# JS Back-End Retake Exam – Glow Alchemy

## Exam Rules:

1. You have **6 hours**.
2. When you are ready, delete the ***node\_modules***folder, make sure all dependencies are listed in the ***package.json*** file and submit your archiving project.
3. You must use the provided HTML & CSS resources.
4. You may **add attributes** (such as **class** and **dataset**), but it is forbidden to **change** **existing** attributes (such as **class** and **id**).
5. You may **change "href"** attributes on links and add/change the **method** and **action** attributes of HTML Forms.
6. Use **Express.js** as a back-end framework.
7. Use **MongoDB** as a database with [**mongoose**](https://mongoosejs.com/docs/)**.**
8. You can use whatever **view engine** you like ([**express-handlebars**](https://www.npmjs.com/package/express-handlebars), EJS, Pug, etc.).
9. Use [**bcrypt**](https://www.npmjs.com/package/bcrypt) **for hashing the password.**
10. It is **forbidden** to **use** **React**, **Vue**, **Angular,** etc.
11. All applications will be **examined** as follows:
    * The needed **dependencies** are installed with the command "**npm install**".
    * The application is **started** with the command "**npm run start**".
12. The examiners are **NOT** allowed to **change** anything in your code and will **NOT** use any **instruments** **outside** of the course scope.
13. You decide how to name your links.

**IMPORTANT:**

**1. THE APPLICATION MUST START FROM THE FILE "INDEX.JS" ON PORT 3000!**

**2. ONLY THE LAST SUBMISSION WILL BE EVALUATED!**

## Application Overview

Get acquainted with the provided **HTML and CSS** and create an application for the **cosmetics**.

The visitors can **view** the **Home page,** **Products page** with available cosmetics. They can also **register** with a **name,** an **email** and **password**, allowing them to create their posts and to **recommend cosmetics they are interested in** (if the **current user** is **not** the author of the post). Authors can edit or delete their post at any time.

## Functional Requirements

### Guest (Not logged in)

**Guest** navigation example:



The **application** should provide Guest (not logged in) users with the functionality to Login, Register, and **view** the Home page, **Products** page, and **Details** page.

### Users (Logged in)

**User** navigation example:



The **application** should provide **Users** (logged in) with the functionality to:

* **View the Home page and all other pages with logged-in navigation**
* **View Products page**
* **Create а new Product Post [Add Products]**
* **Access cosmetic details page [Details]**
* **Recommend cosmetics (**if the **current user** is **not** the **owner of the post)**
* **Delete or Edit the cosmetic post depending on the user's authentication (only for the owner of the current post)**

## Database Models

The **Database** of the **GlowAlchemy** application needs to support **two entities**:

### User

* name – string (required),
* email - string (required),
* password - string (required)

### Cosmetic

* **name - string (required),**
* **skin- string (required),**
* **description - string (required),**
* **ingredients - string (required),**
* **benefits – string (required),**
* **price – number (required),**
* **image: string (required),**
* **recommendList - a collection of Users (a reference to the User model)**
* **owner - object ID (a reference to the User model)**

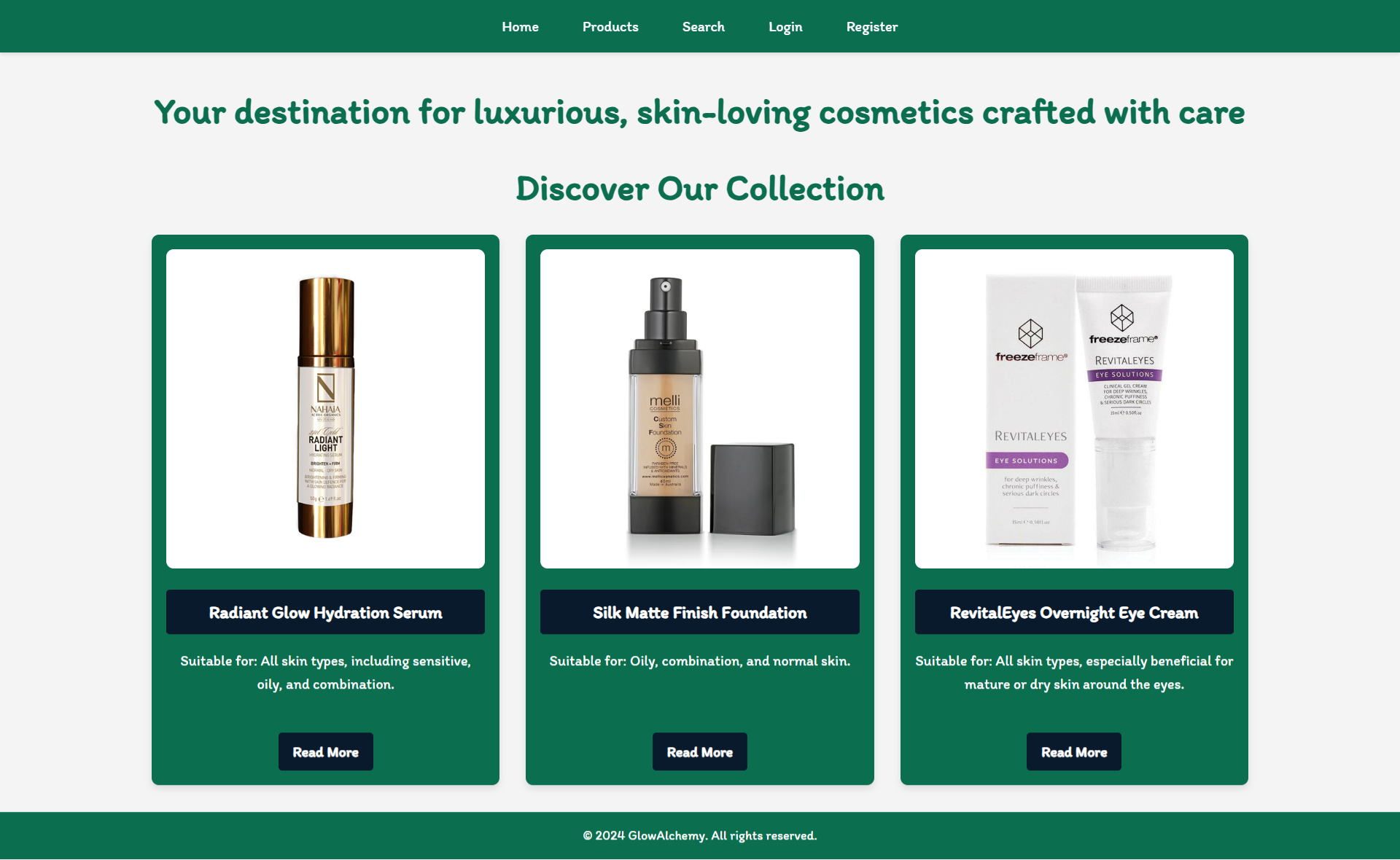
**Note:** When a user **recommend the products**, their **id** is added to the **recommendList** collection. You can add more entities, if you need.

Implement the entities with the correct data types.

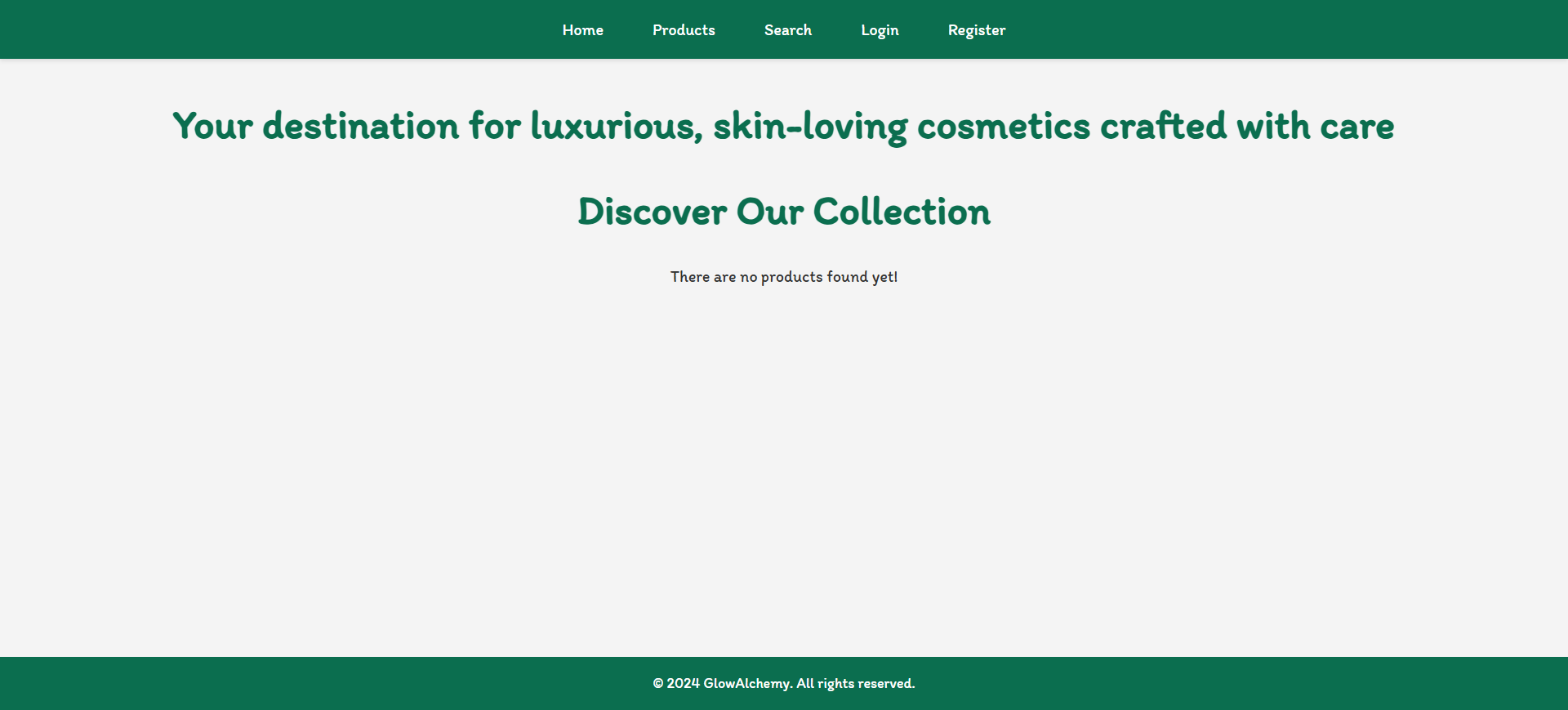
## Application Pages

#### Home Page (For logged in users and logged-out users)

Visualize **the last 3 added post. Each cosmetic must** show information about the **cosmetic name, skin type, image** as well as a page with **details** about the **cosmetic.**

****

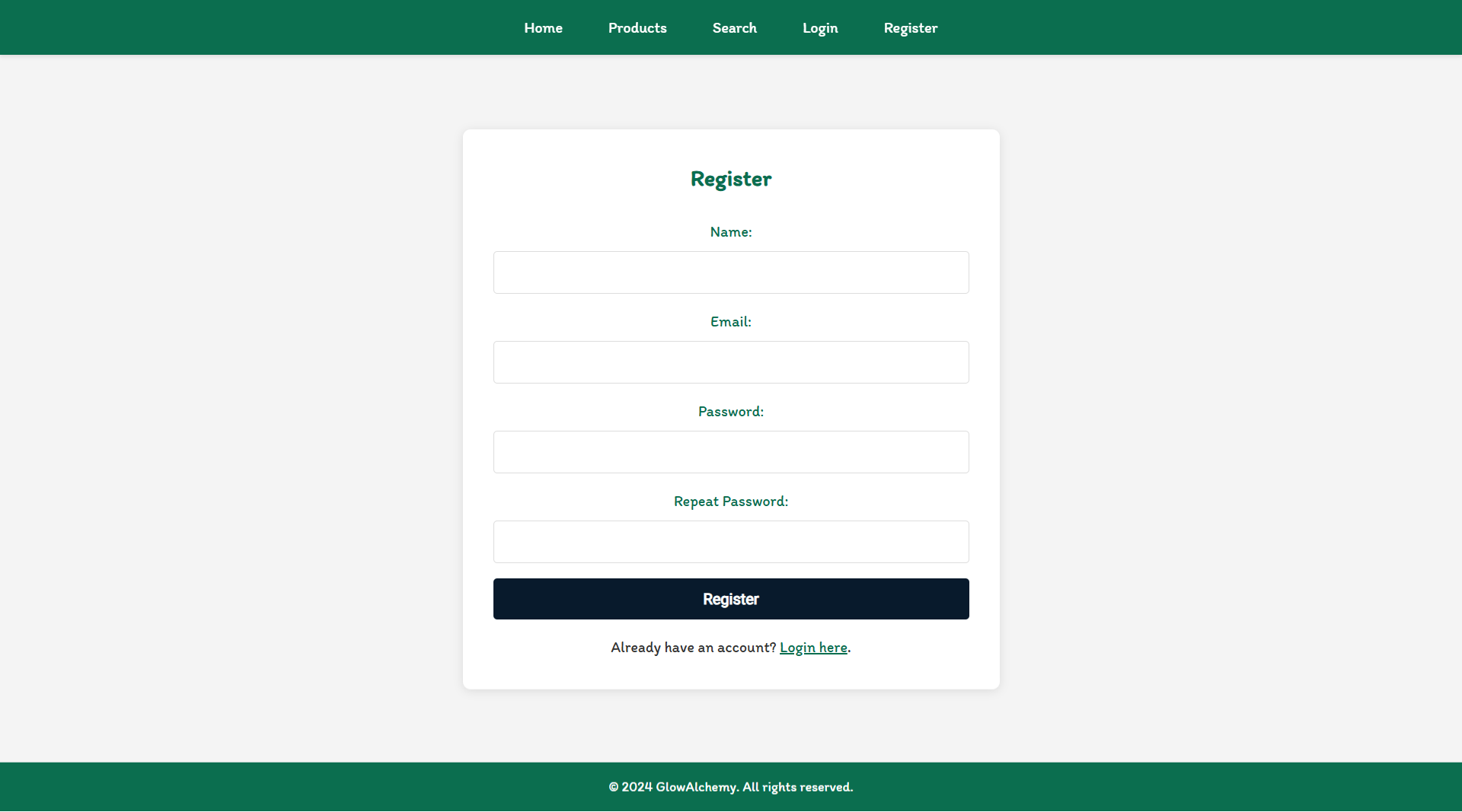
If there are **NO** post in the database yet, display **"There are no products found yet!"**

****

#### Register Page (Logged Out User)

**Register** a user inside the database with a **name**,an **email and password**. The **password** inside the **database** must be hashed (**use bcrypt**), and both passwords **must match**! After **successful** registration, the user should be logged and **redirected** to the **Home page**.If the user tries to submit **invalid data**, an **error message** should be displayed and the already entered **data should be kept** in the input fields (except for the passwords). The **Login here** button should be redirected to the **Login Page**.

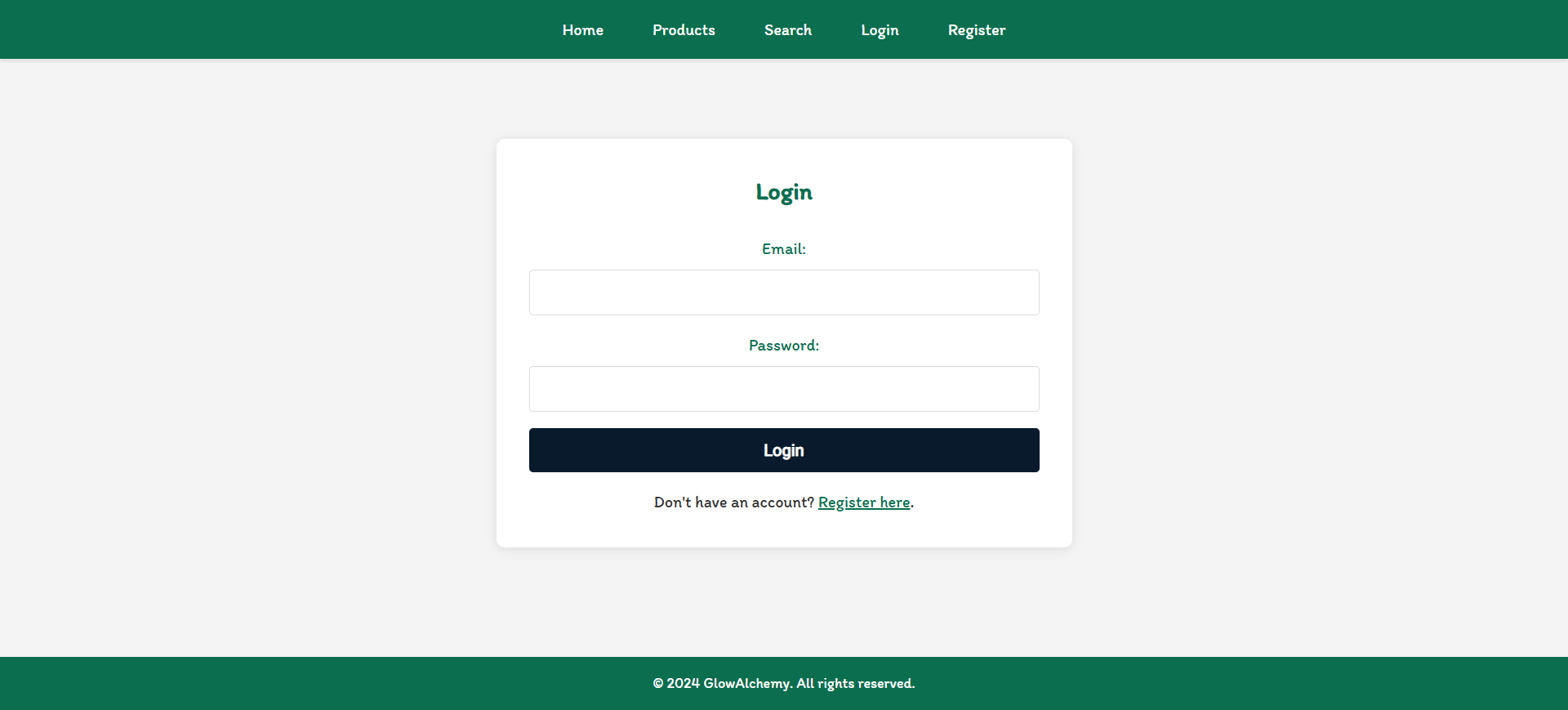
A user who is already logged in should **NOT** be able to reach this page by typing its URL (e.g. "http://localhost:3000/register") in the browser address bar.



#### Login Page (Logged Out User)

Logging an already registered user with the correct **email** and **password**. After successful login, you should **redirect the user to the Home page.** If the user tries to submit **invalid data**, an **error message** should be displayed and the already entered **data should be kept** in the input fields (except for the passwords). The **Register here** button should be redirected to the **Register Page**.

A user who is already logged in should **NOT** be able to reach this page by typing its URL (e.g. "http://localhost:3000/login") in the browser address bar.



#### Logout (Logged in user)

The logout action is available to **logged-in** users. Upon success, clear any session information and **redirect** the user to the **Home** page.

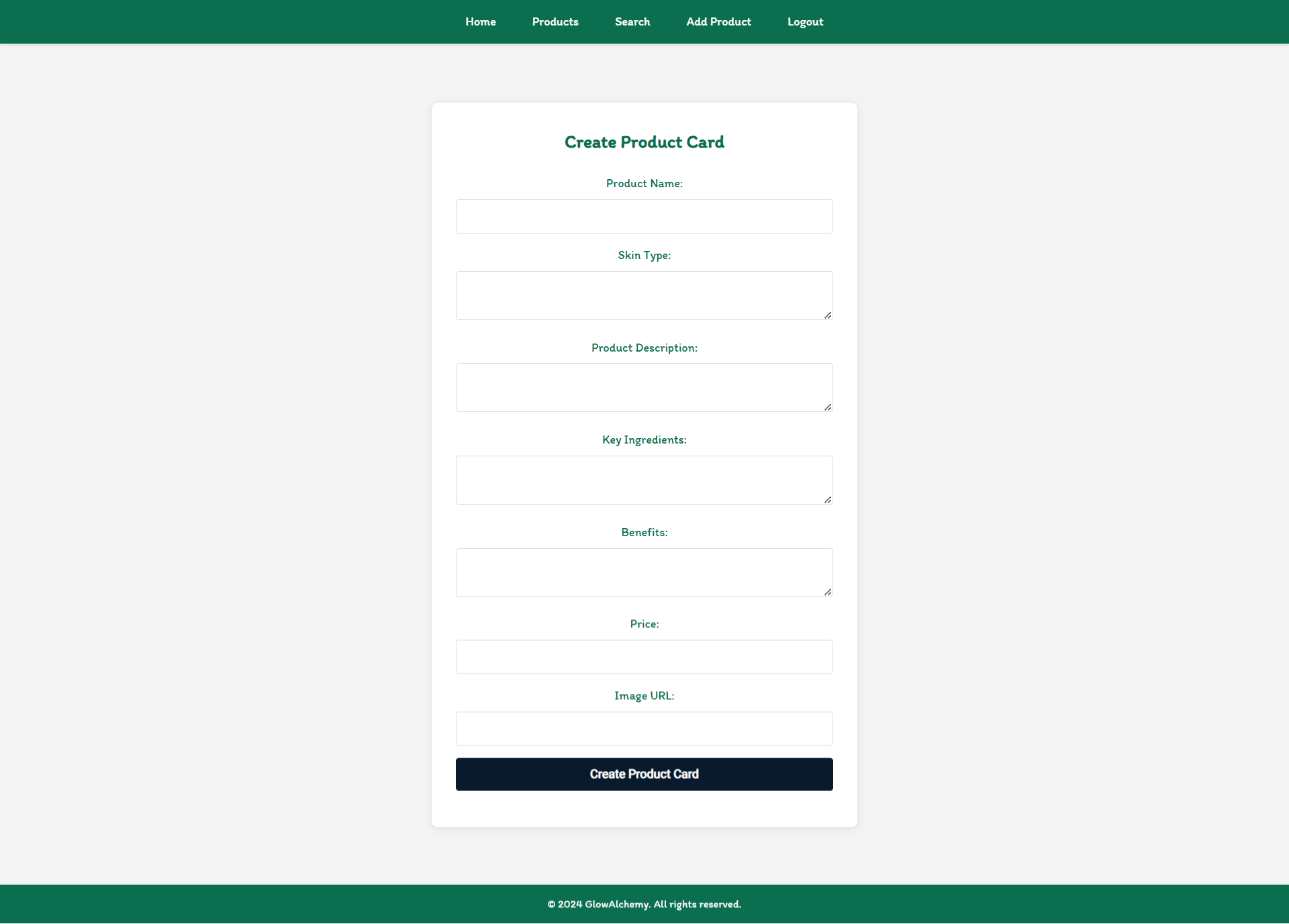
A user who is **NOT** logged in should **NOT** be able to reach this functionality by typing its URL (e.g. "http://localhost:3000/logout") in the browser address bar.

#### Create Product (Logged in User)

The **Create Product** page is available to **logged-in users**. It contains a form for adding new cosmetic post. Upon success, **redirect** the user to the **Products** page. If the user tries to submit **invalid data**, an **error message** should be displayed and the already entered **data should be kept** in the input fields.

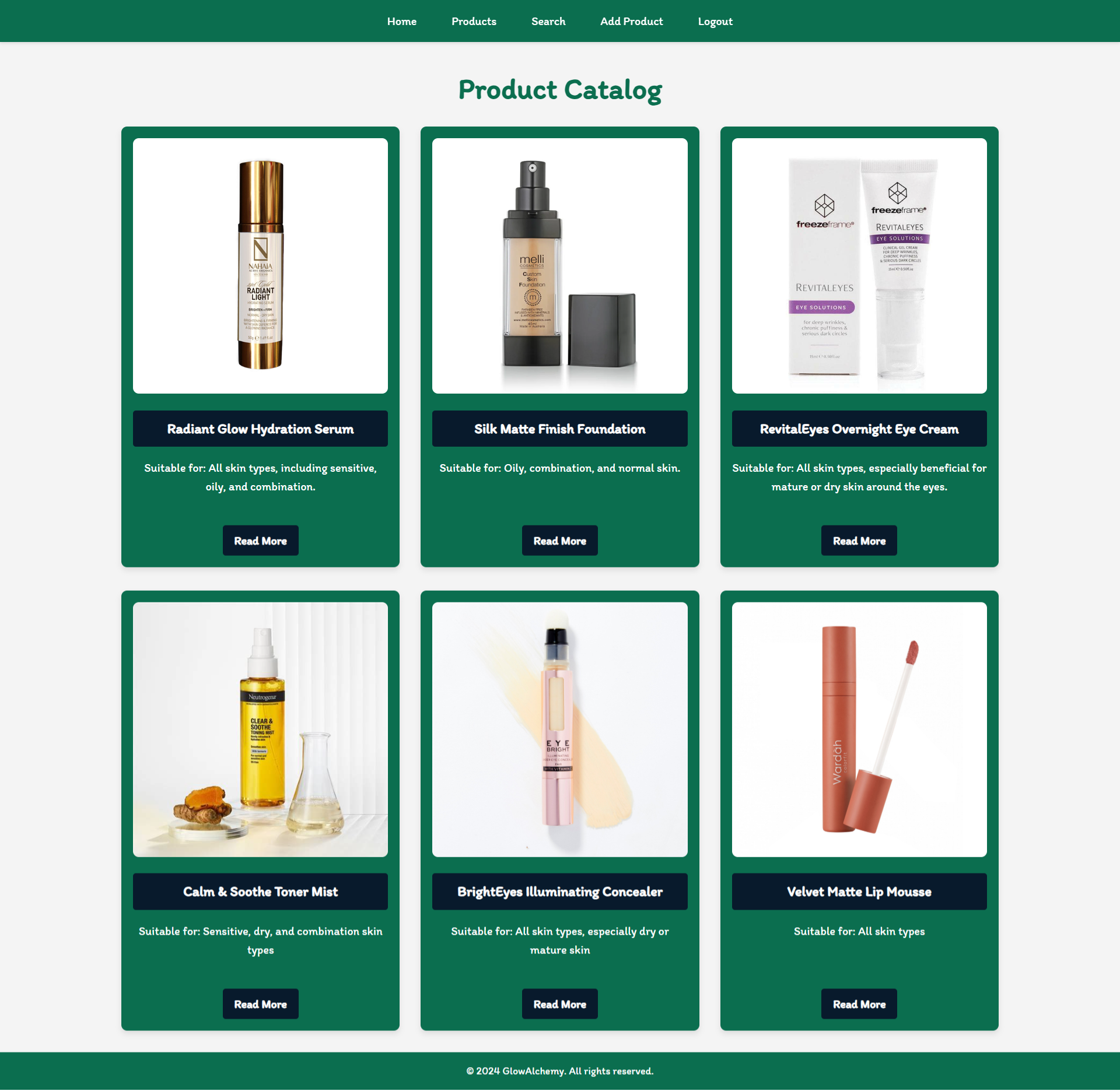
**Ingredient input** should be entered as follows **"{ingredient 1}, { ingredient 2}, {ingredient 3}"**

A guest user (who is not logged in) should **NOT** be able to reach this page by typing its URL (e.g. "http://localhost:3000/cosmetics/create/1234") in the browser address bar.



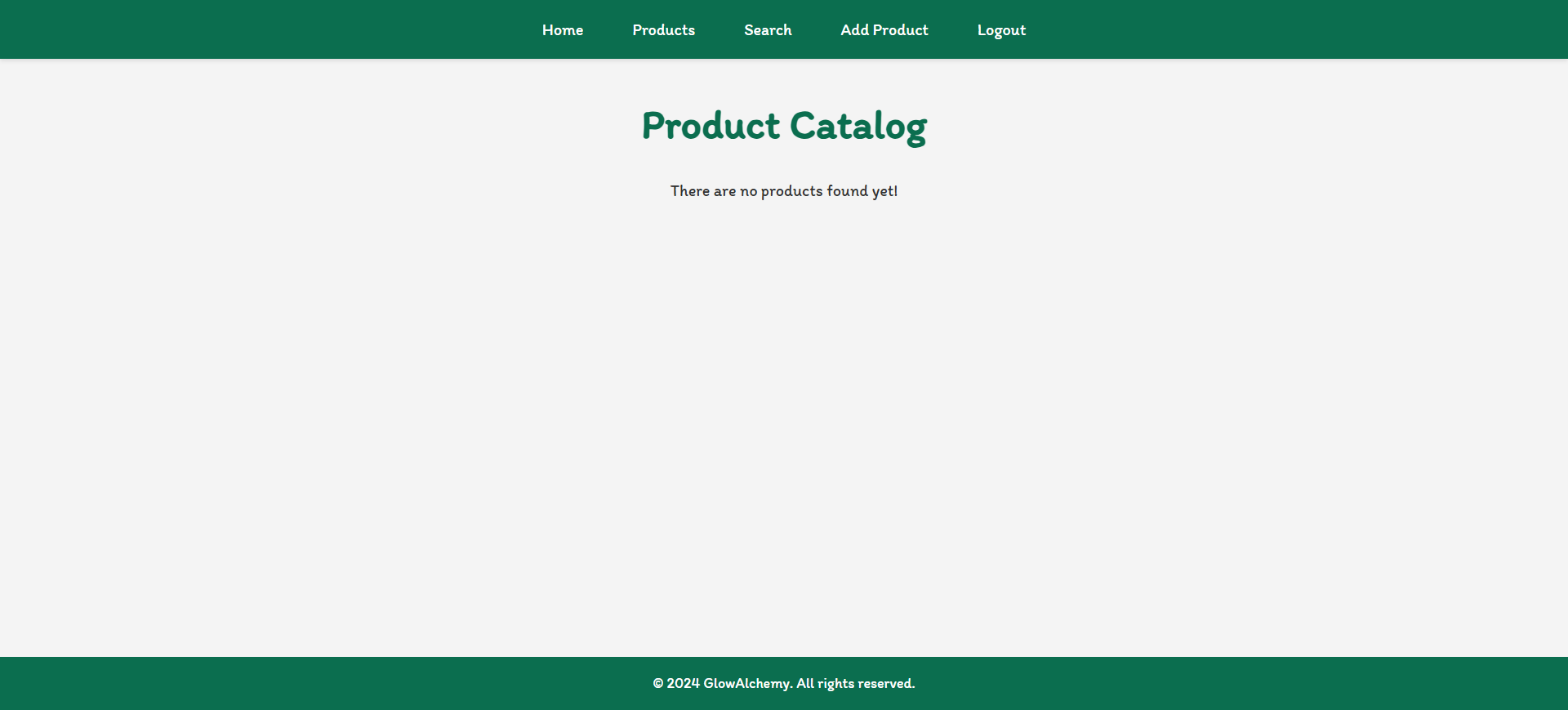
#### Cosmetics (For logged in users and logged out users)

A list of all cosmetics. Each product must be displayed with the **name,** the **skin type,** andthe **image**, and a button for **details** about the **specific post**. As in the picture below:

****

The [**Details**] button should be a link to the **details** **page** for the current product post.

If there are **NO** post in the database yet, display **"There are no products found yet!"**

****

#### Details Page (For logged in users and logged out users)

All users should be able **to see the details about the cosmetics**. Clicking the **Details button** on the post card should display the details page. If the currently registered user **is the creator** of the post, the **Edit** and **Delete** buttons should be displayed. Otherwise, they **should not be available**.

**Ingredient** should be visualize as follows **"{ingredient 1} / { ingredient 2} / {ingredient 3}"**.

Information about the cosmetics:

* **Name**
* **Description**
* **Key Ingredients**
* **Benefits**
* **Skin Type**
* **Price**
* **Recommend**
* **Buttons** (Depending on the status of the currently logged in user)

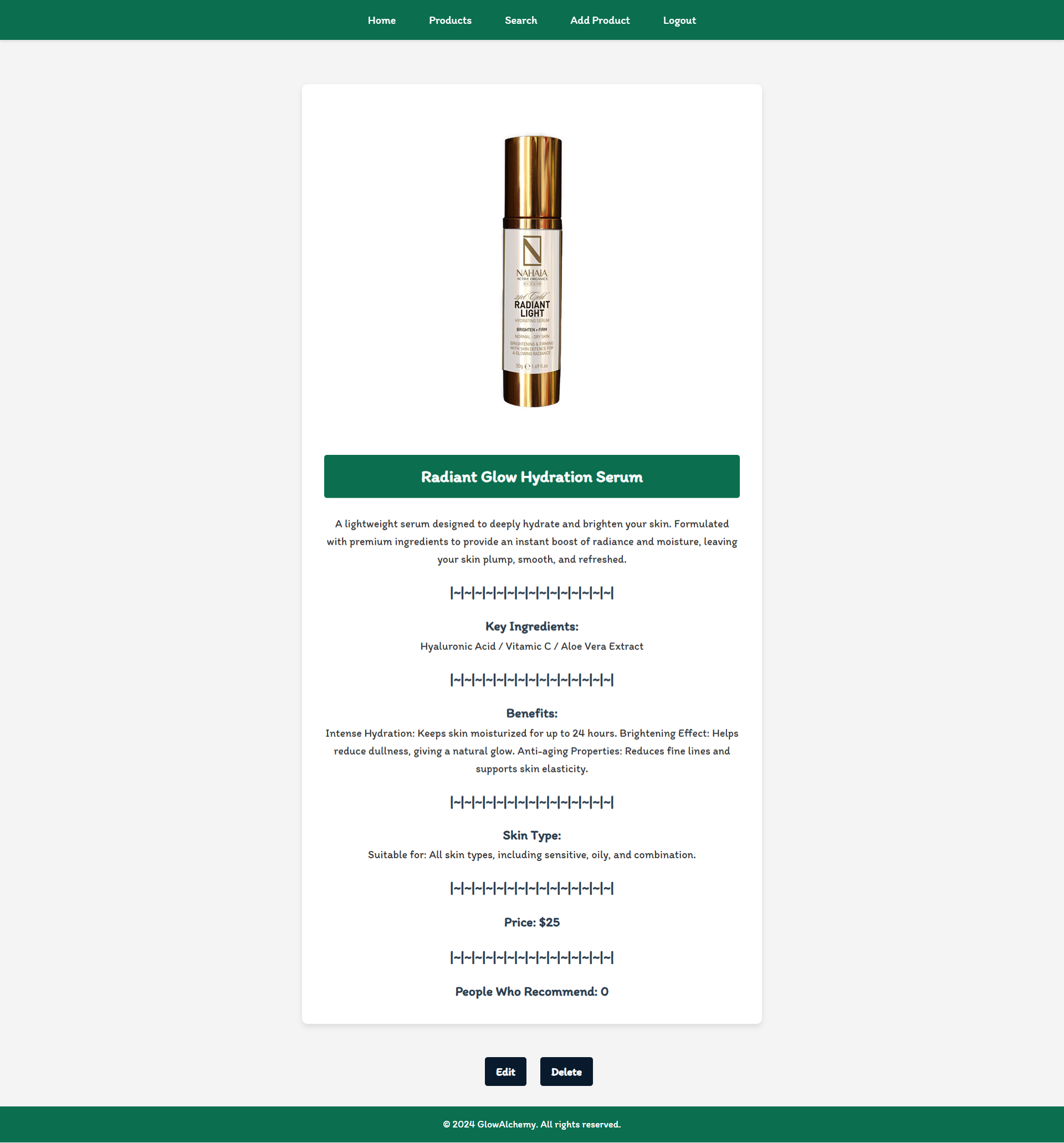
#### Details Page (Logged out users)

If the user has not **logged in**, **no buttons** should be displayed.



#### Details Page (Logged in user and creator of the current post)

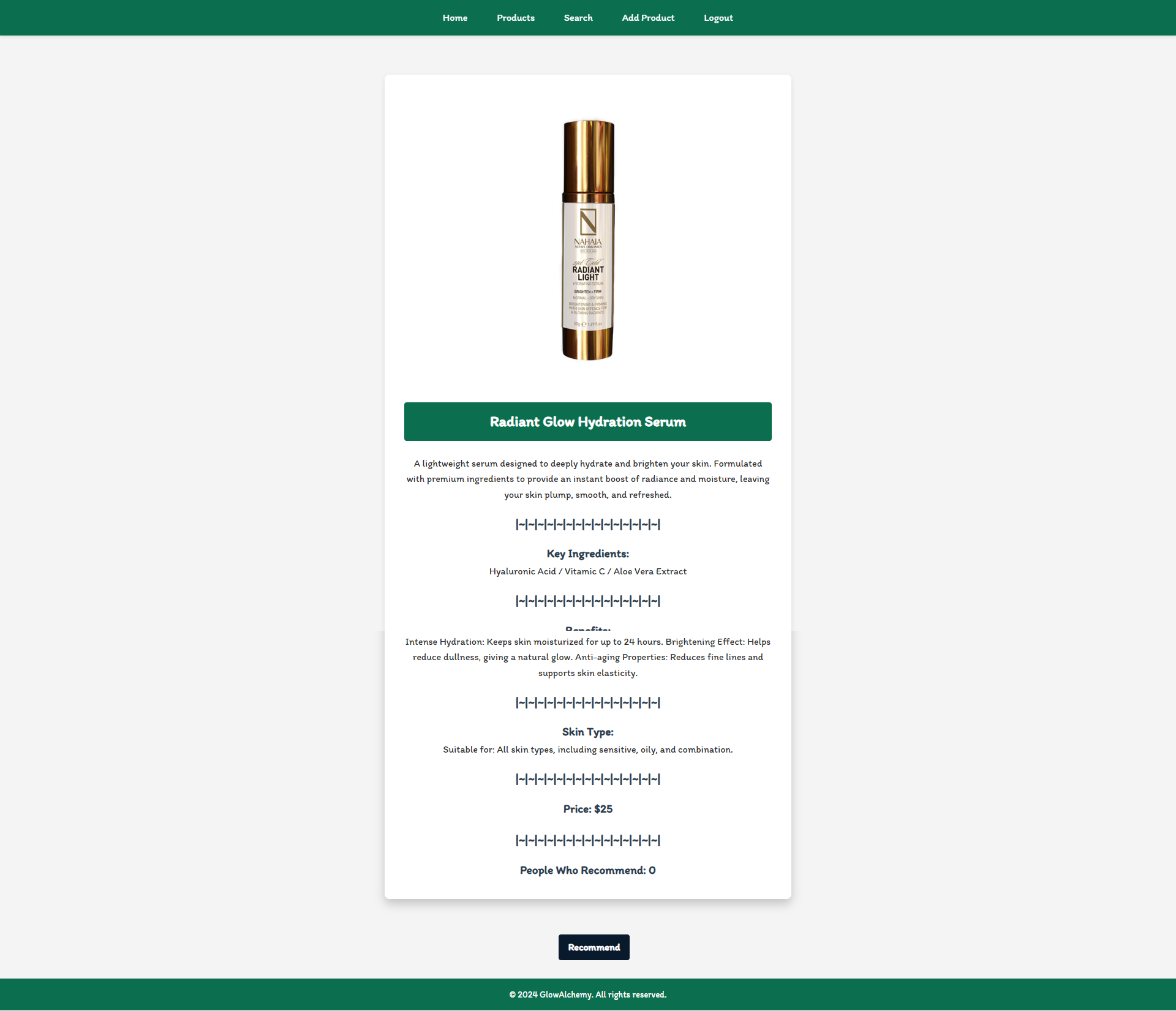
If the **currently logged-in** user is the **owner** (the user who **created the cosmetic post**), he should see the **[Delete]** and **[Edit]** buttons.

****

#### Details Page (Logged in user who has not recommend post)

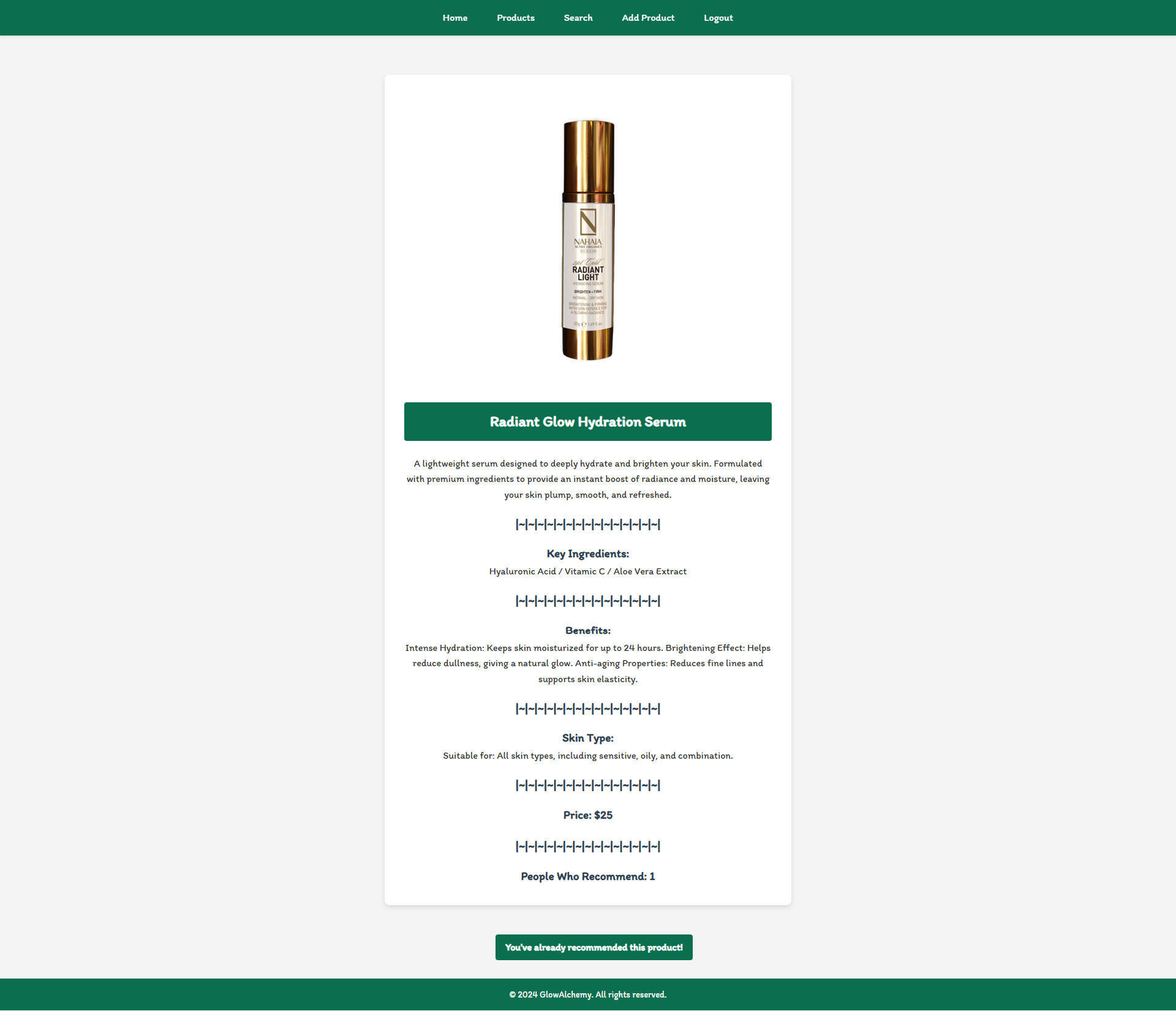
If the currently logged-in user is **not the owner** (a user who is **not** the creator of this cosmetic post) and **he has not recommend the post**, he should see a **Recommend** button. Each **user who is NOT the owner of the current post** must be able to click on the **[Recommend]** button and **recommend current cosmetic**. If **successful**, redirect the user to the **current product post details page**.

A user who is owner of the post should **NOT** be able to reach this functionality by typing its URL (e.g. "http://localhost:3000/cosmetics/recommend/1234") in the browser address bar.

****

#### Details Page (Logged in user already recommend cosmetic post)

If the currently logged in user **is not the owner** and has **already recommend the cosmetic post**, he should see **[You've already recommended this product!]** and **People Who Recommend** must increase by one.

****

#### Delete Cosmetic Post (Logged in user and owner of the current post)

Each **owner** of the product post must be able to click on the **[Delete]** button and **delete the current post** from the database. After this, the user must be redirected to the **Product** page.

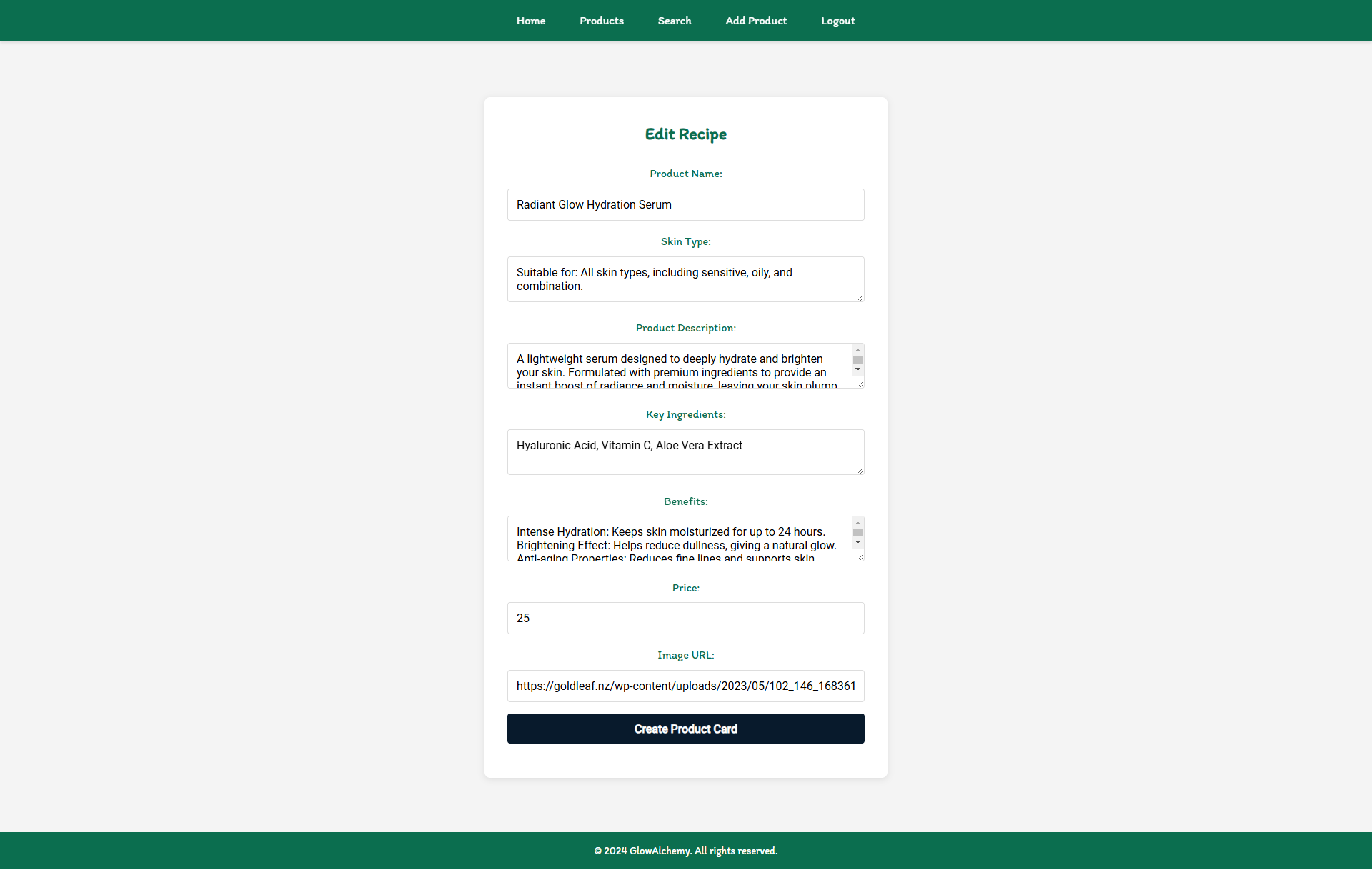
A user who is not the owner of the post should **NOT** be able to reach this functionality by typing its URL (e.g. "http://localhost:3000/cosmetics/delete/1234") in the browser address bar.

#### Edit Product Post (Logged in user and owner of the current post)

Each **owner** can edit their **post**. Clicking the **[Edit]** button for a specific post on the details page should display the **Edit page**, all fields being populated with product data. It contains a form with input fields for all relevant properties. If **successful**, redirect the user to the **current cosmetic post details page**. If the user tries to submit **invalid data**, an **error message** should be displayed and the already entered **data should be kept** in the input fields.

**Ingredient input** should be entered as follows **"{ingredient 1}, { ingredient 2}, {ingredient 3}"**

A user who is not the owner of the posts should **NOT** be able to reach this page by typing its URL (e.g. "http://localhost:3000/cosmetics/edit/1234") in the browser address bar.

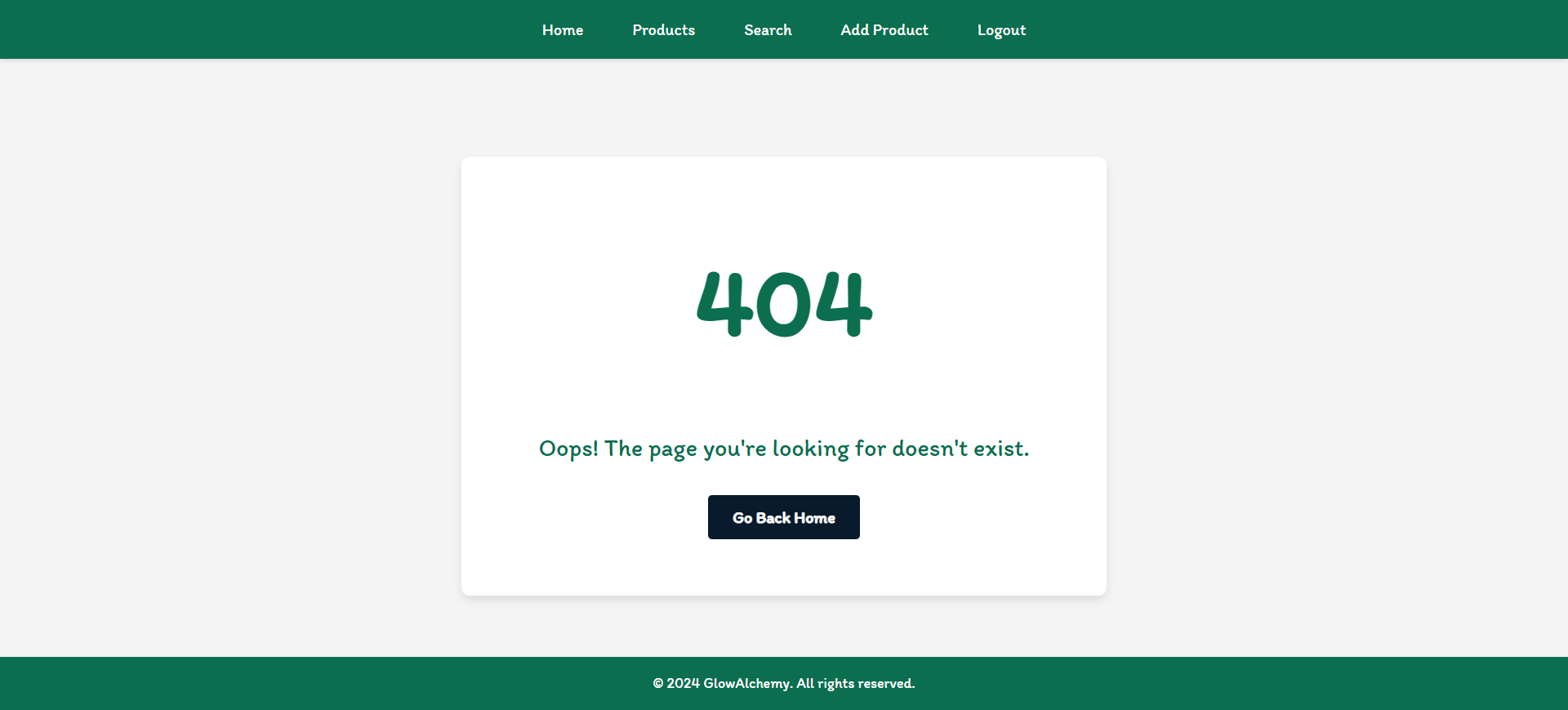


## Security Requirements (Routes Guards)

The Security Requirements are mainly **access** requirements. Configurations about which users can access specific functionalities and pages.

* Guest (not logged in) users can access the Home page.
* Guest (not logged in) users can access the Login page and functionality.
* Guest (not logged in) users can access the Register page and functionality.
* Guest (not logged in) and Users (logged in) can access the **Products** (Listed all cosmetic posts).
* Guest (not logged in) can access the **Details** page without functionality.
* Guest (not logged in) can access the **Search** page and functionality.
* Users (logged in) can access the Home page.
* Users (logged in) can access the Details page and functionality.
  + **Users** (not post owners) can **recommend other's post**.
  + **Users** (post owners) can **Edit** and **Delete** the current post
* Users (logged in) can access the **Search** page and functionality.
* Users (logged in) can access the **Add Product** page and functionality.
* Users (logged in) can access the Logout functionality.

Use the following view for **invalid** **paths**:



The **Go Back Home** button should be redirected to the **Home Page**.

## Validation and Error Handling

### The application should notify the users about the result of their actions.

In case of error, you should display div with class "**error-container**"

### You can choose to display the first error or all of them. You have complete freedom to choose the content of the error message you will display, as long as it is clear what exactly the error is.

### Login / Register

You should make the following validations:

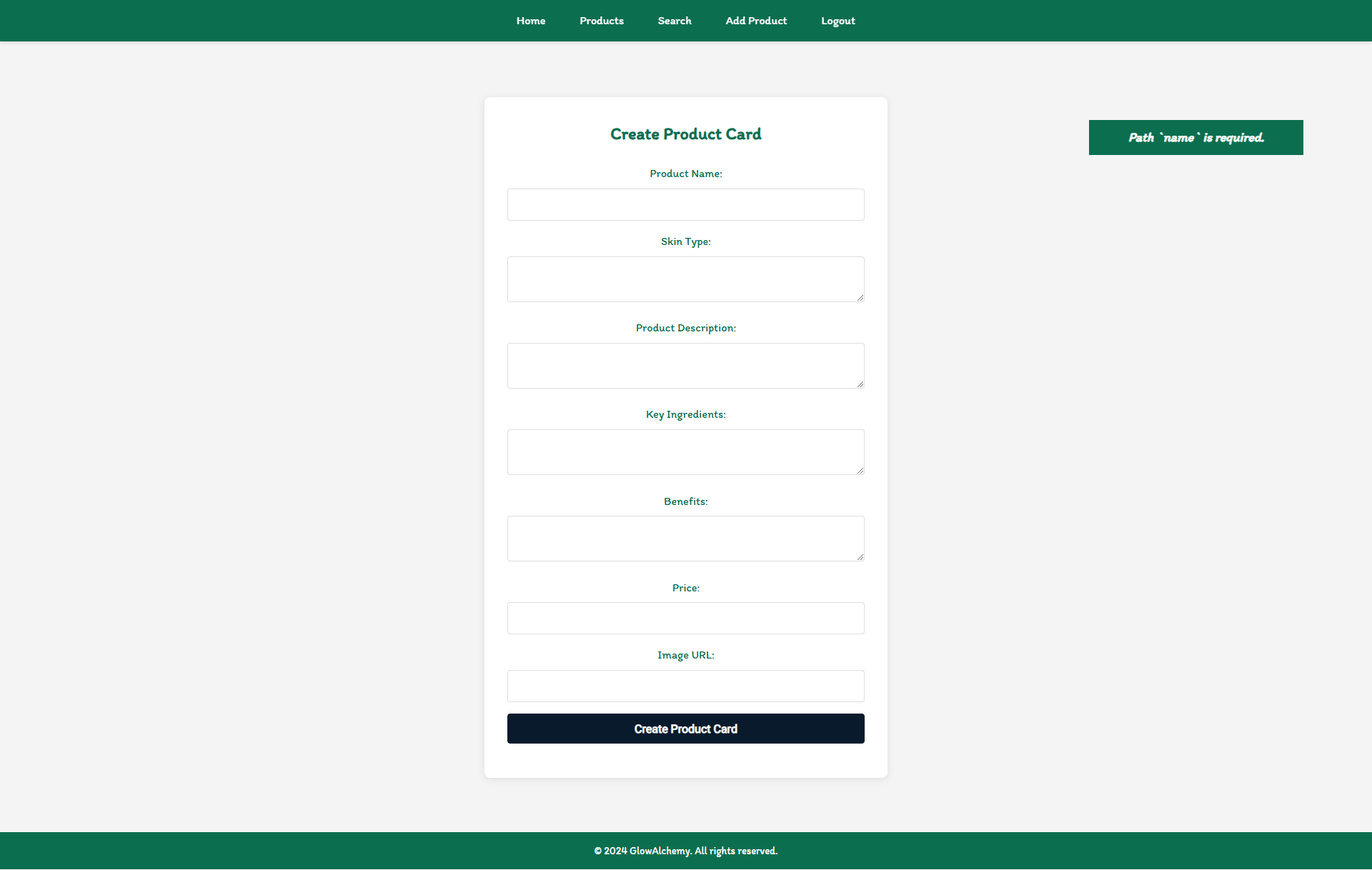
* The **name** should be **between 2 and 20 characters** long
* The **email** should be **at least 10 characters** long
* The **password** should be **at least 4 characters** long
* The **repeat password** should be **equal to the password**



### Cosmetics

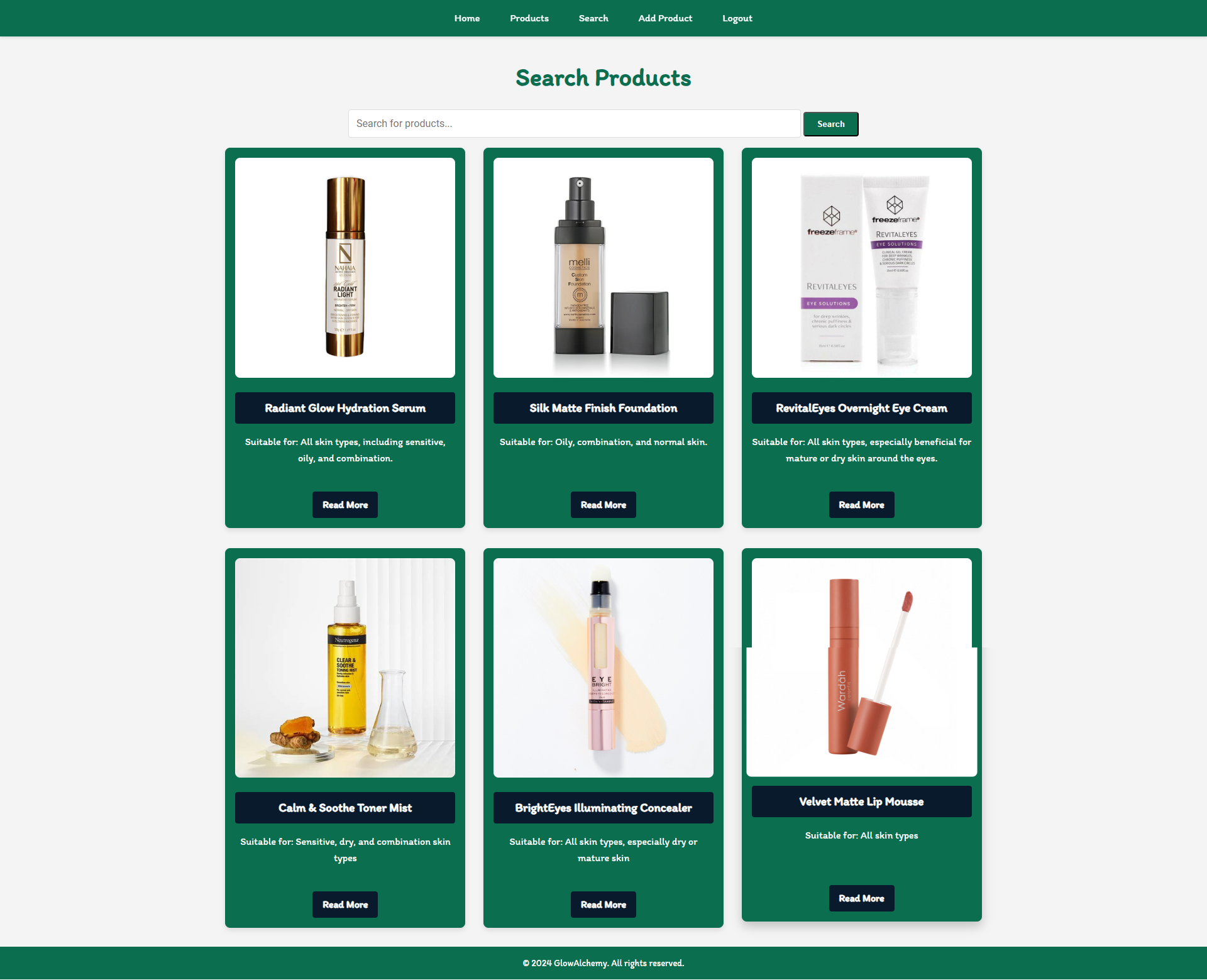
You should make the following validations while **creating** or **editing a** **cosmetic post**:

* The **Name** should be **at least 2 characters**
* The **Skin** should be **between** **10 and 100 characters** **long**
* The **Description** should be **between** **20 and 200 characters** **long**
* The **Ingredients** should be **between** **2 and 50 characters** **long**
* The **Benefits** should be **between** **10 and 100 characters** **long**
* The **Price** should be **a positive number**
* The **Image** should start with **http:// or https://**

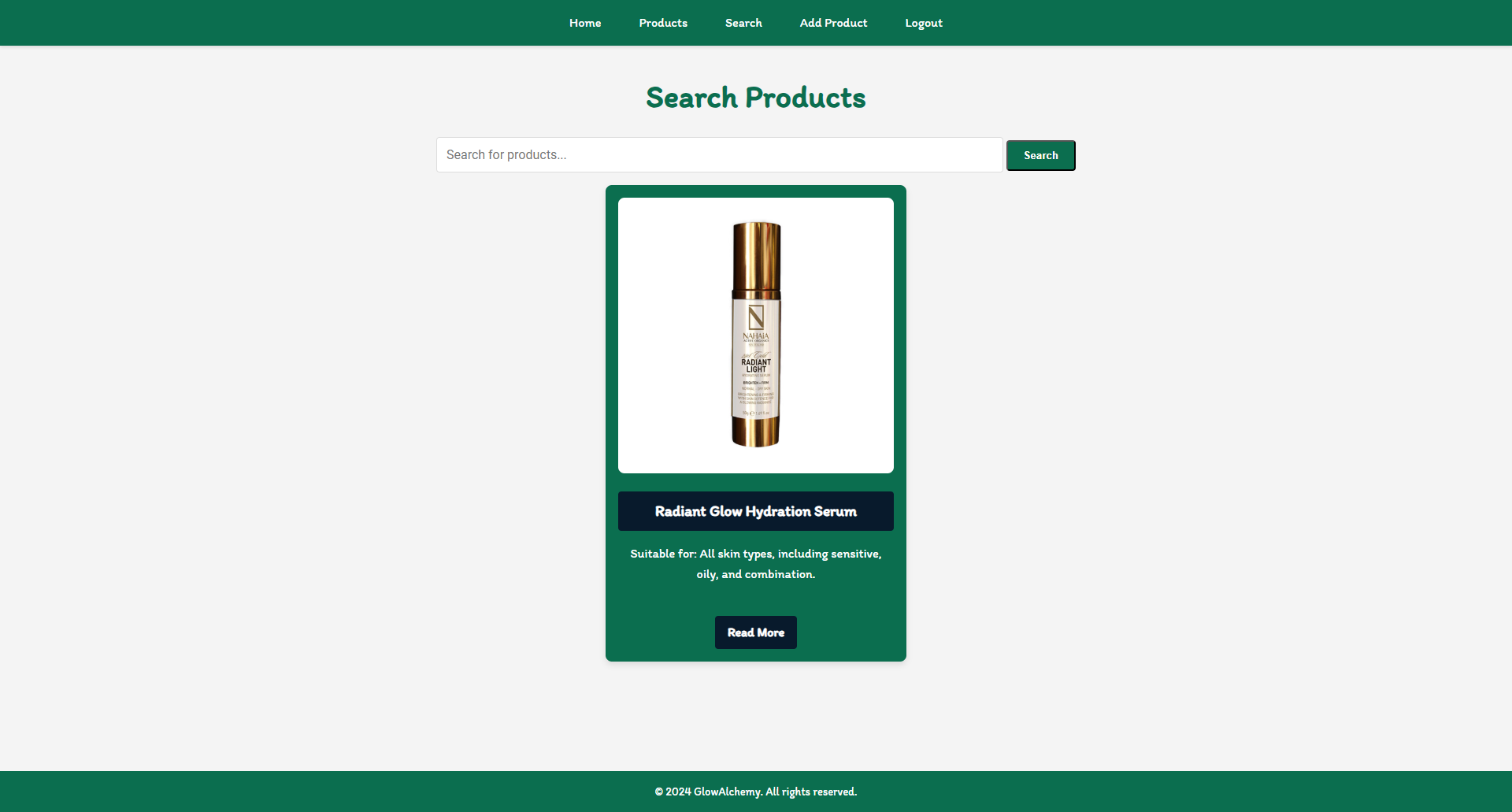


## \* Bonus – Search

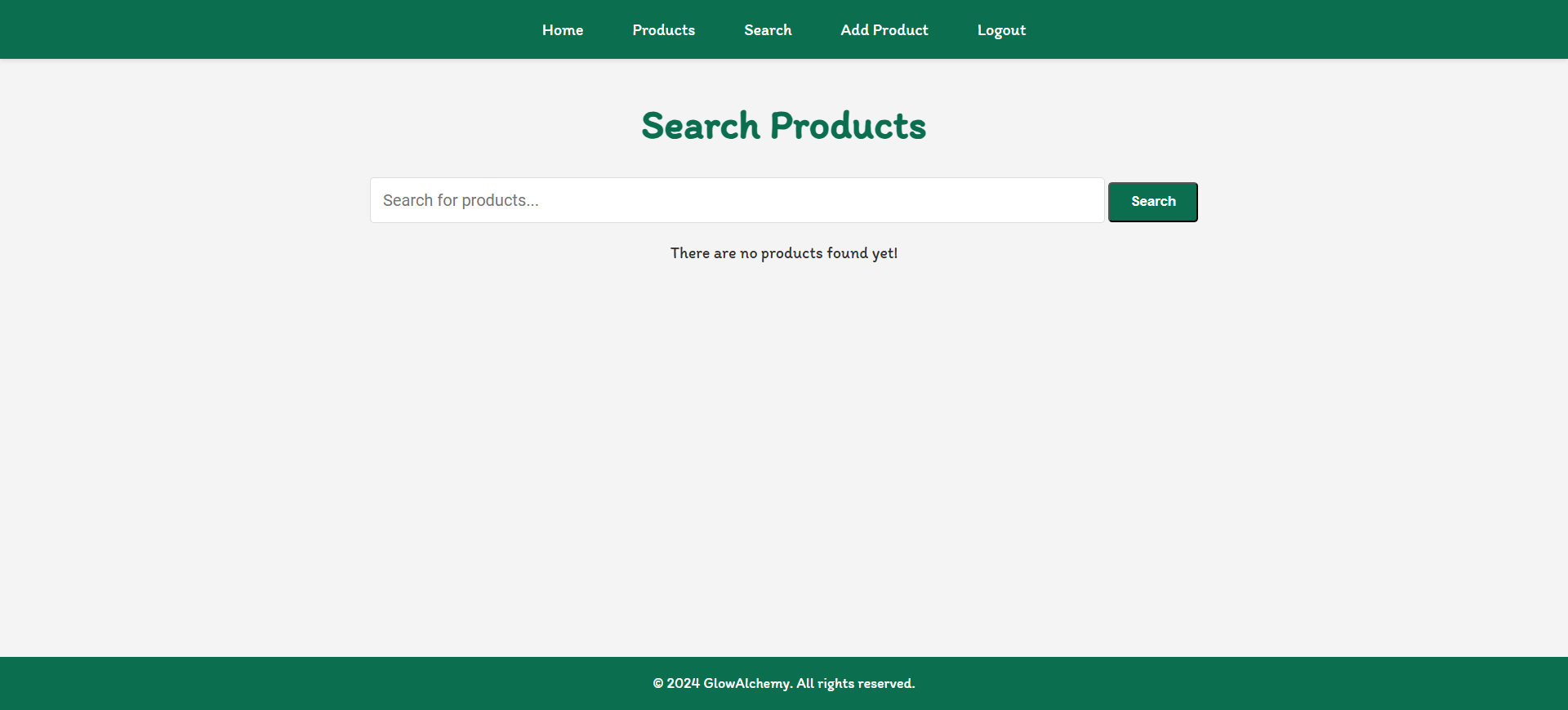
Search **for products**. Filter all **matches** by **name**, which includes a **search string** (case insensitive). When the search is first open, all cosmetics should be displayed.



After clicking on the **[Search]** button If there are any matching matches from the search, show each of them.



If there are **no** search matches, display:



*GOOD LUCK! 😊*