Part 1: Front-End Development (ASP.NET MVC and JavaScript):

ASP.NET MVC web application Prerequisites:-

Software's required:

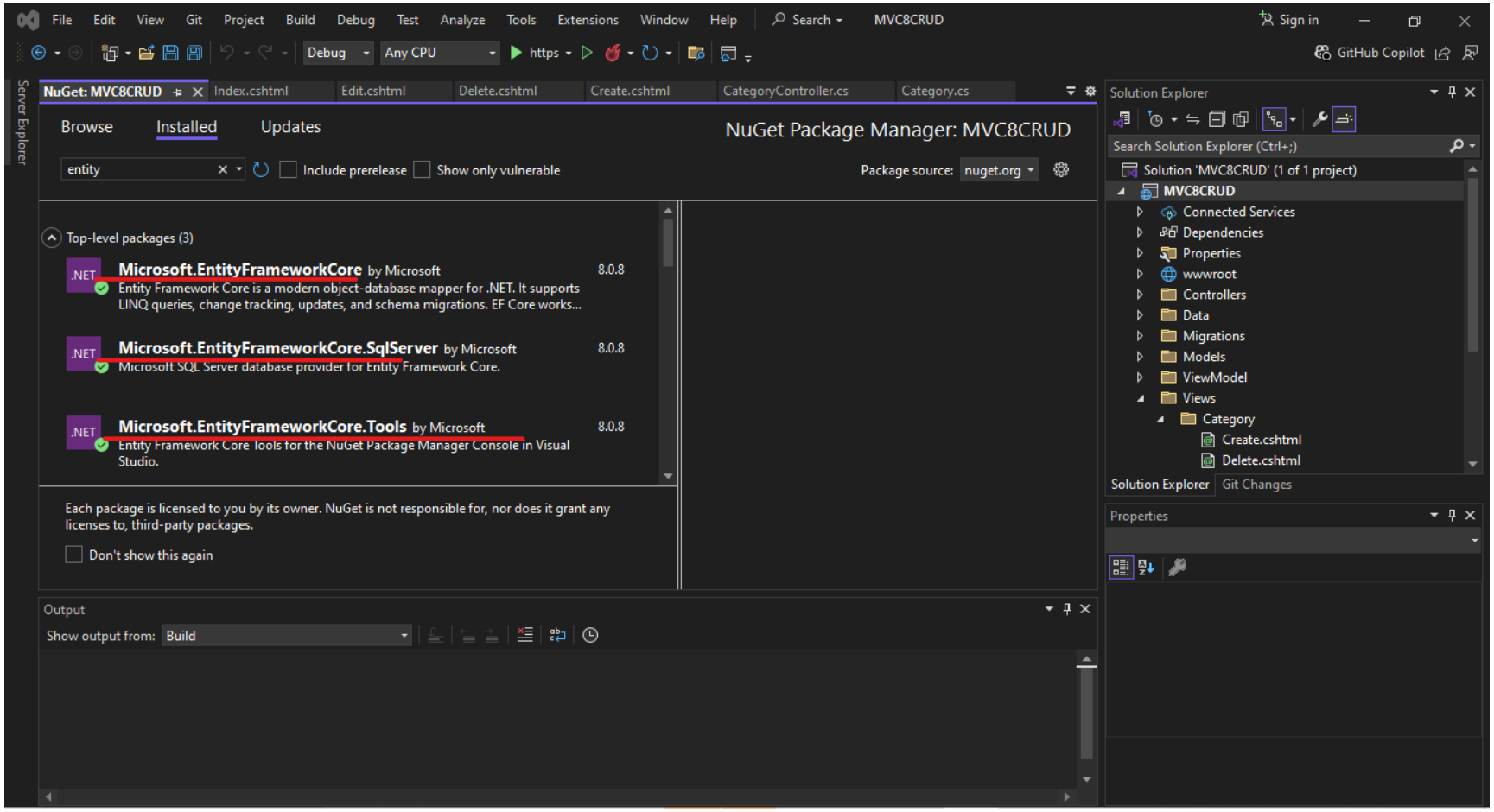
1. **Visual studio**
2. **SQL server management studio**

Install EF Core DB Providers

Go to Tools > NuGet Package Manager > Manage NuGet Packages for Solution and search for

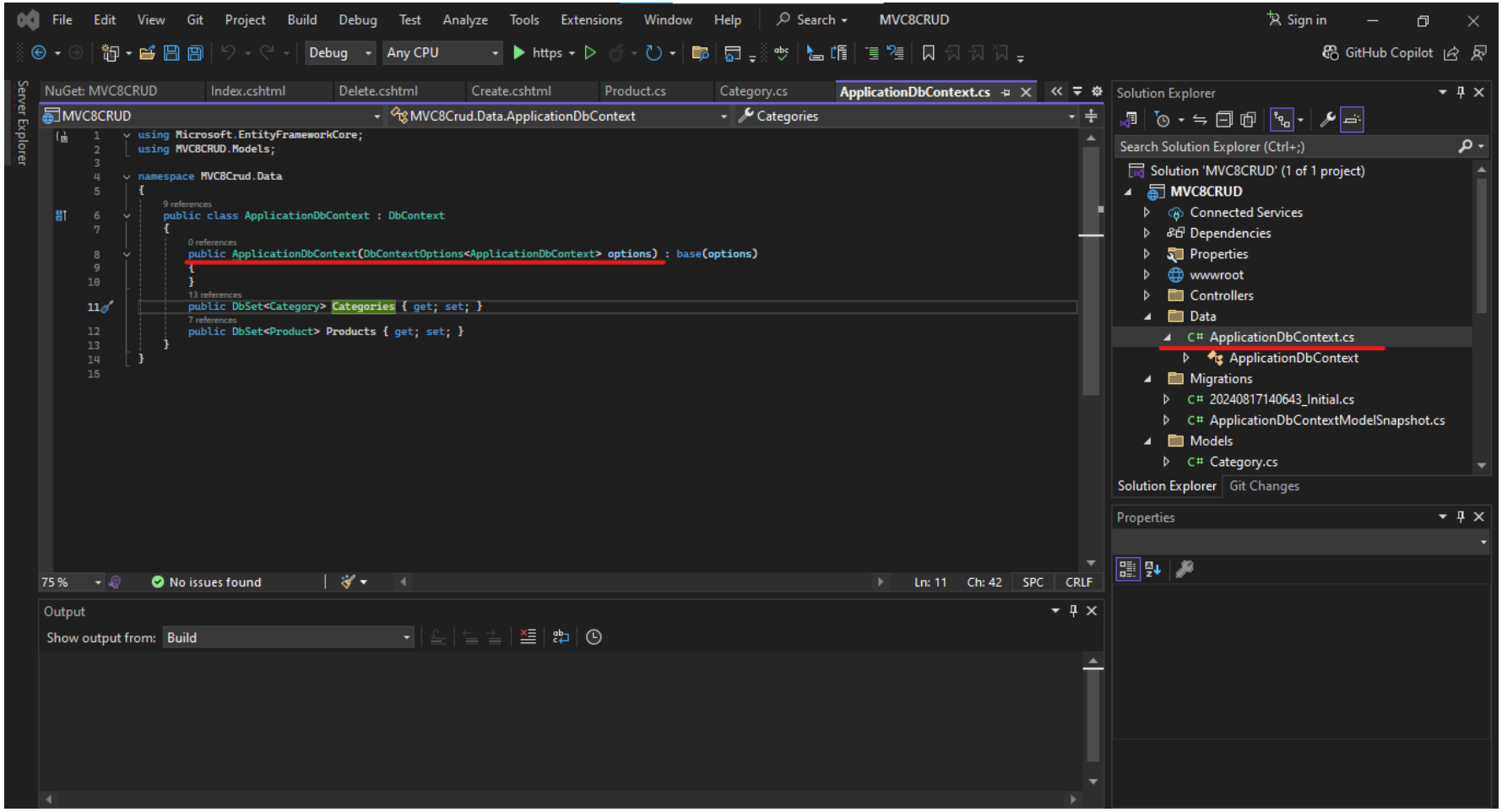
1. Microsoft.EntityFrameworkCore.
2. Microsoft.EntityFrameworkCore.SqlServer.
3. Microsoft.EntityFrameworkCore.Tools

1. **Install EF Code tools packages in the solution created in visual studio as highlighted below.**



2. Adding EF Core DbContext File

We now have to add Entity Framework Core Database Context file for defining your database and tables. So, created a new class called ApplicationDbContext.cs in data folder



3. Creating a database using EF core migration

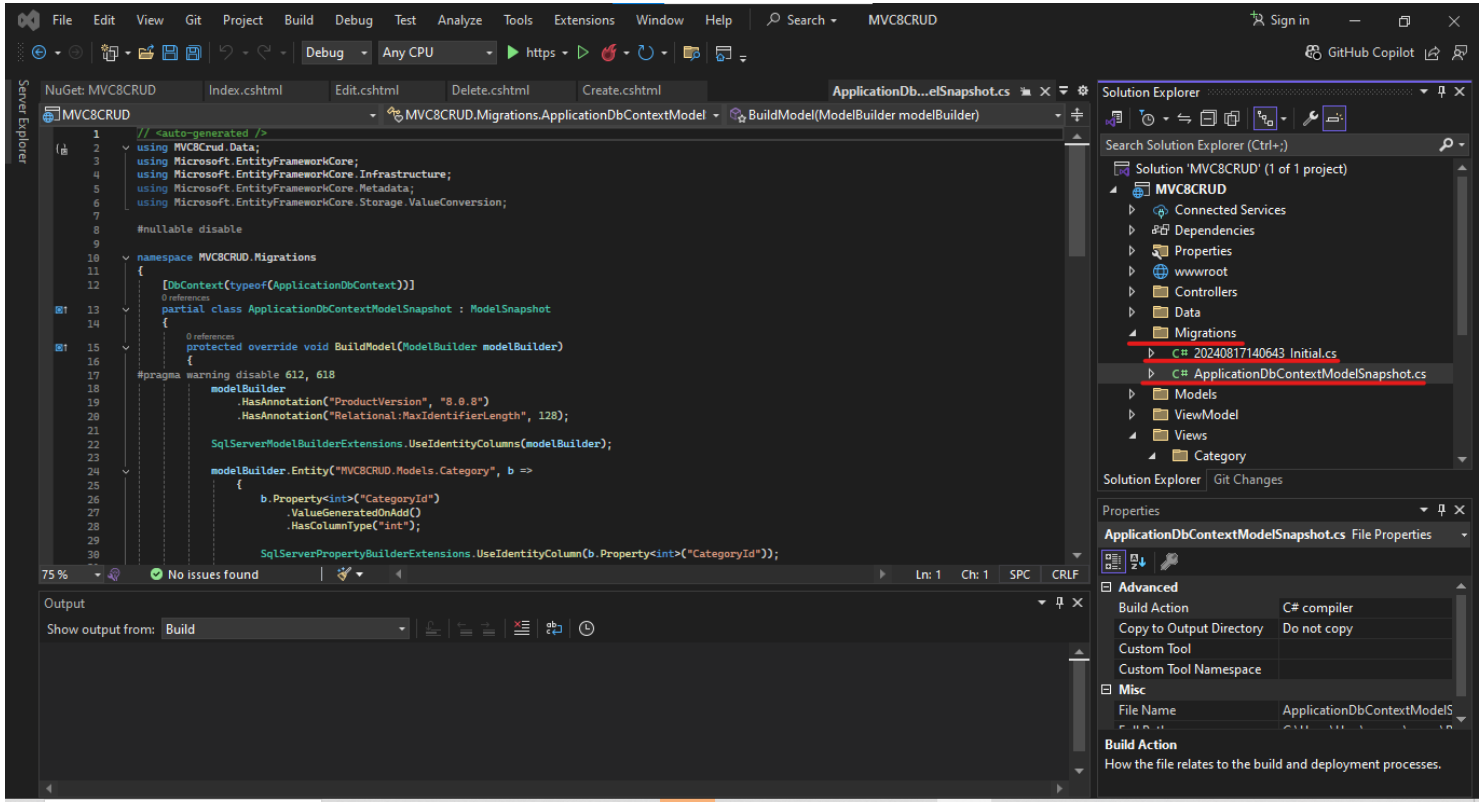
Now run the Migrations in Entity Framework Core so that the '[MVC8CrudDb]' database and '[dbo].[Categories]' table are created. So open the Package Manager Console window in Visual Studio then run this below command,

PM> add-migration Initial

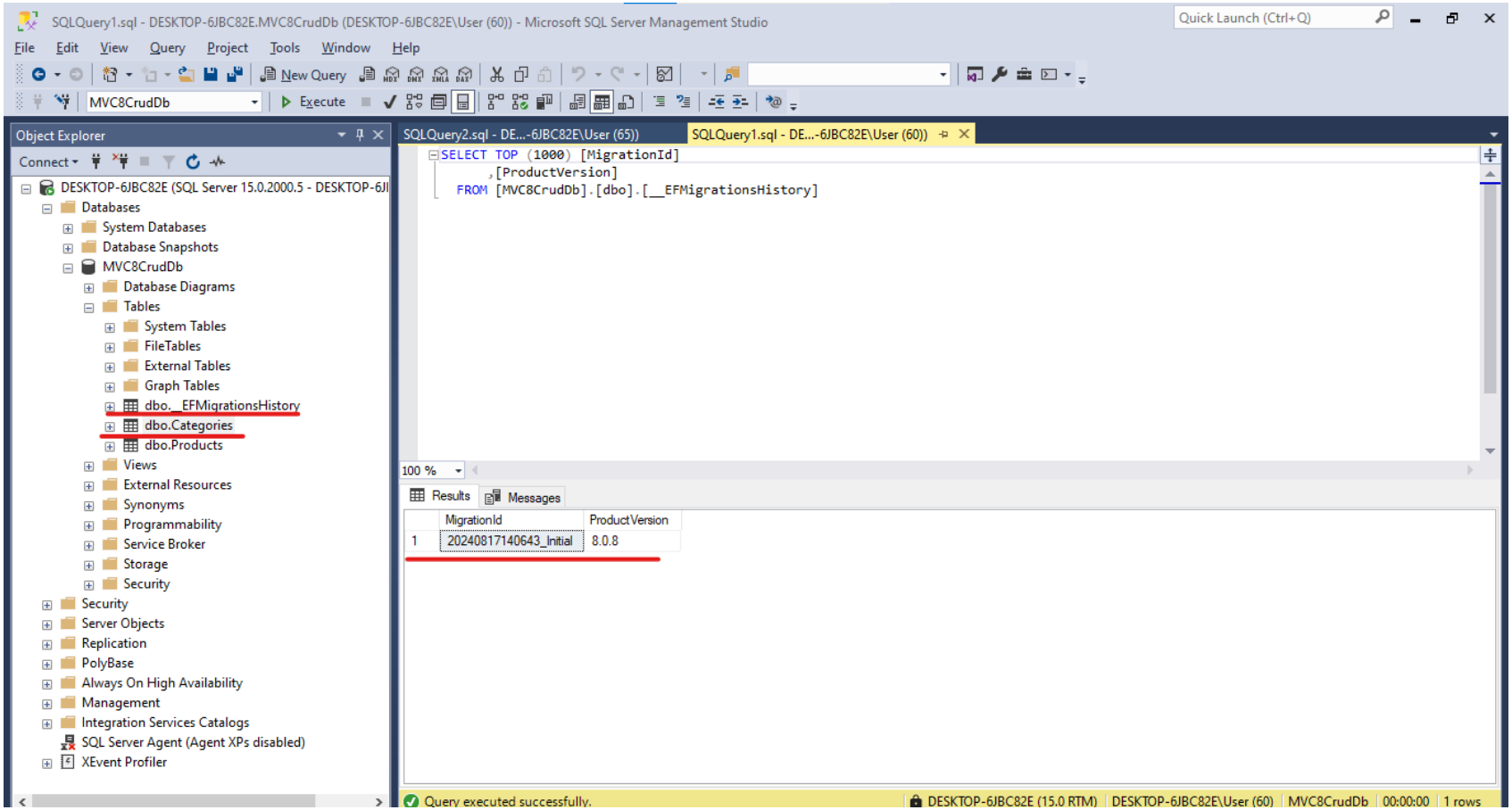
C#

This command will create Migration classes inside Migrations folder in the root of the app. Next, execute the migrations by the below command,

PM> Update-Database

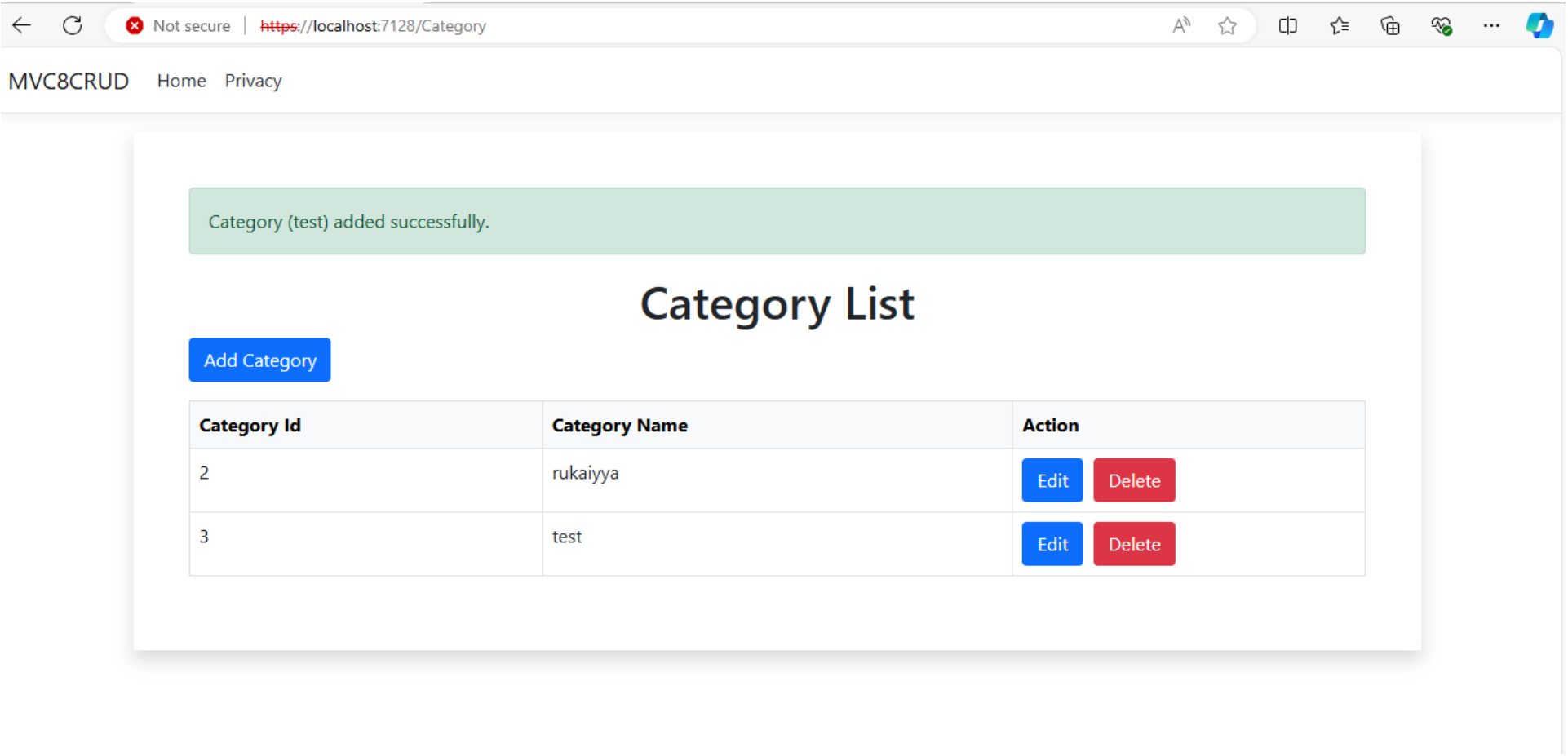


This will create the database and table in your SQL Server LocalDB database.

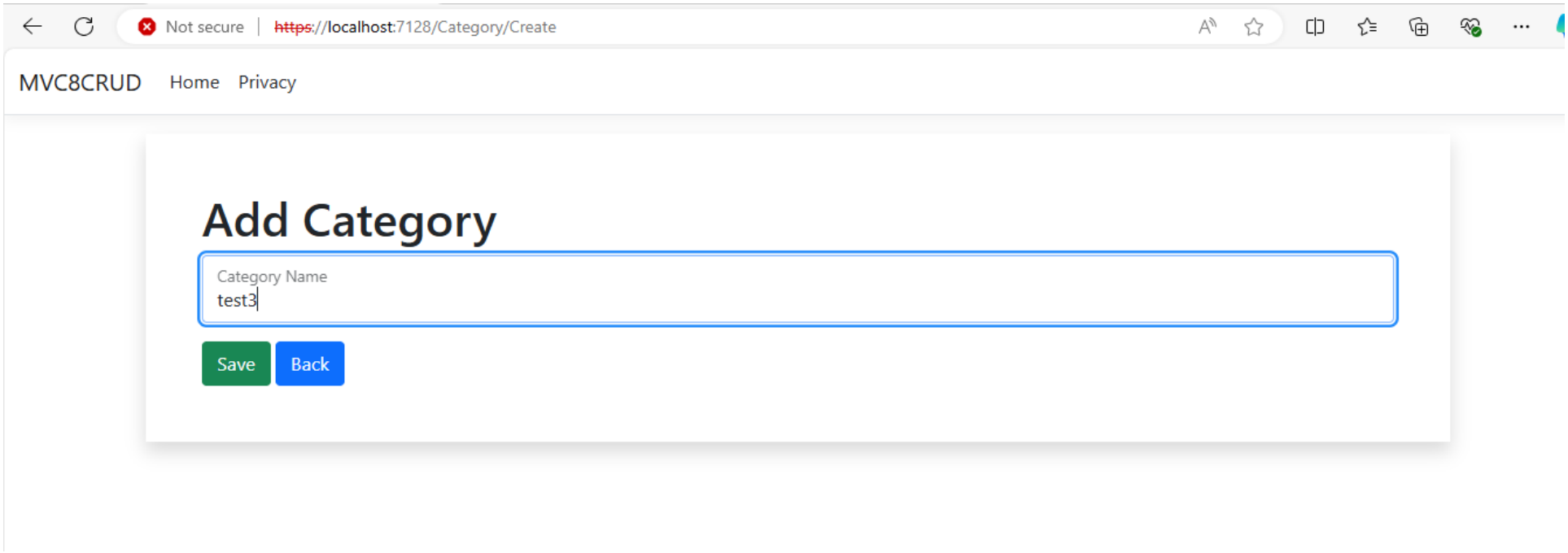


ASP.NET MVC web application with two views: one for displaying a list of items and another for adding a new item.

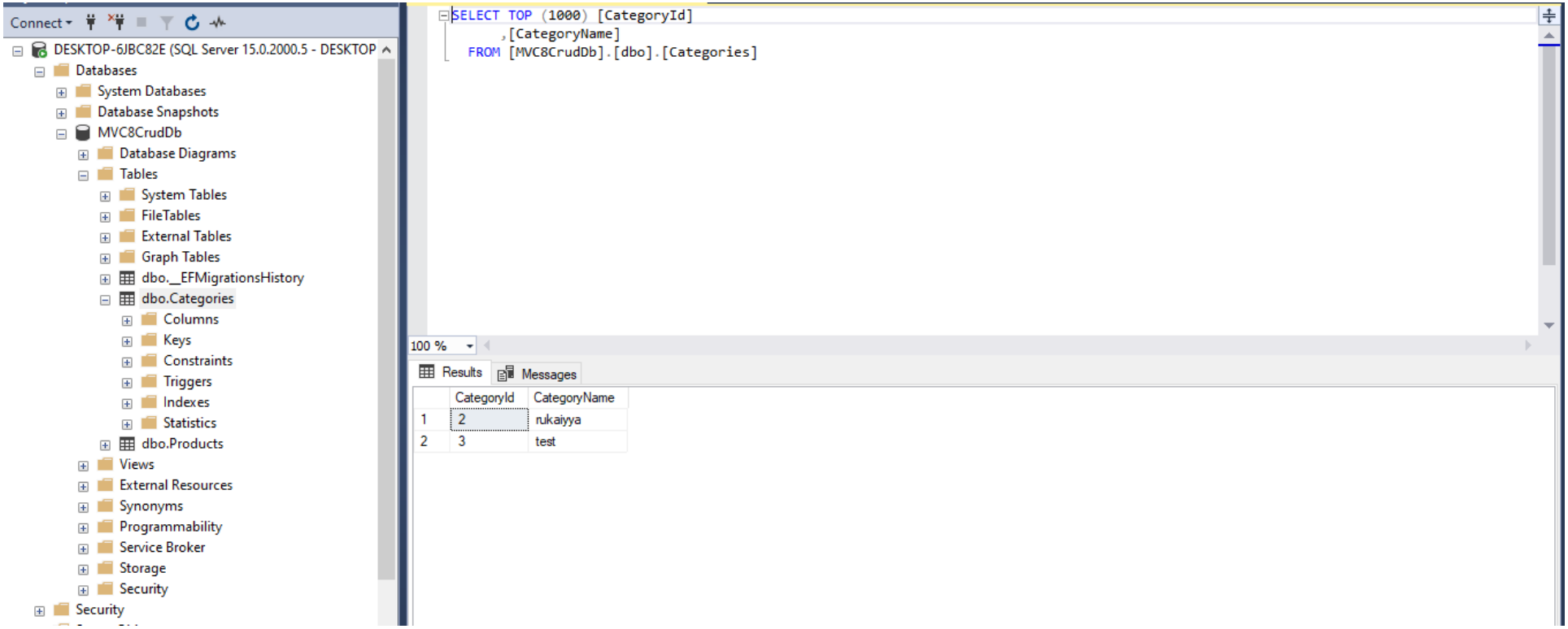
Displaying category



Adding category



Database Interaction entry.



Model to represent the items.

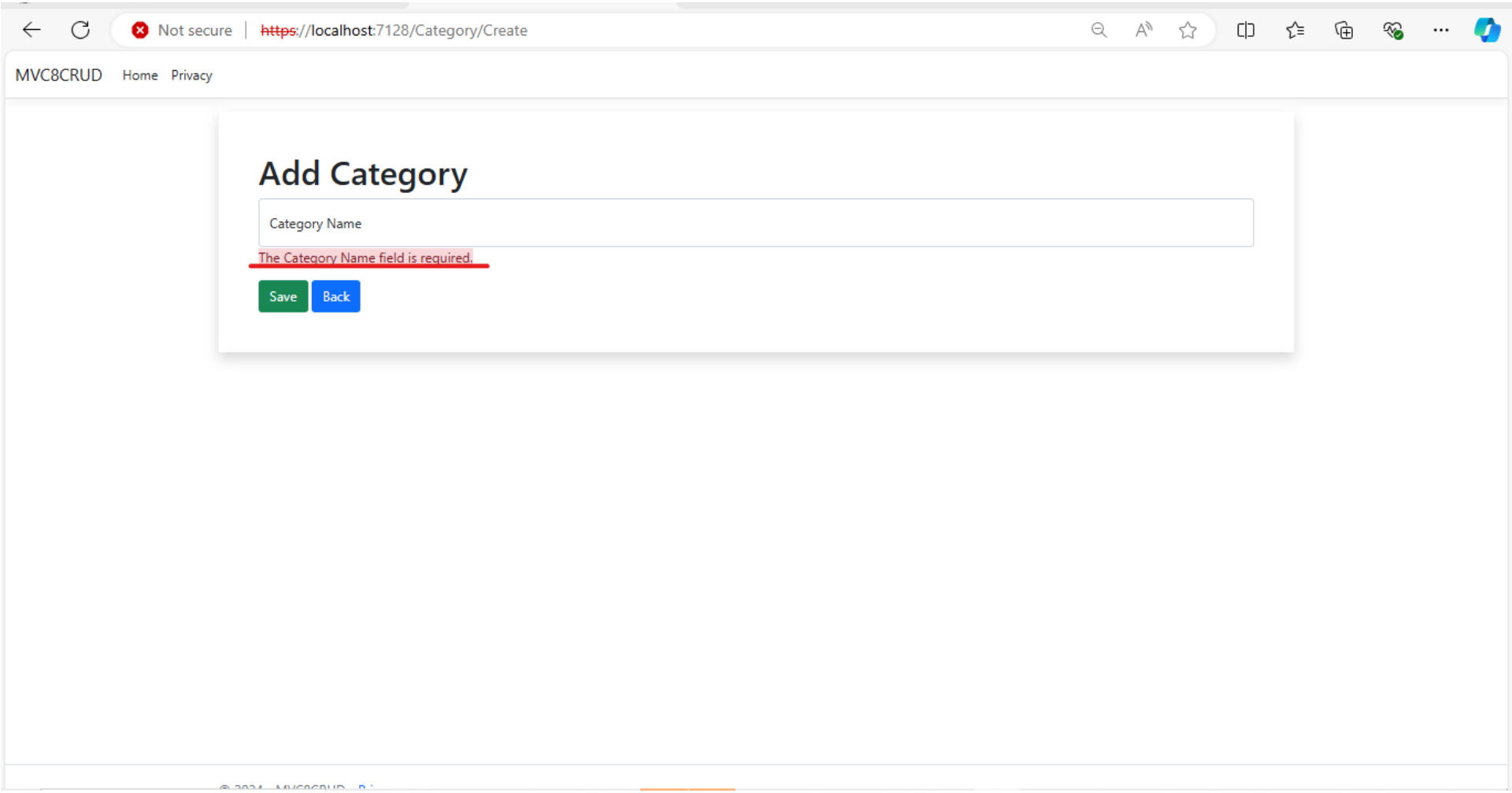
```
using System.ComponentModel.DataAnnotations;

namespace MVC8CRUD.Models
{
    16 references
    public class Category
    {
        [Key]
        13 references
        public int CategoryId { get; set; }
        [Required]
        [Display(Name = "Category Name")]
        16 references
        public string? CategoryName { get; set; }
    }
}
```

Implementing a controller to handle CRUD operations for the items.

Controller with create code.

```
        return View(categoryList);
    }

    0 references
    public IActionResult Create()
    {
        return View();
    }

    [HttpPost]
    [ValidateAntiForgeryToken]
    0 references
    public IActionResult Create(Category category)
    {
        if (ModelState.IsValid)
        {
            _context.Categories.Add(category);
            _context.SaveChanges();
            TempData["SuccessMsg"] = "Category (" + category.CategoryName + ") added successfully.";
            return RedirectToAction("Index");
        }
        return View(category);
    }

    0 references
    public IActionResult Edit(int? categoryId)
    {
        var category = _context.Categories.Find(categoryId);

        if (category == null)
        {
            return NotFound();
```

Create code view and Client Side validation for input fields.



Controller with delete code.

```
69                  {
70                      return NotFound();
71                  }
72                  return View(category);
73              }
74
75              [HttpPost]
76              [ValidateAntiForgeryToken]
                0 references
77              public IActionResult DeleteCategory(int? categoryId)
78              {
79                  var category = _context.Categories.Find(categoryId);
80                  if (category == null)
81                  {
82                      return NotFound();
83                  }
84                  _context.Categories.Remove(category);
85                  _context.SaveChanges();
86                  TempData["SuccessMsg"] = "Category (" + category.CategoryName + ") deleted successfully.";
87                  return RedirectToAction("Index");
88              }
89          }
90      }
91
```

Controller with read code

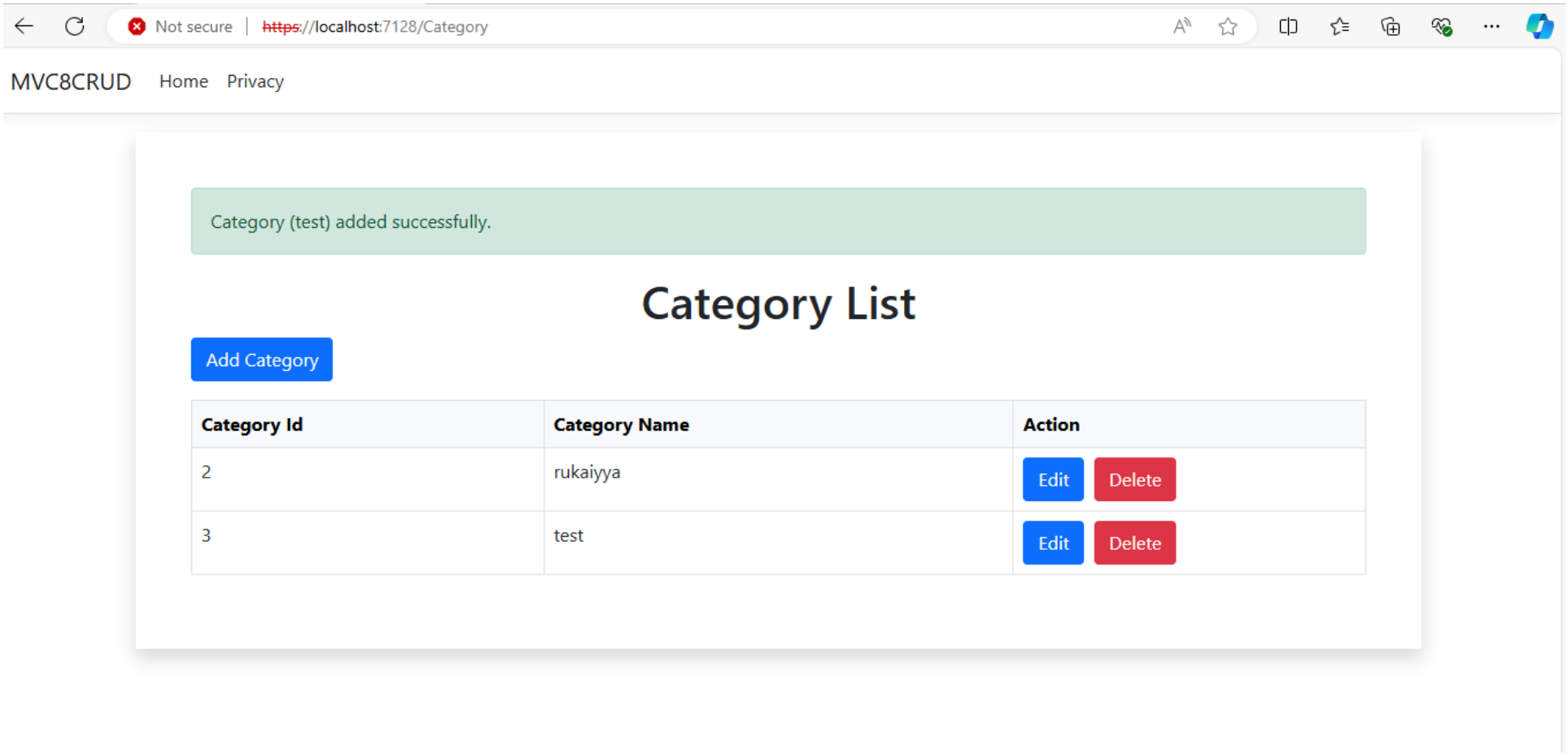
```
9           private readonly ApplicationDbContext _context;
            0 references
10          public CategoryController(ApplicationDbContext context)
11          {
12              _context = context;
13          }
            0 references
14          public IActionResult Index()
15          {
16              IEnumerable<Category> categoryList = _context.Categories;
17              return View(categoryList);
18          }
19
            0 references
20          public IActionResult Create()
21          {
22              return View();
23          }
24
25          [HttpPost]
26          [ValidateAntiForgeryToken]
            0 references
27          public IActionResult Create(Category category)
28          {
29              if (ModelState.IsValid)
30              {
31                  _context.Categories.Add(category);
32                  _context.SaveChanges();
33                  TempData["SuccessMsg"] = "Category (" + category.CategoryName + ") added successfully.";
34                  return RedirectToAction("Index");
35              }
36              return View(category);
37          }
```

View using razor syntax

```razor
@model Category
@{
    ViewData["Title"] = "Create Category";
}

<div class="container shadow p-5">
    <div class="row">
        <h1>Add Category</h1>
    </div>
    <form method="post">
        @*<div asp-validation-summary="All" class="alert-danger"></div> *@
        <div class="form-floating mb-3">
            <input type="text" class="form-control" asp-for="CategoryName" placeholder="Enter category name" />
            <label for="CategoryName">Category Name</label>
            <span asp-validation-for="CategoryName" class="alert-danger"></span>
        </div>
        <button type="submit" class="btn btn-success">Save</button>
        <a asp-controller="Category" asp-action="Index" class="btn btn-primary">Back</a>
    </form>
</div>
```

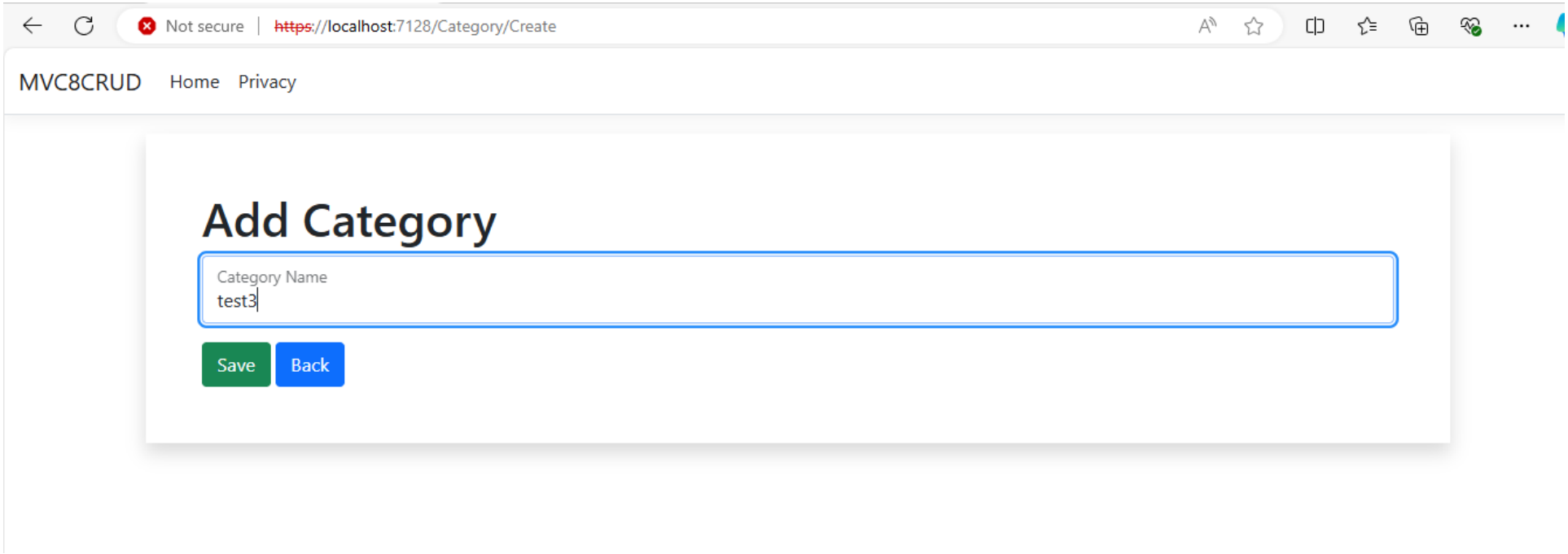
CRUD OPERATIONS PERFORMED HERE

1. Read Operation:

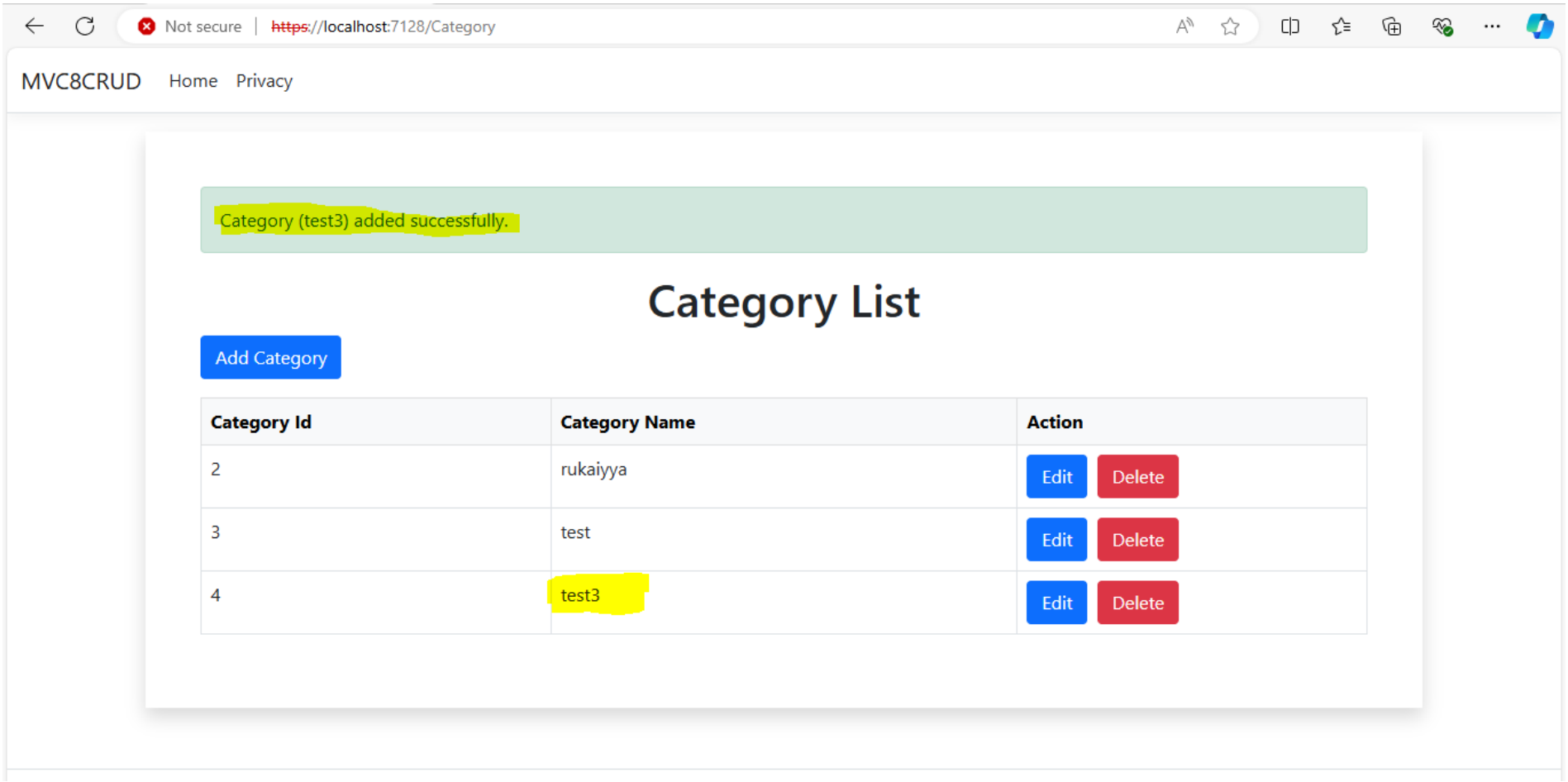


2. Add Operation:-

Adding category test3

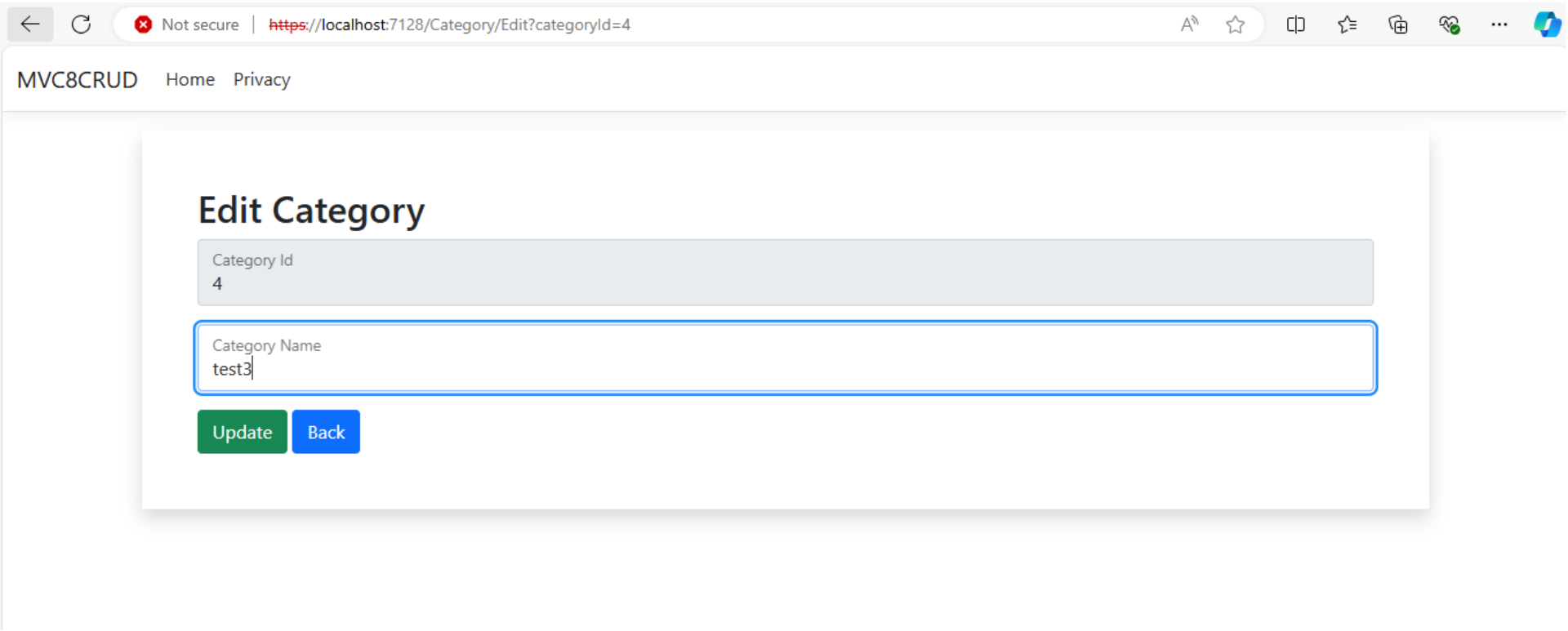


Category test3 with ID 4 added successfully

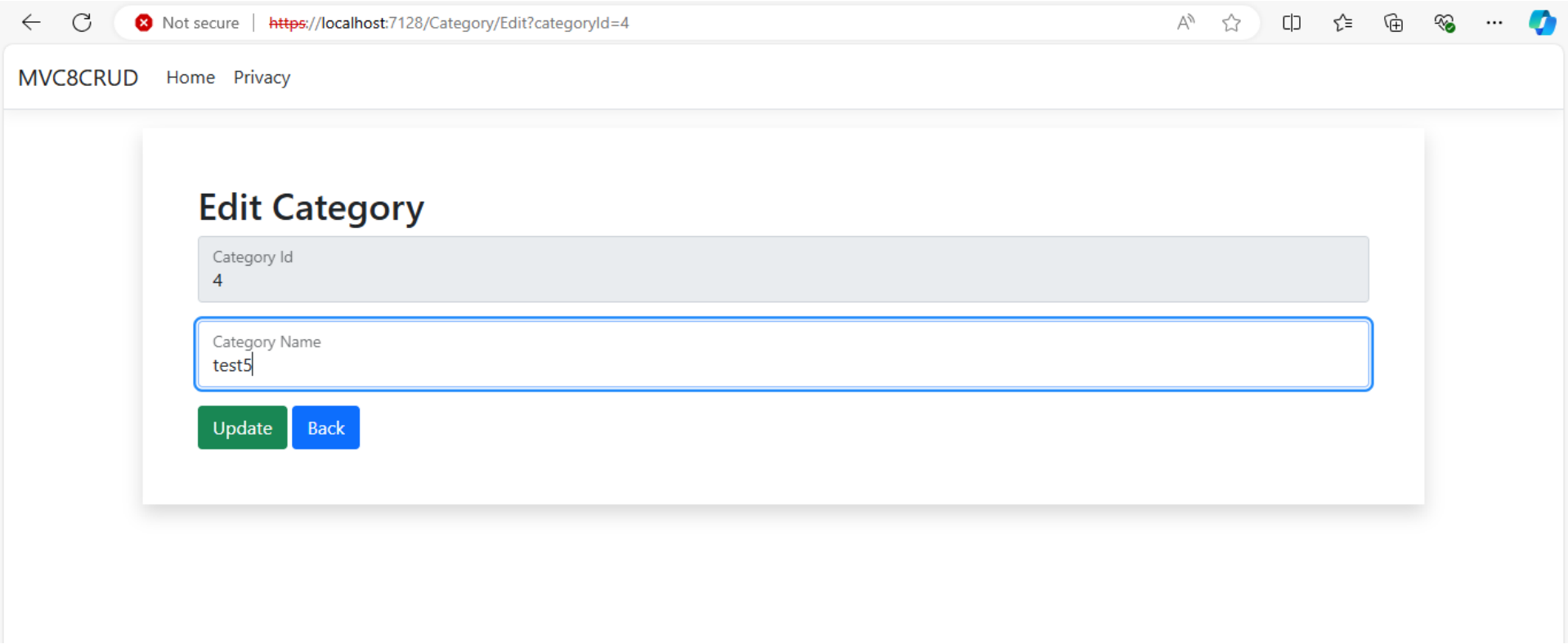


3. Edit/Update Operation:-

Category field before update (test 3)



Category field after update (test3 to test5)



Category with ID 4 updated successfully to (test5)

MVC8CRUD Home Privacy

Category (test5) updated successfully.

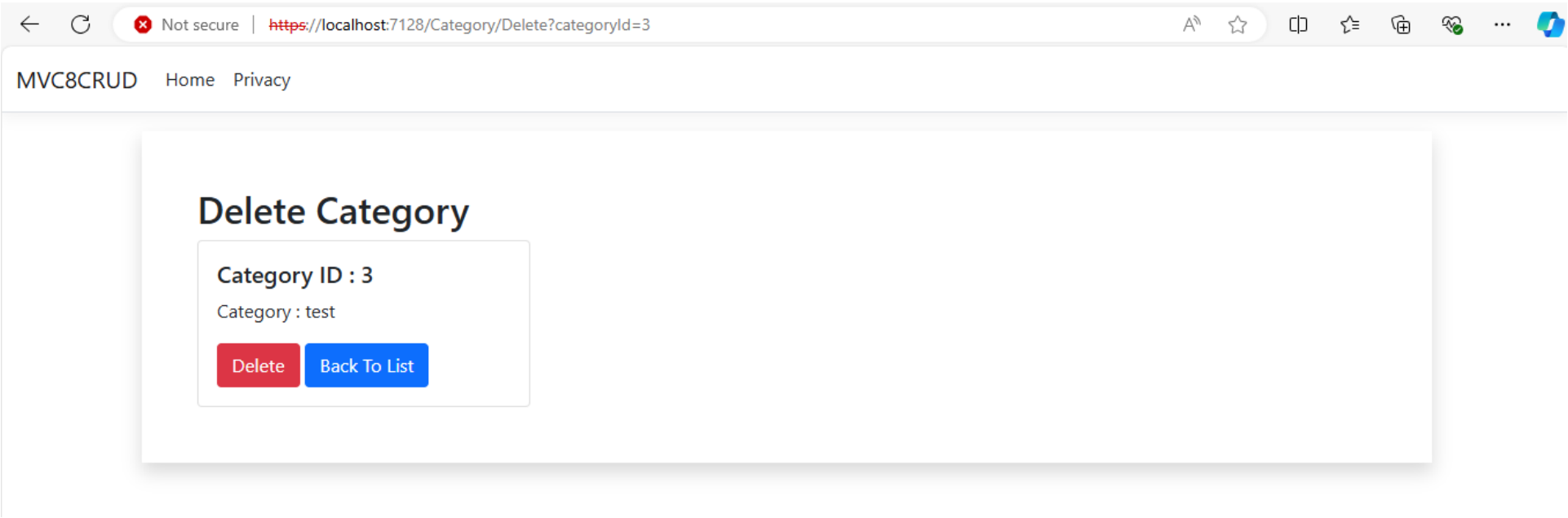
Category List

Add Category

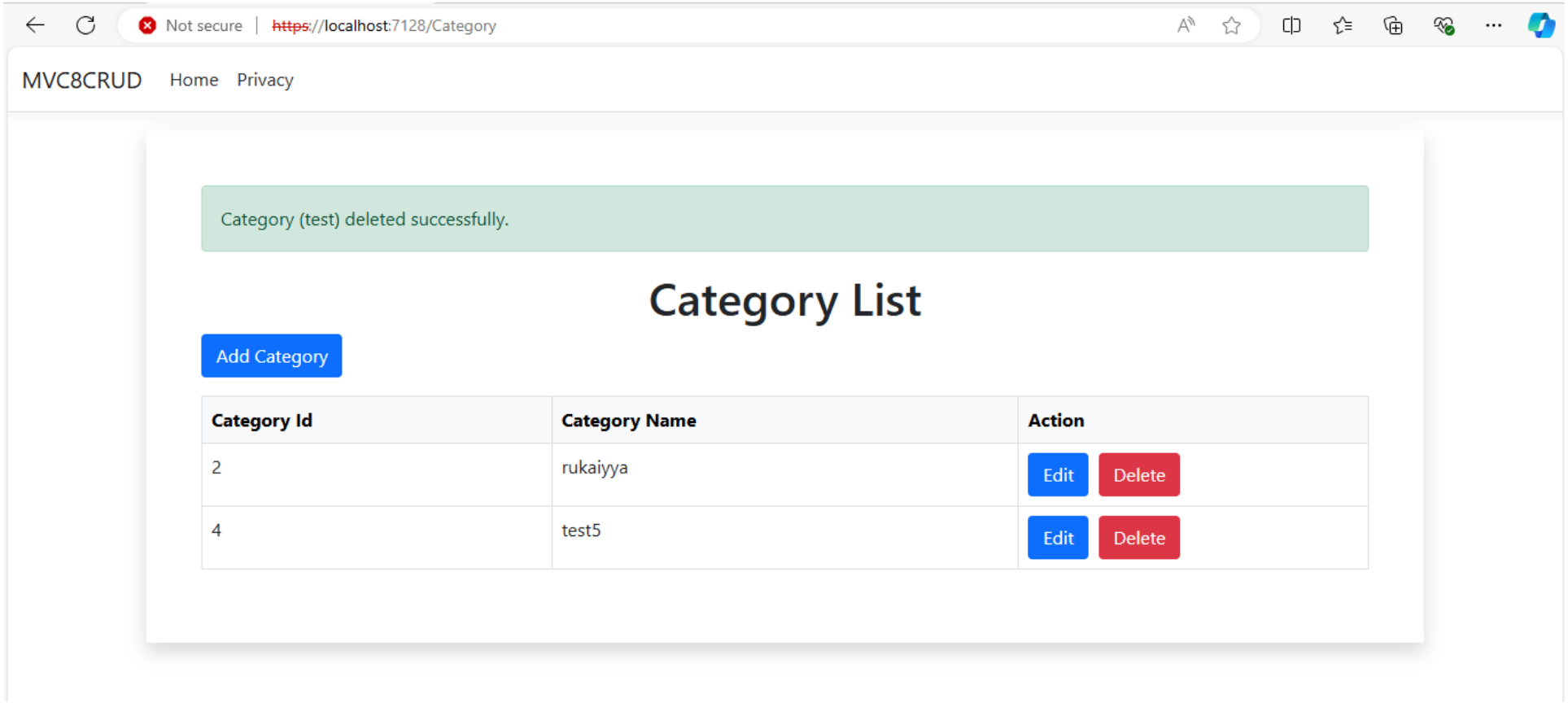
Category Id	Category Name	Action
2	rukaiyya	Edit Delete
3	test	Edit Delete
4	test5	Edit Delete

4. Delete Operation:-

Deleting Category with ID 3



Category with ID 3 (test) deleted successfully



Backend connectivity

```
C:\Users\User\source\repos\Rukaiyya\MVC8CRUD\MVC8CRUD\bin\Debug\net8.0\MVC8CRUD.exe
      Executed DbCommand (17ms) [Parameters=[@p0='?' (DbType = Int32), @p1='?' (Size = 4000), @p2='?' (Size = 4000), @p3
='?' (Size = 4000), @p4='?' (Size = 4000), @p5='?' (DbType = Int32)], CommandType='Text', CommandTimeout='30']
      SET IMPLICIT_TRANSACTIONS OFF;
      SET NOCOUNT ON;
      INSERT INTO [Products] ([CategoryId], [Color], [Description], [Image], [Name], [Price])
      OUTPUT INSERTED.[Id]
      VALUES (@p0, @p1, @p2, @p3, @p4, @p5);
info: Microsoft.EntityFrameworkCore.Database.Command[20101]
      Executed DbCommand (1ms) [Parameters=[], CommandType='Text', CommandTimeout='30']
      SELECT [p].[Id], [p].[CategoryId], [p].[Color], [p].[Description], [p].[Image], [p].[Name], [p].[Price]
      FROM [Products] AS [p]
info: Microsoft.EntityFrameworkCore.Database.Command[20101]
      Executed DbCommand (2ms) [Parameters=[@__item_CategoryId_0='?' (DbType = Int32)], CommandType='Text', CommandTimeo
ut='30']
      SELECT TOP(1) [c].[CategoryName]
      FROM [Categories] AS [c]
      WHERE [c].[CategoryId] = @__item_CategoryId_0
```

Another ASP.net MVC web application is created for products, Reference attached in the ZIP archive File name is Product.

Part 2: Back-End Development (APIs and Database Interaction):

Web API controller

```
using Microsoft.EntityFrameworkCore;
using System.Collections.Generic;
using System.Threading.Tasks;

namespace WebAPIMVC8CRUD.Controllers
{
    [Route("api/[controller]")]
    [ApiController]
    1 reference
    public class CategoryController : ControllerBase
    {

        private readonly Mvc8crudDbContext _context;

        0 references
        public CategoryController(Mvc8crudDbContext context)
        {
            _context = context;
        }

        [HttpGet]
        0 references
        public async Task<IEnumerable<Category>> Get()
        {
            return await _context.Categories.ToListAsync();
        }

        [HttpGet("{CategoryId}")]
        0 references
        public async Task<IActionResult> Get(int CategoryId)
        {
            if (CategoryId < 1)
                return BadRequest();
```

WEB API Controller-Asynchronous-GET-PUT-POST method code

```
23      public async Task<IEnumerable<Category>> Get()
24      {
25          return await _context.Categories.ToListAsync();
26      }
27
28      [HttpGet("{CategoryId}")]
        0 references
29      public async Task<IActionResult> Get(int CategoryId)
30      {
31          if (CategoryId < 1)
32              return BadRequest();
33          var Category = await _context.Categories.FirstOrDefaultAsync(m => m.CategoryId == CategoryId);
34          if (Category == null)
35              return NotFound();
36          return Ok(Category);
37
38      }
39
40      [HttpPost]
        0 references
41      public async Task<IActionResult> Post(Category category)
42      {
43          _context.Add(category);
44          await _context.SaveChangesAsync();
45          return Ok();
46      }
47
48      [HttpPut]
        0 references
49      public async Task<IActionResult> Put(Category categoryData)
50      {
51          if (categoryData == null || categoryData.CategoryId == 0)
```

WEB API Controller-Asynchronous-DELETE method code

```
            if (categoryData == null || categoryData.CategoryId == 0)
                return BadRequest();

            var category = await _context.Categories.FindAsync(categoryData.CategoryId);
            if (category == null)
                return NotFound();
            category.CategoryName = categoryData.CategoryName;
            //category.Description = productData.Description;
            //product.Price = productData.Price;
            await _context.SaveChangesAsync();
            return Ok();
        }

        [HttpDelete("{CategoryId}")]
        0 references
        public async Task<IActionResult> Delete(int CategoryId)
        {
            if (CategoryId < 1)
                return BadRequest();
            var Category = await _context.Categories.FindAsync(CategoryId);
            if (Category == null)
                return NotFound();
            _context.Categories.Remove(Category);
            await _context.SaveChangesAsync();
            return Ok();

        }
    }
}
```

CRUD operation in web API

GET method output

localhost:5126/swagger/index.html

GET /api/Category/{CategoryId}

Parameters

Cancel

Name	Description
CategoryId * required integer($int32) (path)	4

Execute | Clear

Responses

Curl

```
curl -X 'GET' \
  'http://localhost:5126/api/Category/4' \
  -H 'accept: */*'
```

Request URL

```
http://localhost:5126/api/Category/4
```

Server response

Code	Details
200	Response body

```
{
  "categoryId": 4,
  "categoryName": "EARPODS"
}
```

Download

Response headers

```
content-type: application/json; charset=utf-8
date: Sat,17 Aug 2024 21:56:31 GMT
server: Kestrel
transfer-encoding: chunked
```

GET Method response

Responses

Curl

```
curl -X 'GET' \
  'http://localhost:5126/api/Category' \
  -H 'accept: application/json'
```

Request URL

```
http://localhost:5126/api/Category
```

Server response

Code	Details
200	Response body

```
[
  {
    "categoryId": 2,
    "categoryName": "rukaiyya"
  },
  {
    "categoryId": 4,
    "categoryName": "headset"
  }
]
```

Download

Response headers

```
content-type: application/json; charset=utf-8
date: Sat,17 Aug 2024 21:40:47 GMT
server: Kestrel
transfer-encoding: chunked
```

Responses

Code	Description	Links

GET method data in Database.

SQLQuery3.sql - DESKTOP-6JBC82E.MVC8CrudDb (DESKTOP-6JBC82E\User (53)) - Microsoft SQL Server Management Studio

File Edit View Project Tools Window Help

New Query | MVC8CrudDb | Execute

Object Explorer

Connect

- DESKTOP-6JBC82E (SQL Server 15.0.2000.5 - DESKTOP-6J
  - Databases
    - System Databases
    - Database Snapshots
    - MVC8CrudDb
      - Database Diagrams
      - Tables
        - System Tables
        - FileTables
        - External Tables
        - Graph Tables
        - dbo.__EFMigrationsHistory
        - dbo.Categories
        - dbo.Products
      - Views
      - External Resources
      - Synonyms
      - Programmability
      - Service Broker
      - Storage
      - Security
  - Security
  - Server Objects
  - Replication
  - PolyBase
  - Always On High Availability
  - Management
  - Integration Services Catalogs
  - SQL Server Agent (Agent XPs disabled)
  - XEvent Profiler

SQLQuery3.sql - DE...-6JBC82E\User (53)) | SQLQuery2.sql - DE...-6JBC82E\User (60)) | SQLQuery1.sql - DE...-6JBC82E\User (54))

```
SELECT TOP (1000) [CategoryId]
      ,[CategoryName]
  FROM [MVC8CrudDb].[dbo].[Categories]
```

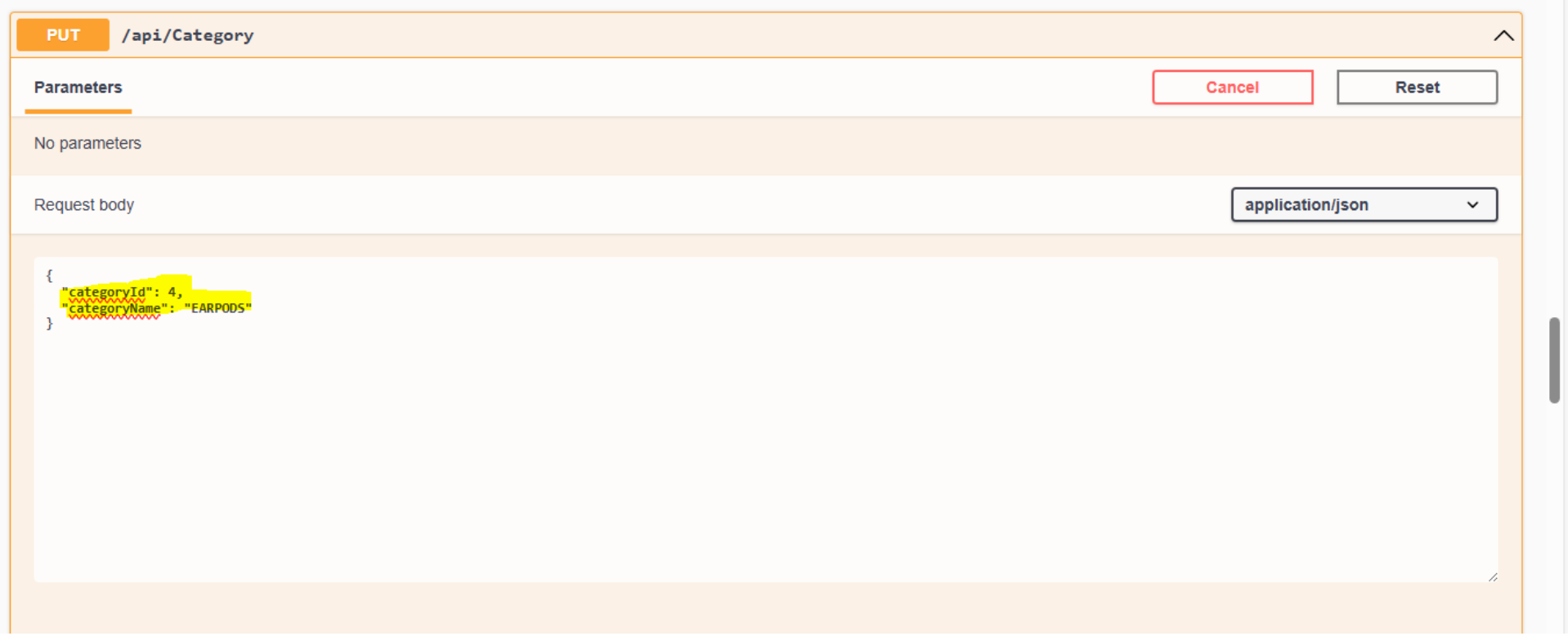
100 %

Results | Messages

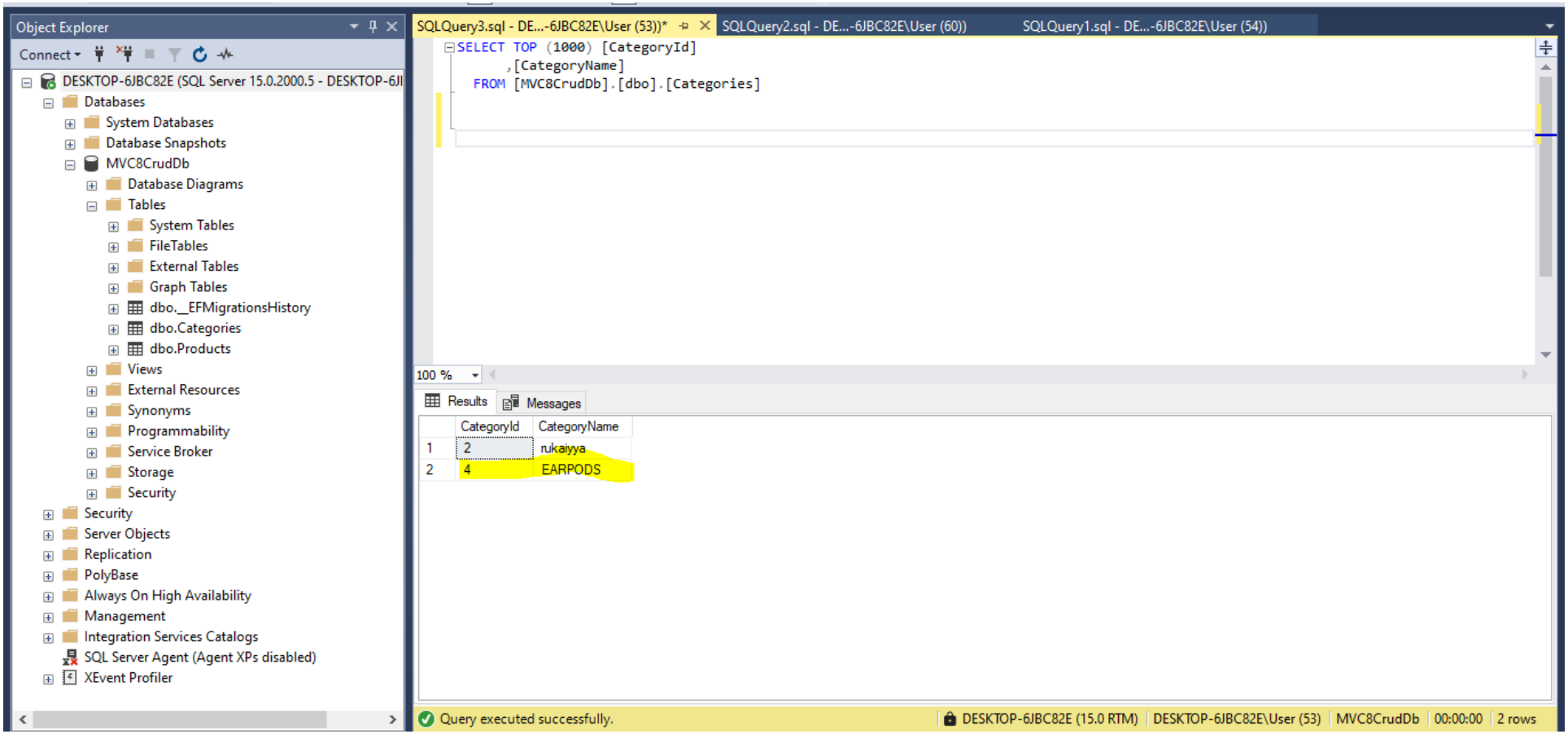
	CategoryId	CategoryName
1	2	rukaiyya
2	4	headset

Query executed successfully. | DESKTOP-6JBC82E (15.0 RTM) | DESKTOP-6JBC82E\User (53) | MVC8CrudDb | 00:00:00 | 2 rows

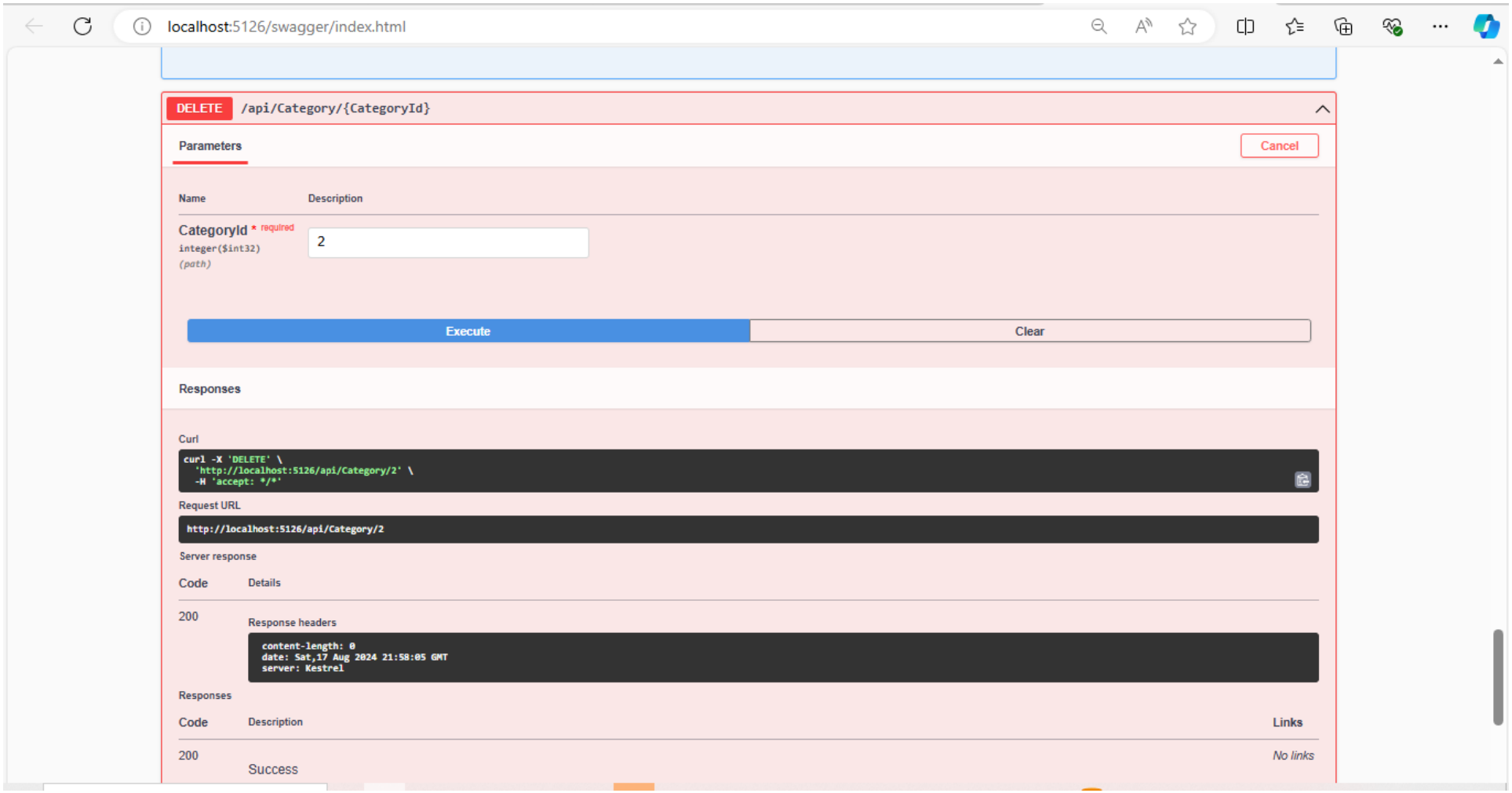
PUT method output



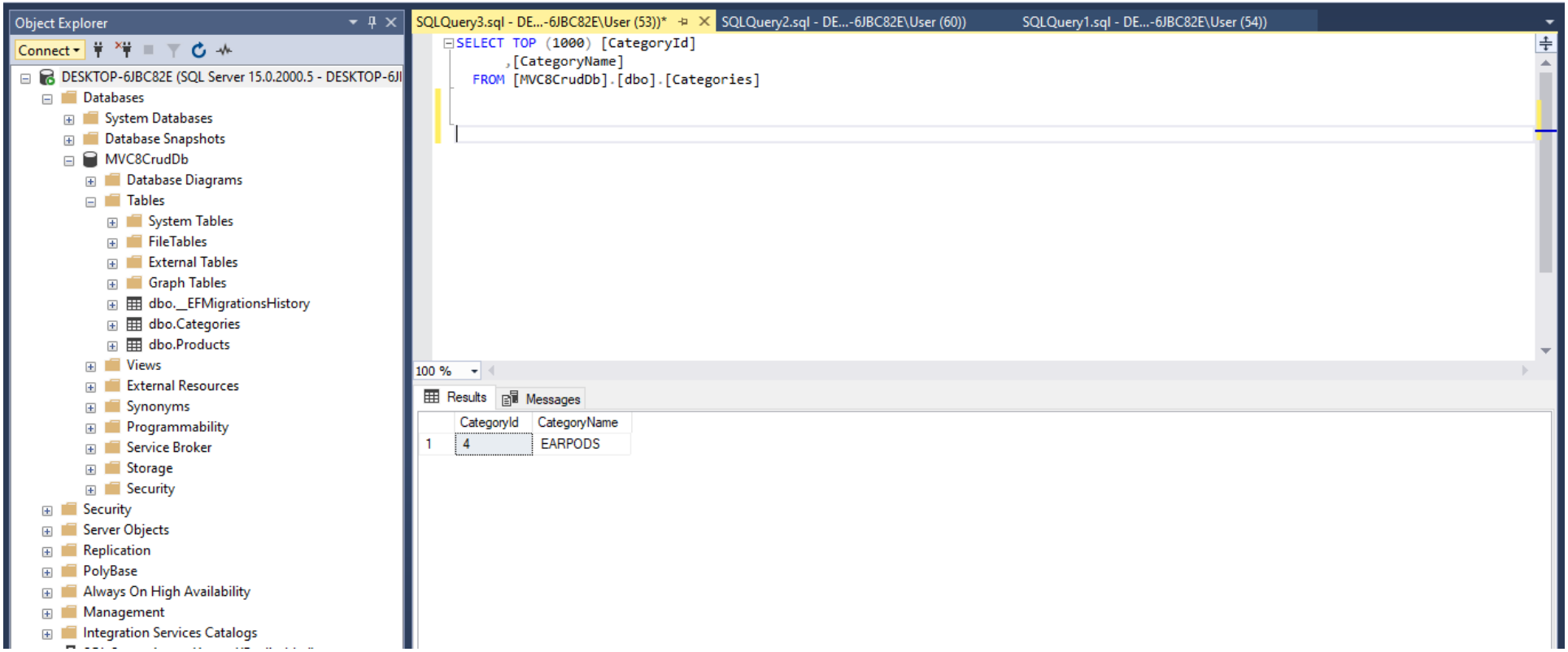
PUT method changes in Database



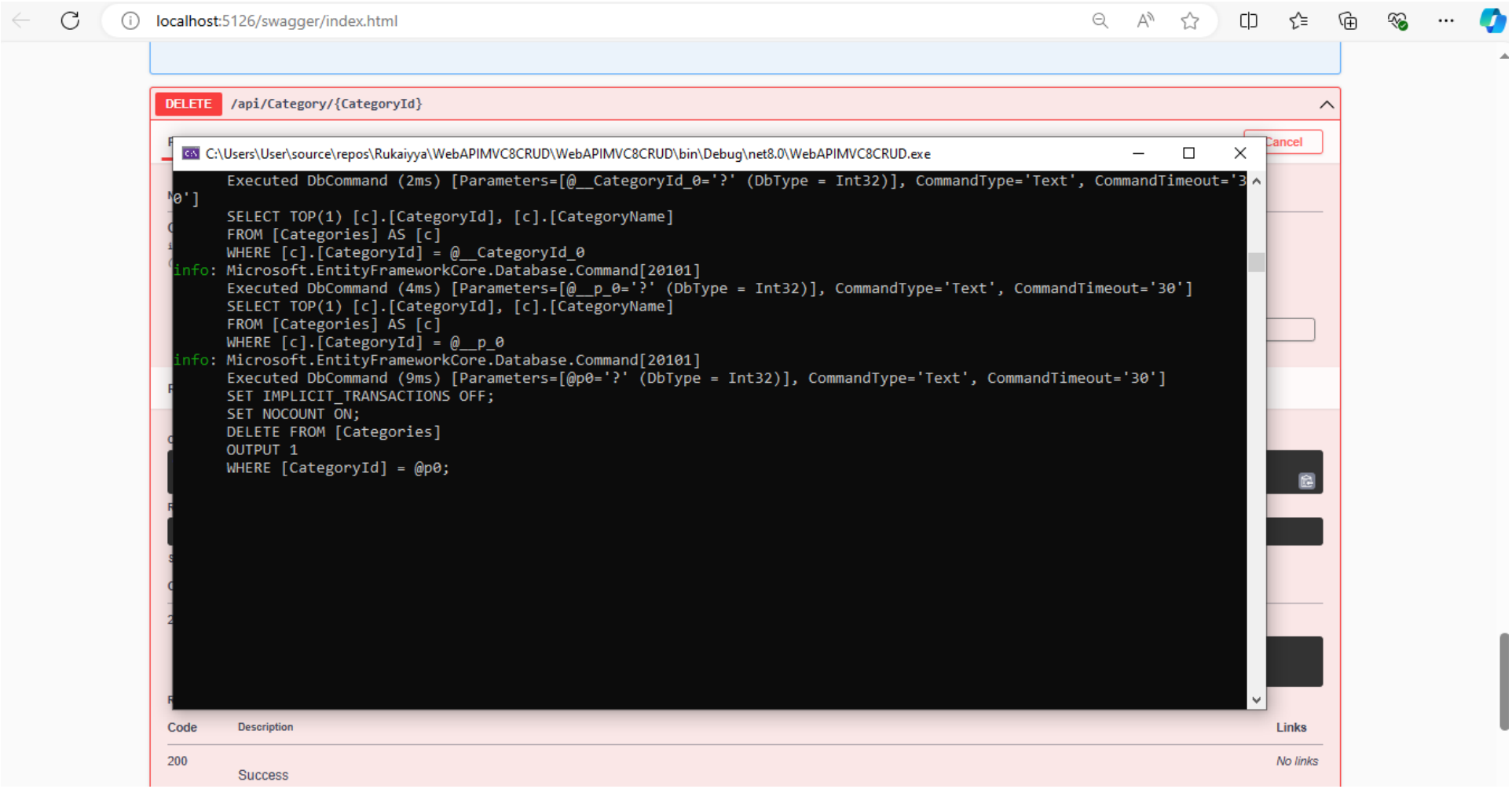
DELETE method output



DELETE method changes in database



WEB API Database Interaction.



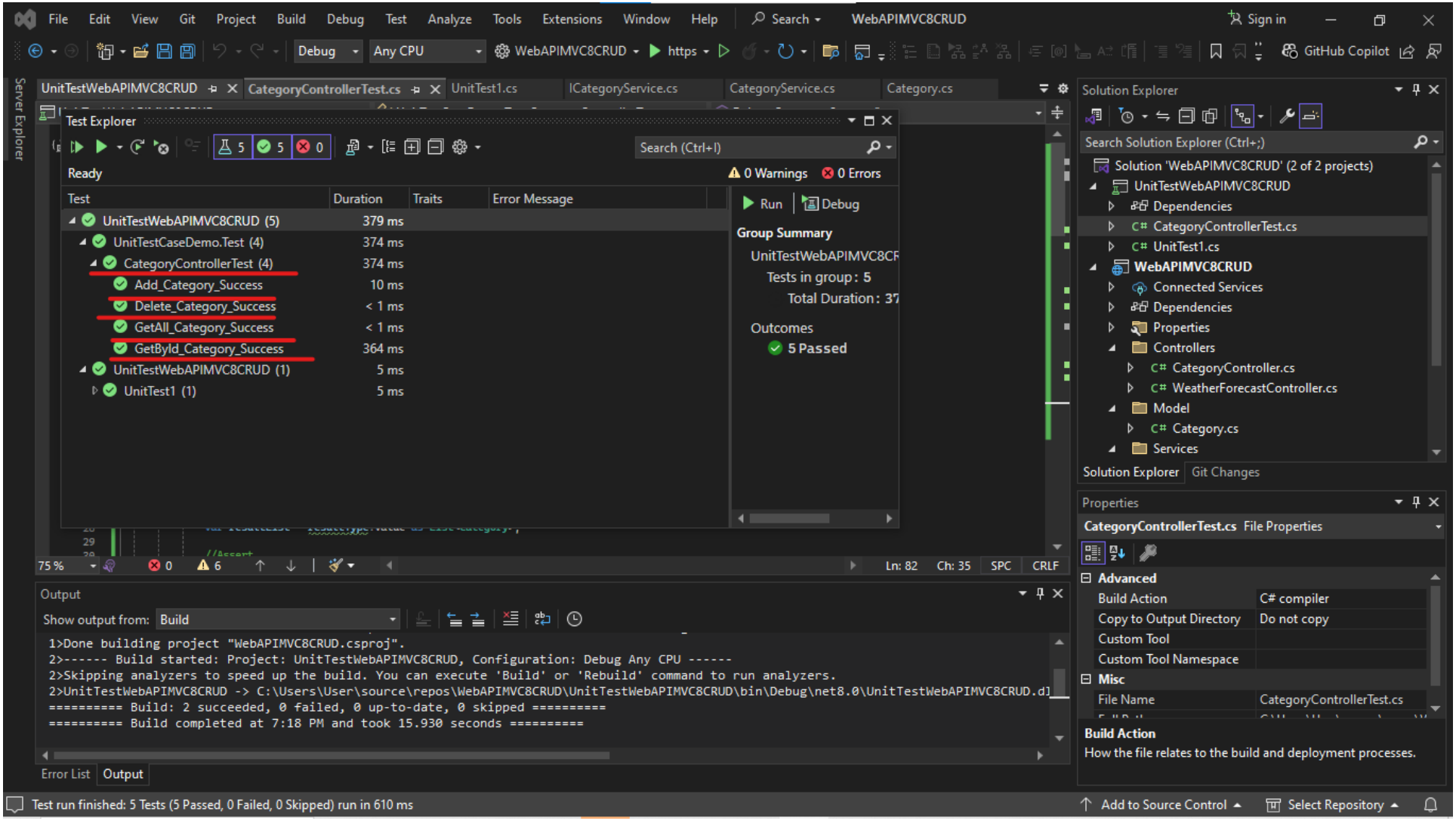
Part 3: Integration and Testing

Controller class with set of integration test

```
namespace UnitTestCaseDemo.Test
{
    1 reference
    public class CategoryControllerTest
    {
        CategoryController _controller;
        ICategoryService _service;

        0 references
        public CategoryControllerTest()
        {
            _service = new CategoryService();
            _controller = new CategoryController(_service);
        }

        [Fact]
        | 0 references
        public void GetAll_Category_Success()
        {
            //Arrange

            //Act
            var result = _controller.Get();
            var resultType = result as OkObjectResult;
            var resultList = resultType.Value as List<Category>;

            //Assert
            Assert.NotNull(result);
            Assert.IsType<List<Category>>(resultType.Value);
            Assert.Equal(3, resultList.Count);
        }

        [Fact]
```

```
1>Done building project "WebAPIMVC8CRUD.csproj".
2>------ Build started: Project: UnitTestWebAPIMVC8CRUD, Configuration: Debug Any CPU ------
2>Skipping analyzers to speed up the build. You can execute 'Build' or 'Rebuild' command to run analyzers.
2>UnitTestWebAPIMVC8CRUD -> C:\Users\User\source\repos\WebAPIMVC8CRUD\UnitTestWebAPIMVC8CRUD\bin\Debug\net8.0\UnitTestWebAPIMVC8CRUD.dll
========== Build: 2 succeeded, 0 failed, 0 up-to-date, 0 skipped ==========
========== Build completed at 7:18 PM and took 15.930 seconds ==========
```

Xunit test result.



Here are the details for running the solutions locally:

Note: Please update **appsettings.json** file with the database string connection of your local SQL server

MVC8CRUD: This is an ASP.NET MVC web application solution. Open the `.sln` file in the ZIP archive to run it.

WEBAPIMVC8CRUD: This is an ASP.NET Web API solution. Open the `.sln` file in the ZIP archive to run it.

UnitTestWebAPIMVC8CRUD: This uses the XUnit testing framework. Open the `.sln` file in the ZIP archive to run it.