

1 HTML Files

1.1 Kitchen html

A skeleton of this file has been provided to you. Please look at it when reading the following instructions.

```
<html lang="en">

<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <link rel="stylesheet" href="../static/css/kitchen.css">
  <script type="text/javascript" src="../static/js/kitchen.js"></script>
  <title>Document</title>
</head>

<body>
  <h1>Kitchen - List of current orders</h1>
  <ul id="customer_order_ul">
    // Add Jinja2 FOR loop code here
    <li class="customer_order_li">
      <h3>Customer Name: // Add Jinja2 variable replacement here </h3>
      <ul>
        // Add Jinja2 FOR loop code here
        <li class="items"> // Add Jinja2 variable replacement here </li>
        // Add Jinja2 END FOR loop code here
        </li>
        // Add Jinja2 END FOR loop code here
      </ul>
      <a href="/delete// Add Jinja2 variable replacement here">Delete
Order</a>
    </li>
    // Add Jinja2 END FOR loop code here
  </ul>
</body>

</html>
```

You need to provide an array of orders when parsing this file via Flask render_template function. Each order is a JSON object containing the following data:

- Order id
- Customer name
- A single string with the orders; different orders in this string are separated by comma. To generate an array of orders, use the split method
- Date that the order was created

Inside an unordered list, create a Jinja2 FOR structure to loop over the orders array and create containing

- Customer name
- List of items in the order. To do so, you will need to use Jinja2 For loop again and split the single orders string into items
- Create a link <a> for the delete operation, which points to your app endpoint “/delete/orderId”

1.2 Main_menu.html

A skeleton of this file has been provided to you. Please look at it when reading the following instructions.

```
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <link rel="stylesheet" href="../static/css/main_menu.css">
  <title>Document</title>
</head>
<body>
  <h1 id="title">Atarashy Main Menu</h1>
  <ul id="main_plates">
    // Add Jinja2 FOR loop code here
    <li>
      <a href=".." // Add Jinja2 variable replacement here">
        
        <h2> // Add Jinja2 variable replacement here </h2>
      </a>
    </li>
    // Add Jinja2 END FOR loop code here
  </ul>
</body>
```

You need to provide an array of JSON objects, containing the main plates available in your store when parsing this file via Flask render_template function. Each main_plate JSON object has the following fields:

- a) An Id (however we will not be using it for anything on this page)
- b) url location of the given plate, such as sushis.html (to be used in the link described below)
- c) image path
- d) plate name, such as sushis

Inside an unordered list, create a Jinja2 FOR structure to loop over the main_plates array and create containing

- a) a link to the plate url
- b) plate image
- c) plate name (h2 tag)

1.3 Order_summary.html

A skeleton of this file has been provided to you. Please look at it when reading the following instructions.

```
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <script src="../../static/js/order_summary.js"></script>
  <title>Document</title>
</head>
<body>
  <h1>Order Summary</h1>
  <h2>Customer Name: // Add Jinja2 variable replacement here</h2>
  <ul>
    // Add Jinja2 FOR loop code here
    <li>
      <h2>// Add Jinja2 variable replacement here</h2>
    </li>
    // Add Jinja2 END FOR loop code here
  </ul>
</body>
```

You need to provide the customer name and an array of order_items when parsing this file via Flask render_template function. In this case the order_items is just an array of Strings containing each item, which can be easily created by getting the contents of the text area and splitting it by “\n” (this should be done in the python route endpoint, not here)

With this info:

- a) Display the customer name using H2 tag
- b) Inside an unordered list, Loop over the order items
- c) For each item, create an H2 containing the item

1.4 Sushi_menu.html

A skeleton of this file has been provided to you. Please look at it when reading the following instructions.

You need to provide an array of JSON sushi objects when parsing this file via Flask render_template function. Each sushi object contains:

- a) Sushi Id
- b) Sushi price
- c) Sushi image
- d) Sushi type

Inside an unordered list, create a Jinja2 FOR structure to loop over the sushis array and create containing

- a) The sushi image
- b) Its name (h3)
- c) Its price (h3)

Inside a html form tag, the only thing you need to add in the given skeleton is to add the url for the "action" attribute. Please use url_for in your main python app and pass the url via an input variable in the render_template when rendering this page.

```
<body>
  <h1 id="title"> Sushi Menu </h1>
  <ul id="sushis">
    // Add Jinja2 FOR loop code here
    <li onClick="handle_clicked('Sushi: {{sushi.type}} - Price:
      ${{sushi.price}}')">
      
      <h3>// Add Jinja2 variable replacement here</h3>
      <h3>// Add Jinja2 variable replacement here</h3>
    </li>
    // Add Jinja2 END FOR loop code here
  </ul>
  <form id="order_form" action="//ADD Jinja2 variable replacement for /order_request/
path" method="POST">
    <h2>Order Summary</h2>
    <textarea name="order_summary" id="order_summary" rows="10" cols="35"></textarea>
    <div id="customer_info">
      <label for="customer_name">Customer Name: </label>
      <input type="text" id="customer_name" name="customer_name">
    </div>
    <input type="submit" id="place_order">
  </form>
</body>
```

1.5 Welcome_page.html

A skeleton of this file has been provided to you. Please look at it when reading the following instructions.

You need to provide a variable that contains the href for the main menu page when parsing this file via Flask `render_template` function.

```
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <link rel="stylesheet" href="../static/css/welcome_page.css">
  <title>Document</title>
</head>
<body>
  
  
  <h1><a href="// add Jinja2 replacement variable here">Click here to start your
order</a></h1>
</body>
```

2 Database Models

2.1 Order_model.py

Create a database model, named `order_model.py`, with the following features:

- Use `Base_order` for the `declarative_base` object
- Create a class named `Order`
- Name the table `Orders`
- Class fields:
 - `id`
 - `customer_name`
 - `orders`
 - `date_created`
- Create a class function, named `toJSON` that returns a JSON object of the class object (as done in our hands-on)

2.2 Main_plate_model.py

Create a database model, named `main_plate_model.py`, with the following features:

- Use `Base_main_plate` for the `declarative_base` object
- Create a class named `Plate`
- Name the table `Plates`
- Class fields:
 - `id`
 - `type`
 - `image`
 - `url (string)`
- Create a class function, named `toJSON` that returns a JSON object of the class object (as done in our hands-on)

2.3 sushi_model.py

Create a database model, named `sushi_model.py`, with the following features:

- Use `Base_sushi` for the `declarative_base` object
- Create a class named `Sushi`
- Name the table `Sushis`
- Class fields:
 - `id`
 - `type`
 - `image`
 - `price`
- Create a class function, named `toJSON` that returns a JSON object of the class object (as done in our hands-on)

3 The Restaurant Flask Server Application

3.1 Imports

The following imports shall be in your server application:

```
from flask import Flask, render_template, request, url_for, abort, redirect
from sqlalchemy import create_engine
from sqlalchemy.orm import sessionmaker
from sushi_model import Base_sushi, Sushi
from main_plate_model import Base_main_plate, Plate
from order_model import Base_order, Order
```

3.2 Binding your application with database files

Create session objects for all three database files needed for this application: order_model, main_plate_model.py, and sushi_model.py. The respective binary files that will hold the data should be named orders.db, main_plates.db, and sushis.db, respectively.

3.3 Application routes

All the routes described below shall be placed in the restaurant_app.py file.

3.3.1 The root route

- Create the root route of your application: @app.route("/")
- Create its associated function, welcome_page, with the following features:
 - This function shall render the welcome.html template
 - The main menu url path shall also be passed as an input argument for the render_template function. Please use url_for function to define the url path

3.3.2 The regenerate_options route

As discussed in our last lecture, we need to have a way to generate the meal options by looking at a text file, which is an easy way for the customer to add or remove items by themselves. This route could then read that text file, create objects, and repopulate the main_plate and sushi databases. For this final project, you do not need to read a text file, but you will need to create commands to create main_plate and sushi objects and repopulate the databases. Before using this URL, manually remove the db files from your project, so your application doesn't duplicate contents.

So,

- create a route "/regenerate_database/" and its associated function, regenerate_database()
- create 5 sushi objects and add them to the sushi database by using session_sushi.add and session_sushi.commit functions. Look at the sushi model constructor to see how to create elements correctly
- Repeat this process for the main_plates

3.3.3 The main menu page route

- Create a “/main_menu_page/” route and its associated main_menu_page() function
- Retrieve all main_plate objects from the database
- Transform the main_plate objects to JSON objects
- Render_template “main_menu.html”, with one additional input:
 - The JSON main_plates array

3.3.4 The sushis route

- Create a “/sushis/” route and associated sushi_page() function
- Retrieve all sushi objects from the database
- Transform the sushi objects to JSON objects
- Render_template “sushi_menu.html”, with one additional input:
 - The JSON sushis array

3.3.5 The order_request route

- Create a “/order_request/” route that accepts POST requests, and associated order_request() function
- Retrieve the order, a single long String from the html textarea, via request.form[] command
- Split this long string into order requests (array of items), by using split(“\n”)
- Retrieve customer name from the request.form[] command
- Create a new Order object by calling the Order class constructor
- Try (and Except) to add the new order using session_orders and save it using commit()
- Render the order_summary.html template, with the following input:
 - An array with order_requests
 - The customer_name, by using request.form[]

3.3.6 The kitchen route

It is exactly the same as the one we worked on the class hands-on, so look at it.

- Create a “/kitchen/” route and its associated function kitchen()
- Get the orders from the orders database, via session_orders object
- Convert the orders from an array of Order objects to an array of JSON objects, using the toJSON() method
- Render_template “kitchen.html” with the array of JSON objects as input

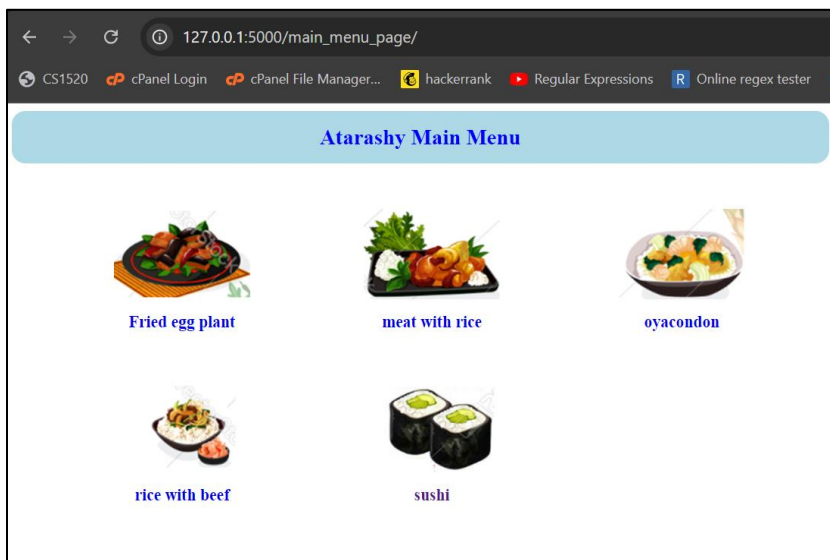
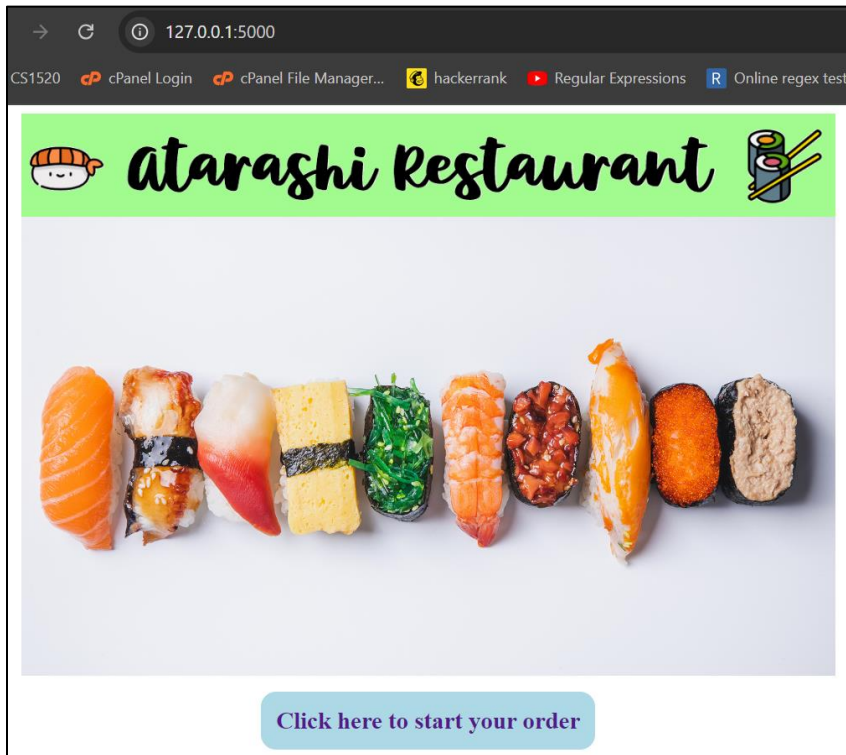
3.3.7 The delete order route

It is exactly the same as the one we worked on the class hands-on, so look at it.

- Create a “/delete/<id>” route and its associated function delete(id)
- Get the order specified by its id from the orders database, via session_orders object
- Check if the order exists
- Issue a delete command using session_orders.delete()
- Commit deletion
- Return redirect(url_for(“kitchen”))

4 General Comments


- After running the application, visit the endpoint `/regenerate_database/`, so that the database is populated with objects to be used in your pages
- Study the hands-on Mac Donald project we did in class. This project has the same structure as this one
- See the figures below showing the pages, so you can see what is expected from your application




→ 127.0.0.1:5000/sushis/

CS1520 cPanel Login cPanel File Manager... hackerrank Regular Expressions Online regex tes


Sushi Menu




Salmon
11.6




Crab
10.5




Tuna
12.9



Seaweed
9.3



California
12.6



Salmon
11.6

Order Summary

Sushi: Tuna - Price: \$12.9
Sushi: Crab - Price: \$10.5
Sushi: Salmon - Price: \$11.6

Customer Name:

myPitt | All Campuses Topic: Final version of the McD

← → 127.0.0.1:5000/order_request/

CS1520 cPanel Login cPanel File Manager... hackerrank Reg

Order Summary

Customer Name: Paulo

- Sushi: Tuna - Price: \$12.9
- Sushi: Crab - Price: \$10.5
- Sushi: Salmon - Price: \$11.6

→ 127.0.0.1:5000/kitchen/

CS1520 cPanel Login cPanel File Manager... hackerrank >> | All Bookmarks

Kitchen - List of current orders

Customer Name: Maria

Sushi: Salmon - Price: \$11.6

Sushi: California - Price: \$12.6

Customer Name: Joanna

Sushi: Seaweed - Price: \$9.3

Sushi: California - Price: \$12.6

Sushi: Salmon - Price: \$11.6

Customer Name: Paulo

Sushi: Tuna - Price: \$12.9

Customer Name: Paulo

Sushi: Tuna - Price: \$12.9

Sushi: Crab - Price: \$10.5

Sushi: Salmon - Price: \$11.6

It was a blessing to still be here and be able to teach you guys. Keep in touch once in a while, by telling me when you got a job, etc.

Obrigado

Paulo Brasko