COS30041

Creating Secure and Scalable Software

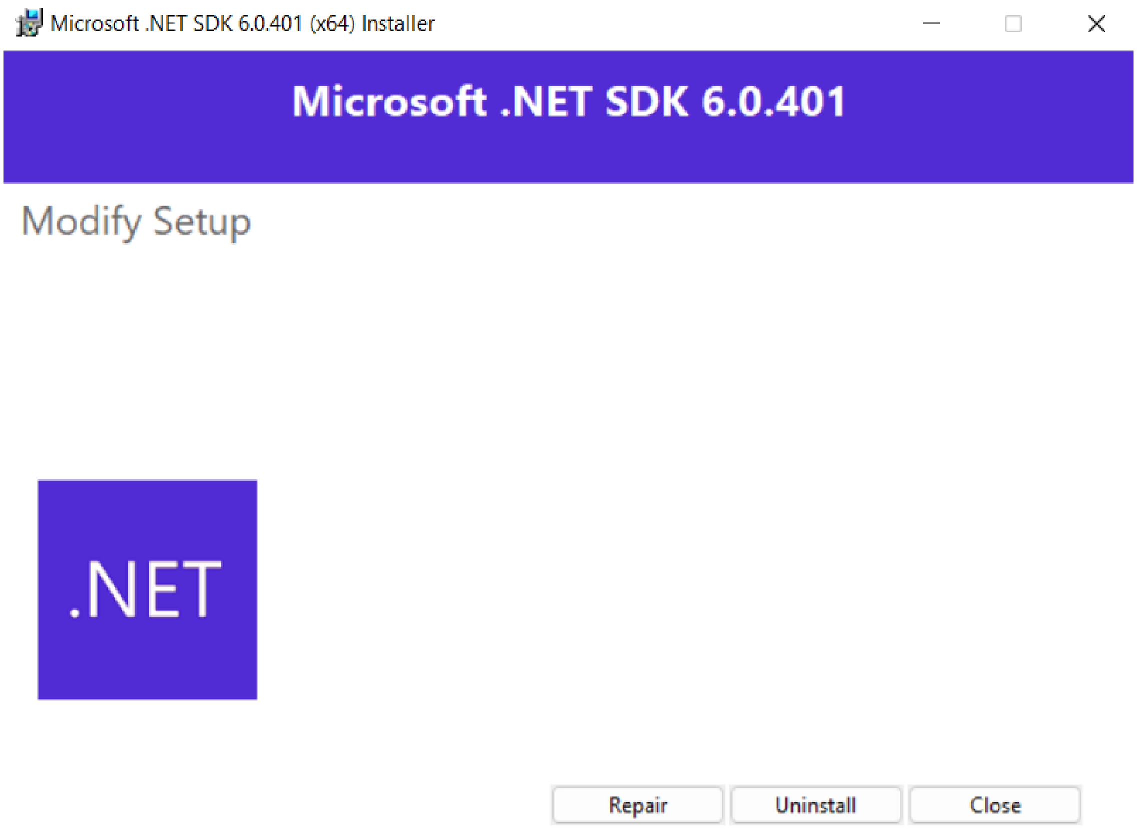
Pass Task 2

Noor Ul Ain KHURSHID
102763334

LAB TASKS

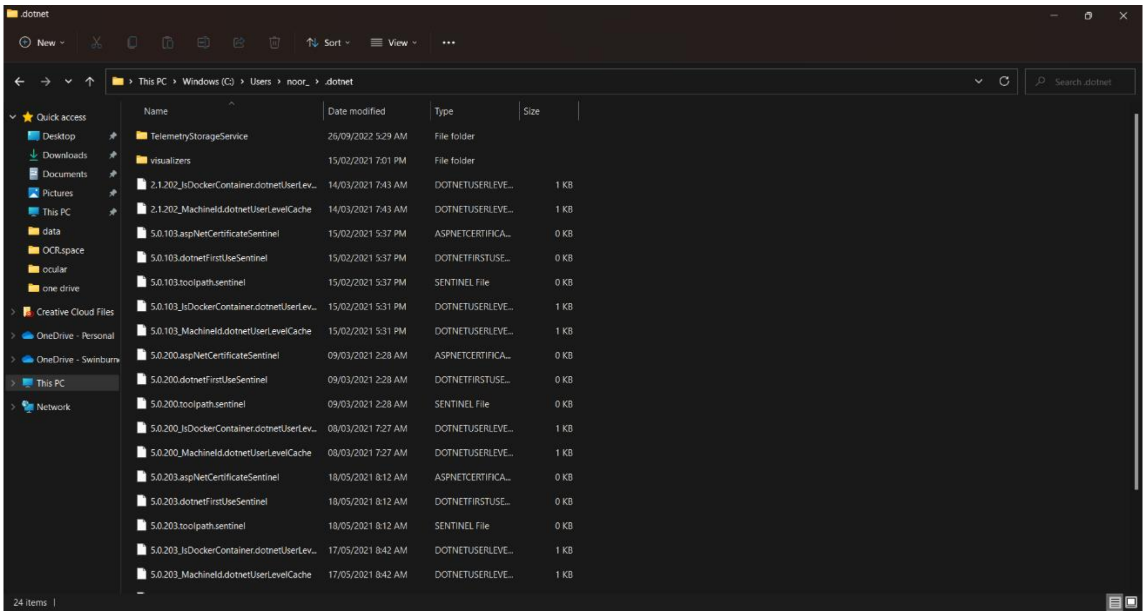
LT1. Install SDK and set up environment variables for the compiler

LT1.1. Download SDK from the URL listed above


Microsoft .NET SDK 6.0.401 (x64) Installer
Microsoft .NET SDK 6.0.401
Modify Setup
.NET
Repair
Uninstall
Close

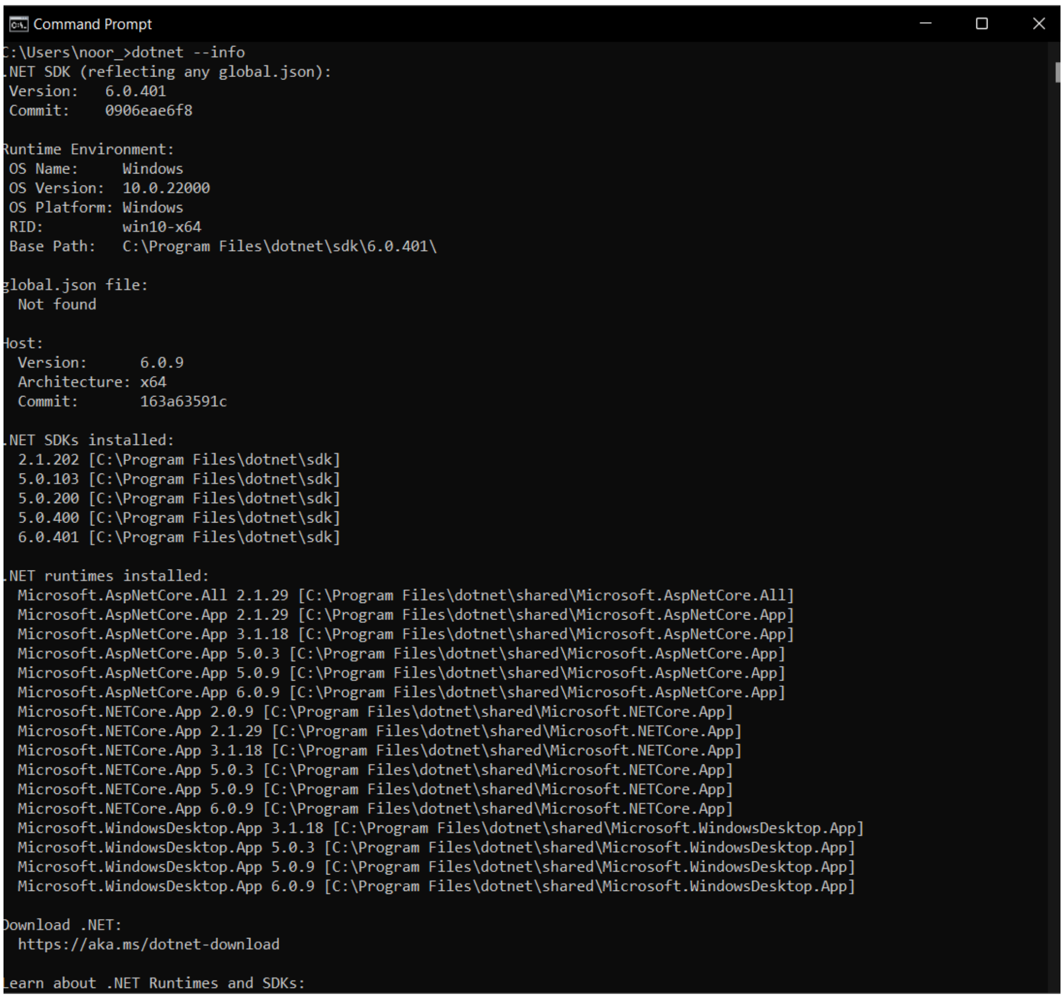
LT1.2. Install SDK and note the folders where SDK are installed

C:\Users\noor_\.dotnet



Name	Date modified	Type	Size
TelemetryStorageService	26/09/2022 5:29 AM	File folder	
visualizers	15/02/2021 7:01 PM	File folder	
2.1.202_IsDockerContainer.dotnetUserLev...	14/03/2021 7:43 AM	DOTNETUSERLEVE...	1 KB
2.1.202_MachineId.dotnetUserLevelCache	14/03/2021 7:43 AM	DOTNETUSERLEVE...	1 KB
5.0.103.aspNetCertificateSentinel	15/02/2021 5:37 PM	ASPNETCERTIFICA...	0 KB
5.0.103.dotnetFirstUseSentinel	15/02/2021 5:37 PM	DOTNETFIRSTUSE...	0 KB
5.0.103.toolpath.sentinel	15/02/2021 5:37 PM	SENTINEL File	0 KB
5.0.103_IsDockerContainer.dotnetUserLev...	15/02/2021 5:31 PM	DOTNETUSERLEVE...	1 KB
5.0.103_MachineId.dotnetUserLevelCache	15/02/2021 5:31 PM	DOTNETUSERLEVE...	1 KB
5.0.200.aspNetCertificateSentinel	09/03/2021 2:28 AM	ASPNETCERTIFICA...	0 KB
5.0.200.dotnetFirstUseSentinel	09/03/2021 2:28 AM	DOTNETFIRSTUSE...	0 KB
5.0.200.toolpath.sentinel	09/03/2021 2:28 AM	SENTINEL File	0 KB
5.0.200_IsDockerContainer.dotnetUserLev...	08/03/2021 7:27 AM	DOTNETUSERLEVE...	1 KB
5.0.200_MachineId.dotnetUserLevelCache	08/03/2021 7:27 AM	DOTNETUSERLEVE...	1 KB
5.0.203.aspNetCertificateSentinel	18/05/2021 8:12 AM	ASPNETCERTIFICA...	0 KB
5.0.203.dotnetFirstUseSentinel	18/05/2021 8:12 AM	DOTNETFIRSTUSE...	0 KB
5.0.203.toolpath.sentinel	18/05/2021 8:12 AM	SENTINEL File	0 KB
5.0.203_IsDockerContainer.dotnetUserLev...	17/05/2021 8:42 AM	DOTNETUSERLEVE...	1 KB
5.0.203_MachineId.dotnetUserLevelCache	17/05/2021 8:42 AM	DOTNETUSERLEVE...	1 KB

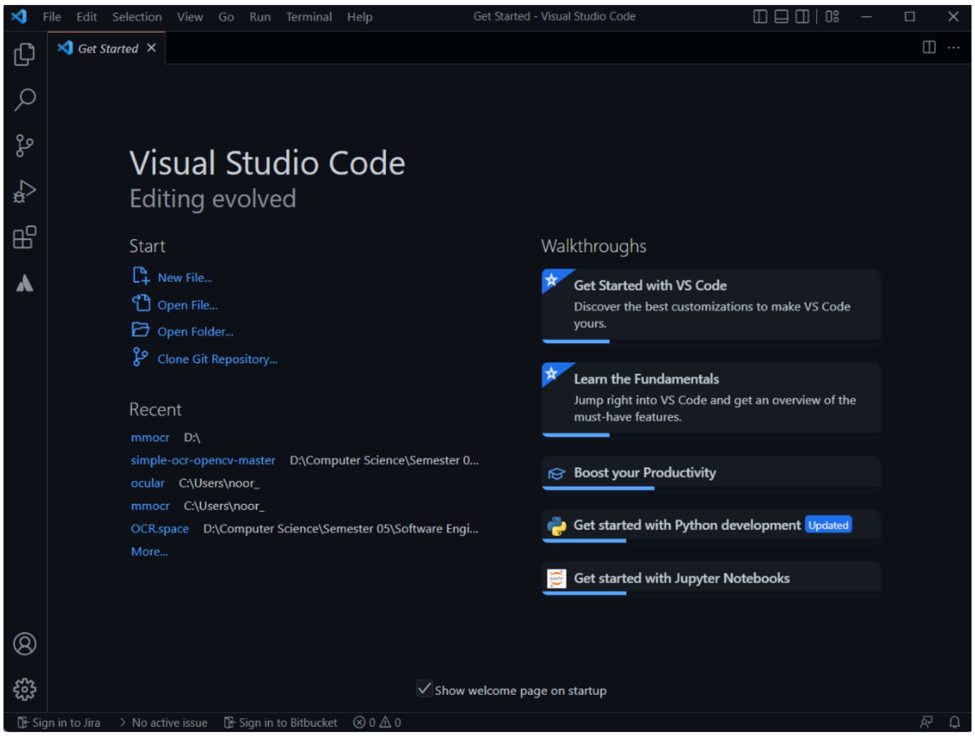
LT1.3. Run the ‘dotnet --info’ command



```
:\Users\noor_>dotnet --info
NET SDK (reflecting any global.json):
 Version:   6.0.401
 Commit:    0906eae6f8

Runtime Environment:
 OS Name:     Windows
 OS Version:  10.0.22000
 OS Platform: Windows
 RID:         win10-x64
 Base Path:   C:\Program Files\dotnet\sdk\6.0.401\

global.json file:
  Not found

Host:
  Version:      6.0.9
  Architecture: x64
  Commit:       163a63591c

.NET SDKs installed:
  2.1.202 [C:\Program Files\dotnet\sdk]
  5.0.103 [C:\Program Files\dotnet\sdk]
  5.0.200 [C:\Program Files\dotnet\sdk]
  5.0.400 [C:\Program Files\dotnet\sdk]
  6.0.401 [C:\Program Files\dotnet\sdk]

.NET runtimes installed:
  Microsoft.AspNetCore.All 2.1.29 [C:\Program Files\dotnet\shared\Microsoft.AspNetCore.All]
  Microsoft.AspNetCore.App 2.1.29 [C:\Program Files\dotnet\shared\Microsoft.AspNetCore.App]
  Microsoft.AspNetCore.App 3.1.18 [C:\Program Files\dotnet\shared\Microsoft.AspNetCore.App]
  Microsoft.AspNetCore.App 5.0.3 [C:\Program Files\dotnet\shared\Microsoft.AspNetCore.App]
  Microsoft.AspNetCore.App 5.0.9 [C:\Program Files\dotnet\shared\Microsoft.AspNetCore.App]
  Microsoft.AspNetCore.App 6.0.9 [C:\Program Files\dotnet\shared\Microsoft.AspNetCore.App]
  Microsoft.NETCore.App 2.0.9 [C:\Program Files\dotnet\shared\Microsoft.NETCore.App]
  Microsoft.NETCore.App 2.1.29 [C:\Program Files\dotnet\shared\Microsoft.NETCore.App]
  Microsoft.NETCore.App 3.1.18 [C:\Program Files\dotnet\shared\Microsoft.NETCore.App]
  Microsoft.NETCore.App 5.0.3 [C:\Program Files\dotnet\shared\Microsoft.NETCore.App]
  Microsoft.NETCore.App 5.0.9 [C:\Program Files\dotnet\shared\Microsoft.NETCore.App]
  Microsoft.NETCore.App 6.0.9 [C:\Program Files\dotnet\shared\Microsoft.NETCore.App]
  Microsoft.WindowsDesktop.App 3.1.18 [C:\Program Files\dotnet\shared\Microsoft.WindowsDesktop.App]
  Microsoft.WindowsDesktop.App 5.0.3 [C:\Program Files\dotnet\shared\Microsoft.WindowsDesktop.App]
  Microsoft.WindowsDesktop.App 5.0.9 [C:\Program Files\dotnet\shared\Microsoft.WindowsDesktop.App]
  Microsoft.WindowsDesktop.App 6.0.9 [C:\Program Files\dotnet\shared\Microsoft.WindowsDesktop.App]

Download .NET:
  https://aka.ms/dotnet-download

Learn about .NET Runtimes and SDKs:
```

LT2. Install and setup Visual Code

LT2.1. Download Visual Code from the URL listed above



Visual Studio Code
Editing evolved

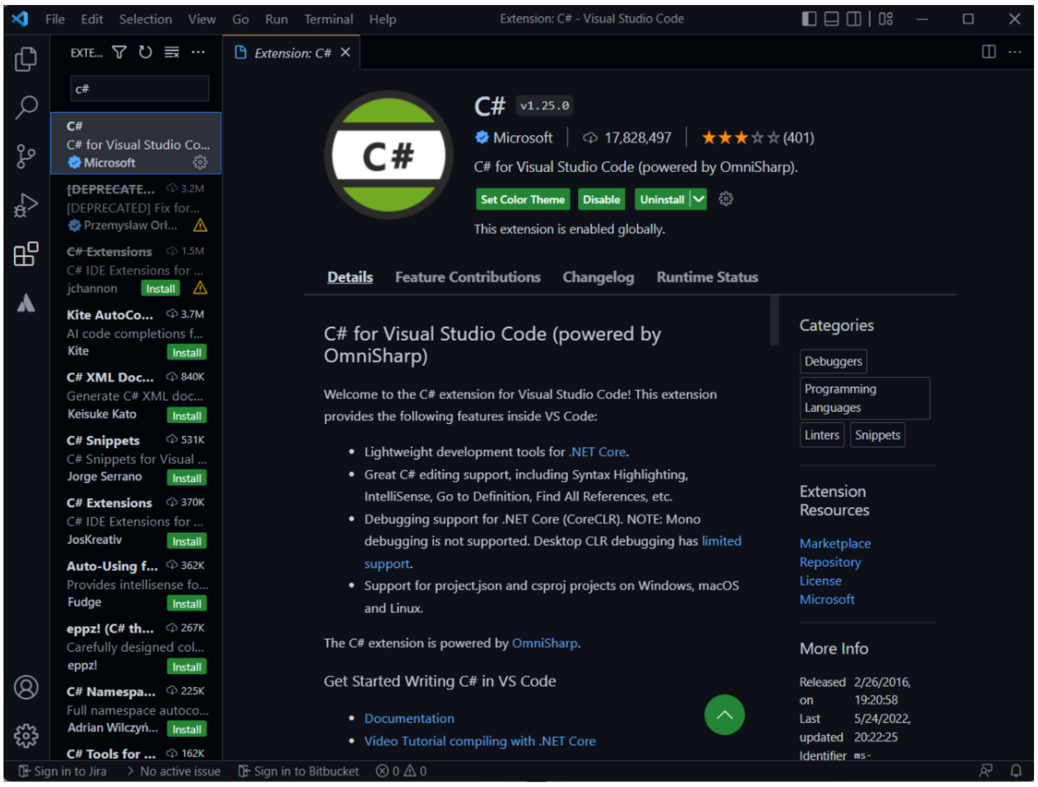
Start
New File...
Open File...
Open Folder...
Clone Git Repository...

Recent
mmocr D:\
simple-ocr-opencv-master D:\Computer Science\Semester 0...
ocular C:\Users\noor_
mmocr C:\Users\noor_
OCR.space D:\Computer Science\Semester 05\Software Engi...
More...

Walkthroughs
Get Started with VS Code
Discover the best customizations to make VS Code yours.
Learn the Fundamentals
Jump right into VS Code and get an overview of the must-have features.
Boost your Productivity
Get started with Python development Updated
Get started with Jupyter Notebooks

✓ Show welcome page on startup

LT2.2. Install the ‘C# Extension’ from the Visual Code marketplace


File Edit Selection View Go Run Terminal Help
Extension: C# - Visual Studio Code
EXTE...
c#
Extension: C#

C#
C# for Visual Studio Co...
Microsoft

[DEPRECATE... 3.2M
[DEPRECATED] Fix for...
Przemysław Orl...

C# Extensions 1.5M
C# IDE Extensions for ...
jchannon Install

Kite AutoCo... 3.7M
AI code completions f...
Kite Install

C# XML Doc... 840K
Generate C# XML doc...
Keisuke Kato Install

C# Snippets 531K
C# Snippets for Visual ...
Jorge Serrano Install

C# Extensions 370K
C# IDE Extensions for ...
JosKreativ Install

Auto-Using f... 362K
Provides intellisense fo...
Fudge Install

eppz! (C# th... 267K
Carefully designed col...
eppz! Install

C# Namespa... 225K
Full namespace autoco...
Adrian Wilczyń... Install

C# Tools for ... 162K

C# v1.25.0
Microsoft | 17,828,497 | ★★★☆☆ (401)
C# for Visual Studio Code (powered by OmniSharp).
Set Color Theme Disable Uninstall
This extension is enabled globally.

Details Feature Contributions Changelog Runtime Status

C# for Visual Studio Code (powered by OmniSharp)

Welcome to the C# extension for Visual Studio Code! This extension provides the following features inside VS Code:

- Lightweight development tools for .NET Core.
- Great C# editing support, including Syntax Highlighting, IntelliSense, Go to Definition, Find All References, etc.
- Debugging support for .NET Core (CoreCLR). NOTE: Mono debugging is not supported. Desktop CLR debugging has limited support.
- Support for project.json and csproj projects on Windows, macOS and Linux.

The C# extension is powered by OmniSharp.

Get Started Writing C# in VS Code

- Documentation
- Video Tutorial compiling with .NET Core

Categories
Debuggers
Programming Languages
Linters Snippets

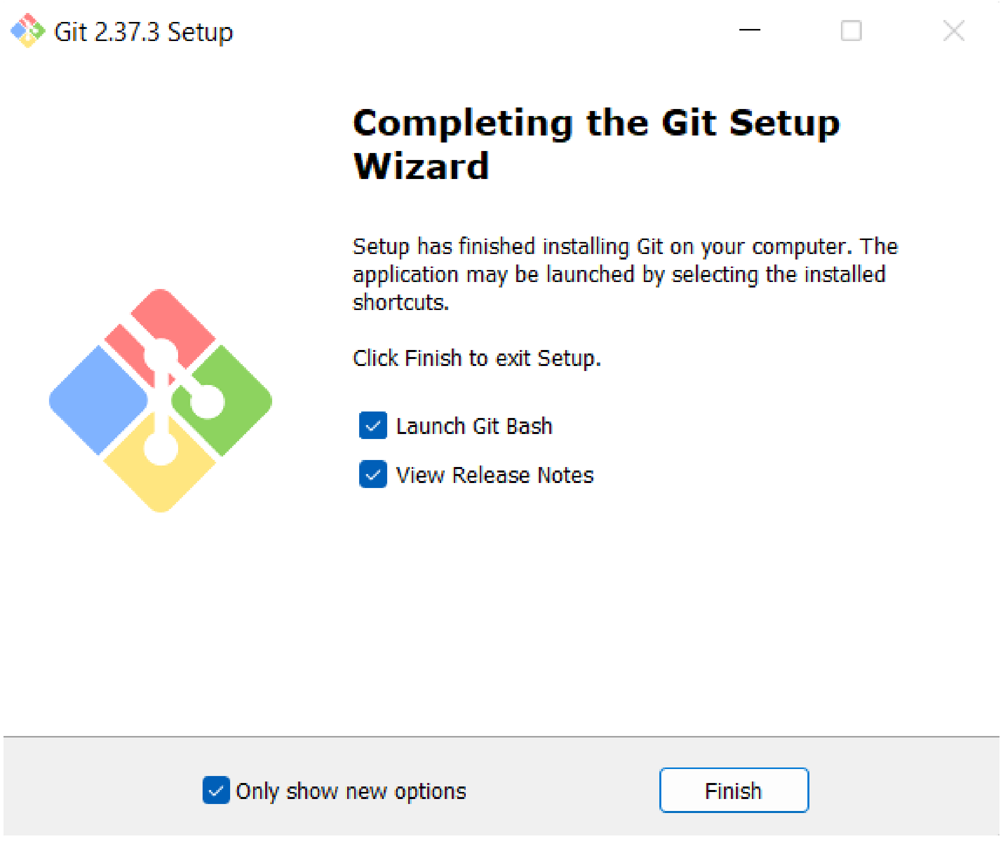
Extension Resources
Marketplace
Repository
License
Microsoft

More Info
Released on 2/26/2016, 19:20:58
Last updated 5/24/2022, 20:22:25
Identifier ms-

Sign in to Jira > No active issue Sign in to Bitbucket ⊗ 0 ⚠ 0

LT3. Build and run .NET application

LT3.1. Download and install git on your system. The download link can be found from the URL listed above


Git 2.37.3 Setup

Completing the Git Setup Wizard

Setup has finished installing Git on your computer. The application may be launched by selecting the installed shortcuts.

Click Finish to exit Setup.

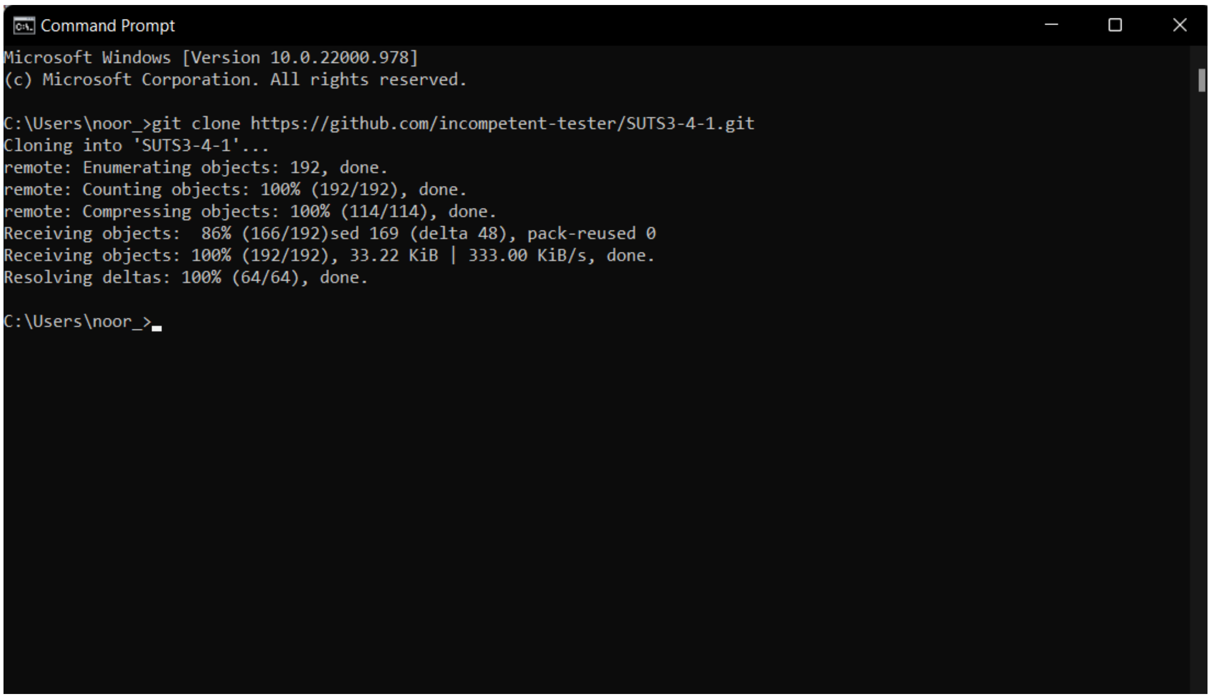
[✓] Launch Git Bash

[✓] View Release Notes

[✓] Only show new options

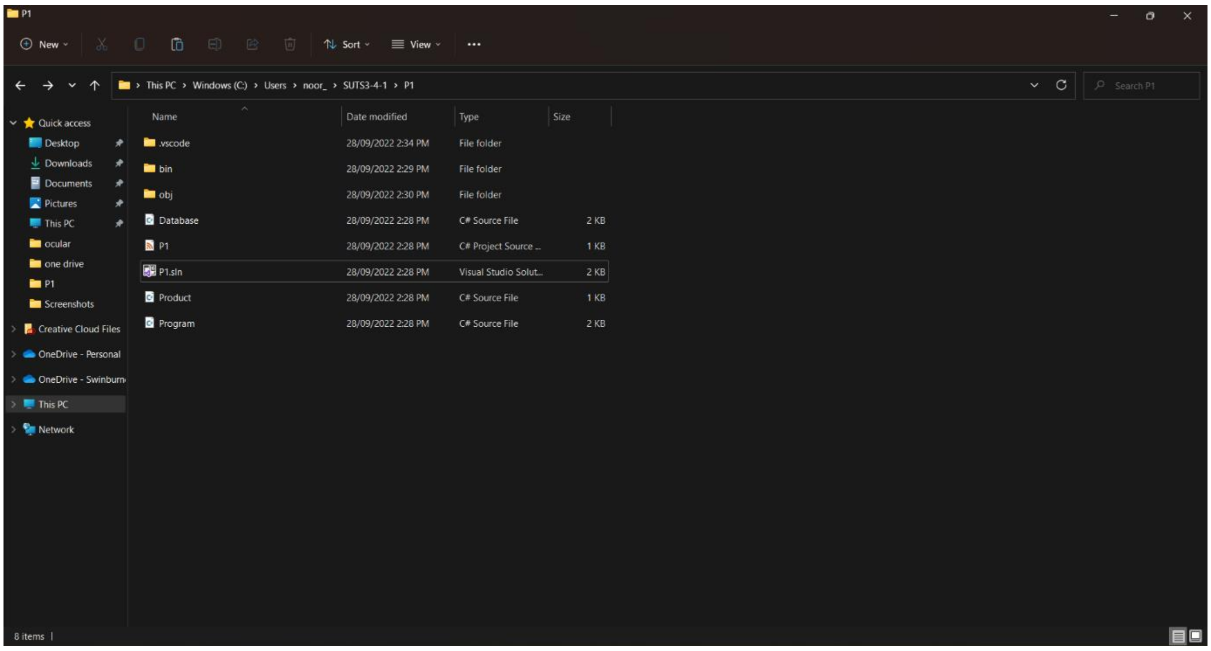
Finish

LT3.2. Git clone the course lab work via command



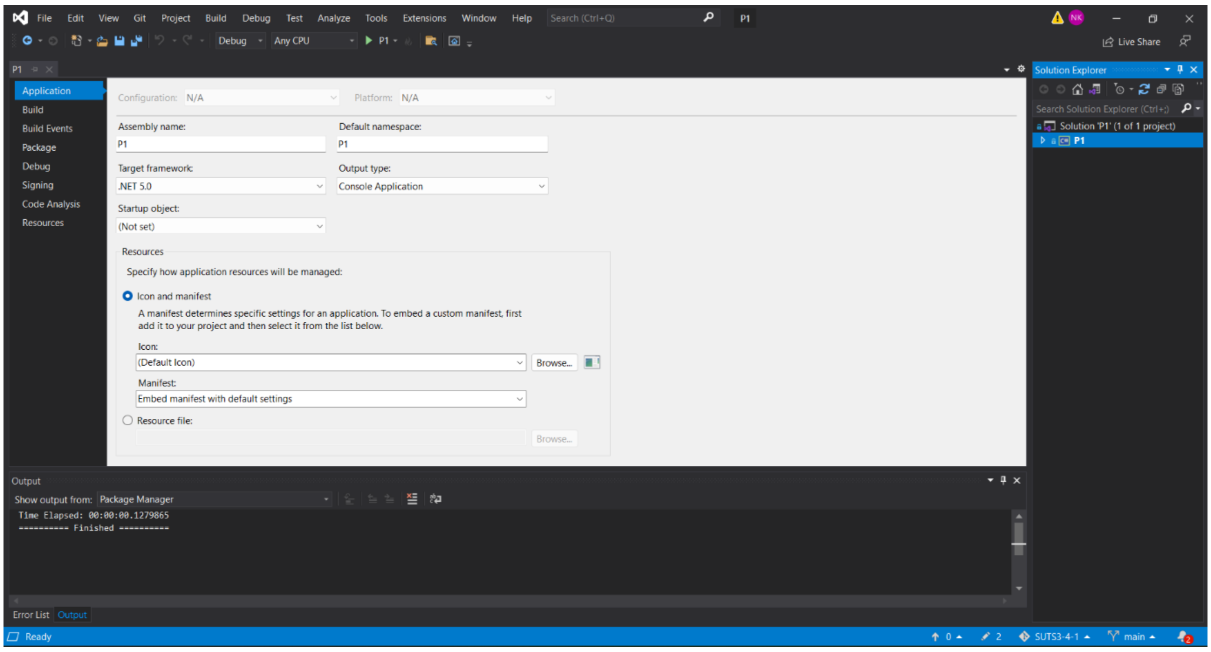
```
Microsoft Windows [Version 10.0.22000.978]
(c) Microsoft Corporation. All rights reserved.

C:\Users\noor_>git clone https://github.com/incompetent-tester/SUTS3-4-1.git
Cloning into 'SUTS3-4-1'...
remote: Enumerating objects: 192, done.
remote: Counting objects: 100% (192/192), done.
remote: Compressing objects: 100% (114/114), done.
Receiving objects:  86% (166/192)sed 169 (delta 48), pack-reused 0
Receiving objects: 100% (192/192), 33.22 KiB | 333.00 KiB/s, done.
Resolving deltas: 100% (64/64), done.

C:\Users\noor_>
```

LT3.3. The files for this lab can be found in directory './P1'



Name	Date modified	Type	Size
.vscode	28/09/2022 2:34 PM	File folder	
bin	28/09/2022 2:29 PM	File folder	
obj	28/09/2022 2:30 PM	File folder	
Database	28/09/2022 2:28 PM	C# Source File	2 KB
P1	28/09/2022 2:28 PM	C# Project Source ...	1 KB
P1.sln	28/09/2022 2:28 PM	Visual Studio Solut...	2 KB
Product	28/09/2022 2:28 PM	C# Source File	1 KB
Program	28/09/2022 2:28 PM	C# Source File	2 KB

LT3.4. Run the .NET application



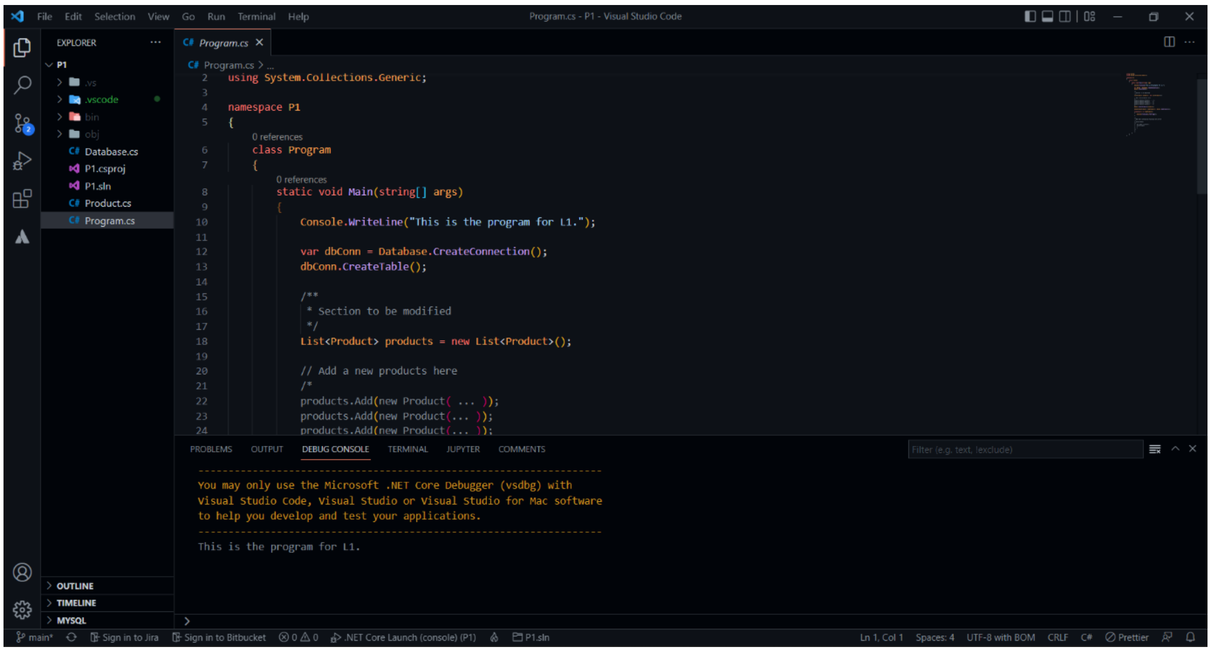
Configuration: N/A   Platform: N/A

Assembly name: P1
Default namespace: P1
Target framework: .NET 5.0
Output type: Console Application
Startup object: (Not set)

Resources
Specify how application resources will be managed:
● Icon and manifest
A manifest determines specific settings for an application. To embed a custom manifest, first add it to your project and then select it from the list below.
Icon: (Default Icon)
Manifest: Embed manifest with default settings
○ Resource file:

Output
Show output from: Package Manager

```
Time Elapsed: 00:00:00.1279865
========== Finished ==========
```



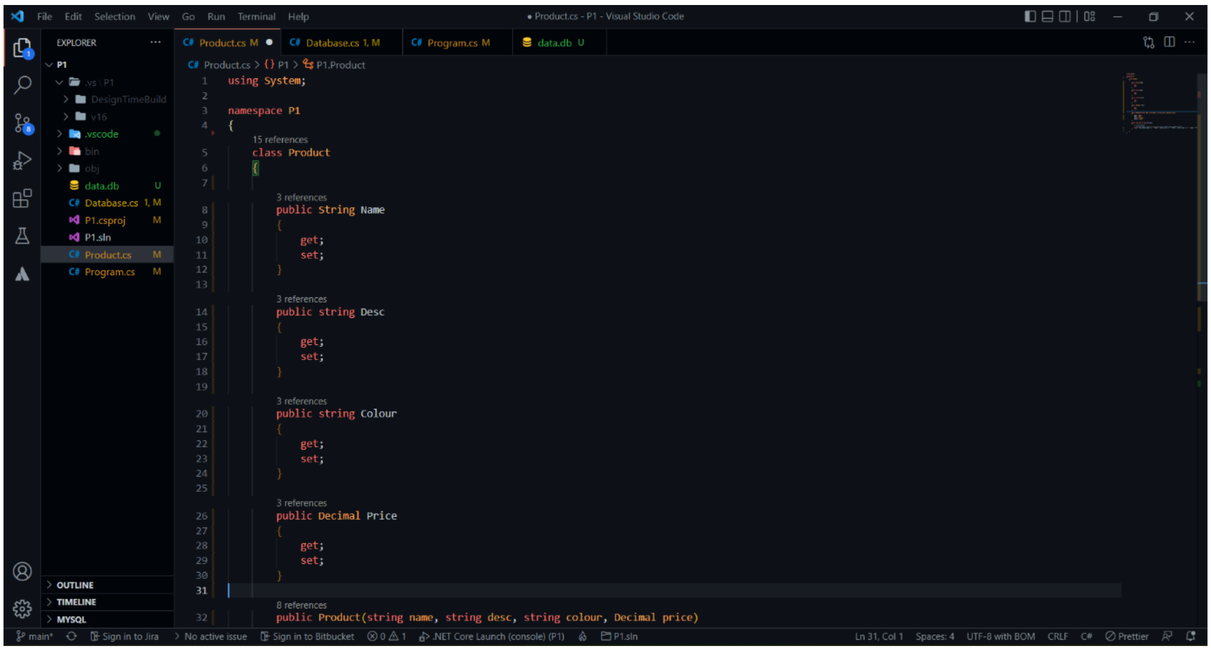
Program.cs - P1 - Visual Studio Code

```
using System.Collections.Generic;

namespace P1
{
    0 references
    class Program
    {
        0 references
        static void Main(string[] args)
        {
            Console.WriteLine("This is the program for L1.");

            var dbConn = Database.CreateConnection();
            dbConn.CreateTable();

            /**
             * Section to be modified
             */
            List<Product> products = new List<Product>();

            // Add a new products here
            /*
            products.Add(new Product( ... ));
            products.Add(new Product(... ));
            products.Add(new Product(... ));
```

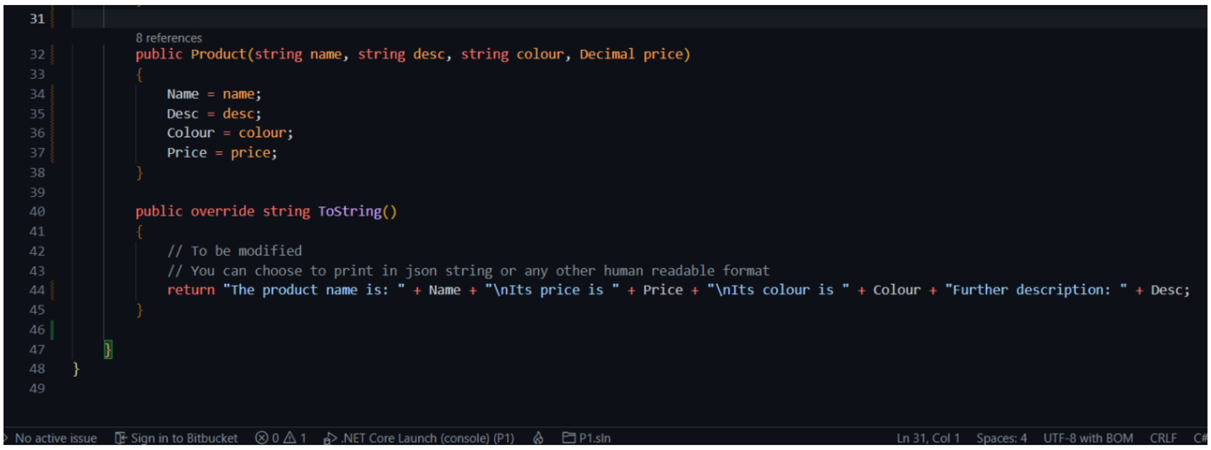
```
-------------------------------------------------------------------
You may only use the Microsoft .NET Core Debugger (vsdbg) with
Visual Studio Code, Visual Studio or Visual Studio for Mac software
to help you develop and test your applications.
-------------------------------------------------------------------
This is the program for L1.
```

LT4. Modify and develop the required .NET package

LT4.1. Edit the ‘Product.cs’ source code to include properties

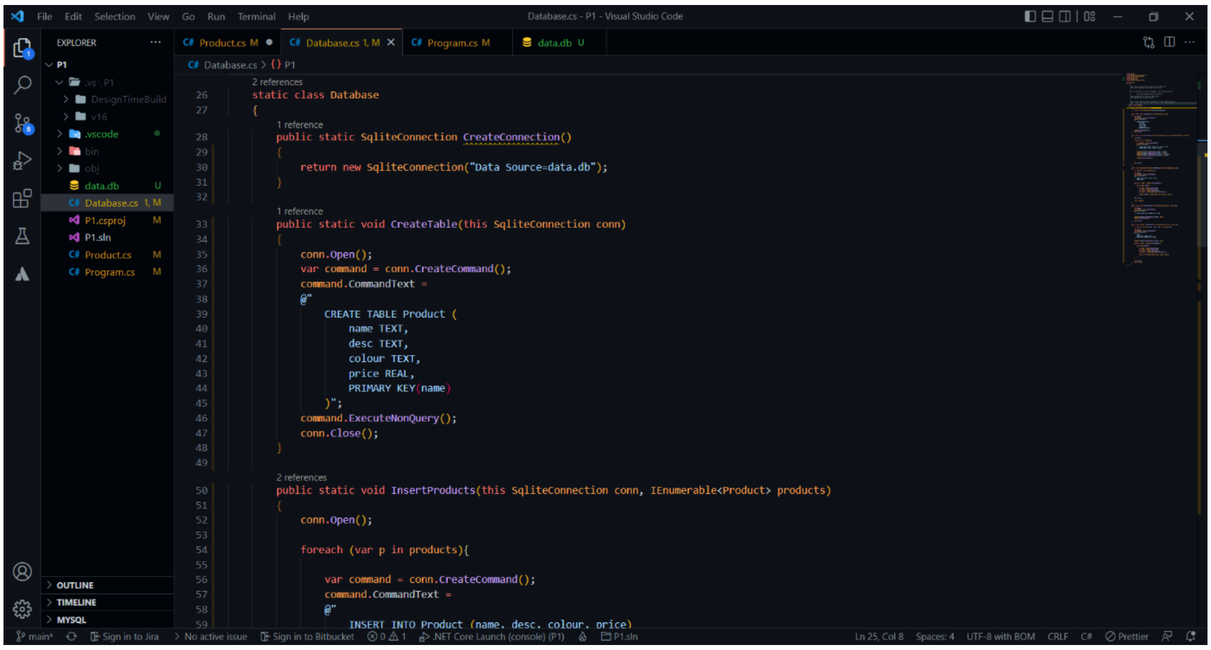


```csharp
using System;

namespace P1
{
    15 references
    class Product
    {

        3 references
        public String Name
        {
            get;
            set;
        }

        3 references
        public string Desc
        {
            get;
            set;
        }

        3 references
        public string Colour
        {
            get;
            set;
        }

        3 references
        public Decimal Price
        {
            get;
            set;
        }

        8 references
        public Product(string name, string desc, string colour, Decimal price)
```

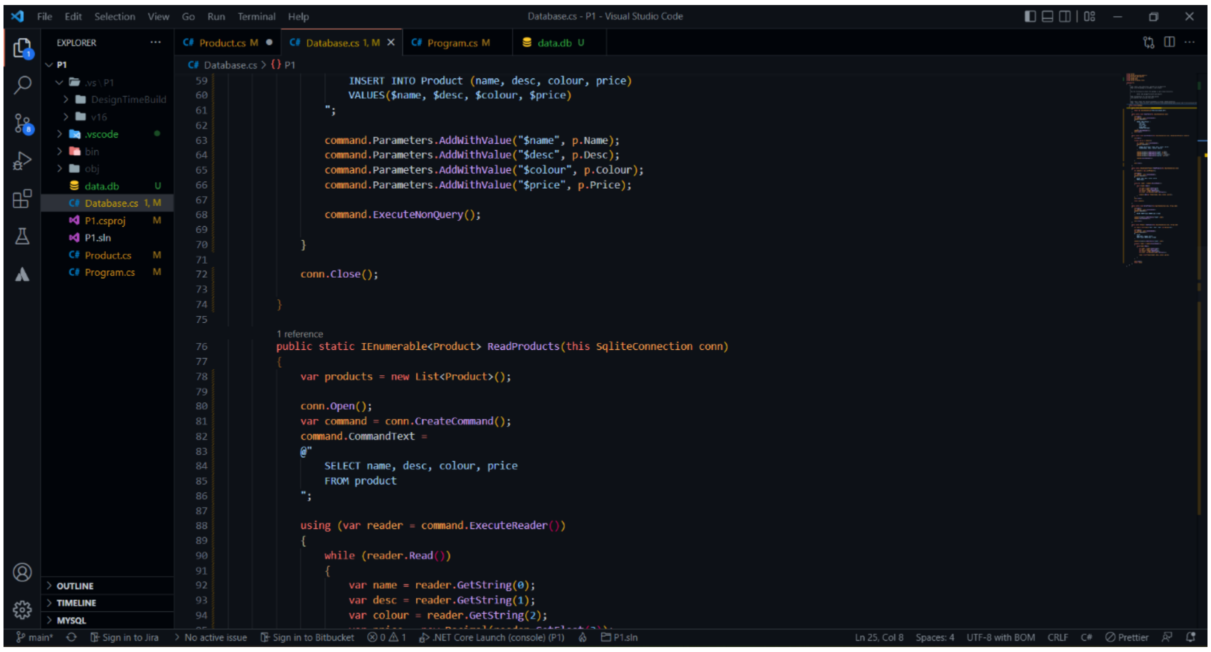


```csharp

        8 references
        public Product(string name, string desc, string colour, Decimal price)
        {
            Name = name;
            Desc = desc;
            Colour = colour;
            Price = price;
        }

        public override string ToString()
        {
            // To be modified
            // You can choose to print in json string or any other human readable format
            return "The product name is: " + Name + "\nIts price is " + Price + "\nIts colour is " + Colour + "Further description: " + Desc;
        }

    }
}

```

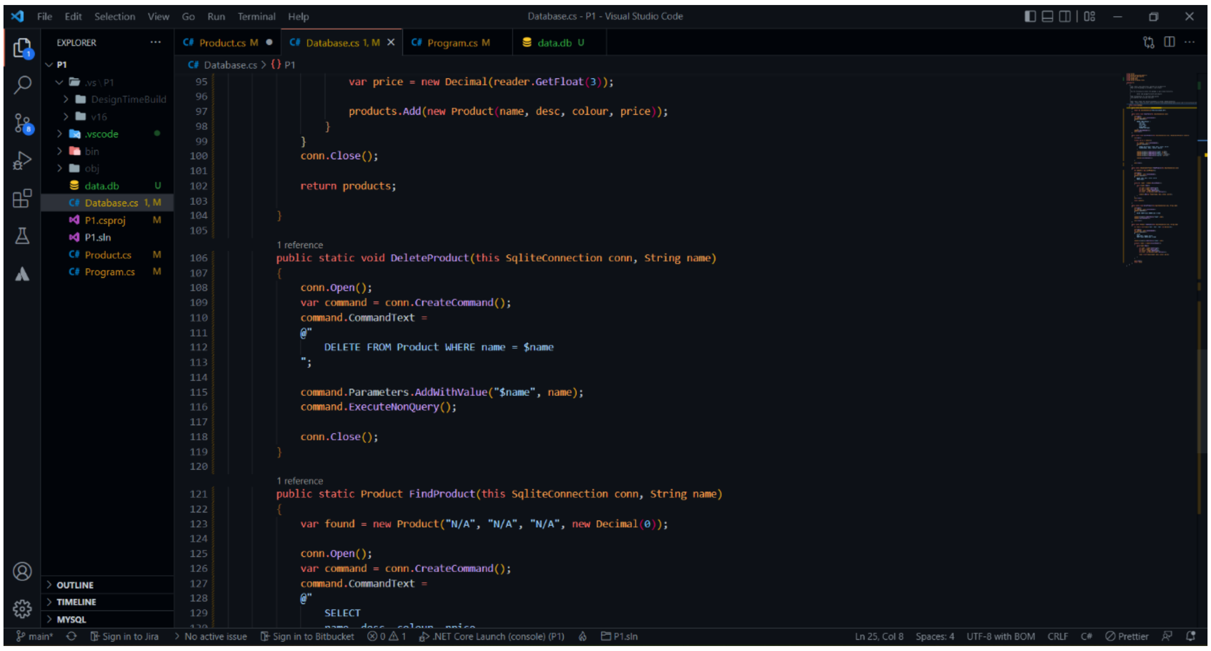
LT4.2. Edit the 'Database.cs' and complete the required functions. Add the relevant code to 'CreateConnection', 'CreateTable', 'InsertProducts' and 'ReadProducts' to have a functioning SQLite Provider.



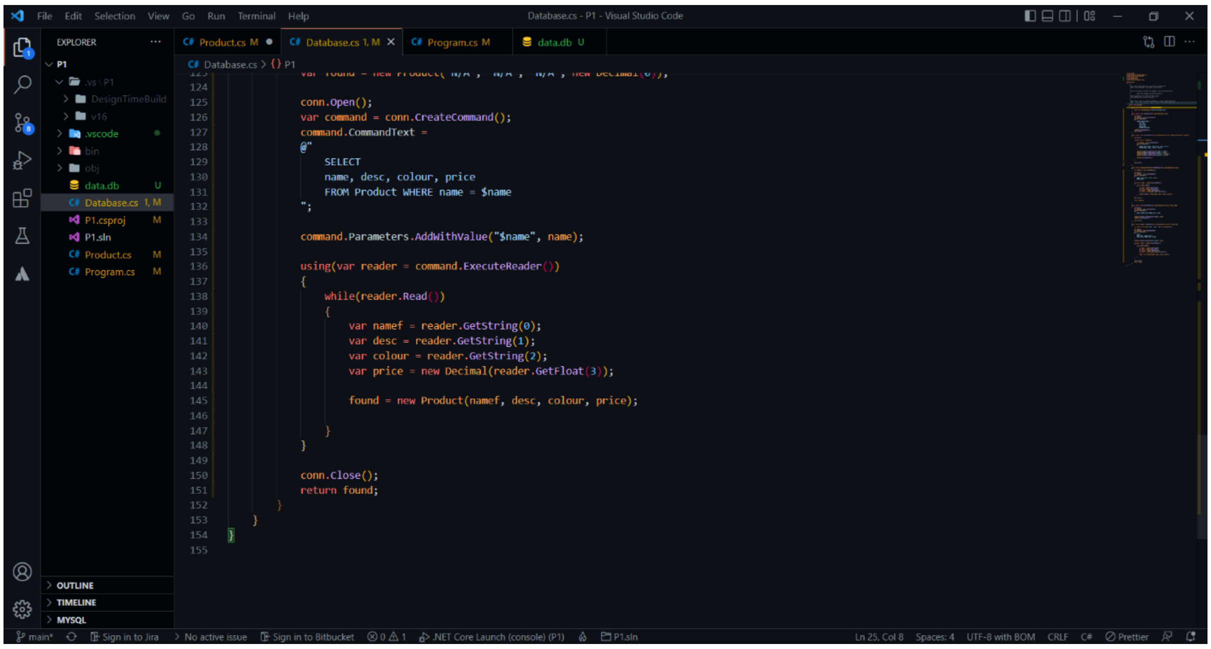
```
    static class Database
    {
        public static SqliteConnection CreateConnection()
        {
            return new SqliteConnection("Data Source=data.db");
        }

        public static void CreateTable(this SqliteConnection conn)
        {
            conn.Open();
            var command = conn.CreateCommand();
            command.CommandText =
            @"
                CREATE TABLE Product (
                    name TEXT,
                    desc TEXT,
                    colour TEXT,
                    price REAL,
                    PRIMARY KEY(name)
                )";
            command.ExecuteNonQuery();
            conn.Close();
        }

        public static void InsertProducts(this SqliteConnection conn, IEnumerable<Product> products)
        {
            conn.Open();

            foreach (var p in products){

                var command = conn.CreateCommand();
                command.CommandText =
                @"
                    INSERT INTO Product (name, desc, colour, price)
```



```
                    INSERT INTO Product (name, desc, colour, price)
                    VALUES($name, $desc, $colour, $price)
                ";

                command.Parameters.AddWithValue("$name", p.Name);
                command.Parameters.AddWithValue("$desc", p.Desc);
                command.Parameters.AddWithValue("$colour", p.Colour);
                command.Parameters.AddWithValue("$price", p.Price);

                command.ExecuteNonQuery();

            }

            conn.Close();

        }

        public static IEnumerable<Product> ReadProducts(this SqliteConnection conn)
        {
            var products = new List<Product>();

            conn.Open();
            var command = conn.CreateCommand();
            command.CommandText =
            @"
                SELECT name, desc, colour, price
                FROM product
            ";

            using (var reader = command.ExecuteReader())
            {
                while (reader.Read())
                {
                    var name = reader.GetString(0);
                    var desc = reader.GetString(1);
                    var colour = reader.GetString(2);
```

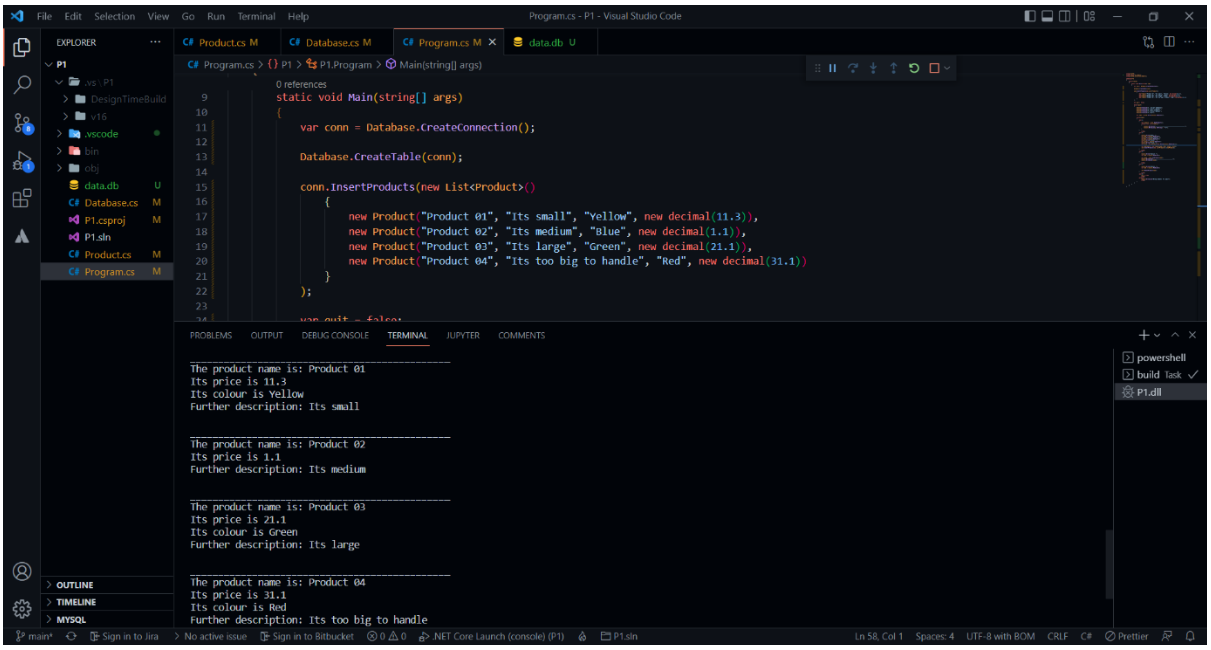


```
                    var price = new Decimal(reader.GetFloat(3));

                    products.Add(new Product(name, desc, colour, price));
                }
            }
            conn.Close();

            return products;

        }

        1 reference
        public static void DeleteProduct(this SqliteConnection conn, String name)
        {
            conn.Open();
            var command = conn.CreateCommand();
            command.CommandText =
            @"
                DELETE FROM Product WHERE name = $name
            ";

            command.Parameters.AddWithValue("$name", name);
            command.ExecuteNonQuery();

            conn.Close();
        }

        1 reference
        public static Product FindProduct(this SqliteConnection conn, String name)
        {
            var found = new Product("N/A", "N/A", "N/A", new Decimal(0));

            conn.Open();
            var command = conn.CreateCommand();
            command.CommandText =
            @"
                SELECT
```



```

            conn.Open();
            var command = conn.CreateCommand();
            command.CommandText =
            @"
                SELECT
                name, desc, colour, price
                FROM Product WHERE name = $name
            ";

            command.Parameters.AddWithValue("$name", name);

            using(var reader = command.ExecuteReader())
            {
                while(reader.Read())
                {
                    var namef = reader.GetString(0);
                    var desc = reader.GetString(1);
                    var colour = reader.GetString(2);
                    var price = new Decimal(reader.GetFloat(3));

                    found = new Product(namef, desc, colour, price);

                }
            }

            conn.Close();
            return found;
        }
    }
}

```

LT4.3. Run the application. It should list all the products in a human readable form.



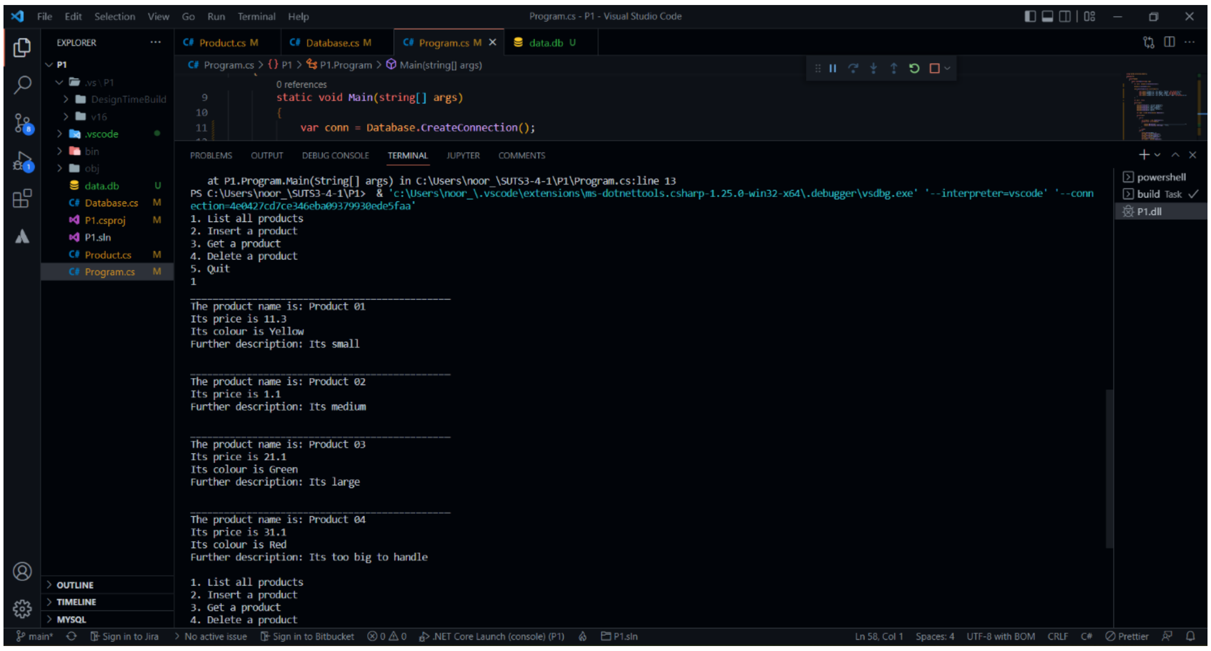
```csharp
static void Main(string[] args)
{
    var conn = Database.CreateConnection();

    Database.CreateTable(conn);

    conn.InsertProducts(new List<Product>()
        {
            new Product("Product 01", "Its small", "Yellow", new decimal(11.3)),
            new Product("Product 02", "Its medium", "Blue", new decimal(1.1)),
            new Product("Product 03", "Its large", "Green", new decimal(21.1)),
            new Product("Product 04", "Its too big to handle", "Red", new decimal(31.1))
        }
    );
```

```
----------------------------------------
The product name is: Product 01
Its price is 11.3
Its colour is Yellow
Further description: Its small


----------------------------------------
The product name is: Product 02
Its price is 1.1
Further description: Its medium


----------------------------------------
The product name is: Product 03
Its price is 21.1
Its colour is Green
Further description: Its large


----------------------------------------
The product name is: Product 04
Its price is 31.1
Its colour is Red
Further description: Its too big to handle
```

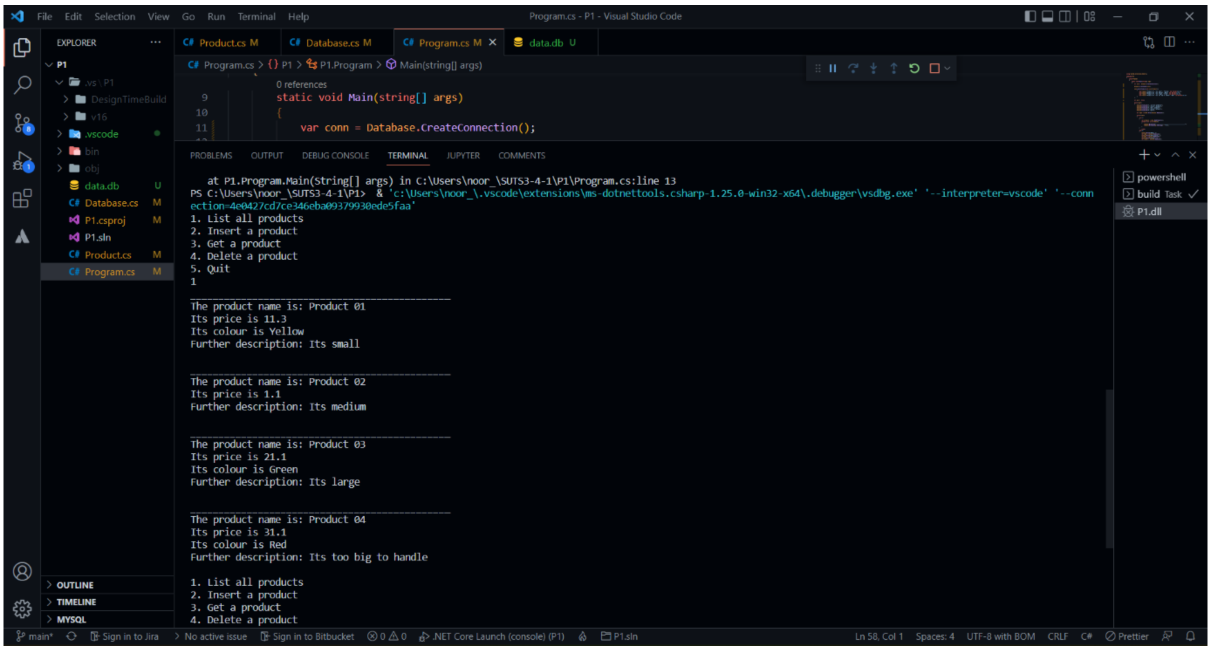
LT4.4. Modify the application again.

This time the program should be able to accept user input. The user should be able to 'Add new product', 'Print all product', 'Get a product', 'Delete a product', 'Quit program'.

This task requires you to edit the 'Product' and possibly 'Database' class. Figure out what you need to add to be able to 'Get a product' or 'Delete a product'.

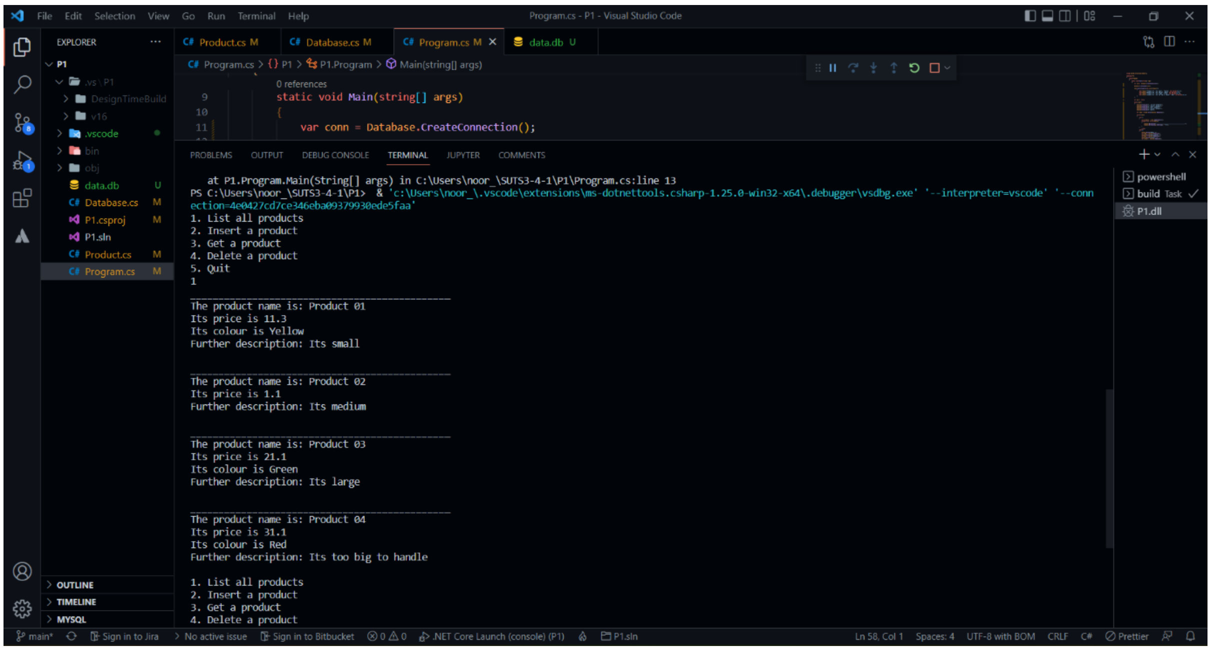


```
   at P1.Program.Main(String[] args) in C:\Users\noor_\SUTS3-4-1\P1\Program.cs:line 13
PS C:\Users\noor_\SUTS3-4-1\P1>  & 'c:\Users\noor_\.vscode\extensions\ms-dotnettools.csharp-1.25.0-win32-x64\.debugger\vsdbg.exe' '--interpreter=vscode' '--conn
ection=4e0427cd7ce346eba09379930ede5faa'
1. List all products
2. Insert a product
3. Get a product
4. Delete a product
5. Quit
1

----------------------------------------
The product name is: Product 01
Its price is 11.3
Its colour is Yellow
Further description: Its small


----------------------------------------
The product name is: Product 02
Its price is 1.1
Further description: Its medium


----------------------------------------
The product name is: Product 03
Its price is 21.1
Its colour is Green
Further description: Its large


----------------------------------------
The product name is: Product 04
Its price is 31.1
Its colour is Red
Further description: Its too big to handle

1. List all products
2. Insert a product
3. Get a product
4. Delete a product
```



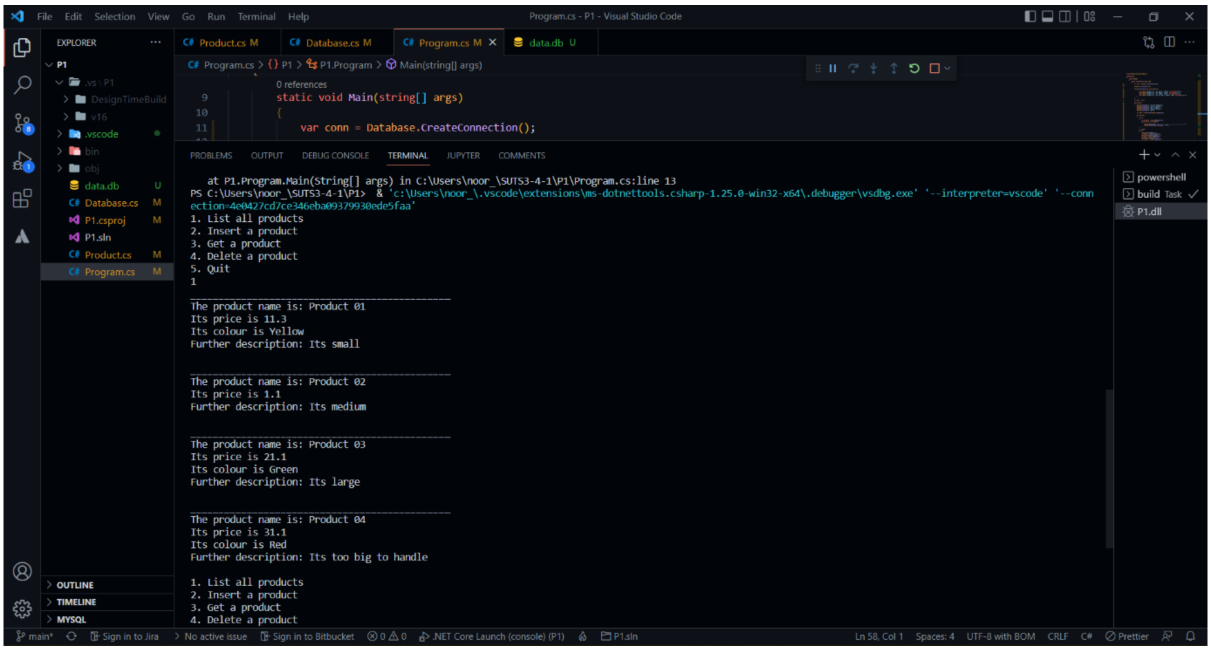
```
static void Main(string[] args)
{
    var conn = Database.CreateConnection();
```

```
   at P1.Program.Main(String[] args) in C:\Users\noor_\SUTS3-4-1\P1\Program.cs:line 13
PS C:\Users\noor_\SUTS3-4-1\P1>  & 'c:\Users\noor_\.vscode\extensions\ms-dotnettools.csharp-1.25.0-win32-x64\.debugger\vsdbg.exe' '--interpreter=vscode' '--conn
ection=4e0427cd7ce346eba09379930ede5faa'
1. List all products
2. Insert a product
3. Get a product
4. Delete a product
5. Quit
1

__________________________________________
The product name is: Product 01
Its price is 11.3
Its colour is Yellow
Further description: Its small


__________________________________________
The product name is: Product 02
Its price is 1.1
Further description: Its medium


__________________________________________
The product name is: Product 03
Its price is 21.1
Its colour is Green
Further description: Its large


__________________________________________
The product name is: Product 04
Its price is 31.1
Its colour is Red
Further description: Its too big to handle

1. List all products
2. Insert a product
3. Get a product
4. Delete a product
```



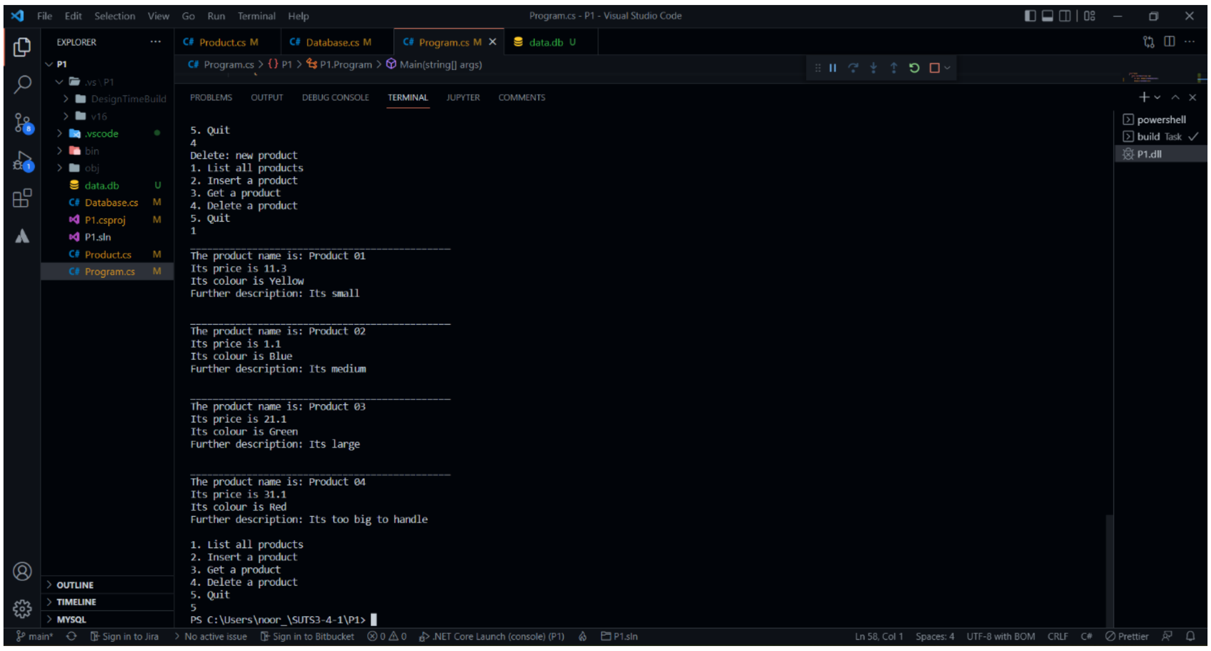
```
static void Main(string[] args)
{
    var conn = Database.CreateConnection();
```

```
   at P1.Program.Main(String[] args) in C:\Users\noor_\SUTS3-4-1\P1\Program.cs:line 13
PS C:\Users\noor_\SUTS3-4-1\P1>  & 'c:\Users\noor_\.vscode\extensions\ms-dotnettools.csharp-1.25.0-win32-x64\.debugger\vsdbg.exe' '--interpreter=vscode' '--conn
ection=4e0427cd7ce346eba09379930ede5faa'
1. List all products
2. Insert a product
3. Get a product
4. Delete a product
5. Quit
1

__________________________________________
The product name is: Product 01
Its price is 11.3
Its colour is Yellow
Further description: Its small


__________________________________________
The product name is: Product 02
Its price is 1.1
Further description: Its medium


__________________________________________
The product name is: Product 03
Its price is 21.1
Its colour is Green
Further description: Its large


__________________________________________
The product name is: Product 04
Its price is 31.1
Its colour is Red
Further description: Its too big to handle

1. List all products
2. Insert a product
3. Get a product
4. Delete a product
```

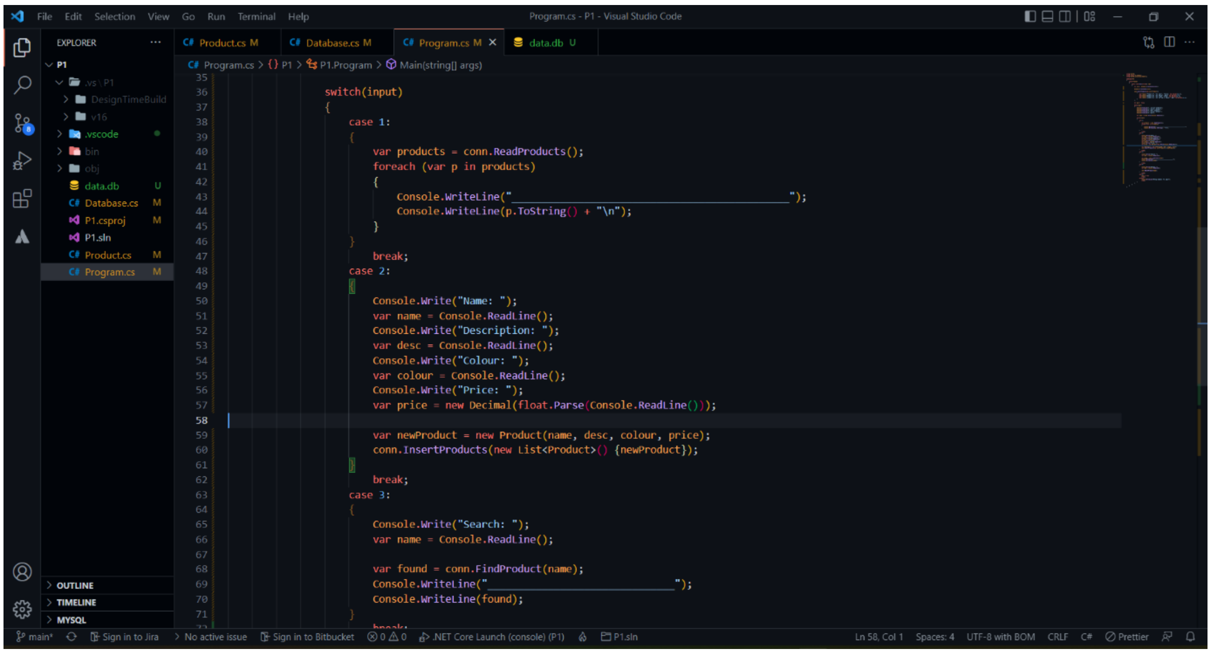


```csharp
static void Main(string[] args)
{
    var conn = Database.CreateConnection();
```

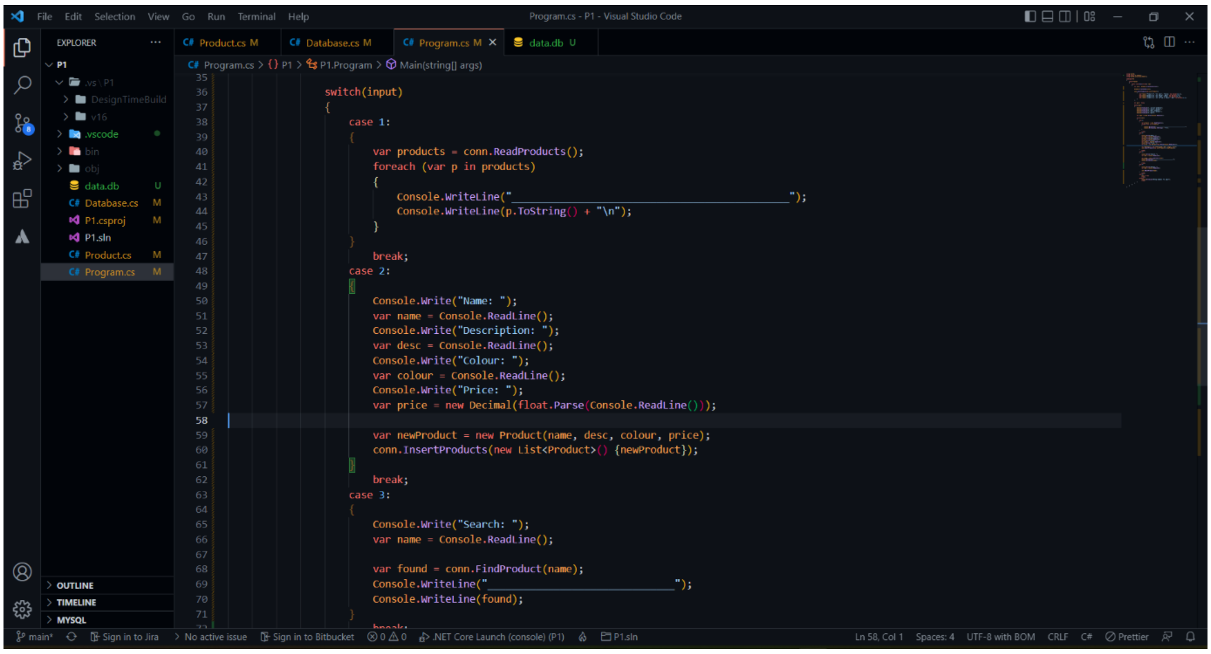
```
   at P1.Program.Main(String[] args) in C:\Users\noor_\SUTS3-4-1\P1\Program.cs:line 13
PS C:\Users\noor_\SUTS3-4-1\P1>  & 'c:\Users\noor_\.vscode\extensions\ms-dotnettools.csharp-1.25.0-win32-x64\.debugger\vsdbg.exe' '--interpreter=vscode' '--conn
ection=4e0427cd7ce346eba09379930ede5faa'
1. List all products
2. Insert a product
3. Get a product
4. Delete a product
5. Quit
1
______________________________________________
The product name is: Product 01
Its price is 11.3
Its colour is Yellow
Further description: Its small

______________________________________________
The product name is: Product 02
Its price is 1.1
Further description: Its medium

______________________________________________
The product name is: Product 03
Its price is 21.1
Its colour is Green
Further description: Its large

______________________________________________
The product name is: Product 04
Its price is 31.1
Its colour is Red
Further description: Its too big to handle

1. List all products
2. Insert a product
3. Get a product
4. Delete a product
```



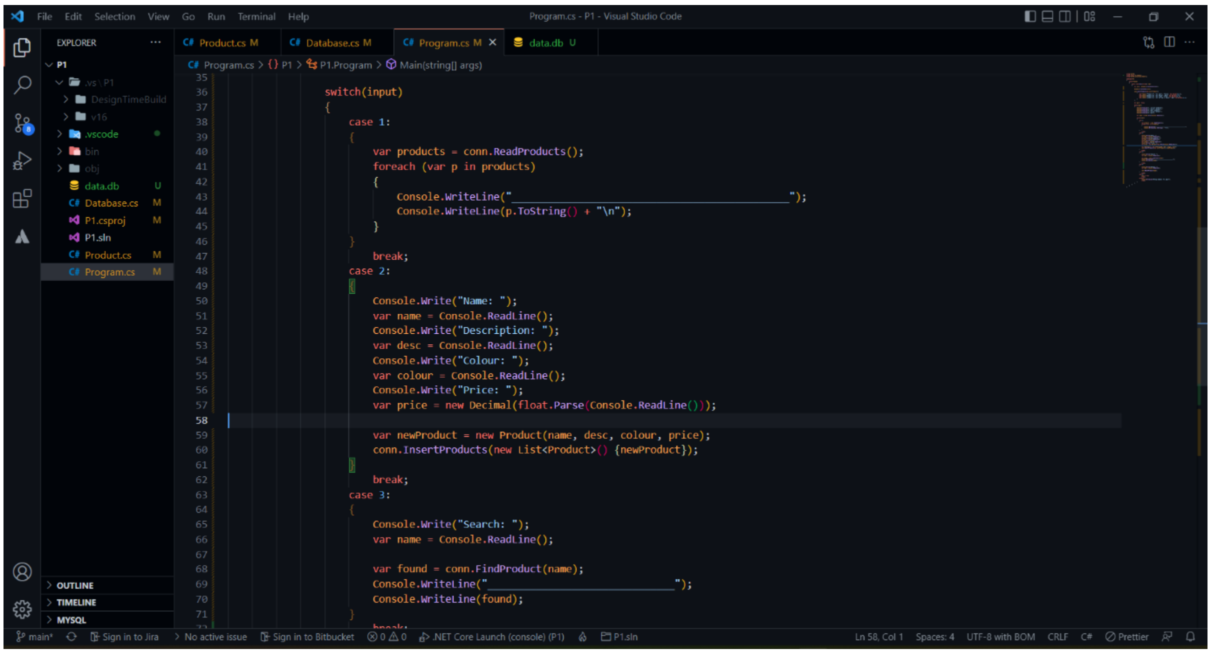
```
5. Quit
4
Delete: new product
1. List all products
2. Insert a product
3. Get a product
4. Delete a product
5. Quit
1
______________________________________________
The product name is: Product 01
Its price is 11.3
Its colour is Yellow
Further description: Its small

______________________________________________
The product name is: Product 02
Its price is 1.1
Its colour is Blue
Further description: Its medium

______________________________________________
The product name is: Product 03
Its price is 21.1
Its colour is Green
Further description: Its large

______________________________________________
The product name is: Product 04
Its price is 31.1
Its colour is Red
Further description: Its too big to handle

1. List all products
2. Insert a product
3. Get a product
4. Delete a product
5. Quit
5
PS C:\Users\noor_\SUTS3-4-1\P1>
```



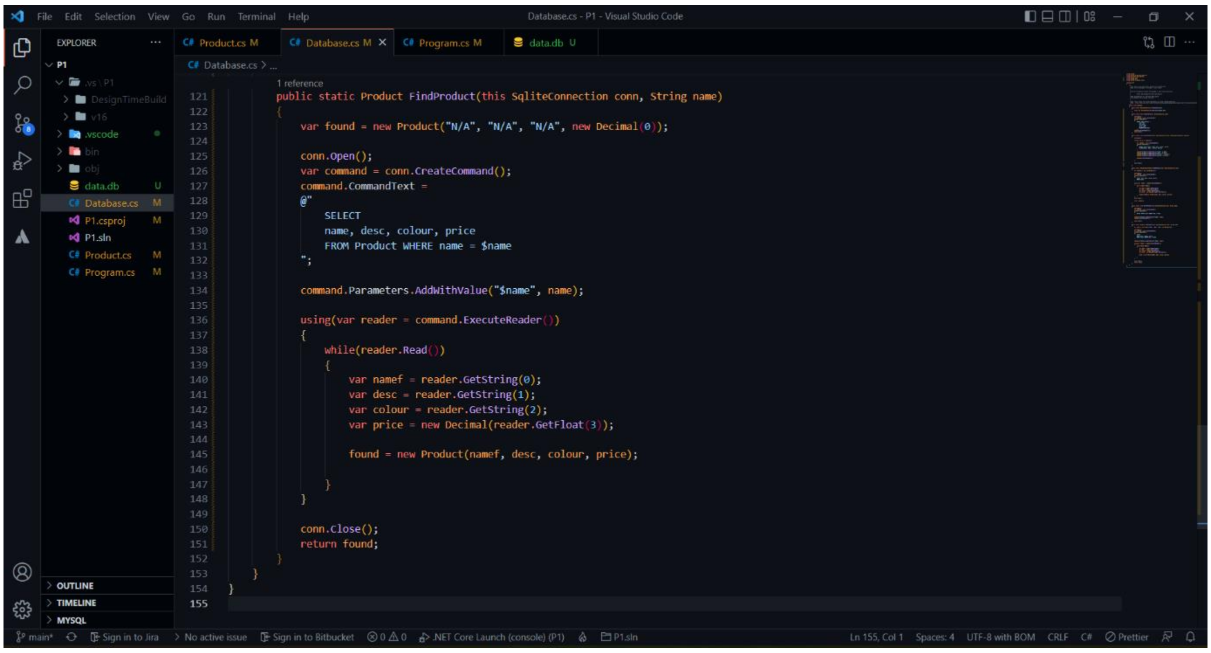
```
                switch(input)
                {
                    case 1:
                    {
                        var products = conn.ReadProducts();
                        foreach (var p in products)
                        {
                            Console.WriteLine("______________________________________________");
                            Console.WriteLine(p.ToString() + "\n");
                        }
                    }
                        break;
                    case 2:
                    {
                        Console.Write("Name: ");
                        var name = Console.ReadLine();
                        Console.Write("Description: ");
                        var desc = Console.ReadLine();
                        Console.Write("Colour: ");
                        var colour = Console.ReadLine();
                        Console.Write("Price: ");
                        var price = new Decimal(float.Parse(Console.ReadLine()));

                        var newProduct = new Product(name, desc, colour, price);
                        conn.InsertProducts(new List<Product>() {newProduct});
                    }
                        break;
                    case 3:
                    {
                        Console.Write("Search: ");
                        var name = Console.ReadLine();

                        var found = conn.FindProduct(name);
                        Console.WriteLine("_______________________________");
                        Console.WriteLine(found);
                    }
```



```
                switch(input)
                {
                    case 1:
                    {
                        var products = conn.ReadProducts();
                        foreach (var p in products)
                        {
                            Console.WriteLine("______________________________________________");
                            Console.WriteLine(p.ToString() + "\n");
                        }
                    }
                        break;
                    case 2:
                    {
                        Console.Write("Name: ");
                        var name = Console.ReadLine();
                        Console.Write("Description: ");
                        var desc = Console.ReadLine();
                        Console.Write("Colour: ");
                        var colour = Console.ReadLine();
                        Console.Write("Price: ");
                        var price = new Decimal(float.Parse(Console.ReadLine()));

                        var newProduct = new Product(name, desc, colour, price);
                        conn.InsertProducts(new List<Product>() {newProduct});
                    }
                        break;
                    case 3:
                    {
                        Console.Write("Search: ");
                        var name = Console.ReadLine();

                        var found = conn.FindProduct(name);
                        Console.WriteLine("_______________________________");
                        Console.WriteLine(found);
                    }
```



```
                switch(input)
                {
                    case 1:
                    {
                        var products = conn.ReadProducts();
                        foreach (var p in products)
                        {
                            Console.WriteLine("_______________________________________________");
                            Console.WriteLine(p.ToString() + "\n");
                        }
                    }
                        break;
                    case 2:
                    {

                        Console.Write("Name: ");
                        var name = Console.ReadLine();
                        Console.Write("Description: ");
                        var desc = Console.ReadLine();
                        Console.Write("Colour: ");
                        var colour = Console.ReadLine();
                        Console.Write("Price: ");
                        var price = new Decimal(float.Parse(Console.ReadLine()));

                        var newProduct = new Product(name, desc, colour, price);
                        conn.InsertProducts(new List<Product>() {newProduct});
                    }
                        break;
                    case 3:
                    {
                        Console.Write("Search: ");
                        var name = Console.ReadLine();

                        var found = conn.FindProduct(name);
                        Console.WriteLine("______________________________");
                        Console.WriteLine(found);
                    }
```

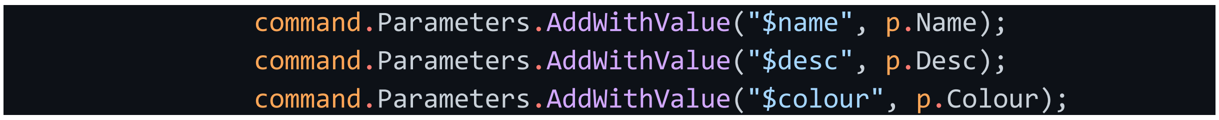


```
        public static Product FindProduct(this SqliteConnection conn, String name)
        {
            var found = new Product("N/A", "N/A", "N/A", new Decimal(0));

            conn.Open();
            var command = conn.CreateCommand();
            command.CommandText =
            @"
                SELECT
                name, desc, colour, price
                FROM Product WHERE name = $name
            ";

            command.Parameters.AddWithValue("$name", name);

            using(var reader = command.ExecuteReader())
            {
                while(reader.Read())
                {
                    var namef = reader.GetString(0);
                    var desc = reader.GetString(1);
                    var colour = reader.GetString(2);
                    var price = new Decimal(reader.GetFloat(3));

                    found = new Product(namef, desc, colour, price);

                }
            }

            conn.Close();
            return found;
        }
    }
}
```

LT4.5. Ensure your SQLite inputs are sanitized to prevent injection. Briefly describe here what are injections and what precautions you did to prevent it.

Injection attacks in SQLite, or SQLIs are attacks that make use of corrupt SQL code for database manipulation and gain access to data that was not intended to be accessible to the public. These attacks can be used to access any form of data and information such as private details and restricted user lists ((What is SQL Injection | SQLI Attack Example & Prevention Methods | Imperva, 2022)).

Some methods where precautionary measures were taken to avoid these attacks in the program is the implementation of parameters.



```
                command.Parameters.AddWithValue("$name", p.Name);
                command.Parameters.AddWithValue("$desc", p.Desc);
                command.Parameters.AddWithValue("$colour", p.Colour);
```

The implementation of columns instead of using * in the SELECT statement.

Parametrized queries also help avoid SQLite injections using a prepared statement by creating several layers or defense. The use of these methods together builds a strong defense against malicious injection attacks and protects data.

References

Learning Center. 2022. *What is SQL Injection | SQLI Attack Example & Prevention Methods | Imperva*. [online] Available at: <https://www.imperva.com/learn/application-security/sql-injection-sqli/#:~:text=SQL%20injection%2C%20also%20known%20as,lists%20or%20private%20customer%20details.> [Accessed 29 September 2022].

Database.cs

```
using System;
using System.Collections.Generic;
using Microsoft.Data.Sqlite;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace P1
{
    /**
     * This static class should be a provider for an SQLite db
     * Make sure the package is included in your project
     *
     *
     * Run the following to install the package in your project directory :
     *
     *          dotnet add package Microsoft.Data.Sqlite
     *
     * Read documentation for Microsoft.Data.Sqlite
     * and complete the following functions.
     *
     *
     * Note "this" within the function parameter is called a method extension
     * Read : https://docs.microsoft.com/en-us/dotnet/csharp/programming-
guide/classes-and-structs/extension-methods
     */
    static class Database
    {
        public static SqliteConnection CreateConnection()
        {
            return new SqliteConnection("Data Source=data.db");
        }

        public static void CreateTable(this SqliteConnection conn)
        {
            conn.Open();
            var command = conn.CreateCommand();
            command.CommandText =
            @"
                CREATE TABLE Product (
                    name TEXT,
                    desc TEXT,
                    colour TEXT,
                    price REAL,
                    PRIMARY KEY(name)
                )";
            command.ExecuteNonQuery();
            conn.Close();
        }

        public static void InsertProducts(this SqliteConnection conn,
IEnumerable<Product> products)
        {
            conn.Open();

            foreach (var p in products){

                var command = conn.CreateCommand();
                command.CommandText =
                @"
                    INSERT INTO Product (name, desc, colour, price)
                    VALUES($name, $desc, $colour, $price)
                ";

                command.Parameters.AddWithValue("$name", p.Name);
                command.Parameters.AddWithValue("$desc", p.Desc);
                command.Parameters.AddWithValue("$colour", p.Colour);
                command.Parameters.AddWithValue("$price", p.Price);

                command.ExecuteNonQuery();

            }

            conn.Close();

        }

        public static IEnumerable<Product> ReadProducts(this SqliteConnection
conn)
        {
            var products = new List<Product>();

            conn.Open();
            var command = conn.CreateCommand();
            command.CommandText =
            @"
                SELECT name, desc, colour, price
                FROM product
            ";
            using (var reader = command.ExecuteReader())
            {
                while (reader.Read())
                {
                    var name = reader.GetString(0);
                    var desc = reader.GetString(1);
                    var colour = reader.GetString(2);
                    var price = new Decimal(reader.GetFloat(3));

                    products.Add(new Product(name, desc, colour, price));
                }
            }
            conn.Close();

            return products;

        }

        public static void DeleteProduct(this SqliteConnection conn, String
name)
        {
            conn.Open();
            var command = conn.CreateCommand();
            command.CommandText =
            @"
                DELETE FROM Product WHERE name = $name
            ";

            command.Parameters.AddWithValue("$name", name);
            command.ExecuteNonQuery();

            conn.Close();
        }

        public static Product FindProduct(this SqliteConnection conn, String
name)
        {
            var found = new Product("N/A", "N/A", "N/A", new Decimal(0));

            conn.Open();
            var command = conn.CreateCommand();
            command.CommandText =
            @"
                SELECT
                name, desc, colour, price
                FROM Product WHERE name = $name
            ";
            command.Parameters.AddWithValue("$name", name);

            using(var reader = command.ExecuteReader())
            {
                while(reader.Read())
                {
                    var namef = reader.GetString(0);
                    var desc = reader.GetString(1);
                    var colour = reader.GetString(2);
                    var price = new Decimal(reader.GetFloat(3));

                    found = new Product(namef, desc, colour, price);

                }
            }

            conn.Close();
            return found;
        }
    }
}
```

Product.cs

```
using System;

namespace P1
{
    class Product
    {

        public String Name
        {
            get;
            set;
        }

        public string Desc
        {
            get;
            set;
        }

        public string Colour
        {
            get;
            set;
        }

        public Decimal Price
        {
            get;
            set;
        }

        public Product(string name, string desc, string colour, Decimal price)
        {
            Name = name;
            Desc = desc;
            Colour = colour;
            Price = price;
        }

        public override string ToString()
        {
            // To be modified
            // You can choose to print in json string or any other human
readable format
            return "The product name is: " + Name + "\nIts price is " + Price
+ "\nIts colour is " + Colour +  "\n" + "Further description: " + Desc;
        }

    }
}
```

Program.cs

```
using System;
using static P1.Product;
using System.Collections.Generic;

namespace P1
{
    class Program
    {
        static void Main(string[] args)
        {
            var conn = Database.CreateConnection();

            Database.CreateTable(conn);

            conn.InsertProducts(new List<Product>()
                {
                    new Product("Product 01", "Its small", "Yellow", new
decimal(11.3)),
                    new Product("Product 02", "Its medium", "Blue", new
decimal(1.1)),
                    new Product("Product 03", "Its large", "Green", new
decimal(21.1)),
                    new Product("Product 04", "Its too big to handle", "Red",
new decimal(31.1))
                }
            );

            var quit = false;

            while(!quit)
            {
                Console.WriteLine("1. List all products");
                Console.WriteLine("2. Insert a product");
                Console.WriteLine("3. Get a product");
                Console.WriteLine("4. Delete a product");
                Console.WriteLine("5. Quit");

                var input = Int16.Parse(Console.ReadLine());

                switch(input)
                {
                    case 1:
                    {
                        var products = conn.ReadProducts();
                        foreach (var p in products)
                        {
                            Console.WriteLine("_______________________________
_______________");
                            Console.WriteLine(p.ToString() + "\n");
                        }
                    }
                        break;
                    case 2:
                    {
                        Console.Write("Name: ");
                        var name = Console.ReadLine();
                        Console.Write("Description: ");
                        var desc = Console.ReadLine();
                        Console.Write("Colour: ");
                        var colour = Console.ReadLine();
                        Console.Write("Price: ");
                        var price = new
Decimal(float.Parse(Console.ReadLine()));
```

```
                        var newProduct = new Product(name, desc, colour,
price);
                        conn.InsertProducts(new List<Product>() {newProduct});
                    }
                        break;
                    case 3:
                    {
                        Console.Write("Search: ");
                        var name = Console.ReadLine();

                        var found = conn.FindProduct(name);
                        Console.WriteLine("_______________________________");
                        Console.WriteLine(found);
                    }
                        break;
                    case 4:
                    {
                        Console.Write("Delete: ");
                        var name = Console.ReadLine();

                        conn.DeleteProduct(name);
                    }
                        break;
                    case 5:
                        quit = true;
                        break;
                    default:
                        Console.WriteLine("Wrong command. Try again");
                        break;
                }
            }
        }
    }
}
```