

# Web Studio: Week 11 Turn-in #10

Name: Ivy Chen

UNI: ic2389

Due: Friday at 2PM

COMS6998 Adv. Web Studio

## Original Goals

### High Level Goal

1. Implement the product search, add, and viewing interface

### Low Level Goal

1. Create user's product page template
2. Add a search bar for all products
3. Create buttons to add and remove products from a user's shelf
4. Update backend based on buttons to add and remove products from shelf
5. Create individual product page
6. Scrape another batch of data
7. Create update button to update owned product quantity

## Lessons Learned Through Iteration

1. My plan was to implement routines this week but I realized that I still needed to build the infrastructure to support just adding products to a user's shelf first. I originally created schemas that would allow users to add products to a list, but realized that I needed a way to store information about a user and their owned products.

I ended up solving my problem by creating actions that allow a user to Add/Update Quantity/Remove a product from their shelf, as well as dedicated product pages and a "My Shelf" page that displayed all products owned by a user.

My Shelf



KLAIRS

Freshly Juiced  
Vitamin C Serum



Josie Maran

100 percent Pure  
Argan Oil

User's Shelf

Drunk Elephant

B-Hydra™ Intensive Hydration Serum

NOT OWNED

None



Quantity

1

Add to Shelf

Add product to shelf

Drunk Elephant

## B-Hydra™ Intensive Hydration Serum

Your Shelf Quantity: 1

None



Quantity

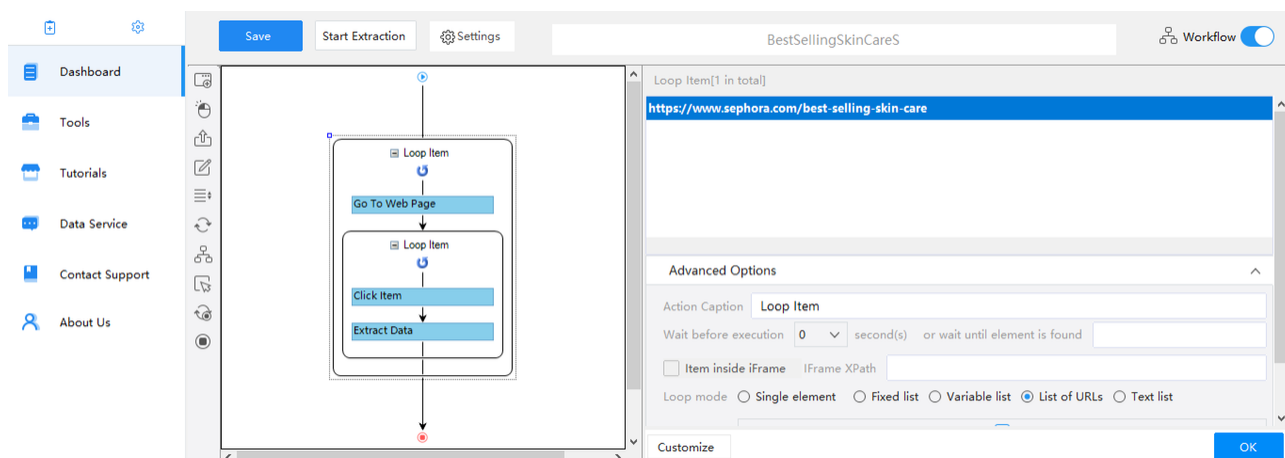
1

Update Quantity

Remove from Shelf

*Update or Remove after added*

- One of my original problems was finding data that I could use to populate my product database. I couldn't find an existing dataset or API that worked for my needs so I ended up scraping an initial set of data rows using Octoparse. However, there is still the case that a user's product is not in the database. I decided to resolve the problem by allowing users to search for products on the site that they can add, or use a form to add their own products to a persistent database.



*Setting up an Octoparse data scrape workflow*

## Add Product

Brand

Name

Image URL

Description

Add Product

### *Form to Add Product*

- One of my problems this week was trying to implement steps X, Y, Z without having implemented dependencies A, B, C. I think this was due to setting my riskiest feature to be steps ahead of what I had already built. For example, I was reading up on Ajax requests in Flask, and how to communicate between client and server in Flask to create some smoother user interactions, but ended needing to backtrack to build basic functionality first.

Going forward, I'll keep in mind a reasonable timeline to build out the features I need as well as the dependencies/pre-requisites needed before my next big milestone.

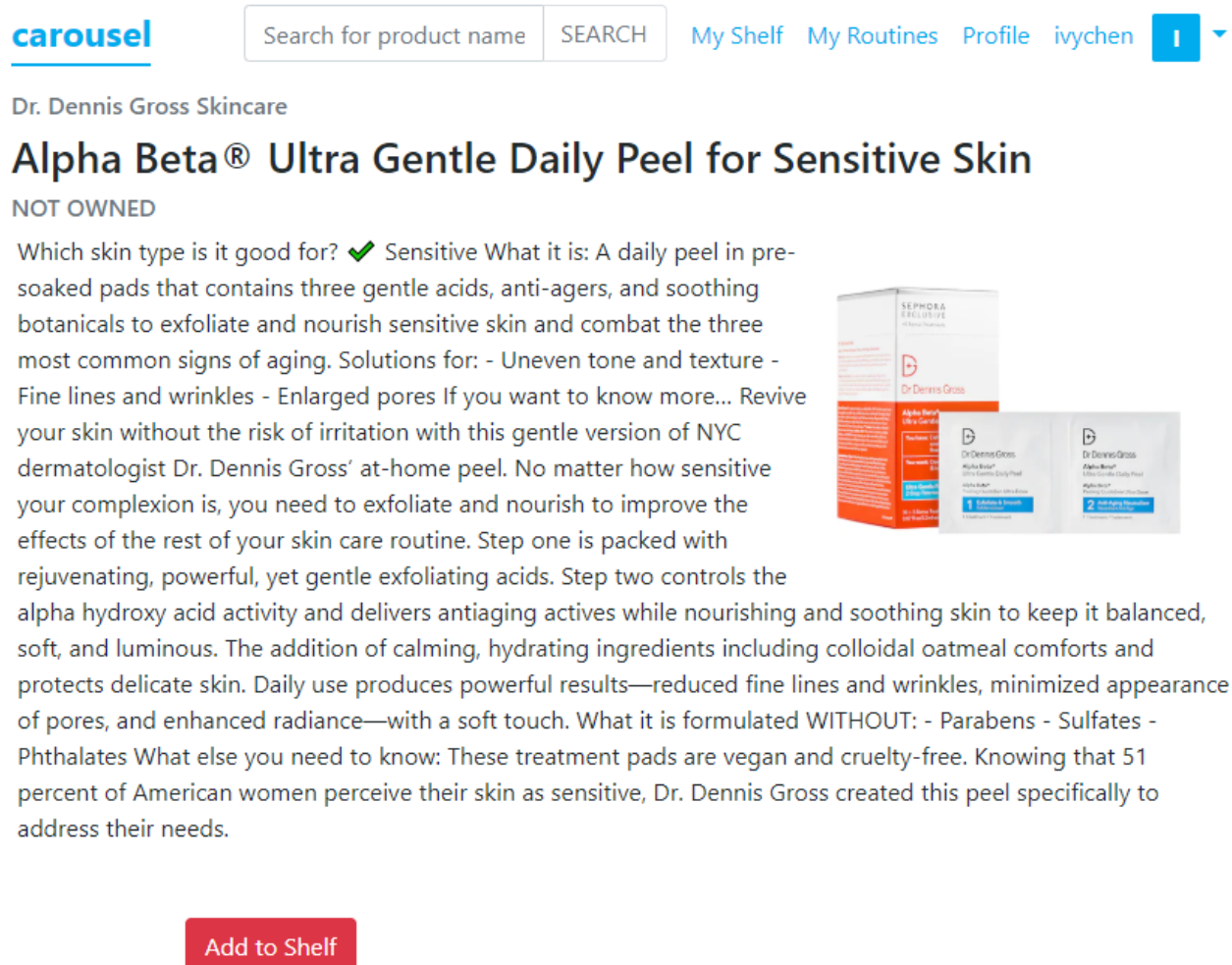
## Goal Progress

### High Level Goal

- Implement the product search, add, and viewing interface

## Low Level Goal

1. Create user's product page template → created individual product pages



2. Add a search bar for all products → added search functionality that performs fuzzy search on brand and product name

## Search Results: good



COSRX

Low-pH Good  
Morning Cleanser



SUNDAY RILEY

Good Genes All-  
In-One Lactic Acid  
Treatment



Goodal

Anti-Wrinkle  
Want Night Sleep  
Cream Pack

3. Create buttons to add and remove products from a user's shelf → created buttons

Drunk Elephant

## B-Hydra™ Intensive Hydration Serum

NOT OWNED

None



Quantity

1

Add to Shelf

Add product to shelf

Drunk Elephant

## B-Hydra™ Intensive Hydration Serum

Your Shelf Quantity: 1

None



Quantity

1

Update Quantity

Remove from Shelf

*Update or Remove after added*

4. Update backend based on buttons to add and remove products from shelf

```

67 @app.route("/shelf/add", methods=['POST'])
68 def add_to_shelf():
69     if request.method == "POST":
70         statement = models.owns.insert().values(username=current_user.username,
71     •         productId=request.form['id'], quantity=request.form['quantity'])
72
73         db.session.execute(statement)
74         db.session.commit()
75
76     return redirect(request.referrer or url_for('main'))
77
78 @app.route("/shelf/update", methods=['POST'])
79 def update_to_shelf():
80     statement = models.owns.insert().values(username=current_user.username,
81     •         productId=request.form['product'], quantity=request.form['quantity'])
82
83     db.session.execute(statement)
84     db.session.commit()
85
86     return redirect(request.referrer or url_for('main'))
87
88 @app.route("/shelf/remove/<product_id>", methods=['POST'])
89 def remove_from_shelf(item_id):
90     statement = models.owns.delete(models.owns.productId==item_id,
91     •         username=current_user.username)
92
93     return redirect(request.referrer or url_for('main'))

```

## 5. Create individual's shelf



## Search Results: good



COSRX

Low-pH Good  
Morning Cleanser



SUNDAY RILEY

Good Genes All-  
In-One Lactic Acid  
Treatment



Goodal

Anti-Wrinkle  
Want Night Sleep  
Cream Pack

### 6. Scrape another batch of data

The screenshot shows the Octoparse web scraper interface. On the left is a sidebar with navigation options: Dashboard, Tools, Tutorials, Data Service, Contact Support, and About Us. The main workspace displays a workflow diagram with the following steps: 'Go To Web Page' (pointing to 'https://www.sephora.com/best-selling-skin-care'), followed by a 'Loop Item' block containing 'Click Item' and 'Extract Data'. The right panel shows the 'Advanced Options' for the 'Loop Item', including 'Wait before execution' (0 seconds), 'Item inside iframe' (unchecked), 'Iframe XPath' (empty), 'Loop mode' (List of URLs selected), and 'Customize' and 'OK' buttons.

*Setting up an Octoparse data scrape workflow*

### 7. Create form to add product

## Add Product

Brand

Name

Image URL

Description

Add Product

*Form to Add Product*