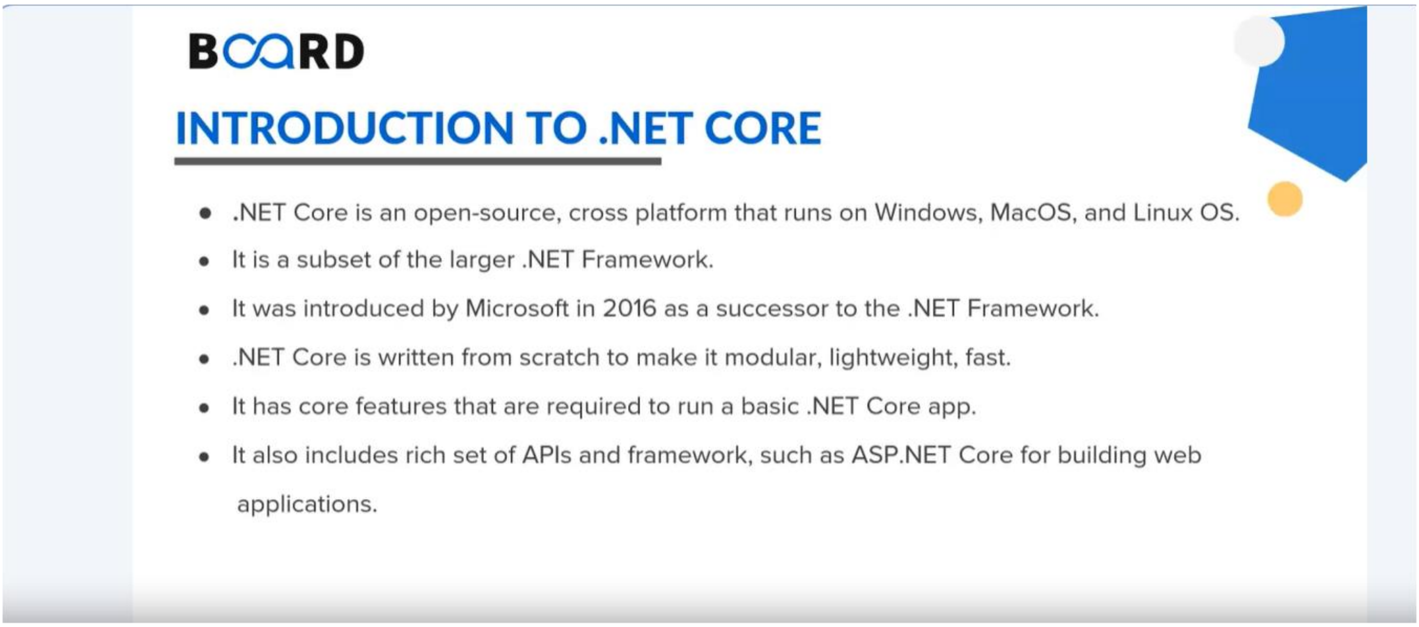
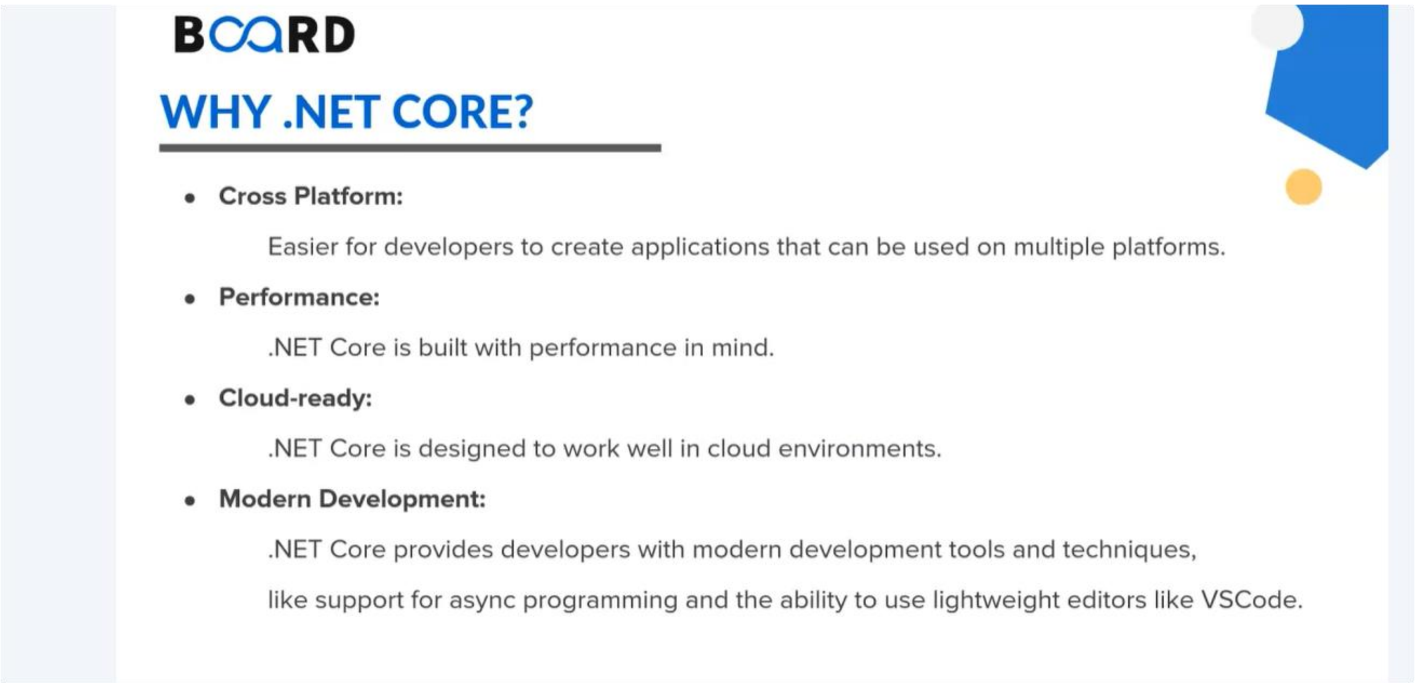


BOARD

INTRODUCTION TO .NET CORE

- .NET Core is an open-source, cross platform that runs on Windows, MacOS, and Linux OS.
- It is a subset of the larger .NET Framework.
- It was introduced by Microsoft in 2016 as a successor to the .NET Framework.
- .NET Core is written from scratch to make it modular, lightweight, fast.
- It has core features that are required to run a basic .NET Core app.
- It also includes rich set of APIs and framework, such as ASP.NET Core for building web applications.



BOARD

WHY .NET CORE?

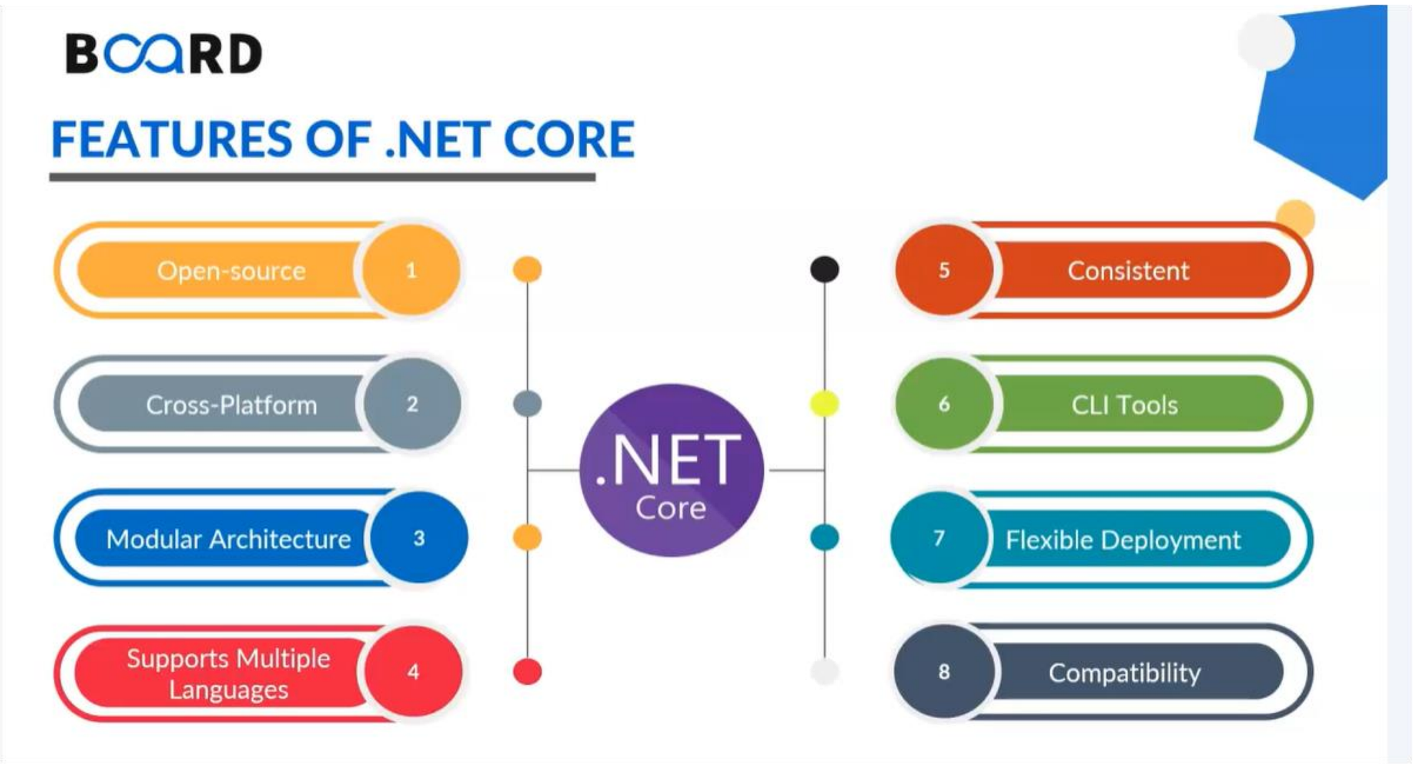
- **Cross Platform:**

  Easier for developers to create applications that can be used on multiple platforms.
- **Performance:**

  .NET Core is built with performance in mind.
- **Cloud-ready:**

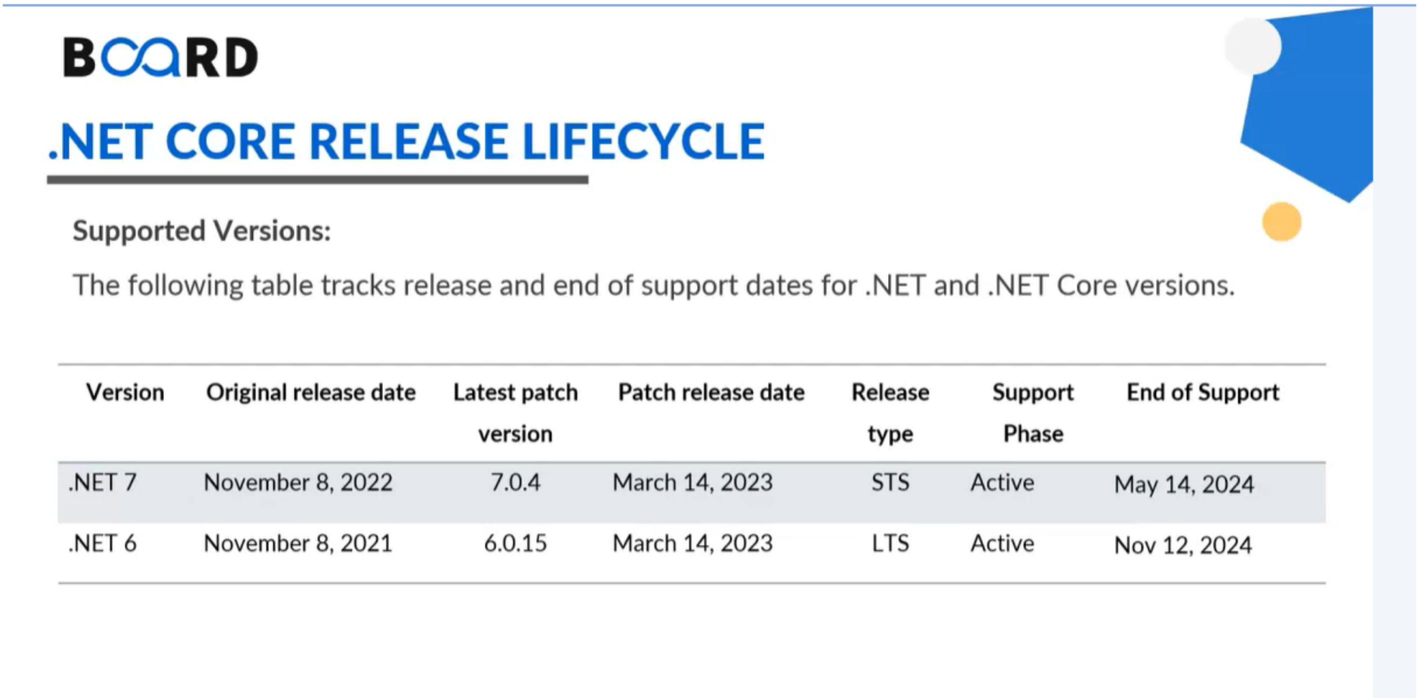
  .NET Core is designed to work well in cloud environments.
- **Modern Development:**

  .NET Core provides developers with modern development tools and techniques, like support for async programming and the ability to use lightweight editors like VSCode.



BOARD

FEATURES OF .NET CORE



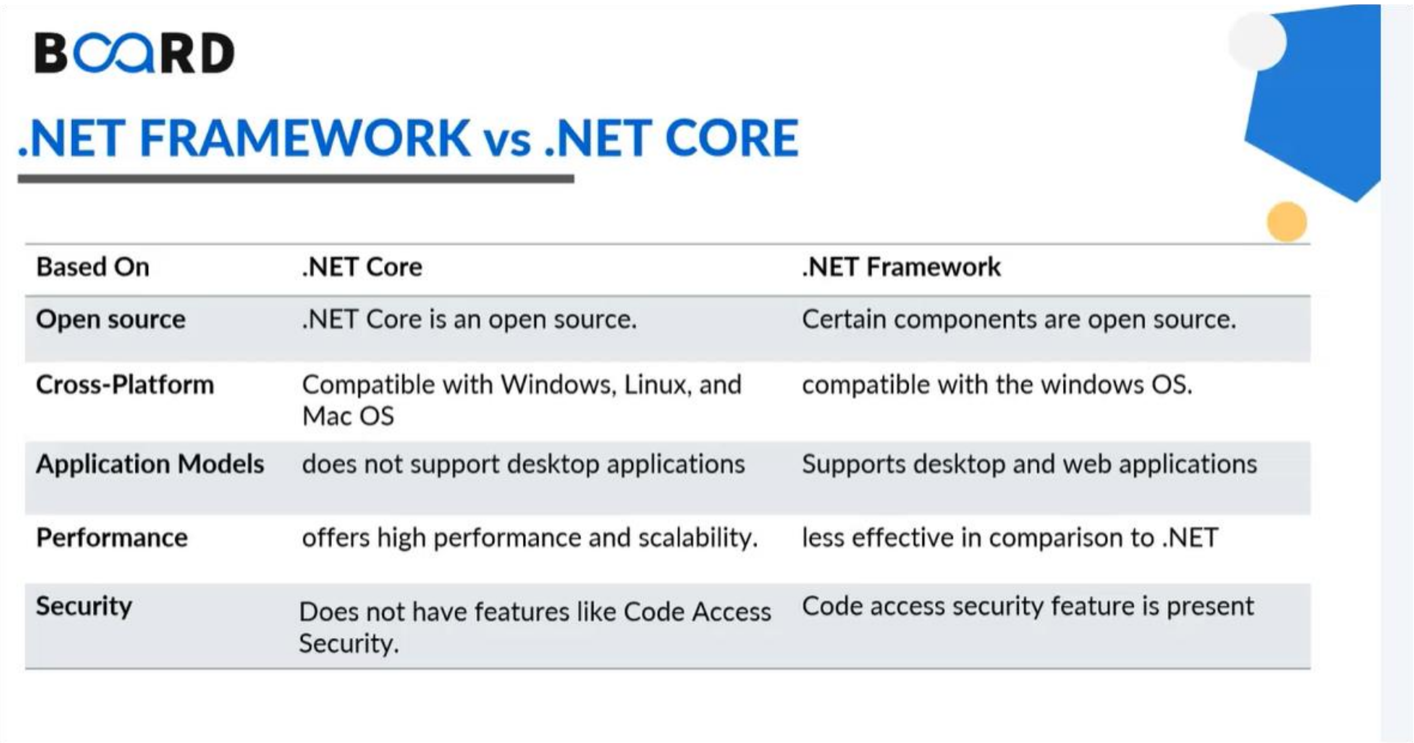
BOARD

.NET CORE RELEASE LIFECYCLE

Supported Versions:

The following table tracks release and end of support dates for .NET and .NET Core versions.

Version	Original release date	Latest patch version	Patch release date	Release type	Support Phase	End of Support
.NET 7	November 8, 2022	7.0.4	March 14, 2023	STS	Active	May 14, 2024
.NET 6	November 8, 2021	6.0.15	March 14, 2023	LTS	Active	Nov 12, 2024

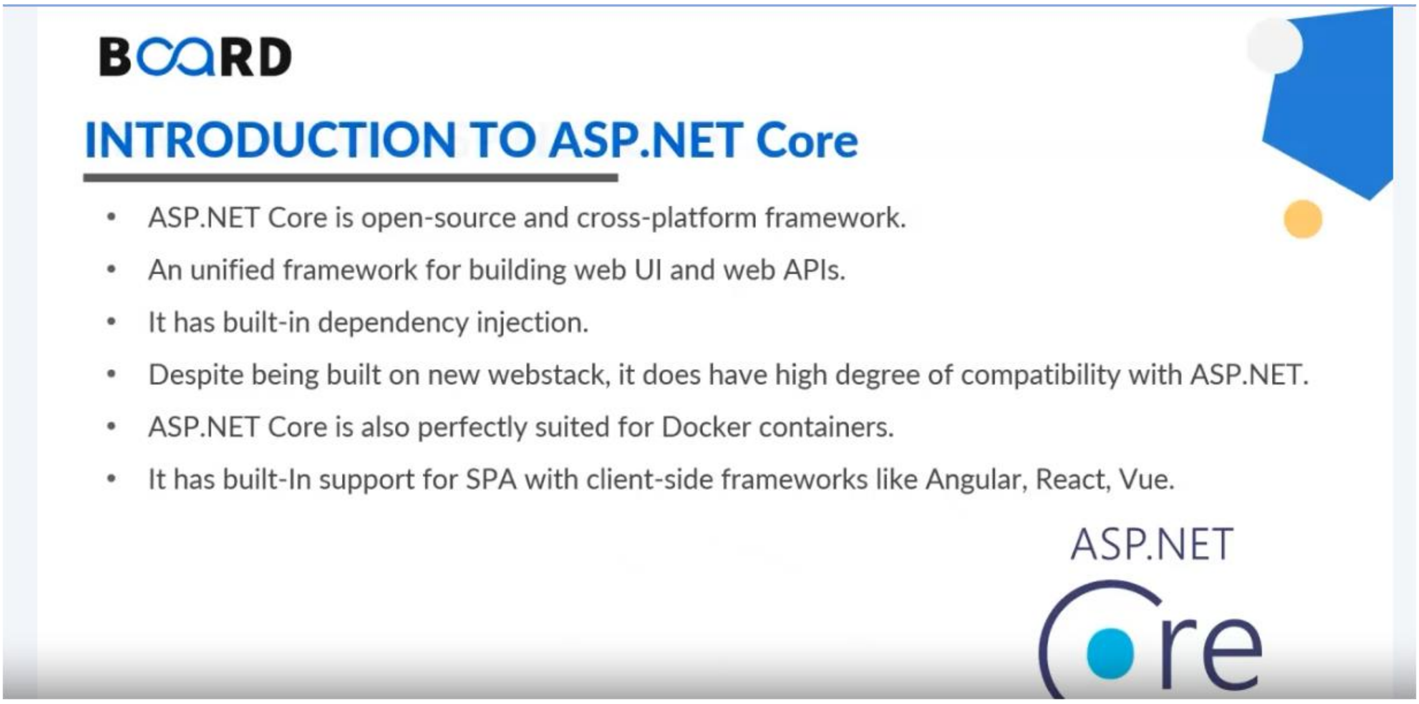


BOARD

.NET FRAMEWORK vs .NET CORE

Based On	.NET Core	.NET Framework
Open source	.NET Core is an open source.	Certain components are open source.
Cross-Platform	Compatible with Windows, Linux, and Mac OS	compatible with the windows OS.
Application Models	does not support desktop applications	Supports desktop and web applications
Performance	offers high performance and scalability.	less effective in comparison to .NET
Security	Does not have features like Code Access Security.	Code access security feature is present

Introduction to ASP.NET Core

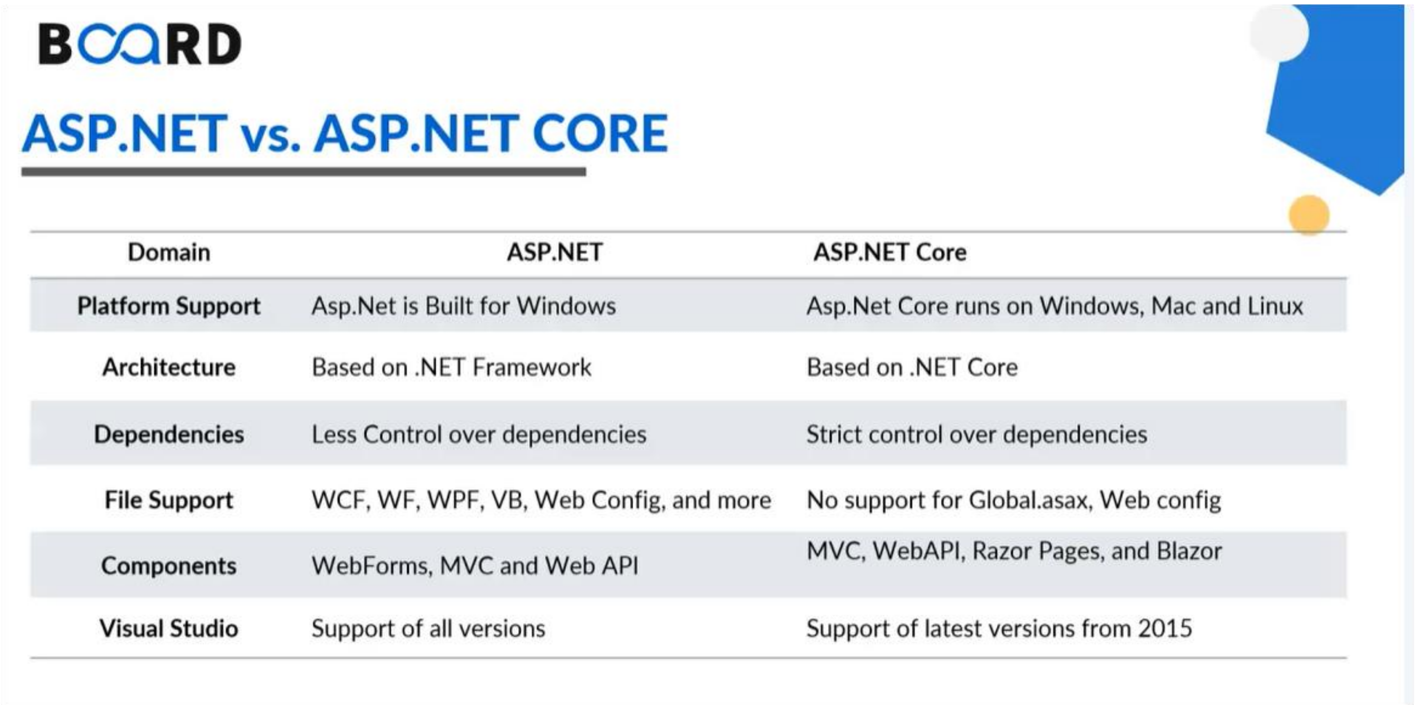


BOARD

INTRODUCTION TO ASP.NET Core

- ASP.NET Core is open-source and cross-platform framework.
- An unified framework for building web UI and web APIs.
- It has built-in dependency injection.
- Despite being built on new webstack, it does have high degree of compatibility with ASP.NET.
- ASP.NET Core is also perfectly suited for Docker containers.
- It has built-In support for SPA with client-side frameworks like Angular, React, Vue.

ASP .NET vs ASP .NET Core Applications



BOARD

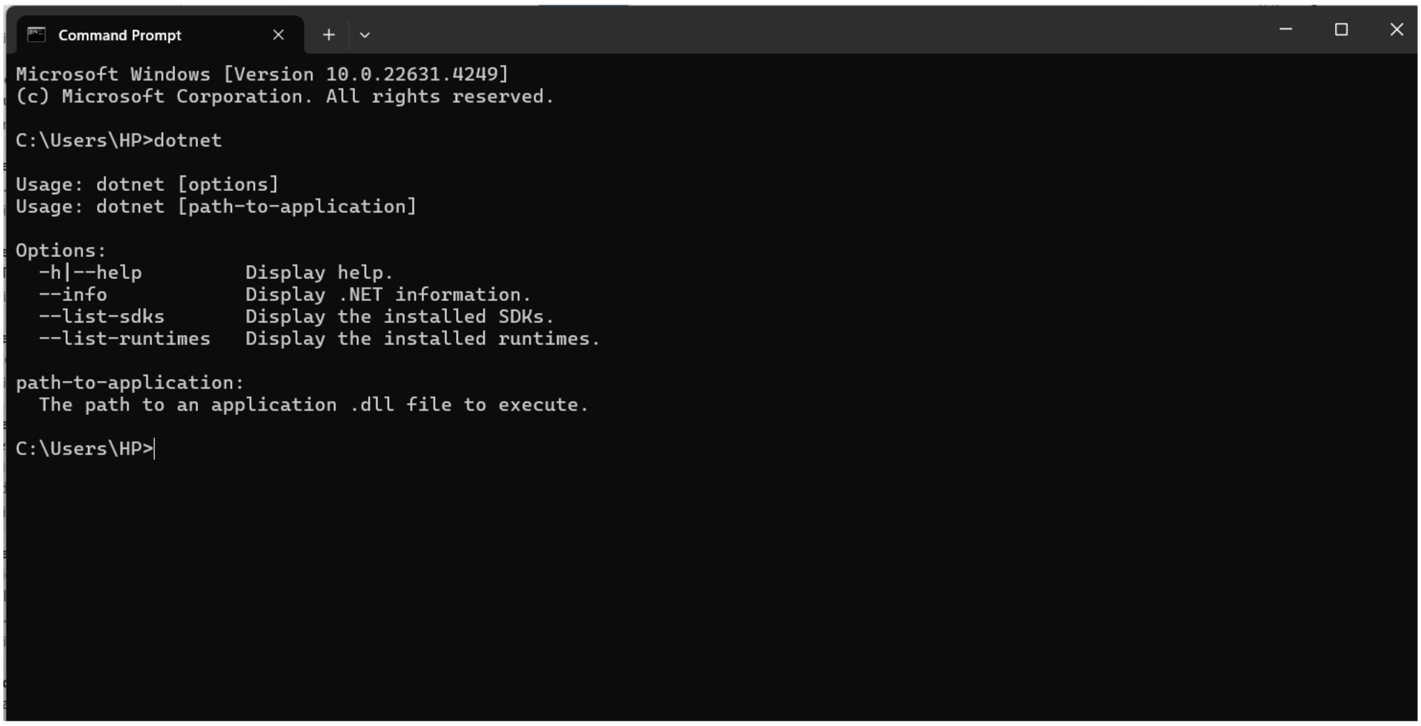
ASP.NET vs. ASP.NET CORE

Domain	ASP.NET	ASP.NET Core
Platform Support	Asp.Net is Built for Windows	Asp.Net Core runs on Windows, Mac and Linux
Architecture	Based on .NET Framework	Based on .NET Core
Dependencies	Less Control over dependencies	Strict control over dependencies
File Support	WCF, WF, WPF, VB, Web Config, and more	No support for Global.asax, Web config
Components	WebForms, MVC and Web API	MVC, WebAPI, Razor Pages, and Blazor
Visual Studio	Support of all versions	Support of latest versions from 2015

ASP.NET Core Project Folder Structure

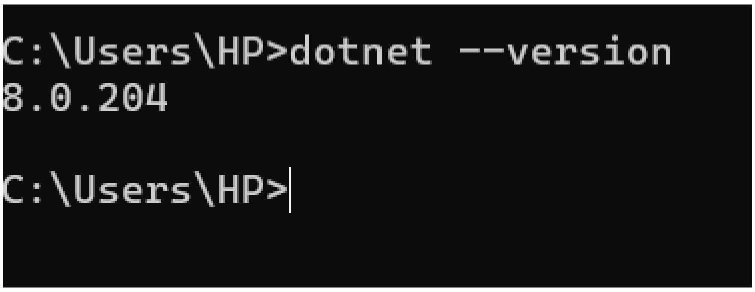
Demo: Setting up a development environment with Visual Studio and/or VS Code

How to check if .net is installed -



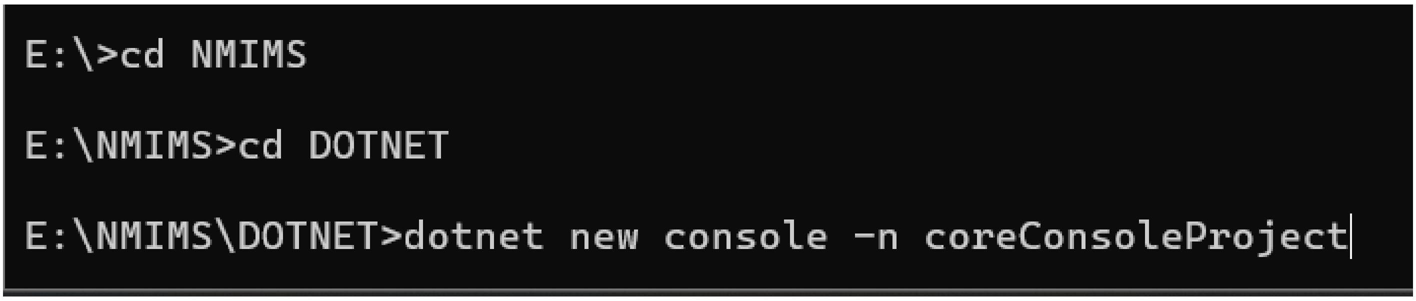
```
Microsoft Windows [Version 10.0.22631.4249]
(c) Microsoft Corporation. All rights reserved.

C:\Users\HP>dotnet

Usage: dotnet [options]
Usage: dotnet [path-to-application]

Options:
  -h|--help         Display help.
  --info            Display .NET information.
  --list-sdks       Display the installed SDKs.
  --list-runtimes   Display the installed runtimes.

path-to-application:
  The path to an application .dll file to execute.

C:\Users\HP>
```

How to check version:



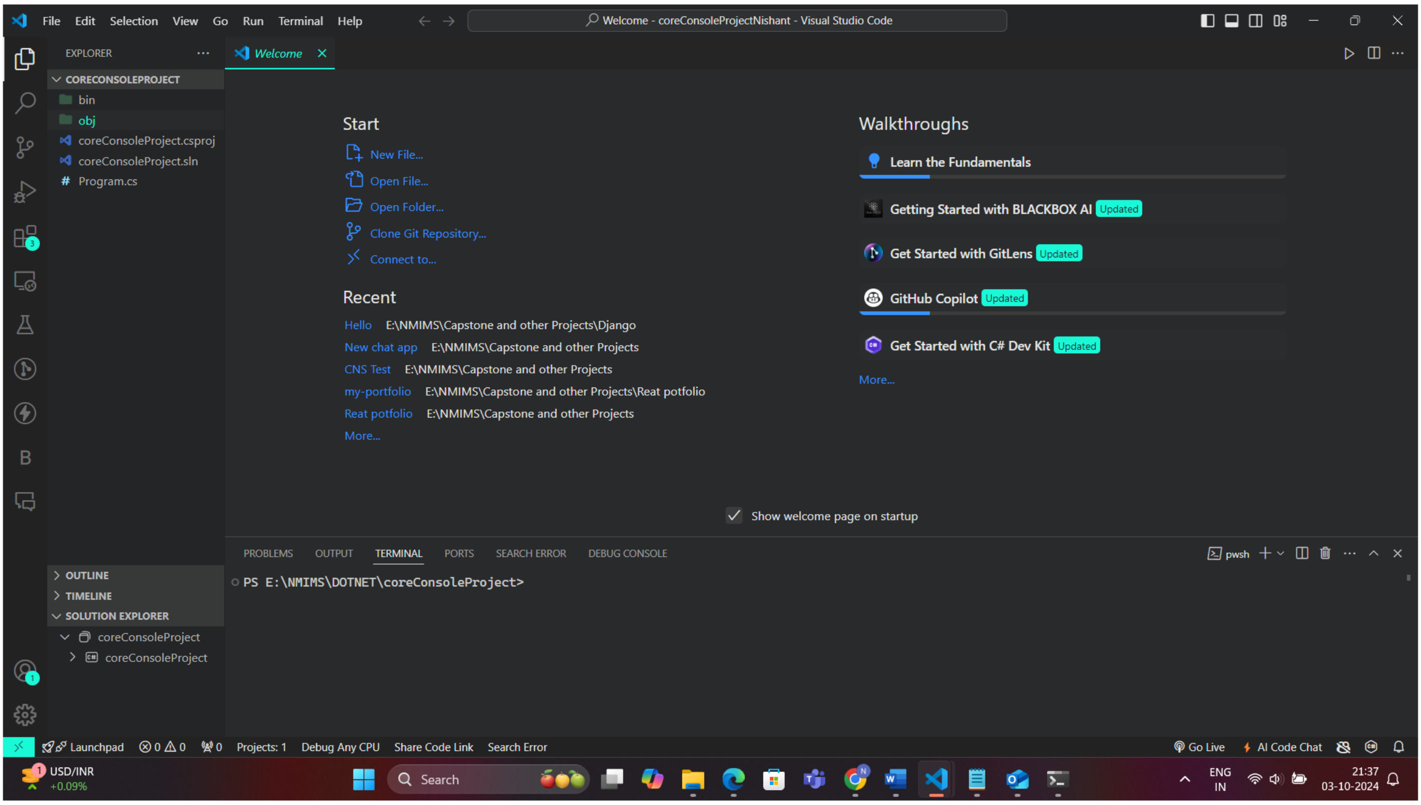
```
C:\Users\HP>dotnet --version
8.0.204

C:\Users\HP>
```

How to start a new project :



```
E:\>cd NMIMS

E:\NMIMS>cd DOTNET

E:\NMIMS\DOTNET>dotnet new console -n coreConsoleProject
```

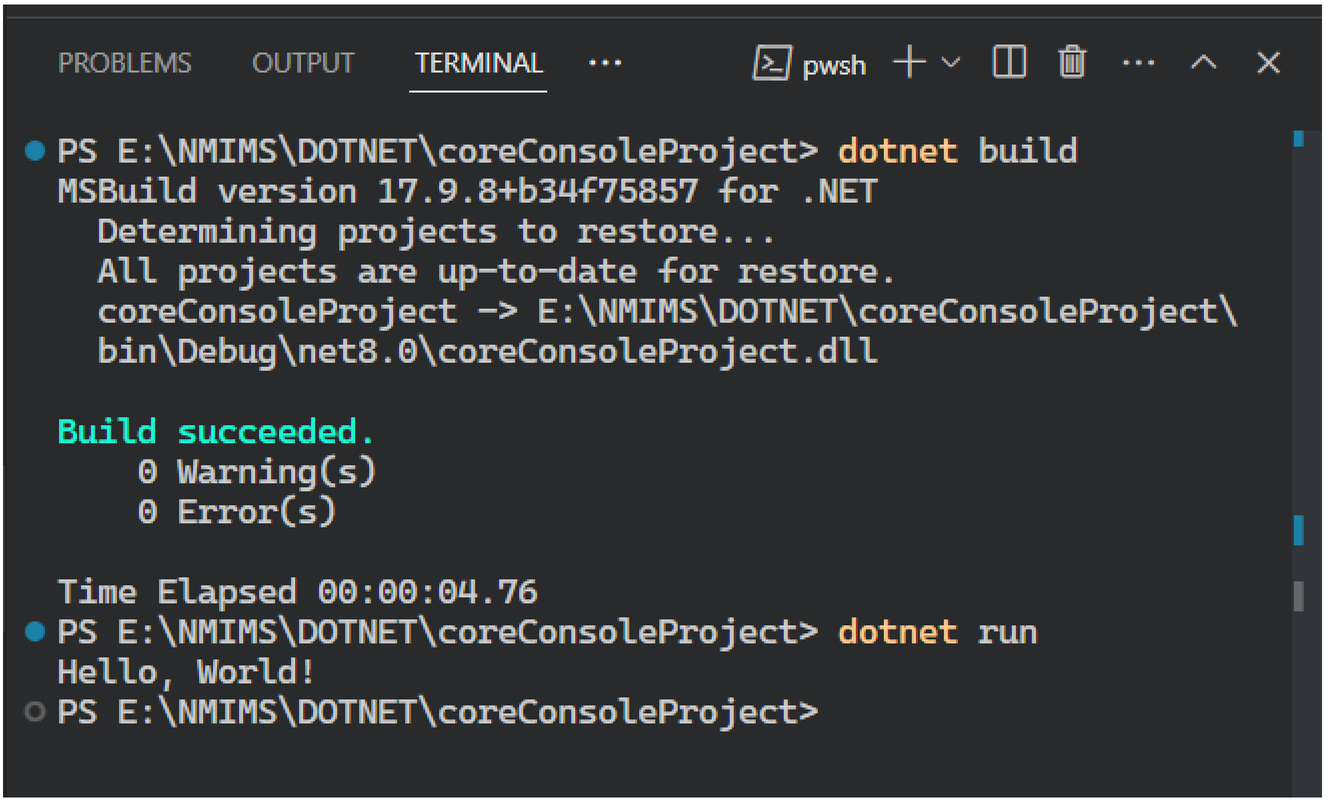
Command to open the current directory in VS code:



File Edit Selection View Go Run Terminal Help
Welcome - coreConsoleProjectNishant - Visual Studio Code
EXPLORER
CORECONSOLEPROJECT
bin
obj
coreConsoleProject.csproj
coreConsoleProject.sln
Program.cs
Welcome
Start
New File...
Open File...
Open Folder...
Clone Git Repository...
Connect to...
Recent
Hello E:\NMIMS\Capstone and other Projects\Django
New chat app E:\NMIMS\Capstone and other Projects
CNS Test E:\NMIMS\Capstone and other Projects
my-portfolio E:\NMIMS\Capstone and other Projects\Reat potfolio
Reat potfolio E:\NMIMS\Capstone and other Projects
More...
Walkthroughs
Learn the Fundamentals
Getting Started with BLACKBOX AI Updated
Get Started with GitLens Updated
GitHub Copilot Updated
Get Started with C# Dev Kit Updated
More...
Show welcome page on startup
OUTLINE
TIMELINE
SOLUTION EXPLORER
coreConsoleProject
coreConsoleProject
PROBLEMS OUTPUT TERMINAL PORTS SEARCH ERROR DEBUG CONSOLE
pwsh
Launchpad ⊗ 0 ⚠ 0 0 Projects: 1 Debug Any CPU Share Code Link Search Error
Go Live AI Code Chat
USD/INR +0.09%
Search
ENG IN
21:37 03-10-2024

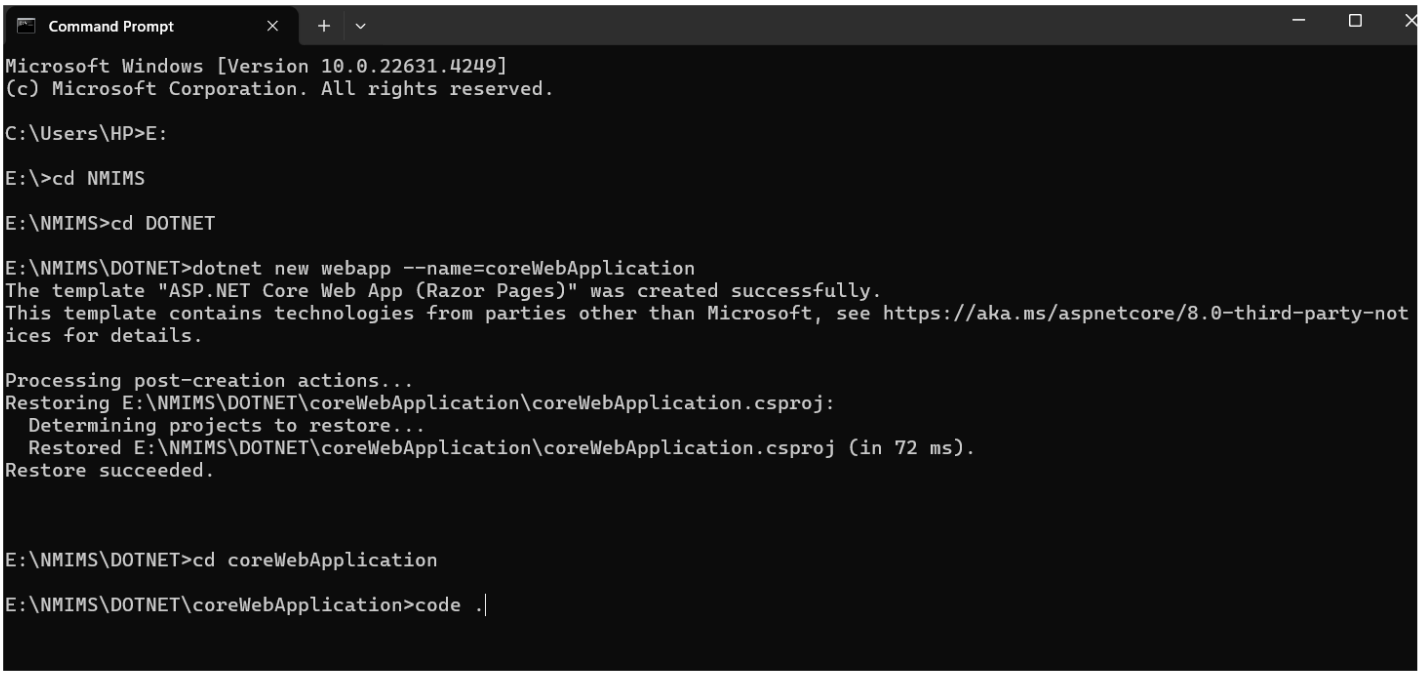
```
PS E:\NMIMS\DOTNET\coreConsoleProject>
```

How to build the project and run it:



```
PS E:\NMIMS\DOTNET\coreConsoleProject> dotnet build
MSBuild version 17.9.8+b34f75857 for .NET
  Determining projects to restore...
  All projects are up-to-date for restore.
  coreConsoleProject -> E:\NMIMS\DOTNET\coreConsoleProject\
  bin\Debug\net8.0\coreConsoleProject.dll

Build succeeded.
    0 Warning(s)
    0 Error(s)

Time Elapsed 00:00:04.76
PS E:\NMIMS\DOTNET\coreConsoleProject> dotnet run
Hello, World!
PS E:\NMIMS\DOTNET\coreConsoleProject>
```

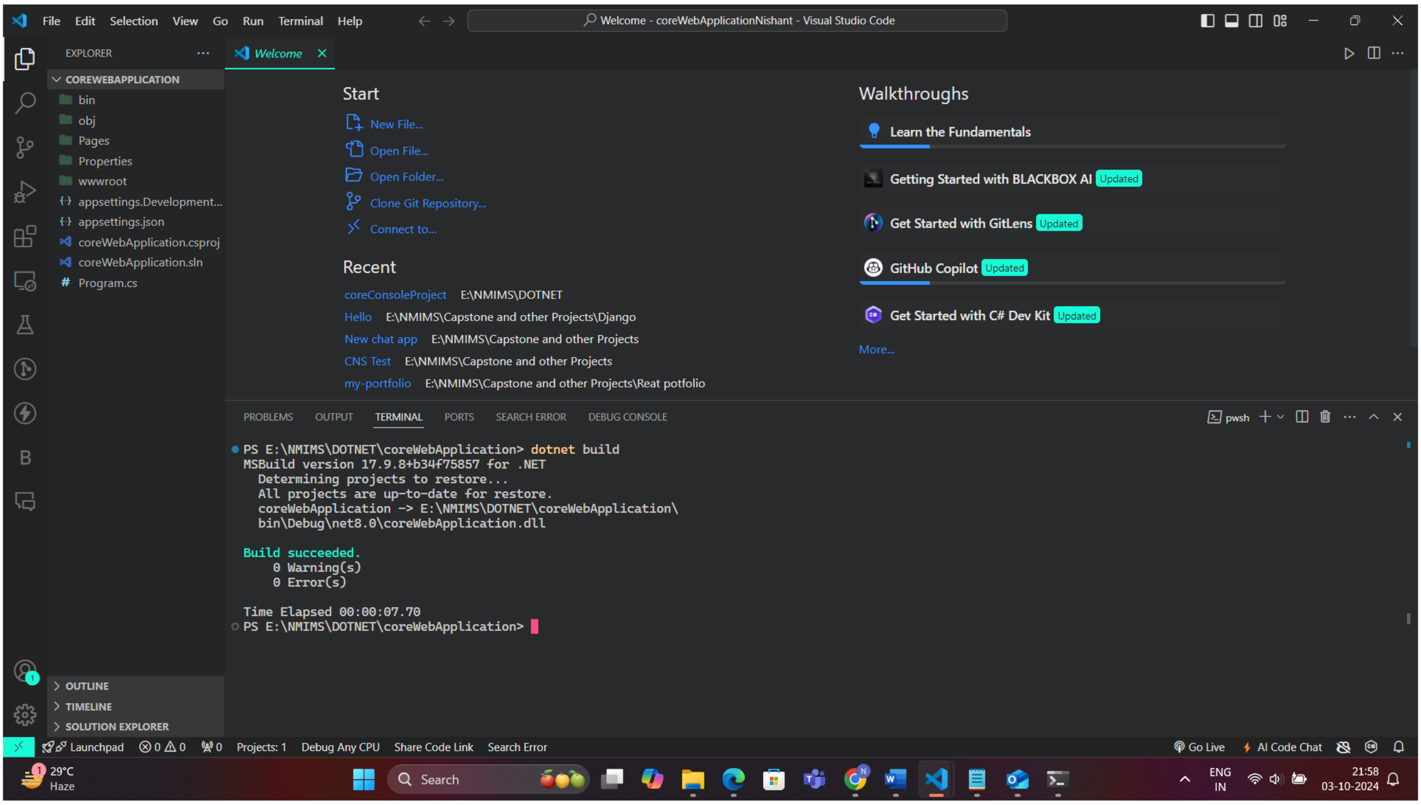
Demo: Building a basic .NET Core web application with ASP.NET Core



```
Microsoft Windows [Version 10.0.22631.4249]
(c) Microsoft Corporation. All rights reserved.

C:\Users\HP>E:

E:\>cd NMIMS

E:\NMIMS>cd DOTNET

E:\NMIMS\DOTNET>dotnet new webapp --name=coreWebApplication
The template "ASP.NET Core Web App (Razor Pages)" was created successfully.
This template contains technologies from parties other than Microsoft, see https://aka.ms/aspnetcore/8.0-third-party-not
ices for details.

Processing post-creation actions...
Restoring E:\NMIMS\DOTNET\coreWebApplication\coreWebApplication.csproj:
  Determining projects to restore...
  Restored E:\NMIMS\DOTNET\coreWebApplication\coreWebApplication.csproj (in 72 ms).
Restore succeeded.


E:\NMIMS\DOTNET>cd coreWebApplication

E:\NMIMS\DOTNET\coreWebApplication>code .
```

In the above the command to create the web app project can be found from the list command of dotnet.

The web app project gets open below and also we have build the web app:


File Edit Selection View Go Run Terminal Help
Welcome - coreWebApplicationNishant - Visual Studio Code
EXPLORER
COREWEBAPPLICATION
bin
obj
Pages
Properties
wwwroot
appsettings.Development...
appsettings.json
coreWebApplication.csproj
coreWebApplication.sln
Program.cs
OUTLINE
TIMELINE
SOLUTION EXPLORER
Welcome
Start
New File...
Open File...
Open Folder...
Clone Git Repository...
Connect to...
Recent
Walkthroughs
Learn the Fundamentals
Getting Started with BLACKBOX AI Updated
Get Started with GitLens Updated
GitHub Copilot Updated
Get Started with C# Dev Kit Updated
More...
PROBLEMS OUTPUT TERMINAL PORTS SEARCH ERROR DEBUG CONSOLE
pwsh
Launchpad 0 0 0 Projects: 1 Debug Any CPU Share Code Link Search Error
Go Live AI Code Chat
29°C Haze
Search
ENG IN
21:58
03-10-2024

Recent

coreConsoleProject E:\NMIMS\DOTNET
Hello E:\NMIMS\Capstone and other Projects\Django
New chat app E:\NMIMS\Capstone and other Projects
CNS Test E:\NMIMS\Capstone and other Projects
my-portfolio E:\NMIMS\Capstone and other Projects\Reat potfolio

```
PS E:\NMIMS\DOTNET\coreWebApplication> dotnet build
MSBuild version 17.9.8+b34f75857 for .NET
  Determining projects to restore...
  All projects are up-to-date for restore.
  coreWebApplication -> E:\NMIMS\DOTNET\coreWebApplication\
  bin\Debug\net8.0\coreWebApplication.dll

Build succeeded.
    0 Warning(s)
    0 Error(s)

Time Elapsed 00:00:07.70
PS E:\NMIMS\DOTNET\coreWebApplication>
```

1. .NET CLI and its Features

.NET CLI (Command-Line Interface) is a cross-platform tool used to develop, build, run, and publish .NET applications. It provides commands for managing .NET projects directly from the command line without the need for an IDE (Integrated Development Environment). Some of its key features include:

- **Cross-platform support**: You can run .NET CLI on Windows, macOS, and Linux.
- **Project management**: Commands like dotnet new (for creating projects) and dotnet restore (for restoring dependencies) help manage .NET projects effortlessly.
- **Build and run**: You can build your application using dotnet build and run it with dotnet run.
- **Package management**: You can add, remove, and update NuGet packages with dotnet add package or dotnet remove package.
- **Testing support**: Run unit tests with dotnet test to ensure your application is functioning as expected.
- **Publishing**: You can publish your project to prepare it for deployment using the dotnet publish command.
- **Tool extensibility**: CLI supports external tools and templates that extend the functionality of the base commands, making it customizable for different development environments.

2. What is .NET CLI?

.NET CLI is a command-line interface that allows developers to create, build, run, and deploy .NET applications without the need for a full-fledged IDE. It simplifies project creation, builds automation, package management, and even deployment. It supports multiple types of .NET applications, including:

- **ASP.NET Core** for web applications and APIs.
- **Console applications** for command-line tools.
- **Class libraries** for reusable code.
- **WPF and WinForms** for desktop apps (on Windows).

.NET CLI works with the SDK (Software Development Kit) that contains tools, libraries, and templates for building applications. You simply run commands in the terminal or command prompt, and the CLI takes care of the backend processes, such as compiling the code or publishing an application.

3. Advantages of .NET CLI

The .NET CLI offers several advantages that make it a go-to tool for .NET developers:

- **Cross-platform development**: You can use it across Windows, macOS, and Linux, giving you flexibility in your development environment.
- **Simplifies workflows**: With just a few commands, you can create, build, run, test, and publish projects, streamlining the development process.
- **No IDE required**: While Visual Studio and other IDEs are powerful, .NET CLI allows developers to work without them, making it lightweight and ideal for scripting or automation.
- **Automation**: CLI commands can be incorporated into build scripts, continuous integration pipelines, and other automated processes, boosting productivity and reducing manual effort.
- **Customizable**: Developers can install third-party tools and templates to extend CLI functionality, making it adaptable for specific workflows or project types.
- **Version control integration**: .NET CLI can easily be integrated with version control systems like Git to create repeatable and shareable builds.

4. How it Simplifies the Development and Deployment Process

.NET CLI simplifies the development and deployment process in several key ways:

- **Quick project setup**: You can initialize a new project with a single command (dotnet new), making the setup time very fast, especially for smaller or repetitive projects.
- **Automated dependency management**: Commands like dotnet restore automatically pull in all necessary NuGet packages, ensuring the project has the required libraries to run.
- **Streamlined builds**: The dotnet build command compiles the code and packages it for you, ensuring that errors are caught early.
- **Testing made simple**: With the dotnet test command, you can quickly run unit tests and validate code changes, ensuring that code quality remains high.
- **Ease of publishing**: The dotnet publish command bundles all files, settings, and dependencies into a format ready for deployment. This simplifies the process of deploying the application to a server or a cloud environment.
- **Consistent environment**: Because the CLI is consistent across operating systems, the development and deployment processes remain the same, regardless of where you're working. This eliminates issues related to platform-specific tools.
- **Integration with CI/CD**: The CLI can easily be integrated into continuous integration/continuous deployment (CI/CD) pipelines, making automated testing, builds, and deployments effortless.

In summary, .NET CLI is a powerful tool that makes developing, building, and deploying .NET applications straightforward, flexible, and efficient. It removes the complexity of setup and configuration and focuses on simplifying everyday tasks for developers.