Web Studio: Week 10 Turn-in #9

Name: Ivy Chen
UNI: ic2389
Due: Friday at 2PM
COMS6998 Adv. Web Studio

Original Goals

High Level Goal

1. The riskiest aspect of my project is allowing users to create custom routines with products and manage the products they own.

Low Level Goal

1. Set up the new environment and file structure
2. Create homepage
3. Create user profile page
4. Define database schemas
5. Populate database with initial batch of data
6. Allow for creation of persistent/saved routines
7. Create dashboard

Lessons Learned Through Iteration

1. My original project idea was to build a platform with an existing database of skincare/beauty products and allow users to rate/review products as well as receive customized recommendations. However, I has trouble finding the data — I thought I could find a product API, but ultimately wasn't able to find a good/updated one with the information about product that I wanted.
   I decided to shift the focus to helping users manage their routines and products that they use in a routine as well as log their reaction to products. The MVP

would allow users to search through an existing database of products or add their own to their collection. Then users can add products to their routine.

2. I ran into the problem of figuring out how to manage routines. I initially thought of creating journal-style records that contain an editable list to allow users to add notes, products, and other content related to that record. However, I wanted the ability to have a persistent record of products owned by a user as well as routines that the user set so they wouldn't have to repeat the same steps each time they visited the site.
   I decided to take inspiration from Trello's drag and drop cards onto boards interface to allow users to add cards onto boards. Each board is a routine and cards are products.

3. I had features in mind such as adding products to a user's collection, being able to manage products, adding products to routines and saving routines, recording effects, etc. However, it seemed that the flow was kind of fragmented. As a result, I went through a couple of design exercises to investigate a coherent user flow that made sense to me (and hopefully would make sense to the end user after testing). It seemed to make sense that a user can browse/add products that they own. In a separate widget, they should be able to add products they own to a routine, or "write-in" their own products.

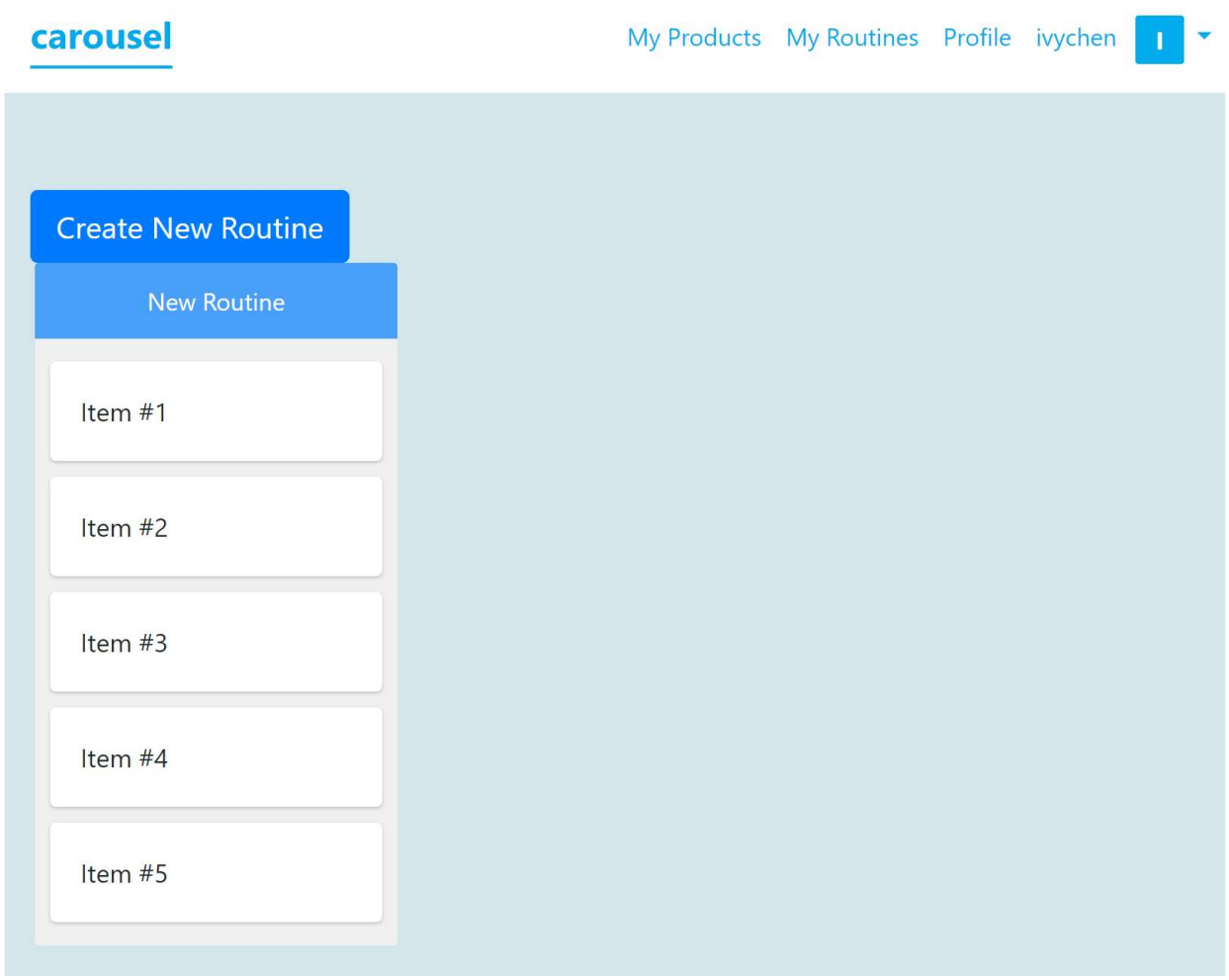
Goal Progress

High Level Goal

1. The riskiest aspect of my project is allowing users to create custom routines with products and manage the products they own.

Low Level Goal

1. Set up the new environment and file structure — Bootstrapped the setup using code from the previous two projects.
2. Create homepage — Created a basic homepage



3. Create user profile page
    a. Created basic profile page

```
{% extends "base.html" %}
{% block title %}Profile{% endblock %}

{% block body %}
  {% block content %}
  <div class="header row">
    <h2>Profile</h2>
  </div>

  <div class="container">

```

```
    <h1>User: {{ user.username }}</h1>
    <h2>Email: {{ user.email }}</h2>

    <form method=post>
      <div class="form-group row">
        <label for="skintype" class="col-sm-2 col-form-label">Skin Type</label>
        <div class="col-sm-10">
          <select class="custom-select" name="skintype" id="skintype" multiple>
            <option selected></option>
            <option value="dry">Dry</option>
            <option value="dehydrated">Dehydrated</option>
            <option value="normal">Normal</option>
            <option value="combination">Combination</option>
            <option value="oil">Oily</option>
          </select>
        </div>
      </div>

      <button type="submit" class="btn btn-primary mb-2">Submit</button>
    </form>
  </div>
  {% endblock %}
{% endblock %}
```

4. Define database schemas

```python
class Product(db.Model):
    id = db.Column('id', db.Integer, primary_key=True)
    timestamp = db.Column(db.DateTime)
    brand = db.Column('brand', db.String(140))
    name = db.Column('name', db.Text)
    description = db.Column('description', db.Text)
    imgsrc = db.Column(db.Text)

    categories = db.relationship('Category', secondary=categories, lazy='subquery', backref=db.backref('products', lazy=True))

    tags = db.relationship('Tag', secondary=tags, lazy='subquery', backref=db.backref('products', lazy=True))


class Category(db.Model):
    category = db.Column(db.String(200), primary_key=True)

class Tag(db.Model):
    tag = db.Column(db.String(140), primary_key=True)
    category = db.Column(db.String(140))

class Concern(db.Model):
    concern = db.Column(db.String(140), primary_key=True)

class List(db.Model):
    id = db.Column('id', db.Integer, primary_key=True)
    title = db.Column('title', db.String(256), nullable=False)

    # Owner of list
    owner = db.Column(db.Integer, db.ForeignKey('user.username', ondelete='CASCADE'), nullable=False)
    messages = db.relationship('Product', secondary=owns, lazy='subquery', backref=db.backref('list', lazy=True))
```


5. Populate database with initial batch of data
   I ended up selecting a few online retailers to scrape ~150 rows of data for an initial dataset using Octoparse, and populated my database with the data.
6. Allow for creation of persistent/saved routines
   Not implemented yet due to the time crunch this week.
7. Create dashboard
   I determined what the dashboard looks like as well as the JS libraries I want to use to implement it (Muuri for drag-and-drop interactions: https://haltu.github.io/muuri/), but haven't had time to flesh out implementation.