📄 Report: .NET Versions, Namespace, .NET Core, and Solution

1️⃣ .NET Versions

● **Summary:**

.NET evolved through three major phases:

- **.NET Framework** (Windows-only, legacy)
- **.NET Core** (cross-platform, open-source)
- **.NET 5 and beyond** (modern, unified platform — now just called **“.NET”**)

● **Key Milestones:**

Version	Highlights
.NET Framework	Legacy, Windows-only, mature but no cross-platform
.NET Core 1-3	Lightweight, cross-platform, CLI support, modular
.NET 5	First unified version (Core + Framework)
.NET 6 (LTS)	Strong performance, simplified dev experience
.NET 7	Performance-focused, non-LTS
.NET 8 (LTS)	Latest stable release (at time of writing), future-oriented

Note: LTS = Long-Term Support (typically supported for 3 years)

2️⃣ Namespace

● **Summary:**

A **namespace** in .NET is a way to organize code and avoid naming conflicts.

● **Key Concepts:**

- Acts like folders for classes, interfaces, enums, etc.
- Helps identify where a class belongs logically (System.IO, Microsoft.AspNetCore, etc.)
- Allows **code reuse** and **modularity**

● **Example:**

```
namespace MyApp.Services
{
   public class UserService { ... }
}
```

● **Common Namespaces:**

- System → Core functionality
- System.Collections.Generic → Data structures like List, Dictionary
- Microsoft.AspNetCore.Mvc → Web APIs and MVC apps

---

3 .NET Core

● **Summary:**

.NET Core was Microsoft's modern, modular, open-source framework.

● **Key Features:**

- **Cross-platform**: Runs on Windows, Linux, macOS
- **Open-source**: Active development on GitHub
- **Performance**: Faster than classic .NET Framework
- **Side-by-side execution**: Multiple versions on the same machine
- **Command-Line Tools**: Build, run, test via dotnet CLI

● **Current Status:**

.NET Core is now **merged** into the main **.NET** platform from version 5 onward.

4️⃣ Solution

● **Summary:**

A **Solution (.sln)** in .NET is a container that holds one or more **Projects**.

● **Purpose:**

- Helps manage large apps split into multiple components (like APIs, libraries, UI)
- Used by IDEs (especially Visual Studio) to organize and build projects together

● **Example Structure:**

```
MySolution/
├── WebApp/       (.csproj)
├── DataAccess/    (.csproj)
├── SharedLibrary/  (.csproj)
├── MySolution.sln
```

● **Key Benefits:**

- Simplifies team collaboration
- Centralized build configuration
- Easy project references and dependencies