

	Morning : (90 mins) Getting started with Git basics
Introduction to Version Control	<p>What is version control and why it is essential?</p> <p>Centralized vs. Distributed version control systems.</p> <p>Git Basics, Branching and Merging working with Remote Repositories</p>
Implementing CI/CD	<p>What is CI/CD and why is it important?</p> <ul style="list-style-type: none"> • Continuous integration / Continuous Development • Faster feedback • Reduce Bugs • Automation • Devops Culture <p>Key concepts: Build, Test, Release, Deploy. Below is a CI/CD lifecycle. Code -> Build -> Test->Release->Deploy</p> <p>Tools for CI/CD: Jenkins -> Enterprise Code Integration GitHub Actions -> Repo native CI/CD GitLab CI, -> tools for CI Azure Devops -> Microsoft ecosystem</p>
	Afternoon (90 mins)Introduction & Basic Programming Concepts
Getting started with .NET platform	<p>Installation of Visual Studio 2019</p> <p>Visual Studio 2019 (IDE)</p> <ol style="list-style-type: none"> 1. Full features IDE 2. Debugger, intellisense,designer support 3. best for enterprise scale projects <p>VS Code (light weight IDE)</p> <ul style="list-style-type: none"> • Light weight & faster development • Crossplatform only • Requires extensions for C#.NET SDK) <p>Web based tools (One Complier)</p> <p>Introduction to .NET framework</p> <p>What is .NET ? key features of it</p> <ul style="list-style-type: none"> • initially NGWS was later renamed to .(DOT) Next generation Technology • Free, open source, Crossplatform developer platform for building application

	<ul style="list-style-type: none"> ○ Web application (Front End & BackEnd) ○ Desktop application (Windows app) ○ Cloud application ○ Mobile & IOT applications <p>Features :</p> <ol style="list-style-type: none"> 1. language INteroperability (C#, F# , VB.NET) 25+ language 2. Automatic memory management 3. Rich base Library 4. Cross platform execution (create once reuse on other OS) <p>C -> C with Classes _> Objective C-> C++ →</p> <p>C ++</p> <p>++ # is used for Higher nodes</p> <p>What is C# (Sharp) ? - Anders Heilsberg -It is a language used in .NET ecosystem. -Modern, Object-oriented, Strongly typed, Programming language developed by Microsoft. -Primarily it is used for creating : Secure , scalable, high performance application : (Enterprise grade) across for web, windows, desktop, cloud, mobile etc.</p> <p>Current version of C# https://learn.microsoft.com/en-us/dotnet/csharp/whats-new/csharp-14</p> <p>.NET Framework .NET Core .NET 6</p>
	.NET framework architecture
	Common type system
	Common language specification
	Common language runtime(IMp)
	Base class library
	Why Object is the ultimate base class for all types ?
	Introduction to Classes
	Reading and Writing in the console
	Assembly (dll/exe)

Types in C#

Private
Shared
Types
Value type and Reference type
Boxing and Unboxing
Strings : immutable
String Builder : mutable
Difference between String and StringBuilder
Nullable types (defined using ?)

Every Day

Evening Practice : 03:30 -05:30 : 120 mins

MCQ assessment : 25-30 mins

Coding Assessment : Platform Code -Eval /Coding

last 15 mins Summarization for the daily topics

Milestone 1,2,3,4 with passing % is 80

Final Milestone

Capstone project (5Days) Multiple Sprints 1 & 2(100 marks)

Why ?

1. Real markers for your skillset
2. Understanding user stories and implementation of solution
3. It will help in getting your first project.

List of softwares that we need to install to get started :

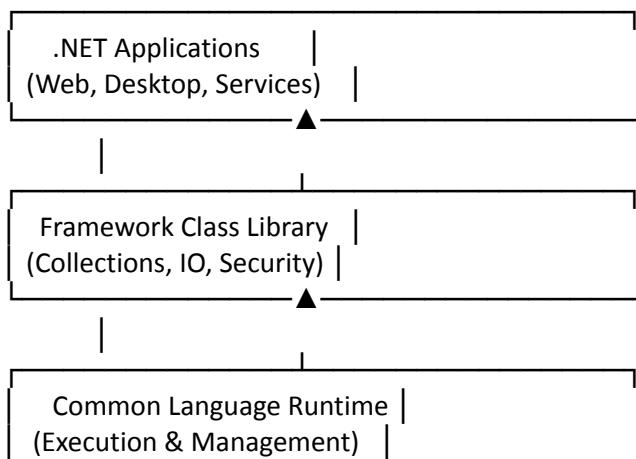
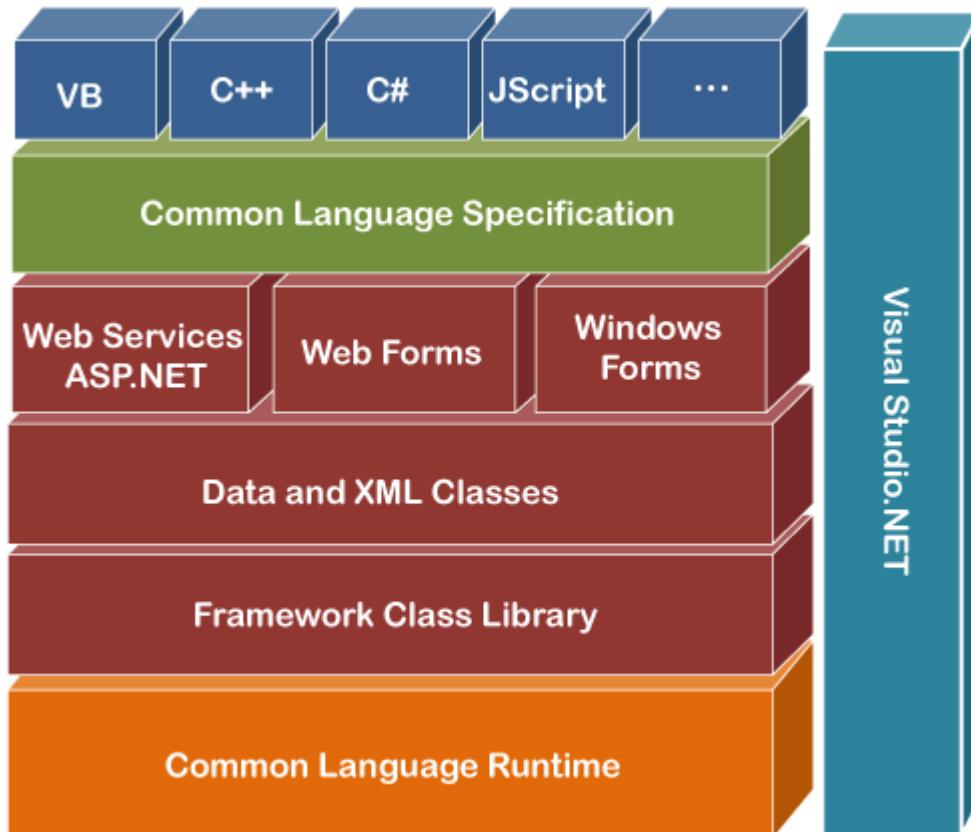
1. **VS code**
2. **For Azure**

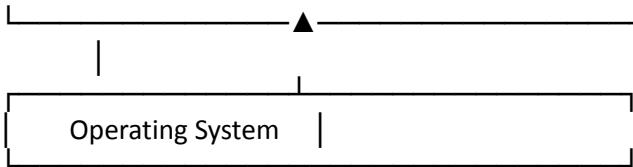
Where C# is used ?

Application Type	Examples
Web Applications & APIs	ASP.NET Core web apps, REST APIs
Enterprise Systems	ERP, CRM, Banking, Insurance platforms
Desktop Applications	Windows Forms, WPF applications
Cloud & Microservices	Azure Functions, microservices
Mobile Applications	Xamarin / .NET MAUI
Game Development	Unity game engine

IoT & Embedded Systems	Device control and telemetry
Data & Backend Services	Data processing, business services

.NET framework architecture





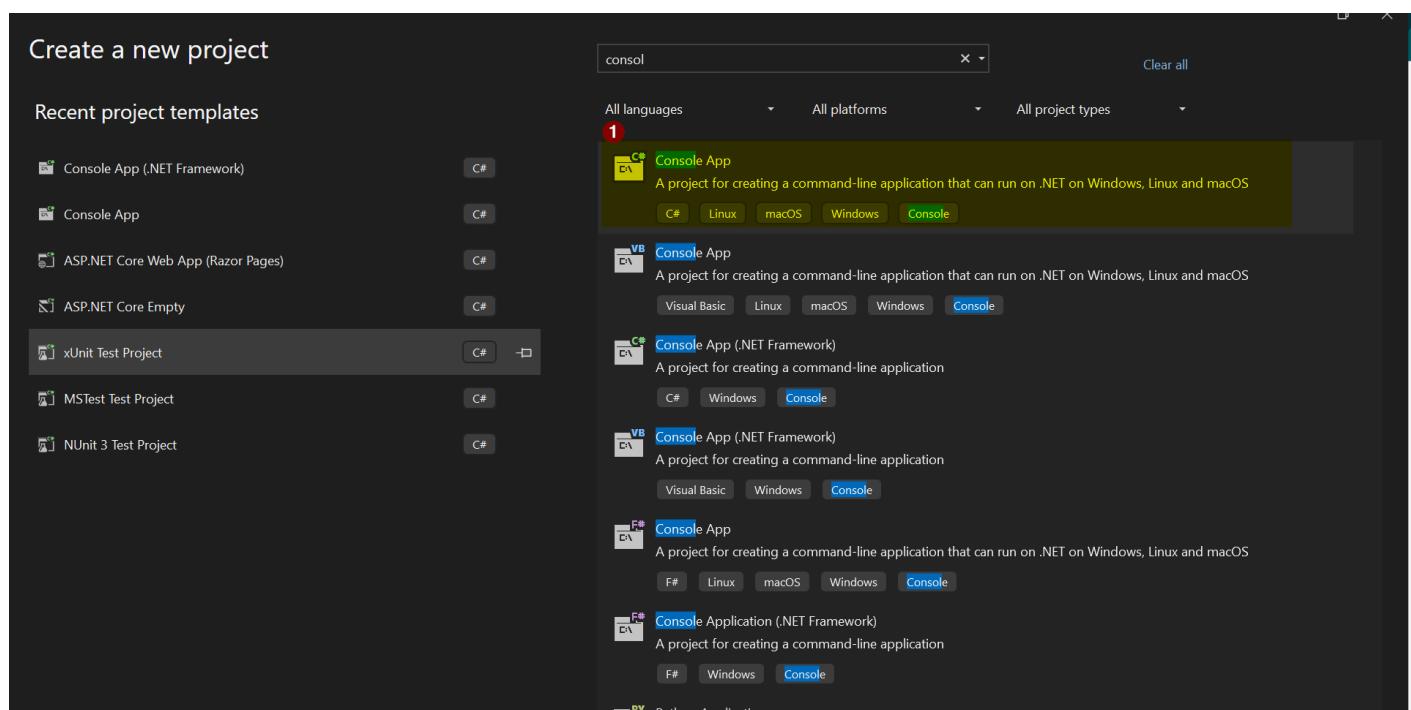
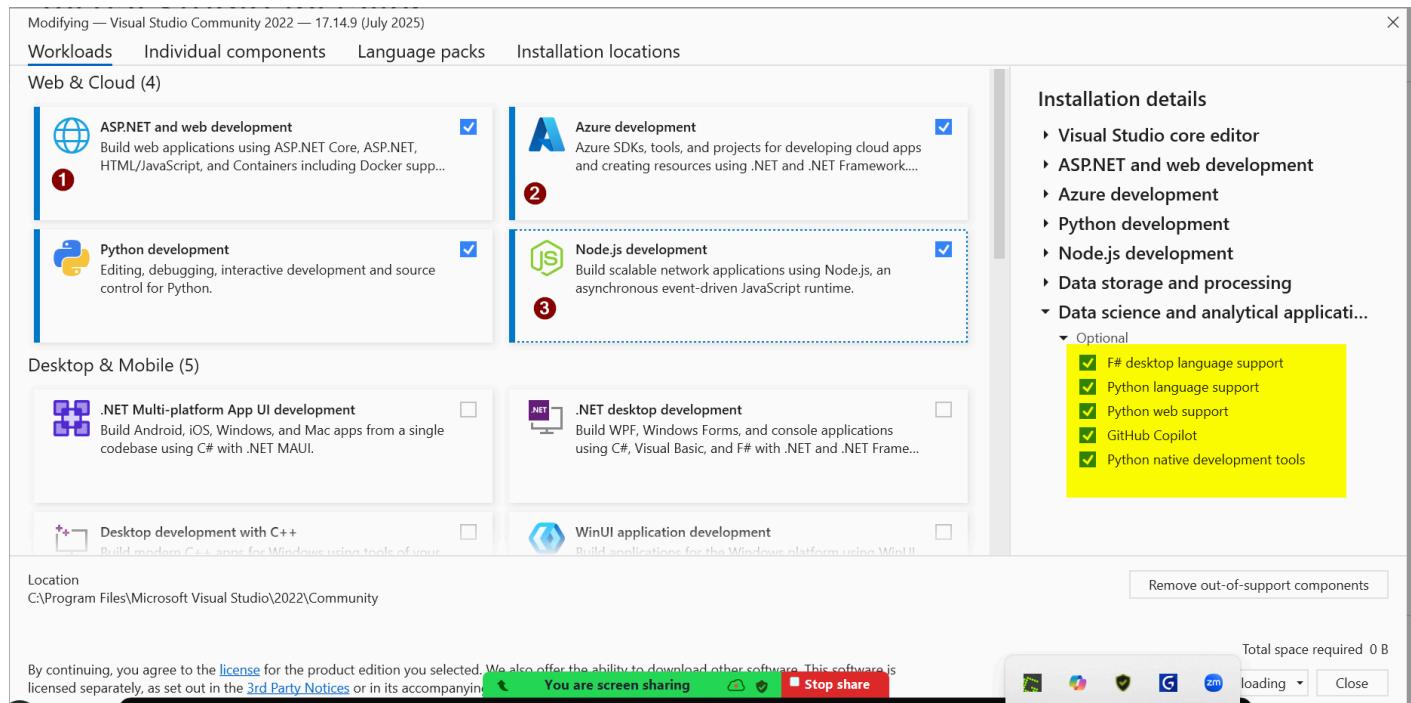
Stages of C# Code compilation :

Step 1: C# Source Code (.cs)

Step 2: C# compiler (csc.exe) -> IL + meta data → Assembly (.exe / .dll) -> CLR loads assembly

Step 3: JIT compiler (just in time)-> Native machine code → Execution on CPU → output

Aspect	.NET Framework	.NET Core	.NET 6+
Platform	Windows OS only	Cross-platform(linux/mac)	Cross-platform
Performance	Moderate	High	Very high
Open Source	✗	✓	✓
Future Support	Legacy	Merged	Current & LTS
CLI support	Limited	Stronger	Full support
Deployment	machine wide install	Side by side	side by side
Cloud support	Weak	Strong	native a& optimized
Web framework	ASP.NET	ASP.NET Core	ASP.NET Core



The screenshot shows a Microsoft Visual Studio interface. On the left, the code editor displays `Program.cs` with the following code:

```

24
25     1 Console.WriteLine("Please enter your age."); // Prompt the user for
26     2 string ageInput = Console.ReadLine(); // Read user input as a string
27     3 int age; // Variable to store the converted age
28     4 bool isValidAge = int.TryParse(ageInput, out age); // Try to convert
29     5 Console.WriteLine(""); // Blank line for better readability
30
31     5 if (isValidAge) // Check if the conversion was successful
32     {
33         6     if (age >= 18) // Check voting eligibility
34         {
35             6     Console.WriteLine("You are eligible to vote.");
36         }
37         else
38         {
39             6     Console.WriteLine("You are not eligible to vote.");
40         }
41     }
42     else
43     {
44         6     Console.WriteLine("Invalid input. Please enter a valid age.");
45     }
46
47
48

```

The output window on the right shows the application's run results:

```

Hello, World!
Please enter your age:
18
6
You are eligible to vote.

C:\Users\Parth\source\repos\Wipro MS_Dynamics_8thJan26\Day 1 C# Basics\Day_1_DEmo2_ReadingInput_User\bin\Debug\net8.0\Day_1_DEmo2_ReadingInput_User.exe (process 17800) exited with code 0 (0x0).
Press any key to close this window . .

```

A callout bubble from line 5 of the code points to the output window, with the text "Here we are converting entered value into number".

//Boxing is the process of converting a value type to an object type.

//Unboxing is the process of converting an object type back to a value type.

//Value type: Store value directly in memory (e.g., int, float, char).

//reference type: Store a reference (address) to the value in memory (e.g., string, arrays, class objects).

//Ex Storing money in wallet compared to storing money in bank account.

//in terms of access speed Value types are faster than reference types.

//IN terms of space efficiency Value types are more space-efficient than reference types.

//Value types are based on Stack memory(fixed size) while reference types are based on Heap memory(unlimited).

//Boxing and unboxing can introduce performance overhead due to additional memory allocation and type conversion.

//ideally we should use less of boxing and unboxing in performance-critical applications.

//In C# Object type is the base type from which all other types derive. ex all types are derived from object type.

//types in C# :

//1. Value types: int, float, double, char, bool, struct, enum

//2. Reference types: string, arrays, class, interface, delegate(function pointer)

Steps for pushing your code on github repo: via Visual studio :

Step 1: Create or opening Project in Visual Studio

Step 2: Sign in into Github from Visual Studio

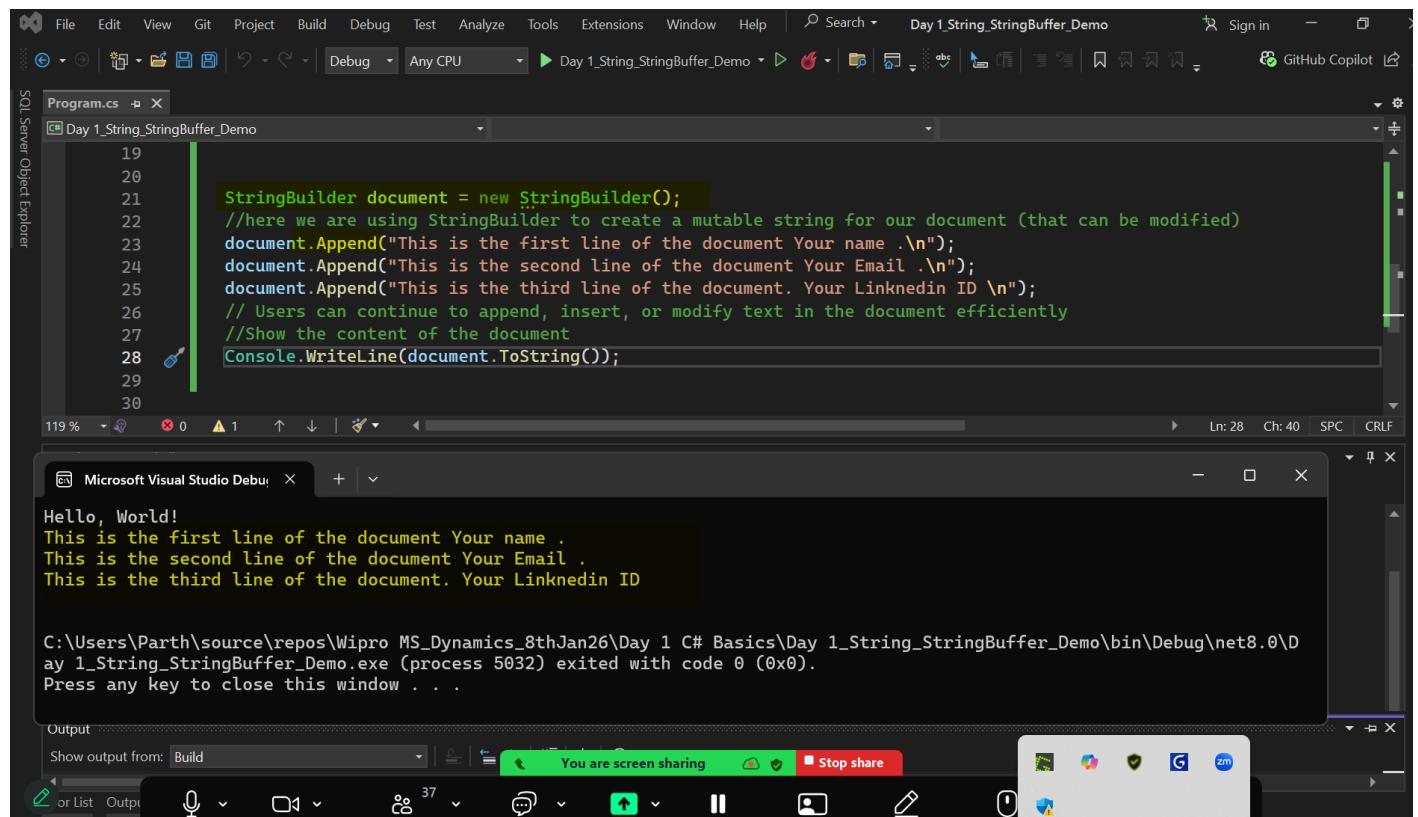
Step 3: Initialize git repo using git menu -> create git repository

- Choose GitHub as the remote
- Provide repository name
- Choose Public or Private
- Click Create and Push

Step 4: review git changes and stage files finally commit changes so that code can be pushed to github

Step 5: verify changes on github

Step No.	Action	Command / Description
1	Open project in VS Code	File → Open Folder → Open project
2	Open terminal	Terminal → New Terminal
3	Initialize Git repository	<code>git init</code>
4	Check repository status	<code>git status</code>
5	Add files to staging	<code>git add .</code>
6	Commit changes	<code>git commit -m "Initial commit"</code>
7	Create GitHub repository	Create repo on GitHub (no README)
8	Add remote origin	<code>git remote add origin https://github.com/<username>/<repo>.git</code>
9	Verify remote	<code>git remote -v</code>
10	Rename branch to main	<code>git branch -M main</code>
11	Push code to GitHub	<code>git push -u origin main</code>
12	Authenticate (if prompted)	Login via browser
13	Verify on GitHub	Check files & commits

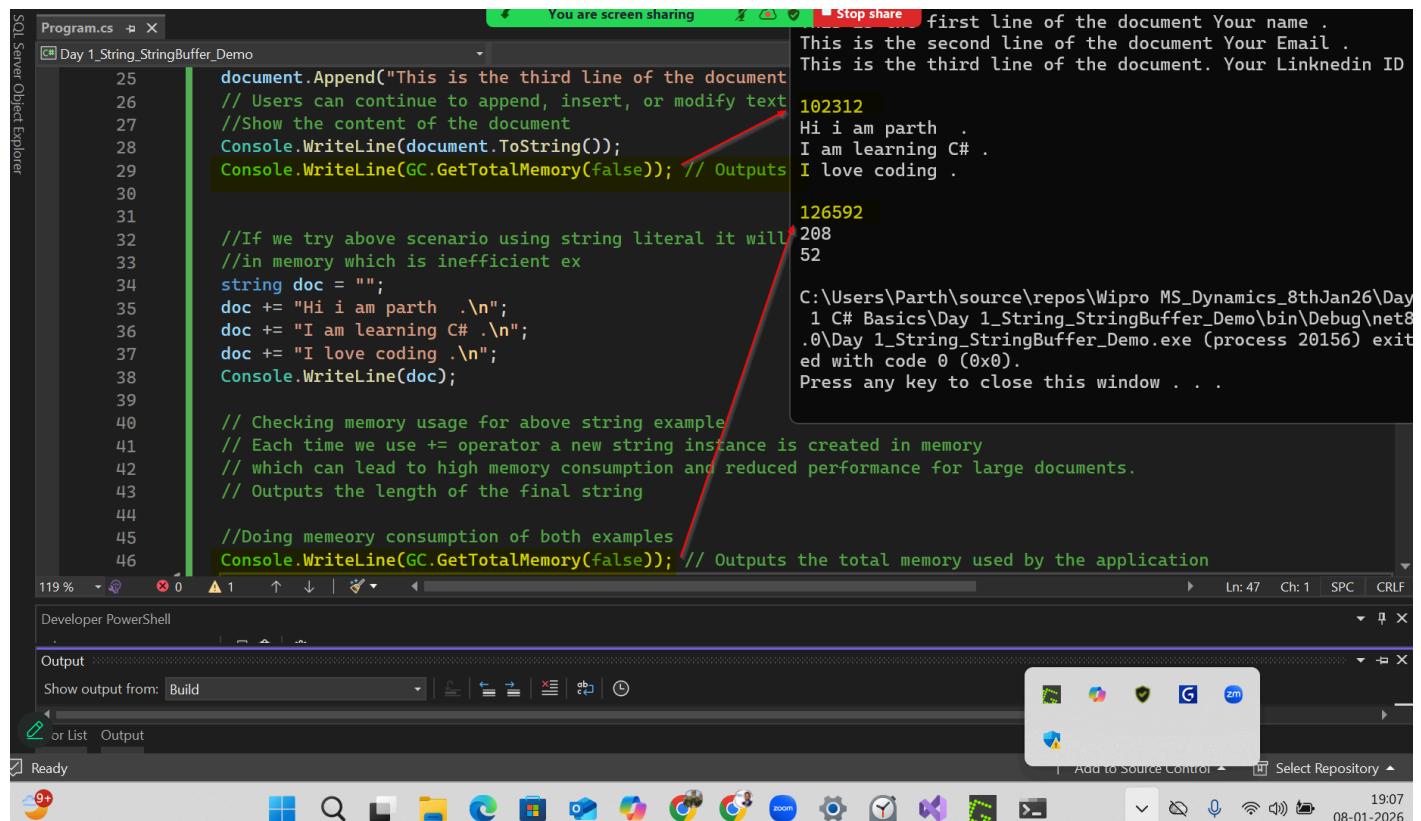


The screenshot shows a Microsoft Visual Studio interface. The top menu bar includes File, Edit, View, Git, Project, Build, Debug, Test, Analyze, Tools, Extensions, Window, Help, Search, and Day 1.String.StringBuffer_Demo. The toolbar has icons for file operations like Open, Save, and Print, along with GitHub Copilot. The Solution Explorer on the left shows a single item: Day 1.String.StringBuffer_Demo. The main code editor window displays the following C# code:

```
19  
20  
21     StringBuilder document = new StringBuilder();  
22     //here we are using StringBuilder to create a mutable string for our document (that can be modified)  
23     document.Append("This is the first line of the document Your name .\n");  
24     document.Append("This is the second line of the document Your Email .\n");  
25     document.Append("This is the third line of the document. Your Linknedin ID \n");  
26     // Users can continue to append, insert, or modify text in the document efficiently  
27     //Show the content of the document  
28     Console.WriteLine(document.ToString());  
29  
30
```

The output window below shows the console output:

```
Hello, World!  
This is the first line of the document Your name .  
This is the second line of the document Your Email .  
This is the third line of the document. Your Linknedin ID  
  
C:\Users\Parth\source\repos\Wipro MS_Dynamics_8thJan26\Day 1 C# Basics\Day 1_String.StringBuffer_Demo\bin\Debug\net8.0\Day 1_String.StringBuffer_Demo.exe (process 5032) exited with code 0 (0x0).  
Press any key to close this window . . .
```



The screenshot shows a Microsoft Visual Studio interface with the following details:

- Code Editor:** The file `Program.cs` contains C# code demonstrating string manipulation and memory usage. It includes examples of using `+=` operator, `ToString()`, and `GetTotalMemory()`.
- Output Window:** Shows the execution results. The output includes three lines of text from the console and memory usage statistics.
- Taskbar:** Shows various pinned application icons.
- System Tray:** Shows the date and time as 08-01-2026 at 19:07.

```

1 first line of the document Your name .
2 This is the second line of the document Your Email .
3 This is the third line of the document. Your Linknedin ID
4
5 102312
6 Hi i am parth .
7 I am learning C# .
8 I love coding .
9
10 126592
11 208
12 52
13
14 C:\Users\Parth\source\repos\Wipro MS_Dynamics_8thJan26\Day
15 1 C# Basics\Day 1_String_Buffer_Demo\bin\Debug\net8
16 .0\Day 1_String_Buffer_Demo.exe (process 20156) exit
17 ed with code 0 (0x0).
18 Press any key to close this window . . .
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46

```