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.

carousel Search for product name SEARCH My Shelf My Routines Profile ivychen I

My Shelf

KLAIRS

Freshly Juiced Vitamin C Serum

Josie Maran

100 percent Pure Argan Oil

User's Shelf

carousel Search for product name SEARCH My Shelf My Routines Profile ivychen I

Drunk Elephant

B-Hydra™ Intensive Hydration Serum

NOT OWNED

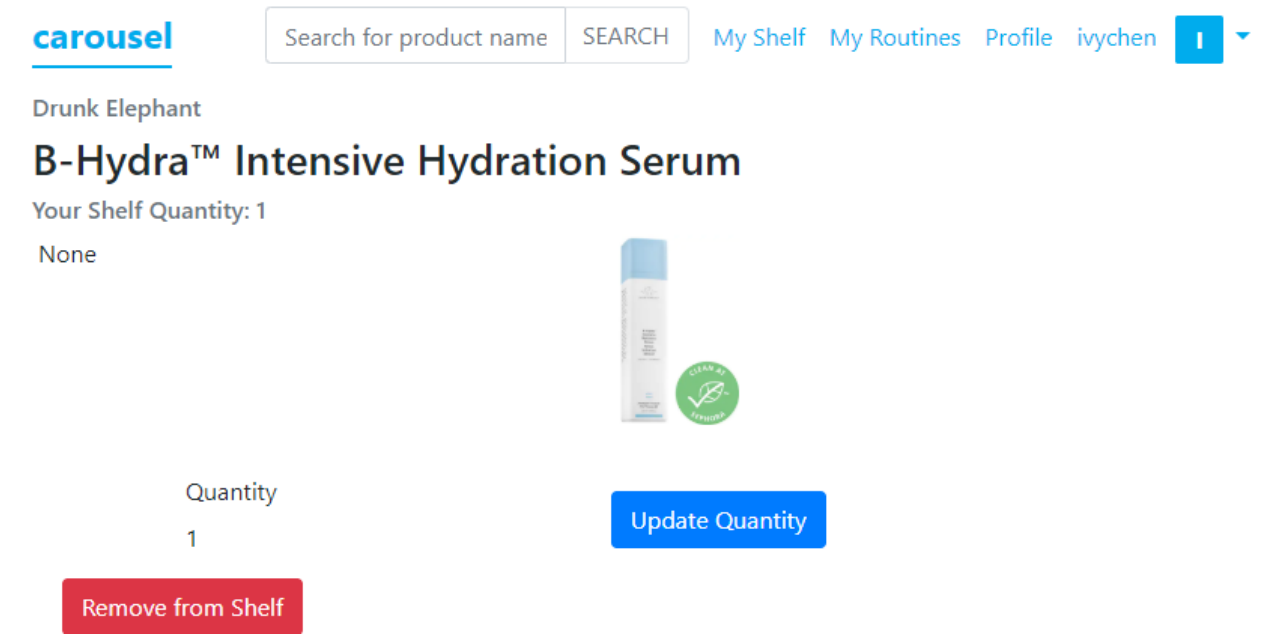
None

Quantity

1

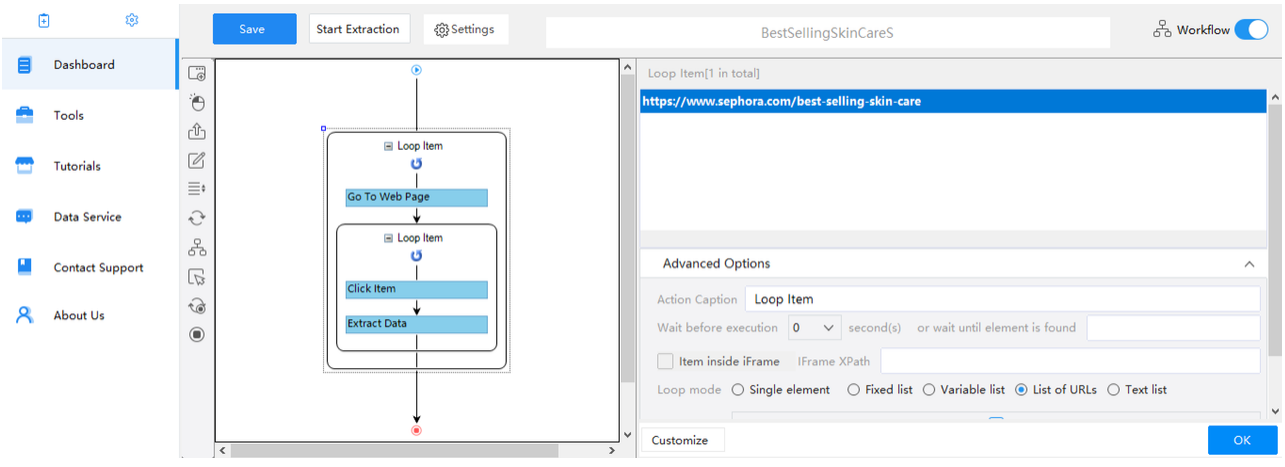
Add to Shelf

Add product to shelf

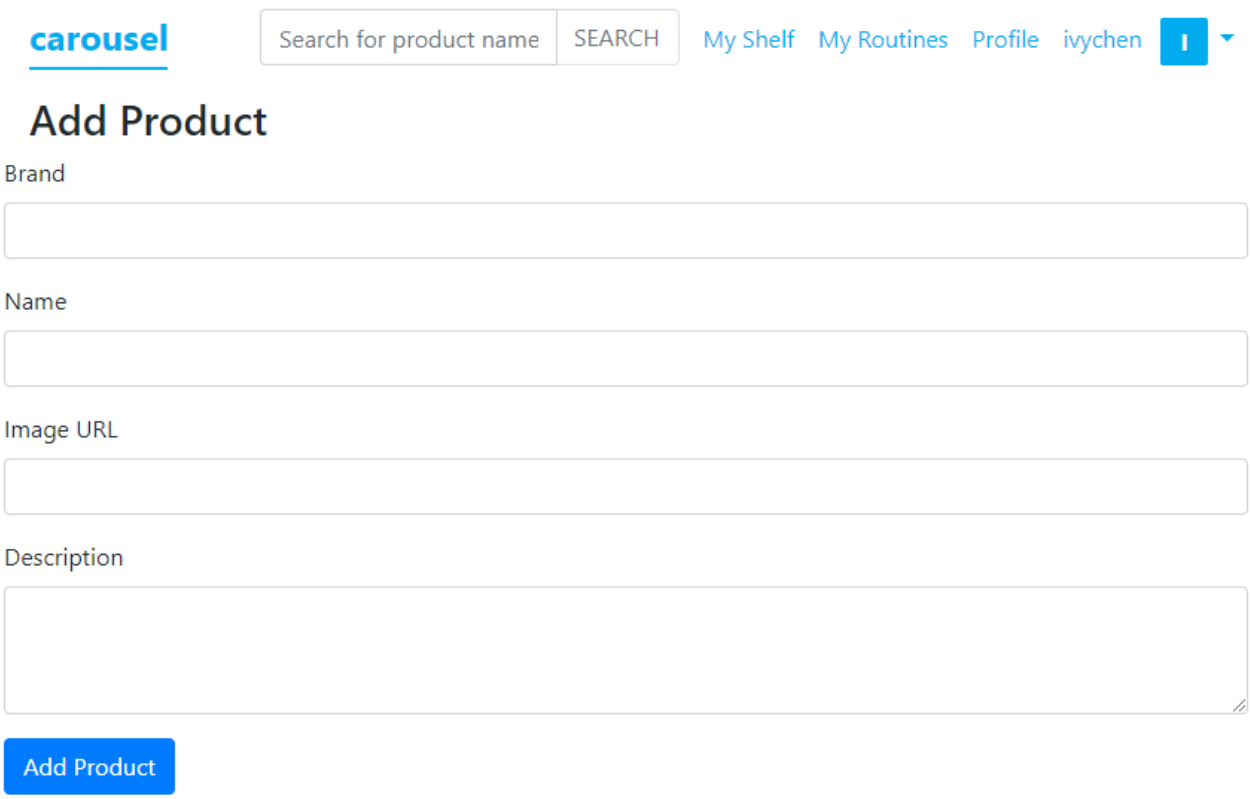


Update or Remove after added

2. 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



Form to Add Product

3. 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.

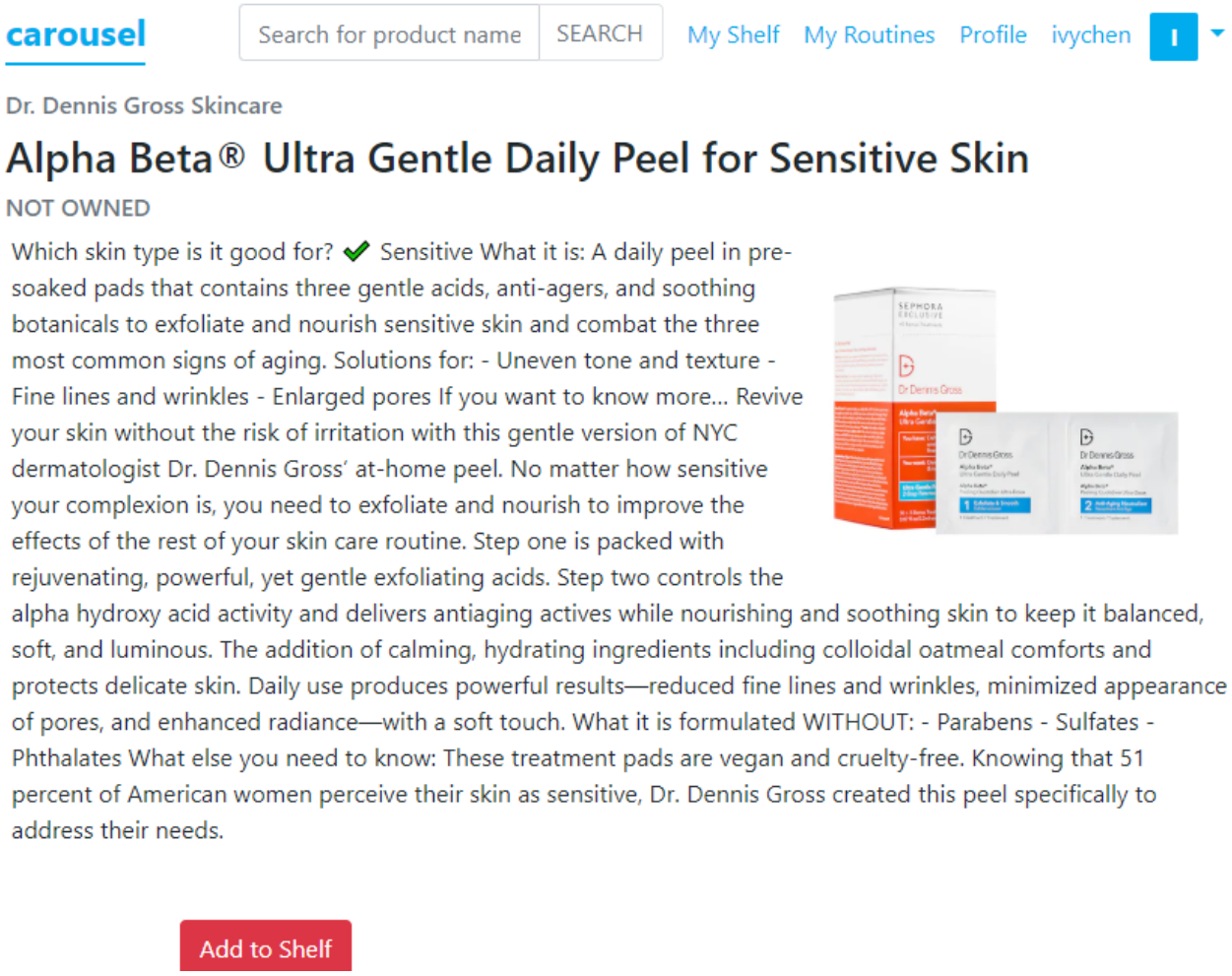
Goal Progress

High Level Goal

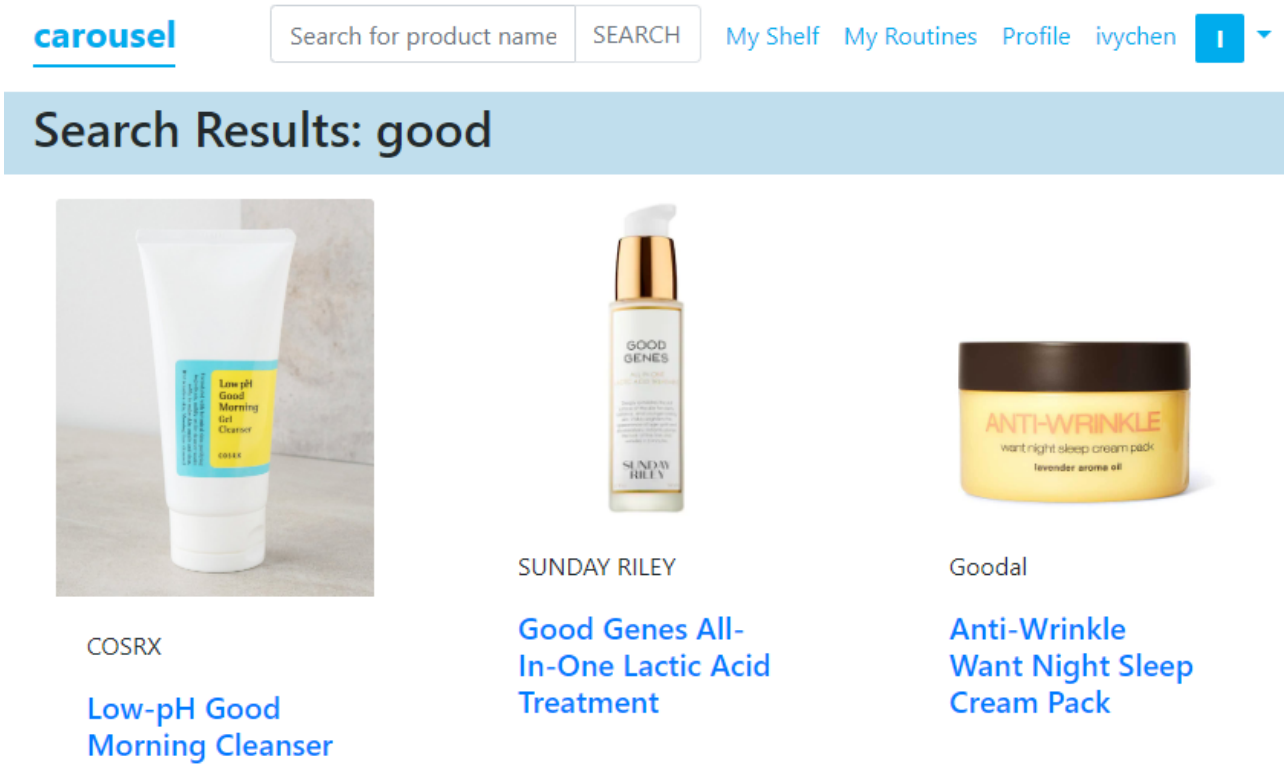
1. Implement the product search, add, and viewing interface

Low Level Goal

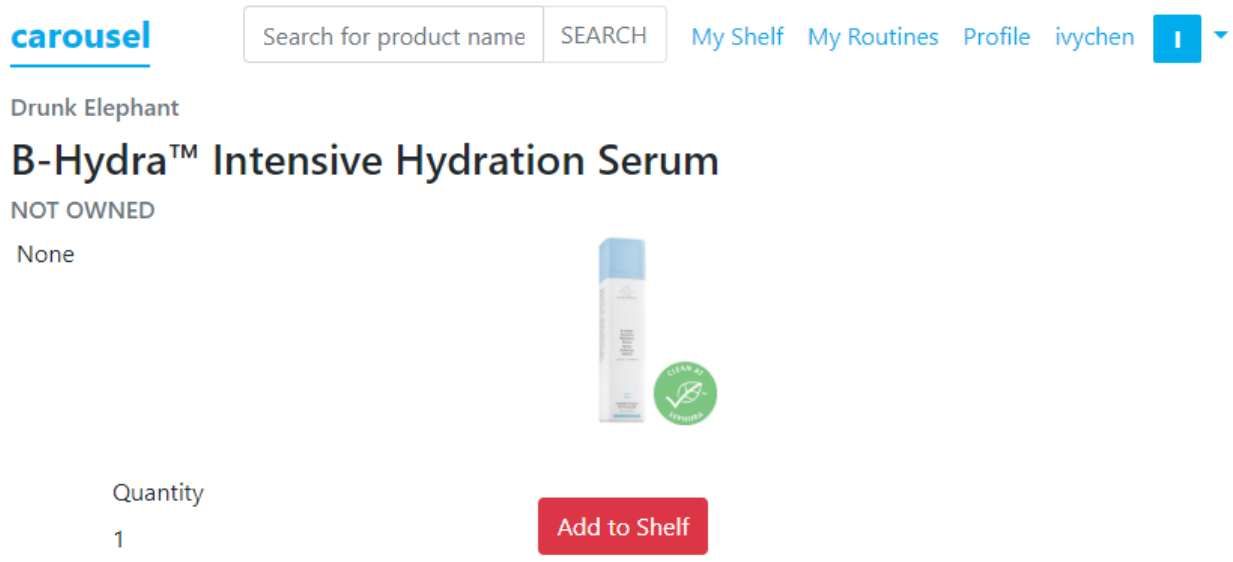
1. Create user's product page template → created individual product pages



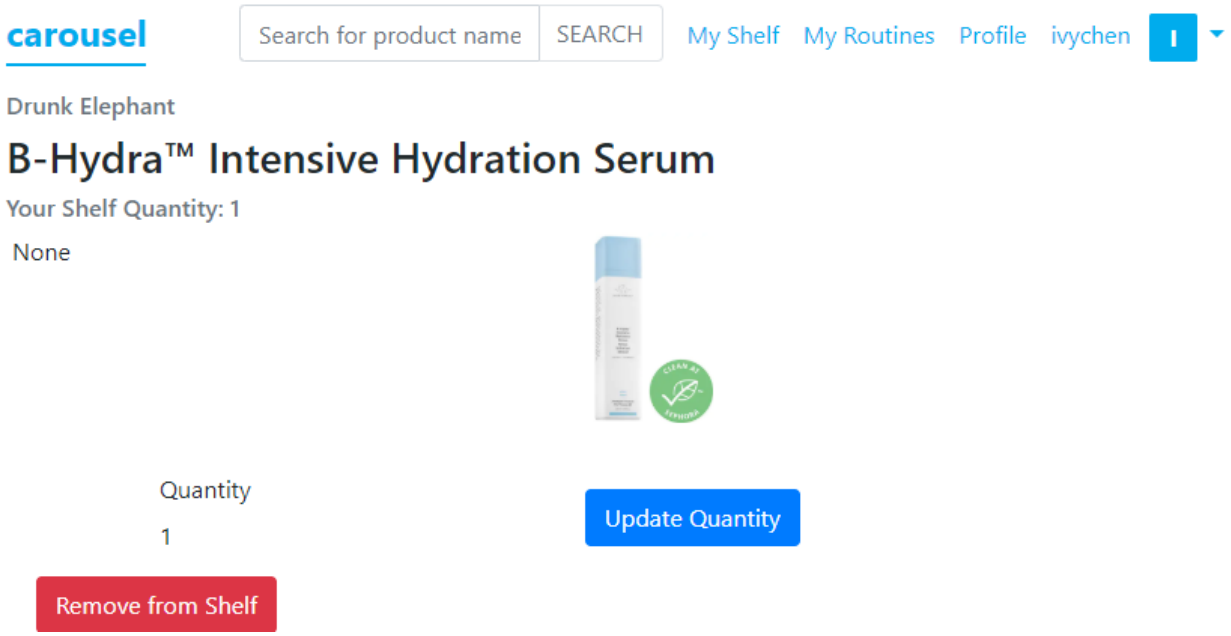
2. Add a search bar for all products → added search functionality that performs fuzzy search on brand and product name



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



Add product to shelf

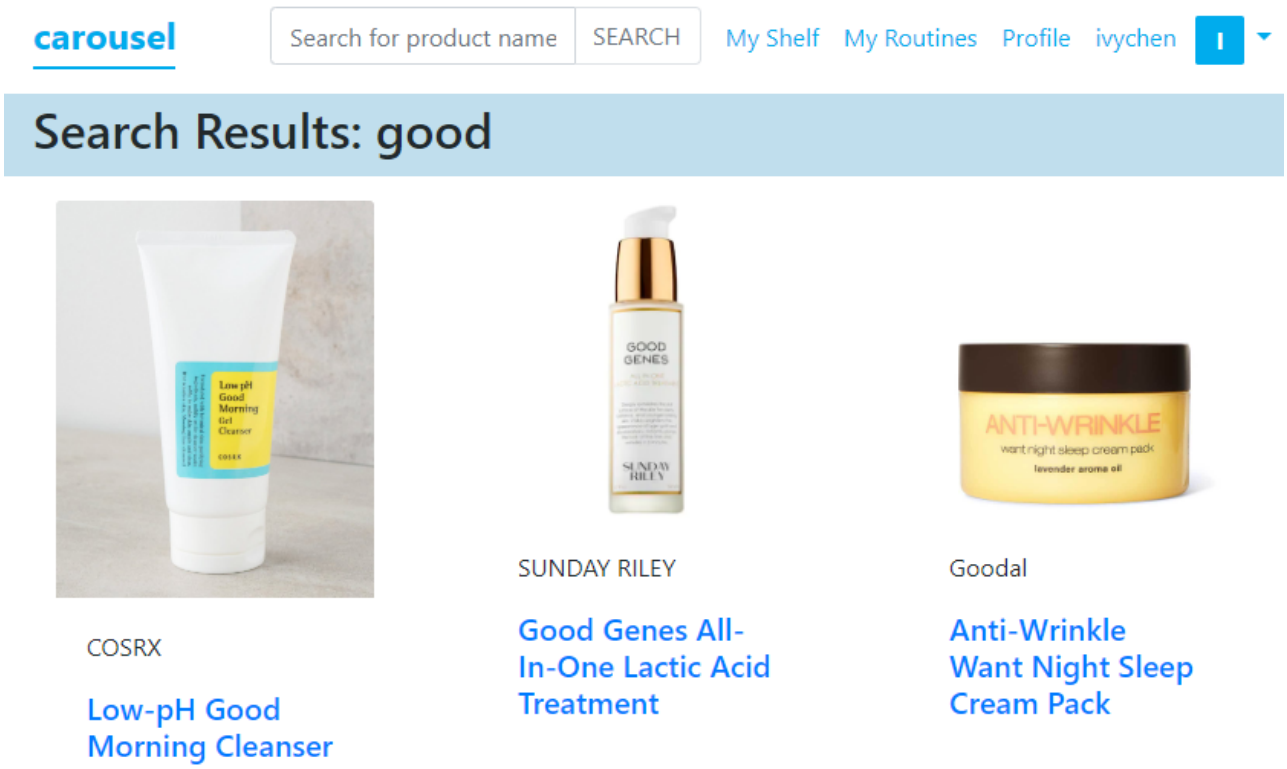


Update or Remove after added

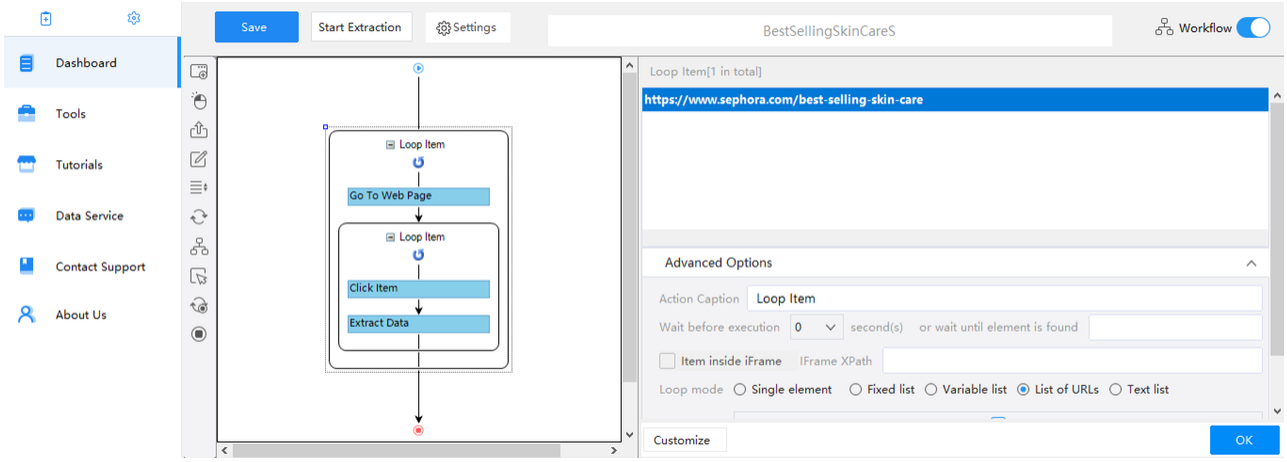
4. Update backend based on buttons to add and remove products from shelf

```python
@app.route("/shelf/add", methods=['POST'])
def add_to_shelf():
    if request.method == "POST":
        statement = models.owns.insert().values(username=current_user.username,
        productId=request.form['id'], quantity=request.form['quantity'])

        db.session.execute(statement)
        db.session.commit()

    return redirect(request.referrer or url_for('main'))

@app.route("/shelf/update", methods=['POST'])
def update_to_shelf():
    statement = models.owns.insert().values(username=current_user.username,
    productId=request.form['product'], quantity=request.form['quantity'])

    db.session.execute(statement)
    db.session.commit()

    return redirect(request.referrer or url_for('main'))

@app.route("/shelf/remove/<product_id>", methods=['POST'])
def remove_from_shelf(item_id):
    statement = models.owns.delete(models.owns.productId==item_id,
    username=current_user.username)

    return redirect(request.referrer or url_for('main'))
```

5. Create individual's shelf

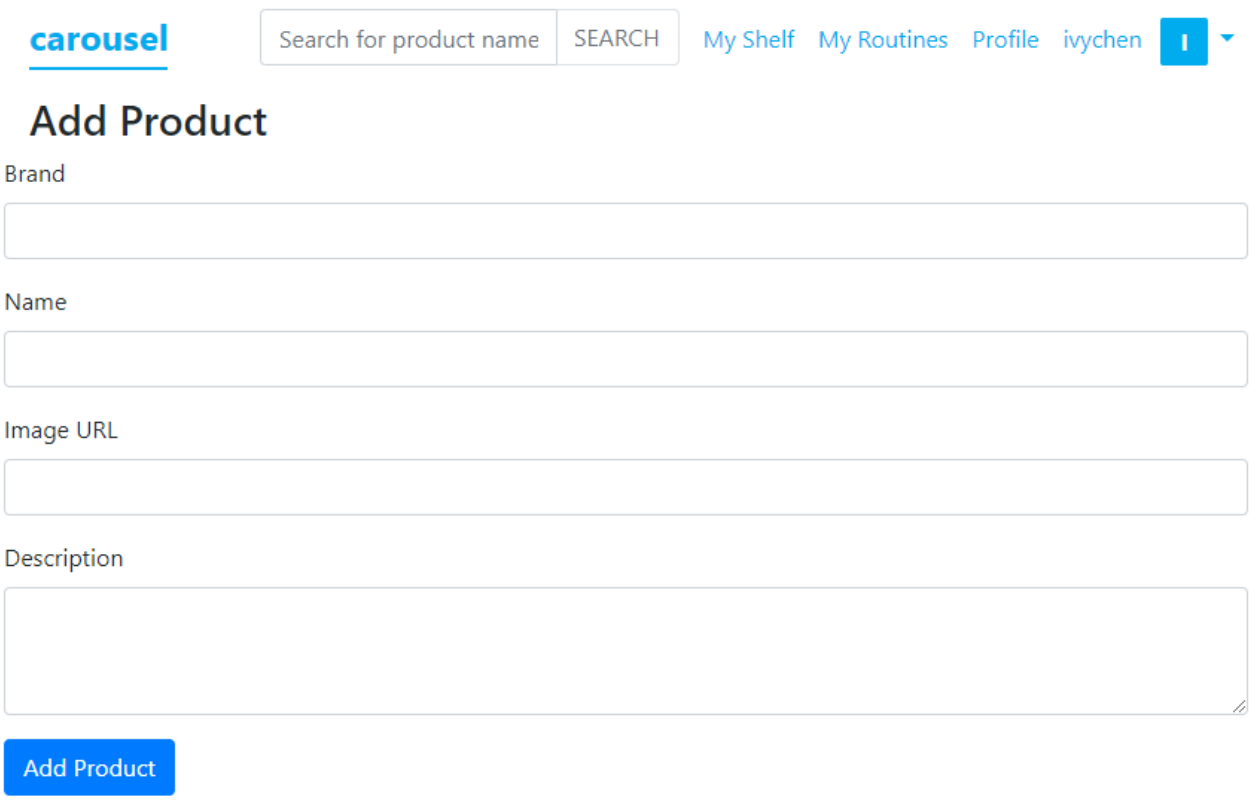


6. Scrape another batch of data



Setting up an Octoparse data scrape workflow

7. Create form to add product



Form to Add Product