Create a Product model with properties like ProductId, Name, QuantityInStock, and Price. Implement GET /products to view all products, POST /products to add a new product, PUT /products/{id} to update the stock for a product, and DELETE /products/{id} to remove a product. Implement stock validation to prevent negative stock values.

2. Folder Structure

Your project folder structure will look like this:

```
markdown
Copy code
/ProductManagement
    /Controllers
    /Models
    /Services
    /Program.cs
```

---

3. Step-by-Step Implementation

Step 1: Create the Product Model

1. **Create the `Models` folder:**
    - Right-click the project > **Add** > **New Folder**.
    - Name it `Models`.
2. **Add `Product.cs`:**
    - Right-click the `Models` folder > **Add** > **Class**.
    - Name it `Product.cs`.

   Product.cs:

```
csharp
Copy code
namespace ProductManagement.Models
{
    public class Product
    {
        public int ProductId { get; set; }
        public string Name { get; set; }
        public int QuantityInStock { get; set; }
        public decimal Price { get; set; }
    }
}
```

Step 2: Create the Product Service

1. **Create the `Services` folder:**
    - Right-click the project > **Add** > **New Folder**.
    - Name it `Services`.

2. **Add `ProductService.cs`**:
    - Right-click the `Services` folder > **Add** > **Class**.
    - Name it `ProductService.cs`.

ProductService.cs:

```
csharp
Copy code
using ProductManagement.Models;
using System.Collections.Generic;
using System.Linq;

namespace ProductManagement.Services
{
    public class ProductService
    {
        private readonly List<Product> _products = new();

        public IEnumerable<Product> GetAllProducts()
        {
            return _products;
        }

        public Product GetProductById(int id)
        {
            return _products.FirstOrDefault(p => p.ProductId == id);
        }

        public void AddProduct(Product product)
        {
            _products.Add(product);
        }

        public bool UpdateProductStock(int id, int newStock)
        {
            var product = GetProductById(id);
            if (product == null || newStock < 0) return false;

            product.QuantityInStock = newStock;
            return true;
        }

        public bool DeleteProduct(int id)
        {
            var product = GetProductById(id);
            if (product == null) return false;

            _products.Remove(product);
            return true;
        }
    }
}
```

Step 3: Create the Product Controller

1. **Create the `Controllers` folder**:
    - Right-click the project > **Add** > **New Folder**.

- Name it `Controllers`.

2. **Add `ProductsController.cs`**:
   - Right-click the `Controllers` folder > **Add** > **Controller**.
   - Choose **API Controller - Empty**.
   - Name it `ProductsController`.

ProductsController.cs:

```
csharp
Copy code
using Microsoft.AspNetCore.Mvc;
using ProductManagement.Models;
using ProductManagement.Services;

[ApiController]
[Route("api/[controller]")]
public class ProductsController : ControllerBase
{
    private readonly ProductService _productService;

    public ProductsController(ProductService productService)
    {
        _productService = productService;
    }

    [HttpGet]
    public IActionResult GetAllProducts()
    {
        var products = _productService.GetAllProducts();
        return Ok(products);
    }

    [HttpPost]
    public IActionResult AddProduct([FromBody] Product product)
    {
        if (product.QuantityInStock < 0)
        {
            return BadRequest("Stock value cannot be negative.");
        }

        _productService.AddProduct(product);
        return Ok("Product added successfully.");
    }

    [HttpPut("{id}")]
    public IActionResult UpdateProductStock(int id, [FromBody] int
newStock)
    {
        if (newStock < 0)
        {
            return BadRequest("Stock value cannot be negative.");
        }

        var result = _productService.UpdateProductStock(id,
newStock);
        if (!result)
        {
            return NotFound($"Product with ID {id} not found.");
        }
```

```
            return Ok("Stock updated successfully.");
        }

        [HttpDelete("{id}")]
        public IActionResult DeleteProduct(int id)
        {
            var result = _productService.DeleteProduct(id);
            if (!result)
            {
                return NotFound($"Product with ID {id} not found.");
            }

            return Ok("Product deleted successfully.");
        }
    }
```

Step 4: Register the Service

1. Open **`Program.cs`**.
2. Register the `ProductService` in the DI container.

   Program.cs:

   ```
   csharp
   Copy code
   var builder = WebApplication.CreateBuilder(args);

   // Add services to the container.
   builder.Services.AddControllers();
   builder.Services.AddSingleton<ProductService>(); // Register
   ProductService

   var app = builder.Build();

   // Configure the HTTP request pipeline.
   app.UseHttpsRedirection();
   app.UseAuthorization();
   app.MapControllers();

   app.Run();
   ```

---

4. Test the Application

Step 1: Run the Application

1. Press **F5** or **Ctrl + F5** to run the application.
2. Use **Swagger**, **Postman**, or any REST client to test the following endpoints.

---

5. API Endpoints

1. GET /api/products

- **Description**: Retrieve all products.
- **Response**:

```
json
Copy code
[
    {
        "productId": 1,
        "name": "Laptop",
        "quantityInStock": 10,
        "price": 1200.99
    },
    {
        "productId": 2,
        "name": "Mouse",
        "quantityInStock": 50,
        "price": 25.50
    }
]
```

2. POST /api/products

- **Description**: Add a new product.
- **Request Body**:

```
json
Copy code
{
    "productId": 1,
    "name": "Laptop",
    "quantityInStock": 10,
    "price": 1200.99
}
```

- **Response**:

```
json
Copy code
"Product added successfully."
```

3. PUT /api/products/{id}

- **Description**: Update the stock of a product.
- **Request URL**: `/api/products/1`
- **Request Body**:

```
json
Copy code
20
```

- **Response**:

```
json
Copy code
"Stock updated successfully."
```

4. DELETE /api/products/{id}

- **Description**: Delete a product.
- **Request URL**: `/api/products/1`
- **Response**:

  ```
  json
  Copy code
  "Product deleted successfully."
  ```

Create ASP.Net MVC Web application for Educational Learning with Master Page and minimum 4 Pages.

2. Folder Structure

The basic folder structure will look like this:

```
vbnet
Copy code
/EducationalLearning
    /Controllers
    /Models
    /Views
        /Shared
        /Home
        /Courses
        /Students
        /Teachers
    /Content
    /Scripts
    /App_Start
    /Global.asax
```

---

3. Step-by-Step Implementation

Step 1: Create the Master Page (Layout)

1. **Right-click the `Views/Shared` folder** > **Add** > **New Item**.
   - Select **MVC Layout Page (Razor)**.
   - Name it `Layout.cshtml`.
2. **Edit `Layout.cshtml`**:

   ```
   html
   Copy code
   <!DOCTYPE html>
   <html lang="en">
   <head>
       <meta charset="utf-8" />
       <meta name="viewport" content="width=device-width, initial-
   scale=1.0" />
   ```

```
    <title>@ViewData["Title"] - Educational Learning</title>
    <link href="~/Content/Site.css" rel="stylesheet" />
</head>
<body>
    <header>
        <h1>Educational Learning Platform</h1>
        <nav>
            <ul>
                <li>@Html.ActionLink("Home", "Index", "Home")</li>
                <li>@Html.ActionLink("Courses", "Index",
"Courses")</li>
                <li>@Html.ActionLink("Students", "Index",
"Students")</li>
                <li>@Html.ActionLink("Teachers", "Index",
"Teachers")</li>
            </ul>
        </nav>
    </header>

    <main>
        @RenderBody()
    </main>

    <footer>
        <p>&copy; 2024 Educational Learning</p>
    </footer>
</body>
</html>
```

Step 2: Create Controllers

1. HomeController

1. **Right-click the `Controllers` folder** > **Add** > **Controller**.
    - Choose **MVC 5 Controller - Empty**.
    - Name it `HomeController`.
2. **Edit `HomeController.cs`:**

```
csharp
Copy code
using System.Web.Mvc;

namespace EducationalLearning.Controllers
{
    public class HomeController : Controller
    {
        public ActionResult Index()
        {
            return View();
        }

        public ActionResult About()
        {
            ViewBag.Message = "This is the Educational Learning
Platform.";
            return View();
        }
        public ActionResult Contact()
        {
            ViewBag.Message = "Contact us for more information.";
            return View();
        }
    }
}
```

2. CoursesController

1. **Right-click the `Controllers` folder** > **Add** > **Controller**.
    - Choose **MVC 5 Controller - Empty**.
    - Name it `CoursesController`.
2. **Edit `CoursesController.cs`**:

```
csharp
Copy code
using System.Web.Mvc;

namespace EducationalLearning.Controllers
{
    public class CoursesController : Controller
    {
        public ActionResult Index()
        {
            return View();
        }

        public ActionResult Details(int id)
        {
            ViewBag.CourseId = id;
            return View();
        }
    }
}
```

3. StudentsController

1. **Right-click the `Controllers` folder** > **Add** > **Controller**.
    - Choose **MVC 5 Controller - Empty**.
    - Name it `StudentsController`.
2. **Edit `StudentsController.cs`**:

```
csharp
Copy code
using System.Web.Mvc;

namespace EducationalLearning.Controllers
{
    public class StudentsController : Controller
    {
        public ActionResult Index()
        {
            return View();
        }

        public ActionResult Details(int id)
        {
            ViewBag.StudentId = id;
            return View();
        }
    }
}
```

4. TeachersController

1. **Right-click the `Controllers` folder** > **Add** > **Controller**.
   - Choose **MVC 5 Controller - Empty**.
   - Name it `TeachersController`.
2. **Edit `TeachersController.cs`**:

```
csharp
Copy code
using System.Web.Mvc;

namespace EducationalLearning.Controllers
{
    public class TeachersController : Controller
    {
        public ActionResult Index()
        {
            return View();
        }

        public ActionResult Details(int id)
        {
            ViewBag.TeacherId = id;
            return View();
        }
    }
}
```

Step 3: Create Views

1. Home Views

1. **Right-click the `Views/Home` folder** > **Add** > **View**.
   - Name it `Index.cshtml`.
2. **Edit `Index.cshtml`**:

```
html
Copy code
@{
    ViewData["Title"] = "Home";
}

<h2>Welcome to the Educational Learning Platform</h2>
<p>Your source for online courses, student resources, and teacher
interactions.</p>
```

3. **Add `About.cshtml`**:
   - Right-click the `Views/Home` folder > **Add** > **View**.
   - Name it `About.cshtml`.

About.cshtml:

```
html
Copy code
@{
    ViewData["Title"] = "About";
}

<h2>About</h2>
<p>@ViewBag.Message</p>
```

4. **Add `Contact.cshtml`**:
    - Right-click the `Views/Home` folder > **Add** > **View**.
    - Name it `Contact.cshtml`.

Contact.cshtml:

```
html
Copy code
@{
    ViewData["Title"] = "Contact";
}

<h2>Contact</h2>
<p>@ViewBag.Message</p>
```

2. Courses Views

1. **Right-click the `Views/Courses` folder** > **Add** > **View**.
    - Name it `Index.cshtml`.

Index.cshtml:

```
html
Copy code
@{
    ViewData["Title"] = "Courses";
}

<h2>Courses</h2>
<p>Explore our range of courses to help you grow your skills.</p>
```

2. **Add `Details.cshtml`** for `Courses`:
    - Right-click the `Views/Courses` folder > **Add** > **View**.
    - Name it `Details.cshtml`.

Details.cshtml:

```
html
Copy code
@{
    ViewData["Title"] = "Course Details";
}

<h2>Course Details</h2>
<p>Details for course with ID: @ViewBag.CourseId</p>
```

3. Students Views

1. **Right-click the `Views/Students` folder** > **Add** > **View**.
   - Name it `Index.cshtml`.

   Index.cshtml:

   ```
   html
   Copy code
   @{
       ViewData["Title"] = "Students";
   }

   <h2>Students</h2>
   <p>Here is a list of all students enrolled in the platform.</p>
   ```

2. **Add `Details.cshtml`** for `Students`:
   - Right-click the `Views/Students` folder > **Add** > **View**.
   - Name it `Details.cshtml`.

   Details.cshtml:

   ```
   html
   Copy code
   @{
       ViewData["Title"] = "Student Details";
   }

   <h2>Student Details</h2>
   <p>Details for student with ID: @ViewBag.StudentId</p>
   ```

4. Teachers Views

1. **Right-click the `Views/Teachers` folder** > **Add** > **View**.
   - Name it `Index.cshtml`.

   Index.cshtml:

   ```
   html
   Copy code
   @{
       ViewData["Title"] = "Teachers";
   }

   <h2>Teachers</h2>
   <p>Explore the list of available teachers.</p>
   ```

2. **Add `Details.cshtml`** for `Teachers`:
   - Right-click the `Views/Teachers` folder > **Add** > **View**.
   - Name it `Details.cshtml`.

   Details.cshtml:

   ```
   html
   Copy code
   @{
   ```

```
    ViewData["Title"] = "Teacher Details";
}

<h2>Teacher Details</h2>
<p>Details for teacher with ID: @ViewBag.TeacherId</p>
```

Step 4: Configure Routing

1. Open **`RouteConfig.cs`** in the **App_Start** folder:

```
csharp
Copy code
public class RouteConfig
{
    public static void RegisterRoutes(RouteCollection routes)
    {
        routes.IgnoreRoute("{resource}.axd/{*pathInfo}");

        routes.MapRoute(
            name: "Default",
            url: "{controller}/{action}/{id}",
            defaults: new { controller = "Home", action = "Index", id
= UrlParameter.Optional }
        );
    }
}
```
