# ASP.NET Fundamentals Regular Exam – 19 October 2024

# A desk with a computer and a computer Description automatically generated

Problems for the retake exam for the ["ASP.NET Core Fundamentals"](https://softuni.bg/trainings/4707/asp-net-fundamentals-september-2024) course @ [SoftUni](https://softuni.bg)

## Welcome to DeskMarket

*DeskMarket is your go-to platform for the latest in technology. Whether you're looking for high-performance laptops, powerful workstations, or top-tier accessories, DeskMarket has everything you need to elevate your work and play.*

## Technological Requirements and Overview

* Use the provided skeleton – [**DeskMarket-Skeleton.zip**](http://svn.softuni.org/admin/svn/soft-tech/Sept-2024/CSharp/15-CSharp-Web-Oct-2024/01-Regular-Exam/DeskMarket-Skeleton.zip)
* All of the needed packages have been installed.

**The Technological Requirements are ABSOLUTE. If you do not follow them, you will NOT be scored for other Requirements.**

The provided skeleton consists of:

* **Areas/Identity/Pages** – you have to scaffold Identity here
* **Controllers** – you should implement the controllers logic here
* **Data** – you should hold the entities models here
* **Models** – you should implement the models here
* **Views** – you are provided with the needed views. Your task is to implement some logic regarding the logged-in/logged-out user
* **appsettings.json** – don't forget to change the your **connection string**
* **Program.cs** – you should fulfil the security and password requirements here

**NOTE:** You should seed the database with provided in advance data regarding the **Category** entity. In order to do this, remove the comments from the block of code in the **protected override void OnModelCreating(ModelBuilder builder)** method of the **DbContext.**

**NOTE: Don't forget to uncomment the code inside the views while you implement your logic.**

Now that you know the **Technological Requirements**, let us see what the **Functional Requirements** are.

## Identity Requirements

You should **scaffold** **Identity** and use the **default** **IdentityUser**.

Remove the unnecessary code from the **Login.cshtml** and **Register.cshtml** files and leave only the needed code in order for the app to be functioning correctly.

**NOTE: Don't worry about the views' style – once you scaffold Identity and remove the unnecessary code, the** [**Login**](#_Login_Page_(logged-out) **and** [**Register**](#_Register_Page_(logged-out) **pages should look like shown below. Don't add any classes to the views of those two pages!**

The **password** requirements for the **IdentityUser** are the following:

* Require confirmed account: **false**
* Require digits: **false**
* Require non-alphanumeric characters: **false**
* Required uppercase letters: **false**

## Database Requirements

The **Database** of **DeskMarket**:

### Product

* Has Id – a unique **integer, Primary Key**
* Has ProductName – a string with **min** length **2** and **max** length **60** (**required**)
* Has Description – string with **min** length **10** and **max** length **250** (**required**)
* Has Price – **decimal in range[1.00m;3000.00m], (required)**
* Has **ImageUrl** – nullable string (**not required**)
* Has SellerId – string (**required**)
* Has Seller – IdentityUser (**required**)
* Has AddedOn – DateTime with format "**dd-MM-yyyy**" (**required**)
  + The **DateTime** format is recommended, if you are having troubles with this one
  + You are **free to use another one**)
* Has CategoryId – **integer, foreign key (required)**
* Has Category – Category (**required**)
* Has IsDeleted – bool **(default value == false)**
* Has **ProductsClients** – a collection of type **ProductClient**

**Feel free to use the new syntax for realization of the many-to-many relation without a mapping table.**

### Category

* Has Id – a unique **integer, Primary Key**
* Has Name – a string with min length **3** and max length **20** (**required**)
* Has **Products** – a collection of type **Product**

### ProductCategory

* Has ProductId – integer, PrimaryKey, foreign key (**required**)
* Has Product – Product
* Has ClientId – string, PrimaryKey, foreign key (**required**)
* Has Client – IdentityUser

Implement the entities with the **correct datatypes** and their **relations**.

## Page Requirements

### Index Page (logged-out user)

A keyboard and mouse with rainbow colored lights

Description automatically generated

### Index Page (logged-in user)

A computer monitor and speaker

Description automatically generated

### Login Page (logged-out user)

A screenshot of a computer

Description automatically generated

### Register Page (logged-out user)

A screenshot of a computer

Description automatically generated

### /Product/Add (logged-in user)

A screenshot of a computer

Description automatically generated

**NOTE:** the **DateTime** format is recommended, if you are having troubles with the one in the image, you are **free** to use **another** one.

### /Product/Index (logged-out user)

A screenshot of a website

Description automatically generated

**NOTE**: If user is not **logged in** and tries to go to the **Home page**, the application must redirect them to the **/Home/Index**

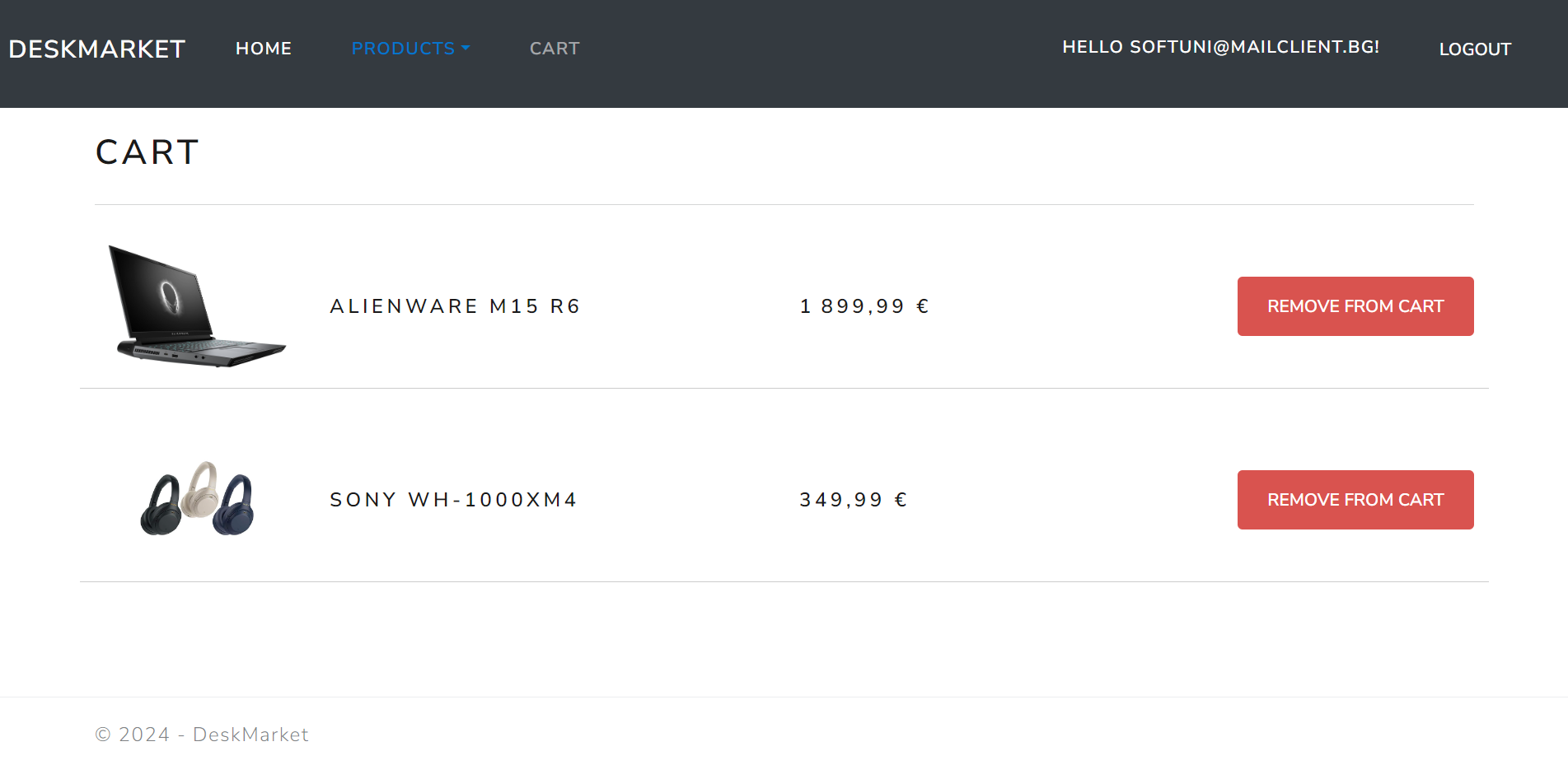
### /Product/Index (logged-in user)

A screenshot of a product

Description automatically generated

**NOTE**: If user is **logged in** and tries to go to the **Home page**, the application must redirect them to the **/Product/Index**

### /Product/Cart (logged-in user)



### /Product/Edit/{id} (logged-in user)

**NOTE:** the **DateTime** format is recommended, if you are having troubles with the one in the image, you are **free** to use **another** one.

### /Product/AddToCart?id={id} (logged-in user)

This feature adds the selected product to the user's cart. If the product is already in the user's cart, it should not be added again. If the addition is successful, the users will be redirected to their cart page "**/Product/Cart**" page.

### /Product/RemoveFromCart?id={id} (logged-in user)

This feature removes the selected product from the user's cart. If the removal is successful, the user will be redirected to their cart page "**/Product/Cart**" page.

**NOTE**: The templates should look **EXACTLY** as shown above.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

**Product/Details/{id} (logged-in user, seller of a product)**

A computer screen shot of a desk

Description automatically generated

**NOTE**: If the **currently logged-in user is the seller of the product**, the **"Delete"** and **"Edit"** buttons should be visible, allowing them full access to the product functionalities.

However, if the **logged-in user is** **not the seller** of the game, the **"Delete"** and **"Edit"** buttons should not be visible. The **"Buy"** button should be visible. This ensures that the product seller and not-seller-user have different application functionalities access.

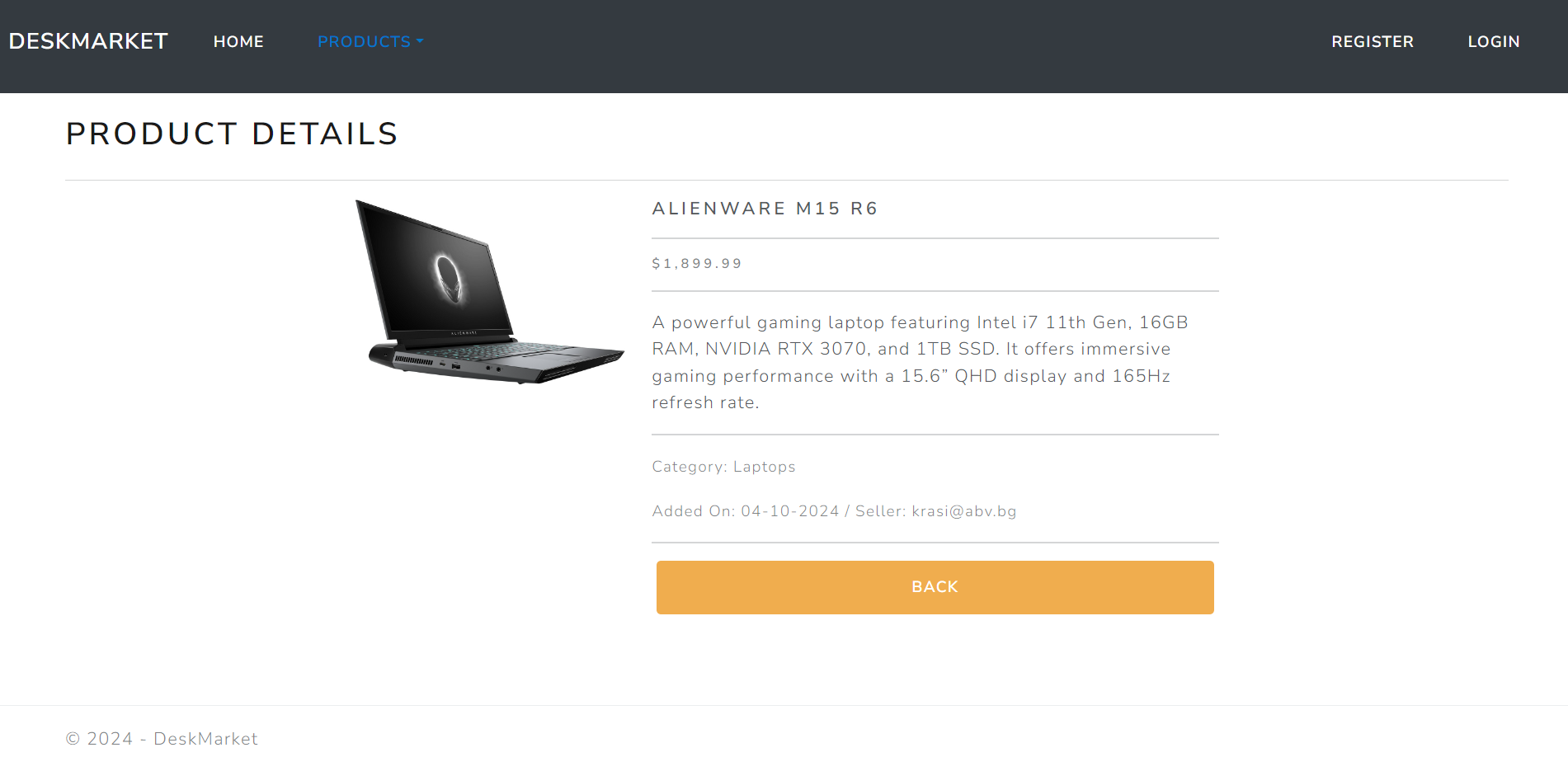
Guests (**unauthenticated users**) should be able to **access the product details page** but **won't have access to actions** like **buying**, **editing**, or **deleting** a product.

**Product/Details/{id} (logged-in user, not a seller of a product)**

**A screenshot of a computer

Description automatically generated**

**Product/Details/{id} (guest user, not logged-in)**

****

**Product/Delete/{id} (logged-in user, seller of a product)**

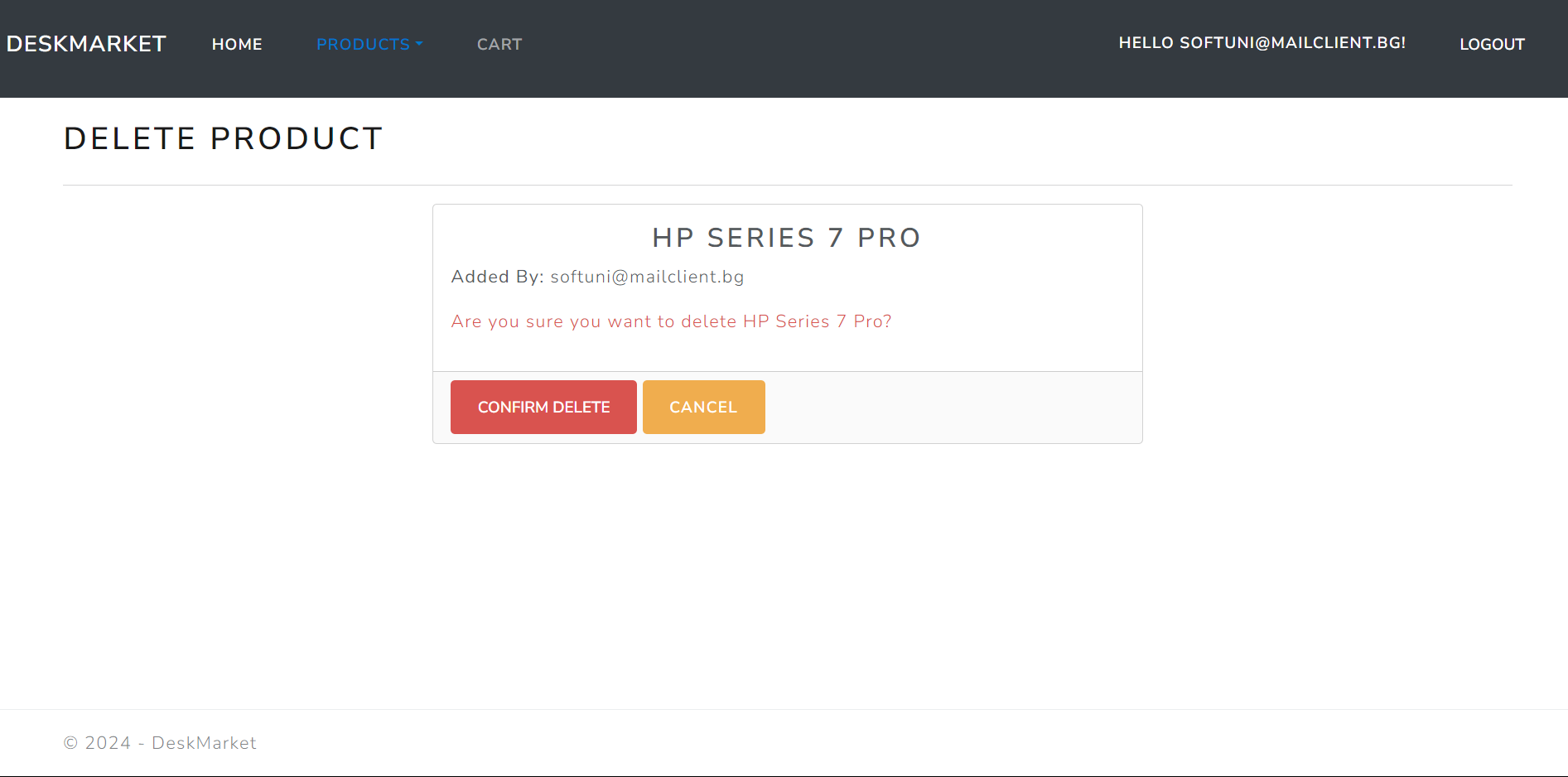
When the logged-in user who is the seller of a product accesses this page, they will be able to initiate the deletion of a product. However, this is a soft delete process, meaning that the product will remain in the database but   
will no longer appear in the product listings. The product can no longer be bought or viewed by users, effectively removing it from the storefront.

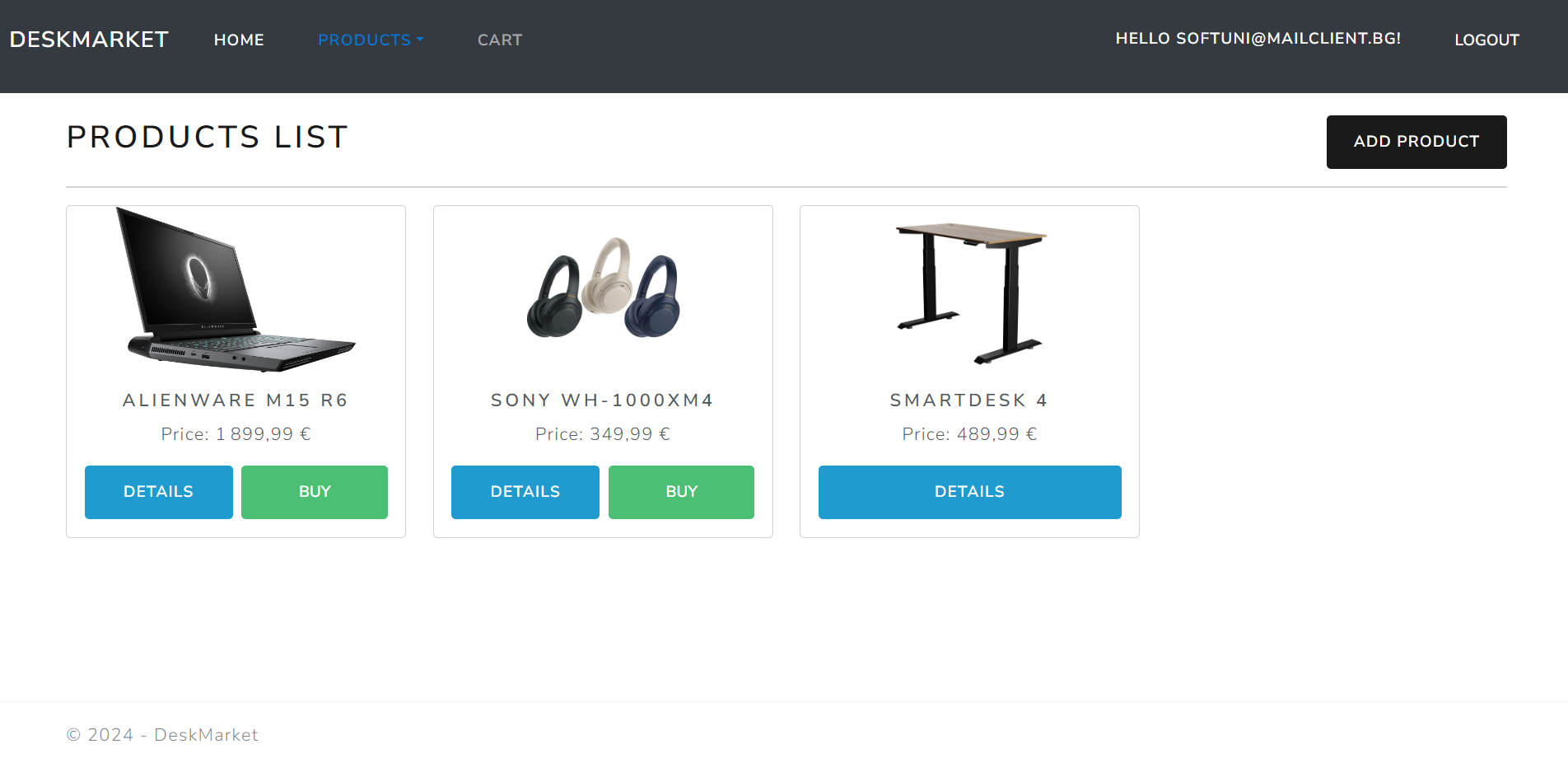
After **confirming the delete**, the seller will be **redirected to the product controller, index** **action**. This process is **one-directional and irreversible** through the user interface, as once deleted, the product will not be displayed in the system again **unless manually restored through the database**.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated****



A screenshot of a computer

Description automatically generated

## Functionality

The functionality of the **DeskMarket** Platform is a very simple application.

### Users

Guests can Register, Login and view the **HomePage** and Index Page.

Users can **publish products** and **edit products** **only they have added**.

Users can see **added** **products** by all users on the Home Page (/Product/Index).

If the **user is the seller of the product**, they can't see the **[Buy]** button. If the user is not the seller of the product, they can add the product to their cart [**Cart**], , they can also see the **[Details]** button

### Product

Product can be **added** by users. All added products are visualized on the Home Page (/Product/Index).

Products are visualized on the Home Page (/Product/Index) with **some** of their information.

Products are visualized on the Index Page (/Product/Index) with **one or two buttons**:

* If the user **IS NOT** the **seller** of the product – **[Details][Buy]**
* If the user **IS** the **seller** of the game – **[Details]**

The **[Buy]** button adds the product to the user's collection of products (Cart), **unless it is already added**.

The **[Details]** button displays a new page with the full information for the selected product. The actions and buttons displayed on this page will vary depending on the user's identity and role:

* **Guest (Not Logged In)**:  
  The user can see the full details of the product but will not be able to buy, edit, or delete the product. No action buttons (**[Buy]**, **[Edit]**, **[Delete]**) will be shown.
* **Logged-in User (Not the Seller)**:  
  The user will see the full details of the product and will have access to the **[Buy]** button, allowing them to add the product to their cart (if it has not already been purchased). The **[Edit]** and **[Delete]** buttons will not be displayed as they are not the seller of the product.
* **Seller of the Product**:  
  The seller will have full access to the product's details and will be able to see both the **[Edit]** and **[Delete]** buttons. However, the **[Buy]** button will not be displayed since the seller cannot purchase their own product.

The **[Edit]** button displays a new page with a form, filled in with **all** of the **info** for the **selected** **product**. Users can change this info and save it.

Usershave a Cart page where only the productsthey have selected to buy are displayed.

* The **[Remove From Cart]** button **removes** the **product** from the **user's** collection of **products**.

### Redirections

* Upon successful Login of an **IdentityUser**, you should be redirected to the /Product/Index
* Upon successful Publishing of a Product, you should be redirected to the /Product/Index
* Upon successful Adding a **Product** to the **User's Cart**, should be redirected to the /Product/Index
* Upon successful **Editing** of a **Product,** you should be redirected to the **/Product/Details/{product\_id}**
* Upon successful Removal of a **Product** from the **User's Cart**, should be redirected to the /Product/Cart
* If a User **buys** a product, the **[Buy]** button disappears, and the user should not be able to duplicate products in the **User's Cart.** The application should also ensure that if a user tries to add a product they've already bought via other means (such as **URL manipulation**), the **system will not duplicate the product in their cart** or collection.
* Upon successful Logout of a User, you should be redirected to the Home Page
* If any of the **validations** in the POST forms **don't pass**, **redirect** to the **same page** (**reload/refresh** it)

## Security

The Security section mainly describes access requirements. Configurations about which users can access specific functionalities and pages:

* Guest (not logged in) users can access the Home page, **Product Index** page, **Product Details** page.
* Guest (not logged in) users can access the Login page.
* Guest (not logged in) users can access the Register page.
* Guests (not logged in) cannot access Users-only pages.
* Users (logged in) cannot access Home pages.
* Users (logged in) can access the Product/Add page and functionality.
* Users (logged in) can access the Product/Edit page and functionality.
* Users (logged in) can access the Product/Index page.
* Users (logged in) can access the **Product Cart** page.
* Users (logged in) can access Logout functionality.
* Users (logged in) cannot access the Product/Edit page of a Product that have another user as a seller.

## Code Quality

Make sure you provide the best architecture possible. Structure your code into different classes, follow the principles of high-quality code (**SOLID**). You will be scored for the Code Quality and Architecture of your project.

## Scoring

### Identity Requirements – 5 points

### Database Requirements – 10 points

### Template Requirements – 10 points

### Functionality – 50 points

### Security – 5 points

### Code Quality – 10 points

### Data Validation – 10 points