

Cloud Application Development

**SIES (NERUL) COLLEGE OF ARTS SCIENCE AND
COMMERCE**

NAAC ACCREDITED ‘A’ GRADE COLLEGE (ISO

9001:2015 CERTIFIED INSTITUTION)

NERUL, NAVI MUMBAI - 400706



**DEPARTMENT OF INFORMATION
TECHNOLOGY
PRACTICAL BOOK ON**

Cloud Application Development

**SUBMITTED TO,
UNIVERSITY OF MUMBAI**

BY

**VISHAL VIJAY VARMA
MASTER OF INFORMATION
TECHNOLOGY**

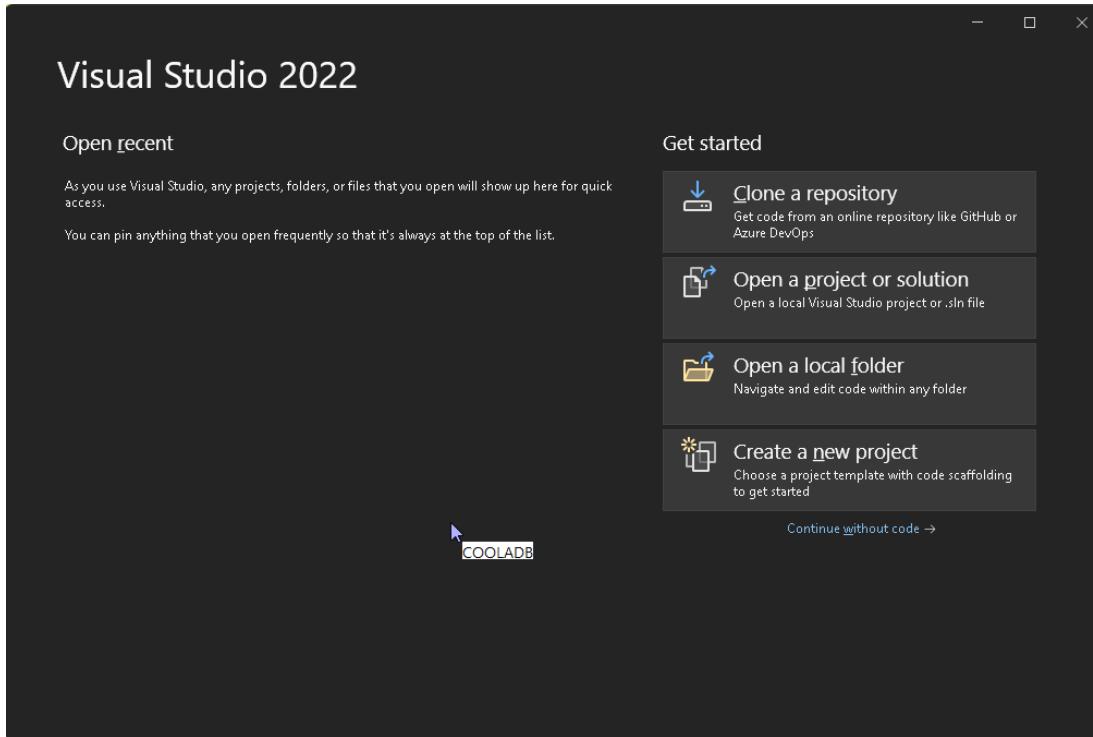
**PART - II (SEMESTER - III)
(2022-2023)**

INDEX

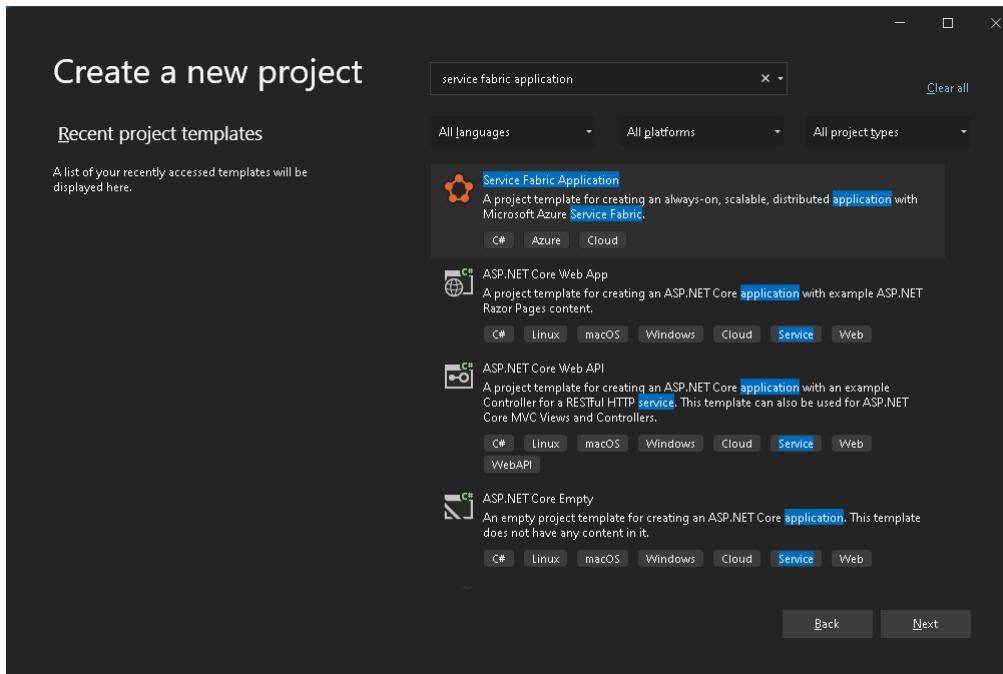
Sr . No	Practical	Page No
1.	Develop an ASP.NET Core MVC based Stateless Web App.	
2.	Develop a Spring Boot API.	
3.	Create an ASP.NET Core Web API and configure monitoring.	
4.	Create an ASP.NET Core Web API and configure monitoring.	
5.	Create an AKS cluster from the portal	
6.	Create an Application Gateway Using Ocelot and Securing APIs with Azure AD,	
7.	Create an API management service	

Steps :-

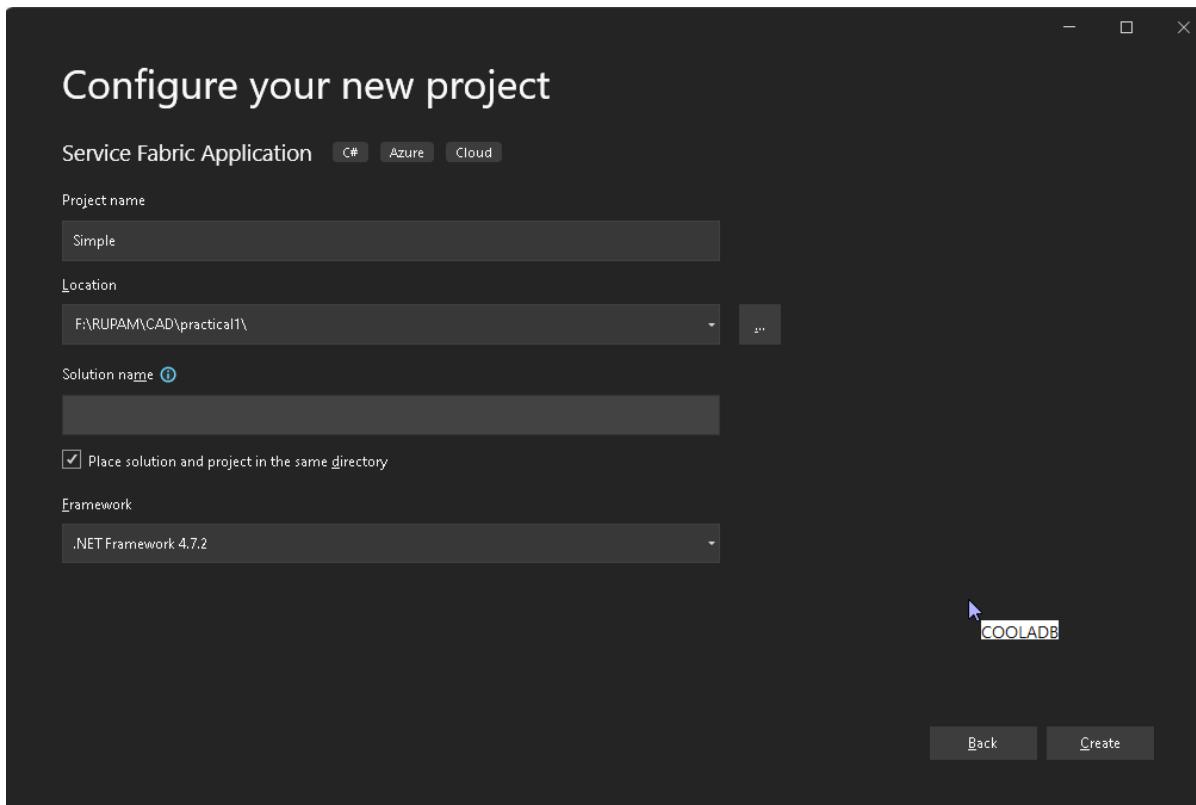
1. Launch Visual Studio 2017/19/22 as an administrator.
2. Create a new project



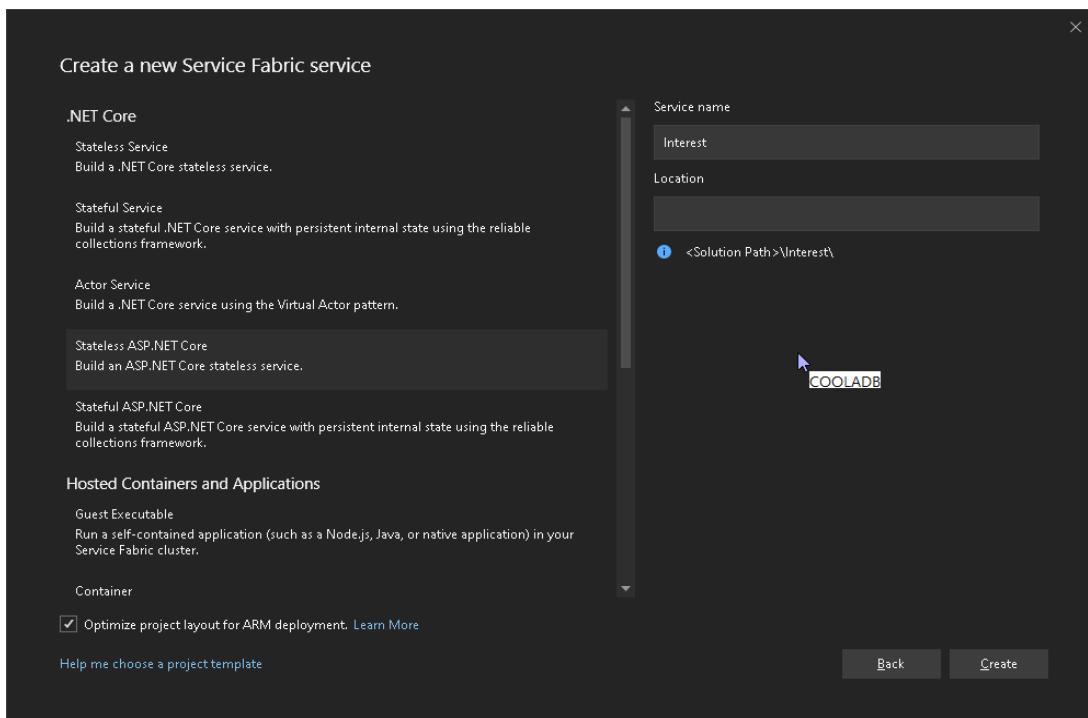
3. In the New Project dialog, choose ► Service Fabric Application and Next .



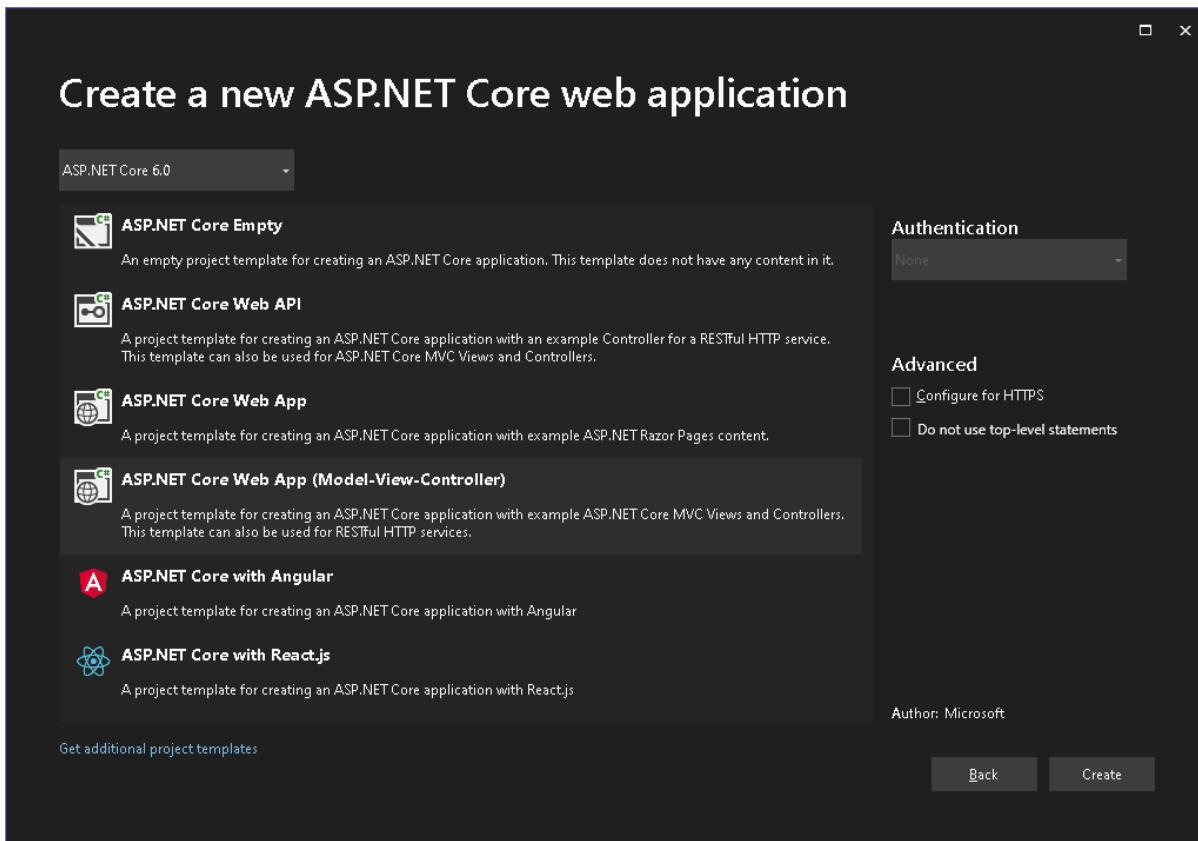
4. Name the Service Fabric application **Simple** and click Create .



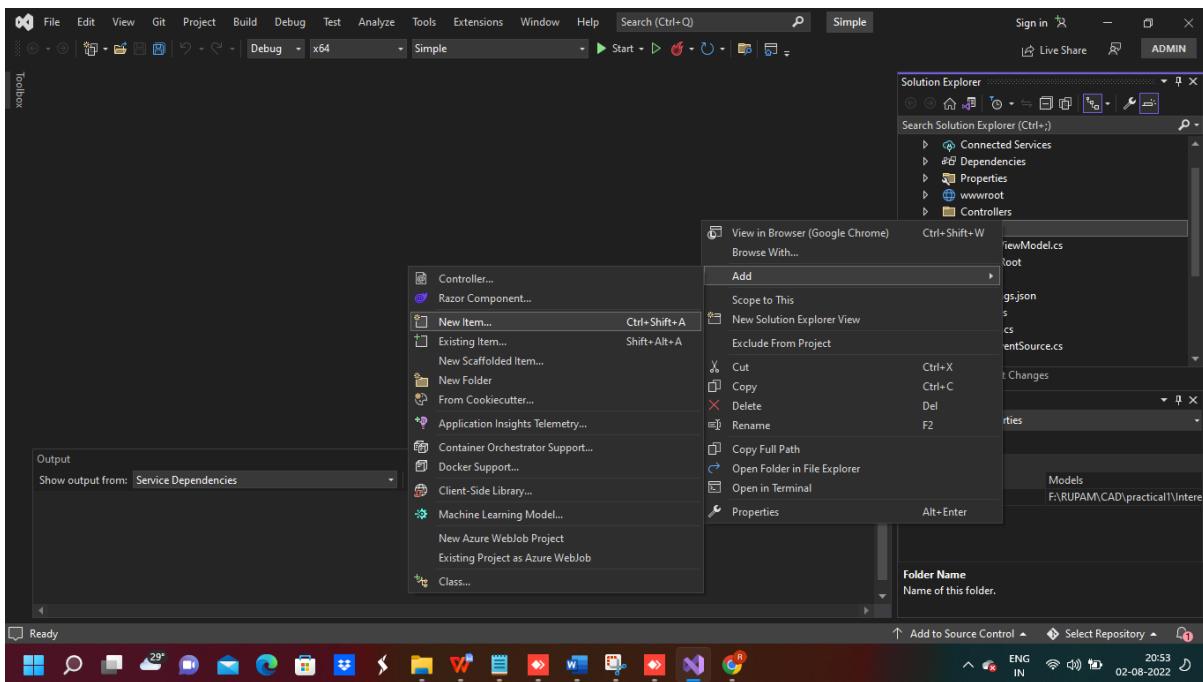
5. Choose Stateful ASP.NET Core, and give the services name **Interest** and click on Create .

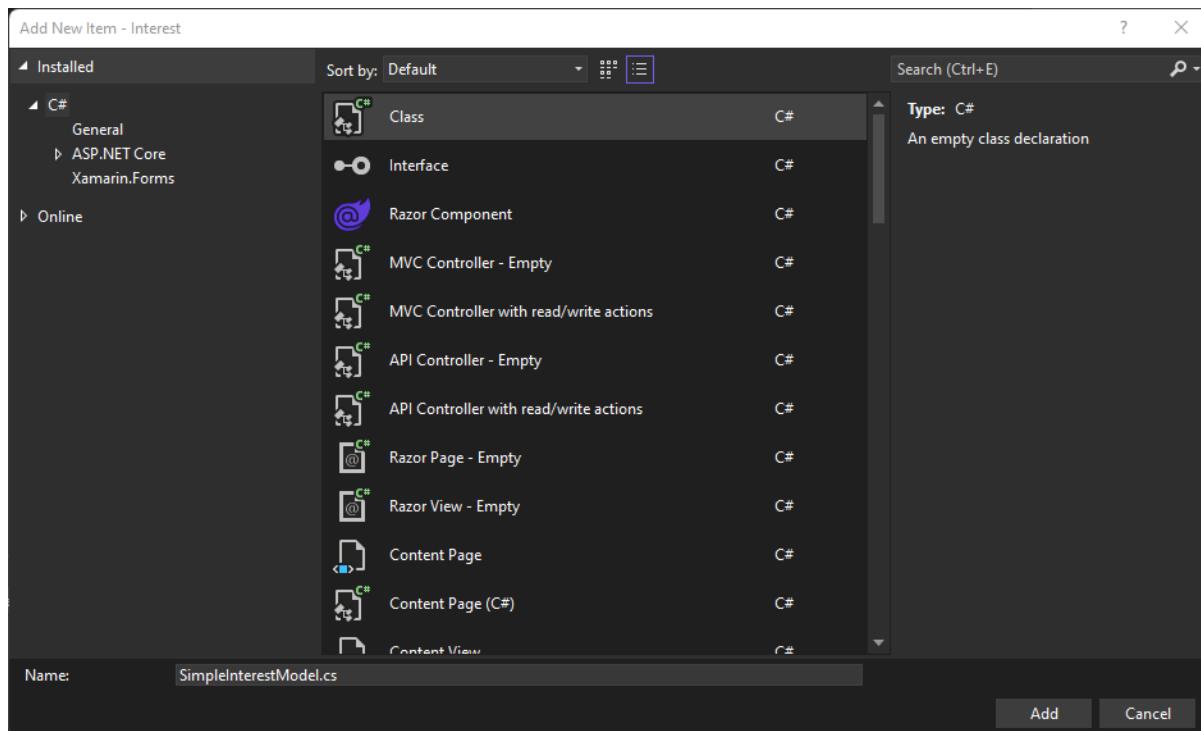


6. Select the ASP.Net Core (MVC) and click on Create .



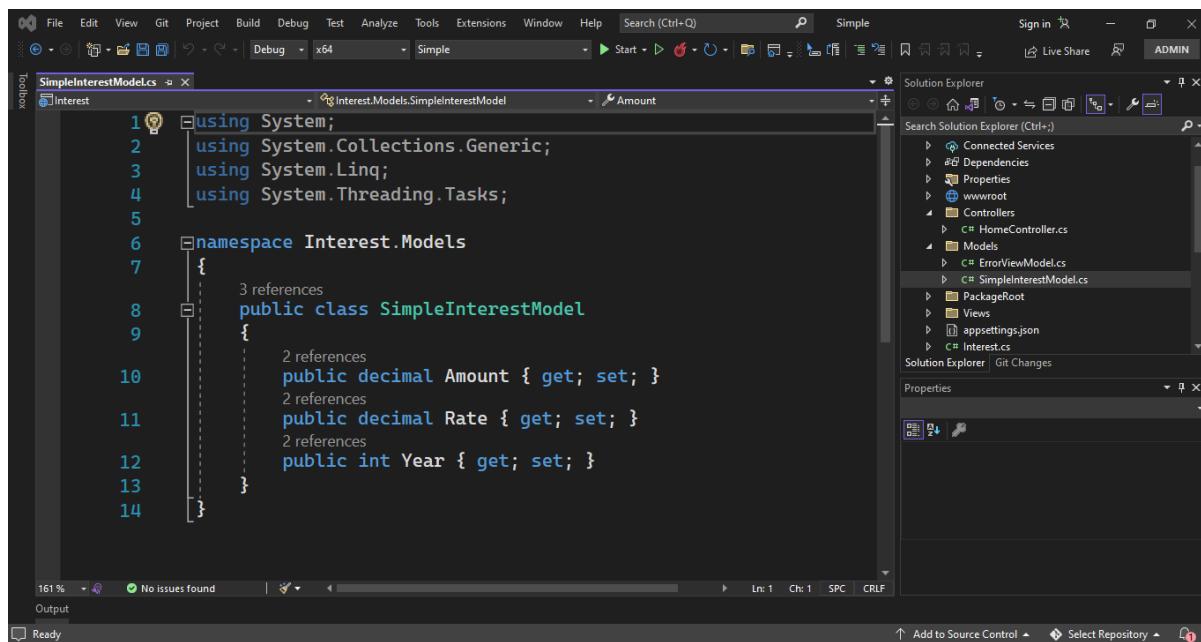
7. Now create a new Folder and name it Models & Right click on Model Folder and click on
►Add ►New Item ► Class ► Name it “SimpleInterestModel.cs ”





8. Type the Following code In **SimpleInterestModel.cs**

```
{  
    public decimal Amount { get; set; }  
    public decimal Rate { get; set; }  
    public int Year { get; set; }  
}
```



9. Now Click on **HomeController.cs** File And Type the Following code .

```
using Interest.Models;
using Microsoft.AspNetCore.Mvc;
using Microsoft.Extensions.Logging;
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using System.Text;
namespace Interest.Controllers
{
    public class HomeController : Controller
    {
        private readonly ILogger<HomeController> _logger;
        public HomeController(ILogger<HomeController> logger)
        {
            _logger = logger;
        }
        public IActionResult Index()
        {
            return View();
        }
        public IActionResult Privacy()
        {
            return View();
        }
        public IActionResult SimpleInterest()
        {
            SimpleInterestModel model = new SimpleInterestModel();
            return View(model);
        }
        [HttpPost]
        public IActionResult CalculateSimpleInterestResult(SimpleInterestModel model)
        {
            decimal simpleInterest = (model.Amount * model.Year * model.Rate) / 100;
            StringBuilder sbInterest = new StringBuilder();
            sbInterest.Append("<b>Amount :</b> " + model.Amount + "<br/>");
            sbInterest.Append("<b>Rate :</b> " + model.Rate + "<br/>");
            sbInterest.Append("<b>Time(year) :</b> " + model.Year + "<br/>");
            sbInterest.Append("<b>Interest :</b> " + simpleInterest);
        }
}
```

```

        return Content(sbInterest.ToString());
    }

[ResponseCache(Duration = 0, Location = ResponseCacheLocation.None, NoStore = true)]
public IActionResult Error()
{
    return View(new ErrorViewModel { RequestId = Activity.Current?.Id ?? 
HttpContext.TraceIdentifier });
}
}
}
}

```

```

Index.cshtml | launchSettings.json | HomeController.cs | SimpleInterestModel.cs
Interest
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
47
48
49
50

public IActionResult Privacy()
{
    return View();
}

public IActionResult SimpleInterest()
{
    SimpleInterestModel model = new SimpleInterestModel();
    return View(model);
}

[HttpPost]
public IActionResult CalculateSimpleInterestResult(SimpleInterestModel model)
{
    decimal simpleInterest = (model.Amount * model.Year * model.Rate) / 100;
    StringBuilder sbInterest = new StringBuilder();
    sbInterest.Append("Amount : " + model.Amount );
    sbInterest.Append("Rate : " + model.Rate );
    sbInterest.Append("Time(year) : " + model.Year );
    sbInterest.Append("Interest : " + simpleInterest);
    return Content(sbInterest.ToString());
}

[ResponseCache(Duration = 0, Location = ResponseCacheLocation.None, NoStore = true)]
public IActionResult Error()
{
    return View(new ErrorViewModel { RequestId = Activity.Current?.Id ?? HttpContext.TraceIdentifier });
}

```

10. Now Click ► view ► Home **Index.cshtml** File And Type the Following code .

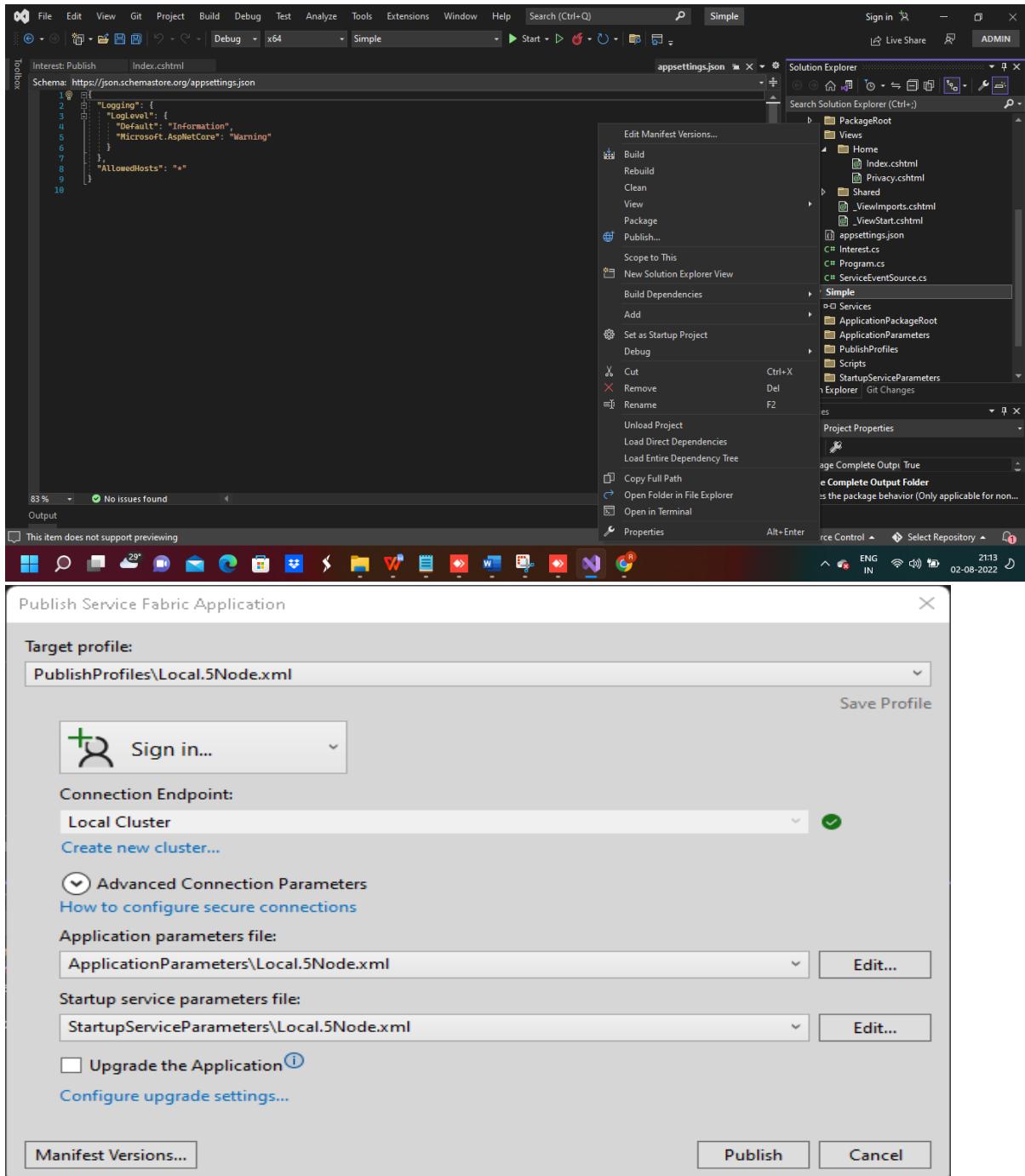
```
@{  
    ViewData["Title"] = "Home Page";  
}  
  
<div class="text-center">  
    <h1 class="display-4">Welcome</h1>  
    <p>Learn about <a href="https://docs.microsoft.com/aspnet/core">building Web apps with  
ASP.NET Core</a>.</p>  
</div>  
@model SimpleInterestModel  
  
{@{  
    ViewBag.Title = "SimpleInterest";  
}  
  
<h2>Calculate Simple Interest</h2>  
  
@using (Html.BeginForm("CalculateSimpleInterestResult", "Home", FormMethod.Post))  
{  
  
    <fieldset>  
        <legend>Calculate Simple Interest</legend>  
        <div id="divInterestDetails"></div>  
  
        <div class="editor-label">  
            @Html.LabelFor(model => model.Amount)  
        </div>  
        <div class="editor-field">  
            @Html.EditorFor(model => model.Amount)  
        </div>  
  
        <div class="editor-label">  
            @Html.LabelFor(model => model.Rate)  
        </div>  
        <div class="editor-field">  
            @Html.EditorFor(model => model.Rate)  
        </div>  
  
        <div class="editor-label">  
            @Html.LabelFor(model => model.Year)  
        </div>  
        <div class="editor-field">  
            @Html.EditorFor(model => model.Year)  
        </div>  
        <p>
```

```
        <input type="submit" value="Calculate" />
    </p>
</fieldset>
}
```

The screenshot shows a code editor window with the tab bar at the top containing "Index.cshtml", "launchSettings.json", "HomeController.cs", and "SimpleInterestModel.cs". The main area displays the following C# Razor code:

```
16
17 @using (Html.BeginForm("CalculateSimpleInterestResult", "Home", FormMethod.Post))
18 {
19     <fieldset>
20         <legend>Calculate Simple Interest</legend>
21         <div id="divInterestDetails"></div>
22
23         <div class="editor-label">
24             @Html.LabelFor(model => model.Amount)
25         </div>
26         <div class="editor-field">
27             @Html.EditorFor(model => model.Amount)
28         </div>
29
30         <div class="editor-label">
31             @Html.LabelFor(model => model.Rate)
32         </div>
33         <div class="editor-field">
34             @Html.EditorFor(model => model.Rate)
35         </div>
36
37         <div class="editor-label">
38             @Html.LabelFor(model => model.Year)
39         </div>
40         <div class="editor-field">
41             @Html.EditorFor(model => model.Year)
42         </div>
43         <p>
44             <input type="submit" value="Calculate" />
45         </p>
46     </fieldset>
47 }
48 }
```

11. Now Right click on Project “Simple” ➤ Publish ➤ Select the Node 5 and click on Publish and see the output .



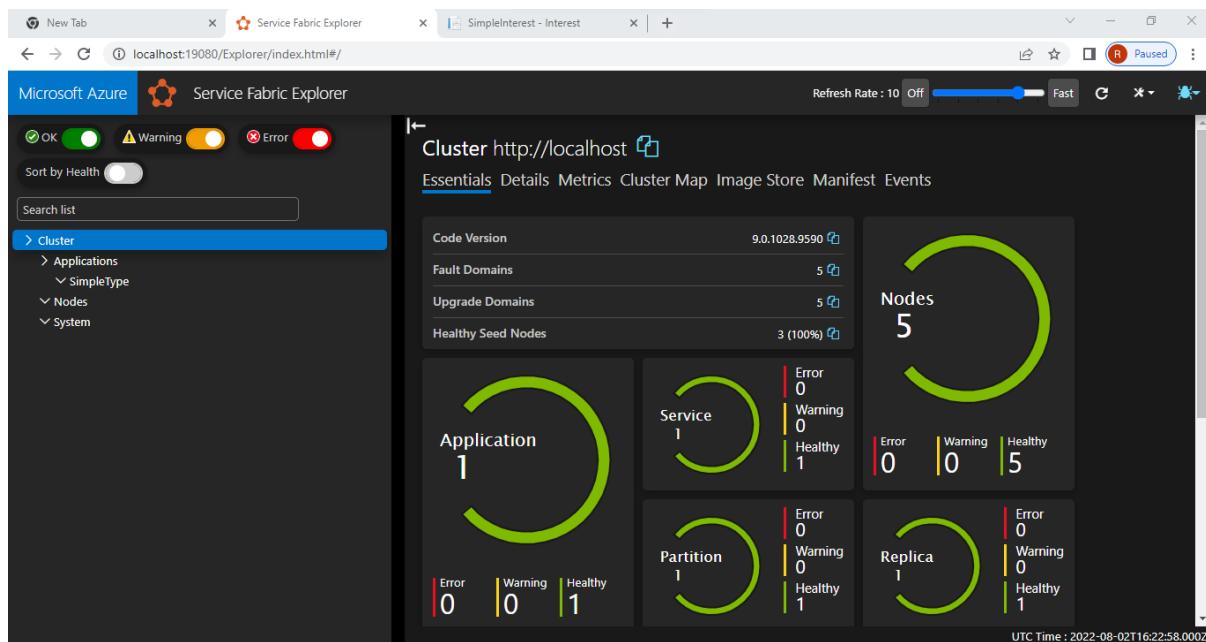
The screenshot shows the Visual Studio IDE interface. The code editor displays the `launchSettings.json` file, which contains configuration for IIS Express and a profile named "Interest". The `appsettings.json` file is also visible. The Solution Explorer pane shows the project structure with controllers, models, views, and shared files. The Output window shows the build and publish logs, indicating a successful publish to the local host.

```

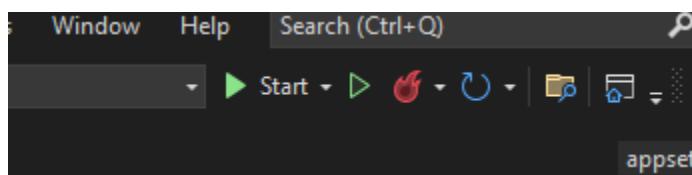
Schema: https://json.schemastore.org/launchsettings.json
1  {
2    "iisSettings": {
3      "windowsAuthentication": false,
4      "anonymousAuthentication": true,
5      "iisExpress": {
6        "applicationUrl": "http://localhost:42697",
7        "sslPort": 0
8      }
9    }
10   "profiles": [
11     "Interest": {
12       "commandName": "Project",
13       "dotnetRunMessages": true,
14       "launchBrowser": true,
15       "applicationUrl": "http://localhost:5193",
16       "environmentVariables": {
17         "ASPNETCORE_ENVIRONMENT": "Development"
18       }
19     }
20   ]
21 }

Output
Show output from: Build
2>Starting executing script 'Get-ServiceFabricApplicationStatus'.
2>powershell -NonInteractive -NoProfile -WindowStyle Hidden -ExecutionPolicy Bypass -Command "[void](Connect-ServiceFabricCluster -TimeoutSec:10 -WarningAction:'SilentlyContinue'); Import-Module ServiceFabricSDK -Force"
2>WARNING: The names of some imported commands from the module 'ServiceFabricSDK' include unapproved verbs that might
2>make them less discoverable. To find the commands with unapproved verbs, run the Import-Module command again with the
2>Verbose parameter. For a list of approved verbs, type Get-Verb.
2>The application has started.
2>Service Status:
2>fabric:/Simple/Interest is ready.
2>
2>The application is ready.
2>Finished executing script 'Get-ServiceFabricApplicationStatus'.
2>Time elapsed: 00:00:00.9901258
===== Build: 1 succeeded, 0 Failed, 1 up-to-date, 0 skipped =====
===== Publish: 1 succeeded, 0 Failed, 0 skipped =====
Publish succeeded

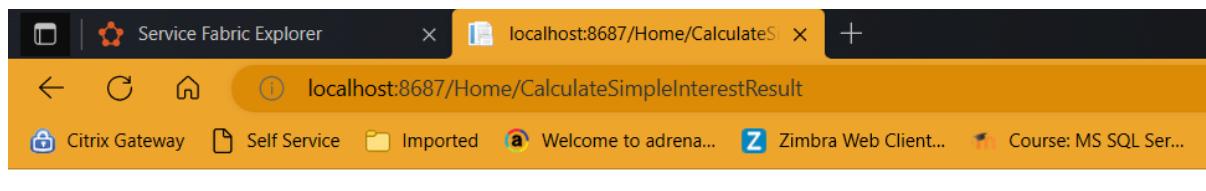
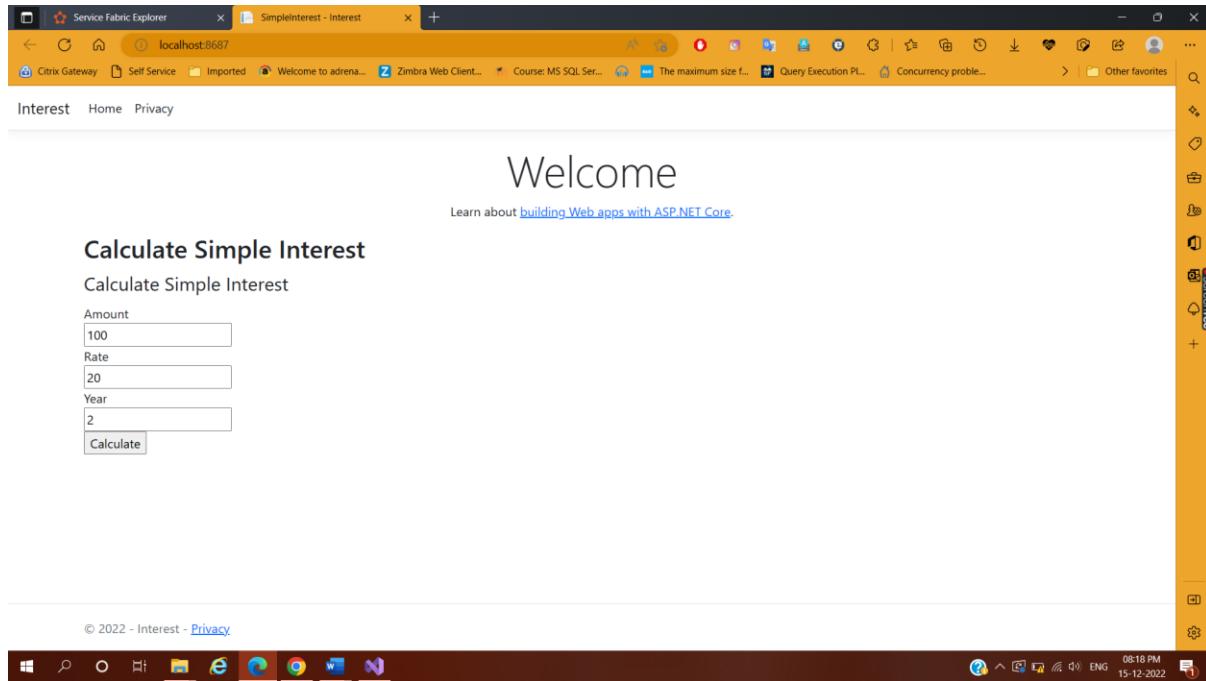
```



The screenshot shows the Microsoft Azure Service Fabric Explorer interface. On the left, there's a navigation sidebar with 'Cluster' selected, followed by 'Applications' (which is also selected), 'Nodes', and 'System'. Under 'Applications', there's a tree view with 'SimpleType' expanded, showing 'fabric/Simple' and 'fabric/Simple/Interest'. Below that is a node named '6861b6a1-11c3-4e1d-930c-6213a05a3009 _Node_2'. On the right, the main panel displays the 'Applications' list with one entry: 'fabric/Simple' (Application Type: SimpleType, Version: 1.0.0, Health State: OK, Status: Ready). At the top right, there's a refresh rate slider set to 'Fast' and a 'Paused' button. The bottom right corner shows the UTC time: 2022-08-02T16:23:33.000Z.

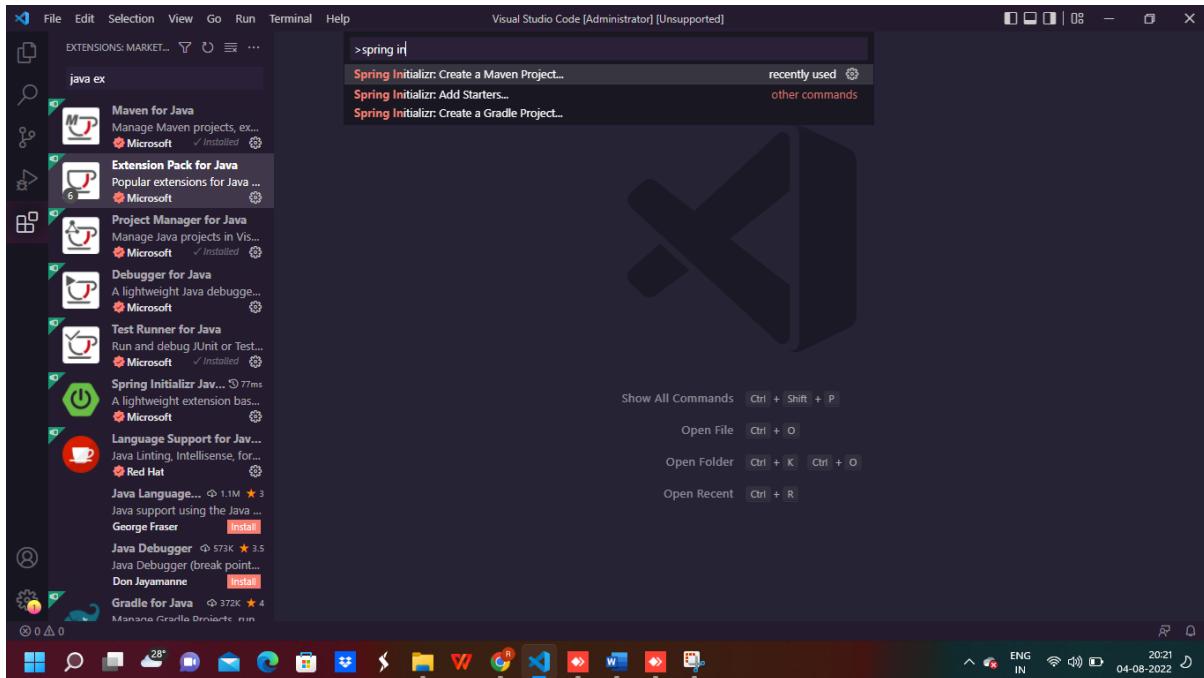


The screenshot shows a web browser window with the URL 'localhost:8453'. The page title is 'SimpleInterest - Interest'. The content includes a 'Welcome' message, a link to 'building Web apps with ASP.NET Core.', and a section titled 'Calculate Simple Interest' with a form. The form has four input fields: 'Amount' (with placeholder '0.00'), 'Rate' (with placeholder '0.00'), and 'Year' (with placeholder '0'). Below the fields is a 'Calculate' button. At the bottom of the page, there's a footer with the text '© 2022 - Interest - [Privacy](#)'.

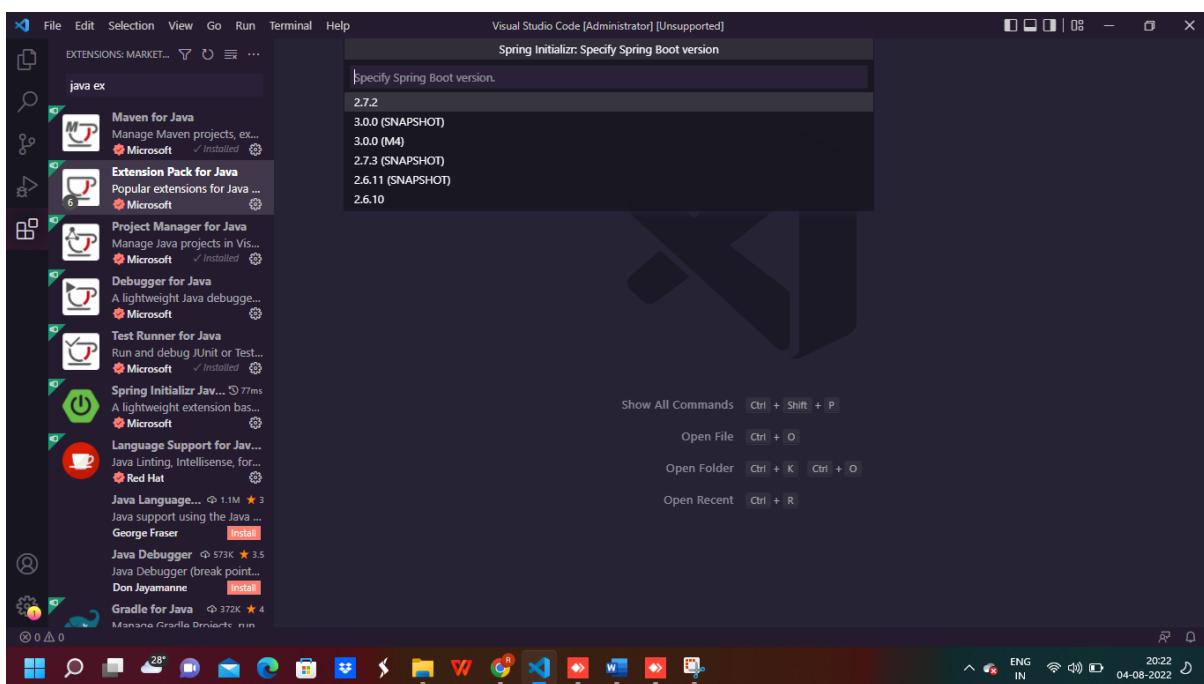


Steps :-

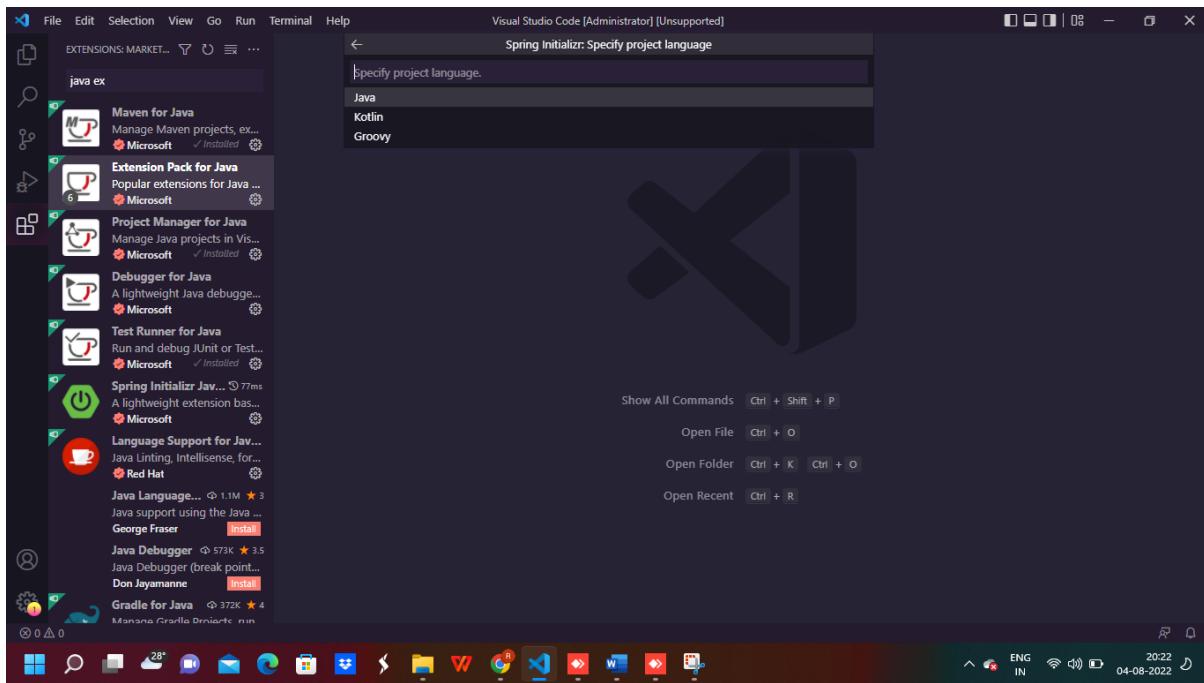
1. Open the Visual Studio Code
2. Click **ctrl+Shift+P** and search for Spring Initializer:Create Maven Project and hit enter button



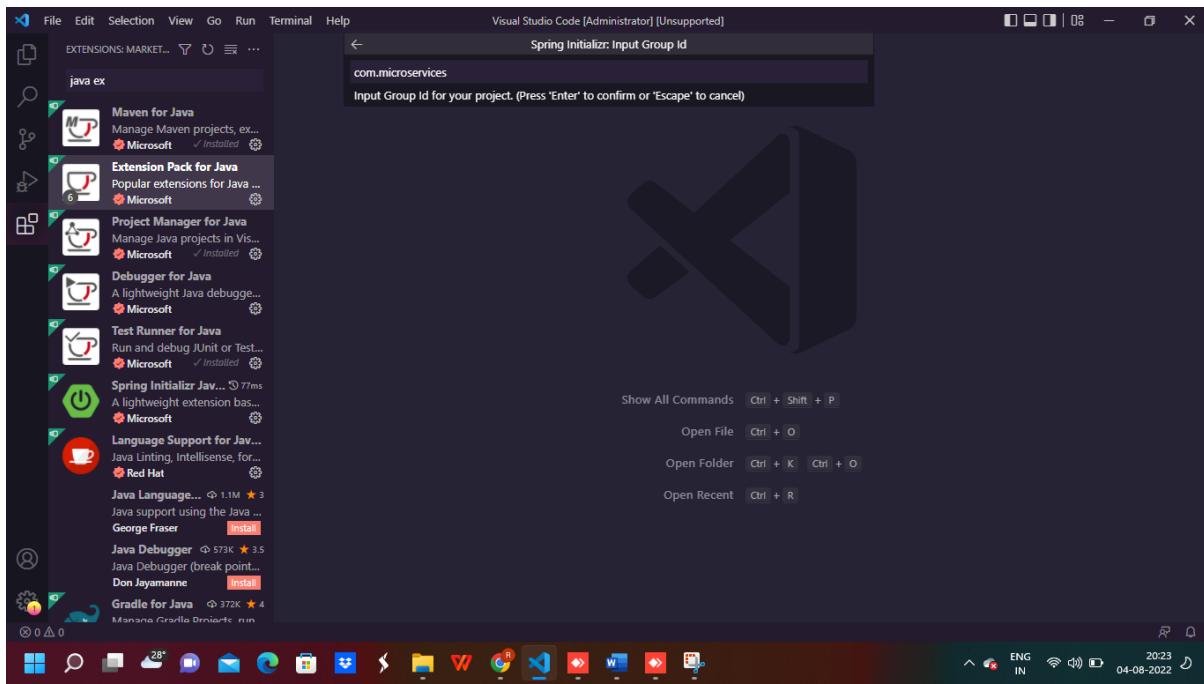
3. Search for 2.7.2 version and press enter .



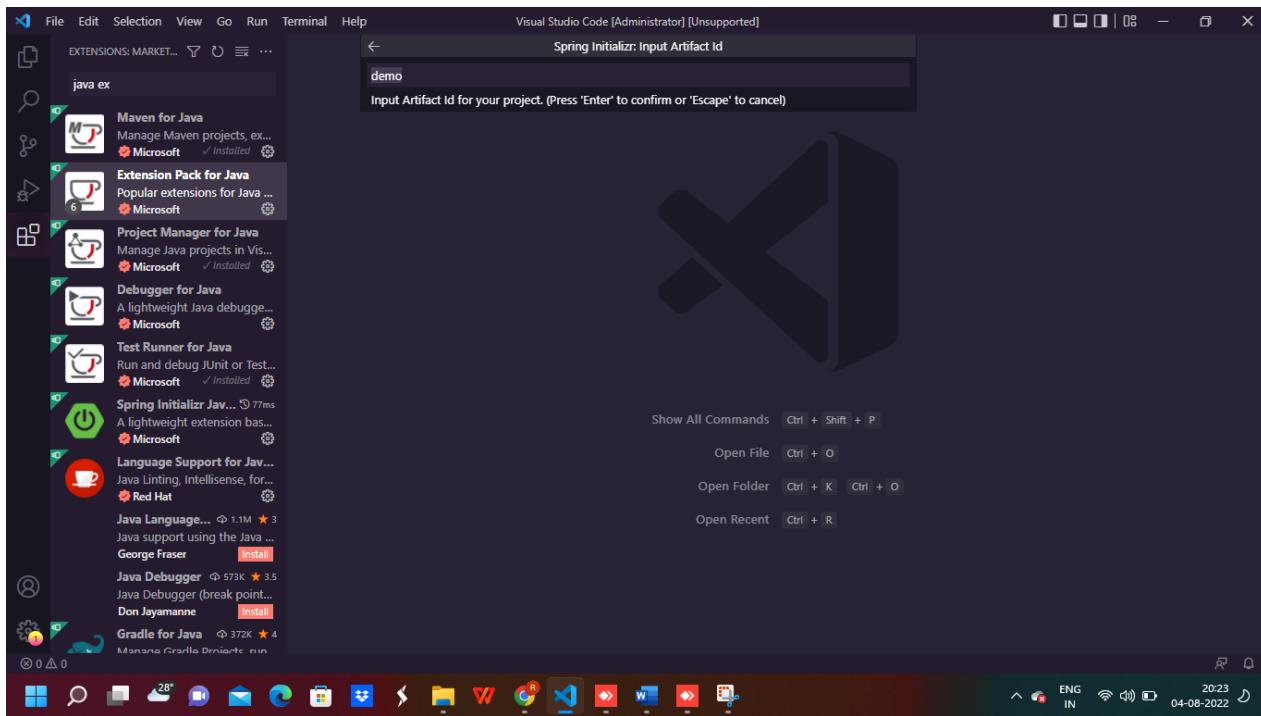
4. Search For Java and press enter .



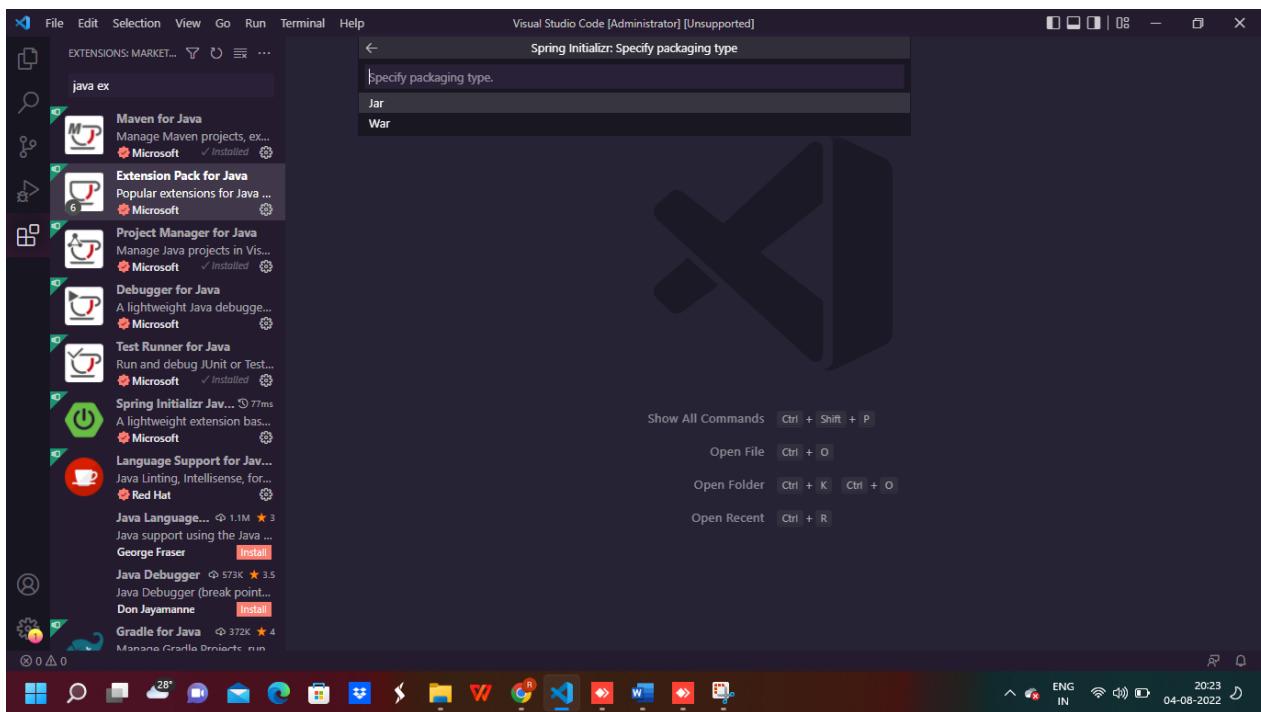
5. Now Type com.microservices and press enter .

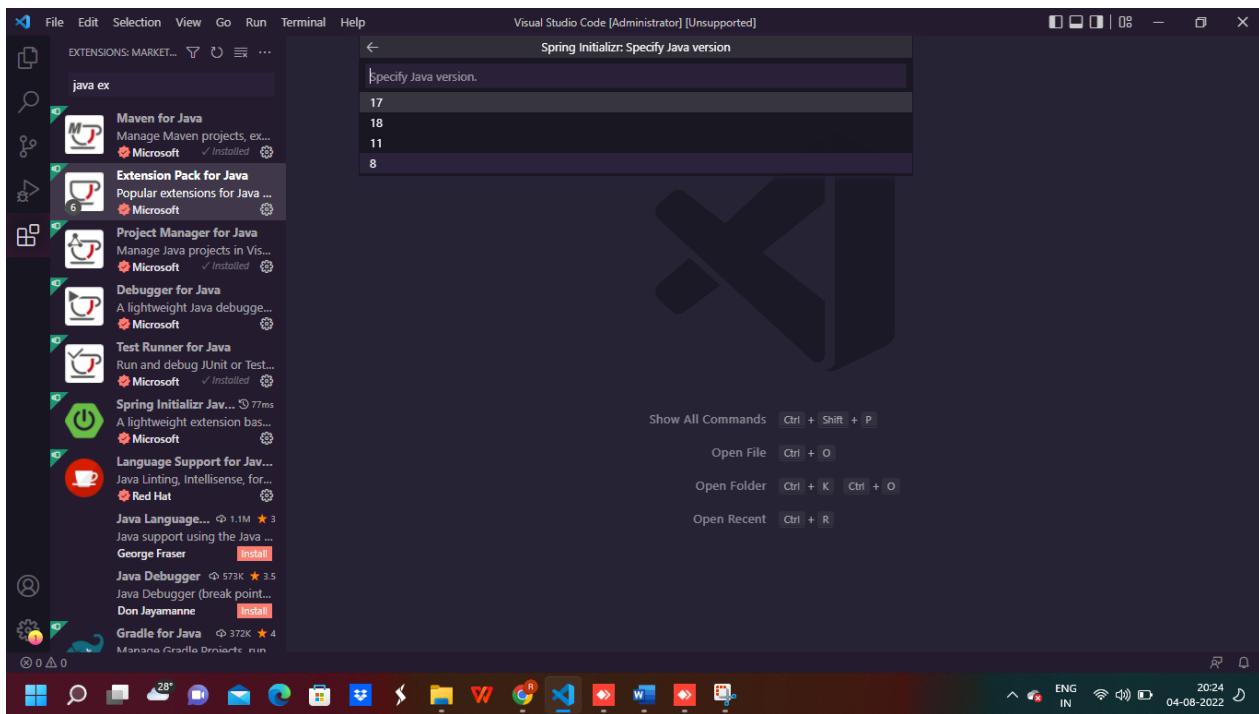


6. Type Demo and press enter .

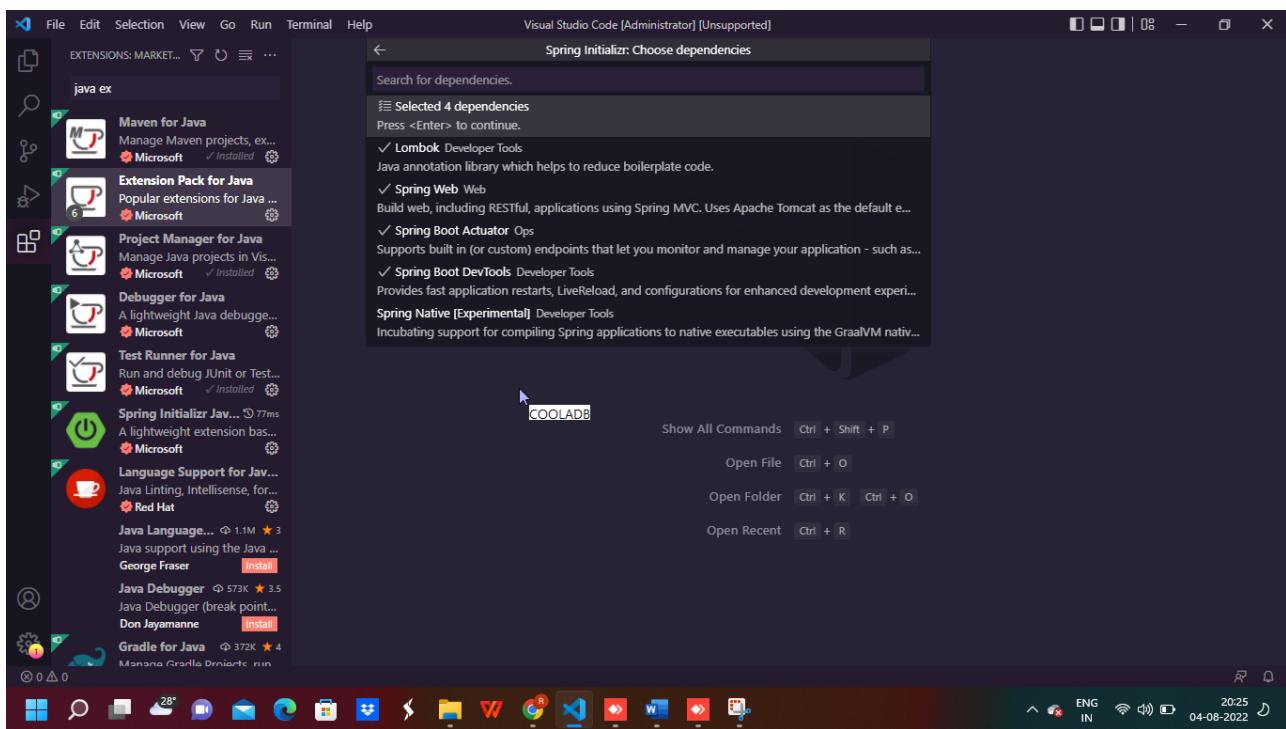


7. Select Jar And 8 and press enter .

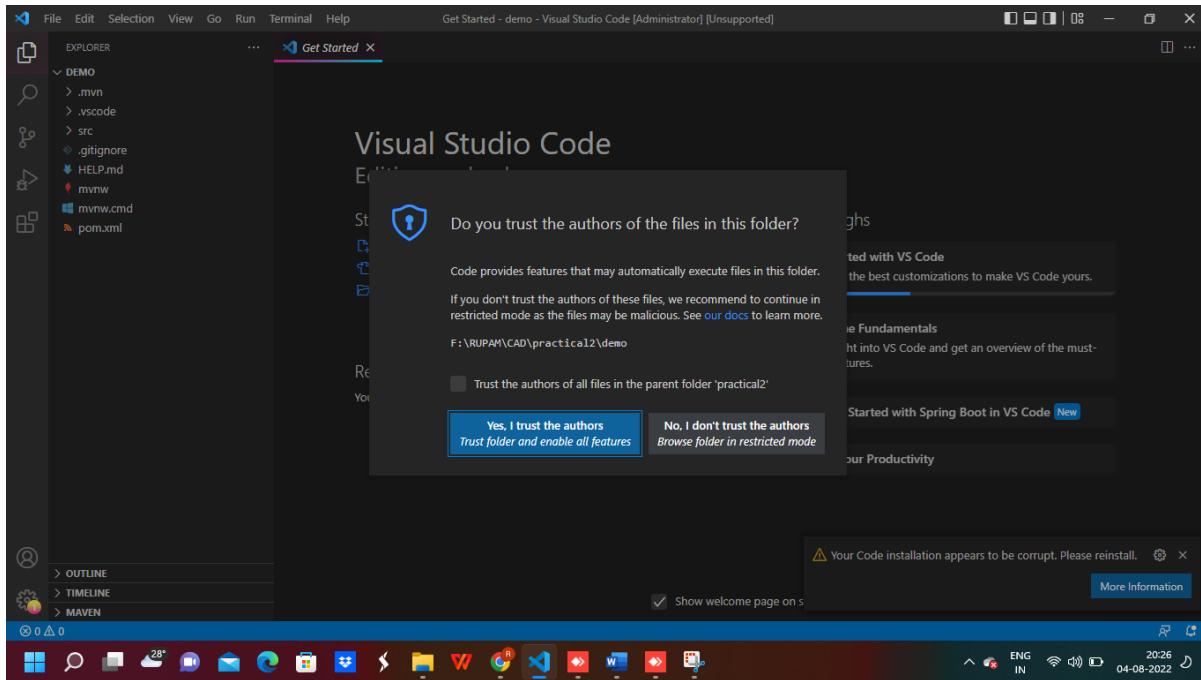




8. Select the four Dependencies :- **Lombok** , **Spring Web** , **Spring Boot Actuator** , **Spring Boot DevTool** and Press Enter .



9. Select the Path To save the Project and open the project



10. Under the Project click src ► Main ► Java ► DemoApplication.java Create the new three file Employee.java , EmployeeService.java , EmployeeController.java and type the following code :

```
① DemoApplication.java X
src > main > java > com > microservices > demo > ① DemoApplication.java > ...
1 package com.microservices.demo;
2
3 import org.springframework.boot.SpringApplication;
4 import org.springframework.boot.autoconfigure.SpringBootApplication;
5 import org.springframework.web.bind.annotation.RequestMapping;
6 import org.springframework.web.bind.annotation.RestController;
7
8 @SpringBootApplication
9 @RestController
10 public class DemoApplication {
11
12     @RequestMapping("/")
13     public String home() {
14         return "Hello Docker World";
15     }
16
17     public static void main(String[] args) {
18         SpringApplication.run(DemoApplication.class, args);
19     }
20
21 }
22
```

Employee.java

```
package com.microservices.demo;
import lombok.AllArgsConstructor;
import lombok.EqualsAndHashCode;
import lombok.Getter;
import lombok.Setter;
@Getter
@Setter
@EqualsAndHashCode
@AllArgsConstructor
```

```
public class Employee {
```

```
    private String firstName;
    private String lastName;
    private String ipAddress;
```

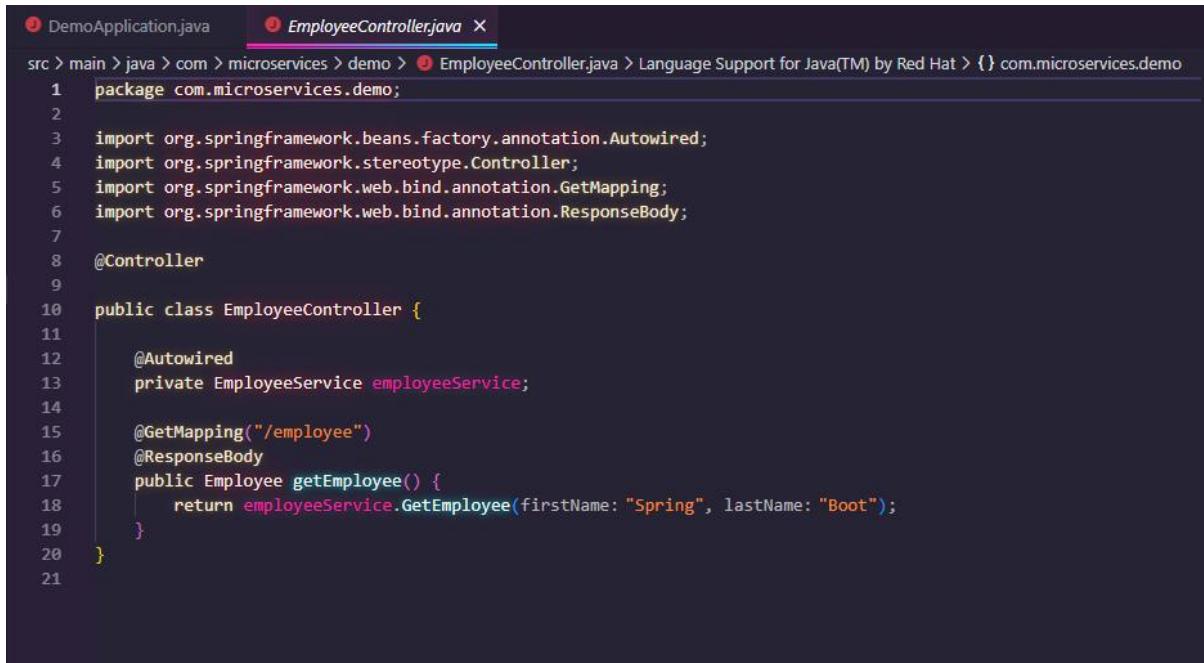
```
}
```

The screenshot shows a code editor with the Employee.java file open. The tab bar at the top has two tabs: "DemoApplication.java" and "Employee.java". The "Employee.java" tab is highlighted with a blue underline. The code editor displays the Java code for the Employee class, including imports for Lombok annotations and fields for first name, last name, and IP address.

```
1 package com.microservices.demo;
2
3 import lombok.AllArgsConstructor;
4 import lombok.EqualsAndHashCode;
5 import lombok.Getter;
6 import lombok.Setter;
7
8 @Getter
9 @Setter
10 @EqualsAndHashCode
11 @AllArgsConstructor
12
13 public class Employee {
14
15     private String firstName;
16     private String lastName;
17     private String ipAddress;
18 }
19
```

EmployeeController.java

```
package com.microservices.demo;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.ResponseBody;
@Controller
public class EmployeeController {
    @Autowired
    private EmployeeService employeeService;
    @GetMapping("/employee")
    @ResponseBody
    public Employee getEmployee(){
        return employeeService.GetEmployee("Spring", "Boot");
    }
}
```



The screenshot shows a code editor window with the tab bar at the top. The active tab is 'EmployeeController.java'. The file path is 'src > main > java > com > microservices > demo > EmployeeController.java'. The code itself is identical to the one shown in the previous code block.

```
1 package com.microservices.demo;
2
3 import org.springframework.beans.factory.annotation.Autowired;
4 import org.springframework.stereotype.Controller;
5 import org.springframework.web.bind.annotation.GetMapping;
6 import org.springframework.web.bind.annotation.ResponseBody;
7
8 @Controller
9
10 public class EmployeeController {
11     @Autowired
12     private EmployeeService employeeService;
13
14     @GetMapping("/employee")
15     @ResponseBody
16     public Employee getEmployee() {
17         return employeeService.GetEmployee(firstName: "Spring", lastName: "Boot");
18     }
19 }
20
21 }
```

EmployeeService.java

```
package com.microservices.demo;
import java.net.InetAddress;
import java.net.UnknownHostException;
import org.springframework.stereotype.Service;
@Service

public class EmployeeService {
    public Employee GetEmployee(String firstName, String lastName)
    {
        String ipAddress;
        try{
            ipAddress=InetAddress.getLocalHost().getHostAddress().toString();
        }
        catch (UnknownHostException e)
        {
            ipAddress=e.getMessage();
        }
        Employee employee= new Employee(firstName,lastName,ipAddress);
        return employee;
    }
}
```

The screenshot shows a code editor window with the tab bar at the top. The active tab is "EmployeeService.java". There are other tabs for "DemoApplication.java" and "Language Support for Java(TM) by Red Hat". The code editor displays the Java code for the EmployeeService class. The code includes imports for InetAddress and UnknownHostException from java.net, and Service from org.springframework.stereotype. It uses the @Service annotation. The GetEmployee method takes two strings, firstName and lastName, and returns an Employee object. Inside the method, it tries to get the local host's IP address using InetAddress.getLocalHost().getHostAddress().toString(). If an UnknownHostException occurs, it catches the exception and returns the message. Finally, it creates an Employee object with the provided first and last names and the retrieved IP address, and returns it.

```
src > main > java > com > microservices > demo > EmployeeService.java > Language Support for Java(TM) by Red Hat > 
```

```
1 package com.microservices.demo;
2
3 import java.net.InetAddress;
4 import java.net.UnknownHostException;
5 import org.springframework.stereotype.Service;
6
7 @Service
8 public class EmployeeService {
9     public Employee GetEmployee(String firstName, String lastName) {
10         String ipAddress;
11         try {
12             ipAddress = InetAddress.getLocalHost().getHostAddress().toString();
13         } catch (UnknownHostException e) {
14             ipAddress = e.getMessage();
15         }
16         Employee employee = new Employee(firstName, lastName, ipAddress);
17         return employee;
18     }
19 }
20 }
```

11. Run the DemoApplication.java by Clicking on Run Java Button .

```

1 package com.microservices.demo;
2
3 import org.springframework.boot.SpringApplication;
4 import org.springframework.boot.autoconfigure.SpringBootApplication;
5 import org.springframework.web.bind.annotation.RequestMapping;
6 import org.springframework.web.bind.annotation.RestController;
7
8 @SpringBootApplication
9 @RestController
10 public class DemoApplication {
11
12     @RequestMapping("/")
13     public String home() {
14         return "Hello Docker World";
15     }
16
17     public static void main(String[] args) {
18         SpringApplication.run(DemoApplication.class, args);
19     }
}

```

Run | Debug

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

PS F:\RUPAM\CAD\practical2\demo> [REDACTED]

2022-08-04 20:51:29.613 INFO 21524 --- [restartedMain] o.s.b.d.a.OptionalLiveReloadServer : LiveReload server is running on port 35729
2022-08-04 20:51:29.630 INFO 21524 --- [restartedMain] o.s.b.a.e.web.EndpointLinksResolver : Exposing 1 endpoint(s) beneath base path '/ac...
2022-08-04 20:51:29.785 INFO 21524 --- [restartedMain] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat started on port(s): 8080 (http) with c...
2022-08-04 20:51:29.833 INFO 21524 --- [restartedMain] com.microservices.demo.DemoApplication : Started DemoApplication in 5.015 seconds (JVM...
2022-08-04 20:51:32.036 INFO 21524 --- [-192.168.64.207] o.a.c.c.C.[Tomcat].[localhost].[/] : Initializing Spring DispatcherServlet 'dispat...
2022-08-04 20:51:32.037 INFO 21524 --- [-192.168.64.207] o.s.web.servlet.DispatcherServlet : Initializing Servlet 'dispatcherServlet'
2022-08-04 20:51:32.039 INFO 21524 --- [-192.168.64.207] o.s.web.servlet.DispatcherServlet : Completed initialization in 2 ms
PS F:\RUPAM\CAD\practical2\demo>

Ln 8, Col 23 Tab Size: 4 UTF-8 LF {} Java ⌂ 20:53 04-08-2022

```

1 package com.microservices.demo;
2
3 import org.springframework.boot.SpringApplication;
4 import org.springframework.boot.autoconfigure.SpringBootApplication;
5 import org.springframework.web.bind.annotation.RequestMapping;
6 import org.springframework.web.bind.annotation.RestController;
7
8 @SpringBootApplication
9 @RestController
10 public class DemoApplication {
11
12     @RequestMapping("/")
13     public String home() {
14         return "Hello Docker World";
15     }
16
17     public static void main(String[] args) {
18         SpringApplication.run(DemoApplication.class, args);
19     }
}

```

Run | Debug

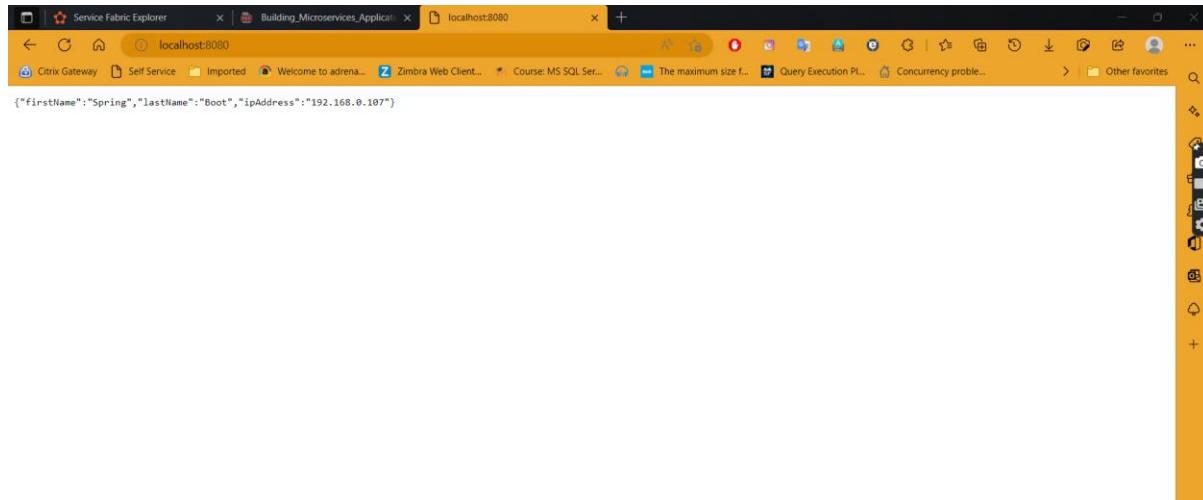
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

PS F:\RUPAM\CAD\practical2\demo> [REDACTED]

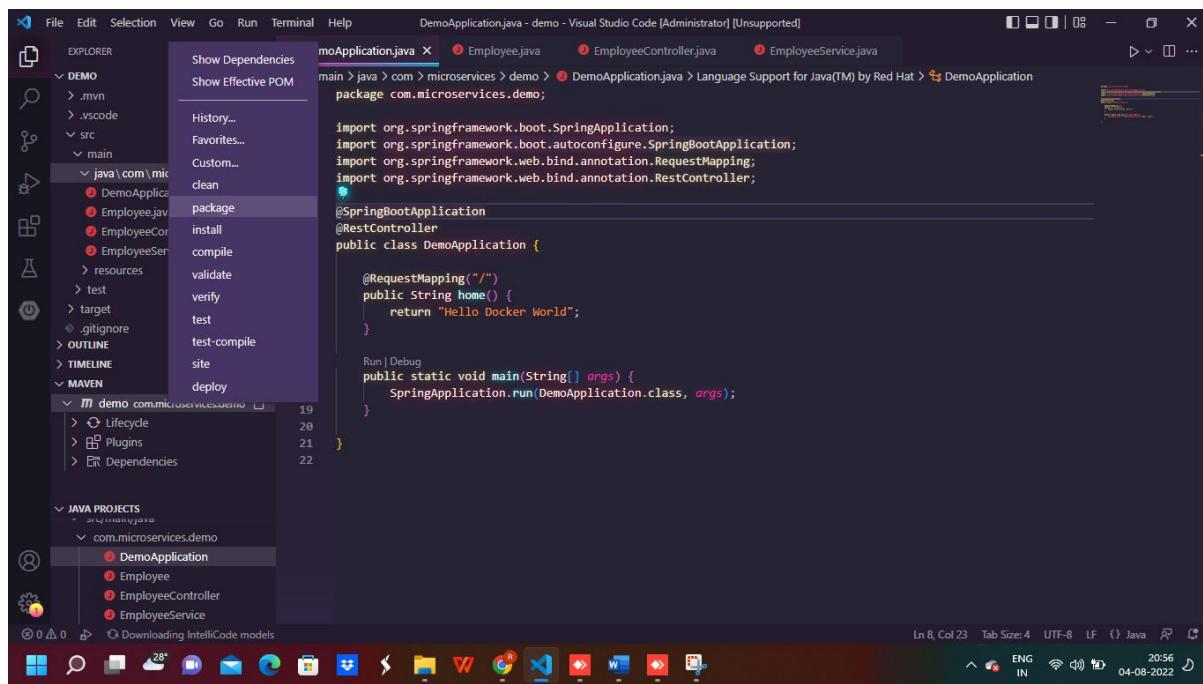
2022-08-04 20:51:25.680 INFO 21524 --- [restartedMain] com.microservices.demo.DemoApplication : Starting DemoApplication using Java 1.8.0_301...
2022-08-04 20:51:25.686 INFO 21524 --- [restartedMain] com.microservices.demo.DemoApplication : No active profile set, falling back to 1 defa...
2022-08-04 20:51:25.793 INFO 21524 --- [restartedMain] e.DevToolsPropertyDefaultsPostProcessor : Devtools property defaults active! Set 'sprin...
2022-08-04 20:51:25.794 INFO 21524 --- [restartedMain] e.DevToolsPropertyDefaultsPostProcessor : For additional web related logging consider s...
2022-08-04 20:51:27.766 INFO 21524 --- [restartedMain] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat initialized with port(s): 8080 (http)
2022-08-04 20:51:27.783 INFO 21524 --- [restartedMain] o.apache.catalina.core.StandardService : Starting service [Tomcat]
2022-08-04 20:51:29.613 INFO 21524 --- [restartedMain] o.s.b.d.a.OptionalLiveReloadServer : LiveReload server is running on port 35729
2022-08-04 20:51:29.630 INFO 21524 --- [restartedMain] o.s.b.a.e.web.EndpointLinksResolver : Exposing 1 endpoint(s) beneath base path '/ac...
2022-08-04 20:51:29.785 INFO 21524 --- [restartedMain] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat started on port(s): 8080 (http) with c...
2022-08-04 20:51:29.833 INFO 21524 --- [restartedMain] com.microservices.demo.DemoApplication : Started DemoApplication in 5.015 seconds (JVM...
2022-08-04 20:51:32.036 INFO 21524 --- [-192.168.64.207] o.a.c.c.C.[Tomcat].[localhost].[/] : Initializing Spring DispatcherServlet 'dispat...
2022-08-04 20:51:32.037 INFO 21524 --- [-192.168.64.207] o.s.web.servlet.DispatcherServlet : Initializing Servlet 'dispatcherServlet'
2022-08-04 20:51:32.039 INFO 21524 --- [-192.168.64.207] o.s.web.servlet.DispatcherServlet : Completed initialization in 2 ms
PS F:\RUPAM\CAD\practical2\demo>

Ln 8, Col 23 Tab Size: 4 UTF-8 LF {} Java ⌂ 20:55 04-08-2022

12. After successful running the project you can check the output on **localhost:8080**



13. Now click on **Right Click on Maven Project** at left hand side of File Explorer Click on **Package** and wait till the project get Build Successfull and **Jar** file get build



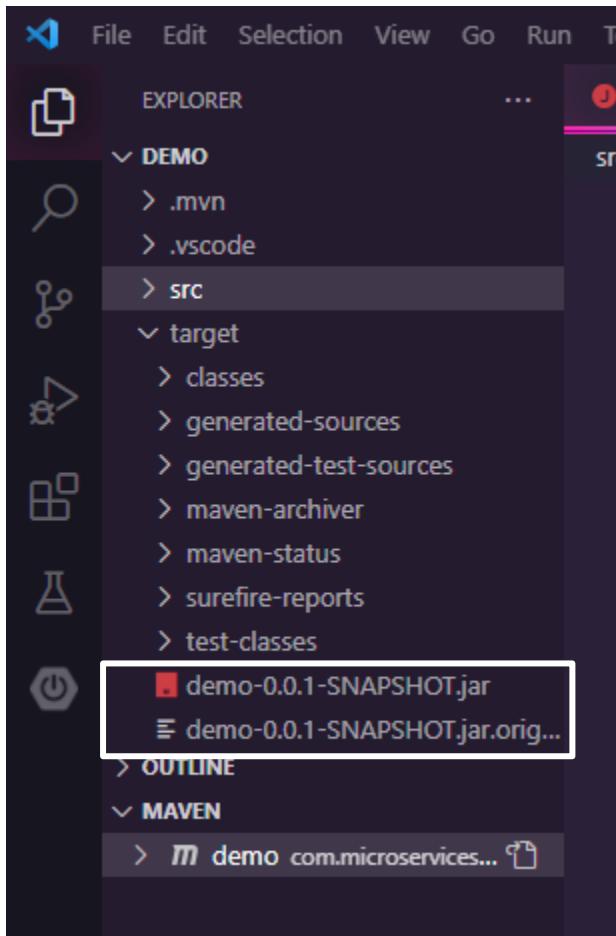
```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL Maven-demo + ×
```

```
Downloaded from central: https://repo.maven.apache.org/maven2/com/google/guava/failureaccess/1.0.1/failureaccess-1.0.1.jar (4.6 kB at 445 B/s)
Downloaded from central: https://repo.maven.apache.org/maven2/com/google/jzobjc/jzobjc-annotations/1.3/jzobjc-annotations-1.3.jar
Downloaded from central: https://repo.maven.apache.org/maven2/org/checkerframework/checker-compat-qual/2.5.5/checker-compat-qual-2.5.5.jar (5.9 kB at 545 B/s)
Downloaded from central: https://repo.maven.apache.org/maven2/com/google/guava/listenablefuture/9999.0-empty-to-avoid-conflict-with-guava/listenablefuture-9999.0-empty-to-avoid-conflict-with-guava.jar (2.2 kB at 204 B/s)
Downloaded from central: https://repo.maven.apache.org/maven2/org/apache/commons/commons-lang3/3.7/commons-lang3-3.7.jar
Downloaded from central: https://repo.maven.apache.org/maven2/com/google/errorprone/error_prone_annotations/2.3.4/error_prone_annotations-2.3.4.jar (14 kB at 1.3 kB/s)
Downloaded from central: https://repo.maven.apache.org/maven2/com/google/jzobjc/jzobjc-annotations/1.3/jzobjc-annotations-1.3.jar (8.8 kB at 803 B/s)
Downloaded from central: https://repo.maven.apache.org/maven2/org/apache/commons/commons-lang3/3.7/commons-lang3-3.7.jar (500 kB at 42 kB/s)
Downloaded from central: https://repo.maven.apache.org/maven2/com/google/guava/guava/28.2-android/guava-28.2-android.jar (2.6 MB at 216 kB/s)
[INFO] Replacing main artifact with repackaged archive
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 01:47 min
[INFO] Finished at: 2022-08-04T20:59:00+05:30
[INFO] -----
```

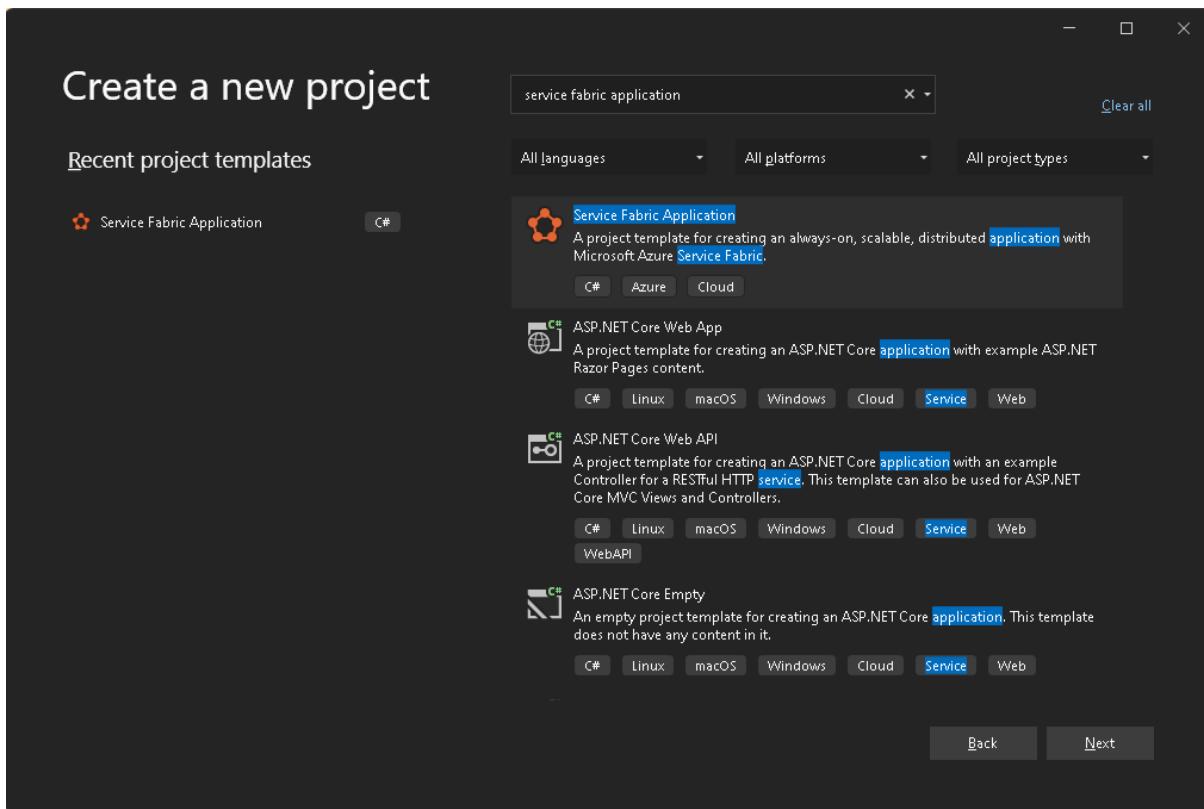
PS F:\RUPAM\CAD\practical2\demo>

In 8 Col 23 Tab Size 4 NTF-B LF Java ⌂ f*

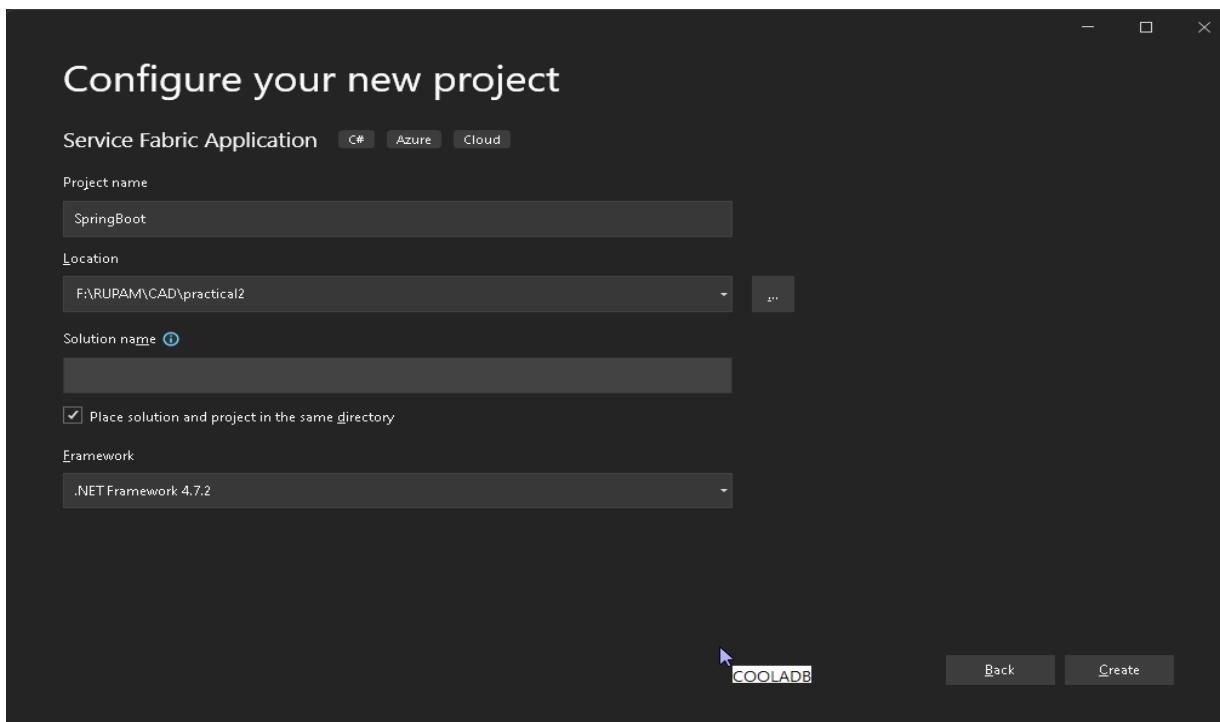
14. Jar File has been created successfully .



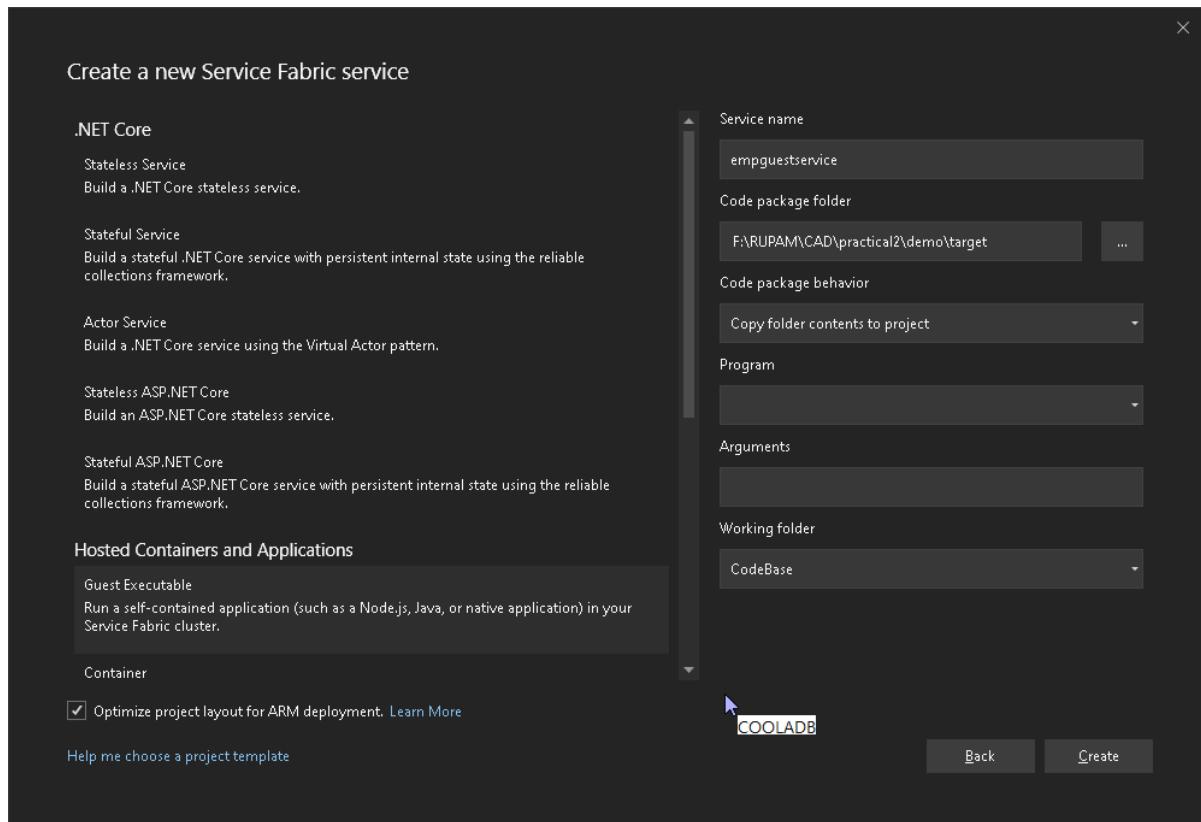
- 15.** Now open the **Visual Studio 17/19/21** in **Admin Mode** and create a new project and select **Services Fabric Application** and Next



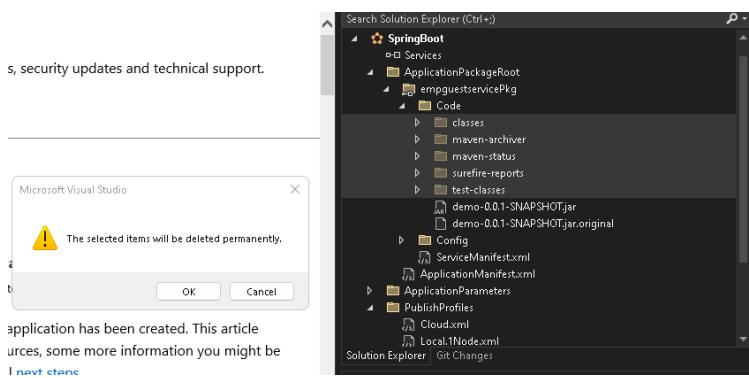
- 16.** Provide the name pf the project “**SpringBoot**” and click in Create



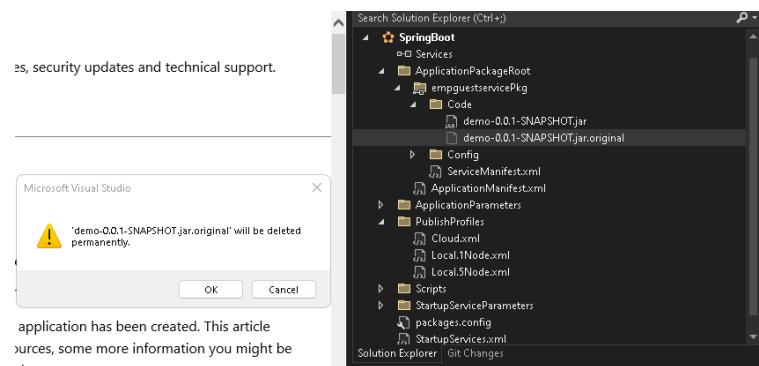
- 17.** Now Select the **Guest Executable** and give the name of the service as “**empguestservice**” & Inside Code package folder, give the path of the folder in which **.jar** file is created. Code package behaviour will be **Copy folder contents to project** and Working folder will be **CodeBase**.



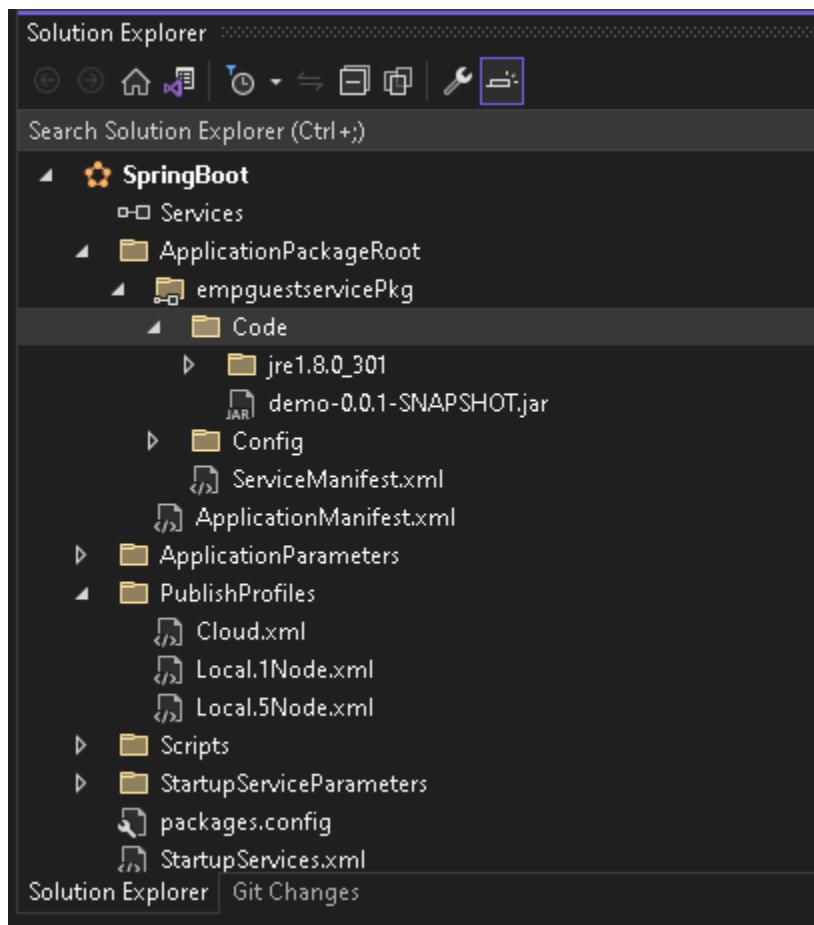
- 18.** Once the project is created, **Delete** the following selected folders.



19. Also Delete the demo-0.0.1-SNAPSHOT.jar.original.

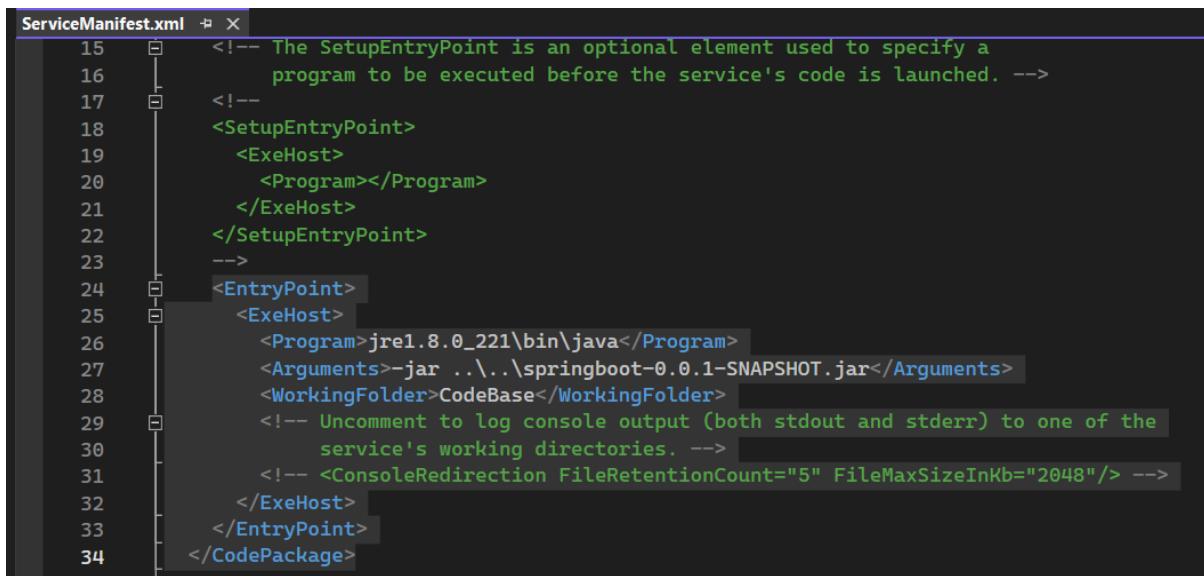


20. Now, we also need to upload the runtime to run the JAR. Generally, it resides in the JDK installation folder (C:\Program Files\Java). Paste it in the Code folder,



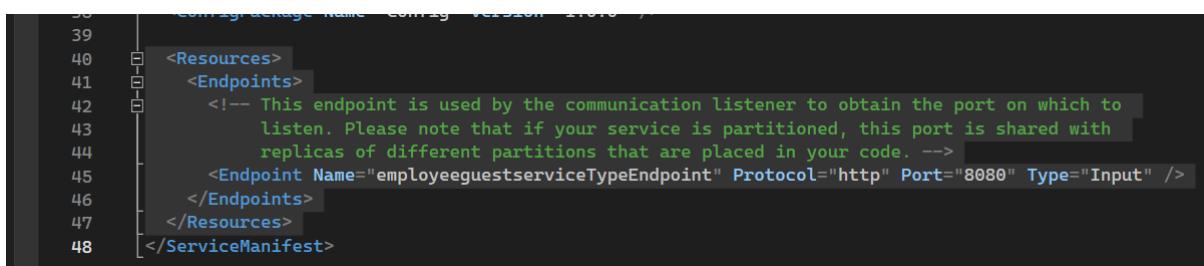
21. Open ServiceManifest.xml and set the following values.

```
<EntryPoint>
  <ExeHost>
    <Program>jre1.8.0_221\bin\java</Program>
    <Arguments>-jar ..\..\springboot-0.0.1-SNAPSHOT.jar</Arguments>
    <WorkingFolder>CodeBase</WorkingFolder>
      <!-- Uncomment to log console output (both stdout and stderr) to one of the service's working
          directories. -->
      <!-- <ConsoleRedirection FileRetentionCount="5" FileMode="FileMaxSizeInKb="2048"/> -->
  </ExeHost>
</EntryPoint>
</CodePackage>
```



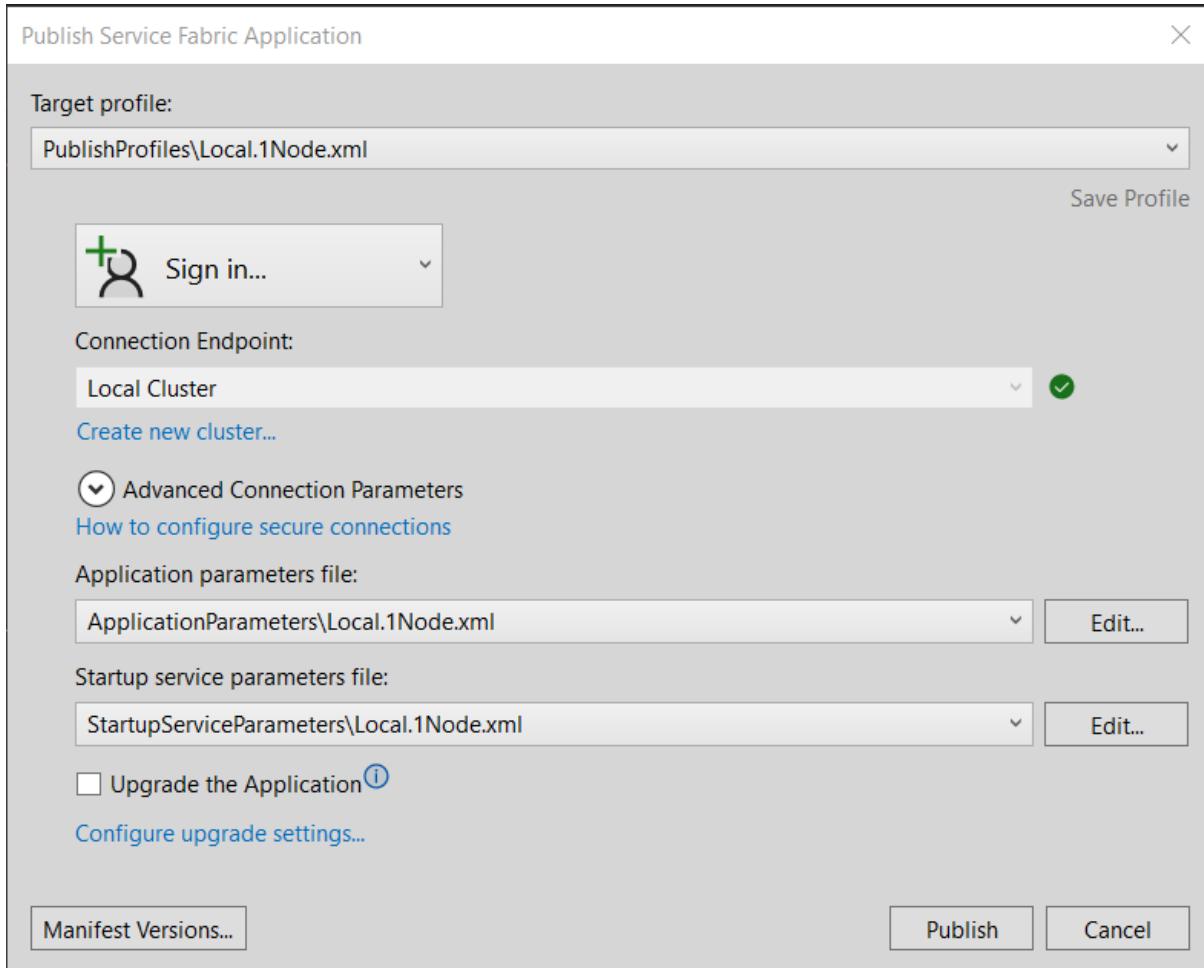
```
ServiceManifest.xml + X
15   <!-- The SetupEntryPoint is an optional element used to specify a
16       program to be executed before the service's code is launched. -->
17   <!--
18     <SetupEntryPoint>
19       <ExeHost>
20         <Program></Program>
21       </ExeHost>
22     </SetupEntryPoint>
23   -->
24   <EntryPoint>
25     <ExeHost>
26       <Program>jre1.8.0_221\bin\java</Program>
27       <Arguments>-jar ..\..\springboot-0.0.1-SNAPSHOT.jar</Arguments>
28       <WorkingFolder>CodeBase</WorkingFolder>
29       <!-- Uncomment to log console output (both stdout and stderr) to one of the
30           service's working directories. -->
31       <!-- <ConsoleRedirection FileRetentionCount="5" FileMode="FileMaxSizeInKb="2048"/> -->
32     </ExeHost>
33   </EntryPoint>
34 </CodePackage>
```

```
<Resources>
  <Endpoints>
    <!-- This endpoint is used by the communication listener to obtain the port on which to listen. Please note
        that if your service is partitioned, this port is shared with replicas of different partitions that are placed in your
        code. -->
    <Endpoint Name="employeeguestserviceTypeEndpoint" Protocol="http" Port="8080" Type="Input" />
  </Endpoints>
</Resources>
```



```
38   <!-- config:age: Name: config: version: 1.0.0 -->
39
40   <Resources>
41     <Endpoints>
42       <!-- This endpoint is used by the communication listener to obtain the port on which to
43           listen. Please note that if your service is partitioned, this port is shared with
44           replicas of different partitions that are placed in your code. -->
45       <Endpoint Name="employeeguestserviceTypeEndpoint" Protocol="http" Port="8080" Type="Input" />
46     </Endpoints>
47   </Resources>
48 </ServiceManifest>
```

22. Now, Publish your application to Service Fabric Cluster.



```
ServiceManifest.xml
Output
Show output from: Service Fabric Tools
Started executing script 'Publish-ServiceFabricApplication'.
powershell -NonInteractive -NoProfile -WindowStyle Hidden -ExecutionPolicy Bypass -Command "[void](Connect-ServiceFabricCluster); Import-Module 'C:\Program Files\Microsoft SDKs\Service Fabric\Tools\PSModule\ServiceFabricSDK\ServiceFabricSDK.psm1'; $sfContext = Get-ServiceFabricClusterContext; $sfContext | Get-Member"
WARNING: The names of some imported commands from the module 'ServiceFabricSDK' include unapproved verbs that might
make them less discoverable. To find the commands with unapproved verbs, run the Import-Module command again with the
Verbose parameter. For a list of approved verbs, type Get-Verb.
Creating application...
ApplicationName      : fabric:/EmployeeSpringAsGuest
ApplicationTypeName   : EmployeeSpringAsGuestType
ApplicationTypeVersion: 1.0.0
ApplicationParameters : { "_WFDebugParams_ = [" ] }

ServiceKind          : Stateless
PartitionScheme      : Singleton
PlacementConstraints : None
ScalingRules          : {}
ServiceTags          : { TagsRequiredToPlace = TagsRequiredToRun; }
PlacementPolicies    : {}
LoadMetrics           : {}
IsDefaultMoveCostSpecified: False
DefaultMoveCost       : None
CorrelatedServices    : {}
InstanceLifecycleDescription: RestoreReplicaLocationAfterUpgrade;
InstanceCount          : 1
MinInstanceCount       : 1
MinInstancePercentage  : 0
InstanceCloseDelayDuration: 0
InstanceRestartWaitDuration: 0
Kind                 : Stateless
ServiceTypeName       : employeeguestserviceType
ApplicationName       : fabric:/EmployeeSpringAsGuest
ServiceName           : fabric:/EmployeeSpringAsGuest/employeeguestservice
PartitionSchemeDescription: System.Fabric.Description.SingletonPartitionSchemeDescription
InitializationData     : {}
Metrics               : {}
Correlations          : {}
ServiceDnsName        : 
```

```

Output
Show output from: Service Fabric Tools
InstanceRestartWaitDuration :
Kind : Stateless
ServiceTypeName : employeguestserviceType
ApplicationName : fabric:/EmployeeSpringAsGuest
ServiceName : fabric:/EmployeeSpringAsGuest/employeguestservice
PartitionSchemeDescription : System.Fabric.Description.SingletonPartitionDescription
InitializationData :
Metrics :
Correlations :
ServiceOrchName :
ServicePackageActivationMode : ExclusiveProcess

Service : fabric:/EmployeeSpringAsGuest/employeguestservice created successfully.
Create application succeeded.

Finished executing script 'Publish-NewServiceFabricApplication'.
Time elapsed: 00:00:07.5062928
Started executing script 'Get-ServiceFabricApplicationStatus'.
powershell NonInteractive -NoProfile -WindowStyle Hidden -ExecutionPolicy Bypass -Command "[void](Connect-ServiceFabricCluster); Import-Module 'C:\Program Files\Microsoft SDKs\Service Fabric\Tools\PSModule\ServiceFabricSDK\ServiceFabric.psm1'; Get-ServiceFabricApplicationStatus"
WARNING: The names of some imported commands from the module 'ServiceFabricSDK' include unapproved verbs that might make them less discoverable. To find the commands with unapproved verbs, run the Import-Module command again with the Verbose parameter. For a list of approved verbs, type Get-Verb.
The application has started.
Service Status:
fabric:/EmployeeSpringAsGuest/employeguestservice is ready.

The application is ready.
Finished executing script 'Get-ServiceFabricApplicationStatus'.
Time elapsed: 00:00:04.2959553
The URL for the launch target is not set or is not an HTTP/HTTPS URL so the browser will not be opened.

```

- 23.** Make sure that the local Service Fabric cluster is up and running. Click F5. Browse the Service Fabric dashboard.

Name	Version	Status	Actions
EmployeeSpringAsGuestType	1.0.0	Available	Unprovision Create

Name	Application Type	Version	Health State	Status
fabric:/EmployeeSpringAsGuest	EmployeeSpringAsGuestType	1.0.0	OK	Ready

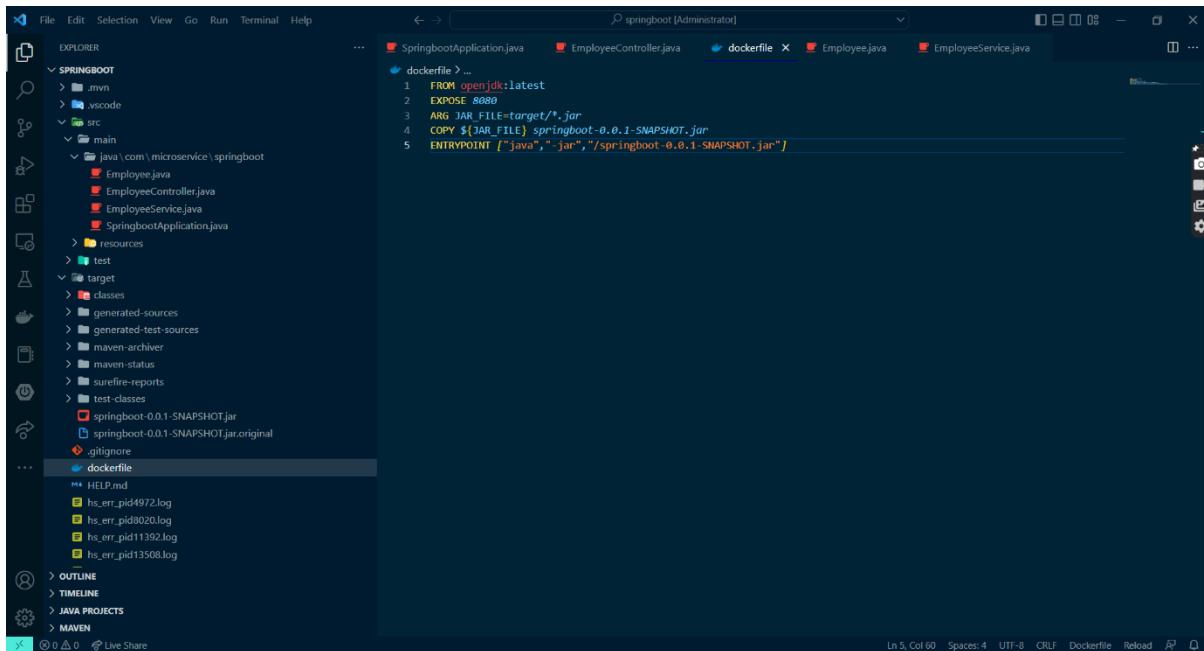
- 24.** Browse <http://localhost:8080> to access your service. In servicemanifest.xml, we specified the service port as 8080; you can browse the same on 8080

Steps :-

We will work on same .jar file that we worked in previous practical.

1. Create **Dockerfile** inside the same Project folder. This **Dockerfile** will have all the contents to build an image. Add the following code in it:

```
FROM openjdk:latest
EXPOSE 8080
ARG JAR_FILE=target/*.jar
COPY ${JAR_FILE} springboot-0.0.1-SNAPSHOT.jar
ENTRYPOINT ["java", "-jar", "/"]
```



2. Go to docs.microsoft.com and Activate Sandbox.

The screenshot shows a Microsoft Learn exercise titled "Exercise - Publish an ASP.NET app from Visual Studio". The page indicates "10 minutes" of time remaining. A message states "Sandbox activated! Time remaining: 4 hr". It also notes that "You have used 1 of 10 sandboxes for today. More sandboxes will be available tomorrow." Below this, it says "You have an ASP.NET Core web application running locally. In this exercise, you'll publish your application to Azure App Service." The main heading is "Publish your ASP.NET Core web app to Azure". The steps listed are:

1. In Solution Explorer, right-click the AlpineSkiHouse project, and select Publish.
2. In the dialog box that appears, select Azure as your publish target, and then select Next to continue.

3. Go to portal.azure.com and Create a Resource of type Container Registry.

The screenshot shows the "Create container registry" wizard on the Azure portal. The "Basics" tab is selected. The "Project details" section includes a "Subscription" dropdown set to "Concierge Subscription" and a "Resource group" dropdown with "Create new" selected. The "Instance details" section includes a "Registry name" field with ".azurecr.io" suffix and a "Location" dropdown set to "West US". Under "Availability zones", there is a checkbox for "Enabled" which is unchecked. A note states: "Availability zones are enabled on premium registries and in regions that support availability zones. [Learn more](#)". At the bottom, there are "Review + create", "< Previous", and "Next: Networking >" buttons.

4. Go to Resources, you will find your resource of **Container Registry** has been created.

The screenshot shows the Microsoft Azure portal interface. The top navigation bar includes links for 'Cisco IOS Images D...', 'Cisco IOS Images fo...', 'Microservice using...', 'Microservices using...', 'Deploy to Kuber...', 'Community edition...', 'iLovePDF | Online P...', 'Implement PhoneB...', and 'Phonetics based Fu...'. The user's email, 'bhagatsd007@ascn.sie...', and 'MICROSOFT LEARN SANDBOX' are also visible. The main content area displays the 'springascontainer' Container Registry resource. The left sidebar lists 'Overview', 'Activity log', 'Access control (IAM)', 'Tags', 'Quick start', 'Events', 'Settings' (with 'Access keys' selected), 'Services' (with 'Repositories', 'Webhooks', and 'Replications'), and 'Networking'. The right panel shows 'Essentials' information: Resource group (move) : learn-2b1ef48e-9115-4081-807f-a6a20827b4f1; Location : West US; Subscription (move) : Concierge Subscription; Subscription ID : 376677b8-4f72-4242-8bba-1d176656158f. It also shows 'Usage' (Included in SKU: 100 GB, Used: 0.00 GB, Additional storage: 0.00 GB) and 'ACR Tasks' (Build, Run, Push and Patch containers in Azure with ACR Tasks. Tasks supports Windows, Linux and ARM with QEMU). Below these are sections for 'Container security integrations' (Microsoft Defender for Cloud) and 'Container logs' (Log Analytics workspace: logAnalytics-12345678).

5. Now, Go to **Access keys** and Enable Admin user. Here, you will find **Login server**, **Username** and **Password** to access your **Container Registry**.

The screenshot shows the 'Access keys' section of the 'springascontainer' Container Registry. The left sidebar is identical to the previous screenshot. The right panel shows the 'Access keys' configuration: Registry name: 'springascontainer'; Login server: 'springascontainer.azurecr.io'; Admin user: 'Enabled'; Username: 'springascontainer'. Two password fields are shown: 'password' (value: eQ97YLSWZARehij0jLi4RH6E+NOzVK0i) and 'password2' (value: 243K-snHFRAQj65qfsRV4RrEjOuSgia9N). A success message 'Successfully enabled admin user' is displayed twice on the right side of the screen.

6. Now, to build an image from the **Dockerfile** we created, execute the following command in command prompt where the Dockerfile exists:

docker build -t my-spring .

The screenshot shows the Visual Studio Code interface with a Spring Boot project open. The left sidebar displays the project structure under 'EXPLORER'. The main editor area shows a 'dockerfile' with the following content:

```
FROM openjdk:latest
EXPOSE 8080
ARG JAR_FILE=target/*.jar
COPY ${JAR_FILE} springboot-0.0.1-SNAPSHOT.jar
ENTRYPOINT ["java","-jar","/springboot-0.0.1-SNAPSHOT.jar"]
```

The 'TERMINAL' tab at the bottom shows the command: PS D:\Siddhesh\Sem 3\CAD\SpringBootPractical\springboot> docker build -t my-spring .

7. After clicking Enter, you will find that the image has been successfully created.

The screenshot shows a Windows terminal window titled "springboot [Administrator]" with several tabs open:

- SpringApplication.java
- EmployeeController.java
- dockerfile
- Employee.java
- EmployeeService.java

The "dockerfile" tab contains the following Dockerfile code:

```
FROM openjdk:latest
EXPOSE 8080
ARG JAR_FILE=target/*.jar
COPY ${JAR_FILE} springboot-0.0.1-SNAPSHOT.jar
ENTRYPOINT ["java", "-jar", "/springboot-0.0.1-SNAPSHOT.jar"]
```

Below the tabs, there are navigation buttons: PROBLEMS, OUTPUT, DEBUG CONSOLE, TERMINAL, and JUPYTER. The TERMINAL tab is selected.

The terminal output shows the build process:

```
PS D:\Siddhesh\See 3\CA\SpringBootPractical\springboot> docker build -t my-spring .
[+] Building 89.0s (8/8) FINISHED
 => [internal] load build definition from dockerfile
 => => transferring dockerfile: 207B
 => [internal] load dockerignore
 => => ignoring 0 files
 => [internal] load metadata for docker.io/library/openjdk:latest
 => [auth] library/openjdk:pull token for registry-1.docker.io
 => [internal] load build context
 => => transferring context: 19,758B
 => [1/2] FROM docker.io/library/openjdk:latest@sha256:d96178b024866d7864coae8bfe25571772225ca10d50c7c795694724658
 => => resolving docker.io/library/openjdk@sha256:d96178b024866d7864coae8bfe25571772225ca10d50c7c795694724658
 => => sha256:118e5140cc0c5f611a243005184e4fd93af526f8526 958B / 958B
 => => sha256:3b2939a28fc8d8a79ec97d26786781ed63f3e0b7814b9798304f279d 12,248B / 12,248B
 => => sha256:3b2939a28fc8d8a79ec97d26786781ed63f3e0b7814b9798304f279d 12,248B / 12,248B
 => => sha256:0bb540a2510735c7571392990be09037813871282ab6a284f85 188,749B / 188,749B
 => => sha256:0bb540a2510735c7571392990be09037813871282ab6a284f85 188,749B / 188,749B
 => => extracting sha256:0bb50c5c024818c7d2abc9b51381985dfcc0ed9845ea60062b28115af012d9b 43,879B / 43,879B
 => => extracting sha256:3b2939a28fc8d8a79ec97d26786781ed63f3e0b7814b97983044f279d
 => => extracting sha256:24546a251b735e75b16695eaf2e339f29990be09037813871282ab6a284f85
 => [1/2] COPY target/*.jar springboot-0.0.1-SNAPSHOT.jar
=> exporting to Image
=> writing manifest
=> writing Image sha256:1d4baef0eb8f83efad996e71049782c2d1caf93052973ae4165158798cd
=> naming to docker.io/library/my-spring
```

At the bottom, a message says "Use 'docker scan' to run Snyk tests against images to find vulnerabilities and learn how to fix them".

8. Now, we will **Login** into our **Container Registry Server** by executing following command:

```
docker login springascontainer.azurecr.io -u springascontainer -p eQ97YLSWZARehij0jLi4RH6E+NozVK0i
```

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL JUPYTER powershell + ↻  
PS D:\Siddhesh\Sem 3\CAD\SpringBootPractical\springboot> docker login springascontainer.azurecr.io -u springascontainer -p eQ97YLSwZArhijojLi4RH6E+NOzVK0I  
WARNING! Using --password via the CLI is insecure. Use --password-stdin.  
Login Succeeded  
PS D:\Siddhesh\Sem 3\CAD\SpringBootPractical\springboot>
```

9. Now, we will tag our image with **Container Registry Server Name** and push it into **Azure** by executing following command:

To Tag - docker tag my-spring springascontainer.azurecr.io/my-spring

To Push - `docker push springascontainer.azurecr.io/my-spring`

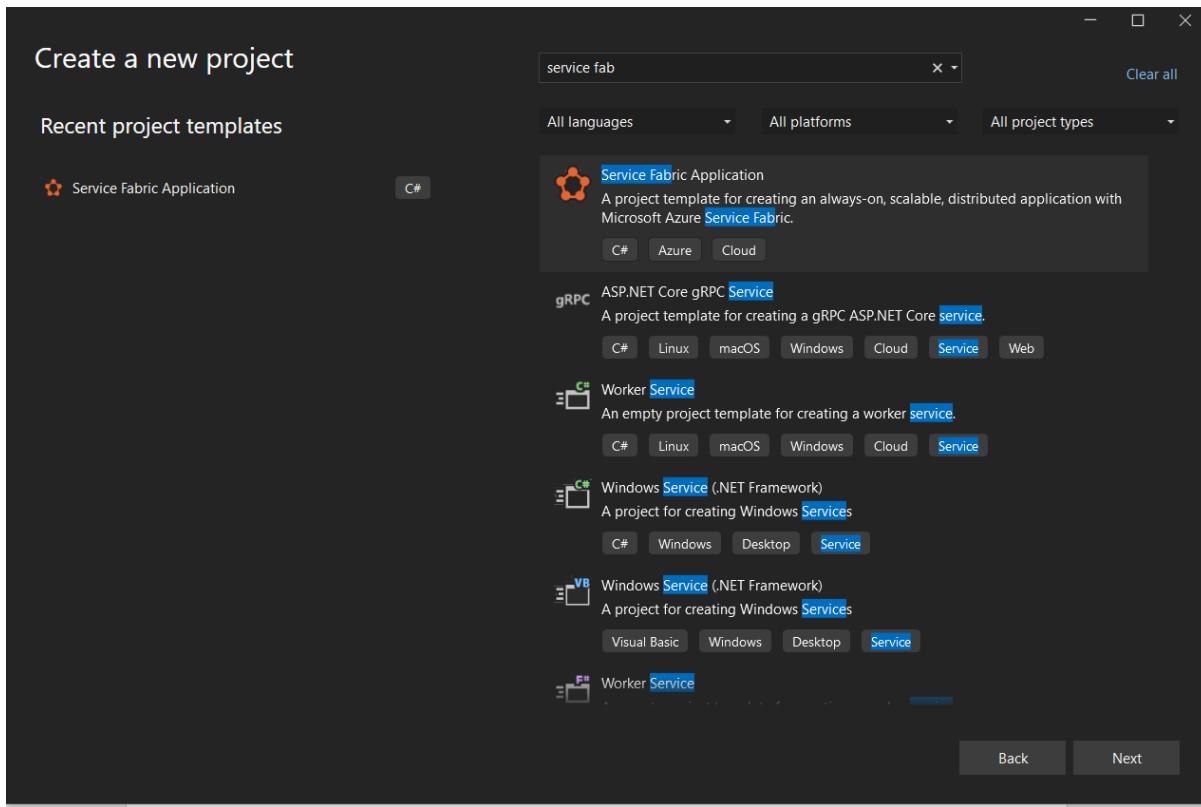
```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL JUPYTER D powershell + □ 🖌 ▲ 🔍

PS D:\Siddhesh\Sem 3\CAD\SpringBootPractical\springboot> docker login springascontainer.azurecr.io -u springascontainer -p e997YLSwZARehij0jLi4RH6e+NOzVKoi
WARNING! Using --password via the CLI is insecure. Use --password-stdin.
Login Succeeded
PS D:\Siddhesh\Sem 3\CAD\SpringBootPractical\springboot> docker tag my-spring springascontainer.azurecr.io/my-spring
PS D:\Siddhesh\Sem 3\CAD\SpringBootPractical\springboot> docker images
REPOSITORY           TAG      IMAGE ID   CREATED        SIZE
my-spring           latest   1d4baedfe0bc  6 minutes ago  486MB
springascontainer.azurecr.io/my-spring  latest   1d4baedfe0bc  6 minutes ago  486MB
PS D:\Siddhesh\Sem 3\CAD\SpringBootPractical\springboot> docker push springascontainer.azurecr.io/my-spring
Using default tag: latest
The push refers to repository [springascontainer.azurecr.io/my-spring]
21fb78da0c37: Pushed
fbda31bd63cf: Pushed
59a7b98be63f: Pushed
4dc0c6342b0f5: Pushed
latest: digest: sha256:a857a7f3d79ff34df26fac5cc40a49327b9d953242ec4579673f9945b5768e77 size: 1166
PS D:\Siddhesh\Sem 3\CAD\SpringBootPractical\springboot>
```

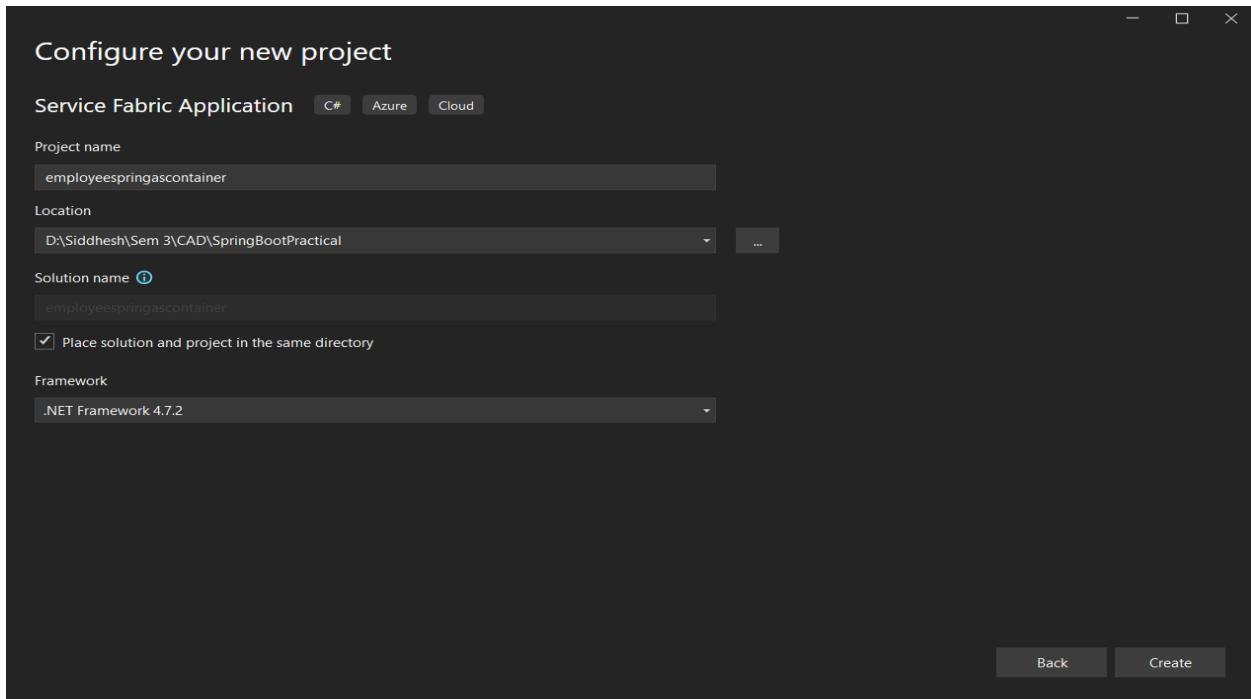
- 10.** Inside **Azure Portal**, navigate to **Repositories** section and you will find that your image has been successfully pushed from local system to **Azure Container Registry**.

The screenshot shows the Microsoft Azure portal interface. The top navigation bar includes links for 'Exercise - Publish an ASP.NET Core application to Azure', 'springascontainer - Microsoft Azure', and several other documentation and service links. The main content area is titled 'springascontainer | Repositories'. On the left, a sidebar lists 'Overview', 'Activity log', 'Access control (IAM)', 'Tags', 'Quick start', 'Events', 'Settings' (with 'Networking' expanded to show 'Microsoft Defender for Cloud' and 'Locks'), and 'Services' (with 'Repositories' selected). The main pane displays a search bar, a refresh button, and a list of repositories. The list shows one repository named 'my spring'. The top right corner of the main pane shows a user profile for 'bhagatfsl070@asusnet.in'.

11. Since the container image is ready and uploaded in Azure Container Registry, let's create a Service Fabric project to deploy the container to the local Service Fabric cluster.

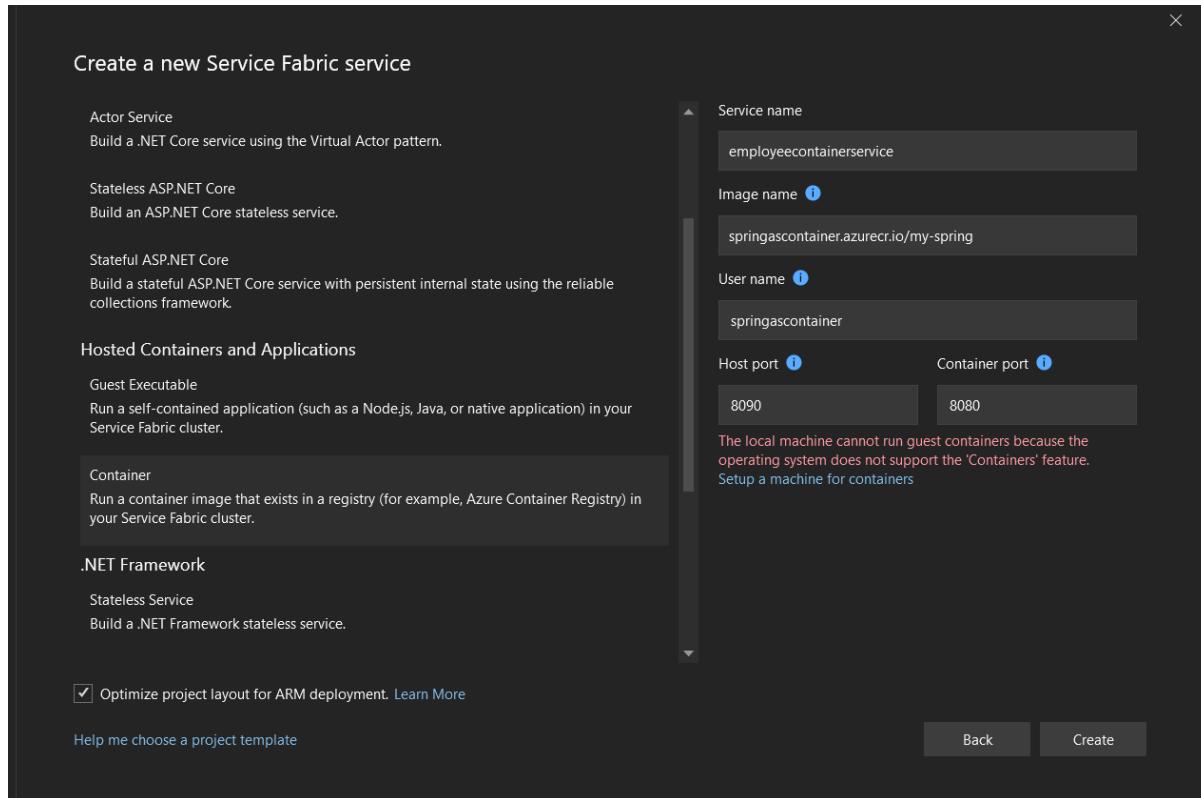


12. Give your project a meaningful name.



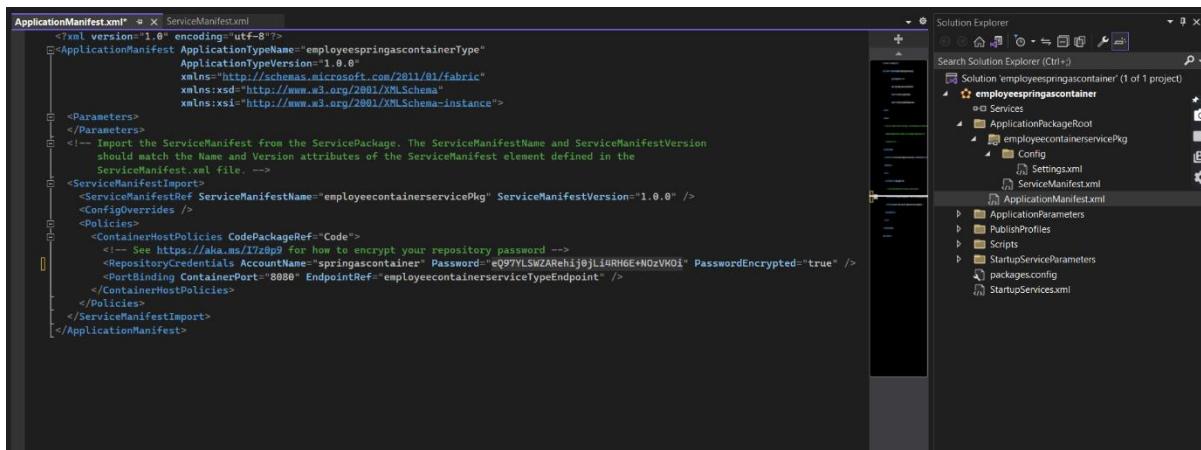
13. In New Service Fabric Service, select the following.
- Service Name: **employeecontainerservice**
 - Image Name: **springascontainer.azurecr.io/my-spring**
 - User Name: Your username in the Azure Container Registry i.e. **springascontainer**
 - Host Port: **8090**
 - Container Port: **8080**

Click on Create

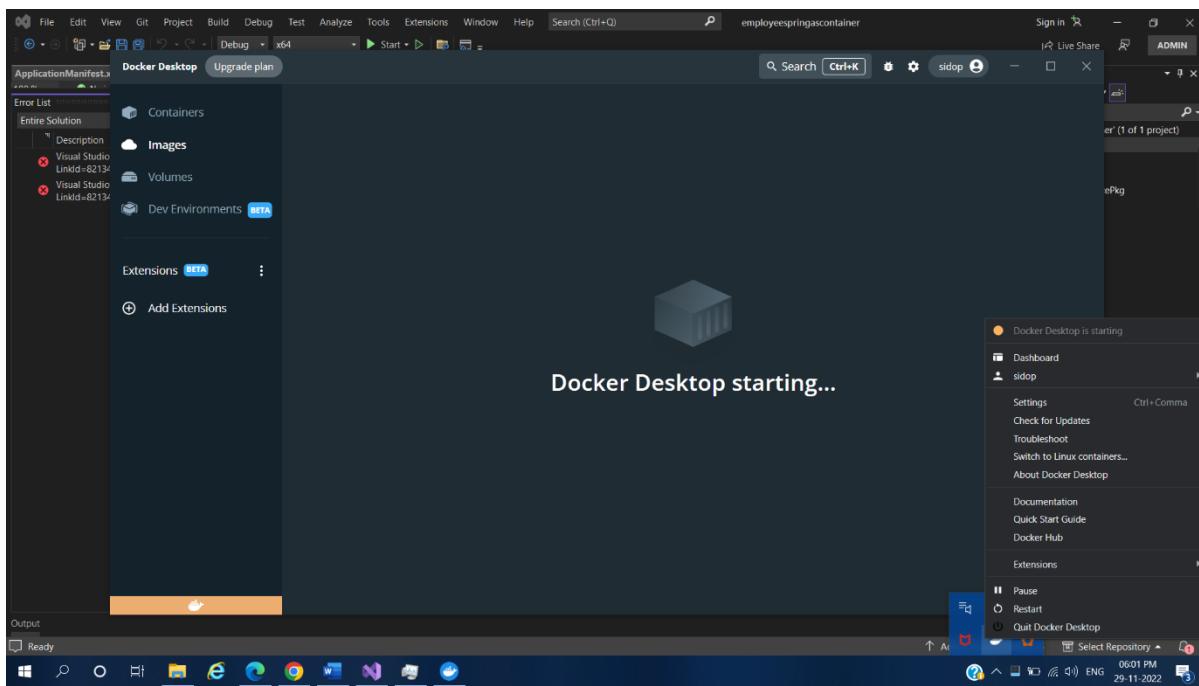


14. Once the solution is created, open the **ApplicationManifest.xml**. Specify the right **password** for the admin user:

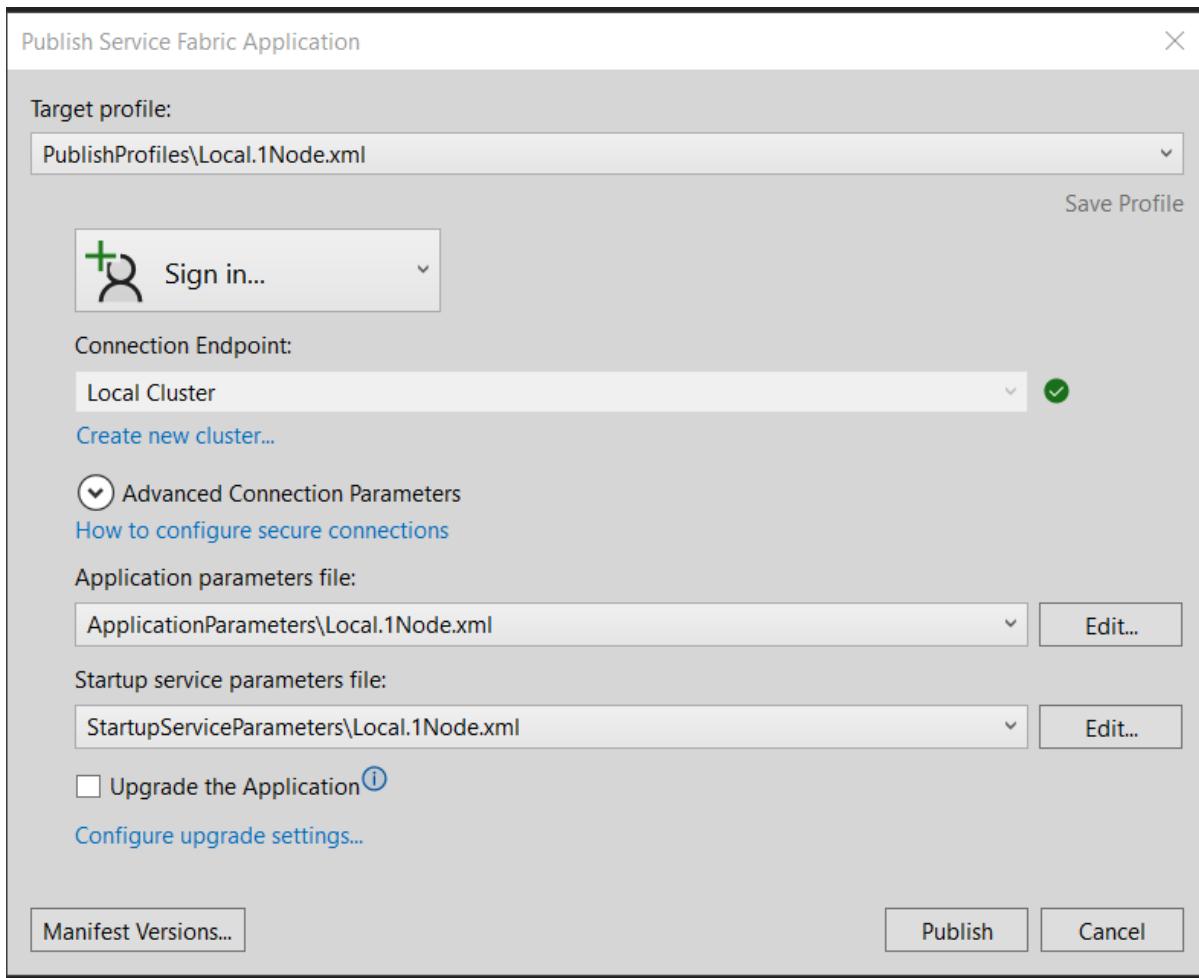
```
<RepositoryCredentials AccountName="springascontainer"  
Password="3TWkwuCxN1aCRa6PeLpoj0pDAdsp5o5giFe1j5YNG0+ACRDoEnfM"  
PasswordEncrypted="true" />
```



15. Select **Switch to Windows container...** in Docker Desktop.

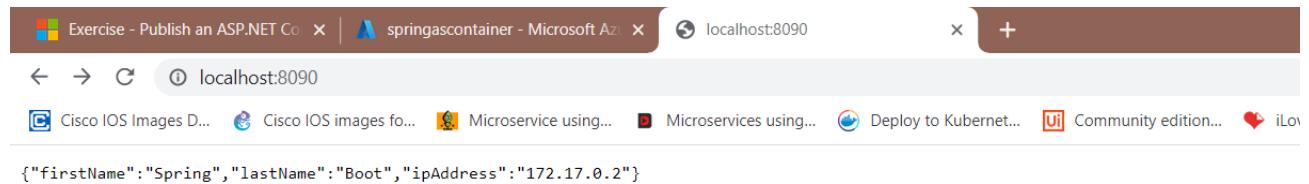


16. Now, Publish your project to the Local Service Fabric Cluster.



```
Output
Show output from: Build
2>InstancesCount : -1
2>MinInstanceCount : 1
2>MinInstancePercentage : 0
2>InstanceCloseDelayDuration :
2>InstanceRestartWaitDuration :
2>Kind : Stateless
2>ServiceTypeName : employeecontainerserviceType
2>ApplicationName : fabric:/employeespringcontainer
2>ServiceName : fabric:/employeespringcontainer/employeecontainerservice
2>PartitionSchemeDescription : System.Fabric.Description.SingletonPartitionSchemeDescription
2>InitializationData :
2>Metrics :
2>Operations :
2>ServiceOnName :
2>ServicePackageActivationMode : ExclusiveProcess
2>
2>Service : fabric:/employeespringcontainer/employeecontainerservice created successfully.
2>Create application succeeded.
2>
2>Finished executing script 'Deploy-FabricApplication.ps1'.
2>Time elapsed: 00:00:16.725598
2>Starting to execute script 'Get-ServiceFabricApplicationStatus'.
2>[WindowsPowerShell -NonInteractive -NoProfile -WindowStyle Hidden -ExecutionPolicy Bypass -Command "[void](Connect-ServiceFabricCluster -TimeoutSec:10 -WarningAction:'SilentlyContinue'); Import-Module 'C:\Program Files\Microsoft SDKs\Service Fabric\Tools\Windows PowerShell\*'; Get-ServiceFabricApplicationStatus"]'
2>WARNING: The names of some reported commands from the module 'ServiceFabricSDK' include unapproved verbs that might
2>make them less discoverable. To find the commands with unapproved verbs, run the Import-Module command again with the
2>Verbose parameter. For a list of approved verbs, type Get-Verb.
2>The application has started.
2>Service Status:
2>fabric:/employeespringcontainer/employeecontainerservice is ready.
2>
2>The application is ready.
2>Finished executing script 'Get-ServiceFabricApplicationStatus'.
2>Time elapsed: 00:00:16.726011
=====
===== Build: 1 succeeded or up-to-date, 0 failed, 0 skipped ======
===== Elapsed: 00:46.346 =====
===== Publish: 1 succeeded, 0 failed, 0 skipped ======
===== Elapsed: 00:46.348 =====
```

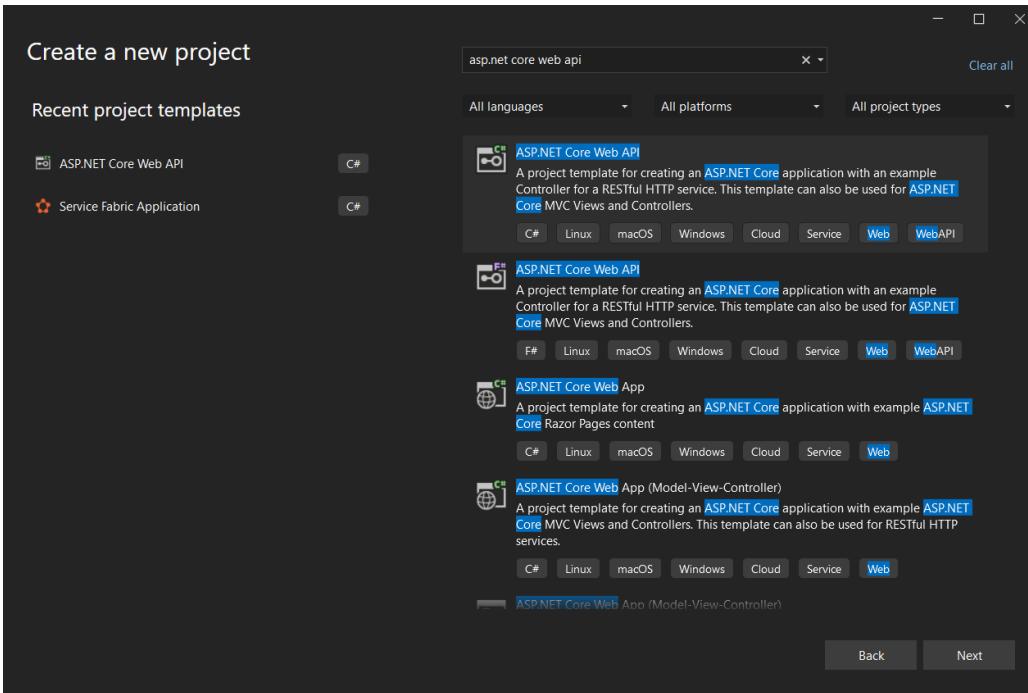
17. Browse to **http://localhost:8090/** to access your service. You get the response as shown below, which is served from the container run by Service Fabric.



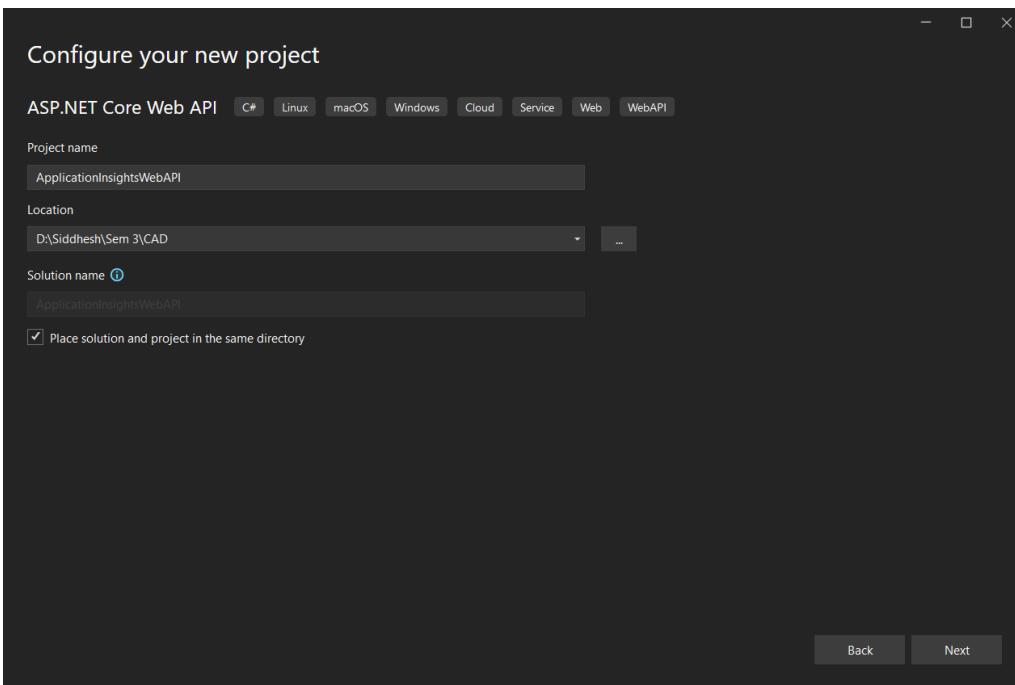
A screenshot of a web browser window. The address bar shows 'localhost:8090'. The page content displays a single line of JSON text: {"firstName": "Spring", "lastName": "Boot", "ipAddress": "172.17.0.2"}

Steps :-

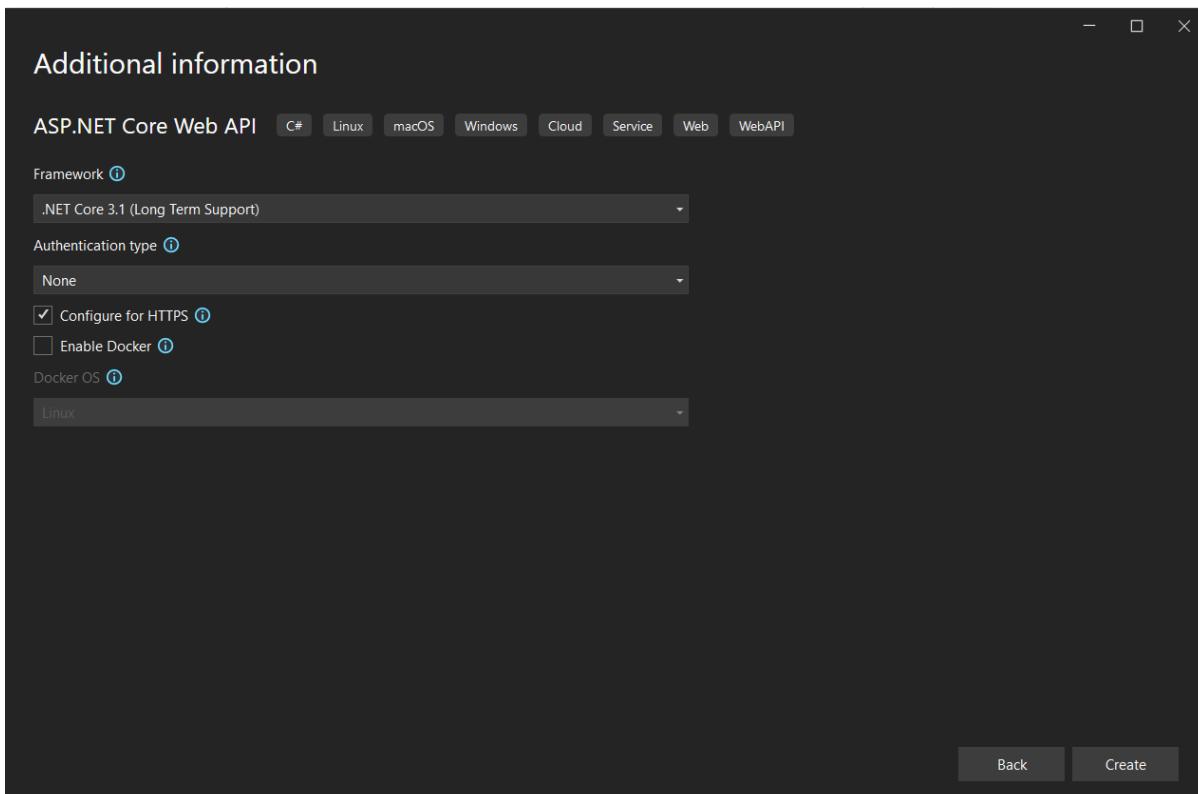
1. Open Visual Studio and Create a new project of the type **ASP.NET Core Web API**.



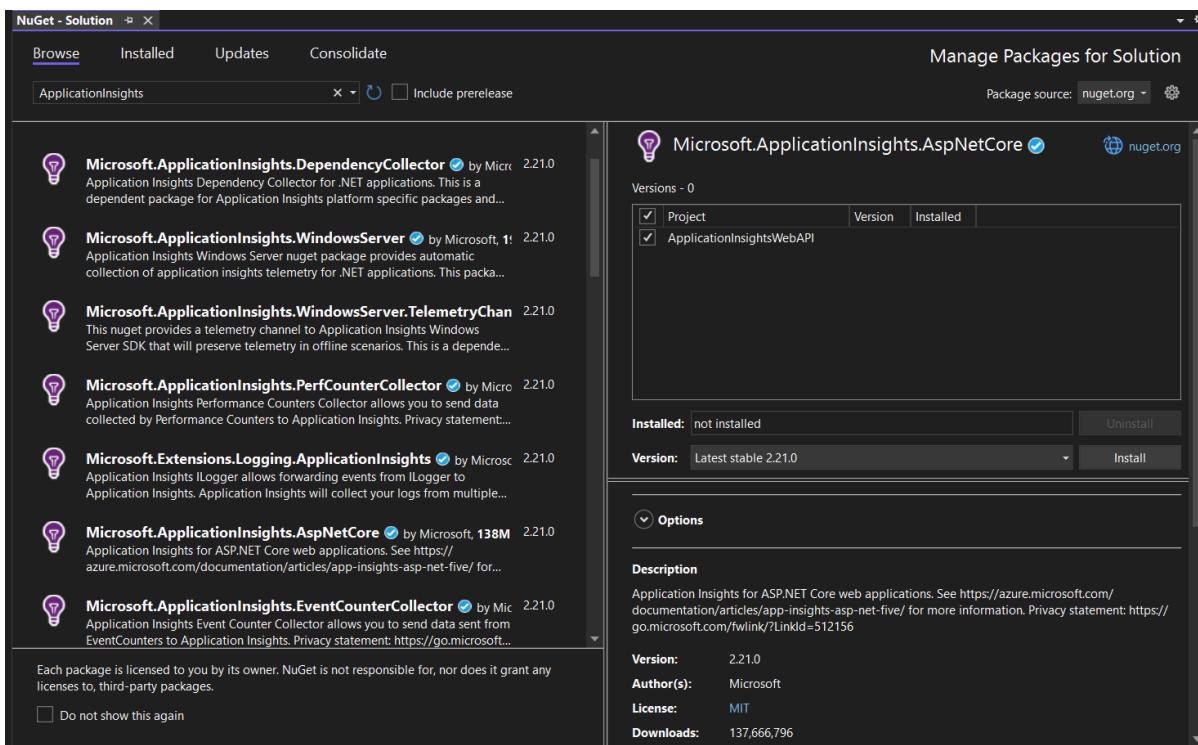
2. Give it a Meaningful name.



3. Select .NET Core 3.1 as Framework for this project and Click on Create.



4. Go inside Nuget Package Manager and Install Microsoft.ApplicationInsights.AspNetCore.



5. Inside Startup.cs file add the Following Code:

```
services.AddApplicationInsightsTelemetry();
```

```
Startup.cs* - ApplicationInsightsWebAPI - ApplicationInsightsWebAPI.Startup - ConfigureServices(IServiceCollection services)
```

```
1  using Microsoft.AspNetCore.Builder;
2  using Microsoft.AspNetCore.Hosting;
3  using Microsoft.AspNetCore.HttpsPolicy;
4  using Microsoft.AspNetCore.Mvc;
5  using Microsoft.Extensions.Configuration;
6  using Microsoft.Extensions.DependencyInjection;
7  using Microsoft.Extensions.Hosting;
8  using Microsoft.Extensions.Logging;
9  using System;
10 using System.Collections.Generic;
11 using System.Linq;
12 using System.Threading.Tasks;
13
14 namespace ApplicationInsightsWebAPI
15 {
16     public class Startup
17     {
18         public Startup(IConfiguration configuration)
19         {
20             Configuration = configuration;
21         }
22
23         public IConfiguration Configuration { get; }
24
25         // This method gets called by the runtime. Use this method to add services to the container.
26         public void ConfigureServices(IServiceCollection services)
27         {
28             services.AddApplicationInsightsTelemetry();
29             services.AddControllers();
30         }
31
32         // This method gets called by the runtime. Use this method to configure the HTTP request pipeline.
33         public void Configure(IApplicationBuilder app, IWebHostEnvironment env)
```

6. Go to docs.microsoft.com and Activate Sandbox.

The screenshot shows a Microsoft Learn exercise titled "Exercise - Publish an ASP.NET app from Visual Studio". The page includes a navigation bar with links like "Training", "Products", "Roles", "Learning Paths", "Courses", "Educator Center", "Student Hub", "FAQ & Help", and a search bar. The main content area displays the exercise title, duration (10 minutes), and a note about sandbox activation. It also contains instructions for publishing an ASP.NET Core web app to Azure.

Exercise - Publish an ASP.NET app from Visual Studio

10 minutes

Sandbox activated! Time remaining: 4 hr

You have used 1 of 10 sandboxes for today. More sandboxes will be available tomorrow.

You have an ASP.NET Core web application running locally. In this exercise, you'll publish your application to Azure App Service.

Publish your ASP.NET Core web app to Azure

1. In Solution Explorer, right-click the **AlpineSkiHouse** project, and select **Publish**.
2. In the dialog box that appears, select **Azure** as your publish target, and then select **Next** to continue.

7. Go to portal.azure.com and Create a Resource of type Application Insights.

The screenshot shows the Microsoft Azure portal interface. The user is navigating through the 'Create a resource' process under the 'Marketplace' category, specifically for 'Application Insights'. The 'Basics' tab is selected. In the 'PROJECT DETAILS' section, the 'Subscription' is set to 'Concierge Subscription' and the 'Resource Group' is 'learn-58a5791e-cc31-4566-9a0c-1db4f1306e41'. Under 'INSTANCE DETAILS', the 'Name' is 'ApplicationInsight' and the 'Region' is '(US) West US'. The 'Resource Mode' is set to 'Workspace-based'. In the 'WORKSPACE DETAILS' section, there are buttons for 'Review + create', 'Previous', and 'Next : Tags >'. The status bar at the bottom indicates the date and time as 13-12-2022 09:55 PM.

8. Click on Go To Resource.

The screenshot shows the Microsoft Azure portal with the 'Microsoft.AppInsights | Overview' page. A prominent message states 'Your deployment is complete' with a green checkmark icon. Deployment details are listed: 'Deployment name: Microsoft.AppInsights', 'Subscription: Concierge Subscription', and 'Resource group: learn-58a5791e-cc31-4566-9a0c-1db4f1306e41'. To the right, a sidebar displays a success message: 'Deployment succeeded' with a green checkmark icon, stating that the deployment to resource group 'learn-58a5791e-cc31-4566-9a0c-1db4f1306e41' was successful. Other sections in the sidebar include 'Cost Management', 'Microsoft Defender for Cloud', 'Free Microsoft tutorials', and 'Work with an expert'. The status bar at the bottom indicates the date and time as 13-12-2022 09:57 PM.

9. You will find your resource of **Application Insights** has been created.

The screenshot shows the Microsoft Azure portal interface. The main title bar says "ApplicationInsight - Microsoft Azure". Below it, the URL is "portal.azure.com/#learn/docs.microsoft.com/resource/subscriptions/1c2e5eea-af00-4e37-b297-7d317d875211/resourcegroups/learn-58a5791e-cc31-4566-9a0c-1db4f1306e41/providers/microsoft.insights/components/learn-58a5791e-cc31-4566-9a0c-1db4f1306e41". The top navigation bar includes links for "Cisco IOS Images", "Microservice using...", "Microservices using...", "Deploy to Kubernetes", "Community edition...", "iLovePDF | Online P...", "Implement Phonetic...", and "Phonetics based Fu...". The user's email "bhagatsd0070@json.sie..." and "MICROSOFT LEARN SANDBOX" are also visible.

The left sidebar shows the "Microsoft Azure" logo and a search bar. Under "Home > MicrosoftAppInsights | Overview", there are sections for "ApplicationInsight" (with a star icon), "Overview", "Activity log", "Access control (IAM)", "Tags", "Diagnose and solve problems", "Investigate" (with sub-options like "Application map", "Smart detection", "Live metrics", "Transaction search", "Availability", "Failures", "Performance", "Troubleshooting guides (preview)", and "Monitoring"), and "Alerts".

The main content area is titled "Essentials" and displays resource group information: "Resource group (move) : learn-58a5791e-cc31-4566-9a0c-1db4f1306e41", "Location : West US", "Subscription (move) : Conierge Subscription", "Subscription ID : 1c2e5eea-af00-4e37-b297-7d317d875211", and "Tags (edit) : Click here to add tags". It also shows the "Instrumentation Key" and "Connection String". A warning message states: "Classic Application Insights is deprecated and will be retired in February 2024. Migrate this resource to Workspace-based Application Insights to gain support for all of the capabilities of Log Analytics, including Customer-Managed Keys and Commitment Tiers. Click here to learn more and migrate in a few clicks." Below this, there is a "Show data for last:" dropdown with options: 30 minutes, 1 hour, 6 hours, 12 hours, 1 day, 3 days, 7 days, and 30 days. Three charts are displayed: "Failed requests", "Server response time", and "Server requests". The bottom status bar shows the date and time: "09:57 PM 13-12-2022".

10. On the Azure portal, inside **Application Insight** resource you will find **Instrumentation Key**; copy and paste it into **appsettings.json** file as:

```
"ApplicationInsights": {  
    "Instrumentation Key" : "Your Key"  
}
```

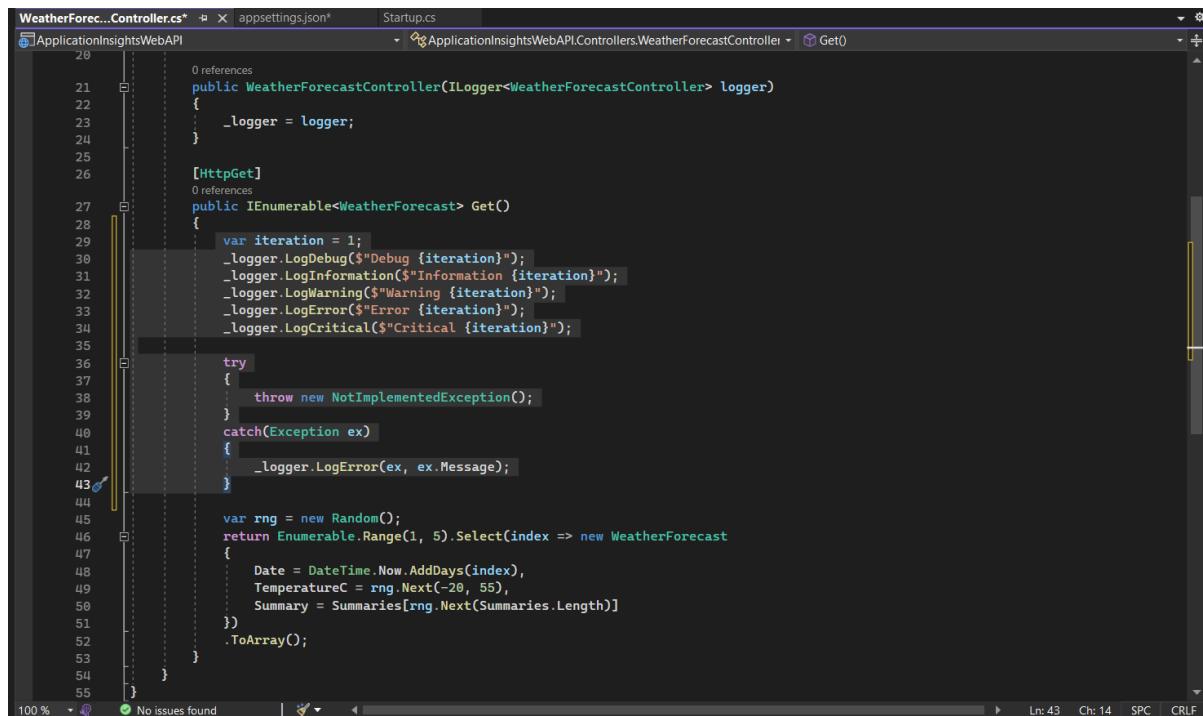
The screenshot shows the Visual Studio code editor with the file "appsettings.json" open. The schema is defined as "Schema: https://json.schemastore.org/appsettings.json". The JSON code is as follows:

```
1  {  
2      "Logging": {  
3          "LogLevel": {  
4              "Default": "Information",  
5              "Microsoft": "Warning",  
6              "Microsoft.Hosting.Lifetime": "Information"  
7          }  
8      },  
9      "AllowedHosts": "*",  
10     "ApplicationInsights": {  
11         "InstrumentationKey": "53842b9d-6946-4f80-89f7-d6acba9ff54"  
12     }  
13 }  
14
```

11. Inside **WeatherForecastController.cs** file add the following code:

```
var iteration = 1;
_logger.LogDebug($"Debug {iteration}");
_logger.LogInformation($"Information {iteration}");
_logger.LogWarning($"Warning {iteration}");
_logger.LogError($"Error {iteration}");
_logger.LogCritical($"Critical {iteration}");

try
{
    throw new NotImplementedException();
}
catch(Exception ex)
{
    _logger.LogError(ex, ex.Message);
}
```



The screenshot shows the Visual Studio code editor with the **WeatherForecastController.cs** file open. The code editor displays the following C# code:

```
WeatherForec...Controller.cs*  appsettings.json*  Startup.cs
ApplicationInsightsWebAPI  ApplicationInsightsWebAPI.Controllers.WeatherForecastController  Get()

public WeatherForecastController	ILogger<WeatherForecastController> logger
{
    _logger = logger;
}

[HttpGet]
public IEnumerable<WeatherForecast> Get()
{
    var iteration = 1;
    _logger.LogDebug($"Debug {iteration}");
    _logger.LogInformation($"Information {iteration}");
    _logger.LogWarning($"Warning {iteration}");
    _logger.LogError($"Error {iteration}");
    _logger.LogCritical($"Critical {iteration}");

    try
    {
        throw new NotImplementedException();
    }
    catch(Exception ex)
    {
        _logger.LogError(ex, ex.Message);
    }

    var rng = new Random();
    return Enumerable.Range(1, 5).Select(index => new WeatherForecast
    {
        Date = DateTime.Now.AddDays(index),
        TemperatureC = rng.Next(-20, 55),
        Summary = Summaries[rng.Next(Summaries.Length)]
    })
    .ToArray();
}
```

The code editor interface includes tabs for **WeatherForec...Controller.cs***, **appsettings.json***, and **Startup.cs**. The **WeatherForecastController.cs** tab is active. The code is syntax-highlighted, and the cursor is positioned at the end of line 43. The status bar at the bottom shows "100 %", "No issues found", "Ln: 43 Ch: 14 SPC", and "CR/LF".

12. Click on **Run**.



OUTPUT:

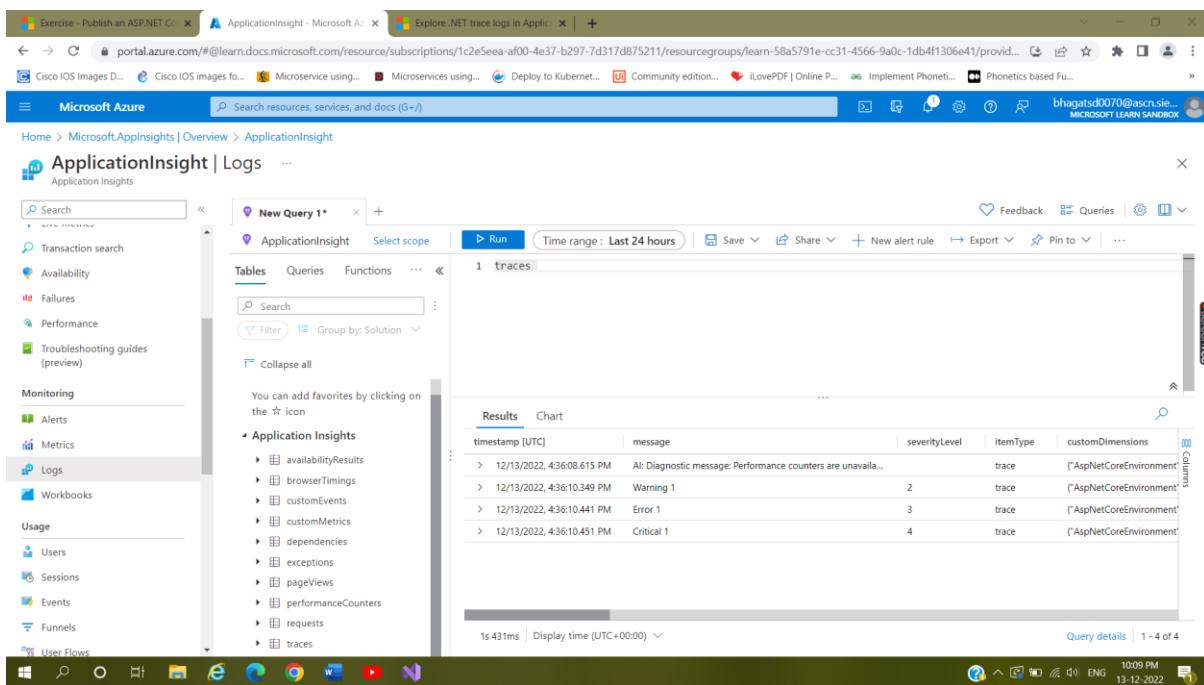


A screenshot of a Microsoft Edge browser window. The address bar shows the URL `https://localhost:44356/weatherforecast`. The page content displays a JSON array of weather forecast data:

```
[{"date": "2022-12-14T22:06:11.0240364+05:30", "temperatureC": 9, "temperatureF": 48, "summary": "Hot"}, {"date": "2022-12-15T22:06:11.0244575+05:30", "temperatureC": -7, "temperatureF": 20, "summary": "Bracing"}, {"date": "2022-12-16T22:06:11.0244594+05:30", "temperatureC": 36, "temperatureF": 96, "summary": "Scorching"}, {"date": "2022-12-17T22:06:11.0244593+05:30", "temperatureC": 18, "temperatureF": 64, "summary": "Bracing"}, {"date": "2022-12-18T22:06:11.0244594+05:30", "temperatureC": 32, "temperatureF": 89, "summary": "Chilly"}]
```

13. Inside ApplicationInsight resource on Azure Portal, inside Monitoring > Logs > Query section; type

traces and click on Run



A screenshot of the Microsoft Azure Application Insights Logs query interface. The left sidebar shows navigation options like Alerts, Metrics, and Logs (which is selected). The main area shows a search bar and a query editor with the text `traces`. Below the editor is a results table with columns: timestamp [UTC], message, severityLevel, itemType, and customDimensions. The table contains four rows of log entries:

timestamp [UTC]	message	severityLevel	itemType	customDimensions
12/13/2022, 4:36:08.615 PM	AI: Diagnostic message: Performance counters are unavaila...	trace		{"AspNetCoreEnvironment"}
12/13/2022, 4:36:10.349 PM	Warning 1	2	trace	{"AspNetCoreEnvironment"}
12/13/2022, 4:36:10.441 PM	Error 1	3	trace	{"AspNetCoreEnvironment"}
12/13/2022, 4:36:10.451 PM	Critical 1	4	trace	{"AspNetCoreEnvironment"}

14. Type the following and click on Run:

```
traces  
| order by timestamp desc  
| where message == "Critical 1"
```

A screenshot of the Azure Log Analytics workspace. At the top, there are navigation links: Run, Save, Share, New alert rule, Export, Pin to, and more. Below this is a code editor window containing the following Kusto query:

```
1 traces
2 | order by timestamp desc
3 | where message == "Critical 1"
4
```

The main area shows a table titled "Results" with the following columns: timestamp [UTC], message, severityLevel, itemType, and customDimensions. One row is displayed:

timestamp [UTC]	message	severityLevel	itemType	customDimensions
> 12/13/2022, 4:36:10.451 PM	Critical 1	4	trace	{"AspNetCoreEnvi...

On the right side, there are icons for camera, video, and settings.

15. Type exceptions and click on Run

A screenshot of the Microsoft Azure Application Insights Logs interface. The left sidebar shows navigation options: Transaction search, Availability, Failures, Performance, Troubleshooting guides (preview), Monitoring, Alerts, Metrics, Logs (selected), Workbooks, Usage, Users, Sessions, Events, Funnels, and User Flows. The main area shows a "New Query 1" window with the following query:

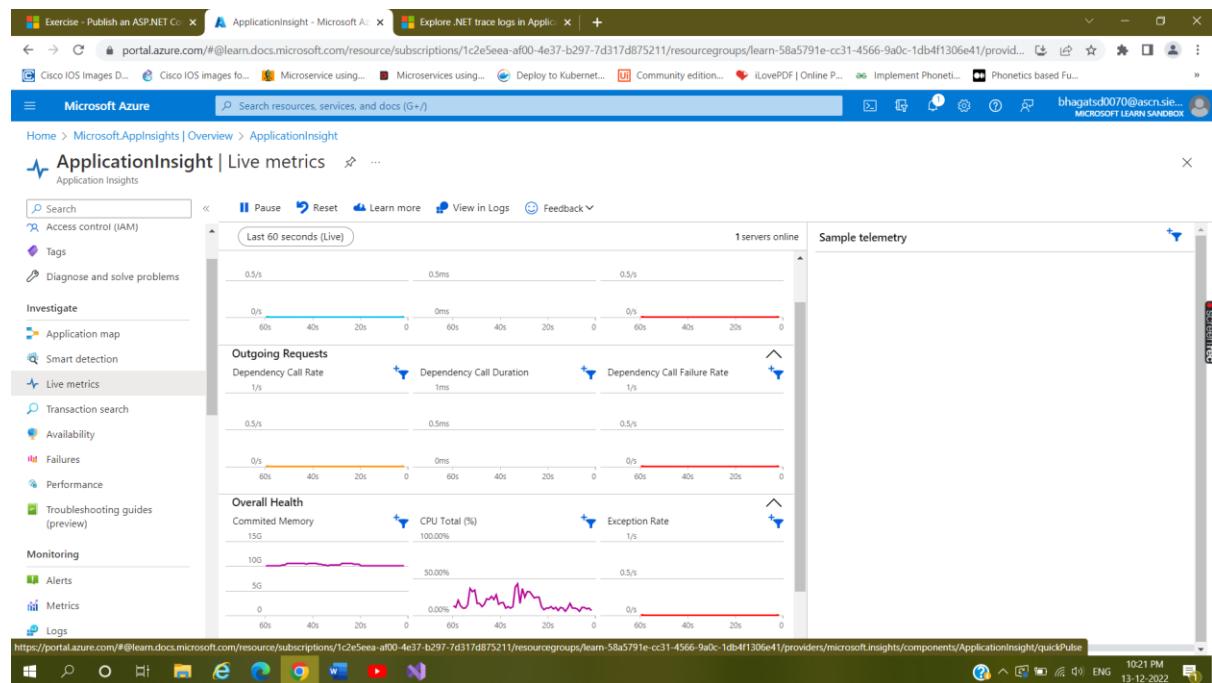
```
1 exceptions
```

The results table has columns: timestamp [UTC], problemId, type, and assembly. One row is shown:

timestamp [UTC]	problemId	type	assembly
> 12/13/2022, 4:36:11.009 PM	System.NotImplementedException at ApplicationInsig...	System.NotImplementedExcept...	ApplicationInsightsWebAPI.Ver...

At the bottom, it says "1s 356ms | Display time (UTC+00:00) ~" and "Query details | 1 - 1 of 1".

16. Go inside **Investigate > Live Metrics section.**



Steps :-

1. Login to Azure portal and click on “Create a resource”.

The screenshot shows the Microsoft Azure portal homepage. At the top, there's a navigation bar with 'Microsoft Azure' and a search bar. Below the navigation bar is a toolbar with various icons for inserting content like tables, images, and links. The main area is titled 'Azure services' and contains several quick access links: 'Create a resource' (with a plus sign icon), 'Resource groups', 'All resources', 'Subscriptions', 'Cost Management ...', 'Dev centers', 'Azure Active Directory', 'Groups', 'Quickstart Center', and a 'More services' button with an arrow icon. Below this is a section titled 'Resources' with 'Recent' and 'Favorite' tabs. The overall interface is clean and modern, typical of cloud service management platforms.

2. Click on “Containers” tab and select **Kubernetes Services** and click **create**.

The screenshot shows the 'Create a resource' page in the Microsoft Azure portal. The left sidebar has a 'Categories' section with options like AI + Machine Learning, Analytics, Blockchain, Compute, Containers (which is selected and highlighted in grey), Databases, Developer Tools, DevOps, and Identity. The main content area has sections for 'Get Started' (with a search bar) and 'Recently created'. It also features 'Popular Azure services' (Kubernetes Service, Web App for Containers, Batch Service, Kubernetes - Azure Arc, Container App) and 'Popular Marketplace products' (NVIDIA GPU-Optimized Image for AI & HPC - v21.06.0, Windows Server 2022 Core Datacenter Minimal OS, Basic, Hyper-V Server on Windows Server 2016, Docker Engine Community on Ubuntu 20.04 LTS). A 'Getting Started? Try our Quickstart center' link is also present. The top navigation bar includes the user's email (priyad@sies.edu.in) and the 'SOUTH INDIAN EDUCATION SOC.' logo.

3. Fill all the details as follows:

Create new resource group.

Azure Kubernetes Service (AKS) manages your hosted Kubernetes environment, making it quick and easy to deploy and manage containerized applications without container orchestration expertise. It also eliminates the burden of ongoing operations and maintenance by provisioning, upgrading, and managing your clusters in the cloud or on-premises.

[Learn more about Azure Kubernetes Service](#)

Project details

Select a subscription to manage deployed resources.

Subscription * ⓘ

Resource group * ⓘ

Name * CAD2022

OK Cancel

Cluster details

Review + create < Previous Next : Node pools >

4. Name the cluster, select the region. Keep remaining details as default. Click **review+create**

Standard (\$\$)

To quickly customize your Kubernetes cluster, choose one of the preset configurations above. You can modify these configurations at any time.

[Learn more and compare presets](#)

Kubernetes cluster name * AKSCAD

Region * ⓘ (Asia Pacific) Central India

Availability zones ⓘ Zones 1,2,3

High availability is recommended for standard configuration.

Kubernetes version * ⓘ 1.23.12 (default)

Review + create < Previous Next : Node pools >

5. Validation passed will be displayed.

The screenshot shows the 'Create Kubernetes cluster' wizard. At the top, there is a toolbar with various icons for inserting content like tables, images, and links. Below the toolbar, the title 'Create Kubernetes cluster' is displayed. A green banner at the top indicates 'Validation passed'. Below the banner, there are tabs for 'Basics', 'Node pools', 'Access', 'Networking', 'Integrations', 'Advanced', 'Tags', and 'Review + create'. The 'Review + create' tab is currently selected. Under the 'Basics' section, the following configuration is shown:

Subscription	Free Trial
Resource group	CAD2022
Region	Central India
Kubernetes cluster name	AKSCAD
Kubernetes version	1.23.12

6. Click Create

The screenshot shows the 'Create Kubernetes cluster' wizard on the Microsoft Azure portal. At the top, there is a navigation bar with the Microsoft Azure logo and a search bar. Below the navigation bar, there are sections for 'Integrations', 'Advanced', and 'Tags'. The 'Integrations' section includes options for Container registry, Microsoft Defender for Cloud, Container monitoring, Log Analytics workspace, and Azure Policy. The 'Advanced' section shows the Infrastructure resource group as 'MC_CAD2022_AKSCAD_centralindia'. The 'Tags' section shows '(none)'. At the bottom, there are buttons for 'Create', '< Previous' and 'Next >', and a link to 'Download a template for automation'.

7. The cluster will proceed for deployment.

The screenshot shows the Microsoft Azure Overview page for a deployment named "microsoft.aks-20221125123219". The status bar indicates "Deployment is in progress". Deployment details show two resources: "VnetDeployment-d09b6d5t" and "SolutionDeployment-20221", both in a "Created" state. A sidebar on the right provides links to Container Insights, Microsoft tutorials, and cost management.

Deployment name: microsoft.aks-20221125123219 Start time: 11/25/2022, 12:48:55 PM
Subscription: Free Trial Correlation ID: fe47cfdd-abae-422c-b65b-f901f4313cf9

Deployment details

Resource	Type	Status	Operation details
VnetDeployment-d09b6d5t	Microsoft.Resources/depl...	Created	Operation details
SolutionDeployment-20221	Microsoft.Resources/depl...	Created	Operation details

The screenshot shows the Microsoft Azure Overview page for the same deployment, now indicating "Your deployment is complete". Deployment details and resource status remain the same. A sidebar on the right provides links to cost management, Container Insights, and Azure Monitor.

Deployment name: microsoft.aks-20221125123219 Start time: 11/25/2022, 12:48:55 PM
Subscription: Free Trial Correlation ID: fe47cfdd-abae-422c-b65b-f901f4313cf9

Deployment details

Next steps

- Create a quick start application Recommended
- Create a Kubernetes deployment Recommended
- Integrate automatic deployments within your cluster Recommended
- Connect to cluster Recommended

[Go to resource](#) [Connect to cluster](#)

8. You can check the details of your resource and Kubernetes Cluster by clicking on them.

The screenshot shows the Microsoft Azure portal's 'Overview' page for a resource group named 'CAD2022'. The top navigation bar includes a search bar, account information ('priyad@si...'), and various icons for creating new resources and managing existing ones. Below the header, there are sections for 'Essentials' and 'Resources'.

Essentials:

- Subscription (move) : [Free Trial](#)
- Deployments : [5 Succeeded](#)
- Subscription ID : cb33d926-6e3b-47c8-981e-87bc261a649c
- Location : Central India
- Tags (edit) : [Click here to add tags](#)

Resources:

Showing 1 to 2 of 2 records. Show hidden types [①](#)

Filter for any field... Type equals all [×](#) Location equals all [×](#) [+ Add filter](#)

No grouping [▼](#) [List view](#)

The screenshot shows the Microsoft Azure portal's 'Overview' page for a Kubernetes service named 'AKSCAD'. The top navigation bar includes a search bar, account information ('priyad@si...'), and various icons for managing the service. Below the header, there are sections for 'Essentials' and 'Properties'.

Essentials:

- Resource group : [CAD2022](#)
- Status : Succeeded (Running)
- Location : Central India
- Subscription : [Free Trial](#)
- Subscription ID : cb33d926-6e3b-47c8-981e-87bc261a649c
- Tags (edit) : [Click here to add tags](#)

Properties:

- Kubernetes version : [1.23.12](#)
- API server address : [akscad-dns-78fae0a6.hcp.centralindia.azmk8s.i...](#)
- Network type (plugin) : [Azure CNI](#)
- Node pools : [1 node pool](#)

Get started [Properties](#) Monitoring Capabilities (3) Recommendations Tutorials

 Kubernetes services			
Get started Properties Monitoring Capabilities (3) Recommendations Tutorials π Ω			
 Insert  Tables  Picture  Screen Clipping  Link  Attach File  File  Printout  Scanner  Record Audio  Record Video  Recording  Date  Time  Date & Time  Equation  Symbol			
Encryption type	Encryption at-rest with a platform-managed key	API server address	akscad-dns-78fae0a6.hcp.centralindia.azmk8s.io
Virtual node pools	Enabled	Network type (plugin)	Azure CNI
[Node pools]			
Node pools	1 node pool	Pod CIDR	-
Kubernetes versions	1.23.12	Service CIDR	10.0.0.0/16
Node sizes	Standard_DS2_v2	DNS service IP	10.0.0.10
[Configuration]			
Kubernetes version	1.23.12	Docker bridge CIDR	172.17.0.1/16
Auto Upgrade Type	Patch	Network Policy	None
Authentication and Authorization	Local accounts with Kubernetes RBAC	Load balancer	Standard
Local accounts	Enabled	HTTP application routing	Not enabled
		Private cluster	Not enabled
		Authorized IP ranges	Not enabled
		Application Gateway	Not enabled
		ingress controller	

Steps :-

1. In the azure portal search devops starter.

The screenshot shows the Azure portal homepage. At the top, there's a search bar and several navigation icons. Below the header, there's a section titled "Azure services" with various icons like "Create a resource", "DevOps Starter", "Resource groups", etc. Under "Resources", there's a "Recent" tab showing a list of resources with columns for Name, Type, and Last Viewed. The list includes "AKSCAD" (Kubernetes service), "CAD2022" (Resource group), "CADP4" (Resource group), and "Free Trial" (Subscription). There's also a "See all" link. At the bottom, there's a "Navigate" section.

2. Click on create.

The screenshot shows the "DevOps Starter" blade in the Azure portal. At the top, it says "Home > DevOps Starter". It has a toolbar with "Create", "Manage view", "Refresh", "Export to CSV", "Open query", and "Assign tags". Below that are filter buttons for "Subscription equals all", "Resource Group Name equals all", and "Location equals all". A search bar at the top says "Name ↑↓". In the center, there's a large "No DevOps starter to display" message with a "Create DevOps starter" button below it. A "Learn more" link is also present.

3. Select .NET and click next.

The screenshot shows the Microsoft Azure DevOps Starter interface. At the top, there's a navigation bar with 'Microsoft Azure' and a search bar. Below it, a banner states 'DevOps starter is being deprecated and will get retired on March 31, 2023. Learn more'. A blue header bar says 'Launch an app running in Azure in a few quick steps: Everything you need, created and ready to go: code repository, CI/CD pipeline or Github Workflow, and the necessary Azure resources.' Below this, a section titled 'Start fresh with a new application' lists several options:

- .NET**: New Web App using ASP.NET or ASP.NET Core, or a new IoT app.
- Node.js**: New Web app using Node.js, Express.js or Sails.js, or a new IoT app.
- PHP**: New Web app using simple PHP.
- Java**
- Static Website**
- Python**

A progress bar at the bottom indicates the current step: '1. Choose a runtime' (blue), '2. Choose an application framework' (green), '3. Choose a service' (grey), and '4. Create' (grey).

4. Under Choose an application framework, select **ASP.NET Core** and then select Next

The screenshot shows the Microsoft Azure DevOps Starter interface, similar to the previous one but with a different step highlighted. The progress bar now shows '2. Choose an application framework' (green) as the active step. The 'Runtime' step (1) has a checkmark. The 'Service' and 'Create' steps (3 and 4) are greyed out.

Choose an application framework

Two options are shown:

- ASP.NET**: Open source web framework for building modern web apps and services.
- ASP.NET Core**: Cross-platform, open-source framework for building modern web apps and services.

At the bottom, there are navigation buttons: '< Previous' and 'Next: Service >'.

5. Select **Kubernetes Service**, and then select Next.

The screenshot shows the Microsoft Azure DevOps Starter interface. At the top, there's a navigation bar with 'Microsoft Azure' and a search bar. Below it, a progress bar indicates steps 1 through 4, with 'Service' highlighted in blue. A message at the top states: 'DevOps starter is being deprecated and will get retired on March 31, 2023. [Learn more](#)'. The main area is titled 'Select an Azure service to deploy the application'. It lists four options: 'Kubernetes Service' (selected), 'Windows Web App', 'Linux Web App', and 'Web App for Containers'. Each option has a brief description. At the bottom, there are 'Next: Create >' and '< Previous' buttons.

6. Click on **Authorize** and provide credentials for **Github Account**.

The screenshot shows the Microsoft Azure DevOps Starter interface. The progress bar now shows 'Create' as the active step. A message at the top states: 'DevOps starter is being deprecated and will get retired on March 31, 2023. [Learn more](#)'. Below it, the 'Select Repository and Subscription' section is shown. A note says: 'Ready to deploy ASP.NET Core app to Azure Kubernetes Service.' A warning message in a box says: 'Azure needs permission to access your GitHub account to create the workflow'. At the bottom is a large blue 'Authorize' button.

7. Fill the details.

The screenshot shows the Microsoft Azure DevOps Starter wizard. At the top, there's a navigation bar with 'Microsoft Azure' and a search bar. Below it, a breadcrumb trail shows 'Home > DevOps Starter > DevOps Starter'. A 'Create' button is visible. The main area has a progress bar with four steps: 'Runtime' (checkmark), 'Framework' (checkmark), 'Service' (checkmark), and 'Create' (step 4). A message at the top states: 'DevOps starter is being deprecated and will get retired on March 31, 2023. Learn more'. The 'Select Repository and Subscription' section contains the following fields:

- Organization *: danielpriya777
- Repository *: AKSCAD (highlighted with a green checkmark)
- Subscription *: Free Trial
- Create new or use existing cluster: Create New Use Existing
- Cluster name *: AKSCAD (highlighted with a green checkmark)

8. Click on **Additional settings** under Node Virtual Machine to reduce node count to 1.

The screenshot shows the Microsoft Azure DevOps Starter wizard at step 4. The interface is similar to the previous one, with the 'Select Repository and Subscription' section. However, the 'Node Virtual Machine Size' field is now highlighted, showing 'Standard_DS2_v2' and a 'Change size' link. Below this, a note says 'Node count: 3' and 'Additional settings'.

Additional settings

X

Kubernetes Service

Resource group * ⓘ

AKSCAD-rg



Application Insights Location ⓘ

Central India



Kubernetes Version ⓘ

1.22.11



Node count * ⓘ

1



Log Analytics location

Central India



Enable HTTP application routing addon ⓘ



Yes



No

Container Registry

OK

Continued.

Additional settings

X

1.0.0.11



Node count * ⓘ

1



Log Analytics location

Central India



Enable HTTP application routing addon ⓘ



Yes



No

Container Registry

Container Registry Name *

AKSCADacr



Container Registry SKU ⓘ

Standard



Container Registry Location

Central India



OK

9. Click OK. Then click **review+create**.

Home > DevOps Starter >

DevOps Starter Create Insert Space Tables Picture Screen Clipping Images Link Attach File File Printout Scanner Record Audio Record Video Date

DevOps starter is being deprecated and will get retired on March 31, 2023. [Learn more](#)

Organization *

Repository * ✓

Subscription * ⓘ

Create new or use existing cluster Create New Use Existing ⓘ

Cluster name * ⓘ ✓

Location ⓘ

Node Virtual Machine Size ⓘ **Standard_DS2_v2**
[Change size](#)

Node count: 1
[Additional settings](#)

By continuing, you agree to the [Terms of Service](#) and the [Privacy Statement](#).

< Previous **Review + Create**

Microsoft Azure  Search resources, services, and docs (G+)

Home >  Deploy_DevOps_Project_AKSCAD | Overview

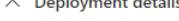


 Overview  Inputs  Outputs  Template

*** Deployment is in progress

 Deployment name: Deploy_DevOps_Project_AKSCAD Start time: 11/26/2022, 1:38:40 PM
Subscription: Free Trial Correlation ID: 43aff925-fa63-4273-aa00-4e18a712c566 

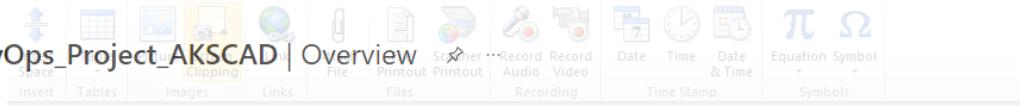
 Deployment details

Resource	Type	Status	Operation details
No results.			

Give feedback  Tell us about your experience with deployment 

Microsoft Azure  Search resources, services, and docs (G+)

Home >  Deploy_DevOps_Project_AKSCAD | Overview

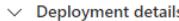


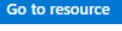
     

 Overview  Inputs  Outputs  Template

✓ Your deployment is complete

 Deployment name: Deploy_DevOps_Project_AKSCAD Start time: 11/26/2022, 1:38:40 PM
Subscription: Free Trial Correlation ID: 43aff925-fa63-4273-aa00-4e18a712c566 

 Deployment details  Next steps



Give feedback  Tell us about your experience with deployment 

The screenshot shows the Microsoft Azure interface. At the top, there's a navigation bar with 'Microsoft Azure' and a search bar. Below it, a ribbon menu includes 'Home', 'AKSCAD DevOps Starter', and various icons for inserting content like tables, images, and files. A message box in the center says 'Azure needs permission to access your GitHub account to view latest workflow execution and status of jobs' with a blue 'Authorize' button.

Azure resources

Azure Resources are not yet created. They will be created by GitHub Workflow. You can track workflow status in 'GitHub Workflow' section

Application Insights

AKSCAD

10. Click on **Authorize**. The Kubernetes service will be **created**.

The screenshot shows the Microsoft Azure interface after the workflow has been authorized. The 'GitHub Workflow' section now displays the 'Latest Run' and 'Jobs' details. The 'Latest Run' table shows a successful workflow execution with a commit pushed to 'master'. The 'Jobs' table shows a single job named 'latest run' that has succeeded. The 'Azure resources' section now lists a 'Kubernetes Service' and its status as 'Succeeded'. The 'Application Insights' section also shows the 'AKSCAD' entry.

Azure resources

Kubernetes Service

AKSCAD

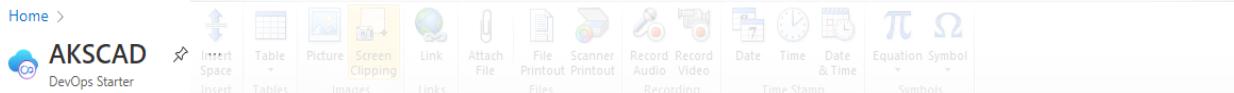
Succeeded

Kubernetes Version 1.22.11

Application Insights

AKSCAD

Continued...



Refresh Delete

Job name

latest run

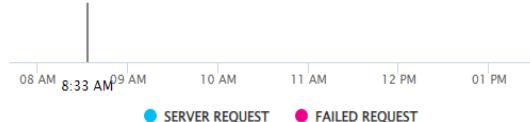
Build and push image to container

11/26/2022 08:09:07 AM 8m 17s

Kubernetes Version 1.22.11

Application Insights

AKSCAD



SERVER REQUEST | FAILED REQUEST

Steps :-

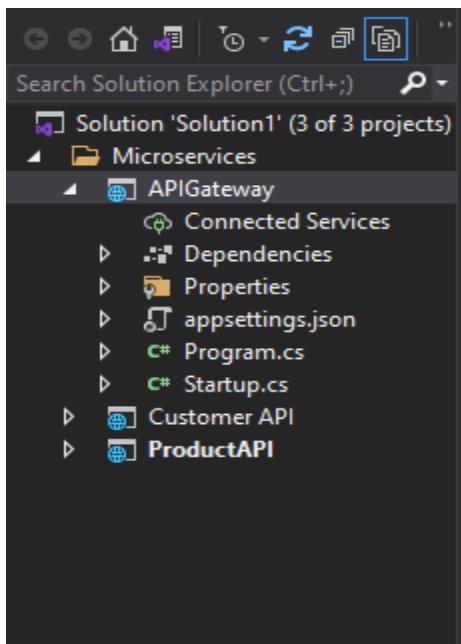
Create a blank solution -> Solution1

Add the following projects in it.

ProductAPI (Web API)

CustomerAPI (Web API)

APIGateway(Asp.net core empty project)



1. Add a controller to **ProductAPI** and add the following Code:

```
using Microsoft.AspNetCore.Http;
using Microsoft.AspNetCore.Mvc;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;

namespace ProductAPI.Controllers
{
    [Route("api/[controller]")]
    [ApiController]
    public class ProductController : ControllerBase
    {
        public ProductController()
    }
}
```

```

    {
    }

    [HttpGet]
    public List<String> Get()
    {
        var productList = new List<String> { "Product1", "Product2", "Product3", "Product4" };
        return productList;
    }

    [HttpDelete]
    [Route("ProductVersion/{id}")]
    public string ProductVersion(int id)
    {
        return $"Product with id {id} is removed";
    }
}

}

```

2. Add controller to customerAPI and add the following code.

```

using Microsoft.AspNetCore.Http;
using Microsoft.AspNetCore.Mvc;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;

namespace Customer_API.Controllers
{
    [Route("api/[controller]")]
    [ApiController]
    public class CustomerController : ControllerBase
    {
        public List<string> Get()
        {
            var productList = new List<string> { "Customer 1", "Customer 2", "Customer 3", "Customer 4" };
            return productList;
        }
    }
}

```

3. Install OCELOT using nuget package manager.

NuGet - Solution

Browse Installed Updates Consolidate

ocelot

Ocelot by Tom Pallister, 4.84M downloads
Ocelot is an API Gateway. The project is aimed at people using .NET running point of entry into their system. In particular I want easy integration with local storage and external services.

MMLib.SwaggerForOcelot by Milan Martinak, 801K downloads
Swagger generator for Ocelot downstream services.

Ocelot.Provider.Polly by Tom Pallister, 651K downloads
Provides Ocelot extensions to use Polly.NET

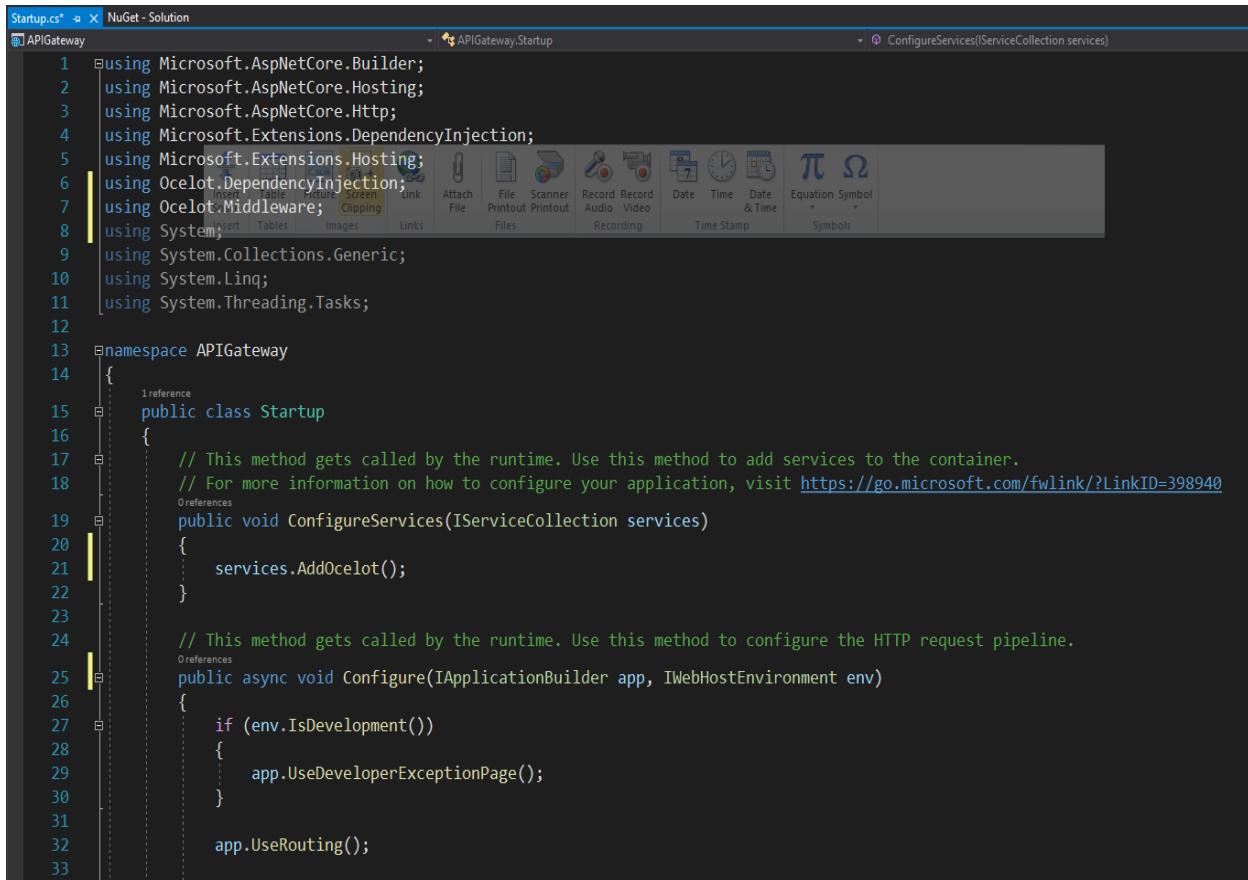
Ocelot.Provider.Consul by Tom Pallister, 583K downloads
Provides Ocelot extensions to use Consul

Ocelot.Cache.CacheManager by Tom Pallister, 504K downloads
Provides Ocelot extensions to use CacheManager.Net

Ocelot.Administration by Tom Pallister, 292K downloads
Provides Ocelot extensions to use the administration API and IdentityService

Ocelot.Provider.Kubernetes by geffzhang, 216K downloads
Provides Ocelot extensions to use kubernetes

4. Open **Startup.cs** for APIGateway. Add **Ocelot** in services under Configure Service.



```

Startup.cs*  NuGet - Solution
APIGateway  APIGateway.Startup  ConfigureServices(IServiceCollection services)

1  using Microsoft.AspNetCore.Builder;
2  using Microsoft.AspNetCore.Hosting;
3  using Microsoft.AspNetCore.Http;
4  using Microsoft.Extensions.DependencyInjection;
5  using Microsoft.Extensions.Hosting;
6  using Ocelot.DependencyInjection;
7  using Ocelot.Middleware;
8  using System;
9  using System.Collections.Generic;
10 using System.Linq;
11 using System.Threading.Tasks;
12
13 namespace APIGateway
14 {
15     public class Startup
16     {
17         // This method gets called by the runtime. Use this method to add services to the container.
18         // For more information on how to configure your application, visit https://go.microsoft.com/fwlink/?LinkID=398940
19         public void ConfigureServices(IServiceCollection services)
20         {
21             services.AddOcelot();
22         }
23
24         // This method gets called by the runtime. Use this method to configure the HTTP request pipeline.
25         public async void Configure(IApplicationBuilder app, IWebHostEnvironment env)
26         {
27             if (env.IsDevelopment())
28             {
29                 app.UseDeveloperExceptionPage();
30             }
31
32             app.UseRouting();
33         }
}

```

5. At the end , write await app.UseOcelot to make sure that the app uses the Ocelot Middleware. So the final Code looks like this for **APIGateway -> Startup.cs**

```

using Microsoft.AspNetCore.Builder;
using Microsoft.AspNetCore.Hosting;
using Microsoft.AspNetCore.Http;
using Microsoft.Extensions.DependencyInjection;
using Microsoft.Extensions.Hosting;
using Ocelot.DependencyInjection;
using Ocelot.Middleware;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;

namespace APIGateway
{
    public class Startup
    {

```

```

/*
This method gets called by the runtime. Use this method to add services to the container.
For more information on how to configure your application, visit
https://go.microsoft.com/fwlink/?LinkID=398940
*/
public void ConfigureServices(IServiceCollection services)
{
    services.AddOcelot();
}

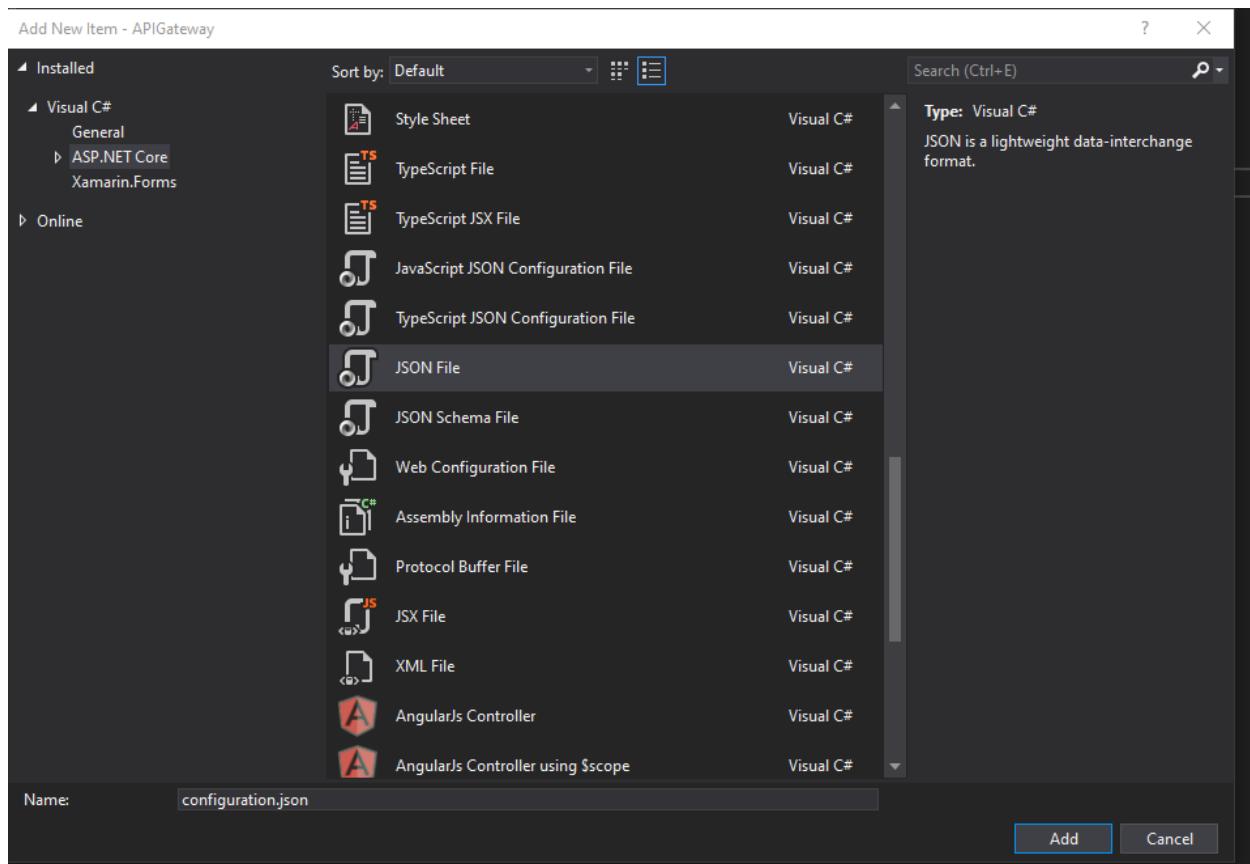
/* This method gets called by the runtime. Use this method to configure the HTTP request
pipeline.*/
public async void Configure(IApplicationBuilder app, IWebHostEnvironment env)
{
    if (env.IsDevelopment())
    {
        app.UseDeveloperExceptionPage();
    }

    app.UseRouting();

    app.UseEndpoints(endpoints =>
    {
        endpoints.MapGet("/", async context =>
        {
            await context.Response.WriteAsync("Hello World!");
        });
    });
    await app.UseOcelot();
}
}
}

```

6. Next, right click on **APIGateway** in solution->New Item->JSON file. Name it **Configuration.json** file.



7. After that, open **program.cs** file of APIGateway and make the following changes.

```
using Microsoft.AspNetCore.Hosting;
using Microsoft.Extensions.Configuration;
using Microsoft.Extensions.Hosting;
using Microsoft.Extensions.Logging;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;

namespace APIGateway
{
    public class Program
    {
        public static void Main(string[] args)
        {
            CreateHostBuilder(args).Build().Run();
        }
    }
}
```

```

public static IHostBuilder CreateHostBuilder(string[] args) =>
    Host.CreateDefaultBuilder(args)
        .ConfigureWebHostDefaults(webBuilder =>
    {
        webBuilder.UseStartup<Startup>();
    })
    .ConfigureAppConfiguration((hostingContext, config) =>
    {
        config.AddJsonFile("configuration.json");
    });
}

```

Microservices is a new software architecture. It is based on the Web Services (Web APIs). Microservices are the architectural approach to build applications from small to large scale applications.

An API gateway is an API management tool that sits between a client and a collection of backend services.

Ocelot is a lightweight, open-source, scalable, and fast API Gateway based on .NET Core and specially designed for microservices architecture. Basically, it is a set of middleware designed to work with [ASP.NET](#) Core.

It has several features such as routing, caching, security, rate limiting, etc.

Configuration File Syntax

```
{
  "Routes": [
    {
      "UpstreamPathTemplate": "/gateway/product",
      "UpstreamHttpMethod": [ "POST", "PUT", "GET" ],
      "DownstreamPathTemplate": "/api/product",
      "DownstreamScheme": "https",
      "DownstreamHostAndPorts": [
        {
          "Host": "localhost",
          "Port": 44339
        }
      ],
      "DangerousAcceptAnyServerCertificateValidator": true
    },
  ]
}
```

```
{  
    "UpstreamPathTemplate": "/gateway/product/{id}",  
    "UpstreamHttpMethod": [ "GET", "DELETE" ],  
    "DownstreamPathTemplate": "/api/product/ProductVersion/{id}",  
    "DownstreamScheme": "https",  
    "DownstreamHostAndPorts": [  
        {  
            "Host": "localhost",  
            "Port": 44339  
        }  
    ],  
  
    "DangerousAcceptAnyServerCertificateValidator": true  
  
,  
{  
    "UpstreamPathTemplate": "/gateway/customer",  
    "UpstreamHttpMethod": [ "POST", "PUT", "GET" ],  
    "DownstreamPathTemplate": "/api/customer",  
    "DownstreamScheme": "https",  
    "DownstreamHostAndPorts": [  
        {  
            "Host": "localhost",  
            "Port": 44398  
        }  
    ],  
  
    "DangerousAcceptAnyServerCertificateValidator": true  
  
,  
{  
    "DownstreamPathTemplate": "/api/customer/{id}",  
    "DownstreamScheme": "https",  
    "DownstreamHostAndPorts": [  
        {  
            "Host": "localhost",  
            "Port": 44398  
        }  
    ],  
    "UpstreamPathTemplate": "/gateway/customer/{id}",  
    "UpstreamHttpMethod": [ "GET", "DELETE" ],  
    "DangerousAcceptAnyServerCertificateValidator": true  
}  
],
```

```
"GlobalConfiguration": {  
    "BaseUrl": "http://localhost:44353"  
}  
}
```

Change the SSL Port in the launchsetting.json file of ProductAPI and CustomerAPI to the value in configuration.jso file of API Gateway.

ProductAPI->Launchsetting.json

```
{  
    "iisSettings": {  
        "windowsAuthentication": false,  
        "anonymousAuthentication": true,  
        "iisExpress": {  
            "applicationUrl": "http://localhost:35108",  
            "sslPort": 44339  
        }  
    },  
    "profiles": {  
        "IIS Express": {  
            "commandName": "IISExpress",  
            "launchBrowser": true,  
            "environmentVariables": {  
                "ASPNETCORE_ENVIRONMENT": "Development"  
            }  
        },  
        "ProductAPI": {  
            "commandName": "Project",  
            "launchBrowser": true,  
            "applicationUrl": "https://localhost:5001;http://localhost:5000",  
            "environmentVariables": {  
                "ASPNETCORE_ENVIRONMENT": "Development"  
            }  
        }  
    }  
}
```

CustomerAPI->Launchsetting.json

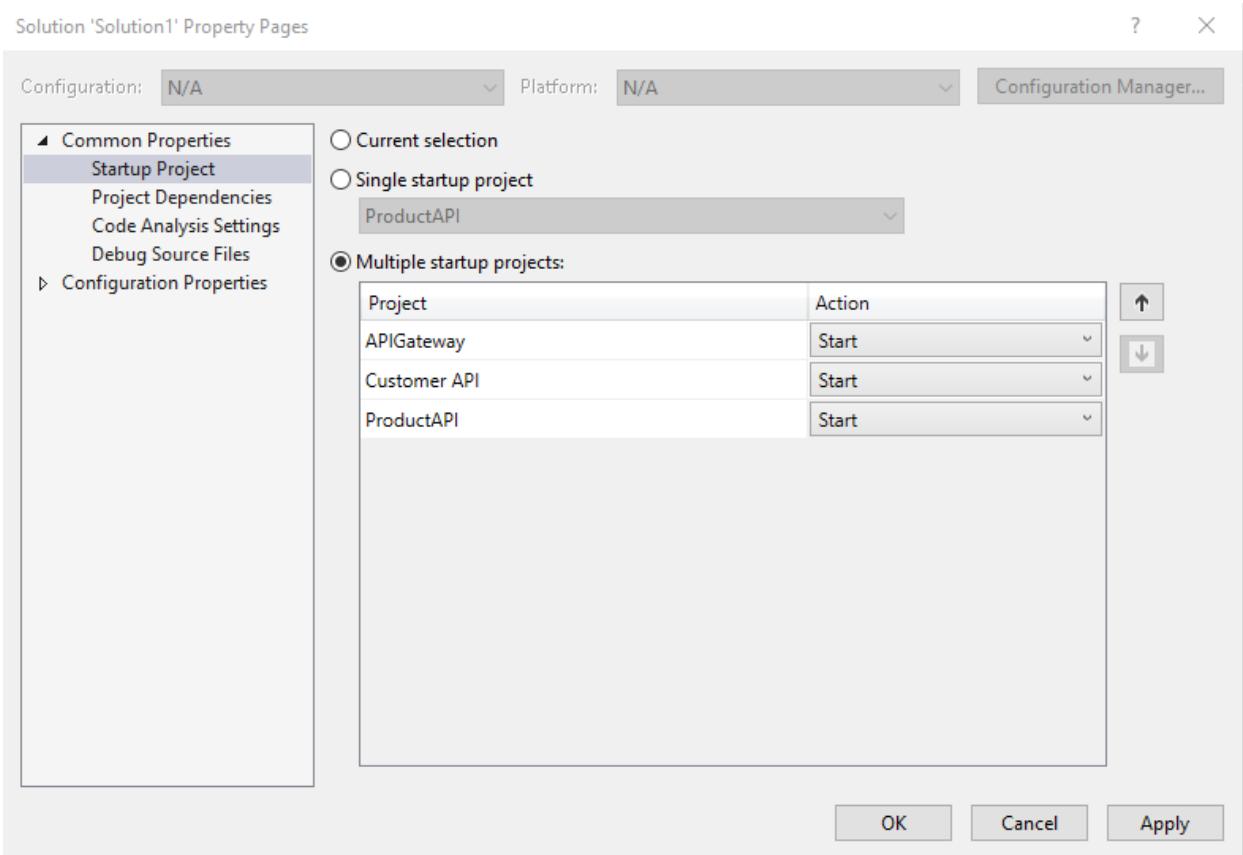
```
{  
    "iisSettings": {  
        "windowsAuthentication": false,  
        "anonymousAuthentication": true,  
        "iisExpress": {  
            "applicationUrl": "https://localhost:5001;http://localhost:5000",  
            "sslPort": 44339  
        }  
    }  
}
```

```
"applicationUrl": "http://localhost:6080",
  "sslPort": 44398
}
},
"profiles": {
  "IIS Express": {
    "commandName": "IISExpress",
    "launchBrowser": true,
    "environmentVariables": {
      "ASPNETCORE_ENVIRONMENT": "Development"
    }
  },
  "CustomerAPI": {
    "commandName": "Project",
    "launchBrowser": true,
    "applicationUrl": "https://localhost:5001;http://localhost:5000",
    "environmentVariables": {
      "ASPNETCORE_ENVIRONMENT": "Development"
    }
  }
}
```

8. Set up multiple startup in VS Studio

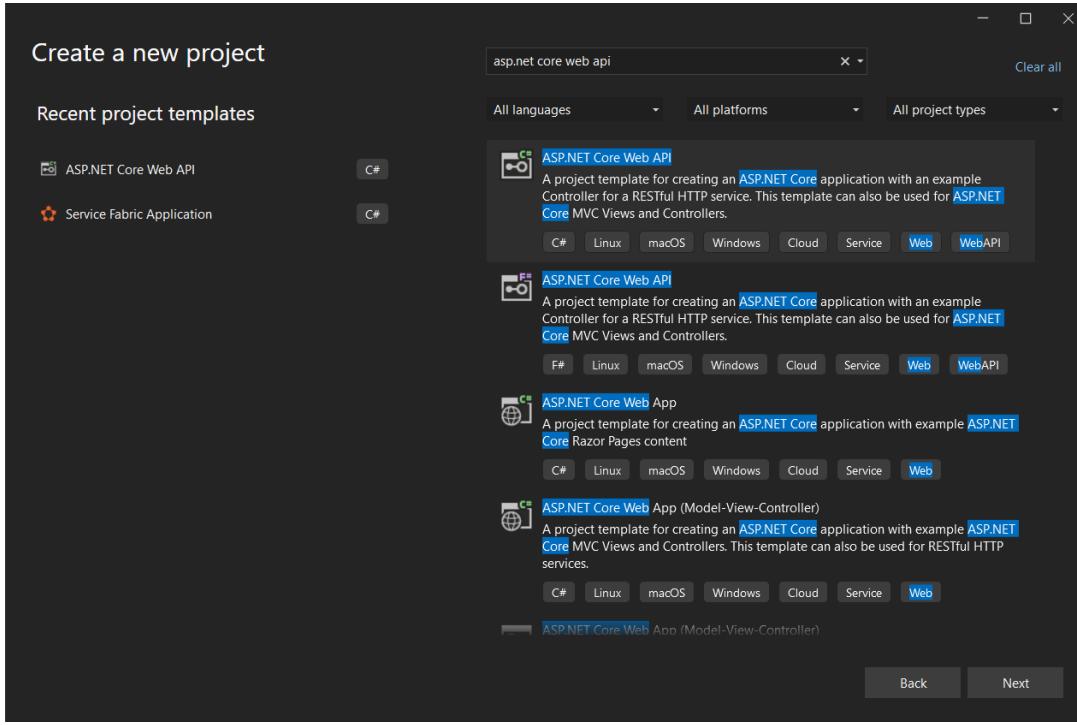
Right click on solution-> properties->Select “Multiple startup projects”. Then set all 3 projects to start.

9. Now run the project.

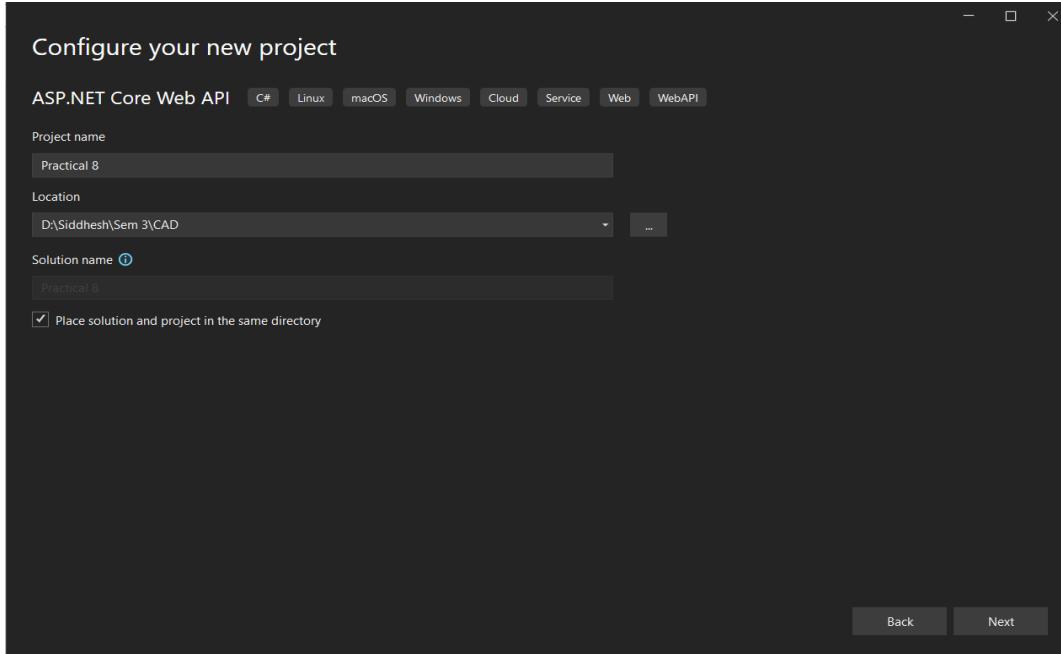


Steps :-

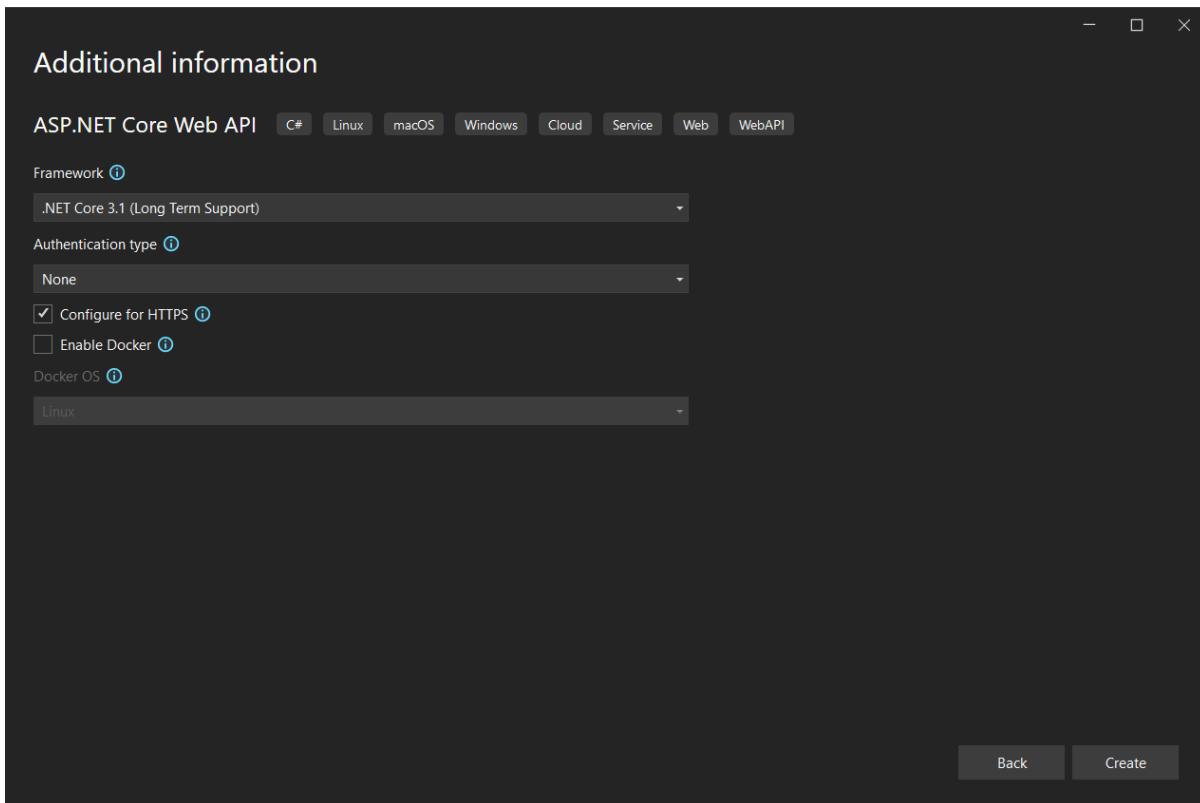
1. Open Visual studio and Create a new project of the type **ASP.NET Core Web API**.



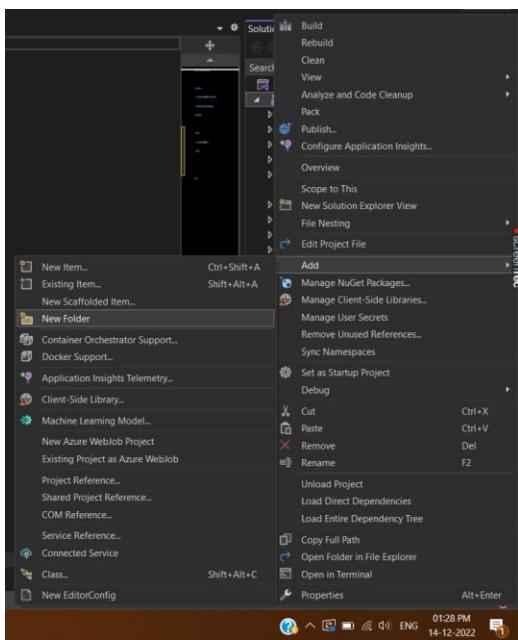
2. Give it a Meaningful name.



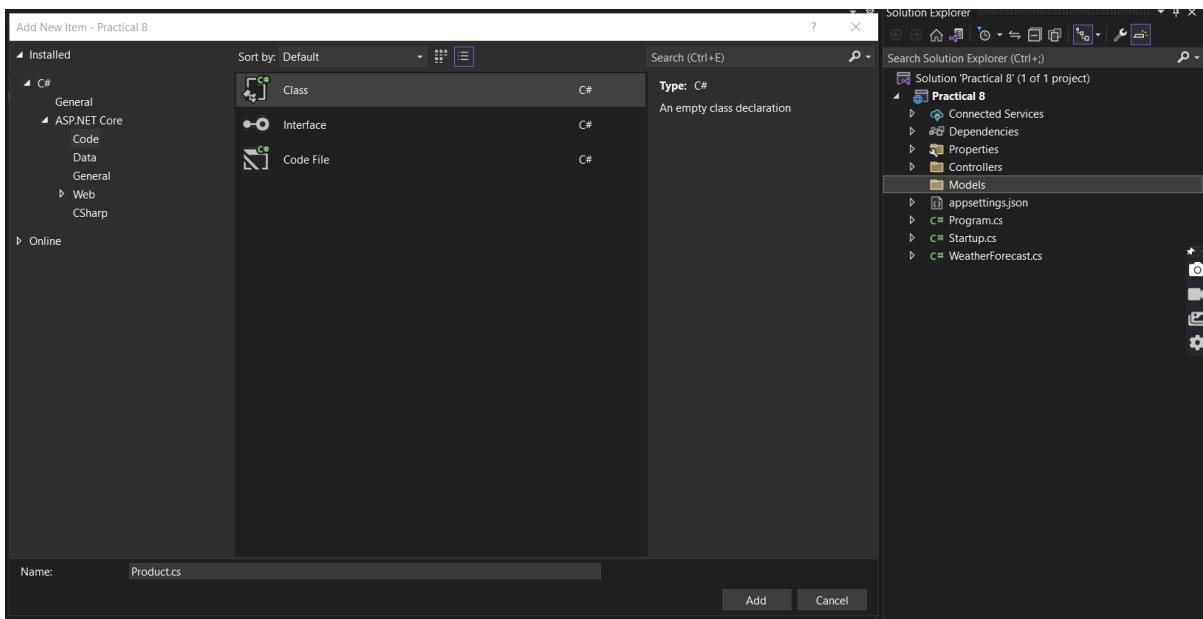
3. Select .NET Core 3.1 as Framework for this project and Click on Create.



4. Add a folder in the project and name it “**Models**”.



5. Create the Product model in this folder. Give it name as **Product.cs**.

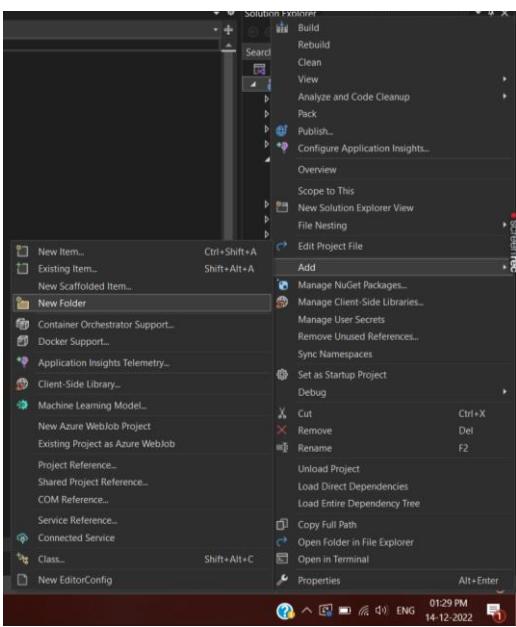


6. Add the following code inside **Product.cs** file:

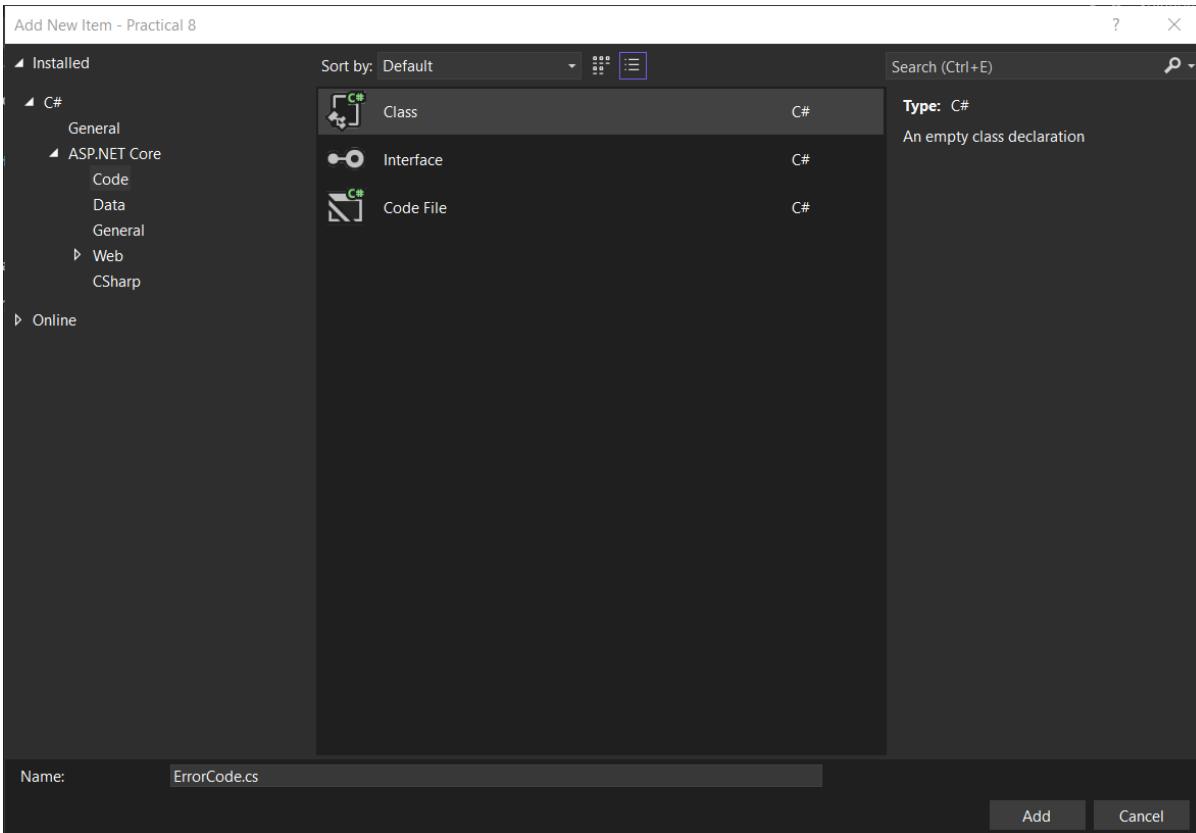
```
public class Product
{
    public int Id { get; set; }
    public string Name { get; set; }
    public DateTime? ModifiedDate { get; set; }
}
```

```
using System;
namespace Practical_8.Models
{
    public class Product
    {
        public int Id { get; set; }
        public string Name { get; set; }
        public DateTime? ModifiedDate { get; set; }
    }
}
```

7. Add another folder and name it “Errors”.

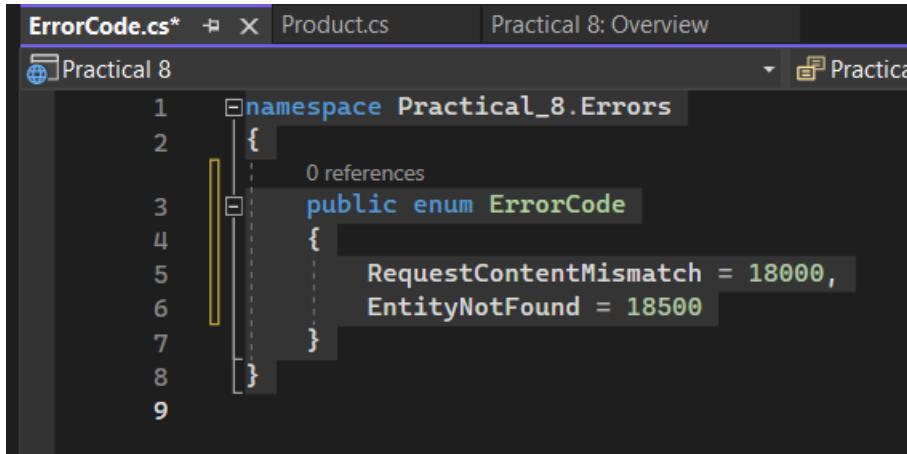


8. Add a class inside Errors folder and Name it as **ErrorCode.cs**.



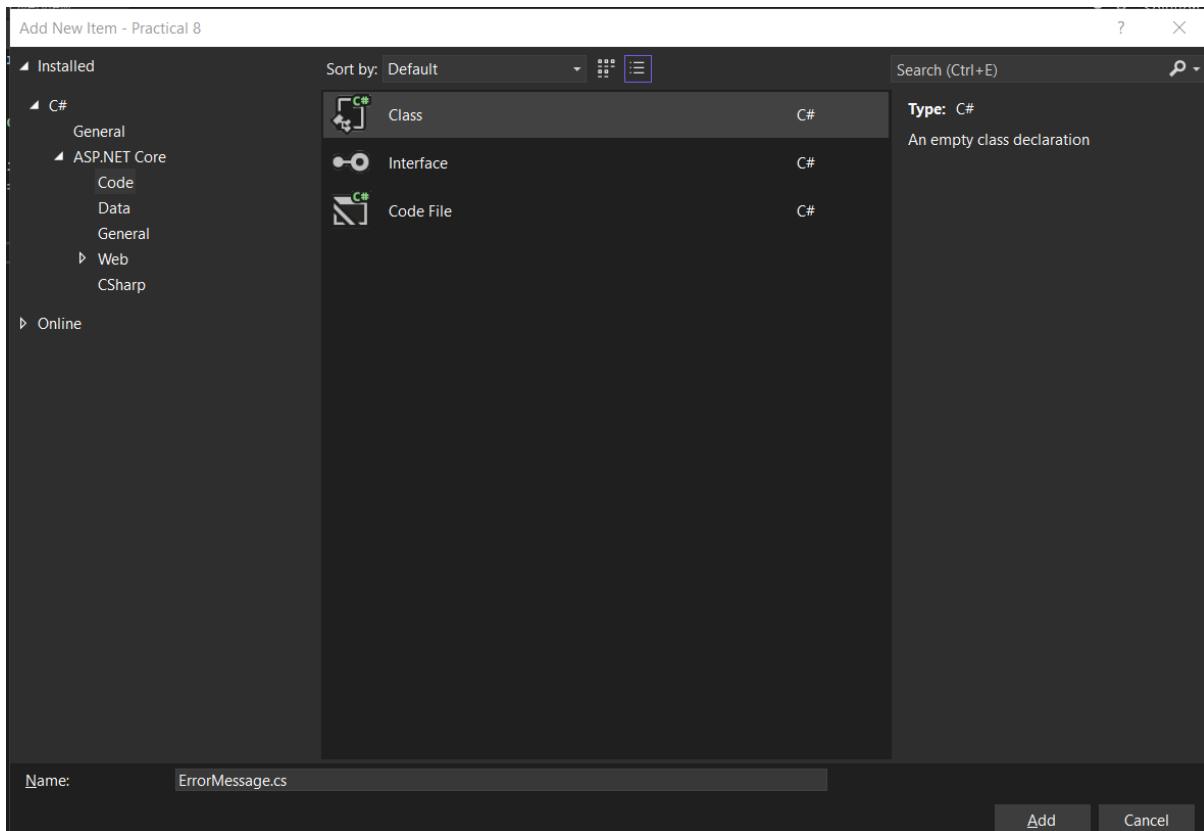
9. Add the following code inside **ErrorCode.cs** file:

```
public enum ErrorCode
{
    RequestContentMismatch = 18000,
    EntityNotFound = 18500
}
```



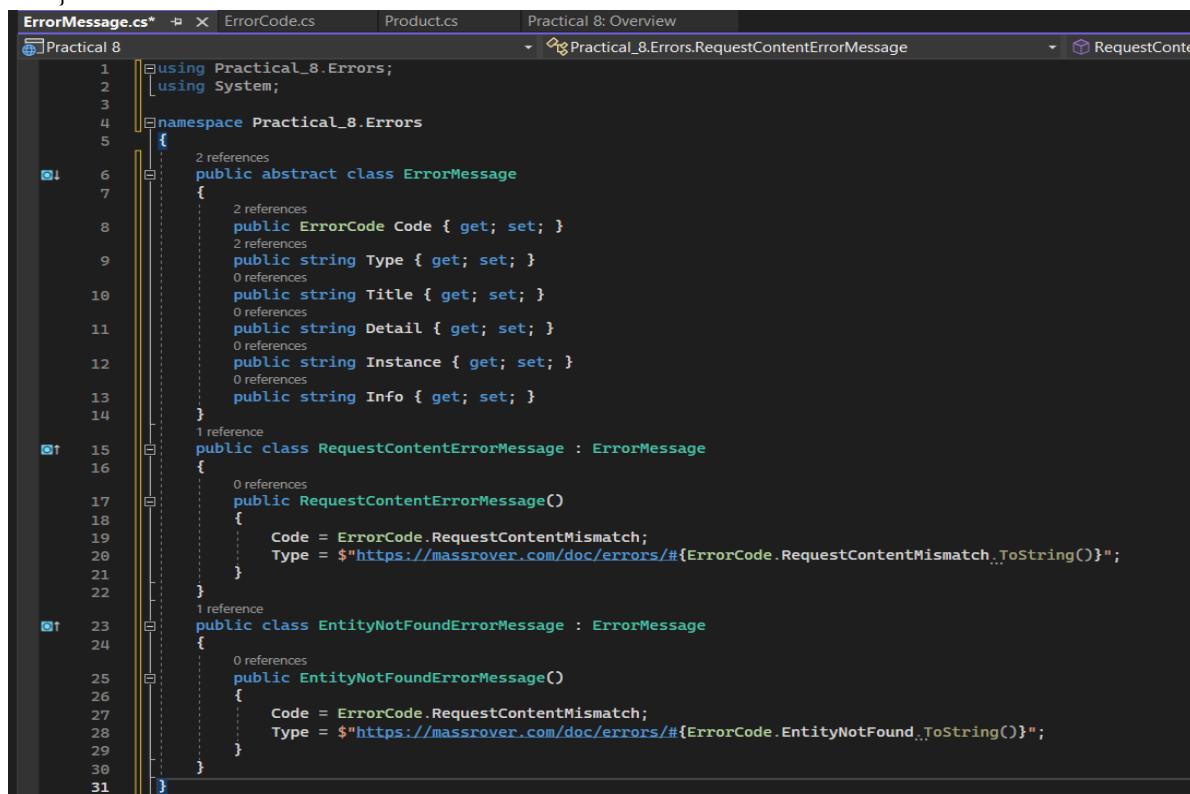
```
namespace Practical_8.Errors
{
    public enum ErrorCode
    {
        RequestContentMismatch = 18000,
        EntityNotFound = 18500
    }
}
```

10. Add an abstract class in the **Errors** folder, named “**ErrorMessage.cs**”



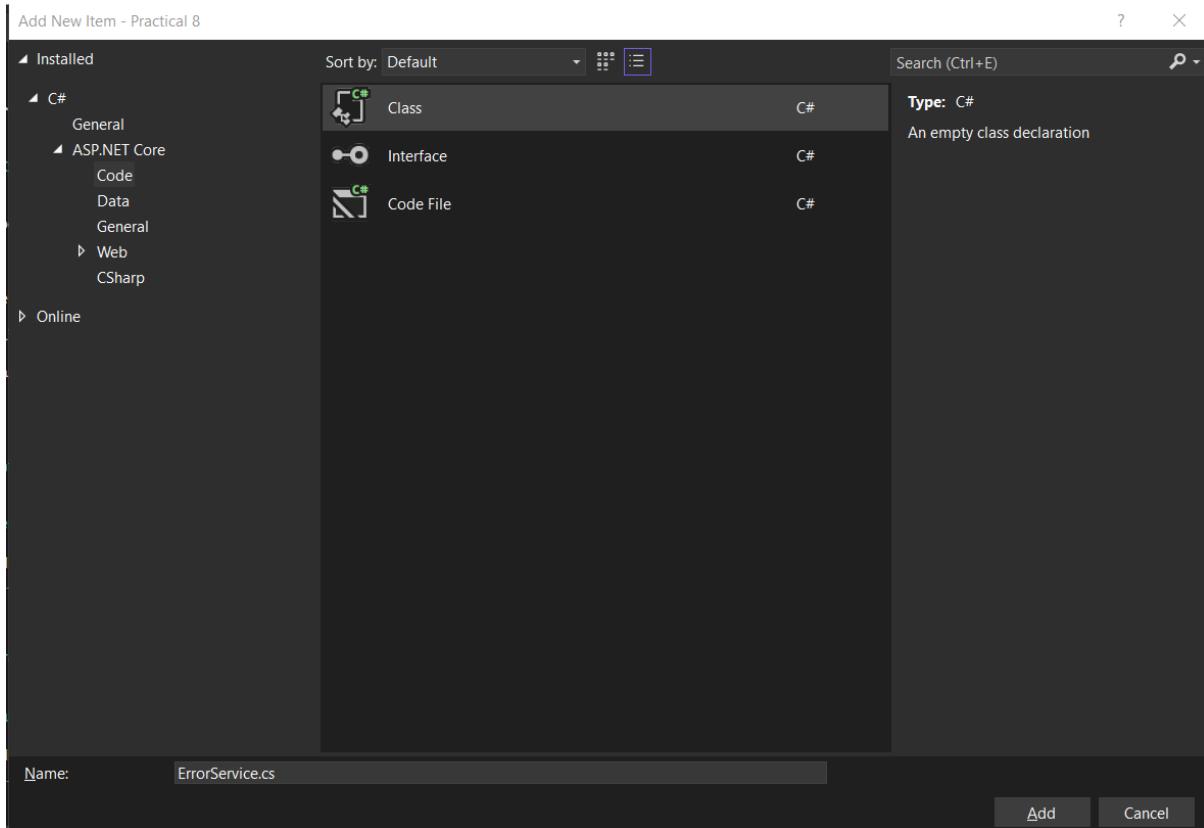
11. Add the following code inside **ErrorMessage.cs** file:

```
public abstract class ErrorMessage
{
    public ErrorCode Code { get; set; }
    public string Type { get; set; }
    public string Title { get; set; }
    public string Detail { get; set; }
    public string Instance { get; set; }
    public string Info { get; set; }
}
public class RequestContentErrorMessage : ErrorMessage
{
    public RequestContentErrorMessage()
    {
        Code = ErrorCode.RequestContentMismatch;
        Type = $"https://massrover.com/doc/errors/#{ErrorCode.RequestContentMismatch.ToString()}";
    }
}
public class EntityNotFoundErrorMessage : ErrorMessage
{
    public EntityNotFoundErrorMessage()
    {
        Code = ErrorCode.RequestContentMismatch;
        Type = $"https://massrover.com/doc/errors/#{ErrorCode.EntityNotFound.ToString()}";
    }
}
```



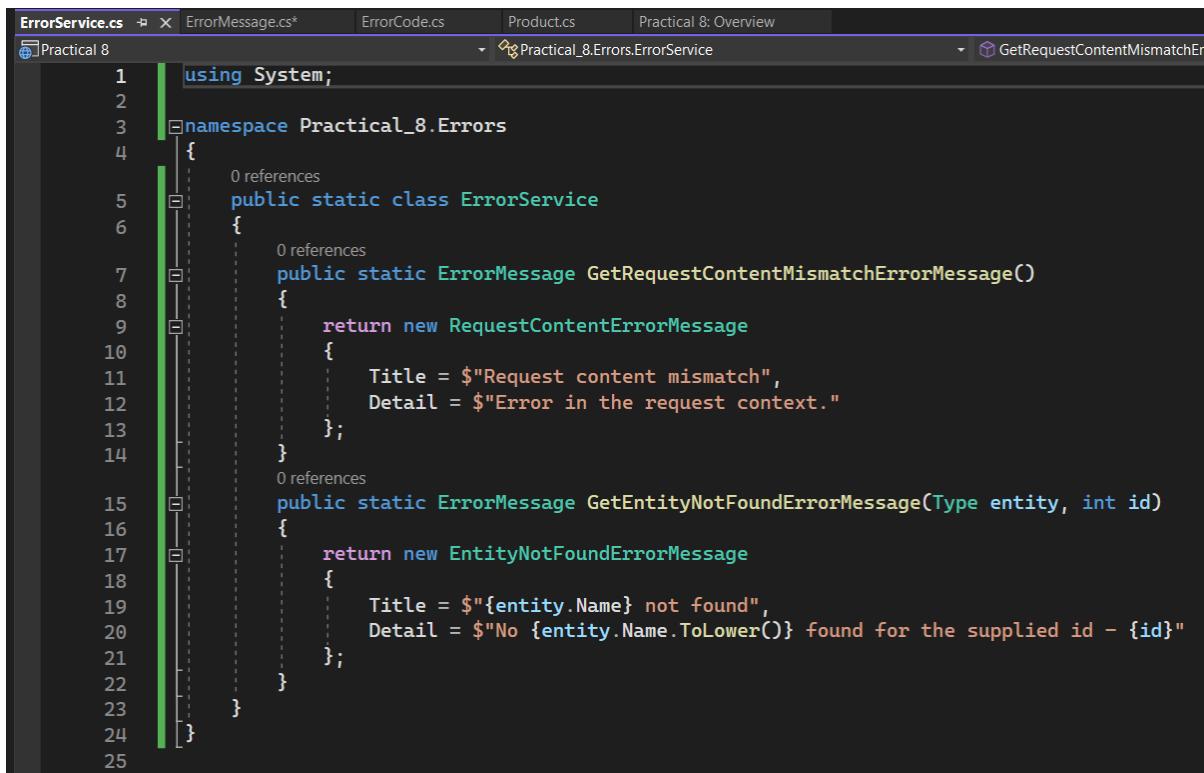
```
ErrorMessage.cs*  ✎ X ErrorCode.cs  Product.cs  Practical 8: Overview
Practical 8
  1  using Practical_8.Errors;
  2  using System;
  3
  4  namespace Practical_8.Errors
  5  {
  6      2 references
  7      public abstract class ErrorMessage
  8      {
  9          2 references
 10          public ErrorCode Code { get; set; }
 11          2 references
 12          public string Type { get; set; }
 13          0 references
 14          public string Title { get; set; }
 15          0 references
 16          public string Detail { get; set; }
 17          0 references
 18          public string Instance { get; set; }
 19          0 references
 20          public string Info { get; set; }
 21      }
 22
 23      1 reference
 24      public class RequestContentErrorMessage : ErrorMessage
 25      {
 26          0 references
 27          public RequestContentErrorMessage()
 28          {
 29              Code = ErrorCode.RequestContentMismatch;
 30              Type = $"https://massrover.com/doc/errors/#{ErrorCode.RequestContentMismatch.ToString()}";
 31          }
 32      }
 33
 34      1 reference
 35      public class EntityNotFoundErrorMessage : ErrorMessage
 36      {
 37          0 references
 38          public EntityNotFoundErrorMessage()
 39          {
 40              Code = ErrorCode.RequestContentMismatch;
 41              Type = $"https://massrover.com/doc/errors/#{ErrorCode.EntityNotFound.ToString()}";
 42          }
 43      }
 44  }
```

12. Add another class to return the correct **ErrorMessage** instance in the right context and name it as **ErrorService.cs**.



13. Add the following code in the **ErrorService.cs** file:

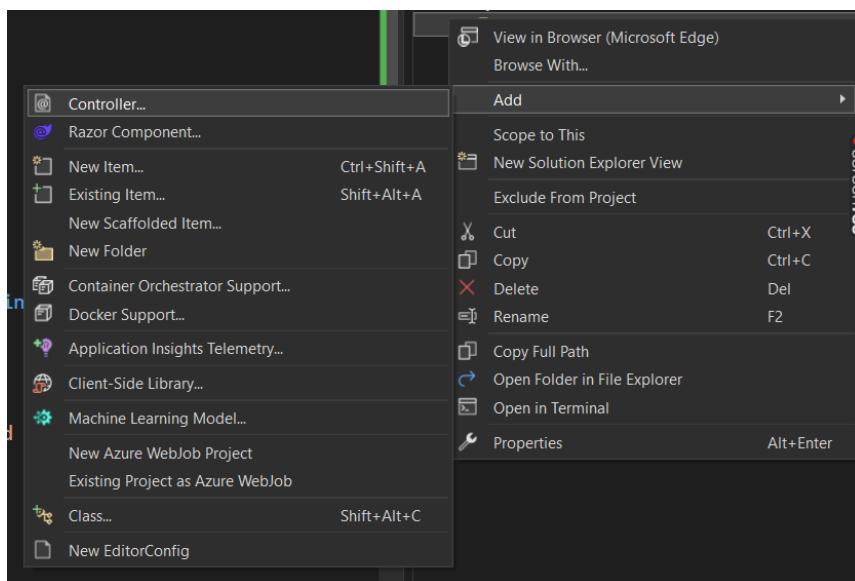
```
public static class ErrorService
{
    public static ErrorMessage GetRequestContentMismatchErrorMessage()
    {
        return new RequestContentErrorMessage
        {
            Title = $"Request content mismatch",
            Detail = $"Error in the request context."
        };
    }
    public static ErrorMessage GetEntityNotFoundErrorMessage(Type entity, int id)
    {
        return new EntityNotFoundErrorMessage
        {
            Title = $"{entity.Name} not found",
            Detail = $"No {entity.Name.ToLower()} found for the supplied id {id}"
        };
    }
}
```

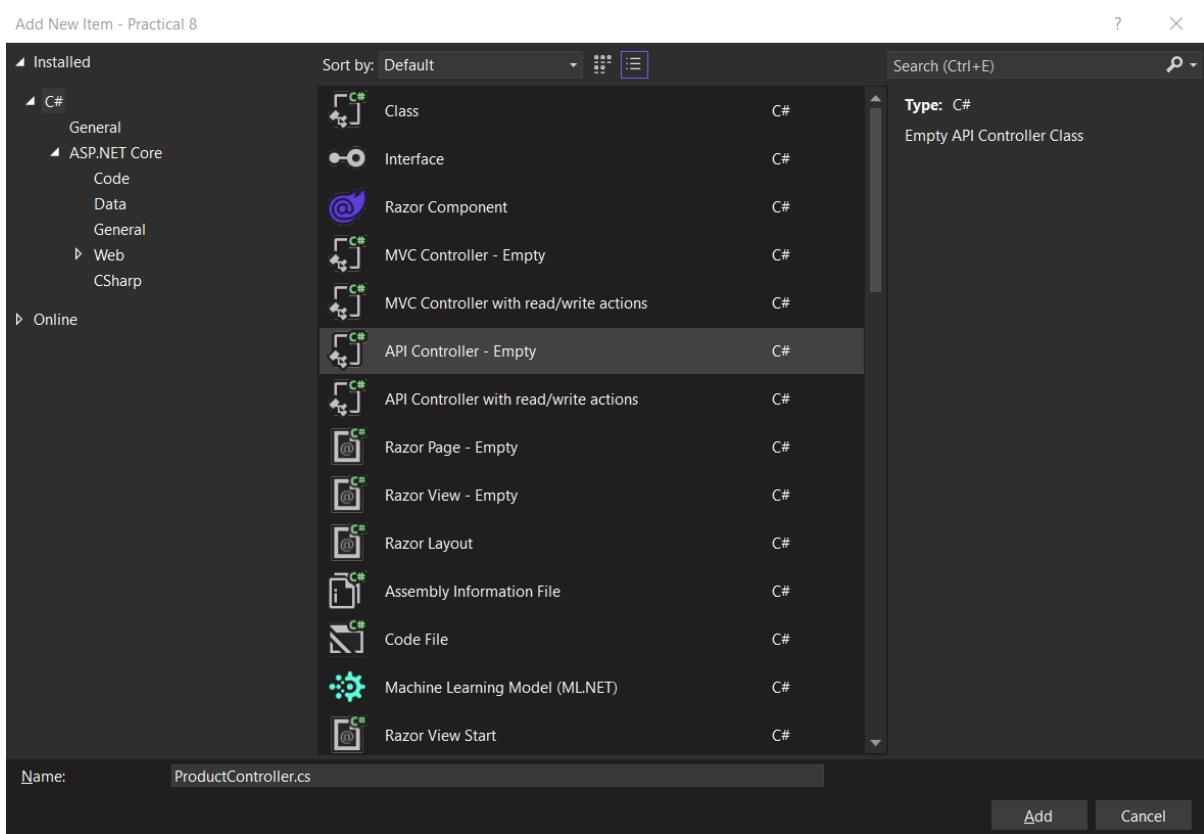
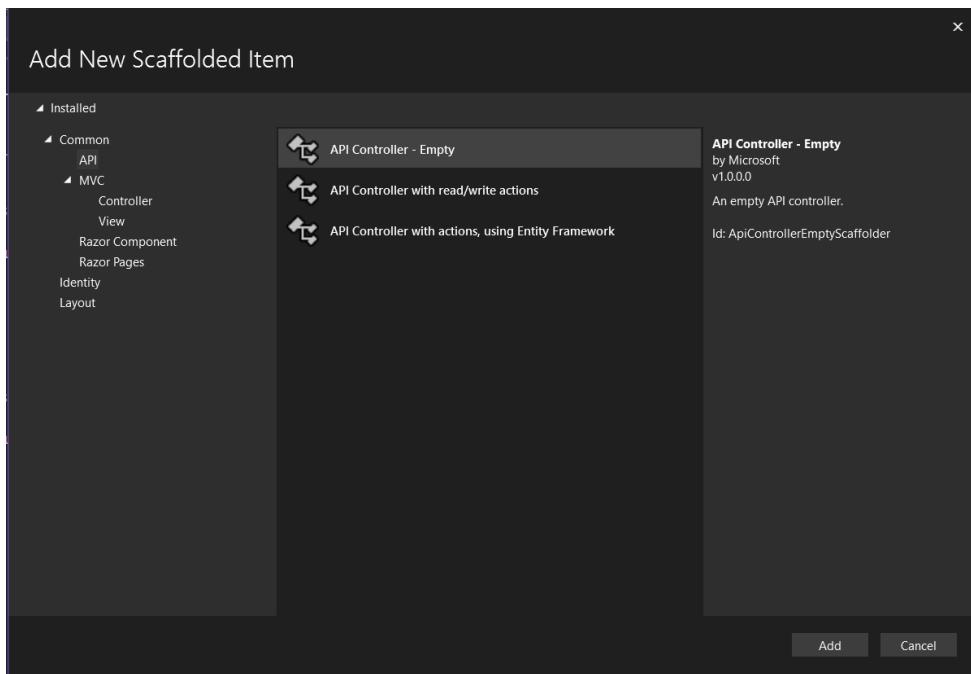


The screenshot shows the Visual Studio code editor with the file `ErrorService.cs` open. The code defines a static class `ErrorService` with two static methods: `GetRequestContentMismatchErrorMessage()` and `GetEntityNotFoundErrorMessage(Type entity, int id)`. Both methods return instances of `ErrorMessage` with specific titles and details.

```
1  using System;
2
3  namespace Practical_8.Errors
4  {
5      public static class ErrorService
6      {
7          public static ErrorMessage GetRequestContentMismatchErrorMessage()
8          {
9              return new RequestContentErrorMessage
10             {
11                 Title = $"Request content mismatch",
12                 Detail = $"Error in the request context."
13             };
14         }
15         public static ErrorMessage GetEntityNotFoundErrorMessage(Type entity, int id)
16         {
17             return new EntityNotFoundErrorMessage
18             {
19                 Title = $"{entity.Name} not found",
20                 Detail = $"No {entity.Name.ToLower()} found for the supplied id - {id}"
21             };
22         }
23     }
24 }
25
```

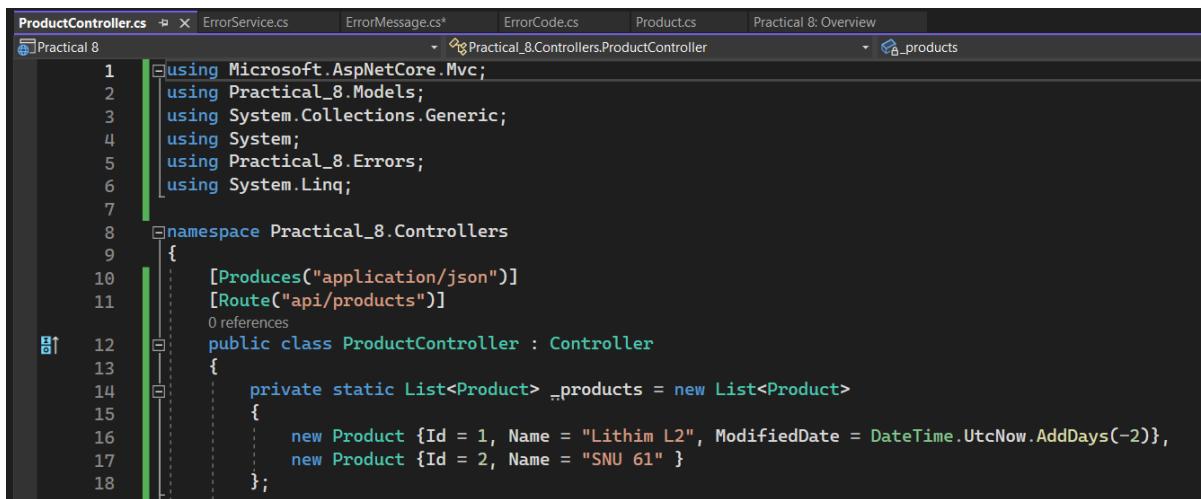
14. Create an **ASP.NET Core controller** with the actions for the CRUD operations of Product entity. In order to this, add an **empty Web API controller** named “**ProductsController.cs**”





15. . First, add a product collection (hardcoded) in the controller as shown below.

```
[Produces("application/json")]
[Route("api/products")]
public class ProductController : Controller
{
    private static List<Product> _products = new List<Product>
    {
        new Product {Id = 1, Name = "Lithium L2", ModifiedDate = DateTime.UtcNow.AddDays(-2)},
        new Product {Id = 2, Name = "SNU 61" }
    };
}
```



```
ProductController.cs ✘ X ErrorService.cs ErrorMessage.cs* ErrorCode.cs Product.cs Practical 8: Overview
Practical 8 - Practical_8.Controllers.ProductController - _products
1  using Microsoft.AspNetCore.Mvc;
2  using Practical_8.Models;
3  using System.Collections.Generic;
4  using System;
5  using Practical_8.Errors;
6  using System.Linq;
7
8  namespace Practical_8.Controllers
9  {
10     [Produces("application/json")]
11     [Route("api/products")]
12     public class ProductController : Controller
13     {
14         private static List<Product> _products = new List<Product>
15         {
16             new Product {Id = 1, Name = "Lithium L2", ModifiedDate = DateTime.UtcNow.AddDays(-2)},
17             new Product {Id = 2, Name = "SNU 61" }
18         };
    }
```

16. Next, let's add **two HTTP GET** actions: one to retrieve all the products, and another to retrieve a product by its ID, parameter is passed via URI path.

```
/// <summary>
/// Gets list of all Products
/// </summary>
/// <returns>List of Products</returns>
/// <response code="200">List of Products</response>
[HttpGet]
[ProducesResponseType(typeof(List<Product>), 200)]
public IActionResult GetProducts()
{
    return Ok(_products);
}

/// <summary>
/// Gets product by id
/// </summary>
/// <param name="id">Product id</param>
/// <returns>Product</returns>
/// <response code="200">Product</response>
/// <response code="404">No Product found for the specified id</response>
[HttpGet("{id}")]
[ProducesResponseType(typeof(Product), 200)]
[ProducesResponseType(typeof(NotFoundErrorMessage), 404)]
public IActionResult GetProductById(int id)
{
    var product = _products.SingleOrDefault(p => p.Id == id);
```

```

        if (product != null)
            return Ok(product);
        else
            return NotFound
                (ErrorService.GetEntityNotFoundErrorMessage(typeof(Product), id));
    }
}

```

```

ProductController.cs  ✘ ErrorService.cs  ErrorMessage.cs  ErrorCode.cs  Products.cs  Practical 8: Overview
Practical 8
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
47
48
49
50
51
52

    /// <summary>
    /// Gets list of all Products
    /// </summary>
    /// <returns>List of Products</returns>
    /// <response code="200">List of Products</response>
    [HttpGet]
    [ProducesResponseType(typeof(List<Product>), 200)]
    0 references
    public IActionResult GetProducts()
    {
        return Ok(_products);
    }

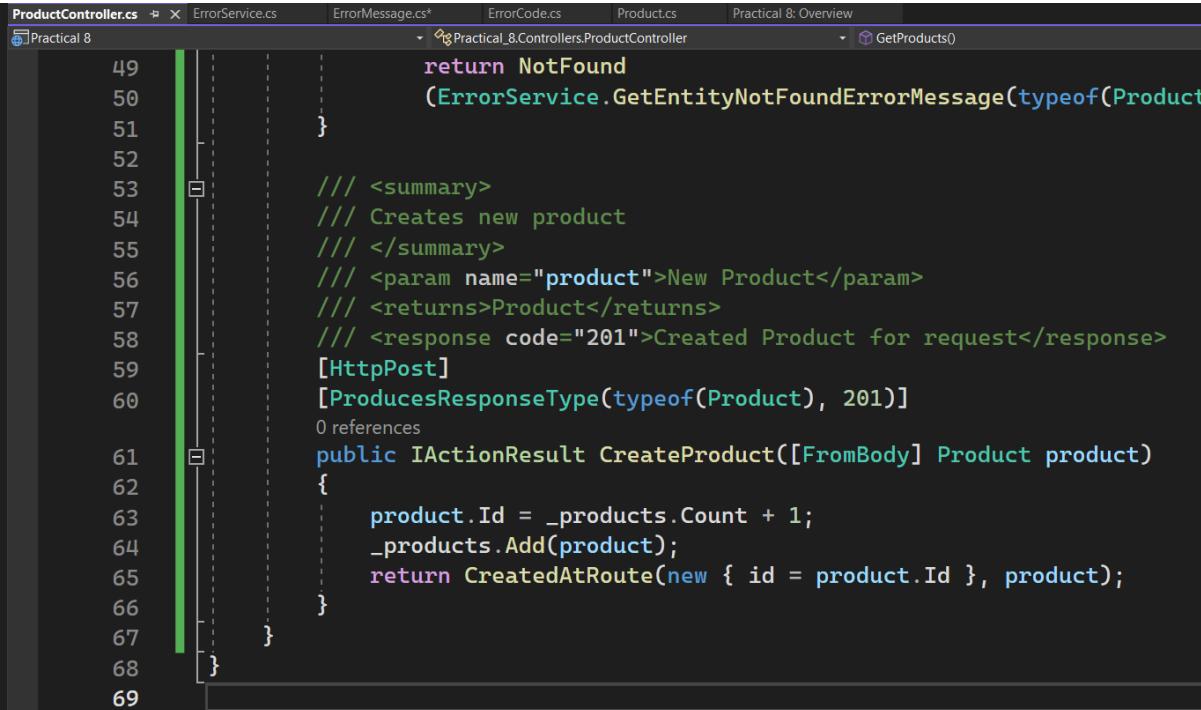
    /// <summary>
    /// Gets product by id
    /// </summary>
    /// <param name="id">Product id</param>
    /// <returns>Product</returns>
    /// <response code="200">Product</response>
    /// <response code="404">No Product found for the specified id</response>

    [HttpGet("{id}")]
    [ProducesResponseType(typeof(Product), 200)]
    [ProducesResponseType(typeof(EntityNotFoundErrorMessage), 404)]
    0 references
    public IActionResult GetProductById(int id)
    {
        var product = _products.SingleOrDefault(p => p.Id == id);
        if (product != null)
            return Ok(product);
        else
            return NotFound
                (ErrorService.GetEntityNotFoundErrorMessage(typeof(Product), id));
    }
}

```

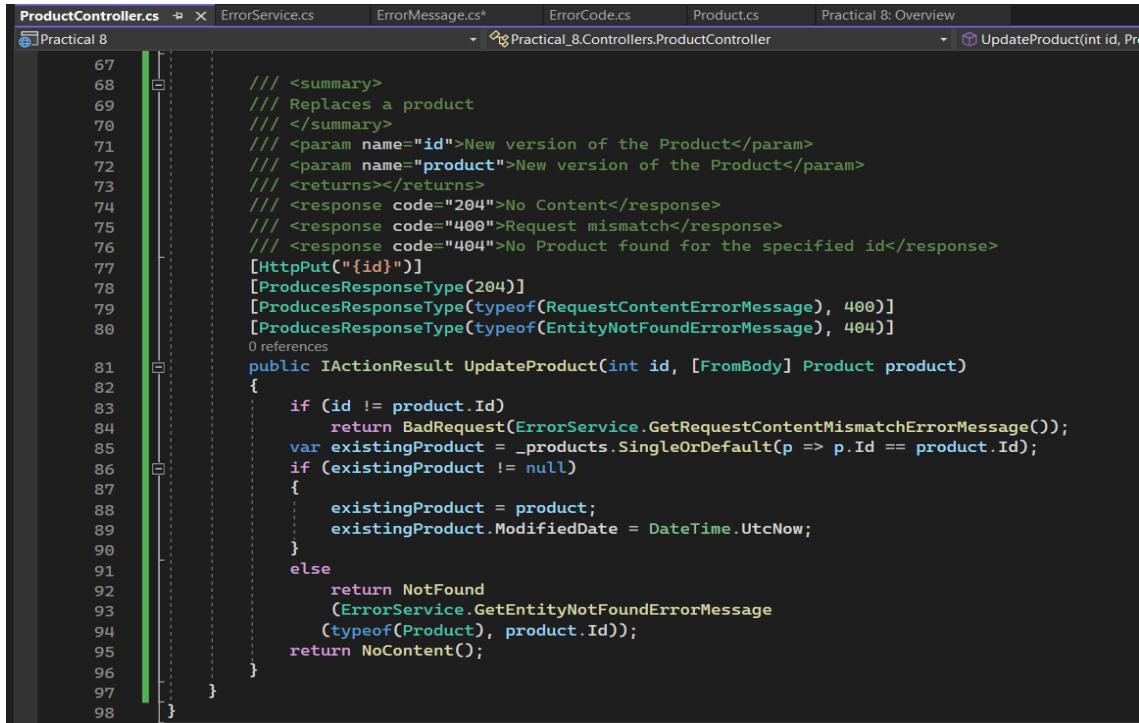
17. Add an action to create new products. **HTTP POST** creates a new product, taking the parameter in the request body.

```
    /// <summary>
    /// Creates new product
    /// </summary>
    /// <param name="product">New Product</param>
    /// <returns>Product</returns>
    /// <response code="201">Created Product for request</response>
    [HttpPost]
    [ProducesResponseType(typeof(Product), 201)]
    public IActionResult CreateProduct([FromBody] Product product)
    {
        product.Id = _products.Count + 1;
        _products.Add(product);
        return CreatedAtRoute(new { id = product.Id }, product);
    }
```



18. Add the **PUT** method for replacing products with the specified ID.

```
    /// <summary>
    /// Replaces a product
    /// </summary>
    /// <param name="id">New version of the Product</param>
    /// <param name="product">New version of the Product</param>
    /// <returns></returns>
    /// <response code="204">No Content</response>
    /// <response code="400">Request mismatch</response>
    /// <response code="404">No Product found for the specified id</response>
    [HttpPut("{id}")]
    [ProducesResponseType(204)]
    [ProducesResponseType(typeof(RequestContentErrorMessage), 400)]
    [ProducesResponseType(typeof(EntityNotFoundException), 404)]
    public IActionResult UpdateProduct(int id, [FromBody] Product product)
    {
        if (id != product.Id)
            return BadRequest(ErrorService.GetRequestContentMismatchErrorMessage());
        var existingProduct = _products.SingleOrDefault(p => p.Id == product.Id);
        if (existingProduct != null)
        {
            existingProduct = product;
            existingProduct.ModifiedDate = DateTime.UtcNow;
        }
        else
            return NotFound(
                ErrorService.GetEntityNotFoundErrorMessage(
                    typeof(Product), product.Id));
        return NoContent();
    }
```

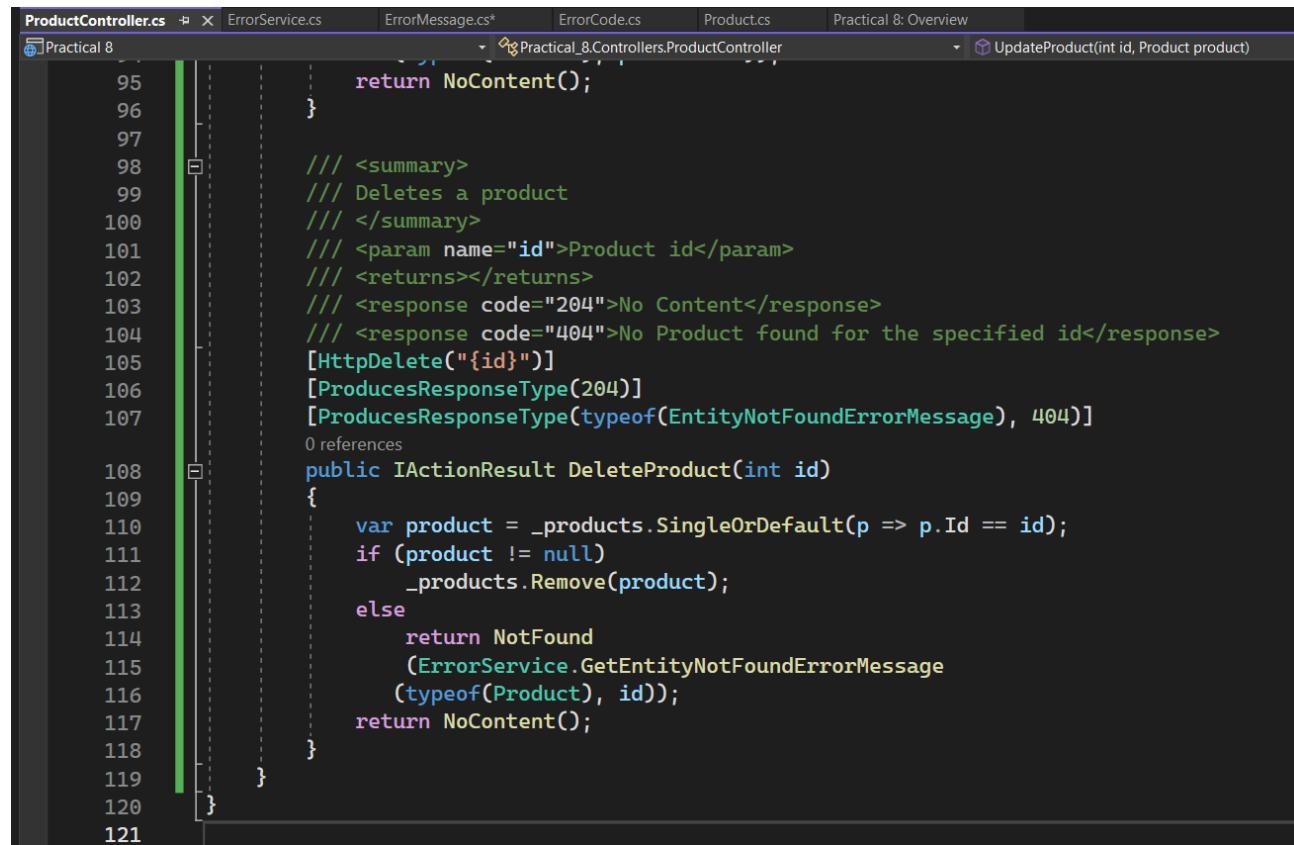


```
ProductController.cs  ✘ ErrorService.cs  ErrorMessage.cs*  ErrorCode.cs  Product.cs  Practical 8: Overview
Practical 8
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
```

```
    /// <summary>
    /// Replaces a product
    /// </summary>
    /// <param name="id">New version of the Product</param>
    /// <param name="product">New version of the Product</param>
    /// <returns></returns>
    /// <response code="204">No Content</response>
    /// <response code="400">Request mismatch</response>
    /// <response code="404">No Product found for the specified id</response>
    [HttpPut("{id}")]
    [ProducesResponseType(204)]
    [ProducesResponseType(typeof(RequestContentErrorMessage), 400)]
    [ProducesResponseType(typeof(EntityNotFoundException), 404)]
    public IActionResult UpdateProduct(int id, [FromBody] Product product)
    {
        if (id != product.Id)
            return BadRequest(ErrorService.GetRequestContentMismatchErrorMessage());
        var existingProduct = _products.SingleOrDefault(p => p.Id == product.Id);
        if (existingProduct != null)
        {
            existingProduct = product;
            existingProduct.ModifiedDate = DateTime.UtcNow;
        }
        else
            return NotFound(
                ErrorService.GetEntityNotFoundErrorMessage(
                    typeof(Product), product.Id));
        return NoContent();
    }
```

19. Add a delete endpoint using the **HTTP DELETE** action.

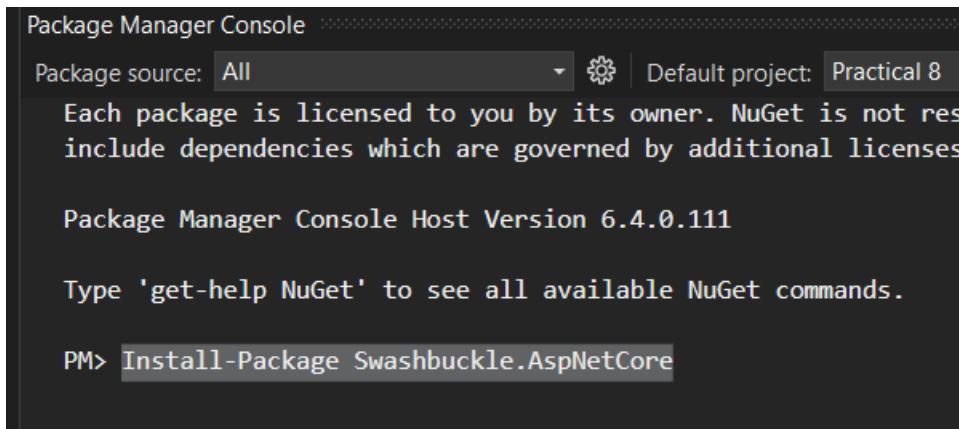
```
/// <summary>
/// Deletes a product
/// </summary>
/// <param name="id">Product id</param>
/// <returns></returns>
/// <response code="204">No Content</response>
/// <response code="404">No Product found for the specified id</response>
[HttpDelete("{id}")]
[ProducesResponseType(204)]
[ProducesResponseType(typeof(EntityNotFoundException), 404)]
public IActionResult DeleteProduct(int id)
{
    var product = _products.SingleOrDefault(p => p.Id == id);
    if (product != null)
        _products.Remove(product);
    else
        return NotFound
            (ErrorService.GetEntityNotFoundException
            (typeof(Product), id));
    return NoContent();
}
```



The screenshot shows the ProductController.cs file in Visual Studio. The code is identical to the one provided in the text block, implementing the DeleteProduct action. The code includes XML documentation for the action, parameters, and responses. The IDE interface shows tabs for other files like ErrorService.cs, ErrorMessage.cs, ErrorCode.cs, and Product.cs, along with a navigation bar and status bar.

- 20.** Install the package **Swashbuckle.AspNetCore** in the project by executing the following command in the **Package Manager Console (PMC)**:

Install-Package Swashbuckle.AspNetCore



Package Manager Console

Package source: All Default project: Practical 8

Each package is licensed to you by its owner. NuGet is not responsible for the contents of this feed. [Read more...](#)

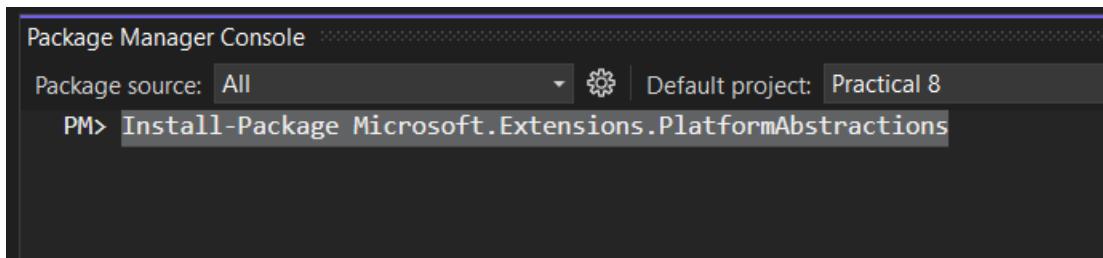
Package Manager Console Host Version 6.4.0.111

Type 'get-help NuGet' to see all available NuGet commands.

PM> **Install-Package Swashbuckle.AspNetCore**

- 21.** In the sample, in order to provide the XML path, install the package **Microsoft.Extensions.PlatformAbstractions** using **PMC**. Execute the following:

Install-Package Microsoft.Extensions.PlatformAbstractions



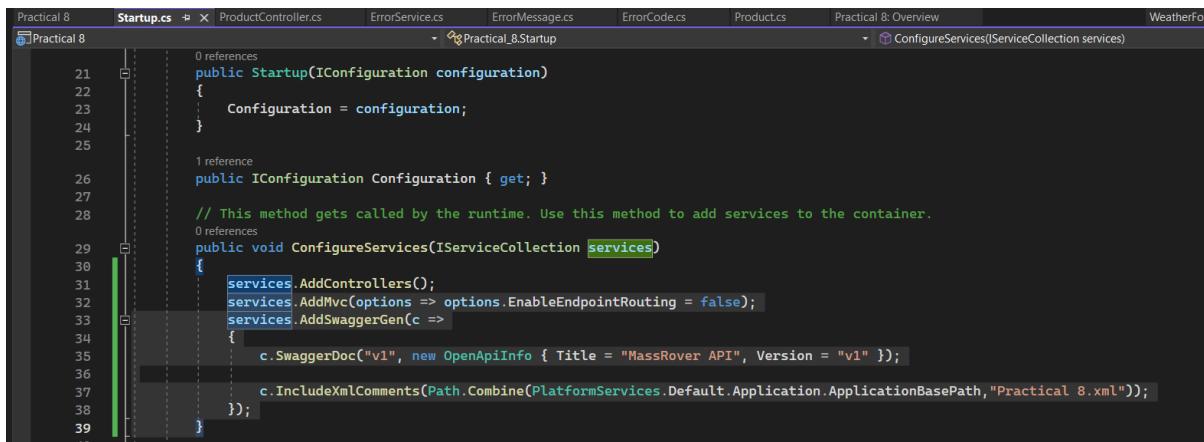
Package Manager Console

Package source: All Default project: Practical 8

PM> **Install-Package Microsoft.Extensions.PlatformAbstractions**

- 22.** In the **Startup.cs**, update the **ConfigureServices** method, as shown below by Adding the following code:

```
public void ConfigureServices(IServiceCollection services)
{
    services.AddControllers();
    services.AddMvc(options => options.EnableEndpointRouting = false);
    services.AddSwaggerGen(c =>
    {
        c.SwaggerDoc("v1", new OpenApiInfo { Title = "MassRover API", Version = "v1" });
        c.IncludeXmlComments(Path.Combine(PlatformServices.Default.Application.ApplicationBasePath, "Practical 8.xml"));
    });
}
```



```
Practical 8 Startup.cs ProductController.cs ErrorService.cs ErrorMessage.cs ErrorCode.cs Product.cs Practical 8: Overview WeatherForConfigureServices(IServiceCollection services)

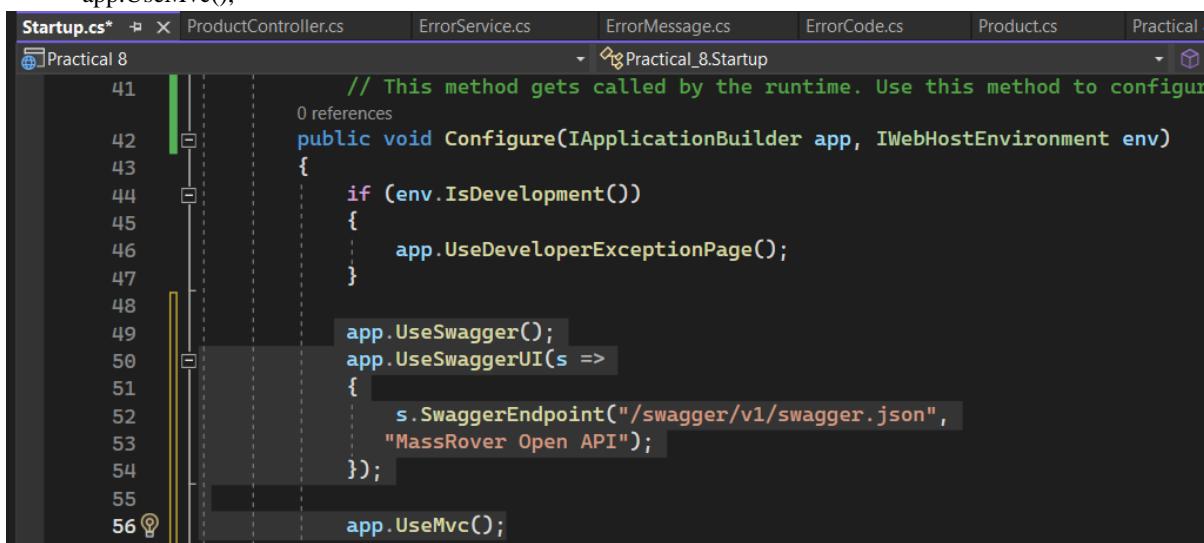
21 0 references
22 public Startup(IConfiguration configuration)
23 {
24     Configuration = configuration;
25 }
26 1 reference
27 public IConfiguration Configuration { get; }
28
29 // This method gets called by the runtime. Use this method to add services to the container.
30 0 references
31 public void ConfigureServices(IServiceCollection services)
32 {
33     services.AddControllers();
34     services.AddMvc(options => options.EnableEndpointRouting = false);
35     services.AddSwaggerGen(c =>
36     {
37         c.SwaggerDoc("v1", new OpenApiInfo { Title = "MassRover API", Version = "v1" });
38
39         c.IncludeXmlComments(Path.Combine(PlatformServices.Default.Application.ApplicationBasePath, "Practical 8.xml"));
        });
    }
}
```

23. Inside **Startup.cs** file, Update the **Configure** method as shown below. This enables the Swagger UI and sets the Swagger definition endpoint.

```
public void Configure(IApplicationBuilder app, IWebHostEnvironment env)
{
    if (env.IsDevelopment())
    {
        app.UseDeveloperExceptionPage();
    }

    app.UseSwagger();
    app.UseSwaggerUI(s =>
    {
        s.SwaggerEndpoint("/swagger/v1/swagger.json", "MassRover Open API");
    });

    app.UseMvc();
}
```



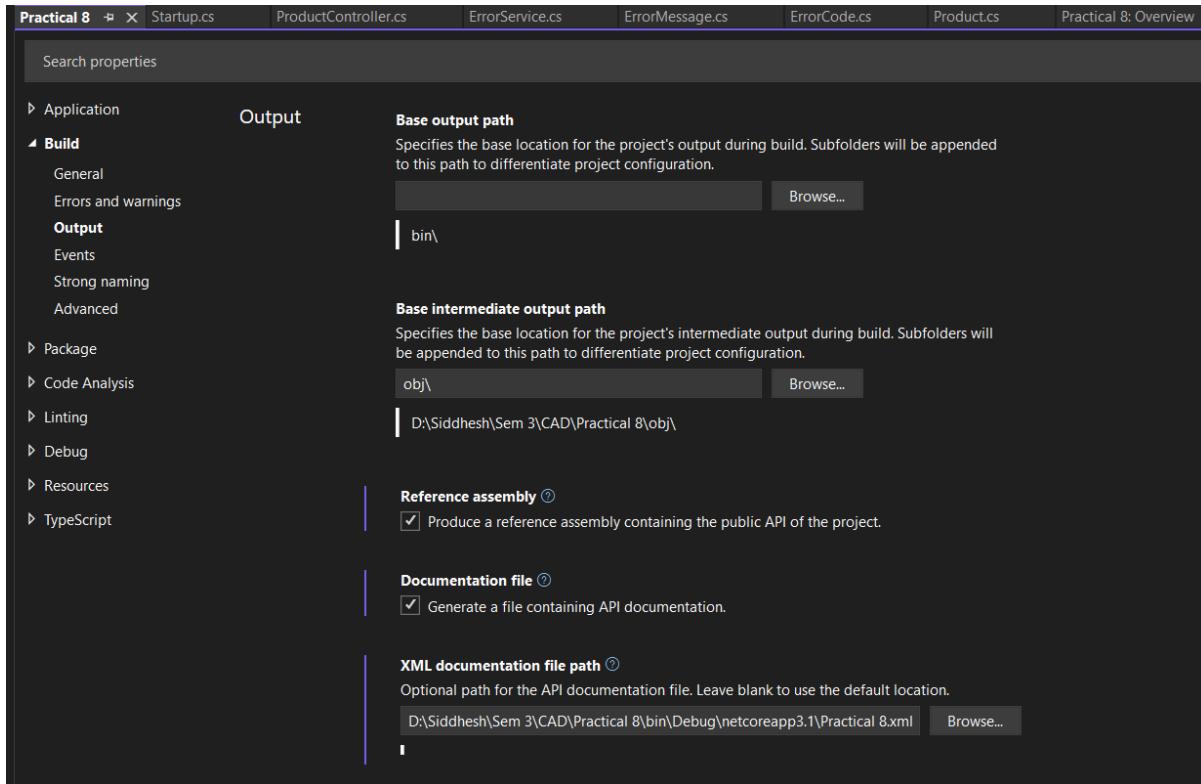
```
Practical 8 Startup.cs ProductController.cs ErrorService.cs ErrorMessage.cs ErrorCode.cs Product.cs Practical 8: Overview WeatherForConfigureServices(IServiceCollection services)

41 // This method gets called by the runtime. Use this method to configure your application.
42 public void Configure(IApplicationBuilder app, IWebHostEnvironment env)
43 {
44     if (env.IsDevelopment())
45     {
46         app.UseDeveloperExceptionPage();
47     }

48     app.UseSwagger();
49     app.UseSwaggerUI(s =>
50     {
51         s.SwaggerEndpoint("/swagger/v1/swagger.json",
52                         "MassRover Open API");
53     });
54
55     app.UseMvc();
56 }
```

24. Finally, we will instruct Visual Studio to generate **XML documentation** based on the **comments**. Navigate to **Project Properties**, then the **Build tab**, and enable the **XML documentation file**. Give it the path of your ‘**.XML**’ file which will be inside your project path i.e.

....\Project\bin\Debug\netcoreapp3.1\Project.xml



25. Now you can **Run** the application.



26. Navigate to the **URL**:

`http://{host}:{specified port}/swagger`.

In This case, <http://localhost:44372/swagger>

The screenshot shows the Swagger UI interface for the MassRover API. The top navigation bar includes tabs for 'Select a definition' (set to 'MassRover Open API') and 'Other favorites'. Below the navigation is the API title 'MassRover API' with version 'v1' and 'OAS3' indicators. The main content area is organized into sections: 'Product' and 'WeatherForecast'. Under 'Product', there are five entries: 'GET /api/products' (Gets list of all Products), 'POST /api/products' (Creates new product), 'GET /api/products/{id}' (Gets product by id), 'PUT /api/products/{id}' (Replaces a product), and 'DELETE /api/products/{id}' (Deletes a product). The 'DELETE' method is highlighted with a red border. Under 'WeatherForecast', there is one entry: 'GET /WeatherForecast'. On the right side of the interface, there is a vertical sidebar with icons for file operations like copy, paste, and search.

<https://stackoverflow.com/questions/21175781/a-namespace-cannot-directly-contain-members-such-as-fields-or-methods> - It Shlould be Inside Class

<https://stackoverflow.com/questions/56502869/info-class-not-recognized-with-swashbuckle-and-swagger-libraries> - It has been replaced with OpenApiInfo so you have to reflect that in your swagger setup:

```
c.SwaggerDoc("v1", new OpenApiInfo { Version = "1.0", Title = "My API" });
```

<https://stackoverflow.com/questions/57684093/using-usemvc-to-configure-mvc-is-not-supported-while-using-endpoint-routing> - 3. Disable endpoint Routing. As the exception message suggests and as mentioned in the following section of documentation: [use mvcwithout endpoint routing](#)

```
services.AddMvc(options => options.EnableEndpointRouting = false);
//OR
services.AddControllers(options => options.EnableEndpointRouting = false);
```