

## **CST-350 Activity 5 CRUD with n-Layer Design**

Alex M. Frear

College of Science, Engineering, and Technology, Grand Canyon University

Course Number: CST-350

Professor Brandon Bass

11/24/2024

**GitHub Link:**

[https://github.com/amfrear/cst350/tree/main/Activity\\_5](https://github.com/amfrear/cst350/tree/main/Activity_5)

## Contents

GitHub Link .....	1
Overview .....	3
Setting up the Database and Application .....	3
Creating the Products Database .....	3
Products Table Schema .....	4
ProductModel Class Implementation .....	5
Initial Application Default Page .....	6
Implementing Product Creation .....	7
Homepage Navigation View .....	7
Empty Create Product Form .....	8
Create Product Form Filled with Data .....	9
Updated Products Table After Submission .....	10
Initial View of All Products Page .....	11
Adding Dynamic Tax Calculation .....	12
Tax Calculation in Create Product Form .....	12
Tax Display on All Products Page .....	13
Selecting an Image for a Product .....	14
Products Displayed with Associated Images .....	15
Products Displayed with Associated Images in Grid View .....	16
Editing and Deleting Products .....	17
Edit Product Page with Pre-Filled Data .....	17
Delete Product Confirmation Dialog .....	18
Product Successfully Deleted from Table .....	19
Searching for Products .....	20
Product Search Input Page .....	20
Search Results Display .....	21
Viewing Product Details .....	22
Detailed View of a Single Product .....	22
Centralized Configuration with appsettings.json .....	23
Configuration Settings in appsettings.json .....	23
Summary of Key Concepts .....	24

## Overview

This activity focuses on building a full-stack CRUD application using the n-layer design approach with ASP.NET MVC. The application includes features such as product creation, updates, deletion, searching, and dynamic configuration using appsettings.json. Below are the detailed steps, implementation screenshots, and key learnings from this activity.

## Setting up the Database and Application

### Creating the Products Database

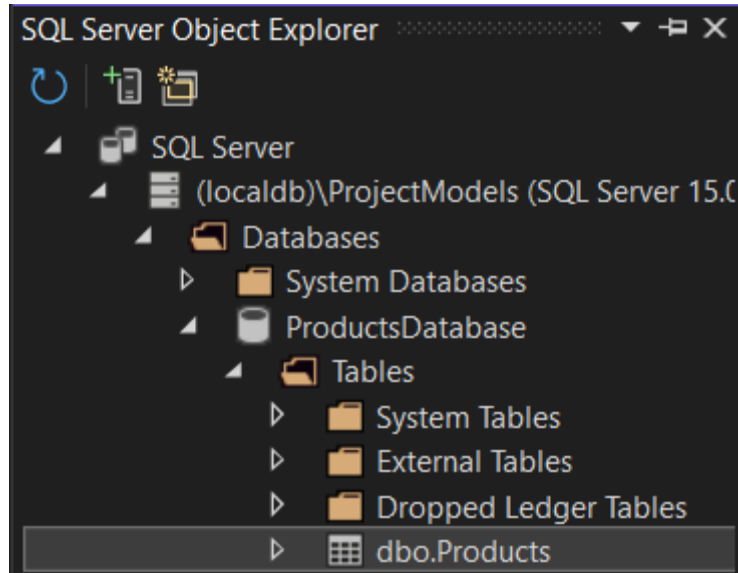


Figure 1 The database schema is set up to store product data, including fields for Name, Price, Description, CreatedAt, and ImageURL.

## Products Table Schema

The screenshot displays the Visual Studio IDE with the SQL Server Enterprise Designer open. The main window shows the 'dbo.Products [Design]' table schema. The table has the following columns:

Name	Data Type	Allow Nulls	Default
Id	int	<input checked="" type="checkbox"/>	
Name	nvarchar(50)	<input checked="" type="checkbox"/>	
Price	decimal(18,2)	<input checked="" type="checkbox"/>	
Description	nvarchar(100)	<input checked="" type="checkbox"/>	(NULL)
CreatedAt	datetime	<input checked="" type="checkbox"/>	
ImageUrl	nchar(200)	<input checked="" type="checkbox"/>	

On the right side of the design view, the 'Keys' section shows a primary key constraint for the 'Id' column, labeled '<unnamed> (Primary Key, Clustered: Id)'. Below this, 'Check Constraints (0)', 'Indexes (0)', 'Foreign Keys (0)', and 'Triggers (0)' are listed.

The bottom pane shows the T-SQL script for creating the table:

```

1 CREATE TABLE [dbo].[Products] (
2     [Id] INT IDENTITY (1, 1) NOT NULL,
3     [Name] NVARCHAR (50) NOT NULL,
4     [Price] DECIMAL (18, 2) NOT NULL,
5     [Description] NVARCHAR (100) DEFAULT (NULL) NULL,
6     [CreatedAt] DATETIME NULL,
7     [ImageUrl] NCHAR (200) NULL,
8     CONSTRAINT PK_Products PRIMARY KEY CLUSTERED ([Id] ASC)
9 )

```

The bottom status bar indicates 'Connection Ready' and 'Data Tools Operations'. The 'Data Tools Operations' pane shows a successful update for the 'ProductsDatabase' at 2:10:32 PM - 2:10:56 PM (0:00:23).

Figure 2 The table schema defines the structure of the Products table with appropriate data types and constraints.

## ProductModel Class Implementation

```
using System;

namespace ProductsApp.Models
{
    // This class is tied closely to the database schema and is used for data access.
    public class ProductModel
    {
        public int Id { get; set; }
        public string Name { get; set; }
        public decimal Price { get; set; }
        public string Description { get; set; }
        public DateTime CreatedAt { get; set; }
        public string ImageURL { get; set; }

        // Constructor with parameters
        public ProductModel(int id, string name, decimal price, string description, DateTime createdAt,
string imageURL)
        {
            Id = id;
            Name = name;
            Price = price;
            Description = description;
            CreatedAt = createdAt;
            ImageURL = imageURL;
        }

        // Parameterless constructor
        public ProductModel() { }

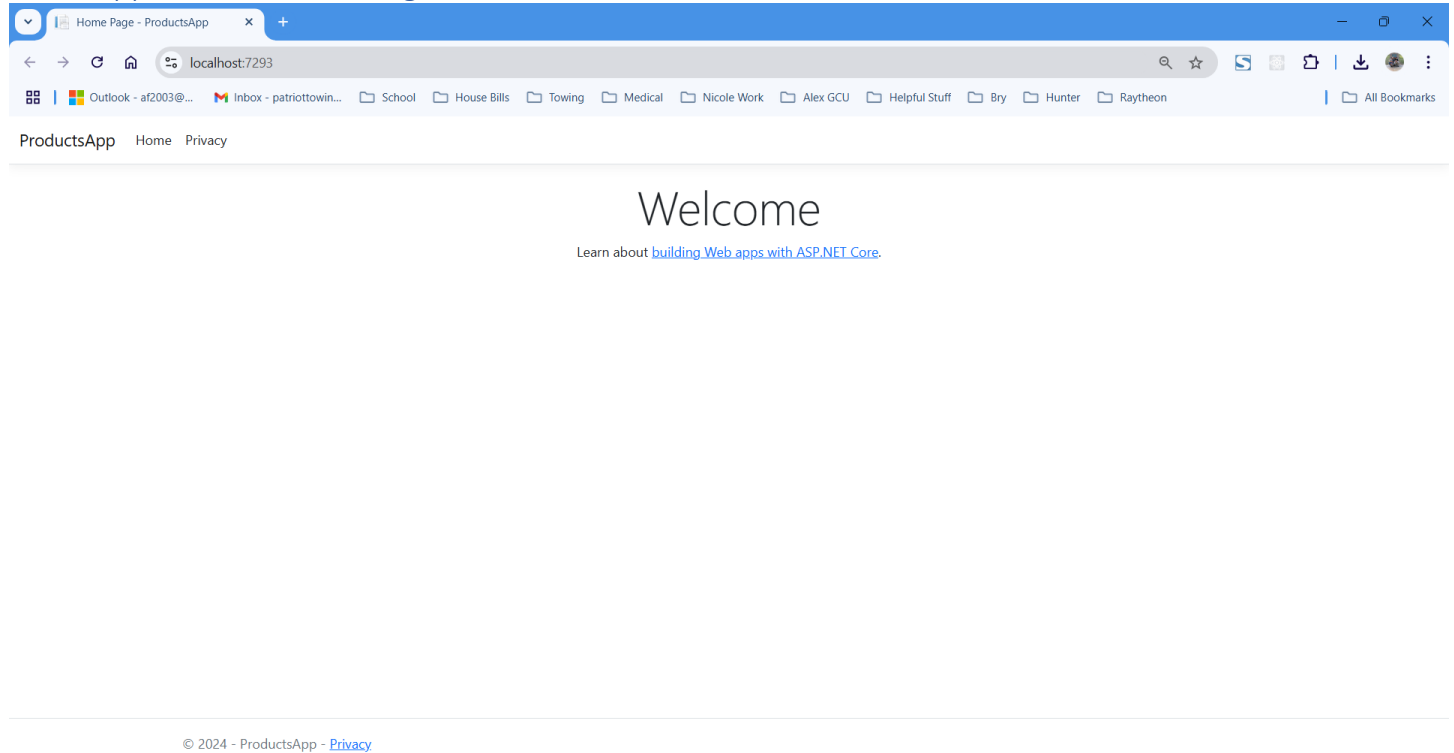
        // Override Equals method
        public override bool Equals(object obj)
        {
            return Equals(obj as ProductModel);
        }

        public bool Equals(ProductModel other)
        {
            return other != null && Id == other.Id;
        }

        // Override GetHashCode
        public override int GetHashCode()
        {
            return Id.GetHashCode();
        }
    }
}
```

Figure 3 The `ProductModel` class represents the data structure for products in the application.

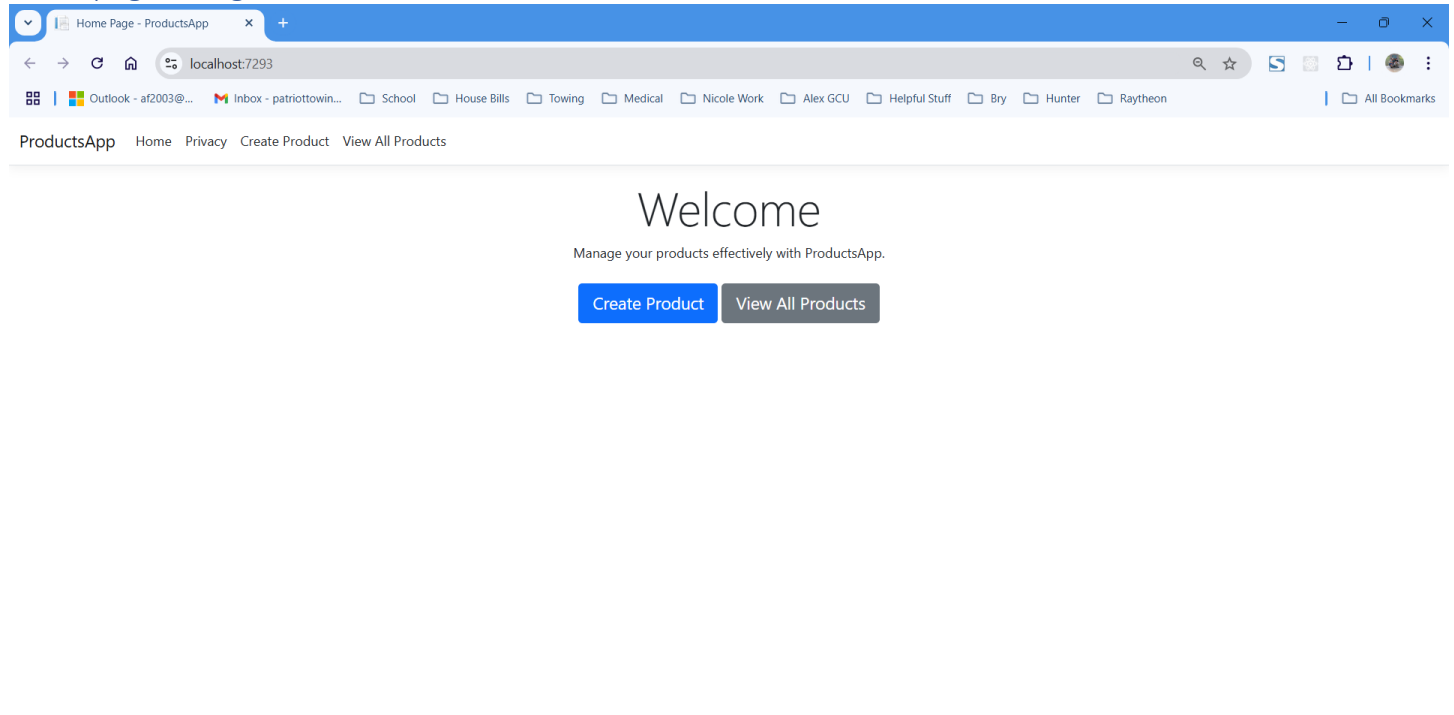
## Initial Application Default Page



*Figure 4 The default landing page of the application after initial setup.*

# Implementing Product Creation

## Homepage Navigation View



*Figure 5 The homepage displays navigation links to access product-related functionalities.*

## Empty Create Product Form

The screenshot shows a web browser window with the title 'Create Product - ProductsApp'. The address bar displays 'localhost:7293/Home/ShowCreateProductForm'. The browser's bookmark bar includes links to Outlook, Gmail, and various folders. The page's navigation bar contains 'ProductsApp', 'Home', 'Privacy', 'Create Product', and 'View All Products'. The main content area is titled 'Create Product' and contains a form with the following fields: 'Name' (text input), 'Price' (text input with '0.00' entered), 'Description' (text area), 'CreatedAt' (text input with '11/19/2024' entered), and 'ImageURL' (text input). Below the form is a blue 'Create' button and a blue link 'Back to List'. The footer of the page shows '© 2024 - ProductsApp - Privacy'.

Create Product

Name

Price

0.00

Description

CreatedAt

11/19/2024

ImageURL

Create

[Back to List](#)

© 2024 - ProductsApp - [Privacy](#)

Figure 6 An empty product creation form is presented to the user, ready for input.



## Create Product Form Filled with Data

The screenshot shows a web browser window with the address bar displaying `localhost:7293/Home/ShowCreateProductForm`. The browser's tab is titled 'Create Product - ProductsApp'. The page has a navigation bar with links: 'ProductsApp', 'Home', 'Privacy', 'Create Product', and 'View All Products'. The main content area is titled 'Create Product' and contains a form with the following fields:

- Name:** Sample Product
- Price:** 19.99
- Description:** A Simple Sample Product
- CreatedAt:** 11/19/2024
- Estimated Tax:** 0.00
- ImageURL:** `https://gidsha.com/assets/uploads/product2.png`

Below the form fields is a blue 'Create' button and a blue link labeled 'Back to List'.

At the bottom of the page, there is a footer with the text: © 2024 - ProductsApp - [Privacy](#)

*Figure 7 The product creation form is filled with details for a new product before submission.*

## Updated Products Table After Submission

The screenshot displays the Visual Studio IDE with the 'ProductsApp' project open. The 'dbo.Products [Data]' table is selected, showing the following data:

Id	Name	Price	Description	CreatedAt	ImageURL
	Sample Product	19.99	A simple Sampl...	11/19/2024 12:...	https://gidsha.c...
NULL	NULL	NULL	NULL	NULL	NULL

The bottom of the window shows the 'Output' window with the following messages:

```
The thread '[Thread Destroyed]' (14420) has exited with code 0 (0x0).  
The thread '[Thread Destroyed]' (25048) has exited with code 0 (0x0).  
The thread '[Thread Destroyed]' (26076) has exited with code 0 (0x0).  
The thread '[Thread Destroyed]' (32576) has exited with code 0 (0x0).  
The thread '[Thread Destroyed]' (7004) has exited with code 0 (0x0).  
The thread '[Thread Destroyed]' (18064) has exited with code 0 (0x0).  
The thread '[Thread Destroyed]' (31016) has exited with code 0 (0x0).
```

The right-hand side of the IDE shows the 'Diagnostic Tools' panel with a 'Diagnostics session: 3:51 minutes' and various performance metrics like Events, Process Memory (MB), and CPU (% of all processors).

Figure 8 The new product is successfully added to the database and displayed in the product table.

## Initial View of All Products Page

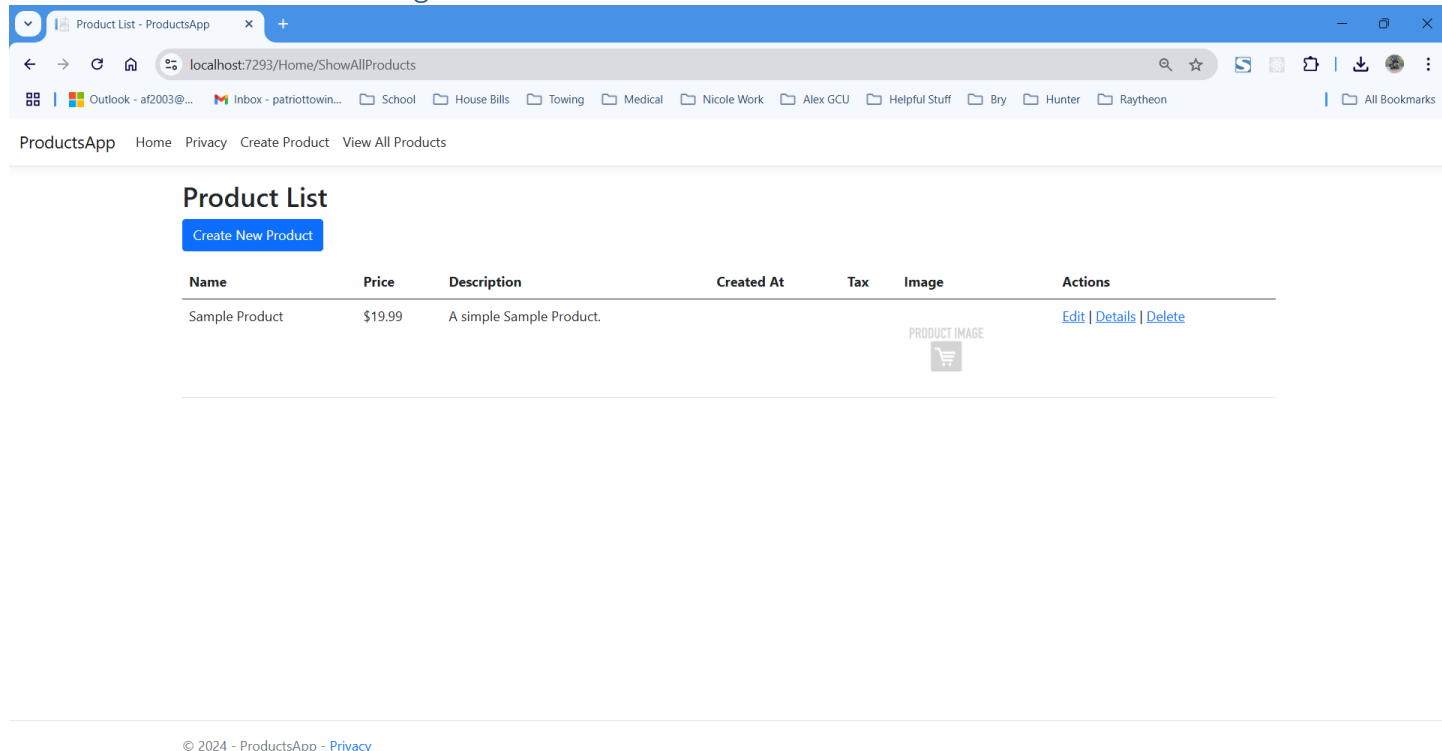


Figure 9 The "View All Products" page shows a complete list of all available products, including the newly added item.

## Adding Dynamic Tax Calculation

### Tax Calculation in Create Product Form

The screenshot shows a web browser window with the address bar displaying 'localhost:7293/Home/ShowCreateProductForm'. The browser's address bar and tabs are visible at the top. Below the browser window, the 'Create Product' form is displayed. The form has the following fields and values:

- Name:** Upgraded Sample Product
- Price:** 29.99
- Description:** An Upgraded Simple Sample Product.
- CreatedAt:** 11/19/2024
- Estimated Tax:** \$2.40
- ImageURL:** https://gidsha.com/assets/uploads/product2.png

At the bottom of the form, there is a blue 'Create' button and a link labeled 'Back to List'.

Figure 10 The tax amount is dynamically calculated based on the entered product price and displayed in real-time.

## Tax Display on All Products Page



Product List - ProductsApp

localhost:7293/Home/ShowAllProducts

ProductsApp Home Privacy Create Product View All Products

### Product List

Create New Product

Name	Price	Description	CreatedAt	Tax	ImageURL	Actions
Sample Product	19.99	A simple sample product	11/19/2024	1.60		<a href="#">Edit</a>   <a href="#">Details</a>   <a href="#">Delete</a>
Upgraded Sample Product	29.99	An upgraded simple sample product	11/19/2024	2.40		<a href="#">Edit</a>   <a href="#">Details</a>   <a href="#">Delete</a>

© 2024 - ProductsApp - [Privacy](#)

Figure 11 The calculated tax for each product is displayed alongside other product details.

## Selecting an Image for a Product

The screenshot shows a web browser window with the address bar at `localhost:7293/Home/ShowCreateProductForm`. The browser's bookmark bar lists various folders like 'Outlook', 'Inbox', 'School', 'House Bills', etc. The page title is 'Create Product - ProductsApp'. The navigation bar includes links for 'ProductsApp', 'Home', 'Privacy', 'Create Product', and 'View All Products'.

The 'Create Product' form contains the following fields and controls:

- Name:** A text input field containing the value 'Laptop'.
- Price:** A text input field containing the value '1500'.
- Description:** A text area containing the value 'A Laptop for Work or Fun.' with a small edit icon on the right.
- CreatedAt:** A read-only text field displaying '11/19/2024'.
- Estimated Tax (Formatted):** A read-only text field displaying '\$120.00'.
- Select Existing Image:** A dropdown menu that is currently open, showing a single option: 'LaptopProduct.png'.
- Upload New Image:** A section with two buttons: 'Choose File' and 'No file chosen'.
- Create:** A blue button to submit the form.
- Back to List:** A blue hyperlink below the 'Create' button.

At the bottom of the page, there is a footer: '© 2024 - ProductsApp - [Privacy](#)'.

Figure 12 Users can select an existing image from a dropdown while creating a new product.

## Products Displayed with Associated Images






Product List - ProductsApp

localhost:7293/Home/ShowAllProducts

ProductsApp Home Privacy Create Product View All Products

### Product List

Create New Product

Name	Price	Description	CreatedAt	Tax	ImageURL	Actions
Laptop	1500.00	A Laptop for Work or Fun.	11/19/2024	120.00		<a href="#">Edit</a>   <a href="#">Details</a>   <a href="#">Delete</a>
Smart Phone	700.00	A smart phone for everyday needs.	11/19/2024	56.00		<a href="#">Edit</a>   <a href="#">Details</a>   <a href="#">Delete</a>
Headphones	100.00	Headphones for gaming or music.	11/19/2024	8.00		<a href="#">Edit</a>   <a href="#">Details</a>   <a href="#">Delete</a>
Smart Watch	400.00	A smart watch to pair with your smart phone.	11/19/2024	32.00		<a href="#">Edit</a>   <a href="#">Details</a>   <a href="#">Delete</a>
Bluetooth Speaker	50.00	A wireless speaker for music.	11/19/2024	4.00		<a href="#">Edit</a>   <a href="#">Details</a>   <a href="#">Delete</a>

© 2024 - ProductsApp - [Privacy](#)

<https://localhost:7293/Home/Details/6>

Figure 13 The product view displays products along with their associated images.

## Products Displayed with Associated Images in Grid View

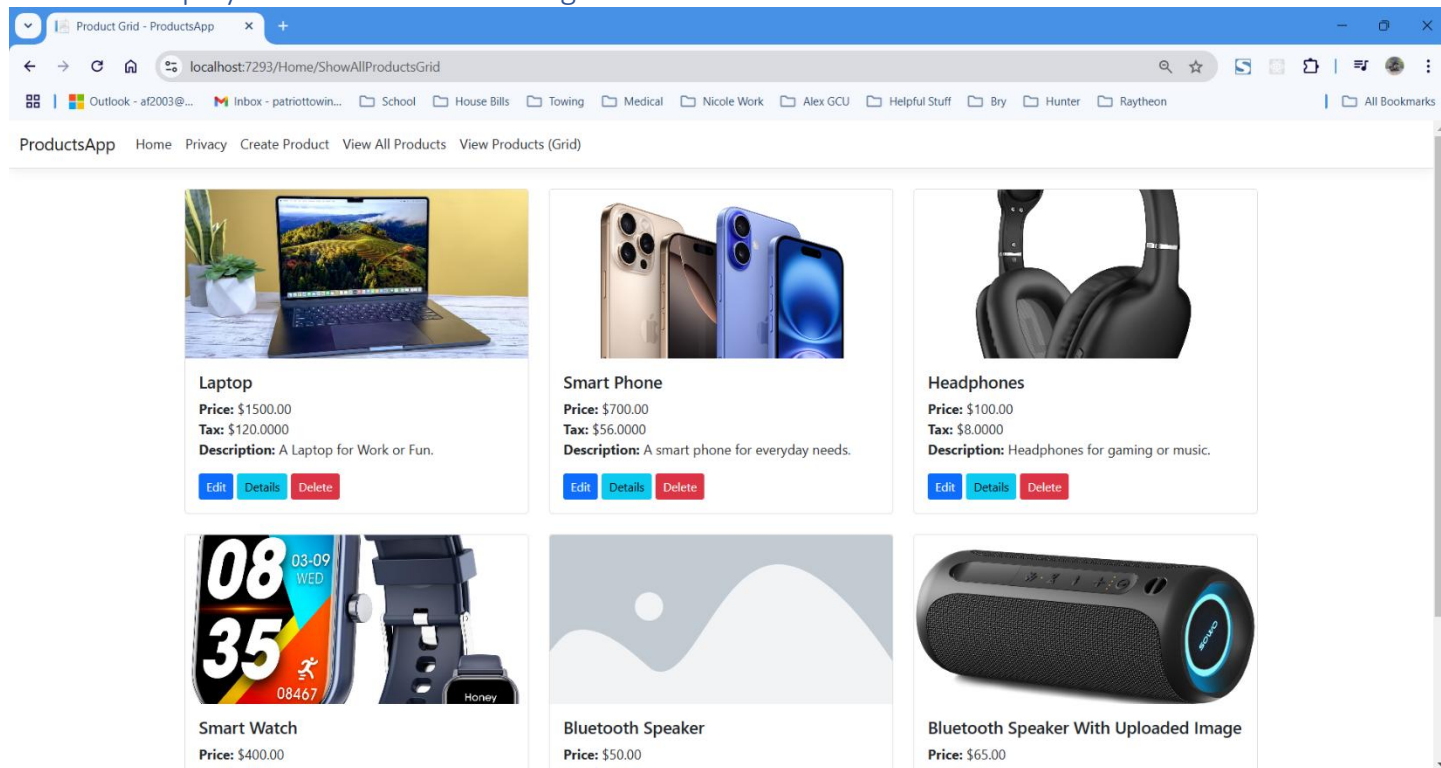


Figure 14 The product grid view displays products along with their associated images.



## Editing and Deleting Products

### Edit Product Page with Pre-Filled Data

The screenshot shows a web browser window with the address bar displaying `localhost:7293/Home/ShowUpdateProductForm/6`. The browser's address bar and tabs are visible at the top. Below the browser window, the 'Edit Product' form is displayed. The form contains the following fields and values:

- Name:** Smart Phone
- Price:** 700.00
- Description:** A smart phone for everyday needs.
- CreatedAt:** 11/19/2024
- Estimated Tax (Formatted):** \$56.00
- Select Existing Image:** -- Select an image --
- Upload New Image:** Choose File No file chosen







At the bottom of the form, there is a blue 'Update' button and a blue link labeled 'Back to List'.

© 2024 - ProductsApp - [Privacy](#)

*Figure 15 The edit product form is pre-populated with existing product data for easy updates.*

## Delete Product Confirmation Dialog

The screenshot shows a web browser window with the address bar at `localhost:7293/Home/ShowAllProducts`. The page title is "Product List" and there is a "Create New Product" button. A modal dialog box is open, titled "localhost:7293 says", with the text "Are you sure you want to delete this product?" and "OK" and "Cancel" buttons. The background shows a table of products with columns: Name, Price, Description, Date, Tax, ImageURL, and Actions. The "Delete" button in the "Actions" column for the "Bluetooth Speaker" row is highlighted with a red border.

Name	Price	Description	Date	Tax	ImageURL	Actions
Laptop	1500.00	A Laptop for Work or Fun.	11/19/2024	120.00		<a href="#">Edit</a>   <a href="#">Details</a>   <a href="#">Delete</a>
Smart Phone	700.00	A smart phone for everyday needs.	11/19/2024	56.00		<a href="#">Edit</a>   <a href="#">Details</a>   <a href="#">Delete</a>
Headphones	100.00	Headphones for gaming or music.	11/19/2024	8.00		<a href="#">Edit</a>   <a href="#">Details</a>   <a href="#">Delete</a>
Smart Watch	400.00	A smart watch to pair with your smart phone.	11/19/2024	32.00		<a href="#">Edit</a>   <a href="#">Details</a>   <a href="#">Delete</a>
Bluetooth Speaker	50.00	A wireless speaker for music.	11/19/2024	4.00		<a href="#">Edit</a>   <a href="#">Details</a>   <a href="#">Delete</a>
Bluetooth Speaker With Uploaded Image	65.00	I needed a second speaker so I could test image uploads	11/19/2024	5.20		<a href="#">Edit</a>   <a href="#">Details</a>   <a href="#">Delete</a>

© 2024 - ProductsApp - [Privacy](#)  
<https://localhost:7293/Home/Delete/9>

Figure 16 The user is prompted to confirm the deletion of a specific product.

## Product Successfully Deleted from Table






Product List - ProductsApp

localhost:7293/Home/ShowAllProducts

ProductsApp Home Privacy Create Product View All Products View Products (Grid)

### Product List

Create New Product

Name	Price	Description	CreatedAt	Tax	ImageURL	Actions
Laptop	1500.00	A Laptop for Work or Fun.	11/19/2024	120.00		<a href="#">Edit</a>   <a href="#">Details</a>   <a href="#">Delete</a>
Smart Phone	700.00	A smart phone for everyday needs.	11/19/2024	56.00		<a href="#">Edit</a>   <a href="#">Details</a>   <a href="#">Delete</a>
Headphones	100.00	Headphones for gaming or music.	11/19/2024	8.00		<a href="#">Edit</a>   <a href="#">Details</a>   <a href="#">Delete</a>
Smart Watch	400.00	A smart watch to pair with your smart phone.	11/19/2024	32.00		<a href="#">Edit</a>   <a href="#">Details</a>   <a href="#">Delete</a>
Bluetooth Speaker With Uploaded Image	65.00	I needed a second speaker so I could test image uploads	11/19/2024	5.20		<a href="#">Edit</a>   <a href="#">Details</a>   <a href="#">Delete</a>

© 2024 - ProductsApp - [Privacy](#)

Figure 17 The deleted product is no longer visible on the "View All Products" page.

## Searching for Products

### Product Search Input Page

The screenshot shows a web browser window with the title 'Search Products - ProductsApp'. The address bar displays 'localhost:7293/Home/ShowSearchForm'. The browser's bookmark bar includes links to Outlook, Gmail, and various folders. The page's navigation bar contains 'ProductsApp', 'Home', 'Privacy', 'Create Product', 'View All Products', 'View Products (Grid)', and 'Search Products'. The main content area is titled 'Search Products' and features a form with two input fields: 'Name' (containing 'Headphones') and 'Description'. A blue 'Search' button is positioned below the 'Description' field. The footer of the page shows the copyright notice '© 2024 - ProductsApp - Privacy'.

ProductsApp Home Privacy Create Product View All Products View Products (Grid) Search Products

### Search Products

Name

Description

Search

© 2024 - ProductsApp - [Privacy](#)

*Figure 18 The search input page allows users to search for products by name or description.*

## Search Results Display

Search Results - ProductsApp


localhost:7293/Home/SearchForProducts

ProductsApp Home Privacy Create Product View All Products View Products (Grid) Search Products

### Search Results

**Search Criteria:**  
Name: Headphones

[Back to Search](#)

Name	Price	Description	CreatedAt	Tax	ImageURL	Actions
Headphones	200.00	Headphones for gaming or music.	11/19/2024	16.00		<a href="#">Edit</a>   <a href="#">Details</a>   <a href="#">Delete</a>

© 2024 - ProductsApp - [Privacy](#)

Figure 19 The search results page displays products that match the search criteria entered by the user.

# Viewing Product Details

## Detailed View of a Single Product

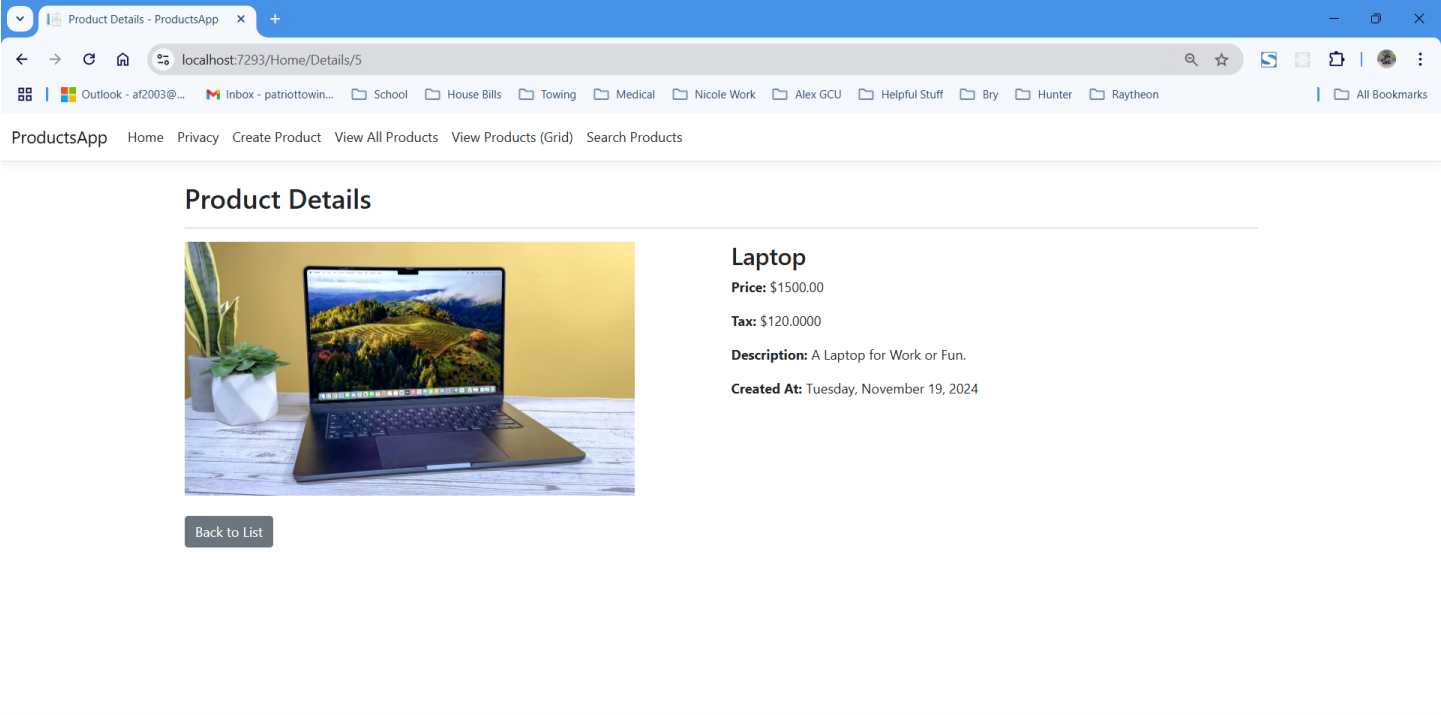


Figure 20 The product details page provides in-depth information about a selected product, including its image, price, tax, description, and creation date.

## Centralized Configuration with appsettings.json

### Configuration Settings in appsettings.json

```
{
  "Logging": {
    "LogLevel": {
      "Default": "Information",
      "Microsoft.AspNetCore": "Warning"
    }
  },
  "ConnectionStrings": {
    "ProductsDatabase": "Data Source=(localdb)\\ProjectModels;Initial Catalog=ProductsDatabase;Integrated Security=True;Connect Timeout=30;Encrypt=False;Trust Server Certificate=False;Application Intent=ReadWrite;Multi Subnet Failover=False"
  },
  "ProductMapperSettings": {
    "CurrencyFormat": "C",
    "DateFormat": "D",
    "TaxRate": 0.08
  },
  "AllowedHosts": "*"
}
```

Figure 21 The `appsettings.json` file centralizes important configuration settings such as the database connection string and tax rate, enabling dynamic updates without requiring recompilation.

## Summary of Key Concepts

Through Activity 5, I gained valuable hands-on experience in building a full-stack CRUD application using the n-layer design approach. This activity reinforced the importance of **separation of concerns**, where the application was divided into distinct layers to ensure modularity and maintainability. I also learned the practical benefits of **dependency injection**, which allowed me to decouple components, promoting both flexibility and testability in the application.

A significant takeaway was the use of **dynamic configuration** through the `appsettings.json` file, which centralized important settings like database connection strings and tax rates. This approach made it easier to update configurations dynamically and adapt to different environments without recompiling the application. The implementation of **CRUD operations**—creating, reading, updating, and deleting data—provided me with a solid understanding of essential functionality in data-driven applications.

I focused on **UI enhancements** to improve the user experience. Features like dynamic tax calculation, image selection for products, and detailed product views added a level of polish and usability to the application. This project highlighted the importance of structured application design, preparing me to develop scalable, maintainable, and user-friendly web applications in the future.