CHOP IT LIKE IT'S HOT, THE DJANGO RECIPE APPLICATION

A case study by Challis Regan, Lead Developer

Mentor: Vicki Nomwesigwa

OVERVIEW

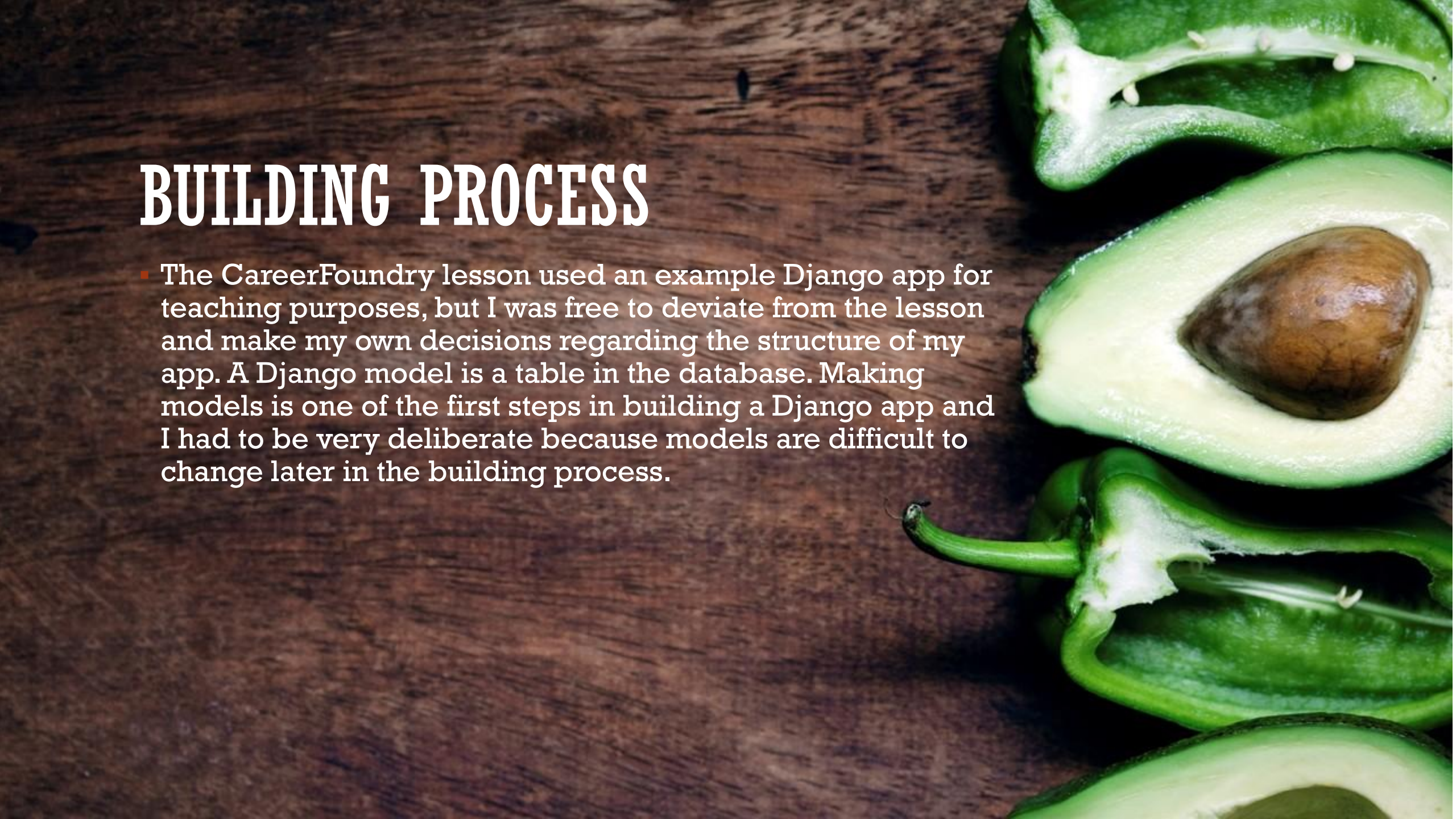
- Chop It Like It's Hot is a recipe web application built with Django. It is the successor to my Python command line recipe app. Users can create and modify recipes with ingredients, cooking time, and a difficulty parameter that is automatically calculated by the app. Users are able to search for recipes by their name and/or ingredients.
- It is a demo project I built for my web development course at CareerFoundry to use Django and Python to build the backend and frontend of a user-friendly recipe app. I finished this project in April 2025.

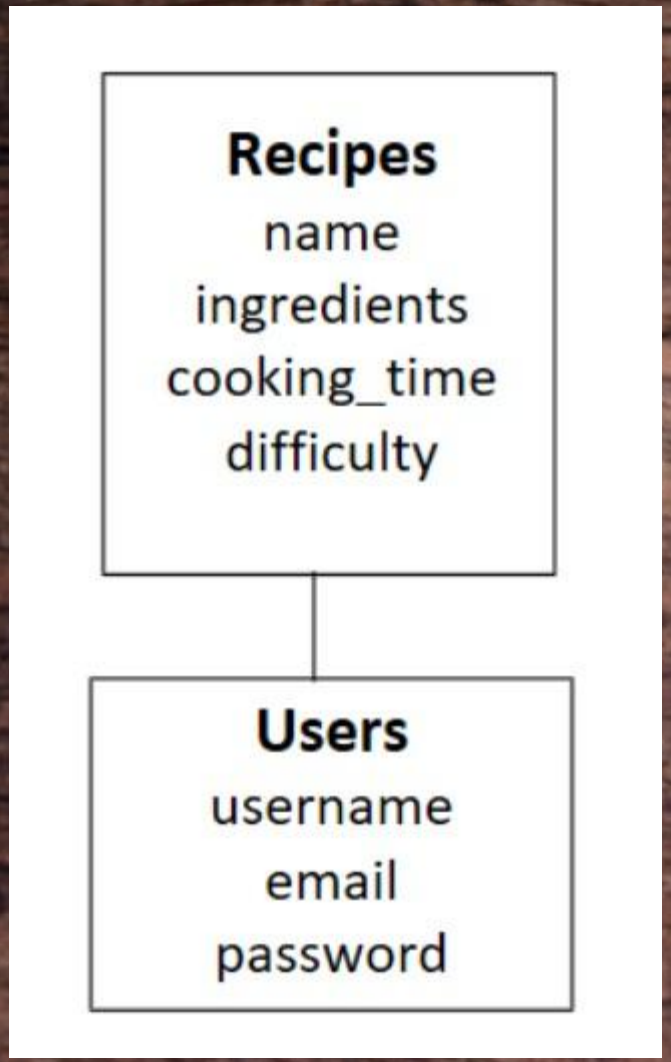
TECHNICAL STACK

- Productivity Tool: Trello
- Backend: Django, SQLite/PostgreSQL
- Frontend: HTML, CSS, Bootstrap
- Media Storage: Amazon S3 (via django-storages)
- Deployment: Heroku
- Data Visualization: Matplotlib
- Authentication: Django auth system (login, logout, register)

BUILDING PROCESS

- The CareerFoundry lesson used an example Django app for teaching purposes, but I was free to deviate from the lesson and make my own decisions regarding the structure of my app. A Django model is a table in the database. Making models is one of the first steps in building a Django app and I had to be very deliberate because models are difficult to change later in the building process.





BUILDING MODELS, FIRST DRAFT

The Recipes is the table and the name, ingredients, cooking_time and difficulty are the attributes. I later realized this model created an issue. The ingredients are only linked to one recipe, even if I use the same ingredient in multiple recipes. Also, I could not add quantities for the ingredients, which is quite important for a recipe website. I decided to make two other models, Ingredients and RecipeIngredients (linking the ingredients to the quantities). I needed to properly link the models together, which was not covered in the lesson. I did online research, troubleshooting and consulted with my mentor to get this to work.

models.py 3

recipe-app > recipes > models.py > ...

```python
from django.db import models
from ingredients.models import Ingredient
from django.shortcuts import reverse


# Create your models here.
class Recipe(models.Model):
    name = models.CharField(max_length=120)
    ingredients = models.ManyToManyField(
        Ingredient,
        related_name="recipes",
        through="recipe_ingredients.RecipeIngredient",
    )
    cooking_time = models.PositiveIntegerField()
    difficulty = models.CharField(max_length=20)
    directions = models.TextField(default="directions")
    pic = models.ImageField(upload_to="recipes", default="no_picture.webp")
```

models.py 2

recipe-app > ingredients > models.py > ...

```python
from django.db import models

# Create your models here.
class Ingredient(models.Model):
    name = models.CharField(max_length=100, unique=True)

    def __str__(self):
        return str(self.name)
```

models.py 2

recipe-app > users > models.py > ...

```python
from django.db import models

class User(models.Model):
    name = models.CharField(max_length=120)
    email = models.EmailField(max_length=254, unique=True)

    def __str__(self):
        return str(self.name)
```

Code for the final versions of the Recipes and Ingredients models. I modified my User model after learning how to set up proper authentication.

Code for the final version of the RecipeIngredients model

models.py 2

recipe-app > recipe_ingredients > models.py > RecipeIngredient > __str__

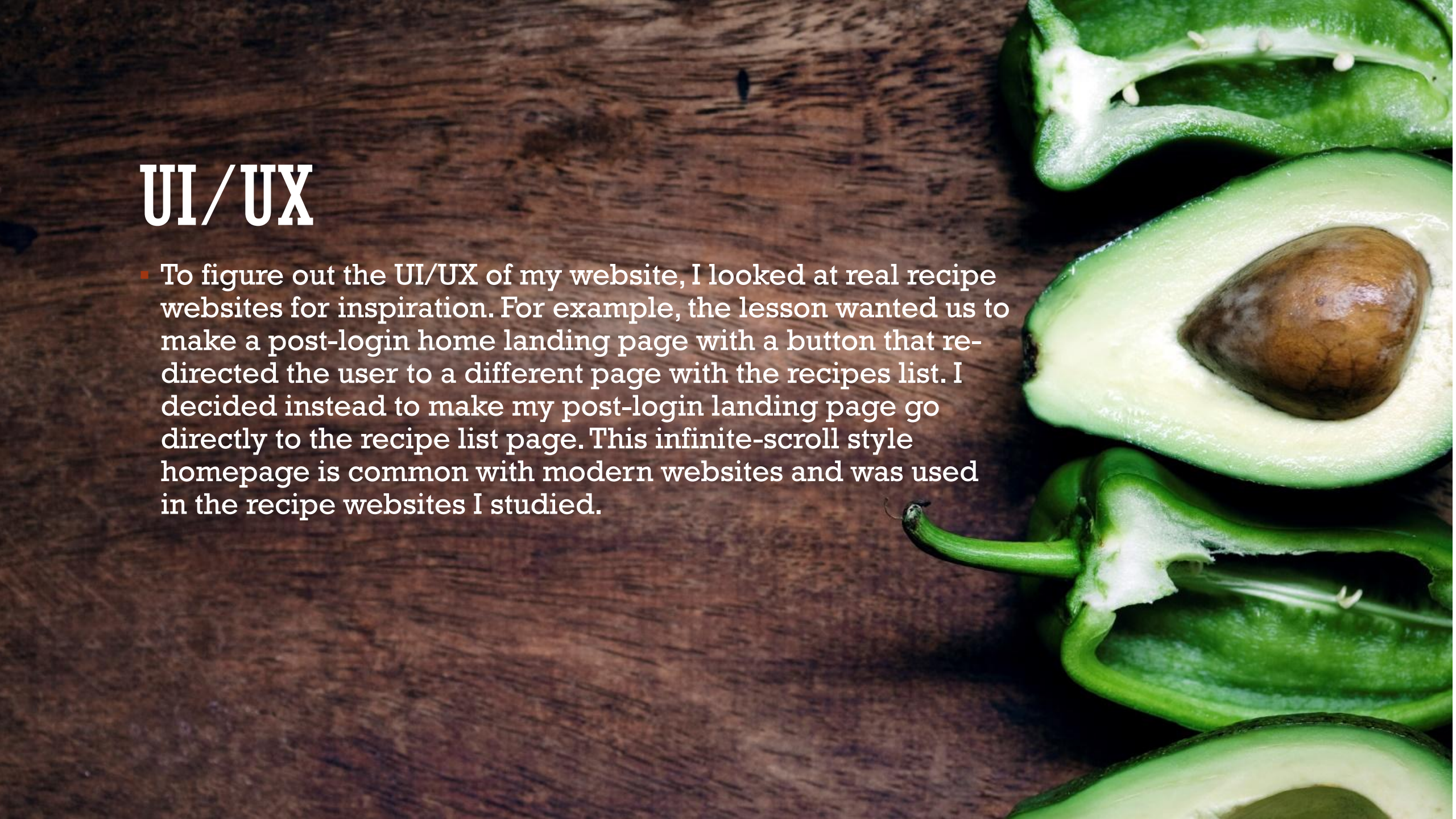
```python
from django.db import models
from ingredients.models import Ingredient
from recipes.models import Recipe

# Create your models here.
class RecipeIngredient(models.Model):
    recipe = models.ForeignKey(Recipe, on_delete=models.CASCADE)
    ingredient = models.ForeignKey(Ingredient, on_delete=models.CASCADE)
    quantity = models.CharField(max_length=50) # e.g., '2 cups', '1 tbsp'

    def __str__(self):
        return f"{self.quantity} of {self.ingredient.name} for {self.recipe.name}"
```

MODELS, VIEWS, TEMPLATES

- After I finalized the models, I created the views and templates and set up the URLs and routes.

UI/UX

- To figure out the UI/UX of my website, I looked at real recipe websites for inspiration. For example, the lesson wanted us to make a post-login home landing page with a button that re-directed the user to a different page with the recipes list. I decided instead to make my post-login landing page go directly to the recipe list page. This infinite-scroll style homepage is common with modern websites and was used in the recipe websites I studied.



Chop It Like It's Hot

Latest Recipes

Peanut Butter & Jelly Sandwich

5-Minute Microwave Hummus

Overnight Oats

RETROSPECTIVE

- How did the original objective compare with the final results?
- I am very careful to read all of the requirements of an assignment before I turn it in. Overall, I feel I successfully transformed my Python command line app to a user-friendly Django recipe app.
- What was the most challenging or surprising part of your project?
- Django has a lot of hidden background actions and I had to read the documentation and do research a lot to figure out what was going on behind the scenes. There were some things I could easily modify and other things I had to work around or do differently than intended.

RETROSPECTIVE PT 2

- What would you do differently next time, or what do you plan to improve on in a second iteration?
- The UI could always be tweaked and re-worked. As a web developer, it is important to keep up with design trends so your website always looks and feels fresh.

THANK YOU! FEEL FREE TO CONNECT WITH ME. ☺

Challis.regan1@gmail.com

GitHub

Portfolio