# HTML Files

## 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:

1. Order id
2. Customer name
3. 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
4. Date that the order was created

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

1. Customer name
2. 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
3. Create a link <a> for the delete operation, which points to your app endpoint “/delete/orderId”

## 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">**

**<img src="// Add Jinja2 variable replacement here" alt="">**

**<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:

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

Inside an unordered list, create a Jinja2 FOR structure to loop over the main\_plates array and create <li> containing

1. a link to the plate url
2. plate image
3. plate name (h2 tag)

## 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:

1. Display the customer name using H2 tag
2. Inside an unordered list, Loop over the order items
3. For each item, create an H2 containing the item

## 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:

1. Sushi Id
2. Sushi price
3. Sushi image
4. Sushi type

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

1. The sushi image
2. Its name (h3)
3. 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}}')">**

**<img src="// Add Jinja2 variable replacement here" alt="">**

**<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>**

## 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>**

**<img src="../static/images/welcome\_page/atarashy\_logo.png" alt="">**

**<img src="../static/images/welcome\_page/welcome.jpg" alt="">**

**<h1><a href="// add Jinja2 replacement variable here">Click here to start your order</a></h1>**

**</body>**

# Database Models

## Order\_model.py

Create a dabase 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)

## Main\_plate\_model.py

Create a dabase 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)

## sushi\_model.py

Create a dabase 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)

# The Restaurant Flask Server Application

## 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**

## 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.

## Application routes

All the routes described below shall be placed in the restaurant\_app.py file.

### 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

### 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 obejcts 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

### 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

### 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

### 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[ ]

### 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

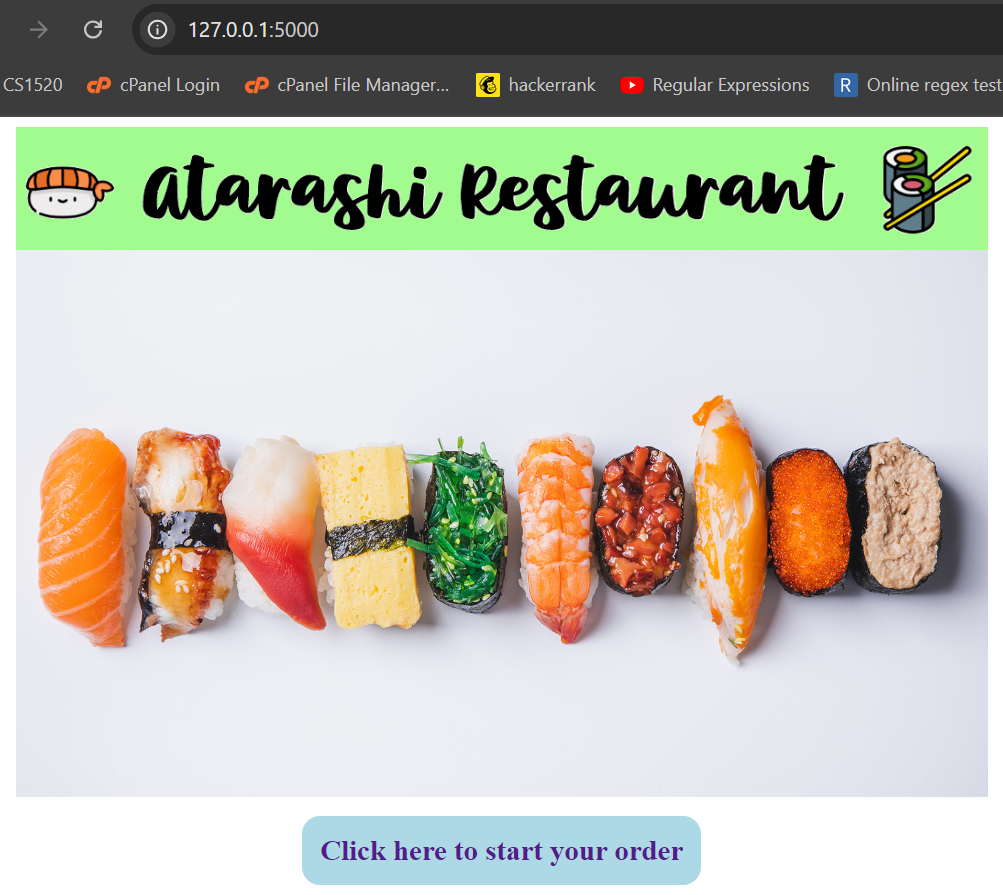
### The delete order route

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”)

# 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



A screenshot of a menu

Description automatically generated

A screenshot of a menu

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

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