g., `Required`, `Range`, `StringLength`).
  - Custom validation in models.
- **Entity Framework**
  - Introduction to Entity Framework (EF) and its importance.
  - CRUD operations with EF (Create, Read, Update, Delete).
  - Code-first and database-first approaches.
  - Setting up database context and connecting to a database.

Module 4: Views, Layouts, and Partial Views

- **Creating Views in MVC**
  - Types of views: Razor views, Partial Views, and Shared Views.
  - Using layout files to create consistent page structures.
  - Understanding and using **_Layout.cshtml**.
- **Partial Views**
  - Creating reusable components with Partial Views.
  - Rendering Partial Views in parent views.
  - Advantages of Partial Views for performance and maintainability.
- **Form Handling in Views**
  - Creating forms for CRUD operations.
  - Binding form data to models.
  - Using **@Html.BeginForm**, **@Html.EditorFor**, and other HTML helpers.

Module 5: Controllers and Actions

- **Controller Fundamentals**
  - Controller actions and action result types: `ActionResult`, `ViewResult`, `RedirectResult`, etc.
  - Action filters: Authorization, logging, custom filters.
  - Handling requests and routing to appropriate actions.
- **Action Parameters**
  - Passing data to action methods.
  - Using query strings, form data, and route data.
  - Accepting parameters in controller methods.
- **Redirecting and Returning Data**
  - Redirecting from one action to another.
  - Returning JSON, RedirectToAction, or RedirectToRoute.

Module 6: Validation and Error Handling

- **Model Validation**
  - Using **Data Annotations** for validation.
  - Validating models in the controller.
  - Client-side validation using jQuery and unobtrusive validation.
- **Error Handling in ASP.NET MVC**
  - Global error handling using `Application_Error` and `Error.cshtml`.
  - Custom error pages (404, 500, etc.).

- Using `try-catch` blocks in controllers.
- Logging errors using logging frameworks like NLog or Serilog.

Module 7: Authentication and Authorization

- **Authentication Basics**
  - Implementing user authentication using forms authentication.
  - Integrating **ASP.NET Identity** for authentication and user management.
  - Implementing login, registration, and logout features.
- **Authorization Basics**
  - Role-based authorization and access control.
  - Attribute-based authorization: `[Authorize]`, `[AllowAnonymous]`.
  - Custom authorization filters.

Module 8: Web API in ASP.NET MVC

- **Introduction to Web API**
  - Difference between Web API and MVC.
  - Creating RESTful services using ASP.NET Web API.
  - HTTP methods: GET, POST, PUT, DELETE.
- **Creating a Web API Controller**
  - Setting up API routes and controllers.
  - Returning data as JSON or XML.
  - Understanding and using `HttpResponseMessage`.
- **Consuming Web API in MVC**
  - Calling Web API services from an MVC controller.
  - Using **HttpClient** to interact with external APIs.
  - Handling API responses.

Module 9: Advanced Topics

- **Filters in MVC**
  - Action Filters: `OnActionExecuting`, `OnActionExecuted`.
  - Result Filters: `OnResultExecuting`, `OnResultExecuted`.
  - Custom filters for logging, security, etc.
- **Dependency Injection in ASP.NET MVC**
  - Introduction to Dependency Injection (DI).
  - Using **Unity** or **Ninject** for DI in MVC applications.
- **Unit Testing MVC Applications**
  - Writing unit tests for controllers and services.
  - Using **MSTest** or **NUnit** for testing.
  - Mocking dependencies with **Moq**.

Module 10: Deployment and Best Practices

- **Deploying ASP.NET MVC Applications**
  - Deploying to IIS (Internet Information Services).
  - Using **Azure App Services** for cloud-based deployment.

    - Configuration management and connection strings.
- **Performance Optimization**
    - Caching in MVC: Output caching, data caching, and application caching.
    - Minification and bundling of scripts and styles.
- **Security Best Practices**
    - Securing web applications: HTTPS, SQL injection prevention, Cross-Site Scripting (XSS), and Cross-Site Request Forgery (CSRF).
    - Configuring and using **AntiForgeryToken**.

Tools and Technologies Covered in the Course:

- **Visual Studio** or **Visual Studio Code**: Integrated development environment (IDE).
- **Entity Framework**: ORM for database interaction.
- **SQL Server** or any relational database.
- **HTML, CSS, and JavaScript**: Frontend technologies.
- **jQuery and Bootstrap**: For UI development.
- **ASP.NET Identity**: For authentication and authorization.
- **Azure**: For cloud deployment.

Projects and Case Studies:

- **End-to-End MVC Application Project**
    - Build a complete application that involves CRUD operations, user authentication, and data display.
- **Capstone Project**
    - Develop a real-world project, such as an e-commerce website, a blog platform, or a task management system.

Learning Outcomes:

By the end of this course, students will be able to:

- Develop dynamic, data-driven web applications using ASP.NET MVC.
- Understand and implement the MVC architecture.
- Work with databases using Entity Framework.
- Implement authentication, authorization, and security in ASP.NET MVC applications.
- Deploy MVC applications to cloud services like Azure or on-premises IIS servers.