#Assignment 2
##A simple 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 web site 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.

##Directions

In this assignment you will be using file-based data storage. The menu items will be stored as a JSON file. 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: the price of a scoop
- Availability: whether the item is available or has been sold out.

The server needs to handle two routes. The first route is a welcome route for displaying a welcome message to the users, and the second route is the menu route to access/operate on the menu.

The welcome route will be the default route for the server. It will only implement the GET operation. The server will retrieve the welcome message from a text file and will respond to the client with the welcome message of a status code of 200.

For the menu route the server needs to implement the CRUD operations. For each operation, the menu needs to be retrieved from the .json file, manipulated according to the required operation, and then stored again in the .json file. Operations will be sent to the server using POSTMAN. The description of each operation on the menu route is as follows:

1. Create: when the server starts for the first time, it *should not assume* that the menu file *exists*, but rather the file should be created with the first POST command. When the server receives a POST command, it parses the query string of the request to get the details of the ice cream specialty and adds the ice cream specialty 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 to the menu the server needs to reply with status code 200 and the string ‘OK’ to the client.

2. Read: When the server receives a GET operation, it will read the menu file to retrieve all menu items and send them as a *JSON string* along with a status code of 200 in the response to the client.

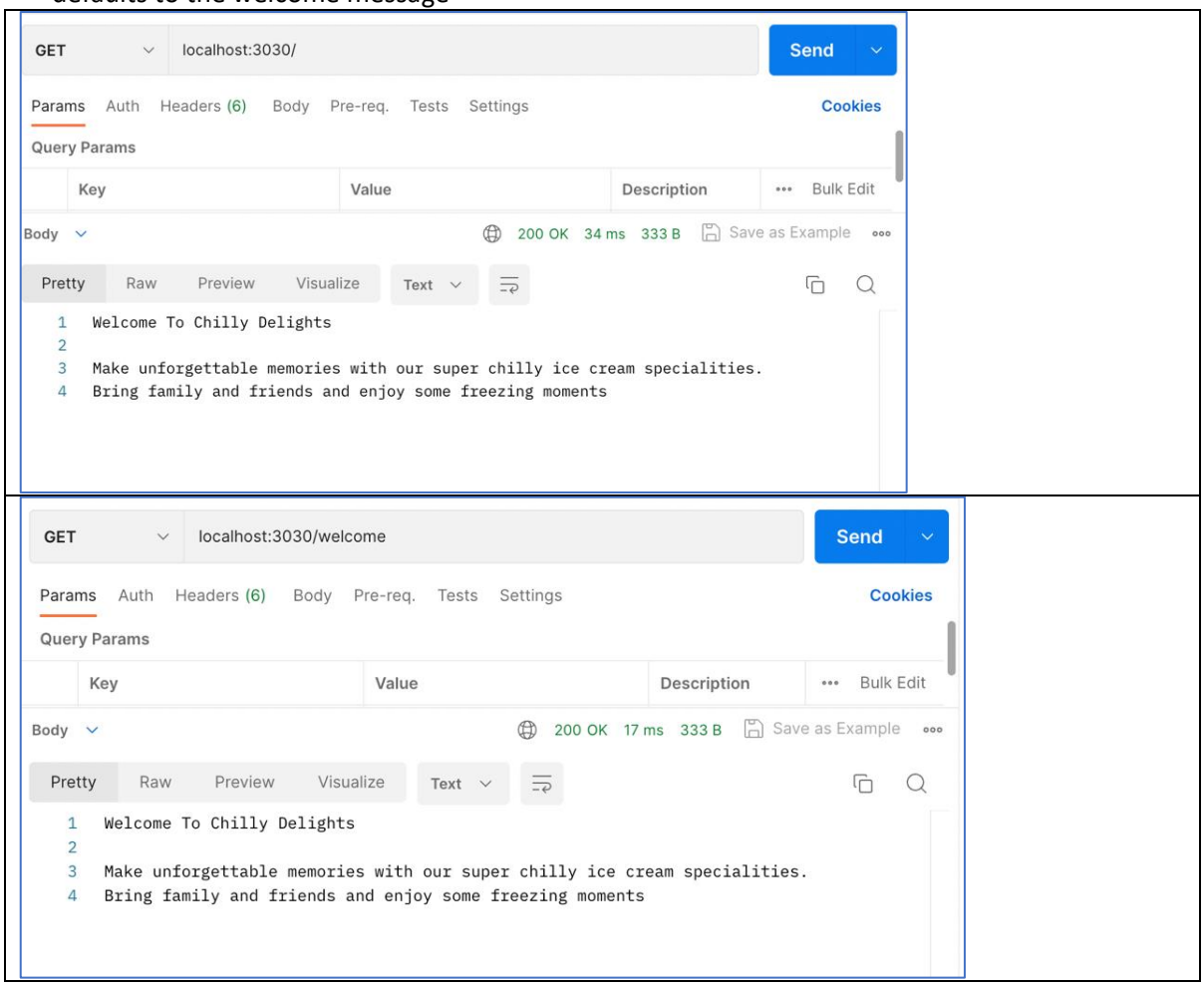
3. Update: For an ice cream specialty, three attributes can be updated: price, ingredients, and availability. For each update operation, the menu will be retrieved from the json file, the specified attribute of the ice cream specialty is updated and then the menu is saved back to the file.

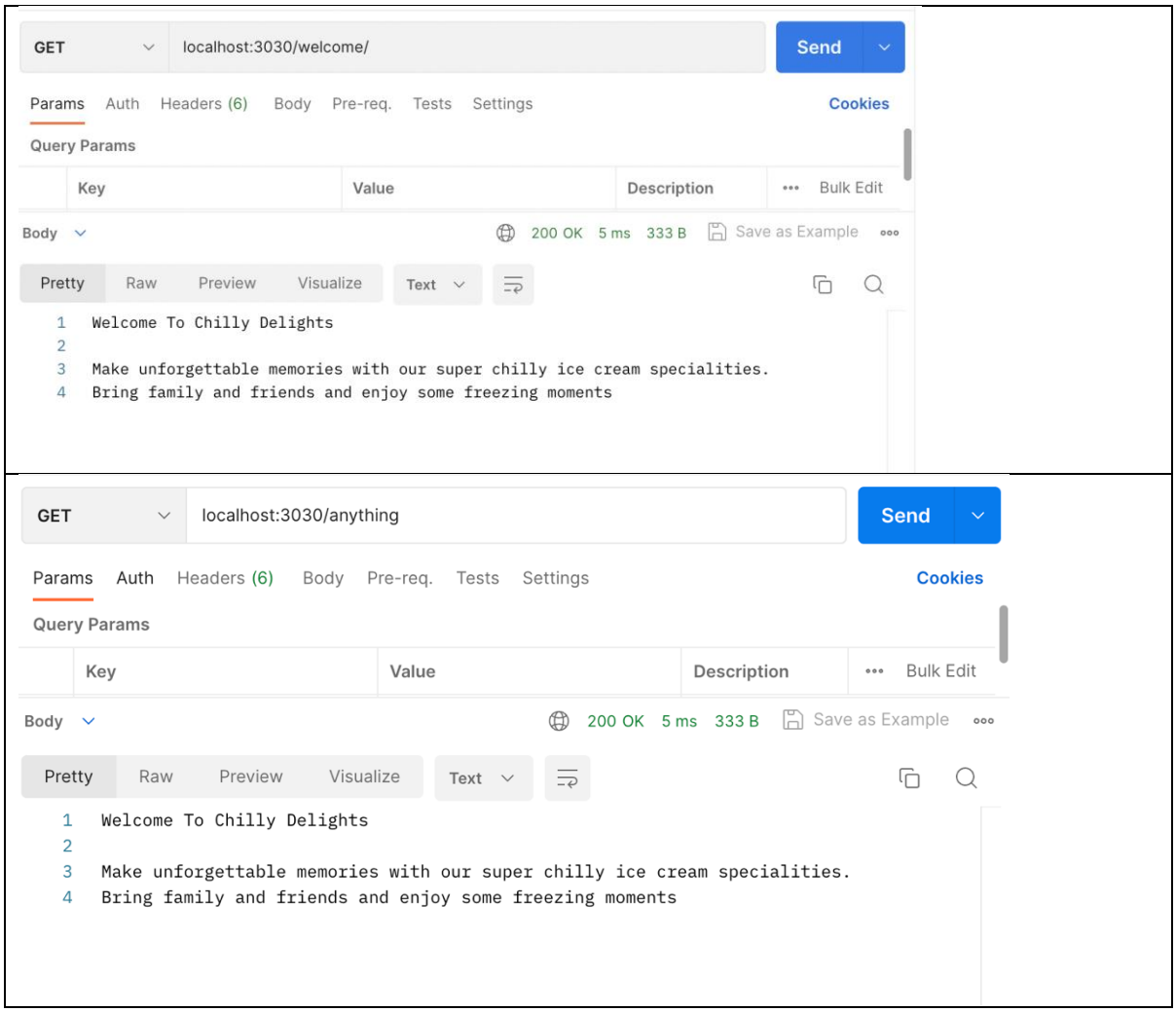
4. Delete: The manager at Chilly Delights might decide to stop serving one of the ice cream specialties on the menu. Accordingly, your server needs to allow deletions from the menu. For deletions, the ice cream specialty to be deleted is identified by its code. The server will retrieve the menu and delete the specified item and then store the menu back to the file.

##Test Cases:

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 route: Note that with GET regardless of the path provided after the / the server defaults to the welcome message





- Menu route: note that for the menu route whether the URL has a '/' after menu or not the server response is the same
    - Create:

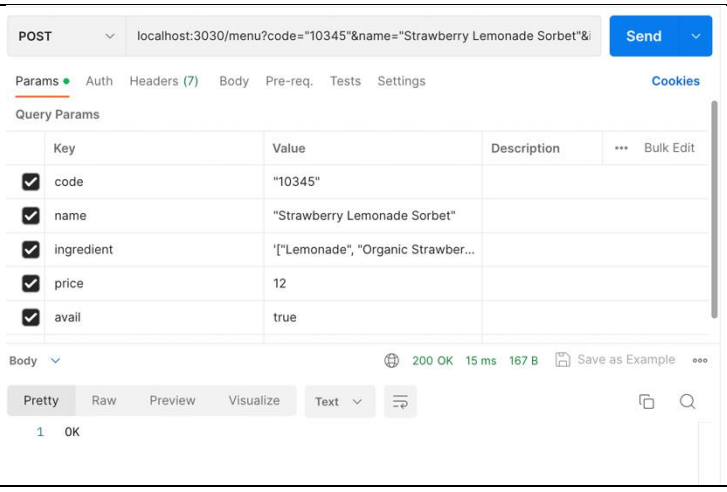
Query String key:value
Code: 10345

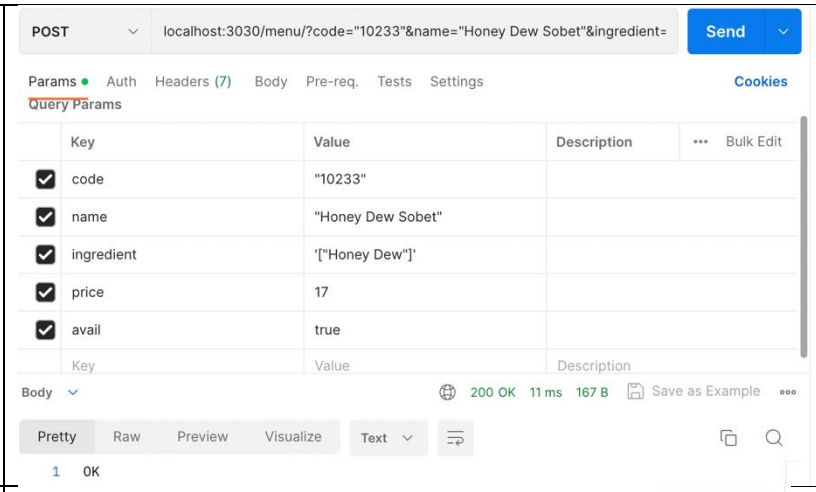
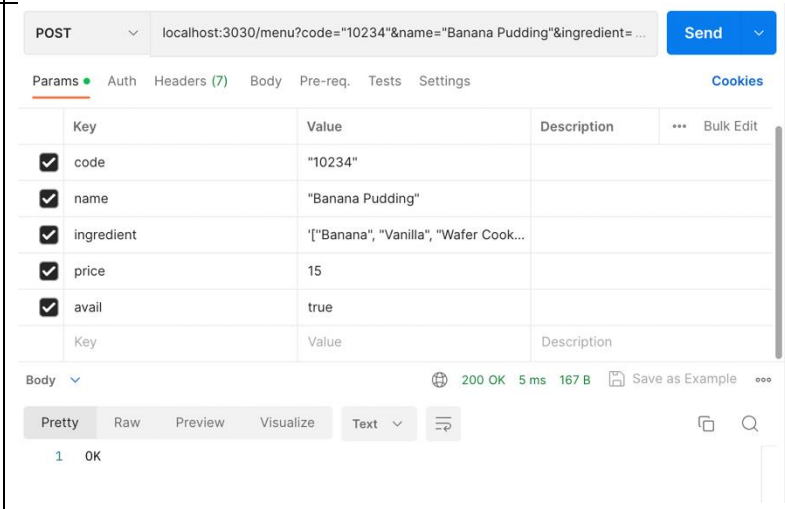
Name: Strawberry Lemonade Sorbet

Ingredient:
- Lemonade
- Organic Strawberries

Price: 12

Avail: true

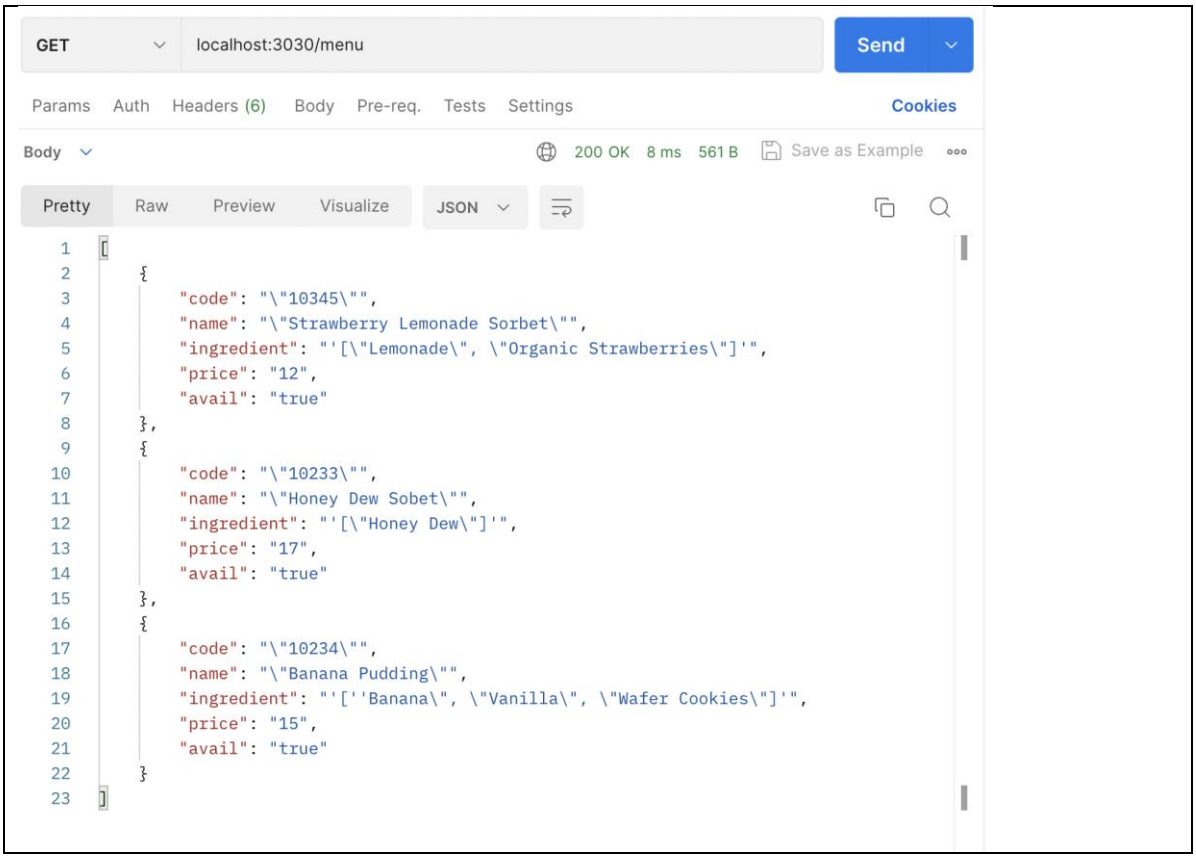


Query String key:value *Code*: 10233 *Name*: Honey Dew Sobet *Ingredient*: - Honey Dew *Price*: 17 *Avail*: true	
Query String key:value *Code*: 10234 *Name*: Banana Pudding *Ingredient*: - Banana - Vanilla - Wafer Cookies *Price*: 15 *Avail*: true	

For other values to add you can use the following for the query string values:

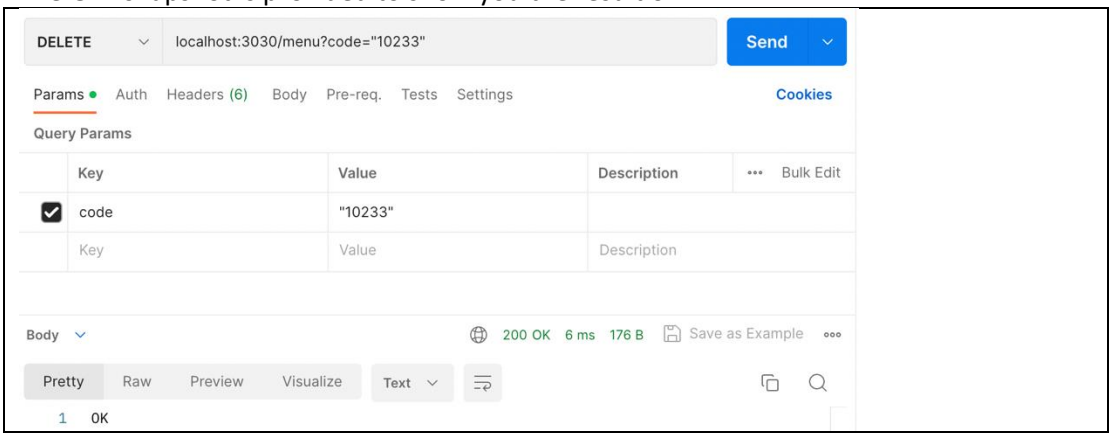
Query String key: value *Code*: 10232 *Name*: Coconut Almond Brownie *Ingredient*: - Coconut - Almond - Brownie *Price*: 17 *Avail*: true	*Query String key: value* *Code*: 10202 *Name*: Black Raspberry Chip *Ingredient*: - Black Raspberry - Chocolate *Price*: 13 *Avail*: true	*Query String key: value* *Code*: 10237 *Name*: Summer Peach Melba *Ingredient*: - Peach - Raspberry *Price*: 18 *Avail*: true

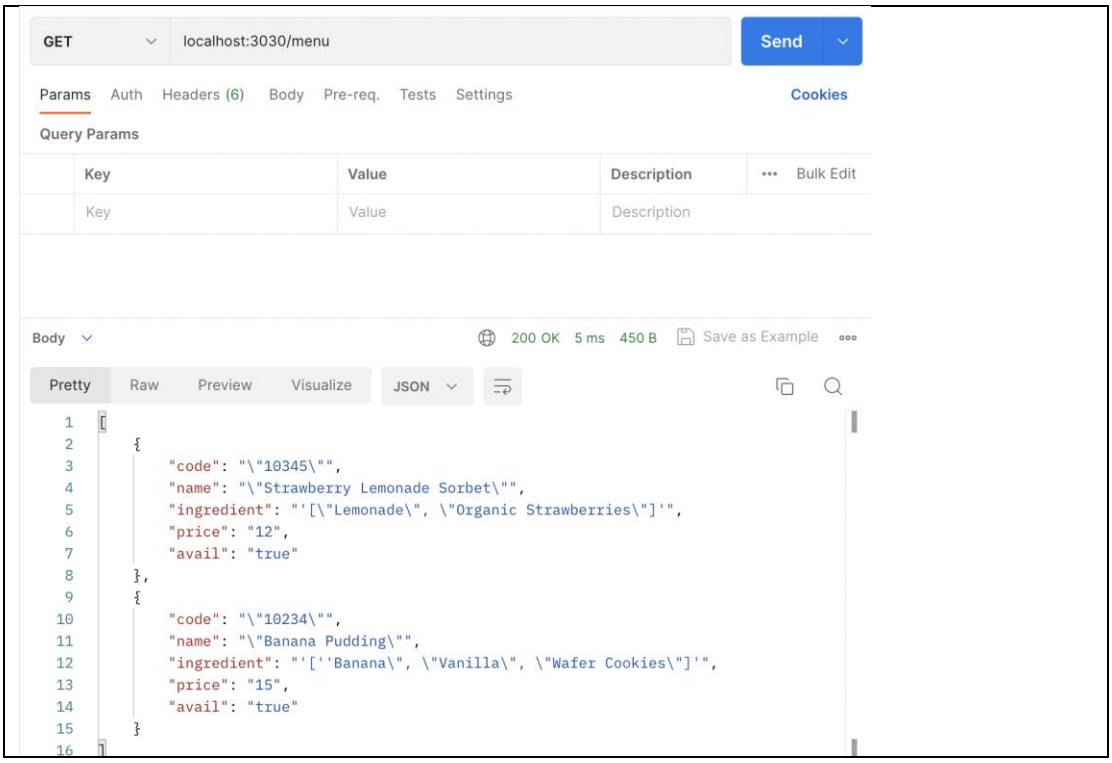
- Read



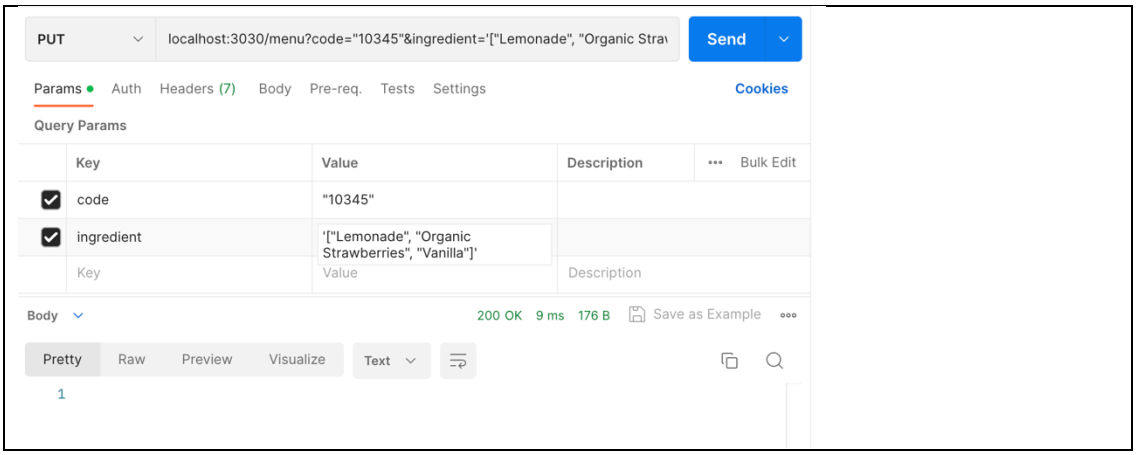
- Delete:

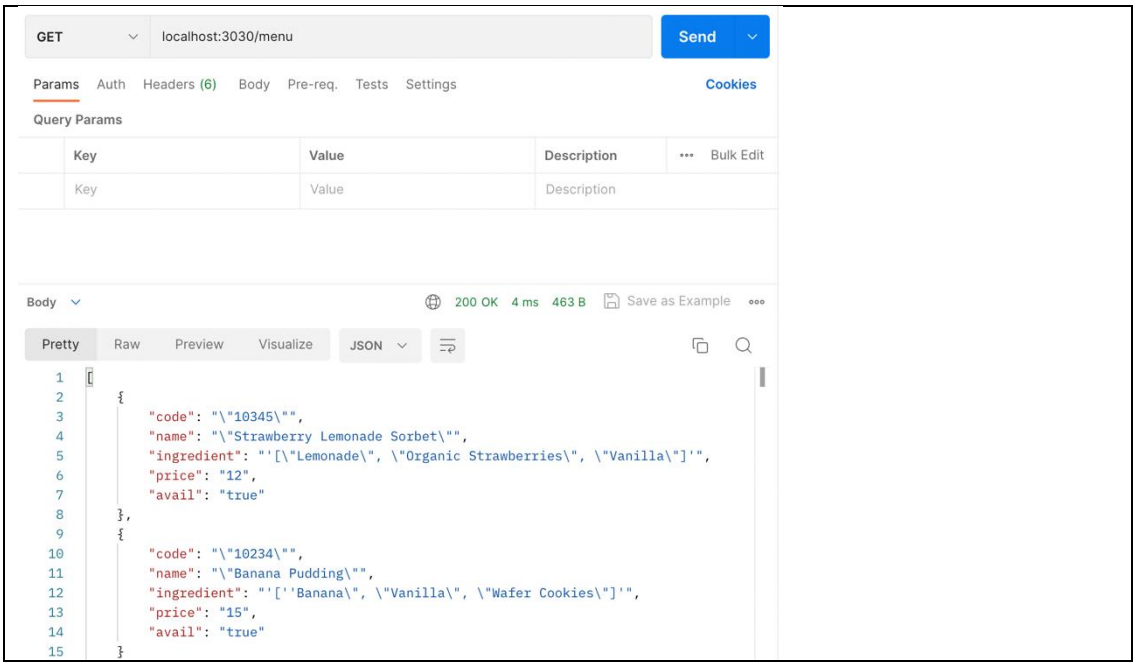
The GET snapshot is provided to show you the result of DELETE





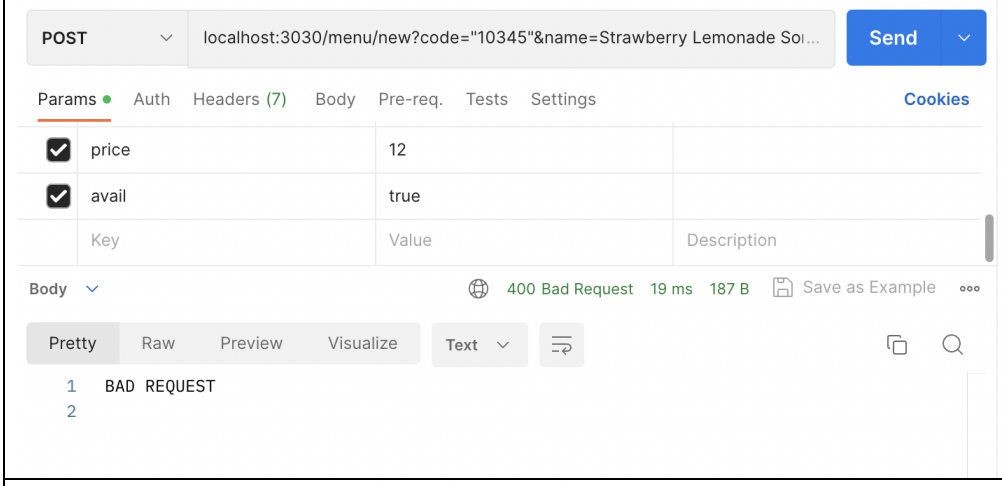
- Update:
  Vanilla was added to the ingredients
  The GET snapshot is provided to show you the result of Update



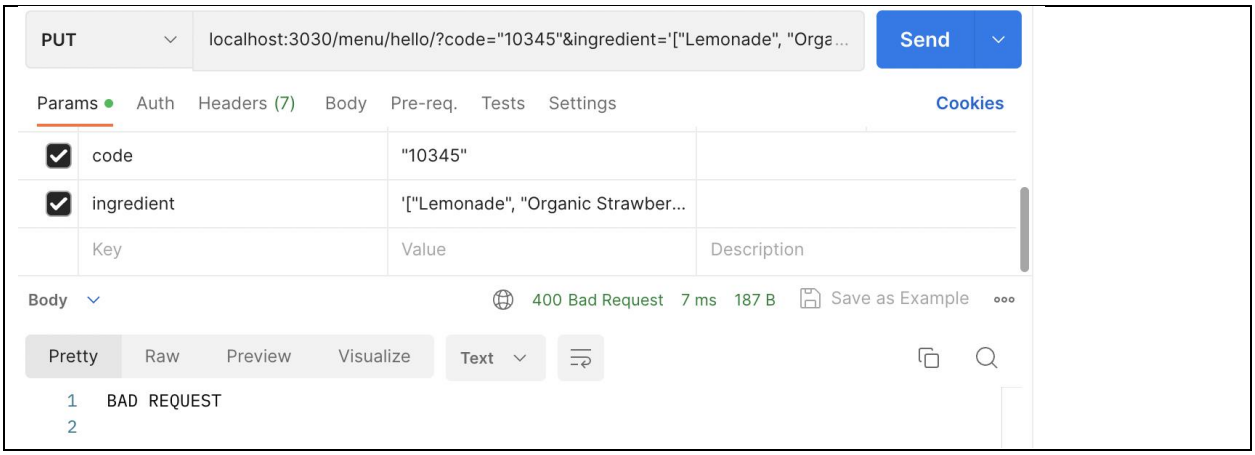


- Invalid routes:
  For any invalid path under the menu route the server will respond with the status code 400 and message ‘BAD REQUEST’

The URL references the path ‘/new’ under ‘/menu’, which is an invalid path



The URL references the path ‘/hello/’ under ‘/menu’, which is an invalid path



Setup and Implementation Hints

1. The first time the server runs, the menu. Json file needs to be created.
2. Provide two helper functions for retrieving and storing the menu. Use these functions for POST, DELETE and PUT operations.
3. For GET you only need to retrieve the menu. You do not need to save the menu back
4. Minimize the nesting of callbacks
5. Provide code to catch any invalid path under '/menu' and respond with '400 BAD REQUEST'
6. Develop the server code incrementally and always check that the response is received by POSTMAN.

Deliverables

`The code file with a .js extension for the server implementation.`

How You Will Be Graded

This assignment is Graded out of **120 points**

#	Task	Points
1.	Setting up the server	5 points
2.	Parsing the request URL to extract the path, query string, and request method	12 points
3.	Helper functions: • Loading the menu from menu.json and • storing the updated menu to menu.json	20 points • 12 points • 8 points
4.	POST	17 points
5.	GET	8 points
6.	DELETE	10 points
7.	PUT	18 points
8.	Code documentation and comments	15 points
9.	Code Quality and Style	15 points