Assignment 5: Chilly Delights REST API Server

Context/Scenario

Chilly Delights is a new ice cream shop that sells several specialties. The manager would like to have a simple website that displays the ice cream menu to customers, as well as manage additions, deletions, and updates to the menu items. You have been called upon to build the server side for Chilly Delights' website.

Instructions:

The assignment consists of two parts.

Part 1 – Base Code: The task for this part is completed offline and represents the base of your code that you will add to in Part 2. The code for Part 1 must be submitted as a separate file.

Part 2 – Skill Demonstration:

The task for this part will be recorded **as completed**. Note that in your recording it is not sufficient to only write the code, but you must also explain the logic and describe how your logic is being implemented. In addition, your recording must show a sample run that demonstrates the correct execution of your code.

Important Guidelines:

- The recording **should not exceed 17 minutes** in duration. Otherwise, the recording will not be considered.
- The code for Part 1 should be the starting point for your implementation of Part 2.
- The code for Part 2 must be developed during the recording. Ready-made code will not be accepted.
- To ensure that your recording will be accepted for grading, it is essential to show both your face and your screen while demonstrating the development steps of the code and explain each step as it is developed.
- Use Kaltura software for recording your video. Kaltura is freely offered by GMU. More information can be found here
- Your presentation needs to be clear, concise, and effectively communicate the steps and logic of your solution.
- Use the grading rubrics as a guide to prepare your presentation.
- Using Frameworks and/or advanced techniques not covered in class will result in a 0 grade.
- Providing code that is copied from ChatGPT and reading the accompanying explanation generated by ChatGPT will not be accepted as a valid skill demonstration for this assignment. This practice will result in a grade of 0.
- Resubmission of the assignment after the due date is not allowed.

Please ensure that your recording adheres to these guidelines to receive full credit for your submission.

Assignment Description:

Part 1 – Base Code: CRUD Operations:

Ice Cream Menu structure

The server will use file-based data storage to store the menu. The menu items will be stored as a JSON list. Each ice cream specialty on the menu will be represented as a JSON object. An ice-cream specialty has the following attributes:

- code: A string of digits that **uniquely identifies** the item.
- name: a string that represents the name of the ice cream specialty.
- ingredients: A list of strings representing the ingredients of the ice cream specialty.
- price: A number representing the price of the ice cream scoop.
- availability: A Boolean indicating whether the item is available or has been sold out.

Server Endpoints

The following endpoints are defined for the server:

1. **/welcome**:
   - **Description**: Displays a welcome message to the client. This endpoint will be the ***default route*** for the server and will implement only the GET operation.
   - **Details**: The server will retrieve a welcome text message from a text file and send it to the client using streams and pipes, after appropriately setting response headers. The server needs to handle all errors on this path.

2. **/menu - /menu/{code}:**
   - **Description:** Implements the CRUD operations on the ice cream menu. Operations will be sent to the server using POSTMAN. The server code needs to handle all errors on this path for all CRUD operations.
   - **CRUD Operations:**
     a. Create:
     Endpoint: /menu or /menu/
     Details: When the server starts for the first time, it should not assume that the menu file exists. The file should be created with the first POST request. When the server receives a POST request, it parses the query string of the request to get the details of the ice cream specialty and adds it to the menu file. The server needs to ensure that each item added to the menu is unique and should not allow duplication. After the item is added, the server needs to reply with status code 200 and the string 'OK' to the client. If the item already exists in the menu, the server needs to reply with status code 400 and the string 'BAD REQUEST' to the client.

     b. Read:
     Endpoint: /menu or /menu/
     Details: When the server receives a GET request, it will read the menu file and send it to the client using streams and pipes after appropriately setting response headers. The server needs to reply with status code 200 and a JSON list representing the menu. If the menu file does not exist, the server needs to reply with the status code 404 and the string 'Not Found' to the client.

c. Update:
Endpoint: /menu/{code} or /menu/{code}/
Description: Updates one of the ice cream items identified by {code}, where {code} is the ice cream code that uniquely identifies the ice cream.
Details: For an ice cream specialty, three attributes can be updated: price, ingredients, and availability, or any combination thereof. It can be assumed that only these three attributes will be sent in the client request. For each update operation, the menu will be retrieved from the JSON file, the specified attribute of the ice cream specialty will be updated, and then the menu will be saved back to the file. After the item is updated, the server needs to reply with status code 200 and the string 'OK' to the client. If the menu file does not exist, or the item to be updated does not exist in the menu, the server needs to reply with the status code 404 and the string 'Not Found' to the client.

d. Delete:
Endpoint: /menu/{code} or /menu/{code}/
Description: Deletes one of the ice cream items identified by {code}, where {code} is the ice cream code that uniquely identifies the ice cream.
Details: The manager at Chilly Delights might decide to stop serving one of the ice cream specialties on the menu. Accordingly, the server needs to allow deletions from the menu. For deletions, the ice cream specialty to be deleted is identified by its code. The server will retrieve the menu, delete the specified item, and then store the menu back to the file. After the item is deleted, the server needs to reply with status code 200 and the string 'OK' to the client. If the menu file does not exist, or the item to be deleted does not exist in the menu, the server needs to reply with the status code 404 and the string 'Not Found' to the client.

Setup and Implementation Hints for Part 1:

1. Use the code developed in Lab 8 as a reference.
2. The welcome file is stored in the server working path.
3. Provide two helper functions for retrieving and storing the menu. Use these functions for POST, DELETE and PUT operations.
4. Use streams and pipes for GET method.
5. Minimize the nesting of callbacks
6. Provide code to catch any invalid path under '/menu' and respond with '400 BAD REQUEST'
7. Develop the server code incrementally and always check that the response is received by POSTMAN.

Test Cases for Part 1:

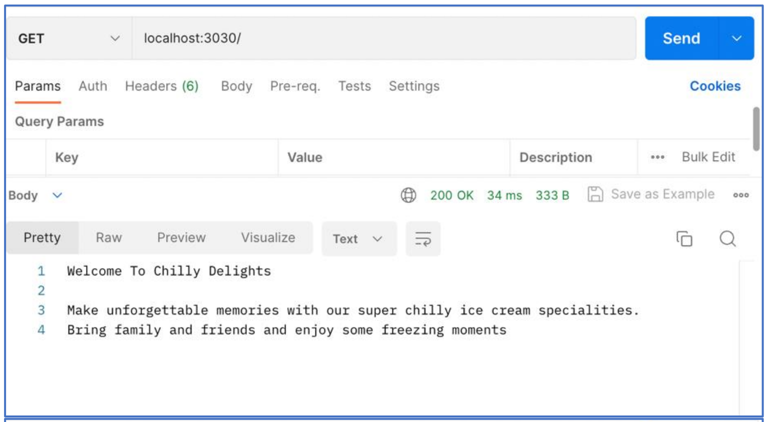
Following are values for the 'key:value' pairs used for inserting ice cream items in the menu for the POST method.

code: 10233 *name*: Honey Dew Sorbet *ingredient*: ['Honey Dew']	*code*: 10234 *name*: Banana Pudding	*code*: 10345 *name*: Strawberry Lemonade Sorbet

price: 17 *avail*: true	*ingredient*: [‘Banana’, ‘Vanilla’, ‘Wafer Cookies’] *price*: 15 *avail*: true	*ingredient*: [‘Lemonade’, ‘Organic Strawberries’] *price*: 12 *avail*: true
code: 10232 *name*: Coconut Almond Brownie *ingredient*: [‘Coconut’, ‘Almond’, ‘Brownie’] *price*: 17 *avail*: true	*code*: 10202 *name*: Black Raspberry Chip *ingredient*: [‘Black Raspberry’, ‘Chocolate’] *price*: 13 *avail*: true	*code*: 10237 *name*: Summer Peach Melba *ingredient*: [‘Peach’, ‘Raspberry’] *price*: 18 *avail*: true

The snapshots provided below are a sample of the expected responses received on POSTMAN. For the following test cases, it is assumed that the server is running on port 3030.

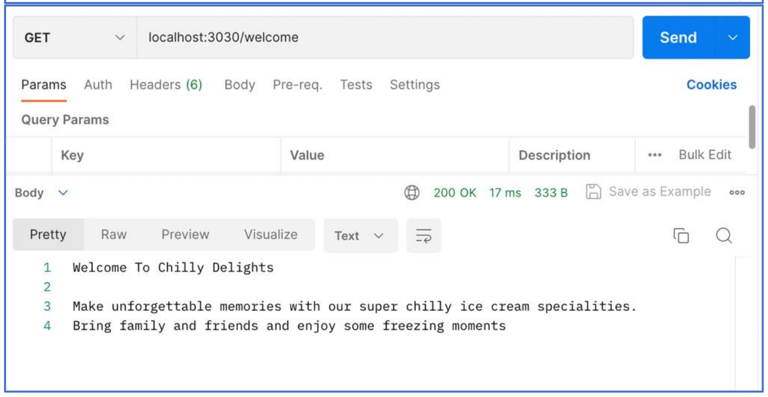
/welcome Endpoint: Note that with GET method, regardless of the pathname provided after the ‘/’ the server defaults to the welcome message.



GET localhost:3030/

Body 200 OK 34 ms 333 B

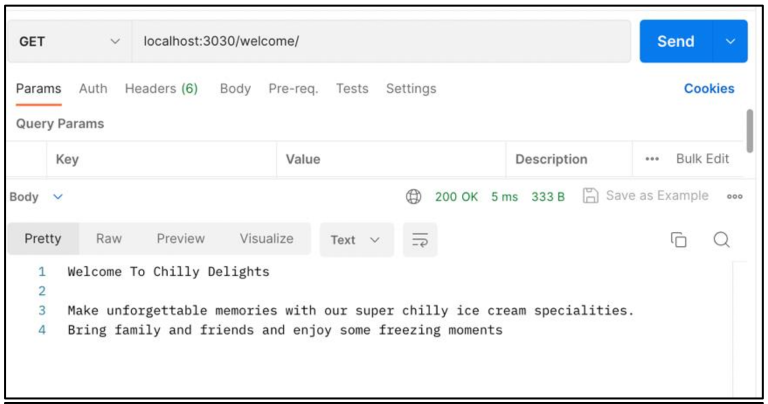
```
Welcome To Chilly Delights

Make unforgettable memories with our super chilly ice cream specialities.
Bring family and friends and enjoy some freezing moments
```



GET localhost:3030/welcome

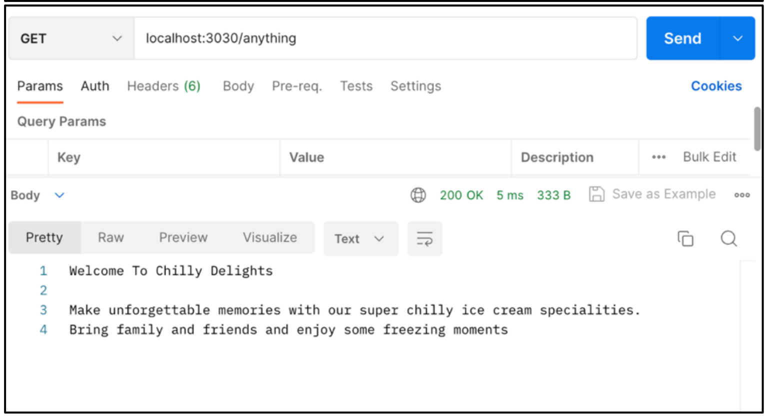
Body 200 OK 17 ms 333 B

```
Welcome To Chilly Delights

Make unforgettable memories with our super chilly ice cream specialities.
Bring family and friends and enjoy some freezing moments
```



localhost:3030/welcome/

```
Welcome To Chilly Delights

Make unforgettable memories with our super chilly ice cream specialities.
Bring family and friends and enjoy some freezing moments
```

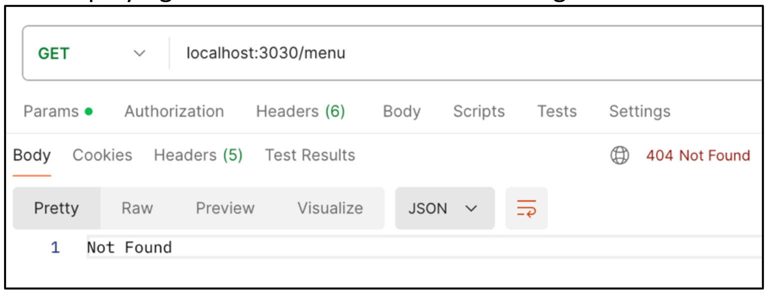


localhost:3030/anything

```
Welcome To Chilly Delights

Make unforgettable memories with our super chilly ice cream specialities.
Bring family and friends and enjoy some freezing moments
```

/menu Endpoint: The server accepts '/menu' or '/menu/' as the pathname for the POST and GET operations.

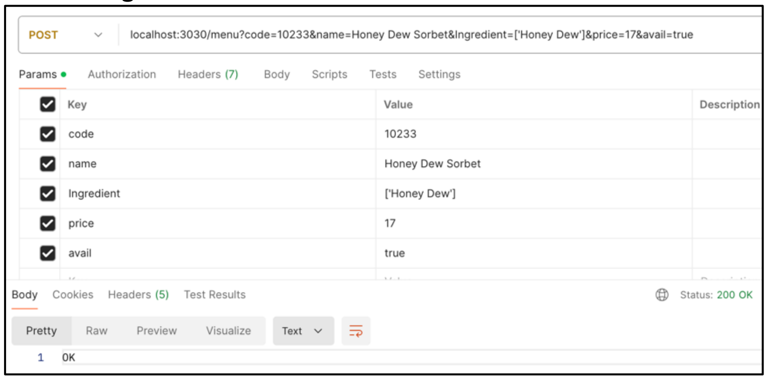
- Displaying the menu before the menu file gets created



localhost:3030/menu

```
Not Found
```

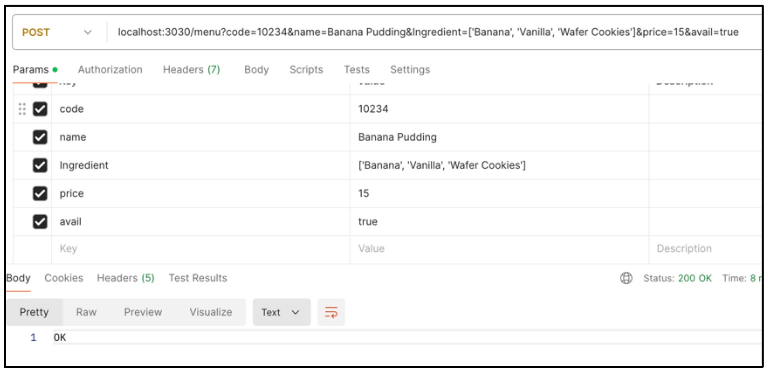
- Adding items to the menu



localhost:3030/menu?code=10233&name=Honey Dew Sorbet&Ingredient=['Honey Dew']&price=17&avail=true

Key	Value	Description
code	10233	
name	Honey Dew Sorbet	
Ingredient	['Honey Dew']	
price	17	
avail	true	

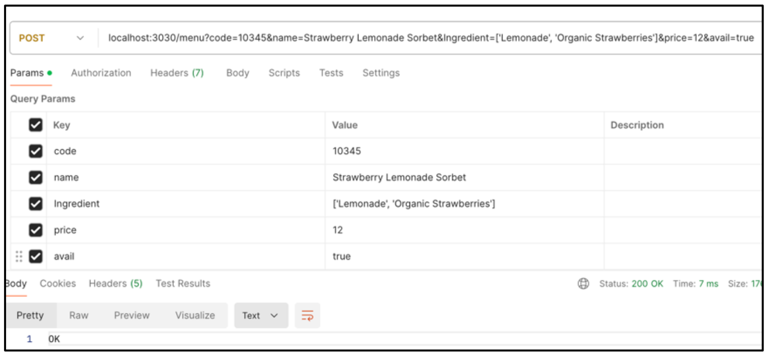
```
OK
```


POST localhost:3030/menu?code=10234&name=Banana Pudding&Ingredient=['Banana', 'Vanilla', 'Wafer Cookies']&price=15&avail=true

Key	Value	Description
code	10234	
name	Banana Pudding	
Ingredient	['Banana', 'Vanilla', 'Wafer Cookies']	
price	15	
avail	true	

Status: 200 OK

```
OK
```


POST localhost:3030/menu?code=10345&name=Strawberry Lemonade Sorbet&Ingredient=['Lemonade', 'Organic Strawberries']&price=12&avail=true

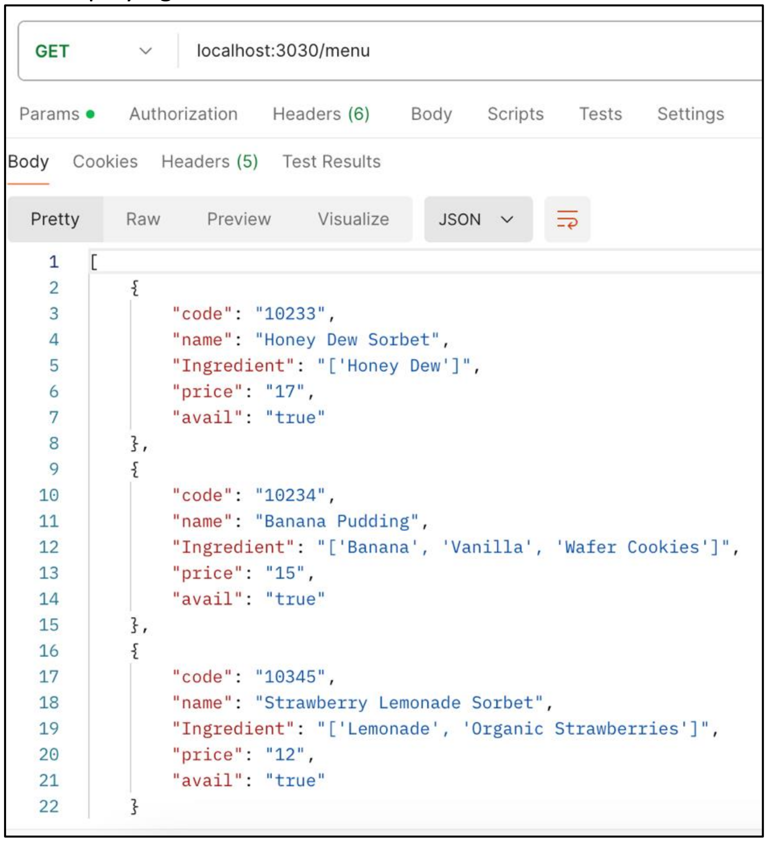
Query Params

Key	Value	Description
code	10345	
name	Strawberry Lemonade Sorbet	
Ingredient	['Lemonade', 'Organic Strawberries']	
price	12	
avail	true	

Status: 200 OK Time: 7 ms

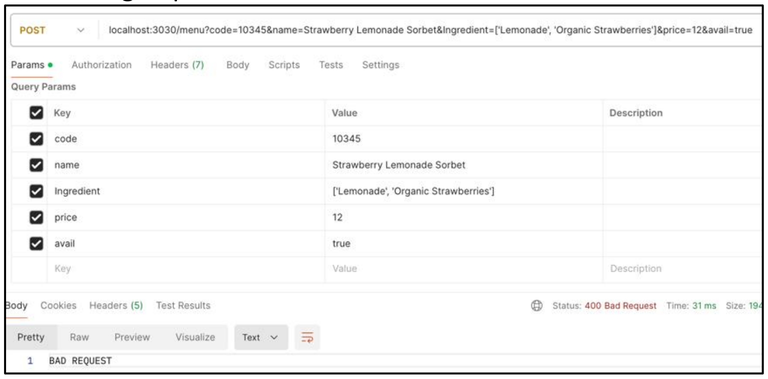
```
OK
```

- Displaying the menu


GET localhost:3030/menu

```
[
    {
        "code": "10233",
        "name": "Honey Dew Sorbet",
        "Ingredient": "['Honey Dew']",
        "price": "17",
        "avail": "true"
    },
    {
        "code": "10234",
        "name": "Banana Pudding",
        "Ingredient": "['Banana', 'Vanilla', 'Wafer Cookies']",
        "price": "15",
        "avail": "true"
    },
    {
        "code": "10345",
        "name": "Strawberry Lemonade Sorbet",
        "Ingredient": "['Lemonade', 'Organic Strawberries']",
        "price": "12",
        "avail": "true"
    }
```

- Adding duplicates to the menu



POST localhost:3030/menu?code=10345&name=Strawberry Lemonade Sorbet&Ingredient=['Lemonade', 'Organic Strawberries']&price=12&avail=true

Query Params

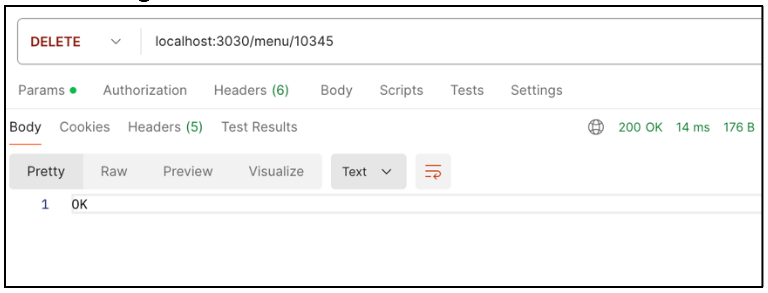
Key	Value	Description
code	10345	
name	Strawberry Lemonade Sorbet	
Ingredient	['Lemonade', 'Organic Strawberries']	
price	12	
avail	true	

Status: 400 Bad Request Time: 31 ms

```
BAD REQUEST
```

/menu/{code} Endpoint: The server accepts '/menu/{code}' or '/menu/{code}/' as the pathname for the PUT and DELETE operations.

- Deleting an item from the menu

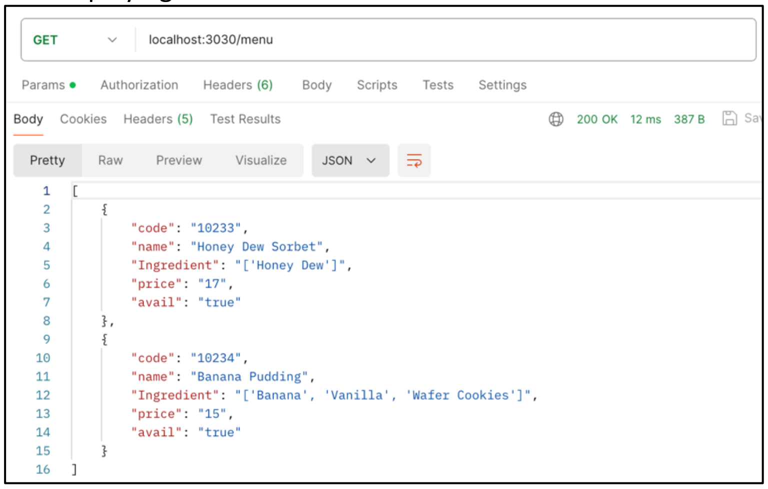


DELETE localhost:3030/menu/10345

200 OK 14 ms 176 B

```
OK
```

- Displaying the menu

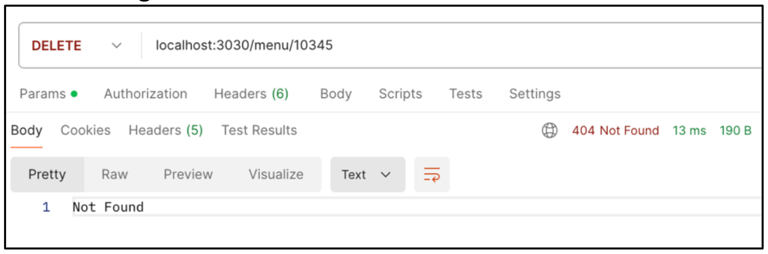


GET localhost:3030/menu

200 OK 12 ms 387 B

```
[
    {
        "code": "10233",
        "name": "Honey Dew Sorbet",
        "Ingredient": "['Honey Dew']",
        "price": "17",
        "avail": "true"
    },
    {
        "code": "10234",
        "name": "Banana Pudding",
        "Ingredient": "['Banana', 'Vanilla', 'Wafer Cookies']",
        "price": "15",
        "avail": "true"
    }
]
```

- Deleting a nonexistent item from the menu

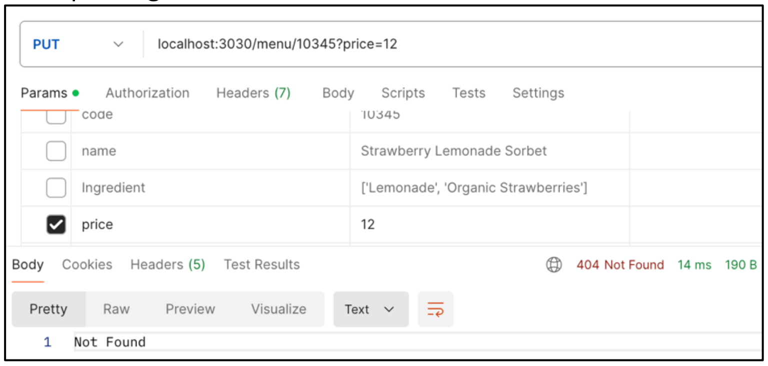


DELETE localhost:3030/menu/10345

404 Not Found 13 ms 190 B

```
Not Found
```

- Updating a nonexistent item from the menu



```
PUT  localhost:3030/menu/10345?price=12
```

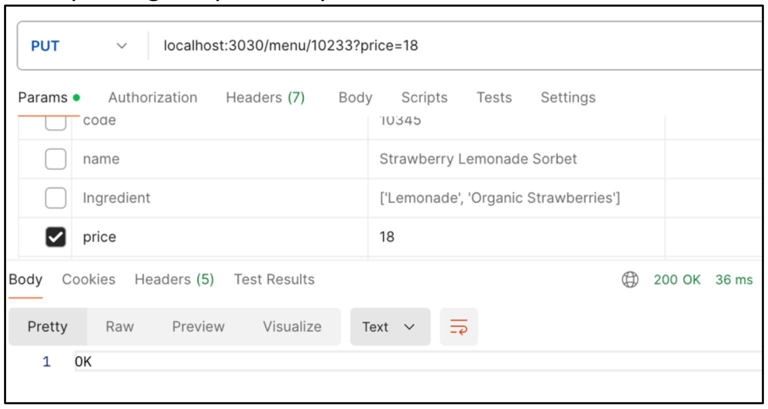
Params ● Authorization Headers (7) Body Scripts Tests Settings

☐	code	10345
☐	name	Strawberry Lemonade Sorbet
☐	Ingredient	['Lemonade', 'Organic Strawberries']
☑	price	12

Body Cookies Headers (5) Test Results — 404 Not Found 14 ms 190 B

```
Not Found
```

- Updating the price only



```
PUT  localhost:3030/menu/10233?price=18
```

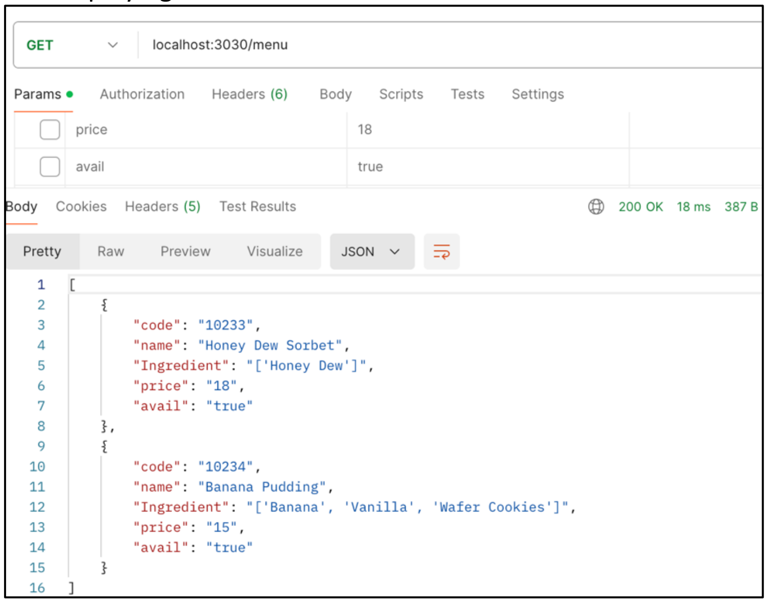
Params ● Authorization Headers (7) Body Scripts Tests Settings

☐	code	10345
☐	name	Strawberry Lemonade Sorbet
☐	Ingredient	['Lemonade', 'Organic Strawberries']
☑	price	18

Body Cookies Headers (5) Test Results — 200 OK 36 ms

```
OK
```

- Displaying the menu



```
GET  localhost:3030/menu
```

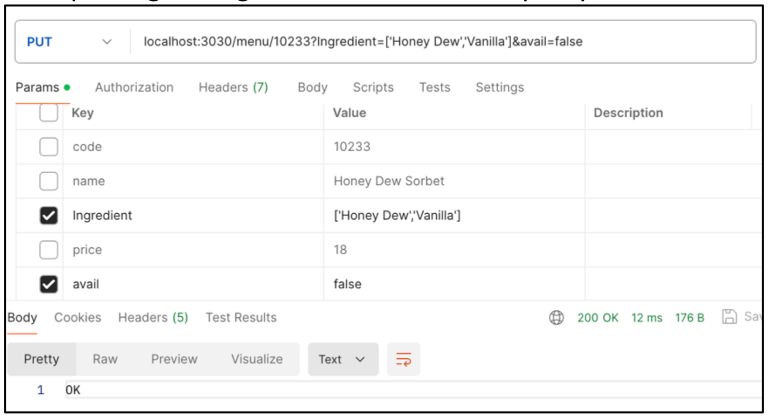
Params ● Authorization Headers (6) Body Scripts Tests Settings

☐	price	18
☐	avail	true

Body Cookies Headers (5) Test Results — 200 OK 18 ms 387 B

```
[
    {
        "code": "10233",
        "name": "Honey Dew Sorbet",
        "Ingredient": "['Honey Dew']",
        "price": "18",
        "avail": "true"
    },
    {
        "code": "10234",
        "name": "Banana Pudding",
        "Ingredient": "['Banana', 'Vanilla', 'Wafer Cookies']",
        "price": "15",
        "avail": "true"
    }
]
```

- Updating the ingredients and availability only



PUT localhost:3030/menu/10233?Ingredient=['Honey Dew','Vanilla']&avail=false

Params ● Authorization Headers (7) Body Scripts Tests Settings

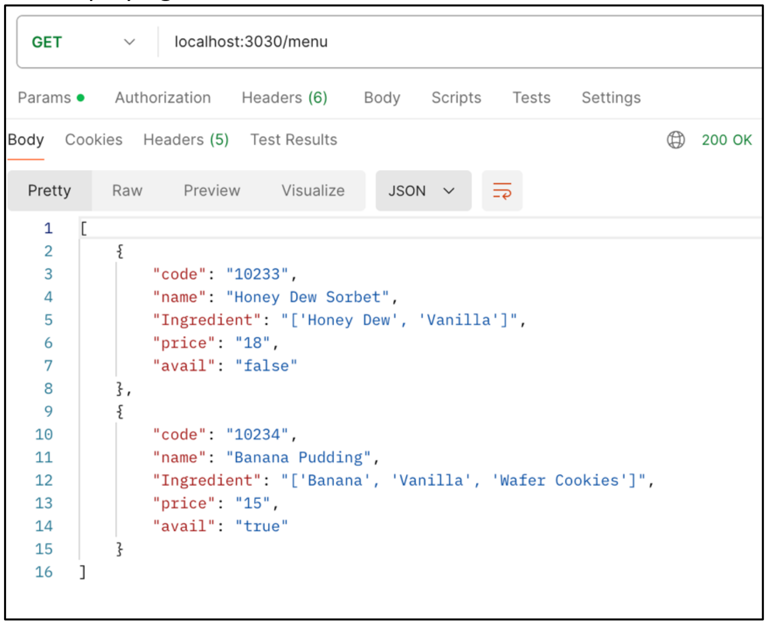
	Key	Value	Description
☐	code	10233	
☐	name	Honey Dew Sorbet	
☑	Ingredient	['Honey Dew','Vanilla']	
☐	price	18	
☑	avail	false	

Body Cookies Headers (5) Test Results 200 OK 12 ms 176 B

Pretty Raw Preview Visualize Text

```
OK
```

- Displaying the menu



GET localhost:3030/menu

Params ● Authorization Headers (6) Body Scripts Tests Settings

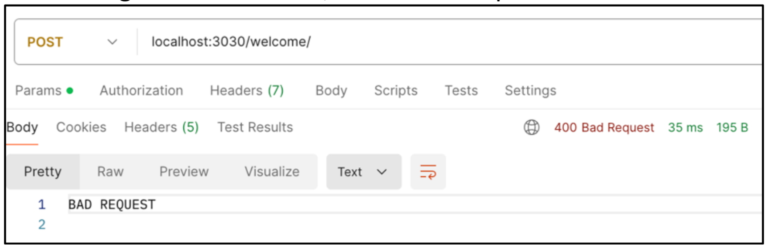
Body Cookies Headers (5) Test Results 200 OK

Pretty Raw Preview Visualize JSON

```
[
    {
        "code": "10233",
        "name": "Honey Dew Sorbet",
        "Ingredient": "['Honey Dew', 'Vanilla']",
        "price": "18",
        "avail": "false"
    },
    {
        "code": "10234",
        "name": "Banana Pudding",
        "Ingredient": "['Banana', 'Vanilla', 'Wafer Cookies']",
        "price": "15",
        "avail": "true"
    }
]
```

Invalid Endpoints: The following are *examples* for the server reply for invalid endpoints

- Wrong method on the /welcome endpoint



POST localhost:3030/welcome/

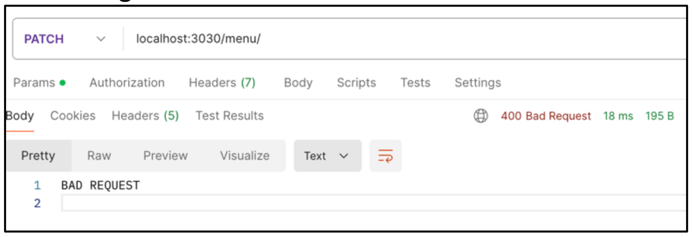
Params ● Authorization Headers (7) Body Scripts Tests Settings

Body Cookies Headers (5) Test Results 400 Bad Request 35 ms 195 B

Pretty Raw Preview Visualize Text

```
BAD REQUEST

```

- Wrong method



PATCH localhost:3030/menu/

Params ● Authorization Headers (7) Body Scripts Tests Settings

Body Cookies Headers (5) Test Results 400 Bad Request 18 ms 195 B

Pretty Raw Preview Visualize Text

```
BAD REQUEST

```

Part 2 – Skill Demonstration: Adding a New Route

The manager at Chilly Delights would like to add the functionality of displaying an ice cream item by its code. Accordingly, you are required to **add a new route** to provide this functionality. The new route description and details are given below.

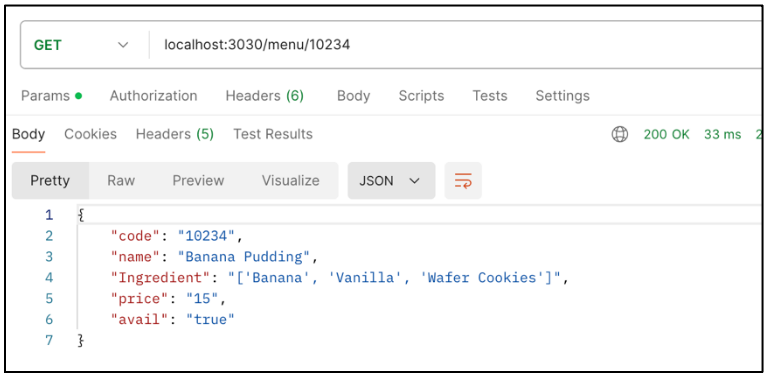
/menu/{code}:

- **Description:** Displays one of the ice cream items identified by {code}, where {code} is the ice cream code that uniquely identifies the ice cream.
- **Details**: For this endpoint, the server will retrieve the menu from the file, locate the ice cream item by its code, and respond to the client with status code 200 and a JSON object representing the ice cream item. If the menu file does not exist, or the item to be displayed does not exist in the menu, the server needs to reply with the status code 404 and the string 'Not Found' to the client.

Test Cases for Part 2:

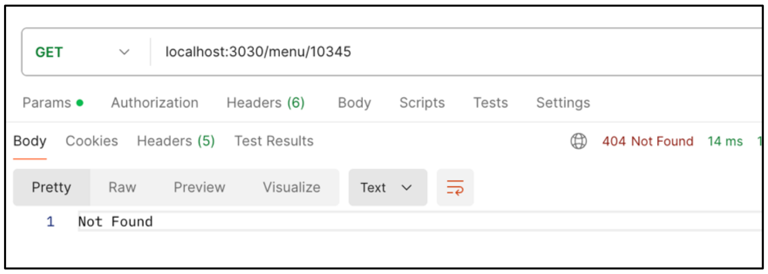
/menu/{code} Endpoint: The server accepts '/menu/{code}' or '/menu/{code}/' as the pathname for the GET method in addition to the PUT and DELETE methods.

- Displaying an ice cream item by code



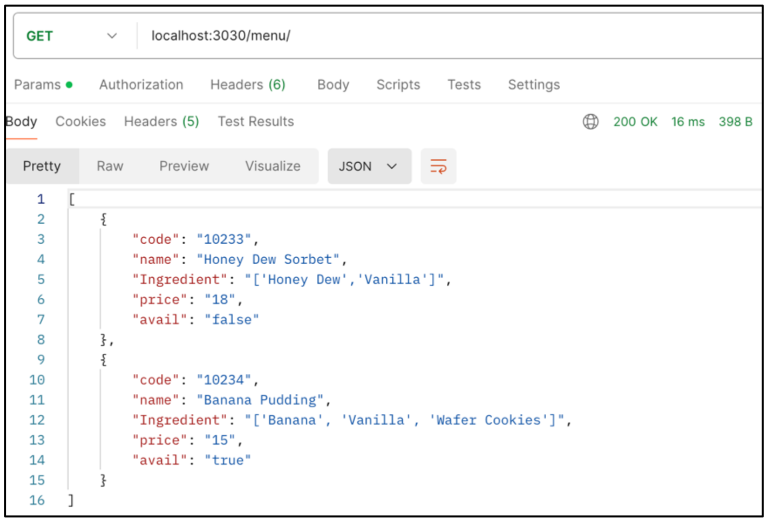
```
{
    "code": "10234",
    "name": "Banana Pudding",
    "Ingredient": "['Banana', 'Vanilla', 'Wafer Cookies']",
    "price": "15",
    "avail": "true"
}
```

- Displaying a nonexistent ice cream item by code

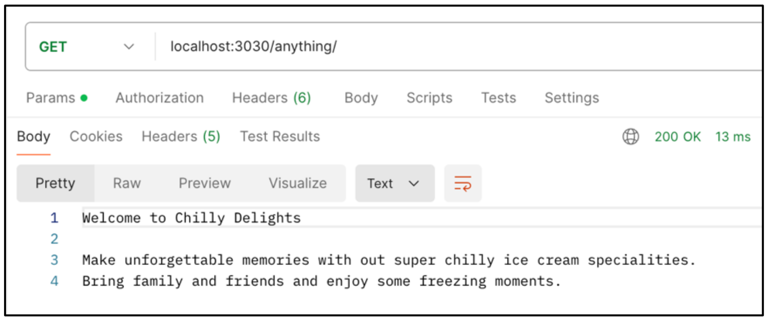


```
Not Found
```

The server still serves the /menu endpoint and the default endpoint for the GET method



```
[
    {
        "code": "10233",
        "name": "Honey Dew Sorbet",
        "Ingredient": "['Honey Dew','Vanilla']",
        "price": "18",
        "avail": "false"
    },
    {
        "code": "10234",
        "name": "Banana Pudding",
        "Ingredient": "['Banana', 'Vanilla', 'Wafer Cookies']",
        "price": "15",
        "avail": "true"
    }
]
```



```
Welcome to Chilly Delights

Make unforgettable memories with out super chilly ice cream specialities.
Bring family and friends and enjoy some freezing moments.
```

Deliverables

JS Files: Submit all your .js files as one zipped folder. Your folder will include the following:

- A .js file for part 1
- A .js file for part 2

Video Recording: Submit your video recording using the 'Create Submission' option and upload your video from Kaltura. More information can be found here

Grading Rubrics:

This assignment is graded out of 120 points

Part 1: 60 Points

Criteria	Points
Comment describing the task for Part 1 at the beginning of the file	**3 points**
POST Handler	**10 points**
- Loading menu	2 points
- Identifying duplicates	4 points
- Inserting new item	2 points
- Saving menu	2 points
DELETE Handler	**10 points**
- **Loading menu**	**2 points**
- **Identifying item by code**	**4 points**
- **Removing item**	**2 points**
- **Saving menu**	**2 points**
PUT Handler	**12 points**
- **Loading menu**	**2 points**
- **Identifying item by code**	**4 points**
- **Updating item**	**4 points**
- **Saving menu**	**2 points**
GET Method	**10 points**
- Handling the default route	**4 points**
- Setting up a stream	**2 points**
- Transferring using pipe	**2 points**
- Handling errors	**4 points**
Request Handler	**12 points**
- **Parsing URL to extract required information**	**6 points**
- **Setting up routing logic**	**10 points**
- **Handling HTTP methods**	**16 points** $(4\ methods \times 4points)$
- **Handling invalid routes or methods**	**10 points** $(5\ cases \times 2points)$
Creating Server and listening on Port	**3 points**
Total Part 1	**60 points**

Part 2: 60 Points

Criteria	Points
Comment describing the task for Part 2 at the beginning of the file	**5 points**
Overview of Part 1 Code	**15 points**
- Describe the implementation of the Default route	5 points
- Describe the code for identifying the duplicate in POST	5 points

- Describe the code for identifying the item in either DELETE or UPDATE (not both)	5 points
Adding New Route for GET Method	**25 points**
- Implement and explain the implementation of the GET Handler (*Copying and pasting code from other handlers in the code is allowable*)	15 points
- Implement and explain the updates done for the GET method in the Request Handler (*Copying and pasting code from other parts in the routing logic is allowable*)	10 points
Demonstrating the Sample Run using POSTMAN	**15 points**
- Display all items in menu using the GET method	5 points
- Display item by code using the GET method	5 points
- Display message for nonexistent code for GET method	5 points
Total Part 2	**60 points**