**75%**

**You may use your textbook, labs and lecture notes. This is an individual assignment meaning you can not get technical assistance from your instructor or anyone else. All code submitted must be your original work. Code must be committed and pushed to your GitHub.com repository within the HOT2 folder.**

**HOT2**

For this assignment you will be building a Sales Order database application with a One-To-One relationship.

The application should have the following features:

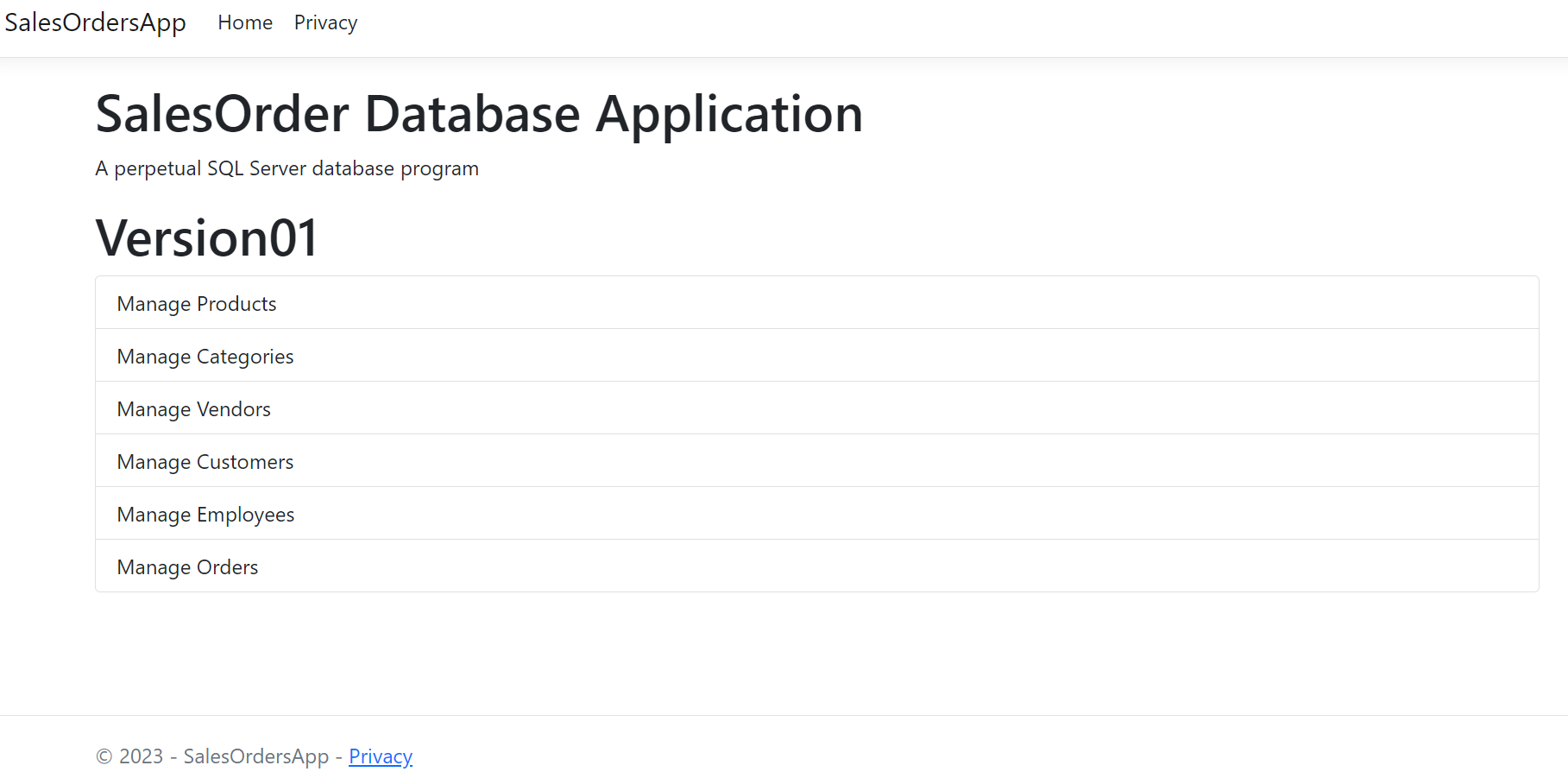
* Show current products, ordered by product name (as shown).
* New products can be added.
* Existing products can be updated.
* Existing products can be deleted.

For this test, the database should have two tables:

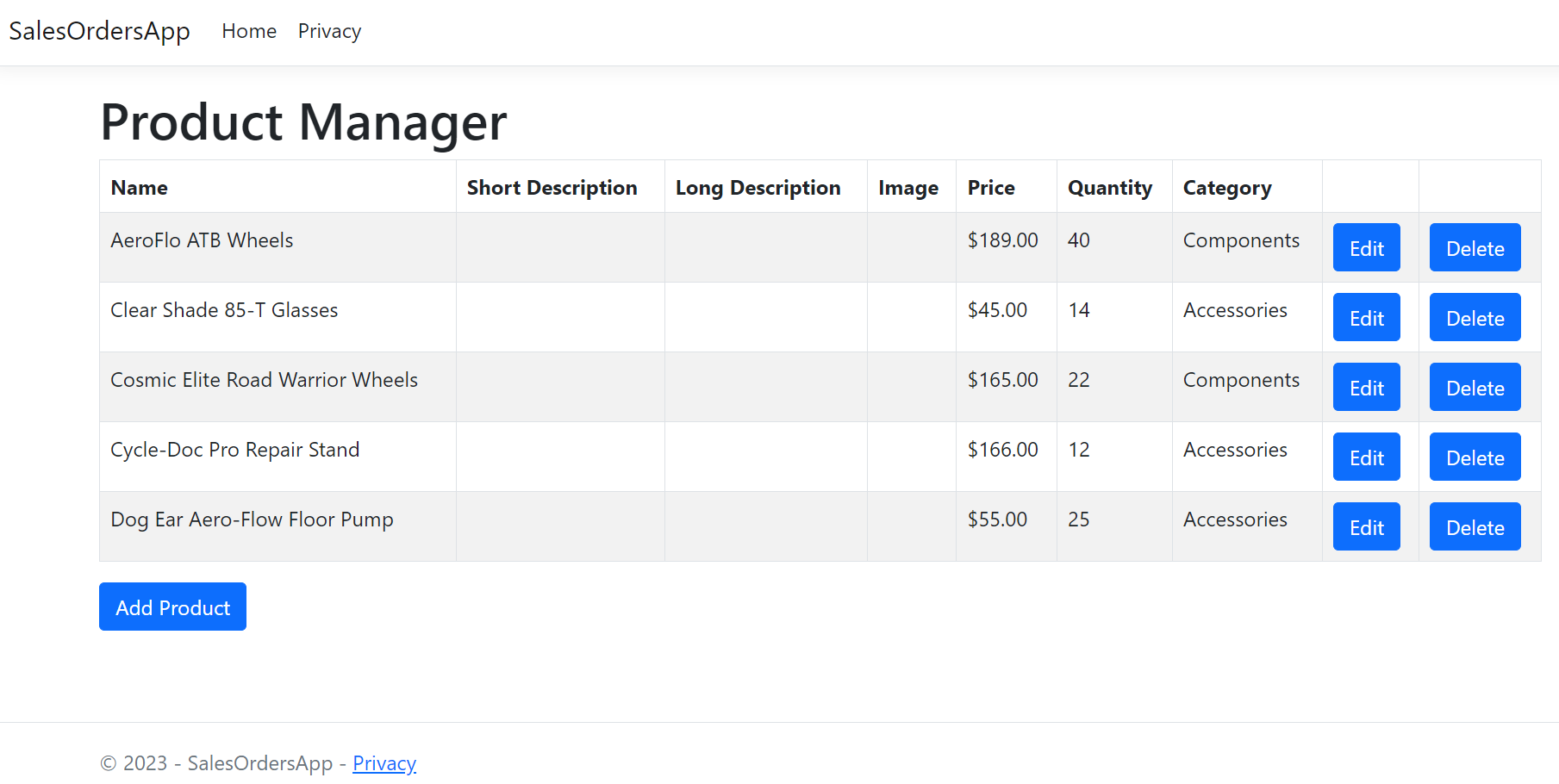
* Products **(Convert/add just the first 5 products from the attached 05-SOProducts.SQL in your seed data)**
  + ProductID: int
  + ProductName: string
  + ProductDescShort: string (will be empty, or hold just “” for now)
  + ProductDescLong: string (will be empty, or hold just “” for now)
  + ProductImage: string (will be empty, or hold just “” for now)
  + ProductPrice: decimal (should default to 0.00)
  + ProductQty int (should default to 0)
  + CategoryID: int
* Categories **(Convert/add all 6 categories from the attached 01-SOCategories.SQL file in your seed data)**
  + CategoryID : int
  + CategoryName : string

**Screen mockups are shown on the next 5 pages.**

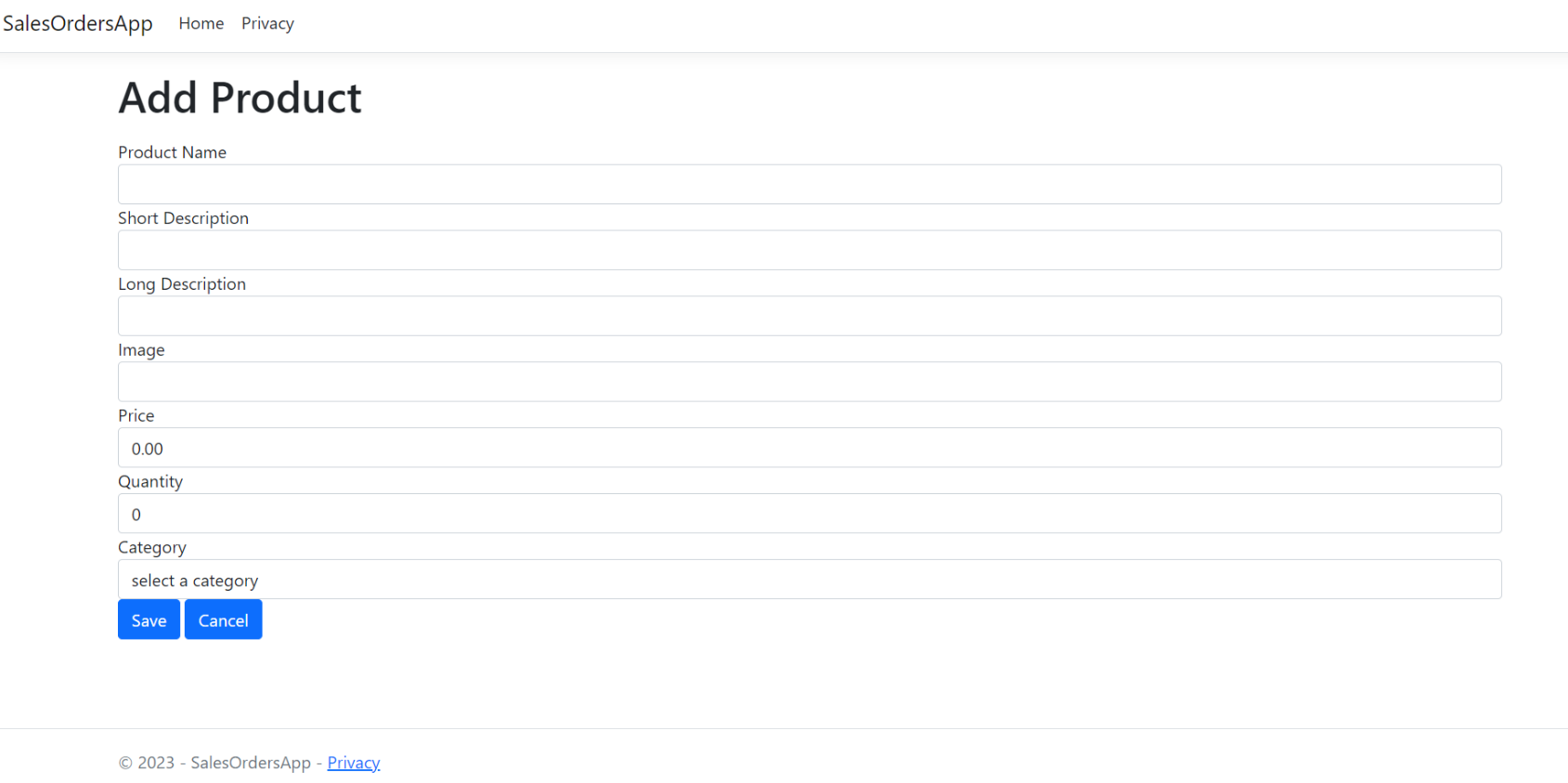
**See the scoring rubric on the last page.**



**SCREEN 1 – OPENING SCREEN WHEN RUNNING PROGRAM**



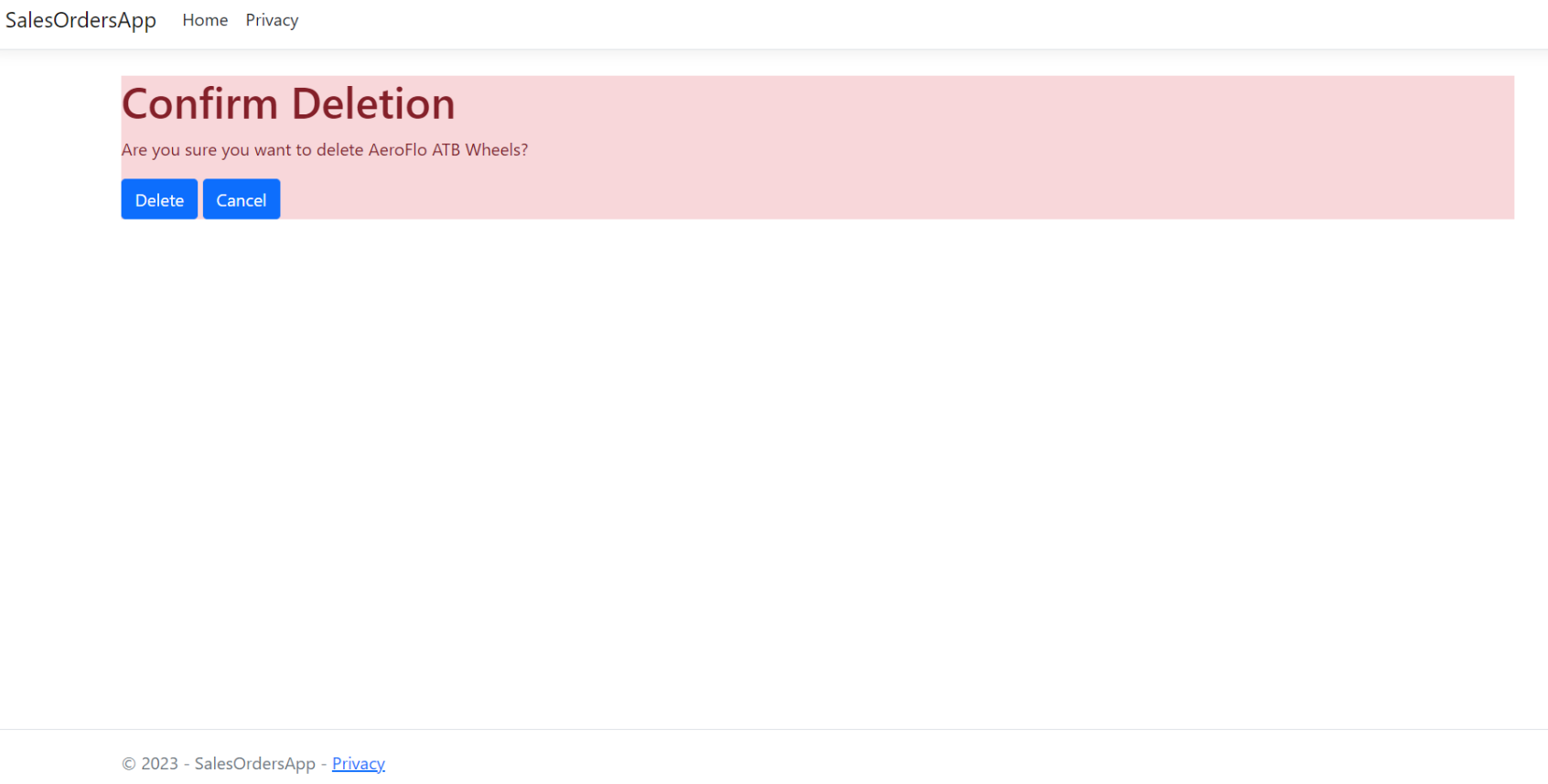
**SCREEN 2 - RESULTS AFTER CLICKING MANAGE PRODUCTS ON THE PREVIOUS SCREEN**



**SCREEN 3 – ADD PRODUCT SCREEN**



**SCREEN 4 – EDIT PRODUCT SCREEN FOR AeroFlo ATB Wheels**



**SCREEN 5 – DELETE SCREEN (After AeroFlo ATB Wheels Chosen To Delete)**

**Rubric** (no partial points)

1. Create an ASP.NET Core MVC app and add the necessary packages for Entity Framework to work with a database (5pts)
2. Name the database SalesOrdersDB (5pts)
3. Create the Product Model file using the properties shown on Page 1 with the following validation attributes (5pts)

* ProductName, ProductImage required. Include associated error messages.
* ProductPrice range should be set to between 1 – 100000
* ProductQty range should be set to between 1 - 1000

1. Add a migration and create the database with **only** a Product table (5pts)
2. Create a ProductController, including List, AddEdit, and Delete methods (5pts)
3. Create the following views:

* List.cshtml and Delete.cshtml (5pts)
* AddEdit.cshtml (5pts)

1. Add a new Categories Model file using the properties shown on Page 1 with the following validation attributes (5pts)

* CategoryName required. Include associated error messages.

1. Add a One-To-One relationship between Product and Category (5pts)
2. Add a second migration and reflect those changes in your database. (5pts)
3. Your application should be able to perform CRUD operations on Products. (no points if Product/Category relationship is not properly created)
   * Create
     1. Adds new Products to the database (5pts) Create breaks after 1st add
     2. A screenshot of a computer

        AI-generated content may be incorrect.
     3. Enforces validation rules (5pts)
   * Read
     1. Lists all Products into a table format (10pts)
   * Update
     1. Edits products in the database (5pts)
     2. Enforces validation rules (5pts)
   * Delete
     1. Removes a product from the database (5tps)
     2. Prompts user with a Confirm Delete before removing the Product (5pts)
4. Add the code to make the URLs lowercase and add a “/” at the end of the URL. (5 points)
5. Add a slug to the URL. This slug should be the name of the product, and be added to the end of the URL. (5 points) data in a Query String