**Hands on Day -3**

**INTCDE21ID008**

**C N S Varun**

**916214**

**HANDSON-1**

Async Await usage – 1

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading;

using System.Threading.Tasks;

namespace AsyncHandsOn

{

class Program

{

public static async Task firstMethod()

{

await Task.Run(() =>

{

string a = secondMethod();

Console.WriteLine(a);

});

}

public static string secondMethod()

{

Console.WriteLine("Wait until string returns");

Thread.Sleep(3340);

return "Call from 2nd method";

}

static void Main(string[] args)

{

firstMethod();

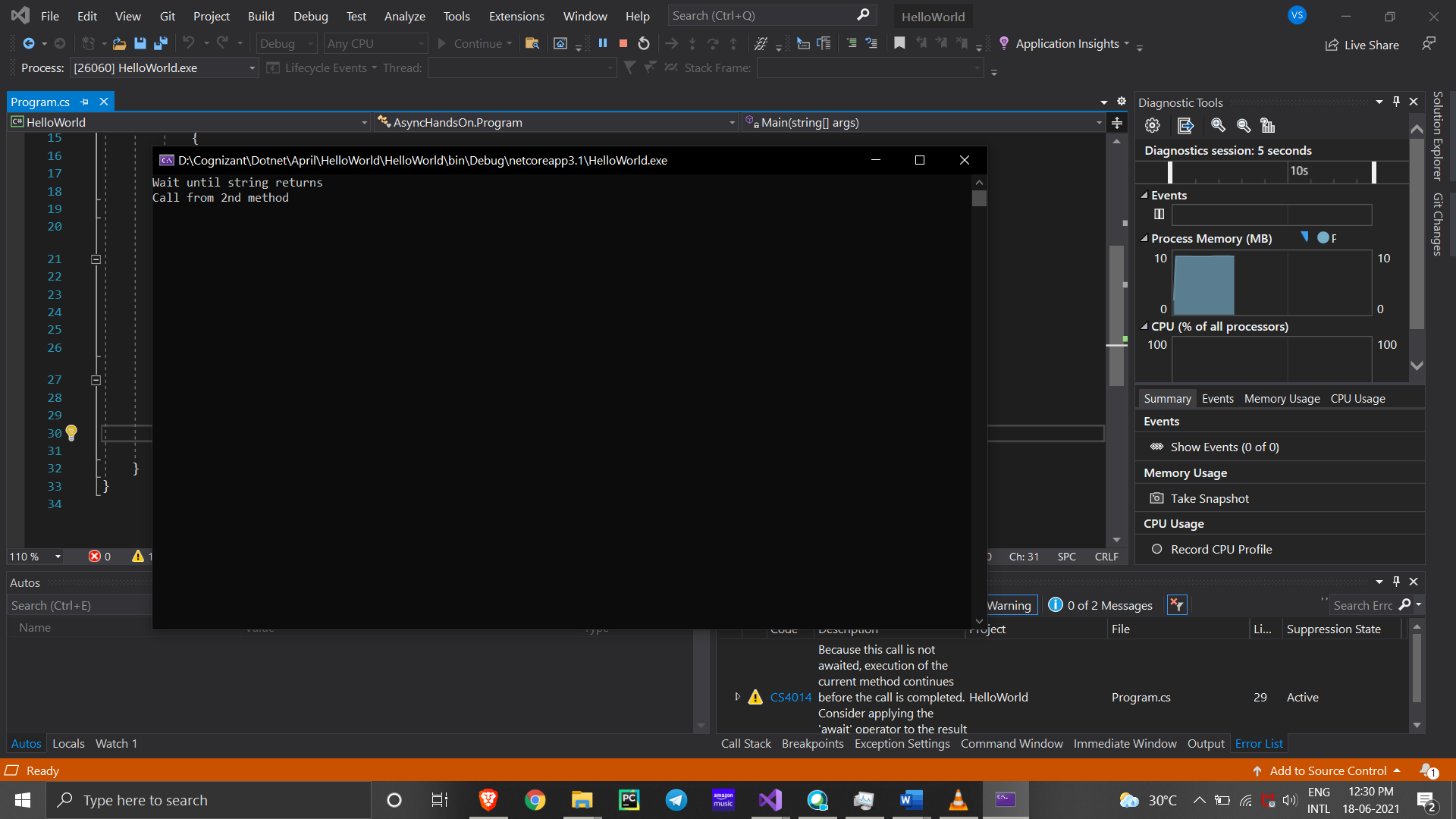
Console.ReadKey();

}

}

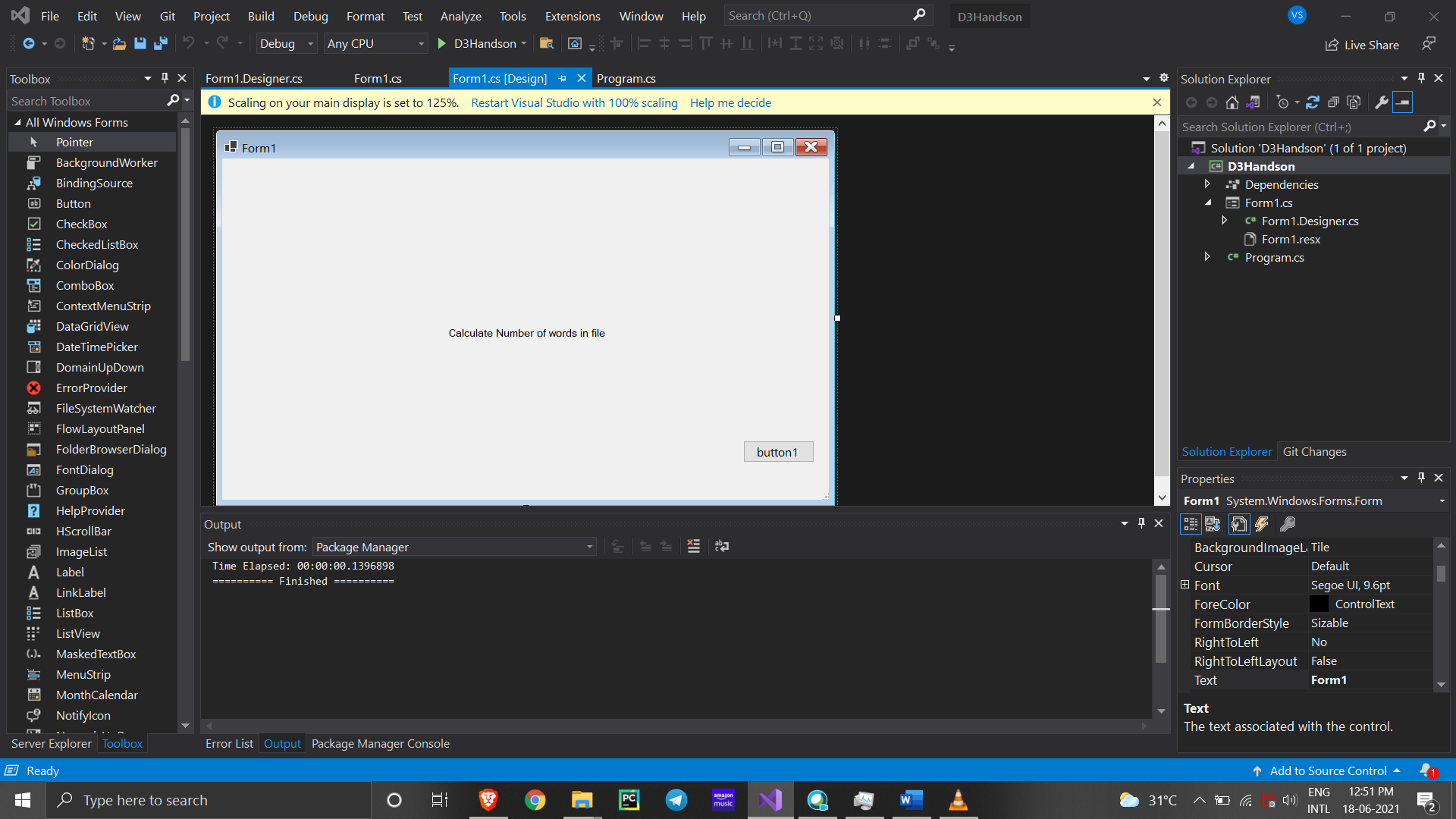
}

**OUTPUT**



Async Await usage – 2

Form



FormDesigner.cs

namespace D3Handson

{

partial class Form1

{

/// <summary>

/// Required designer variable.

/// </summary>

private System.ComponentModel.IContainer components = null;

/// <summary>

/// Clean up any resources being used.

/// </summary>

/// <param name="disposing">true if managed resources should be disposed; otherwise, false.</param>

protected override void Dispose(bool disposing)

{

if (disposing && (components != null))

{

components.Dispose();

}

base.Dispose(disposing);

}

#region Windows Form Designer generated code

/// <summary>

/// Required method for Designer support - do not modify

/// the contents of this method with the code editor.

/// </summary>

private void InitializeComponent()

{

this.label1 = new System.Windows.Forms.Label();

this.button1 = new System.Windows.Forms.Button();

this.SuspendLayout();

//

// label1

//

this.label1.AutoSize = true;

this.label1.Font = new System.Drawing.Font("Microsoft Sans Serif", 8.25F, System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point);

this.label1.Location = new System.Drawing.Point(296, 221);

this.label1.Name = "label1";

this.label1.Size = new System.Drawing.Size(214, 17);

this.label1.TabIndex = 0;

this.label1.Text = "Calculate Number of words in file";

//

// button1

//

this.button1.Location = new System.Drawing.Point(687, 372);

this.button1.Name = "button1";

this.button1.Size = new System.Drawing.Size(94, 29);

this.button1.TabIndex = 1;

this.button1.Text = "button1";

this.button1.UseVisualStyleBackColor = true;

this.button1.Click += new System.EventHandler(this.button1\_Click);

//

// Form1

//

this.AutoScaleDimensions = new System.Drawing.SizeF(9F, 21F);

this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;

this.ClientSize = new System.Drawing.Size(800, 450);

this.Controls.Add(this.button1);

this.Controls.Add(this.label1);

this.Name = "Form1";

this.SizeGripStyle = System.Windows.Forms.SizeGripStyle.Show;

this.Text = "Form1";

this.Load += new System.EventHandler(this.Form1\_Load);

this.ResumeLayout(false);

this.PerformLayout();

}

#endregion

private System.Windows.Forms.Label label1;

private System.Windows.Forms.Button button1;

}

}

Form.cs

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.IO;

using System.Linq;

using System.Text;

using System.Threading;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace D3Handson

{

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

}

public int CountChars()

{

int count = 0;

using (StreamReader streamReader = new StreamReader("D:\\Cognizant\\3 Stage HandsonAssignments\\Sample.txt"))

{

string content = streamReader.ReadToEnd();

count = content.Length;

Thread.Sleep(2000);

}

return count;

}

private void label1\_Click(object sender, EventArgs e)

{

}

private void Form1\_Load(object sender, EventArgs e)

{

}

private async void button1\_Click(object sender, EventArgs e)

{

Task<int> task = new Task<int>(CountChars);

task.Start();

//label1.Text = "File is Processing";

int count = await task;

label1.Text = count.ToString() + " characters";

}

}

}

Program.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace D3Handson

{

static class Program

{

/// <summary>

/// The main entry point for the application.

/// </summary>

[STAThread]

static void Main()

{

Application.SetHighDpiMode(HighDpiMode.SystemAware);

Application.EnableVisualStyles();

Application.SetCompatibleTextRenderingDefault(false);

Application.Run(new Form1());

}

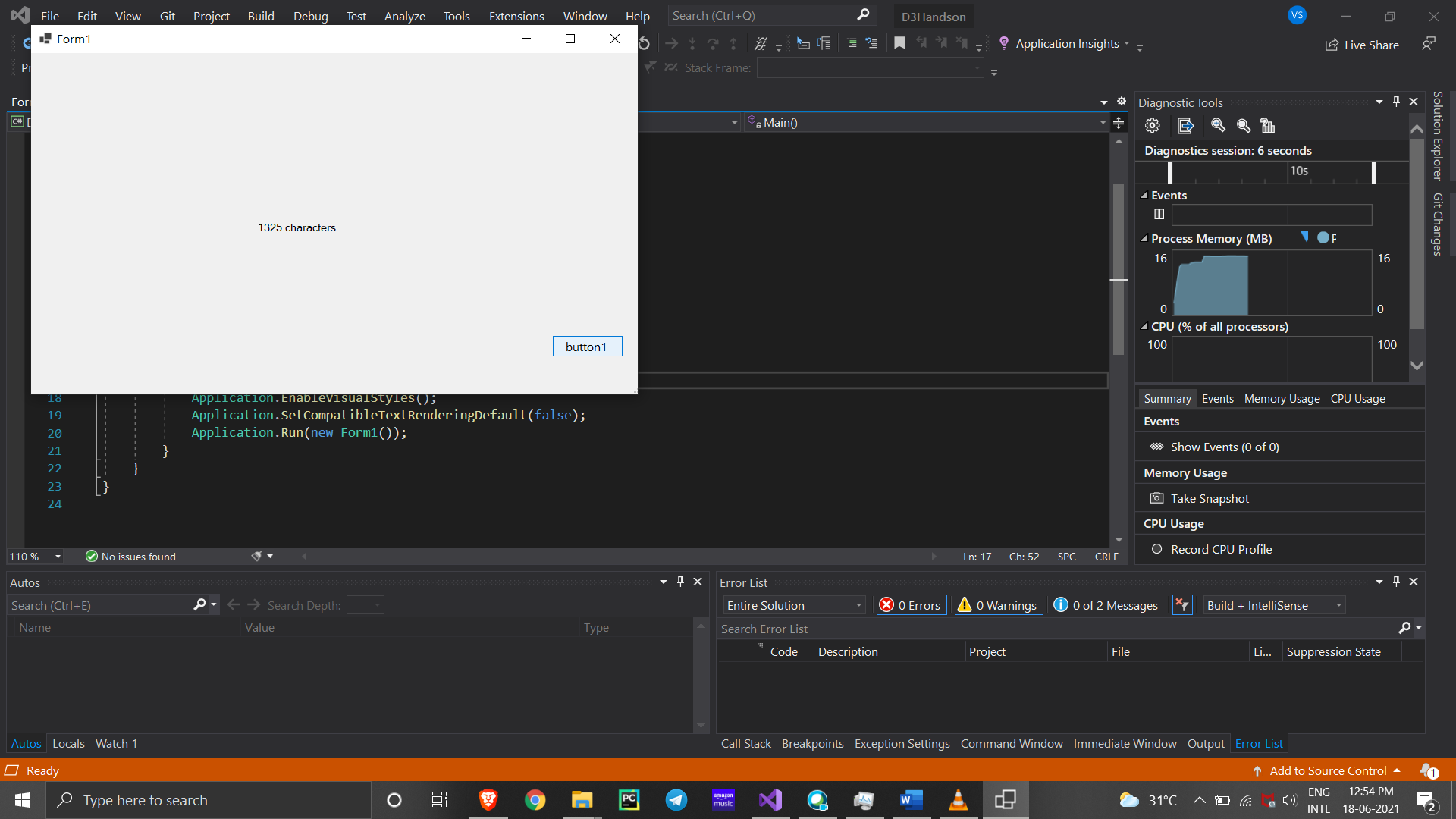
}

}

TextFile:

Augmented Reality (AR) is one of the best technology enhanced and interactive hybrid learning environment that helps to increase the conceptual understanding of any phenomena. One of its most imperative areas is the use of Augmented Reality applications in teaching, training etc. which increases the potential of AR to be incorporated in education. AR technology enables the addition of virtual objects into real environments there by encourage students to show an intense level of interest towards learning and enhance better visualization of concepts in academics. Learning information about any course through AR will be more feasible than learning from the internet or books. The project involves developing an Application/Mobile App (Android) for Augmented lab that will facilitate the students to master the concepts and procedure of experiments in an Augmented way and also can learn about electronic components information like microprocessors, transistors, antenna, etc. through AR. The built in application allows the students to scan the equipment/components through the app that would be recognized and an instant information in the format of text and video of that equipment/component will be displayed on equipment virtually. The Unity 3D software linked with Vuforia Engine is used to develop the application.

OUTPUT



Named parameters – Order of arguments as per the function and modify

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Handson4.\_5

{

class NamedParameter

{

static void GetCohortDetails(string Cohortname, int Genccount, string Mode, string Track, string CurrentModule)

{

Console.WriteLine("It is {0} with {1} GenCs undergoing training for {2} thru {3}. The current module of training is {4}", Cohortname, Genccount, Track, Mode, CurrentModule);

}

// Optional Prameters

static void OrderDetails(string Productname, string Sellername, int Orderquantity = 1, bool returnable = true)

{

Console.WriteLine("Here is the order detail – {0} number of {1} by {2} is ordered. It’s returnable status is {3}", Orderquantity, Productname, Sellername, returnable);

}

static void Main(string[] args)

{

Console.WriteLine("Cohort Details");

// Named parameters

GetCohortDetails(Cohortname: "CDE", Genccount: 20, Track: "Java", Mode: "OBL", CurrentModule: "Stage 3");

GetCohortDetails(Cohortname: "CDE", Genccount: 18, Mode: "PARC", Track: ".Net", CurrentModule: "Stage 3");

Console.WriteLine("-------------------------------------");

// Optional parameters

Console.WriteLine("Order Details");

OrderDetails(Sellername: "abc", Productname: "def", Orderquantity: 10, returnable: false);

OrderDetails(Sellername: "abc", Productname: "def");

Console.WriteLine("-------------------------------------");

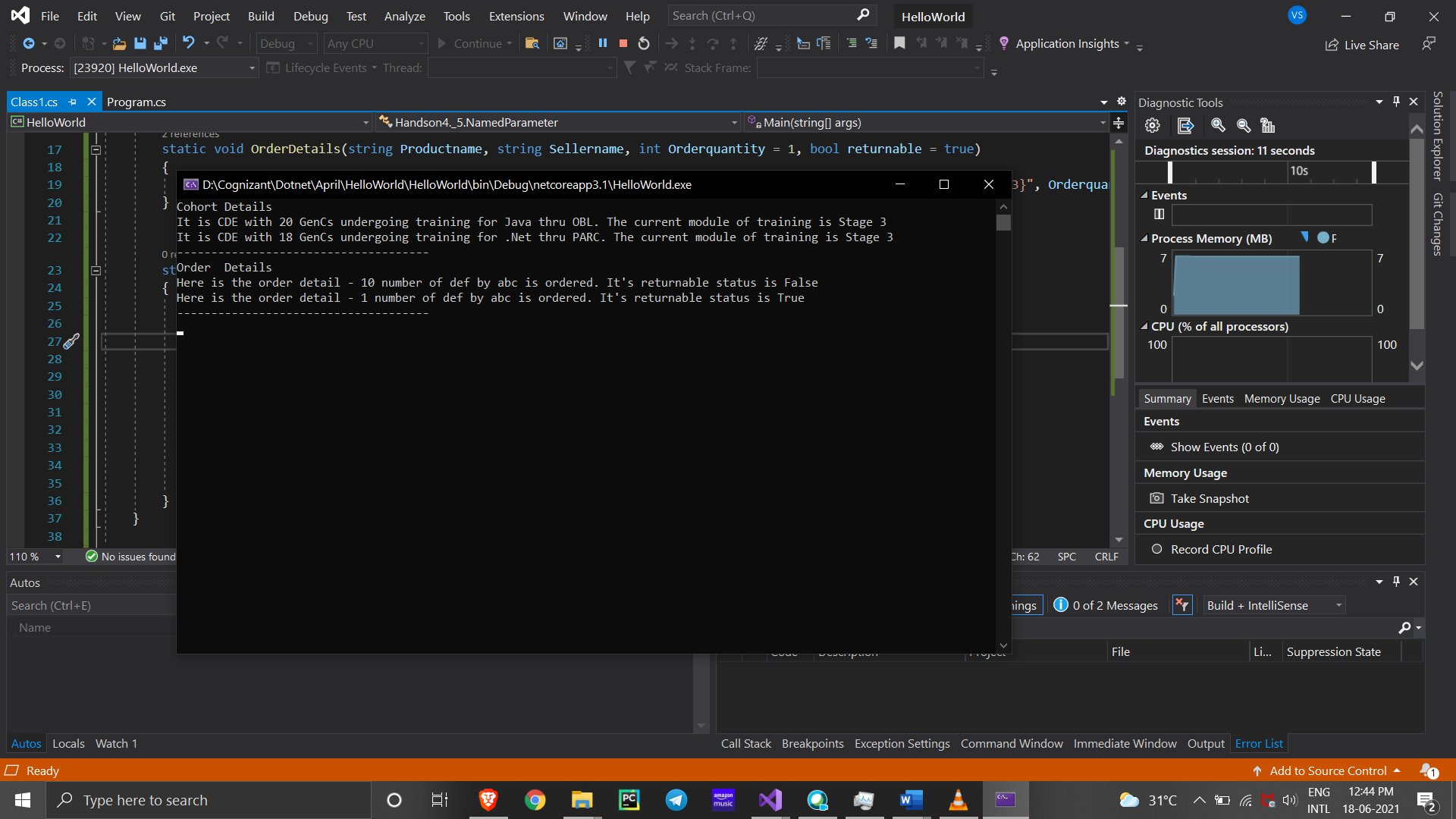
Console.ReadLine();

}

}

}

**OUTPUT**



**HANDSON-2**

**Program.Cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading;

using System.Threading.Tasks;

namespace D3HandOn2

{

class Program

{

static void Main(string[] args)

{

Console.WriteLine("\*\*\*\*\*ThreadStart Delegate App\*\*\*\*\*\n");

Console.Write("Do you want [1] or [2] threads? ");

string threadCount = Console.ReadLine();

// Name the current thread.

Thread primaryThread = Thread.CurrentThread;

primaryThread.Name = "Primary";

// Display Thread info.

Console.WriteLine("-> {0} is executing Main()", Thread.CurrentThread.Name);

// Make worker class.

Printer p = new Printer();

switch (threadCount)

{

case "2":

// Now make the thread.

Thread backgroundThread = new Thread(new ThreadStart(p.PrintNumbers));

backgroundThread.Name = "Secondary";

backgroundThread.Start();// Changes the state of current instance to ThreadState.Running.

break;

case "1":

p.PrintNumbers();

break;

default:

Console.WriteLine("I don't know what you want... you get 1 thread.");

goto case "1";

}

// Do some additional work.

Console.WriteLine("Hello this from main!");

Console.Read();

}

}

}

**Printer.CS**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading;

using System.Threading.Tasks;

namespace D3HandOn2

{

class Printer

{

public void PrintNumbers()

{

// Display Thread info.

Console.WriteLine("-> {0} is executing PrintNumbers()", Thread.CurrentThread.Name);

// Print out numbers.

Console.Write("Your numbers: ");

for (int i = 0; i < 10; i++)

{

Console.Write("{0}, ", i);

Thread.Sleep(2000);

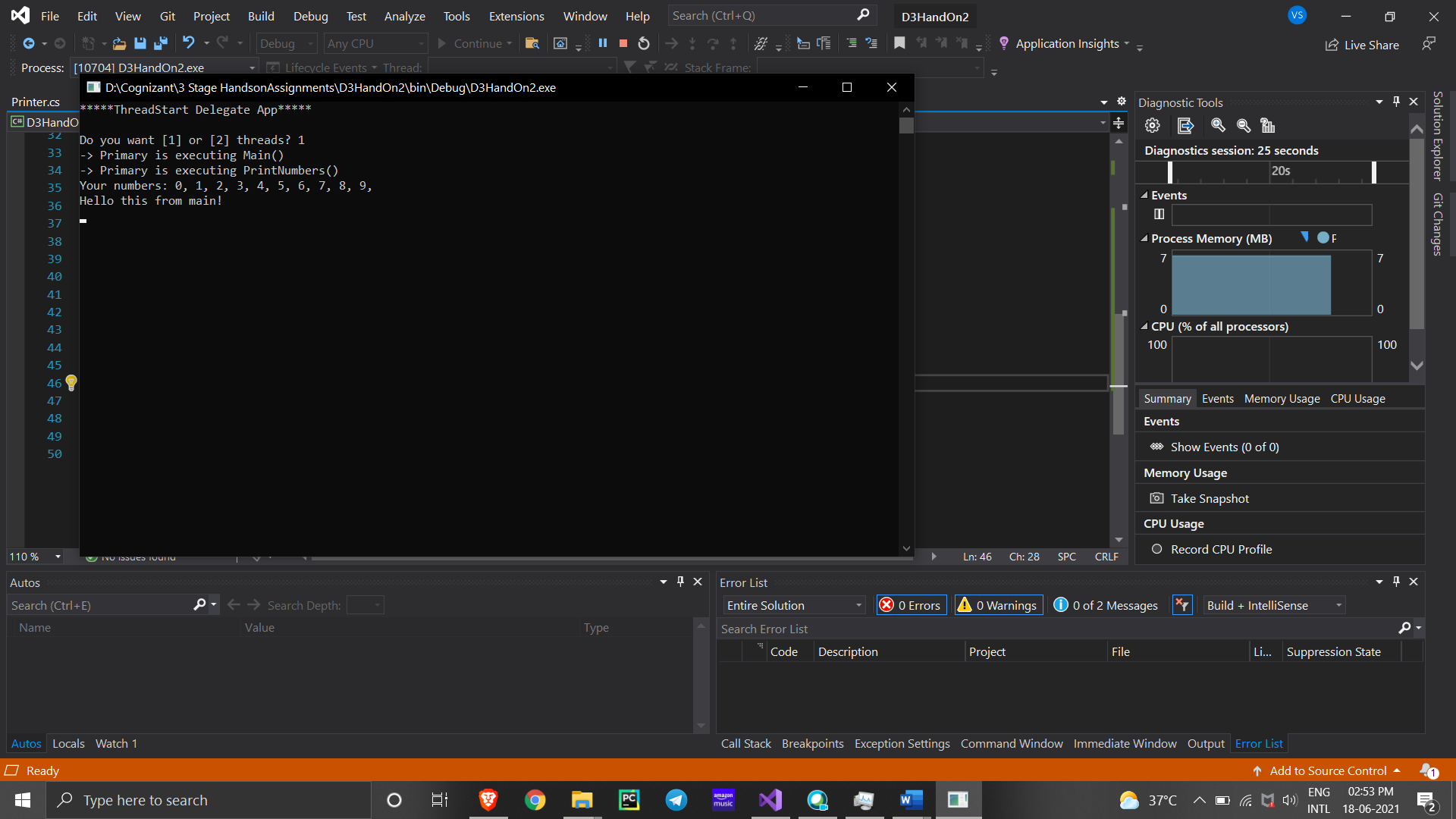
}

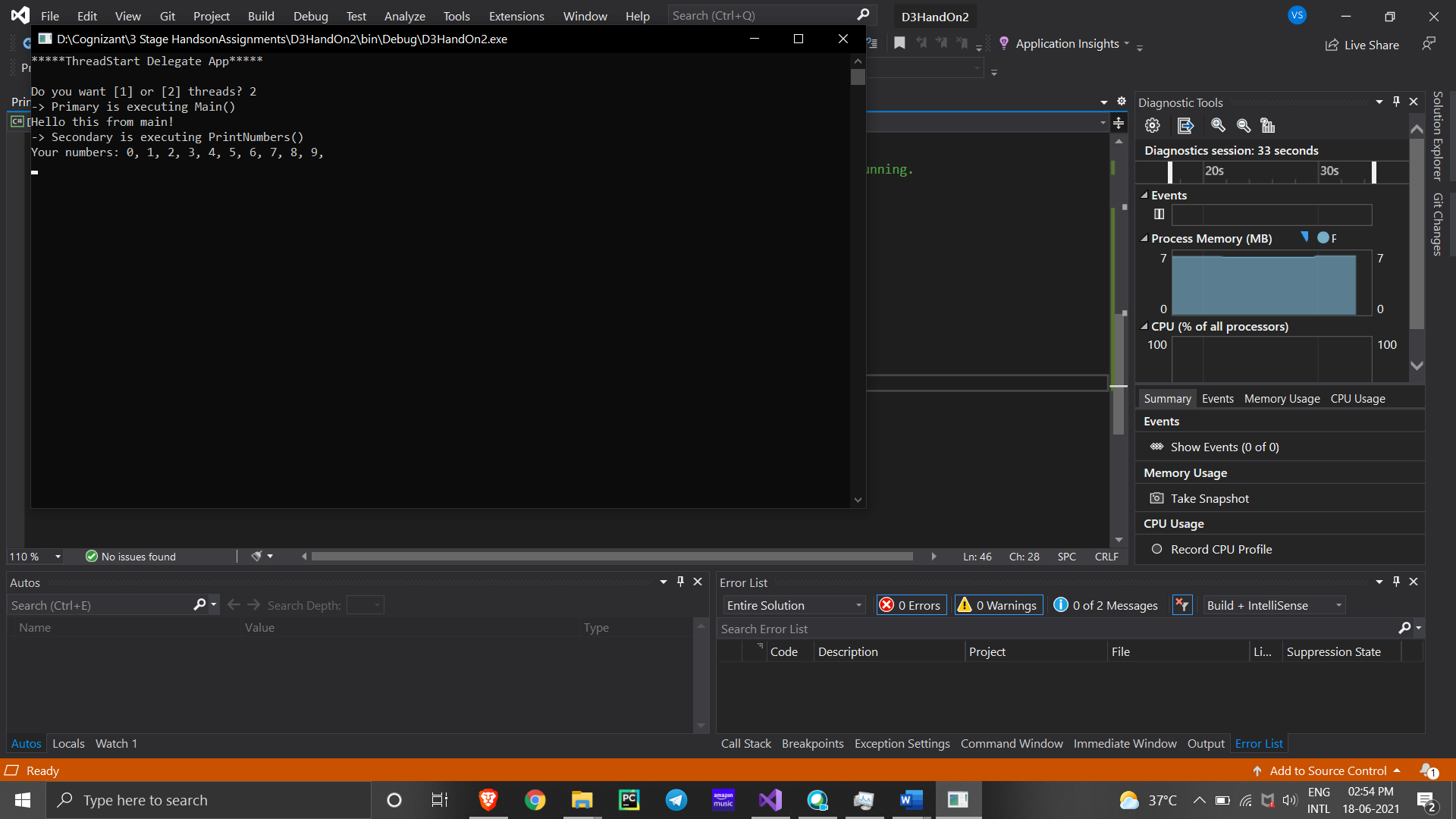
Console.WriteLine();

}

}

}





**Printer.CS**

using System;

using System.Threading;

namespace D3Handon2\_

{

class Printer

{

private object lockToken = new object();

public void PrintNumbers()

{

lock (lockToken)

{

// Display Thread info.

Console.WriteLine("Thread-> {0} started @{1} and executing PrintNumbers() method",

Thread.CurrentThread.ManagedThreadId, DateTime.Now.ToLongTimeString());

// Print out numbers.

Console.Write("Your numbers: ");

for (int i = 0; i < 10; i++)

{

Console.Write("{0}, ", i);

Thread.Sleep(500);

}

Console.WriteLine();

}

}

}

}

**Program.Cs**

using System;

using System.Threading;

namespace D3Handon2\_

{

class Program

{

static void PrintTheNumbers(object state)

{

Printer task = (Printer)state;

task.PrintNumbers();

}

static void Main(string[] args)

{

Console.WriteLine("\*\*\*\*\*Multithreading Program\*\*\*\*\*\n");

Console.WriteLine("Main thread started. ThreadID = {0}",

Thread.CurrentThread.ManagedThreadId);

Printer p = new Printer();

WaitCallback workItem = new WaitCallback(PrintTheNumbers);

// Queue the method 10 times.

for (int i = 0; i < 10; i++)

{

ThreadPool.QueueUserWorkItem(workItem, p);

}

Console.WriteLine("All task Queued");

Console.ReadLine();

}

}

}

