**Exercise 1: Implementing the Singleton Pattern**

using System;

public class LoggerSingletonExample

{

public static void Main(string[] args)

{

Logger logger1 = Logger.GetInstance();

logger1.Log("message (1)");

Logger logger2 = Logger.GetInstance();

logger2.Log("message (2)");

if (object.ReferenceEquals(logger1, logger2))

{

Console.WriteLine("Singleton works");

}

else

{

Console.WriteLine("Singleton has failed");

}

}

}

public class Logger

{

private static Logger instance;

private Logger() {}

public static Logger GetInstance()

{

if (instance == null)

{

instance = new Logger();

}

return instance;

}

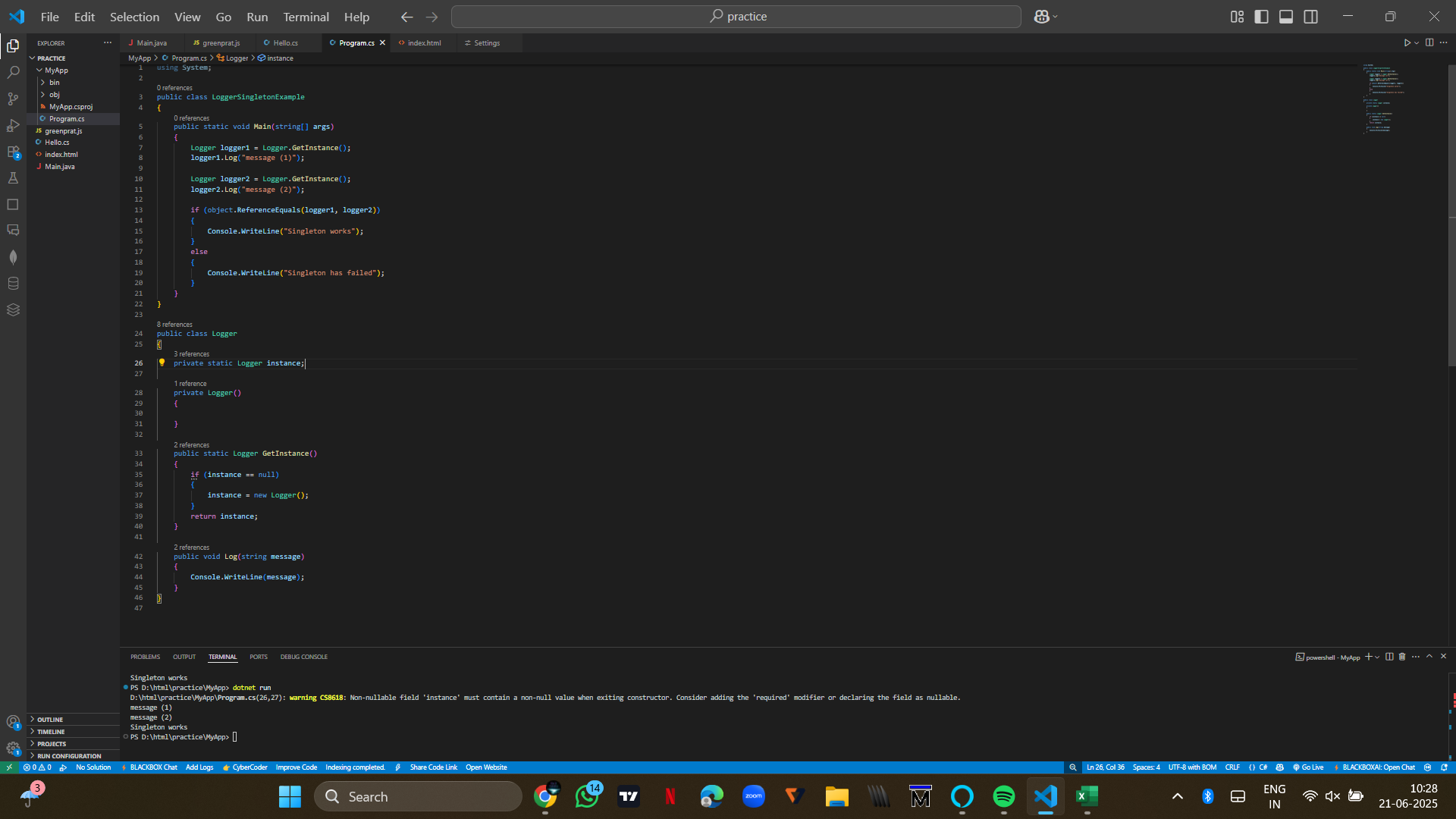
public void Log(string message)

{

Console.WriteLine(message);

}

}



**Exercise 2: Implementing the Factory Method Pattern**

using System;

interface IDocument

{

void Open();

void Save();

}

class WordDocument : IDocument

{

public void Open() => Console.WriteLine("Word document opened");

public void Save() => Console.WriteLine("Word document saved");

}

class PdfDocument : IDocument

{

public void Open() => Console.WriteLine("PDF document opened");

public void Save() => Console.WriteLine("PDF document saved");

}

class ExcelDocument : IDocument

{

public void Open() => Console.WriteLine("Excel document opened");

public void Save() => Console.WriteLine("Excel document saved");

}

abstract class DocumentFactory

{

public abstract IDocument CreateDocument();

}

class WordDocumentFactory : DocumentFactory

{

public override IDocument CreateDocument() => new WordDocument();

}

class PdfDocumentFactory : DocumentFactory

{

public override IDocument CreateDocument() => new PdfDocument();

}

class ExcelDocumentFactory : DocumentFactory

{

public override IDocument CreateDocument() => new ExcelDocument();

}

public class FactoryMethodPatternExample

{

public static void Main(string[] args)

{

DocumentFactory wordFactory = new WordDocumentFactory();

IDocument wordDoc = wordFactory.CreateDocument();

wordDoc.Open();

wordDoc.Save();

DocumentFactory pdfFactory = new PdfDocumentFactory();

IDocument pdfDoc = pdfFactory.CreateDocument();

pdfDoc.Open();

pdfDoc.Save();

DocumentFactory excelFactory = new ExcelDocumentFactory();

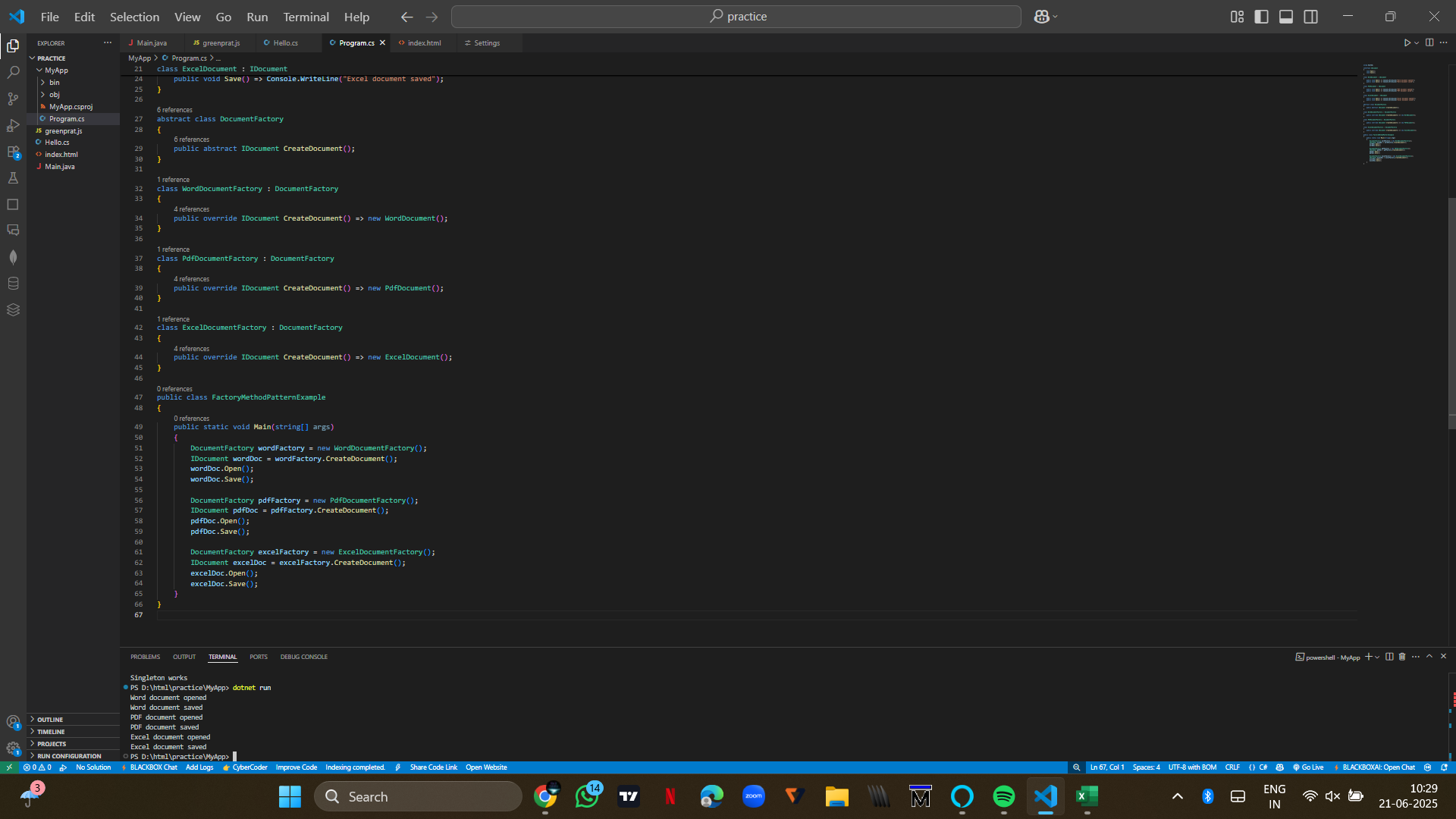
IDocument excelDoc = excelFactory.CreateDocument();

excelDoc.Open();

excelDoc.Save();

}

}}



**Exercise 2: E-commerce Platform Search Function**

using System;

class Product

{

public int Id;

public string Name;

public string Category;

public Product(int id, string name, string category)

{

Id = id;

Name = name;

Category = category;

}

}

class Program

{

static void LinearSearch(Product[] products, string searchName)

{

foreach (var product in products)

{

if (string.Equals(product.Name, searchName, StringComparison.OrdinalIgnoreCase))

{

Console.WriteLine($"Found: {product.Name} in {product.Category}");

return;

}

}

}

static void BinarySearch(Product[] products, string searchName)

{

Array.Sort(products, CompareByName);

int low = 0, high = products.Length - 1;

while (low <= high)

{

int mid = (low + high) / 2;

int result = string.Compare(products[mid].Name, searchName, true);

if (result == 0)

{

Console.WriteLine($"Found: {products[mid].Name} in {products[mid].Category}");

return;

}

else if (result < 0)

low = mid + 1;

else

high = mid - 1;

}

}

static int CompareByName(Product a, Product b)

{

return string.Compare(a.Name, b.Name, true);

}

static void Main()

{

Product[] products = {

new Product(1, "Phone", "Electronics"),

new Product(2, "Shoes", "Footwear"),

new Product(3, "Bag", "Accessories"),

new Product(4, "T-Shirt", "Clothing"),

new Product(5, "Watch", "Accessories")

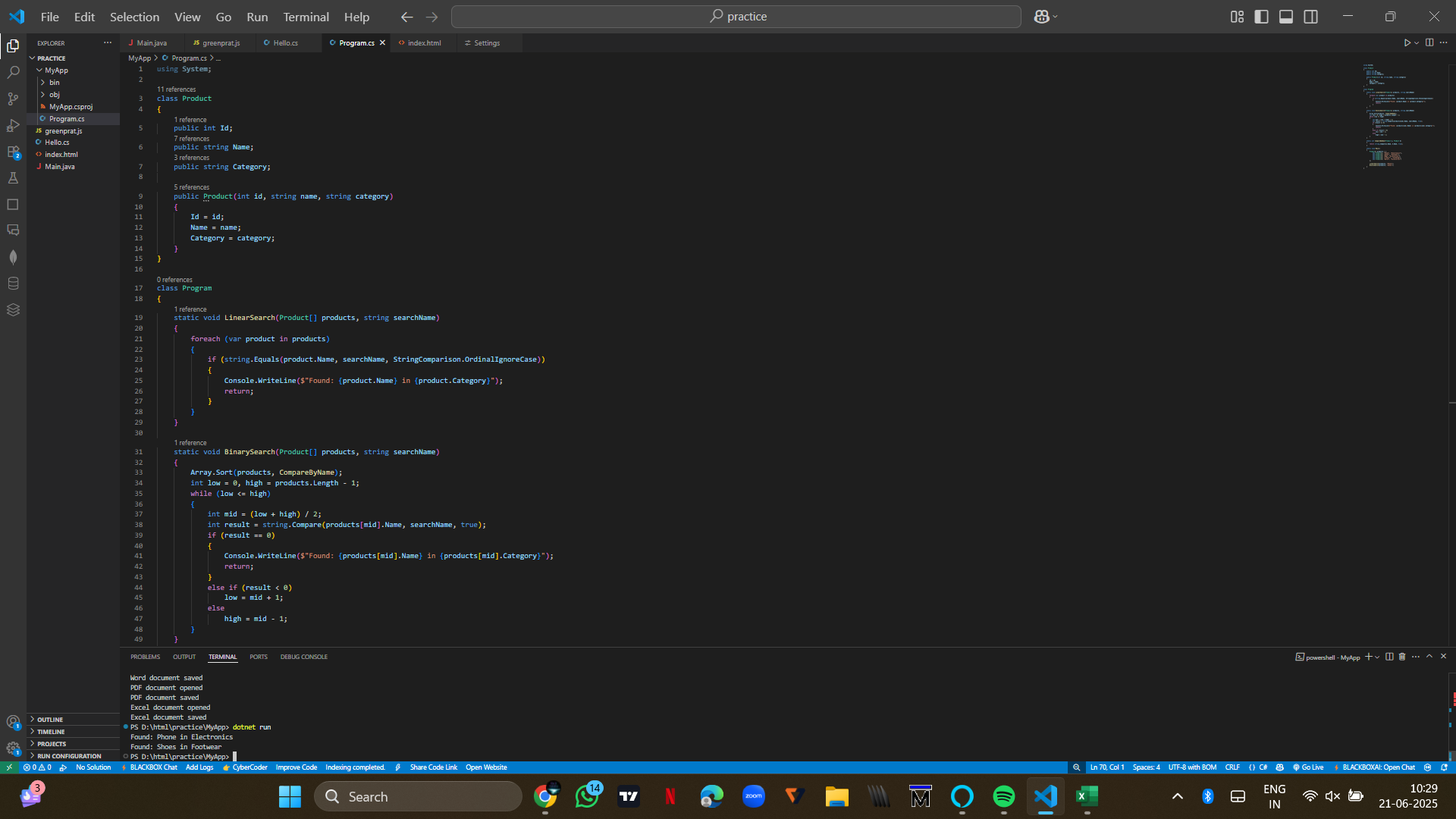
};

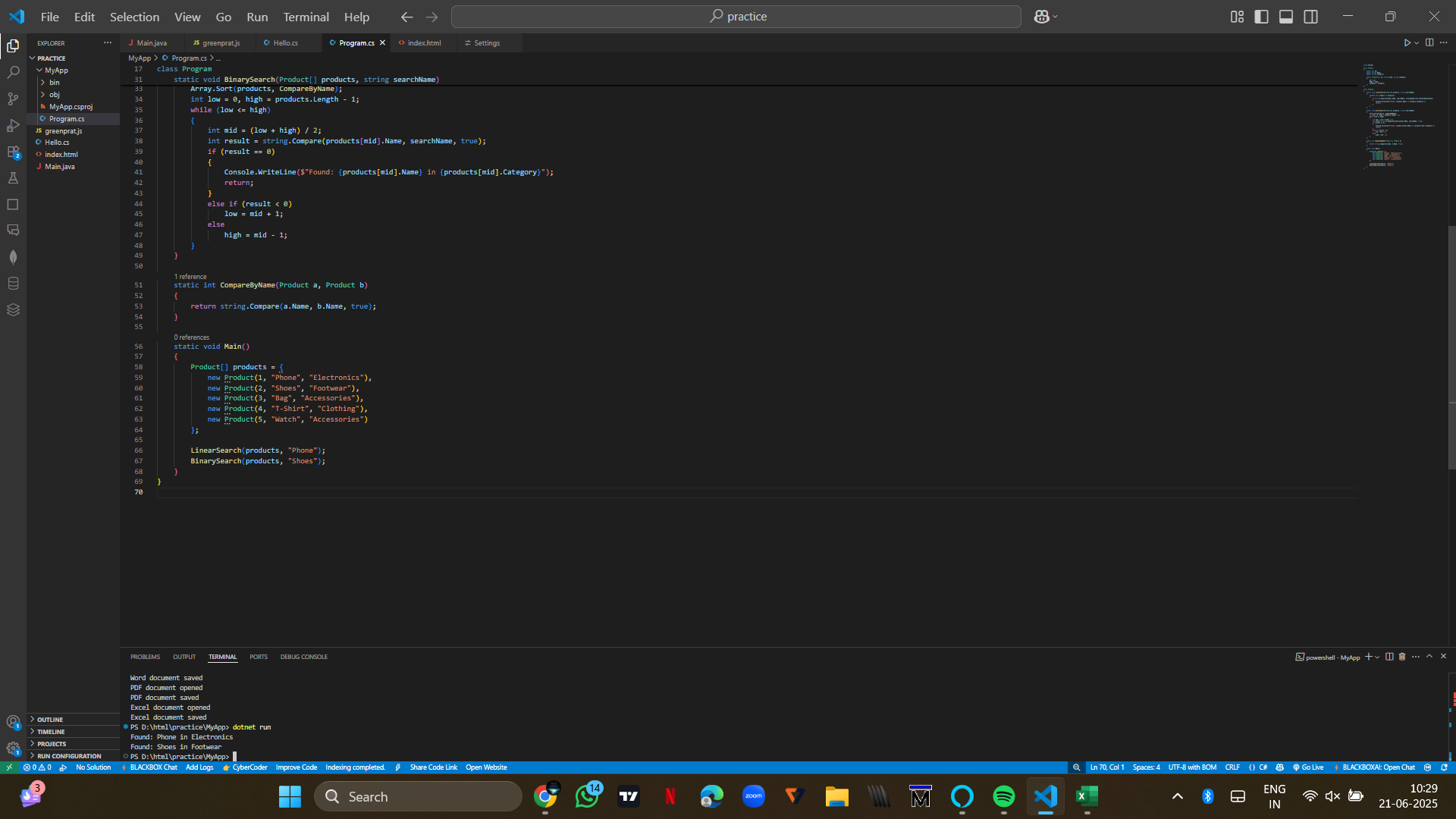
LinearSearch(products, "Phone");

BinarySearch(products, "Shoes");

}

}}





**Exercise 7: Financial Forecasting**

using System;

class Program

{

static void Main()

{

double result = Predict(1000, 0.10, 5);

Console.WriteLine("Future Value: " + result);

}

static double Predict(double value, double rate, int years)

{

if (years == 0) return value;

return Predict(value, rate, years - 1) \* (1 + rate);

}

}

