CTRL-ALT-ELITE
CTRL-ALT-EAT

1. CREATE RECIPE FUNCTIONALITY

app.py file:

```
@app.route('/create_recipe', methods=['GET', 'POST'])
def create_recipe():
   if request.method == 'POST':
       recipe = {
           'id': len(recipes) + 1,
           'name': request.form['recipe_name'],
           'description': request.form['recipe_description'],
           'ingredients': request.form['recipe_ingredients'],
           'instructions': request.form['recipe_instructions']
       } #making a dictionary for the recipe attributes


       recipes.append(recipe) #adding the recipe to the recipes list with users input

       print(recipes) #printing the reciipes list to see if the recipe was added
correctly

       return redirect(url_for('home')) #redirects back to the base url for index.html

   return render_template('index.html', recipes=recipes) #keeps the user on the
index.html page
```

The code snippet above is for the creation of recipes. If the request method equals POST, the user input from the html form will be added to the corresponding fields in the recipe dictionary. The recipe will then be added to the global array recipes. The recipes array is printed in the terminal to ensure that the recipe was added correctly. The user is then redirected to the ‘home’ / index page.

index.html file

```
<button type="button" id="add-recipe-button">Add Recipe</button>


            <div id="recipe-form" style="display:none;">


                <form action="{{ url_for('create_recipe') }}" method="post">
                    <label for="recipe_name">Recipe Name:</label>
                    <input type="text" id="recipe_name" name="recipe_name">
```

```
                    <label for="recipe_description">Recipe Description:</label>
                    <input type="text" id="recipe_description"
name="recipe_description">
                    <label for="recipe_ingredients">Recipe Ingredients:</label>
                    <input type="text" id="recipe_ingredients"
name="recipe_ingredients">
                    <label for="recipe_instructions">Recipe Instructions:</label>
                    <input type="text" id="recipe_instructions"
name="recipe_instructions">
                    <input id="recipe-submit" type="submit" value="Submit">
                </form>
            </div>
        </div>
```

This html file shows the structure for the form used to create recipes. The user must click the button in order to open up the form for recipe creation. It has a post method and url which connects it to the create_recipe method from the app.py file. When the user clicks submit, the recipe is added.

script.js file

```
document.addEventListener('DOMContentLoaded', function() {
   var recipe_button = document.getElementById('add-recipe-button');
   recipe_button.addEventListener('click', function() {
       var form = document.getElementById('recipe-form');
       if (form.style.display === 'none') {
           form.style.display = 'block';  // Show the form
           recipe_button.textContent = 'Close Menu'

       } else {
           form.style.display = 'none';   // Hide the form
           recipe_button.textContent = 'Add Recipe'
       }
   });
});
```

This script file contains an event listener for the submit button for the creation of recipes. It toggles the button, so that if the user clicks it, it opens the form or the form can be closed.

2. VIEWING RECIPES FUNCTIONALITY

index.html file

```
<div class="recipe-list">
            <h2>Recipes</h2>
            <ul>

                    {% for recipe in recipes %}
```

```
                        <div class="recipe-item">
                                <h3>{{ recipe.name }}</h3>
                                <p>Description</p>
                                <p>{{ recipe.description }}</p>
                                <p>Ingredients</p>
                                <p>{{ recipe.ingredients }}</p>
                                <p>Instructions</p>
                                <p>{{ recipe.instructions }}</p>
                                <a href="{{ url_for('edit_recipe',
recipe_id=recipe.id) }}">Edit Recipe</a>
                            </div>
                    {% endfor %}

            </ul>
        </div>
```

This snippet takes from the ‘recipes=recipes’ from the create_recipe function and displays the attributes related to the recipes found in the array. It uses jinja blocks to go through the recipes array and display the recipes attributes.

3. SEARCHING FOR RECIPE FUNCTIONALITY

app.py file

```
@app.route('/search_recipe', methods = ['GET', 'POST'])
def search_recipe():
   if request.method == "POST": #using post method to search for a recipe
       recipe_name = request.form.get('search_name') #getting the recipe name
from the form
       print(recipe_name)
       recipe = next((r for r in recipes if r['name'] == recipe_name), None)
#searching for the recipe by its name to match search input
       print(recipe)
       if not recipe: #if the recipe is not found it will return to the home
page
           return redirect(url_for('home'))

   return render_template('search_recipe.html', recipe = recipe)
```

In the following code snippet, the search_recipe functionality is shown. The method waits for POST from search bar form, then gets the recipe name. Printing name to verify recipe. Then the recipe name is searched for in the array. If the name in the array is found, the user is redirected to the search_recipe.html file, where the recipe is displayed. The recipe variable is also passes to the html file.

search_recipe.html

```
<div class="recipe-item">
            <h3>{{ recipe.name }}</h3>
            <p>Description</p>
            <p>{{ recipe.description }}</p>
            <p>Ingredients</p>
            <p>{{ recipe.ingredients }}</p>
            <p>Instructions</p>
            <p>{{ recipe.instructions }}</p>
            <a href="{{ url_for('edit_recipe', recipe_id=recipe.id) }}">Edit
Recipe</a>
            <a class = "return-main-page" href="{{ url_for('home') }}">Return to
Main Page</a>
        </div>
```

With the passing of the recipe variable, the recipe's attributes are showcased.

4. EDITING RECIPE FUNCTIONALITY

app.py file:

```
@app.route('/edit_recipe/<int:recipe_id>', methods=['GET', 'POST'])
def edit_recipe(recipe_id):
    recipe = next((r for r in recipes if r['id'] == recipe_id), None) #searching for
recipe by its id
    if not recipe:
        return redirect(url_for('home')) # returns to the home page if recipe is not
found

    if request.method == 'POST':          #using post method to edit the recipe
        recipe['name'] = request.form['recipe_name']
        recipe['description'] = request.form['recipe_description']
        recipe['ingredients'] = request.form['recipe_ingredients']
        recipe['instructions'] = request.form['recipe_instructions']
        return redirect(url_for('home'))

    return render_template('edit_recipe.html', recipe=recipe) #renders the
edit_recipe.html page with the recipe to be edited
```

This code snippet shows the edit_recipe function. The function matches the recipe by ID, else it simply returns home. When the recipe is found and the method is POST, the attributes for the

recipe from the dictionary are open to be edited from the form when redirected to the edit_recipe.html file. The function returns/ passes the recipe variable to the html file as well.

edit_recipe.html

```
<div class = "edit-recipe-form">
          <form action="{{ url_for('edit_recipe', recipe_id=recipe['id']) }}"
method="post">
              <label for="recipe_name">Recipe Name:</label>
              <input type="text" id="recipe_name" name="recipe_name" value="{{
recipe.name }}">

              <label for="recipe_description">Recipe Description:</label>
              <input type="text" id="recipe_description" name="recipe_description"
value="{{ recipe.description }}">

              <label for="recipe_ingredients">Recipe Ingredients:</label>
              <input type="text" id="recipe_ingredients" name="recipe_ingredients"
value="{{ recipe.ingredients }}">

              <label for="recipe_instructions">Recipe Instructions:</label>
              <input type="text" id="recipe_instructions" name="recipe_instructions"
value="{{ recipe.instructions }}">

              <input id="recipe-submit" type="submit" value="Update Recipe">
          </form>
          <a class = "return-main-page" href="{{ url_for('home') }}">Cancel</a>
      <div>
```

The form for editing the recipe is shown below. The form action connects the form to the url of the edit_recipe function. It also sets the variable recipe_id equal to the value of the id of the recipe being edited. There is a cancel button that takes the user back to the home page.

5. DELETE RECIPE FUNCTIONALITY

app.py file

```
@app.route('/delete_recipe/<int:recipe_id>', methods=['POST'])
def delete_recipe(recipe_id):
   global recipes
   recipes = [recipe for recipe in recipes if recipe['id'] != recipe_id]
   flash('Recipe successfully deleted!')
   return redirect(url_for('home'))
```

This function finds the recipe id of the recipe targeted for deletion. If the id matches an id of a recipe in the recipes array, it will be removed from the array.

Index.html file

```
<div class="recipe-list">
              <h2>Recipes</h2>
              <ul>

                      {% for recipe in recipes %}
                          <div class="recipe-item">
                                  <h3>{{ recipe.name }}</h3>
                                  <p>Description</p>
                                  <p>{{ recipe.description }}</p>
                                  <p>Ingredients</p>
                                  <p>{{ recipe.ingredients }}</p>
                                  <p>Instructions</p>
                                  <p>{{ recipe.instructions }}</p>
                                  <a href="{{ url_for('edit_recipe',
recipe_id=recipe.id) }}">Edit Recipe</a>
                                                          <!-- Delete button -->
              <form action="{{ url_for('delete_recipe', recipe_id=recipe.id) }}"
method="post" style="display:inline;">
                  <button type="submit" onclick="return confirm('Are you sure you
want to delete this recipe?')">Delete</button>
              </form>

                                  </div>
                      {% endfor %}

              </ul>
          </div>
```

This code snippet shows the form action for the delete button. It ensure the recipe id is the same as the recipe being deleted. When the button is clicked, an alert appears, letting the user know they've deleted the recipe.