Task 2.7

Recipe Search Engine

For the Task in Exercise 2.7, I will develop a Dynamic Search Engine in the Recipes Application with following specific functions:

1. Search by Recipe Name (incl. Wildcard)

Using the __icontains filter, I will allow users to type "Choc" to find "Chocolate Cake" or "Dark Chocolate Mousse”. This also fulfills the bonus requirement for partial matches.

2. Search by Ingredient

Since my Recipe model has a ManyToManyField to Ingredient, I will create a search that looks up recipes based on their components. For example, searching for "Flour" will return every recipe that includes flour in its RecipeIngredient junction table.

3. Search by Difficulty (Data Grouping)

Since my model automatically calculates difficulty, I will allow users to filter for "Easy" or "Hard" recipes. This provides excellent data for the **Charts** to be generated (e.g., a pie chart showing the distribution of difficulty across search results).

4. Clickable DataFrame Output

The primary difference between this and my current Recipe Application "List View" is the **Data Processing** layer:

- **The Query**: Finds the matching recipes.
- **The DataFrame**: Converts those recipes into a structured Pandas table.
- **The Transformation**: I will inject HTML anchor tags into the "Name" column of the DataFrame. This means when the table is displayed, clicking a recipe name will take the user directly to its Detail Page.

Data Analysis

I will implement a "Data Lab" component that visualizes the results of the search query. These charts will be generated based on user input (via a dropdown menu in the search form), allowing users to decide which perspective of the data he wants to analyze.

1. Bar Chart: Cooking Time Comparison

- Scenario: Visualizing how long each recipe takes to prepare among the search results.
- X-Axis: Recipe Names (Labels).
- Y-Axis: Cooking Time (in minutes).
- Purpose: To identify which recipes are the most and least time-consuming within the filtered set.

2. Pie Chart: Difficulty Distribution

- Scenario: Understanding the complexity of the searched recipes.
- Labels: Difficulty Levels (Easy, Medium, Intermediate, Hard).
- Values: The count of recipes belonging to each difficulty category.
- Purpose: To show the percentage of "Quick & Easy" meals versus "Chef Level" challenges in the current results.

3. Line Chart: Ingredient Count vs. Cooking Time

- Scenario: Tracking if there is a correlation between the number of ingredients and the time it takes to cook.
- X-Axis: Recipe Names (ordered by number of ingredients).
- Y-Axis: Cooking Time.
- Purpose: To visualize trends in recipe complexity. While usually used for time-series data, a line chart here can show the "curve" of prep-time as ingredient lists grow longer.

Summary of what is to be done

- **Currently**: I have a list of all recipes and a basic JS filter for ingredients.
- **Task 2.7**: I am adding a "Data Lab" search that analyzes results, converts them to a DataFrame for professional table display, and visualizes trends via Matplotlib.

User Journey

This section outlines the path a user takes to interact with the data-driven features of the Recipe Application. The goal is to move from general discovery to specific data analysis and detailed recipe inspection.

1. **Entry & Authentication:** The user lands on the Home Page. Seeing the "Explore Recipes" link, he realizes this is a protected feature. He navigates to the Login page, authenticates his session, and is redirected to the main page.
2. **Search & Discovery:** On the Recipe List page, the user interacts with the new "Data Lab" search form. He inputs a partial name (wildcard search) or selects specific ingredients. Simultaneously, he selects a "Visualization Preference" from the dropdown (e.g., Bar Chart for cooking times).
3. **Data Processing & Visualization:** Upon submission, the system processes the request. The user is presented with a Pandas-generated HTML table containing the specific results. Above or below this table, a Matplotlib chart appears, providing a visual summary of the data just filtered.
4. **Detail Inspection:** The user identifies a recipe of interest within the DataFrame table. Because the names are rendered as clickable links, he clicks a title and is taken to the Recipe Detail page to view the full ingredient list and cooking instructions.
5. **Exit:** Once the analysis is complete, the user uses the navbar to log out, terminating the session and securing the data.

Execution Flow

The following flow represents the logical sequence of events from the moment the user initiates a request to the final rendering of data.

- **Step 1: Homepage** → User views public content.
- **Step 2: Login View** → User submits credentials; Django validates and starts a session.
- **Step 3: Search Form (Input)** → User selects criteria (Name/Ingredient/Difficulty) and Chart Type.
- **Step 4: View Logic (Processing**):
    - QuerySet: Django filters the database using icontains or ManyToManyField lookups.
    - Pandas: The QuerySet is converted into a DataFrame.
    - Transformation: Recipe IDs are mapped to URL paths and injected into the "Name" column as HTML links.
    - Matplotlib: A chart is generated based on the user's "Chart Type" selection.

- **Step 5: Results Page (Output**) → The browser renders the dynamic table (safe filter) and the Base64-encoded image.
- **Step 6: Detail View** → User clicks a link in the table; get_absolute_url redirects to the specific recipe record.
- **Step 7: Logout** → Session is cleared; user is redirected to the logout success page.