**Day 1 – .NET Basics**

**1. What is .NET?**

* A **free, open-source, cross-platform** developer platform.
* Used to build **web, desktop, mobile, cloud, gaming, IoT** applications.

**2. .NET Architecture**

* **Common Language Runtime (CLR)** – Executes code, handles memory, exceptions, GC.
* **.NET Class Library (BCL)** – Pre-built classes for IO, collections, security, etc.
* **Languages Supported** – C#, F#, VB.NET.
* **App Models** – ASP.NET (web), WPF/WinForms (desktop), Blazor, Xamarin (.NET MAUI).

**3. Compilation Flow**

* Code (.cs) → **Compiler** → **Intermediate Language (IL)** → **CLR** → Native Machine Code via JIT (Just-In-Time compiler).

**4. Common Language Runtime (CLR)**

* Core execution engine for .NET.
* Responsibilities:
  + Memory management
  + Type safety
  + Exception handling
  + Thread management
  + Garbage collection

**5. Just-In-Time (JIT) Compiler**

* Converts IL to native machine code at runtime.
* Optimizes performance.
* Variants: Pre-JIT, Econo-JIT, Normal JIT.

**6. Base Class Library (BCL)**

* A collection of reusable classes.
* Examples: System.IO, System.Net, System.Collections, System.Text.

**7. Key Terms**

* **Managed Code**: Code executed by CLR.
* **Unmanaged Code**: Runs outside CLR (e.g., C/C++ DLLs).
* **Assemblies**: Compiled code in .exe or .dll.
* **Namespace**: Logical grouping of classes.

**8. .NET Versions**

* .NET Framework – Windows-only (legacy)
* .NET Core – Cross-platform
* .NET 5/6/7/8+ – Unified platform, open-source, fast

**9. Advantages of .NET**

* Language interoperability
* Cross-platform development
* Automatic memory management (GC)
* Strong security model
* Large ecosystem

**10. Tools**

* IDE: Visual Studio / VS Code
* CLI: dotnet CLI (dotnet new, dotnet run, dotnet build, dotnet publish)