

CS-1520 Fall 2024 – Final Project

1 Introduction

In this final project, you will create a Flask app that serves an online restaurant menu. This menu is normally accessed by a QR code on the table that the customer is sitting on. Your application shall present the menu to the customers, take their orders, notify the kitchen staff of the order request, so they can prepare your mean, and finally let the waitress know exactly on what table the customer is sitting at.

In previous labs, hands-on projects, and midterm exam, you have already demonstrated your ability of creating:

- HTML pages (for content)
- CSS (for styling)
- JavaScript (for manipulating the DOM and event listeners)
- Responsive web design (for displaying your page in any screen size)

Due to that, in this final project, all the HTML, CSS, JavaScript, and Responsive web design implementation has been provided to you. Your main task is:

- to implement the server (Flask app) to provide your app endpoints
- add associated functionality to each endpoint
- to create a server database to store the orders
- deploy your app into PythonAnywhere server provider site
- create a QR code that would be placed on a table and that would access the main menu
- create a second QR code that displays the kitchen page, that shows the current orders to be processed

As project deliverables, you will need to submit a zip file that contains:

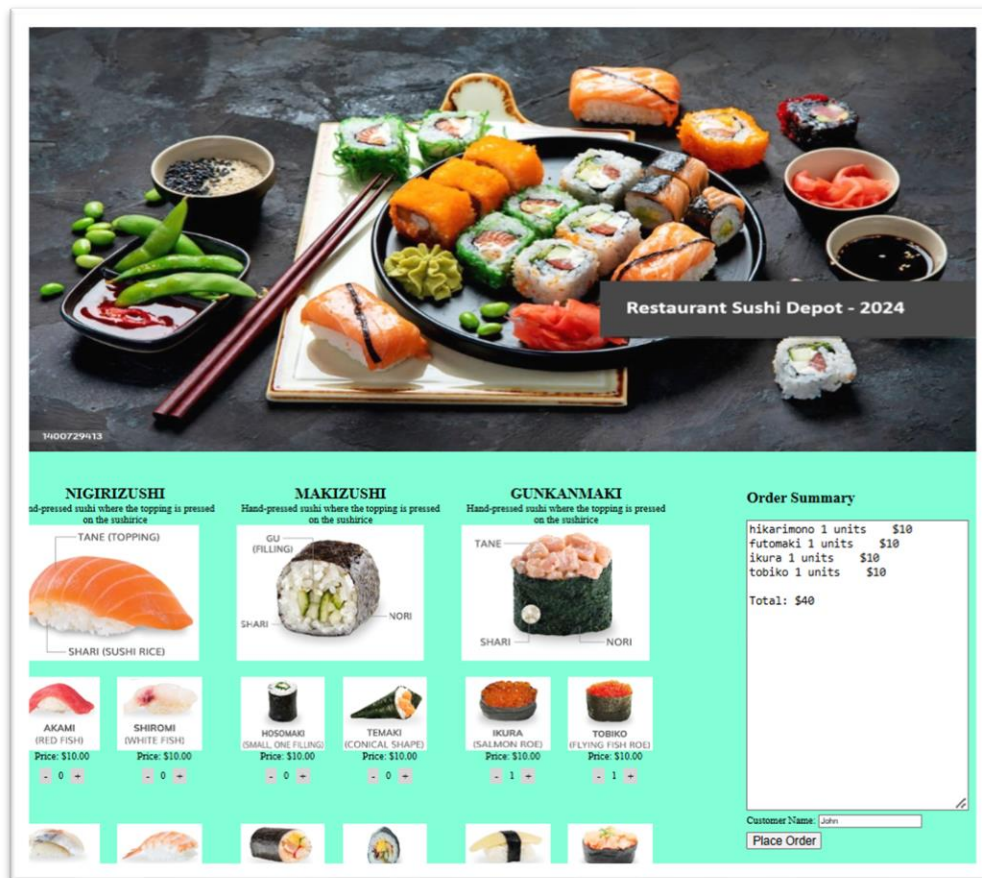
- all your project files
- a PDF file (or images) of the two QR codes that you will generate so that I can scan them with my phone and visit your pages.

2 The Project General Overview

As the App developer, you will need to create two “main” pages: one that will be displayed to the customer and another one that will be displayed to the kitchen staff.

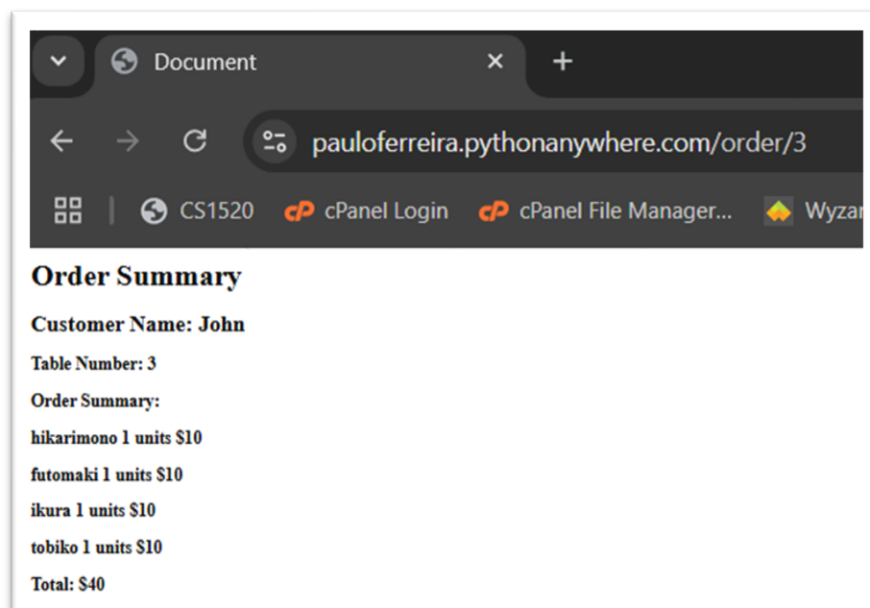
2.1 Main menu page

After the customer scans the QR code found on a given table, the following main menu should be displayed in their smartphone.



Note: I have added Responsive Web Design functionality on the main menu page, so, if you flip your smartphone to landscape or portrait orientation, you should see the different ways of displaying this page. Try it later on.

After choosing the food and adding a name to the order, the customer shall click on the “Place Order” button to have their order submitted. A confirmation screen must be presented to the customer, confirming that the order is being processed, as shown below.

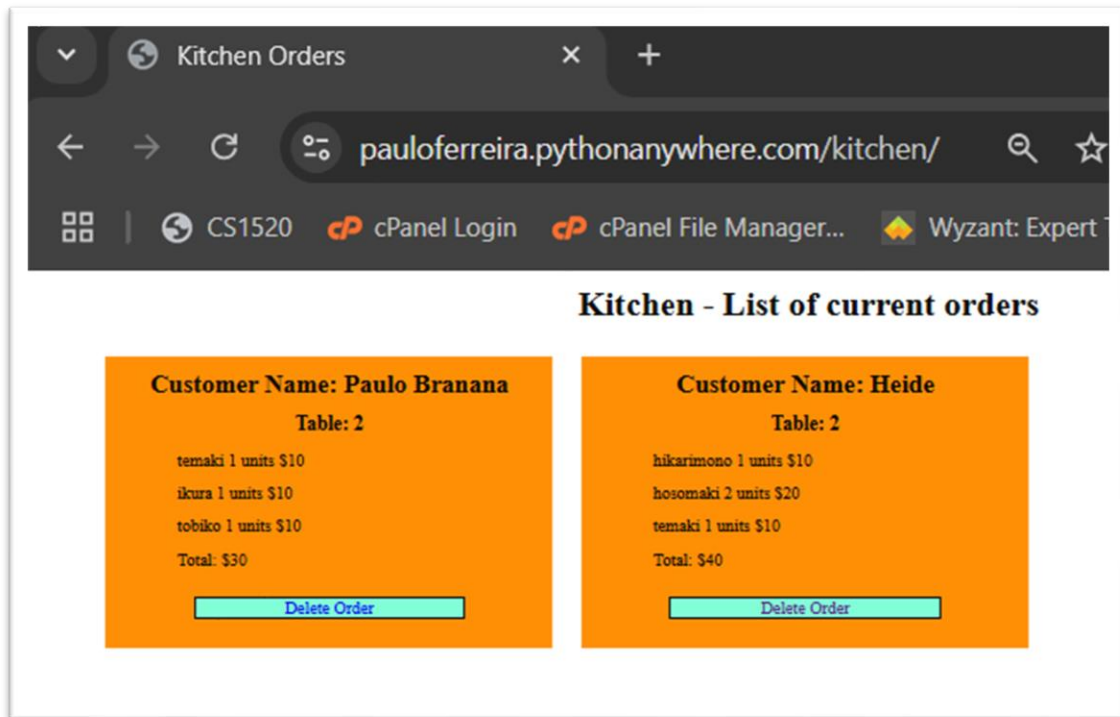


After about 15 seconds, the main menu page should be displayed automatically (for future customers).

During the ordering process, your application shall save the current order into a database, using Flask Models, as discussed in class.

2.2 The Kitchen Endpoint

In the kitchen, where the staff prepares the customer's order, there will be a monitor containing the list of orders that must be processed. Each order contains a description of the customer's name, the table they are sitting at, and an itemized description of what they have ordered.



The kitchen route accesses the database to retrieve and display the unfulfilled orders. This page will also allow the kitchen staff to delete a processed order by clicking the “delete order” button (this will take to another endpoint: /delete/<id>).



2.3 The QR codes

As “deliverables” of your final project, you will submit your project as a zip file and please include a PDF file with two QR codes: one for the Main Menu page and another for the Kitchen page. The QR code for the main menu must contain the table number in it:

Examples:

For the main menu: <https://pauloferreira.pythonanywhere.com/7>

For the kitchen page: <https://pauloferreira.pythonanywhere.com/kitchen/>

Main menu with table number	Kitchen
	

3 The project Skeleton

On the course canvas, you will find the final project skeleton, provided as a zip file. Upon extracting the files from within, you will find the following types of files:

- HTML files
- CSS files
- JavaScript files
- Image files

Note: strictly speaking, you should not change anything in the HTML, CSS, JavaScript, and Image files. The only exception is the `order_summary.js`. You will need to change the path provided in that file to point to your main menu page, not mine.

As discussed in our class, the HTML, CSS, and JavaScript files are supposed to be in their specific directories in a Flask application. Please move them accordingly.

4 The Restaurant Flask Application – General Steps

Even though you can develop your project entirely inside the PythonAnywhere web page IDE, I would strongly recommend you create your Flask application in your local computer first, using the VS Code IDE, so you can quickly debug errors and test your final application. It is faster this way. After you are sure that your application is working as expected, then you move it to pythonanywhere site.

In the next section of this document, I provide a guide on how you would need to implement the necessary steps to successfully create your project (kind of our hands-on “code inserts” that we are used to use in our project classes, but with no code, just directions on what is needed to implement everything required in your application).

Disclaimer: the following guidelines are not perfect, so you might need to fix things here and there, as you go along. I am open to any suggestions you can provide me with to improve it for future use.

4.1 Moving files to the correct folder

Before doing any code implementation, please move the HTML, CSS, JavaScript, and the images to their correct folders. Remember: Flask expects these files to be placed in specific directories, as discussed in class. The images, you can place under static directory.

4.2 Creating the basic restaurant app file

- a) Create the Flask file, such as restaurant_app.py
- b) Add the usual imports that we have used in our hands-on projects (our wait until they are needed)
- c) Create the “app” Flask object
- d) Create the first route of your application: the root route (“/”):
 - i) Pay attention: this route must accept the table number, so that the waitress will know what table the food is to be delivered.
 - ii) When rendering the html page, remember to pass the table number along, as an input parameter for the render_template function.
- e) Add the Python command to run your applicant.
- f) Run your application from VS Code terminal window and visit your page from the Browser. Note: remember, you will need to manually add the table number in the Browser URL input, since your root page needs that info. (e.g., <http://localhost.../7>, where 7 is the table number in this example).

4.3 Update the main_menu.html page

Update the <form> tag in the main_menu.html page:

- the action attribute should point to the “order” route with the table number passed along (you can use the url_for here with the additional input parameter for the table number.
- The “methods” attribute should be set to the specific HTTP request type (I will not say here which one is here, you figure out by yourself).

4.4 The order route

- a) Create the “order” route. This route must accept the table number as input parameter (so the waitress can deliver the food to the right table).
- b) Retrieve the order summary and customer name data from the HTTP Post request object.
- c) Create a “JSON” object (in reality, since we are in a Python program, it is a “dictionary” object) containing the customer’s name, order summary (an array of order items), and the table number.
- d) Render the order_summary.html page, with the JSON object as one of the inputs in the render_template function.

4.5 Update the order_summary.html file

- a) Update the order_summary.html file to accept the JSON object from the previous section. Use this object to extract the table number, customer name, and order items, and dynamically create the page content using Jinja2 commands.
- b) Run your application, submit an order and check if you get your order information correctly. If not, debug your code before proceeding.

4.6 Saving the current order into a database

- a) Create a model:
 - i) Create an order_model.py in the “models” directory.
 - ii) Add the necessary imports as discussed in class.
 - iii) Create a python “class” named Order.
 - iv) Add all the necessary columns to your model (ID, customer_name, order items (one single long string containing orders separated by comma, and table number).
- b) Linking the model to your main Flask application:
 - i) Add the necessary commands to link your database file (e.g., order.db) with your program (see our hands-on discussed in class) (engine, Base, Session, and session elements).
 - ii) At the top of the restaurant_app.py file, add the necessary imports for the sqlalchemy functionality
- c) Update the “order” route:
 - i) Create an instance (object) of the Order class with the necessary fields in the constructor.
 - ii) Using Try / Expect structure, save the order into the database.
- d) Testing your code:
 - i) Run your code, place an order, and visually check if your orders.db file has been created in the project’s directory.
 - ii) If you have not installed it yet, install the VS Code extension “SQL Viewer” and open the order.db file and check its contents (your order should be written there).
 - iii) If your program is not running as expected, debug your code before proceeding.

4.7 The kitchen endpoint

- a) Create a kitchen endpoint in your restaurant_app.py file. For this one, use “/kitchen/” (with a trailing forward slash, since your browser will probably add it even if you don’t type it in the browser.)
- b) Add a command to query the database to retrieve the current orders. Note: database queries return a list of “Order” objects, not “JSON” objects, and that’s ok, but you will need to convert them to JSON (dictionary) objects to be able to send them to the kitchen html page, via input argument of the render_template function.

- c) Create an empty array that will contain the converted JSON objects.
- d) Loop over the “Order” object array, convert each Order object into JSON object, and append it to the empty array created in item c.
- e) Place a command to render the kitchen.html page and include the JSON array as input.
- f) Update your kitchen.html to accept the array of JSON order objects and process each order, i.e., dynamically create order summaries in the kitchen page as shown in the Kitchen figure shown at the beginning of this document.
- g) Remember to include the delete option for each order. This is very similar to what we have discussed in our hands-on class project.

4.8 The delete endpoint

- a) Create the delete endpoint. Remember, the delete endpoint must accept the order ID number, such as “delete/2”, where 2 represents the order with ID set to 2. See example in our hands-on project.
- b) The implementation of the delete endpoint is very similar to what we have done in our hands-on project. Just follow that to implement your functionality here.

5 Running your application locally

From the VS Code IDE, run your Flask application, place a few orders, and check if the kitchen is getting them. (Remember, you need to update a path in the kitchen.js, or you will be seeing my kitchen, not yours!).

If there is any error in your program implementation, fix it before proceeding to the next step.

6 Deploying your application into a web server provider

We will be using the free tear of pythonanywhere web site to host your application. Please watch the intro video on how to do it found on our course canvas.

6.1 Create your pythonanywhere account

Nothing much here, just create one.

6.2 Copy the project files

Copy the project files from your computer to the web server location. Create directories as needed.

6.3 Few Necessary Updates

Pythonanywhere server runs your Flask application in a way a little bit different from what is done when running it in your your computer. Some changes to your original code are necessary.

- a) Remove the run Python commands from your restaurant_app.py program. Pythonanywhere will run your program using another way.

```
if __name__ == "__main__":  
    app.run(debug=True)
```

- b) Let PythonAnywhere know the name of your python program: Since we have removed the command to run your application, we need to tell PythonAnywhere the name of your application, so it can run it.
 - i) Go to “Web” tab.
 - ii) Click on the WWSGI configuration file link.
 - iii) Replace the current program name with your python file name (without the .py)
from <your_app_name_here> import app as application
- c) You don’t need to install Flask library; if you followed the tutorial video found on Canvas, you have picked the Flask framework already. However, you will need to install the flask-sqlalchemy library yourself:
 - i) Go to “Consoles” tab.
 - ii) Run a Bash terminal window.
 - iii) Type “pip install flask-sqlalchemy” in the terminal window.
- d) Define the templates and static directories: When running your Flask application on your computer, Flask assumes that the HTML files are stored in “templates” directory and your JavaScript and CSS files are stored in the “static” directory. However, in pythonanywhere you need to explicitly state that. Make the change shown below in your restaurant_app.py file:

Replace the line: app = Flask("__name__")

With: app = Flask("__name__",
 template_folder="/home/<yourusernamehere>/mysite/templates",
 static_folder="/home/<yourusernamehere>/mysite/static")

6.4 Run your application from PythonAnywhere

- a) Run your application from PythonAnywhere.
- b) Visit your site from your smartphone
- c) Place a few orders and check if for each one, the order_summary page displays it correctly (including the table number).
- d) Open the kitchen page and see if your order is also there.
- e) In the kitchen page, delete an order and check if this page updates correctly.

7 Creating your QR codes



Create QR codes for your main page (with a table number in it) and the kitchen. Just use any free site to generate these QR codes.

Examples of my URL addresses:

<https://pauloferreira.pythonanywhere.com/7>
<https://pauloferreira.pythonanywhere.com/kitchen/>

Note: do not forget to put the forward slash at the end of the kitchen

With the URLs shown above, I got the following QR codes.

Main menu with table number	Kitchen
	

Note: a small square area right in the center of the QR code can be replaced with an image. After the site generated the QR code for me, I copied it to Paintbrush application and added the logos shown above.

Your two QR codes must be submitted to me along with the zip file of your project.

Good luck!

Paulo Brasko