Course Name:	**Electronics Application Using Python Programming**	**Semester:**	**V**
Date of Performance:	**10 / 09 / 2023**	**Batch No:**	**APPA 2**
Faculty Name:	**Prof. Deepa Jain**	**Roll No:**	**16014022096**
Faculty Sign & Date:		**Grade/Marks:**	**/25**

Experiment No: 6
Title: Study of Socket Programming

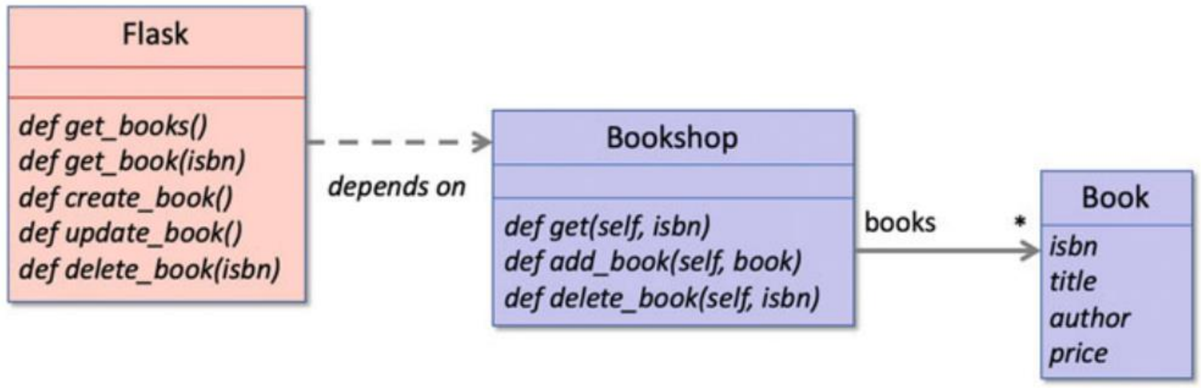
Aim and Objective of the Experiment:
To understand the implementation of RESTFull API

COs to be achieved:
CO3: Understand socket based and web service approaches to inter process communications.

Theory
Introduction:

Before we look at the implementation of the Bookshop RESTful API we will consider what elements we for the services services. One question that often causes some confusion is how web services relate to traditional design approaches such as object oriented design. The approach adopted here is that the Web Service API provides a way to implement an interface to appropriate functions, objects and methods used to implement the application/ domain model. This means that we will still have a set of classes that will represent the Bookshop and the Books held within the bookshop. In turn the functions implementing the web services will access the bookshop to retrieve, modify, update and delete the books held by the bookshop.

The overall design is shown below:



This shows that a Book object will have an isbn, a title, an author and a price attribute. In turn the Bookshop object will have a books attribute that will hold zero or more Books. The books attribute will actually hold a

List as the list of books needs to change dynamically as and when new books are added or old books deleted. The Bookshop will also define three methods that will • allow a book to be obtained via its isbn,
• allow a book to be added to the list of books and
•enable a book to be deleted (based on its isbn). Routing information will be provided for a set of functions that will invoke appropriate methods on the Bookshop object. The functions to be decorated with @app.route, and the mappings to be used, are listed below:
• get_books() which maps to the /book/list URL using the HTTP Get method request.
• get_book(isbn) which maps to the /book/ URL where isbn is a URL parameter that will be passed into the function. This will also use the HTTP GET request.
• create_book() which maps to the /book URL using the HTTP Post request.
• update_book() which maps to the /book URL but using the HTTP Put request.
• delete_book() which maps to the /book/ URL but using the HTTP Delete request.

Tools required:

Any python editor tool

Code:

- Write a Python program to perform CRUD Operation using Flask Python Framework

CODE:

```python
from flask import Flask, jsonify, request

app = Flask(__name__)


book_db = [
    {'name': 'Bungo Stray Dogs', 'price': 40},
    {'name': 'One piece', 'price': 30},
    {'name': 'Jujustu Kaisen', 'price': 70},
    {'name': 'Naruto', 'price': 60}
]


@app.route('/')
def home():
    return "WELCOME TO vwarrier BOOK-STORE"


@app.route('/books', methods=['GET'])
def get_books():
    return jsonify({'books  on vwarrier book store': book_db})
```

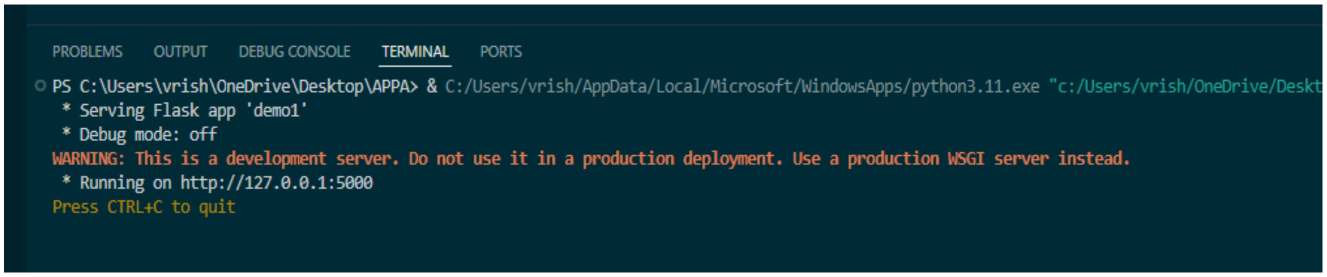
```
@app.route('/book/<string:name>', methods=['GET'])
def get_book(name):
    for book in book_db:
        if book['name'].lower() == name.lower():
            return jsonify(book)
    return jsonify({'message': 'Book not found on vwarrier book store'})


@app.route('/book', methods=['POST'])
def create_book():
    new_book = request.get_json()
    book_db.append(new_book)
    return jsonify({'message': 'Book created on vwarrier book store', 'book':
new_book})


@app.route('/book/<string:name>', methods=['PUT'])
def update_book(name):
    update_data = request.get_json()
    for book in book_db:
        if book['name'].lower() == name.lower():
            book.update(update_data)
            return jsonify({'message': 'Book updated on vwarrier book store',
'book': book})
    return jsonify({'message': 'Book not found on vwarrier book store'})


@app.route('/book/<string:name>', methods=['DELETE'])
def delete_book(name):
    for book in book_db:
        if book['name'].lower() == name.lower():
            book_db.remove(book)
            return jsonify({'message': 'Book deleted on vwarrier book store'})
    return jsonify({'message': 'Book not found on vwarrier book store'})


app.run(port=5050)
```

PRACTICE CODE DURING LAB

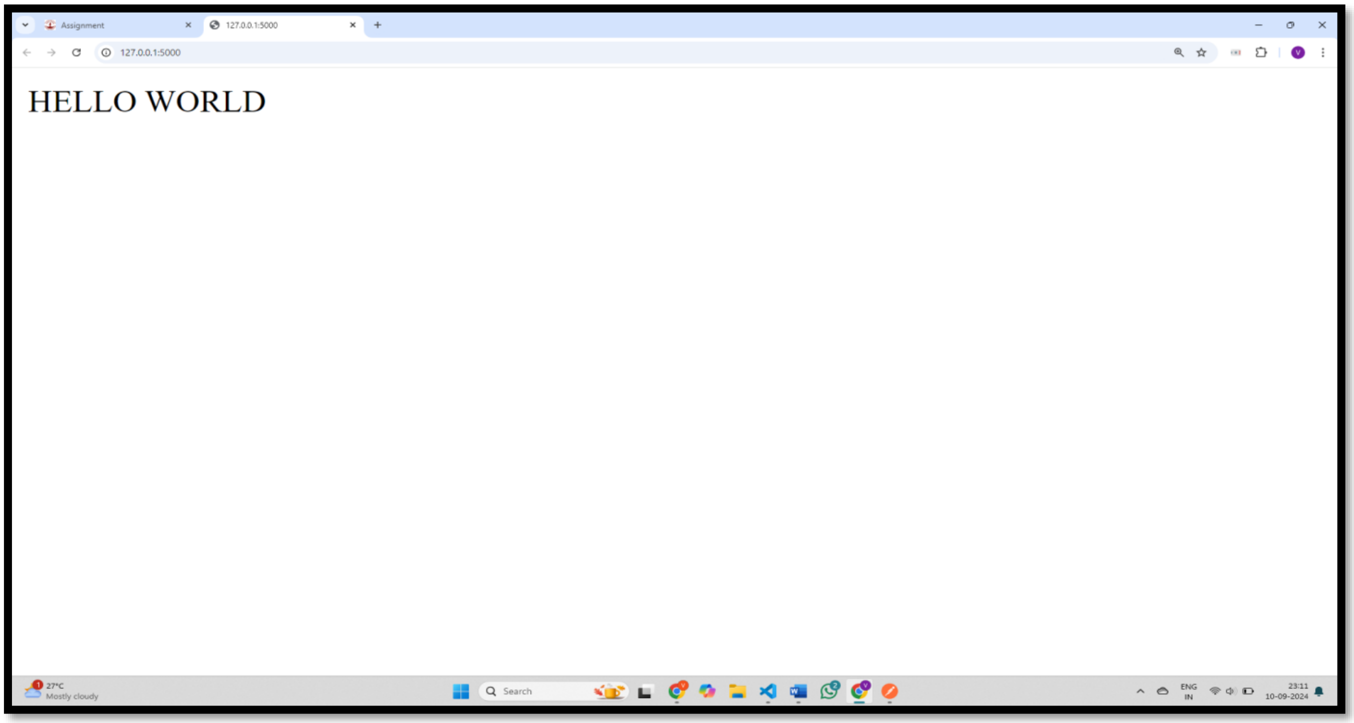
CODE:

```
from flask import Flask

app = Flask(__name__)

@app.route("/")
def welcome():
    return "HELLO WORLD"

@app.route("/home")
def home():
    return "HELLO HOME PAGE"

app.run(port=5000)
```

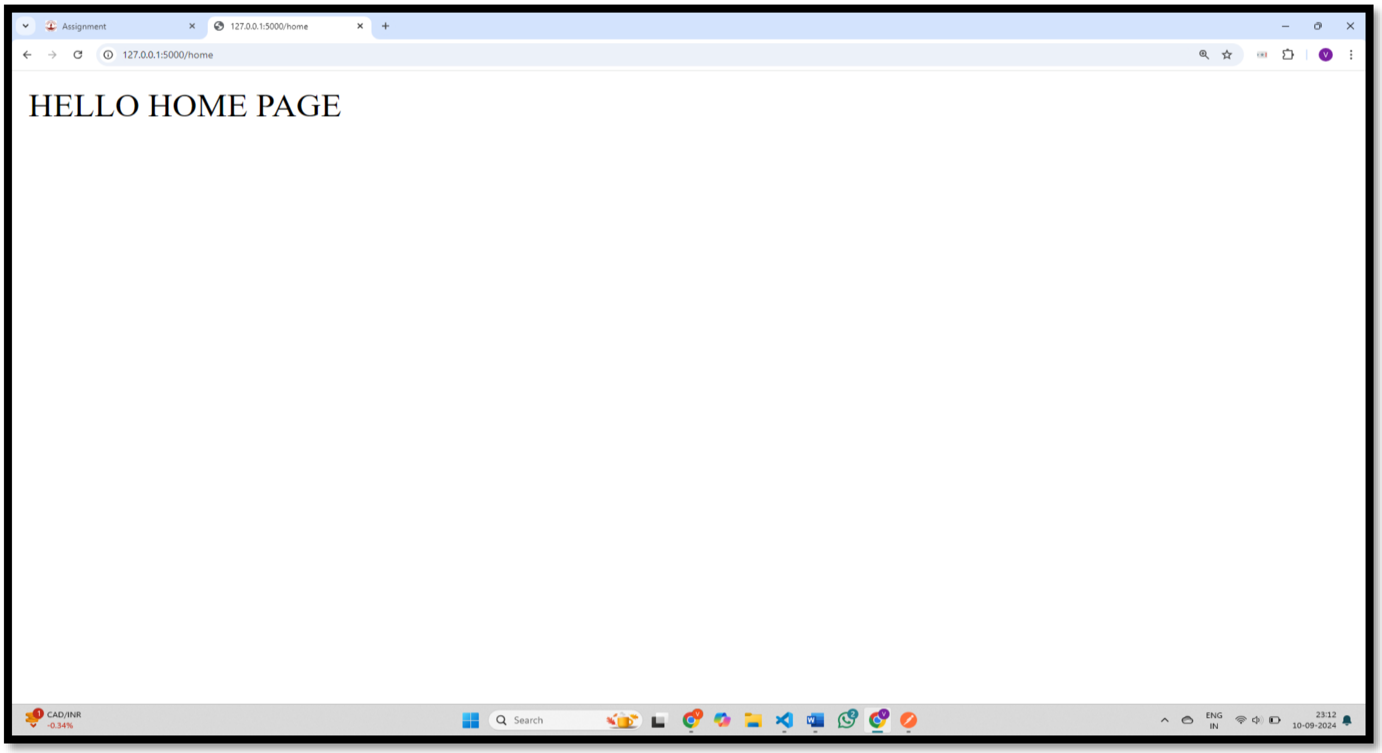
OUTPUT:



```
PS C:\Users\vrish\OneDrive\Desktop\APPA> & C:/Users/vrish/AppData/Local/Microsoft/WindowsApps/python3.11.exe "c:/Users/vrish/OneDrive/Deskt
 * Serving Flask app 'demo1'
 * Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on http://127.0.0.1:5000
Press CTRL+C to quit
```



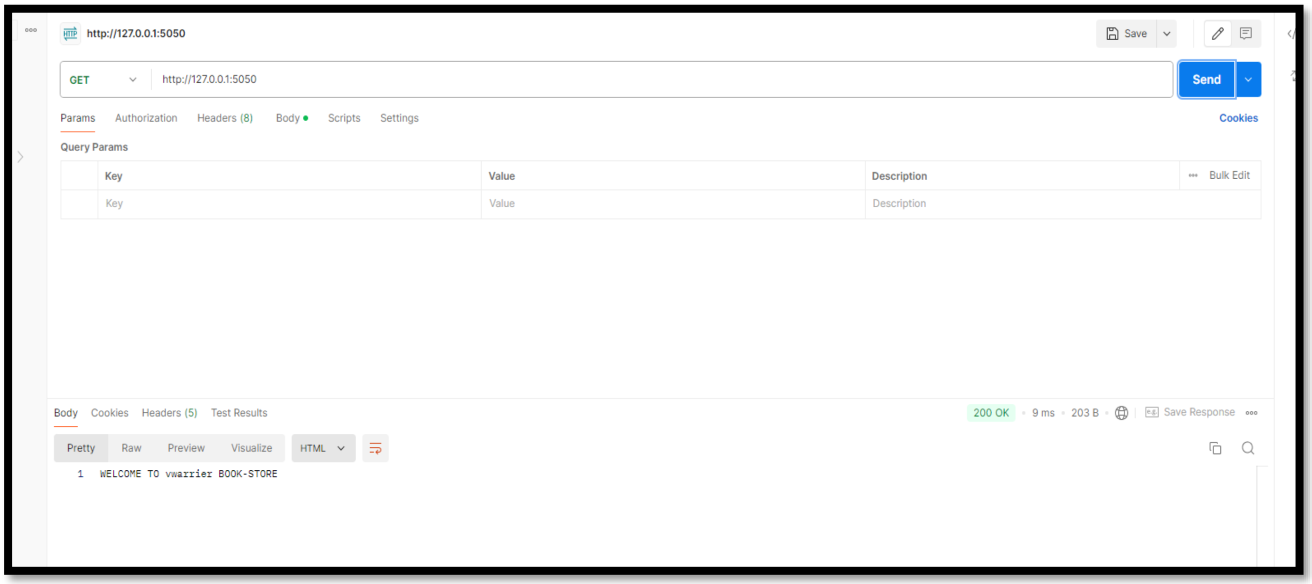
HELLO WORLD



HELLO HOME PAGE

Output:

Home page

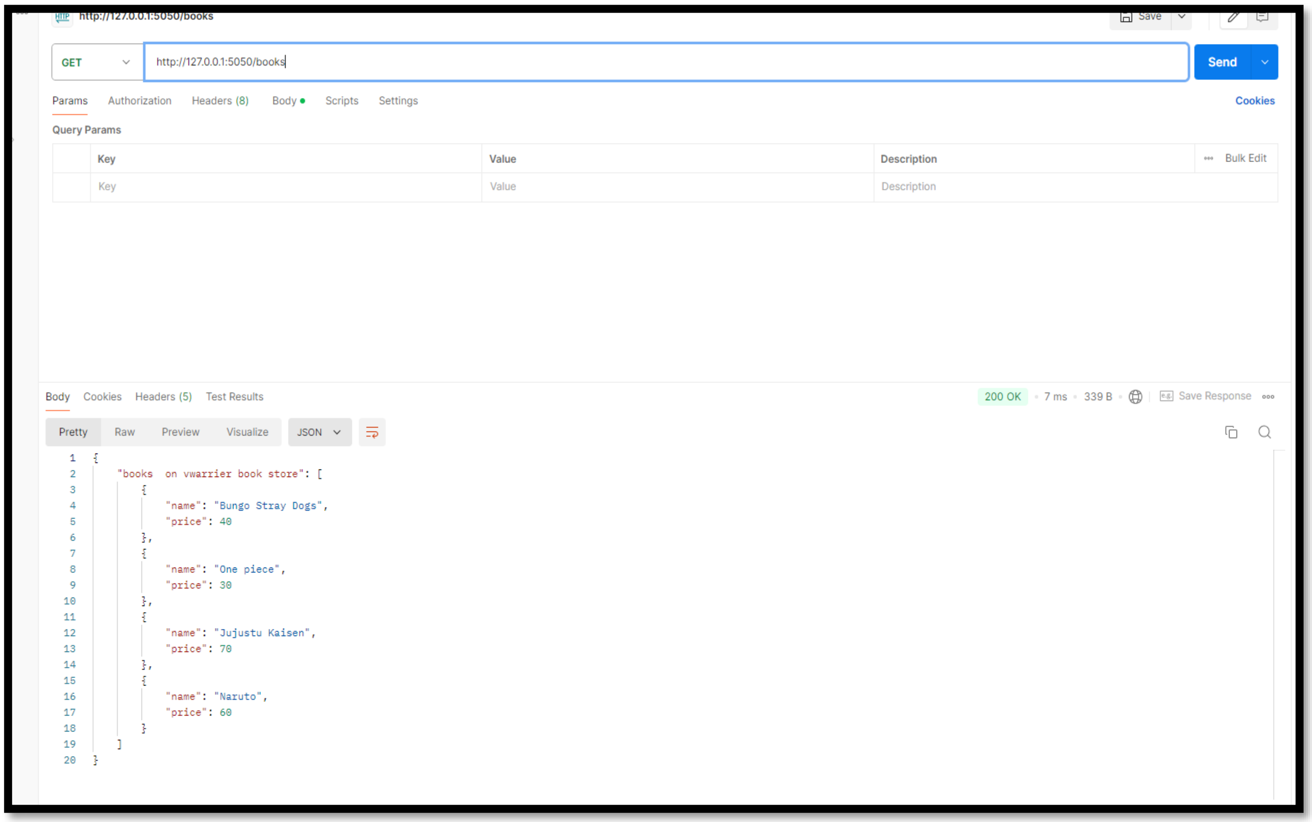


GET http://127.0.0.1:5050

200 OK · 9 ms · 203 B

```
WELCOME TO vwarrier BOOK-STORE
```

List of books

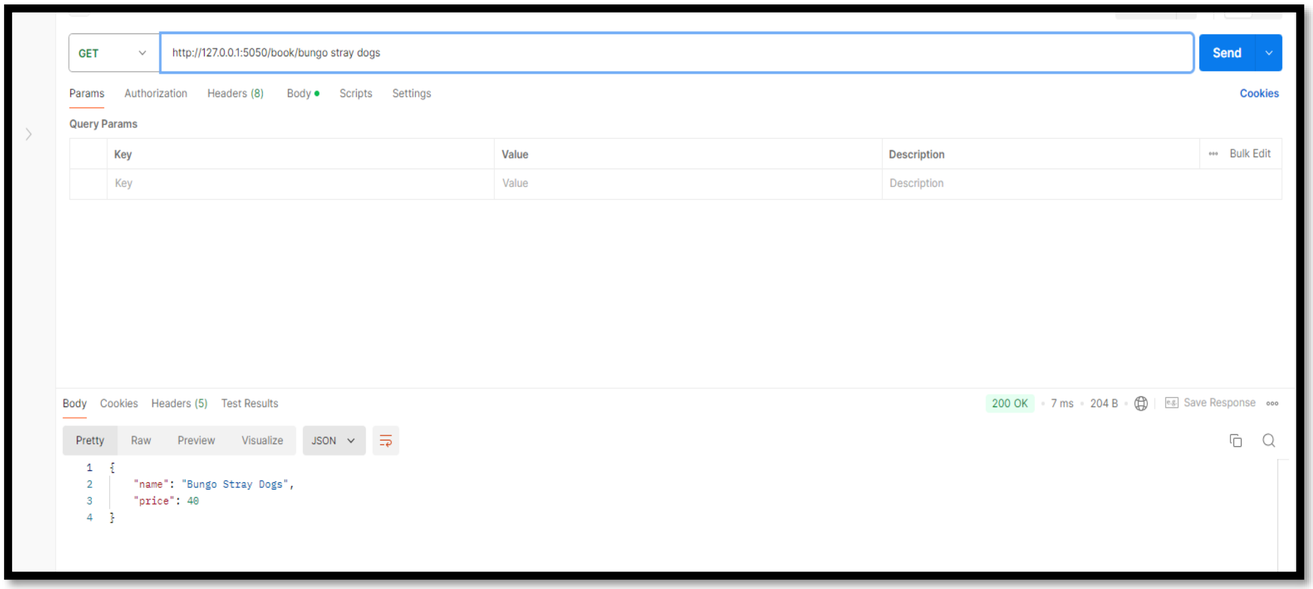


GET http://127.0.0.1:5050/books

200 OK · 7 ms · 339 B

```
{
    "books  on vwarrier book store": [
        {
            "name": "Bungo Stray Dogs",
            "price": 40
        },
        {
            "name": "One piece",
            "price": 30
        },
        {
            "name": "Jujustu Kaisen",
            "price": 70
        },
        {
            "name": "Naruto",
            "price": 60
        }
    ]
}
```

Bungo stray dogs book details

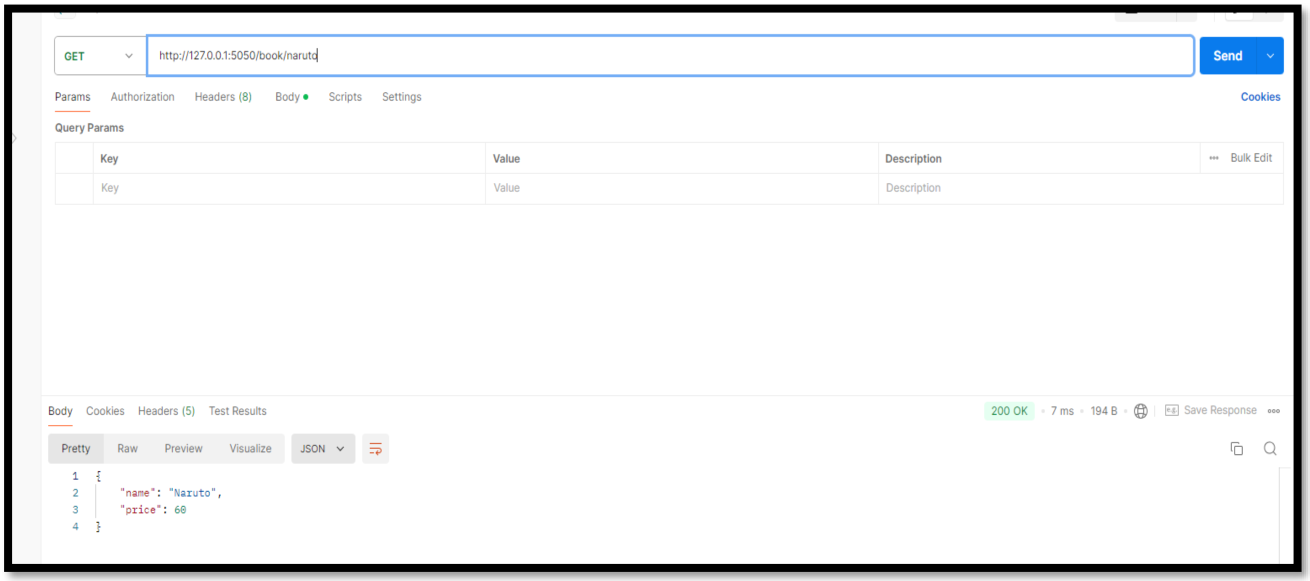


GET http://127.0.0.1:5050/book/bungo stray dogs

200 OK · 7 ms · 204 B

```
{
    "name": "Bungo Stray Dogs",
    "price": 40
}
```

Naruto book details

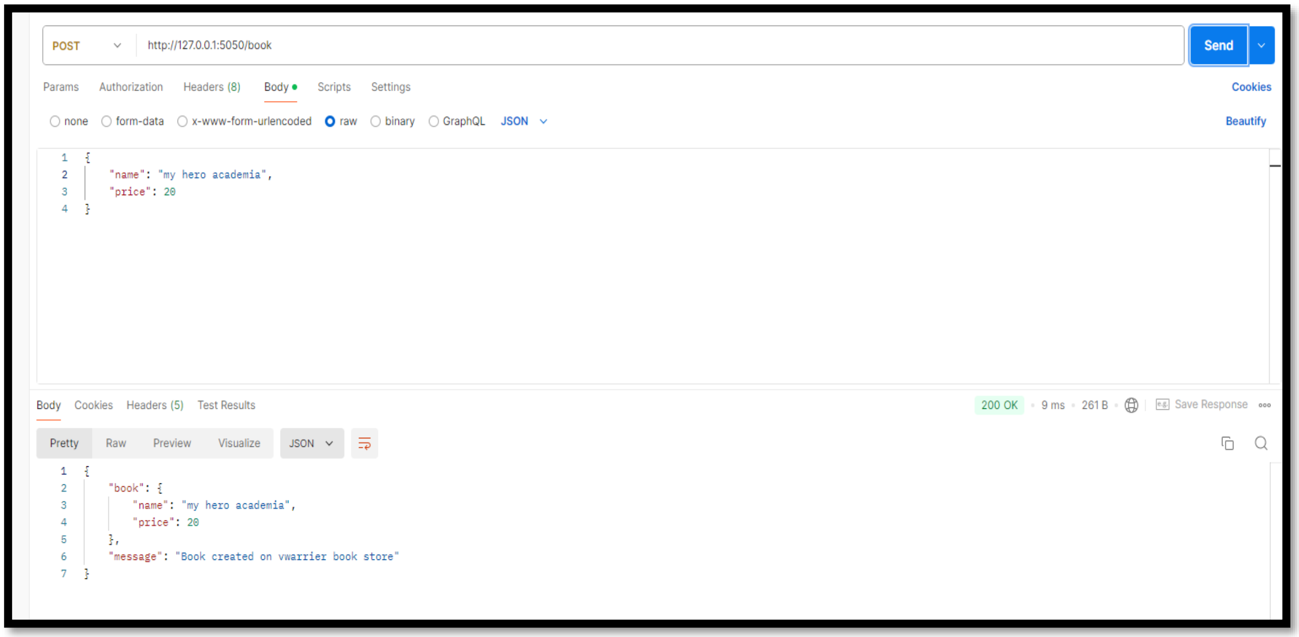


GET http://127.0.0.1:5050/book/naruto

200 OK · 7 ms · 194 B

```
{
    "name": "Naruto",
    "price": 60
}
```

Creating new book


POST http://127.0.0.1:5050/book

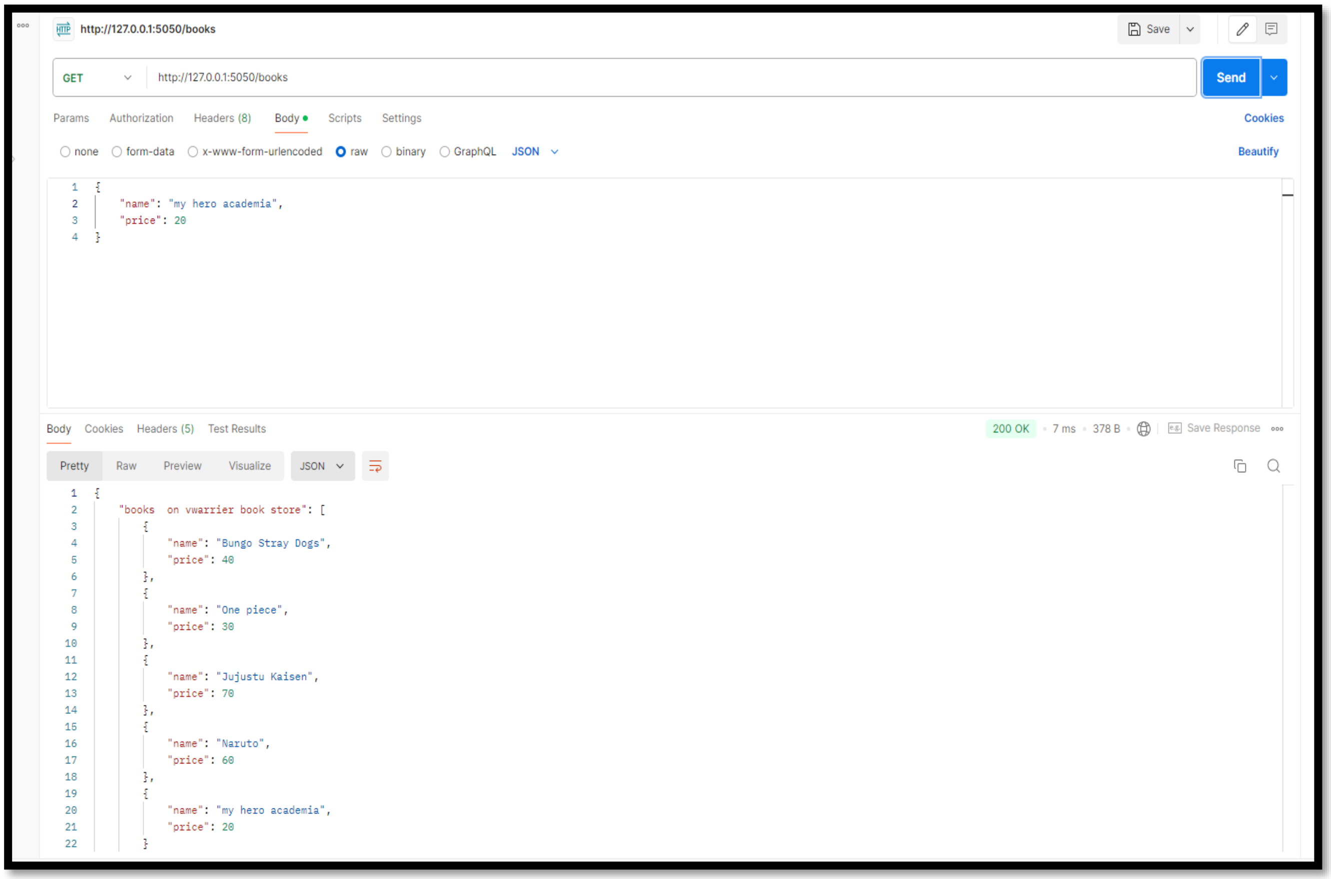
Body: raw JSON

```
{
    "name": "my hero academia",
    "price": 20
}
```

200 OK 9 ms 261 B

```
{
    "book": {
        "name": "my hero academia",
        "price": 20
    },
    "message": "Book created on vwarrier book store"
}
```

New list of books


GET http://127.0.0.1:5050/books

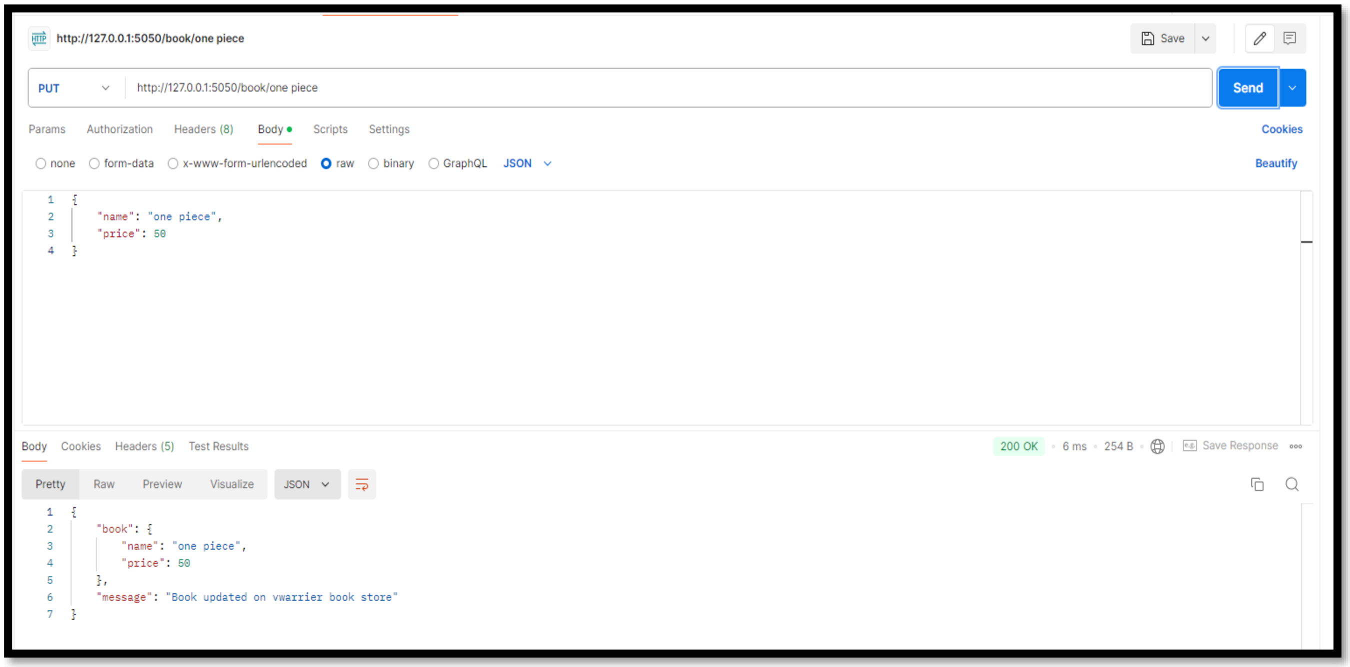
Body: raw JSON

```
{
    "name": "my hero academia",
    "price": 20
}
```

200 OK 7 ms 378 B

```
{
    "books  on vwarrier book store": [
        {
            "name": "Bungo Stray Dogs",
            "price": 40
        },
        {
            "name": "One piece",
            "price": 30
        },
        {
            "name": "Jujustu Kaisen",
            "price": 70
        },
        {
            "name": "Naruto",
            "price": 60
        },
        {
            "name": "my hero academia",
            "price": 20
        }
```

Updating a book


http://127.0.0.1:5050/book/one piece

PUT http://127.0.0.1:5050/book/one piece

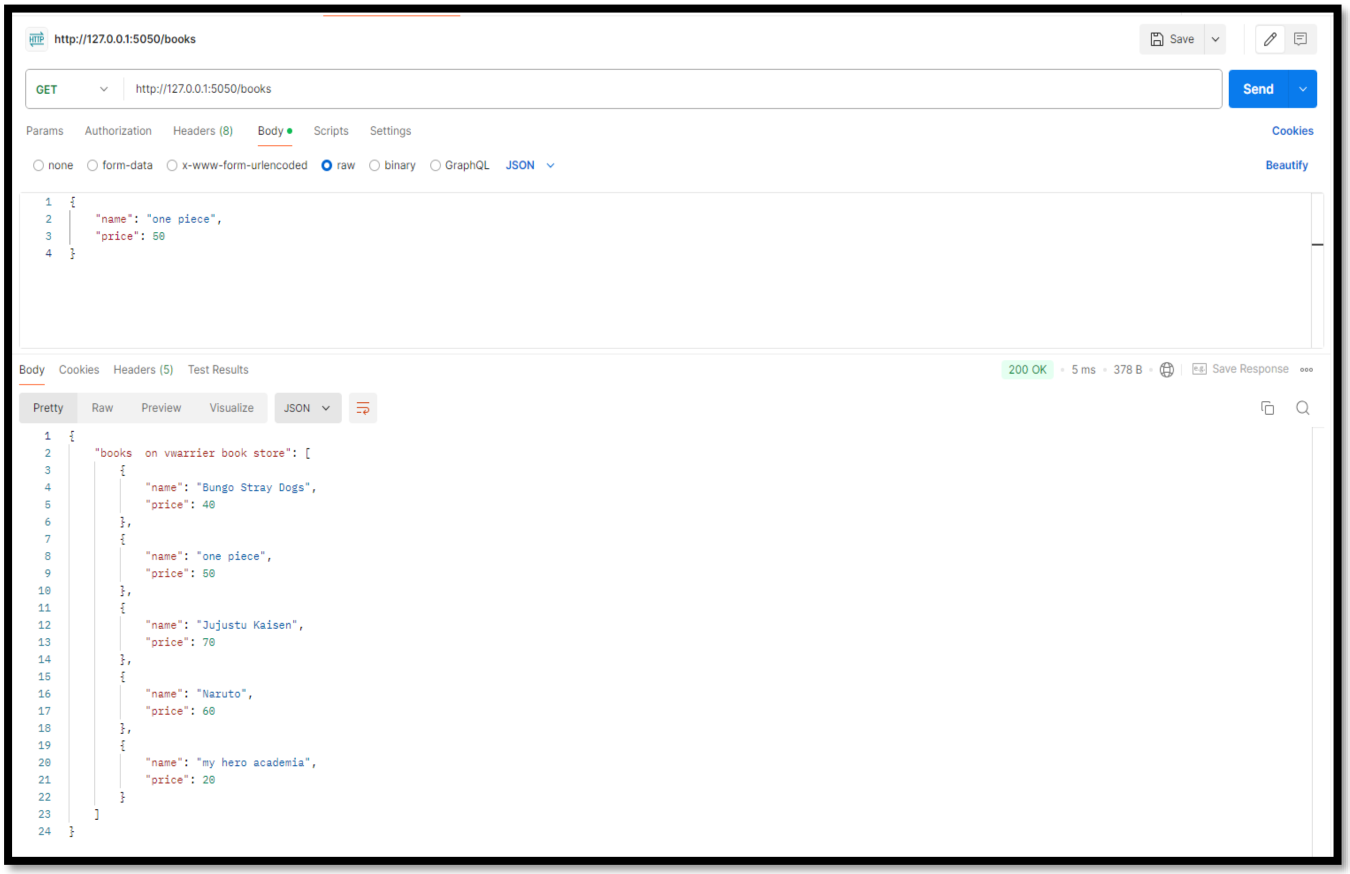
Body: raw JSON

```
{
    "name": "one piece",
    "price": 50
}
```

200 OK · 6 ms · 254 B

```
{
    "book": {
        "name": "one piece",
        "price": 50
    },
    "message": "Book updated on vwarrier book store"
}
```

New list after updating


http://127.0.0.1:5050/books

GET http://127.0.0.1:5050/books

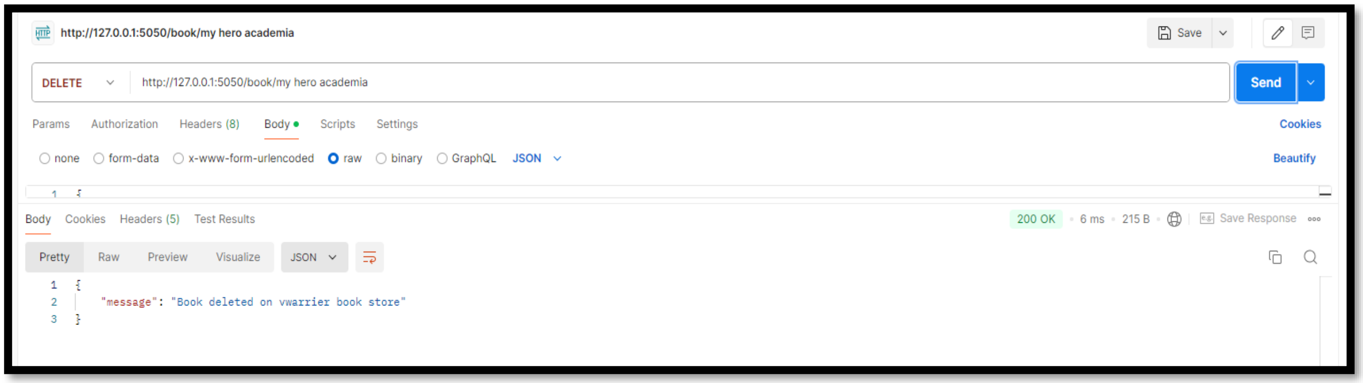
Body: raw JSON

```
{
    "name": "one piece",
    "price": 50
}
```

200 OK · 5 ms · 378 B

```
{
    "books  on vwarrier book store": [
        {
            "name": "Bungo Stray Dogs",
            "price": 40
        },
        {
            "name": "one piece",
            "price": 50
        },
        {
            "name": "Jujustu Kaisen",
            "price": 70
        },
        {
            "name": "Naruto",
            "price": 60
        },
        {
            "name": "my hero academia",
            "price": 20
        }
    ]
}
```

Deleting a book



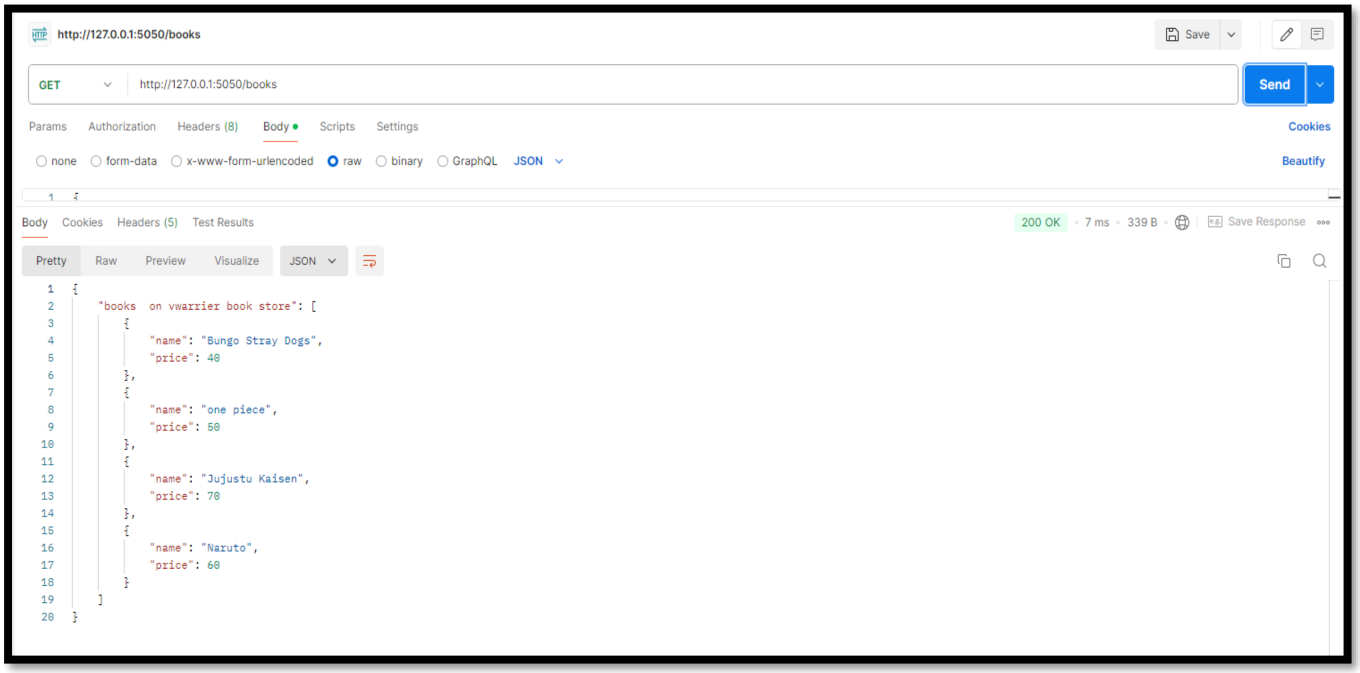
DELETE http://127.0.0.1:5050/book/my hero academia

Body: raw JSON

200 OK

```
{
    "message": "Book deleted on vwarrier book store"
}
```

New list after deleting a book



GET http://127.0.0.1:5050/books

Body: raw JSON

200 OK

```
{
    "books  on vwarrier book store": [
        {
            "name": "Bungo Stray Dogs",
            "price": 40
        },
        {
            "name": "one piece",
            "price": 50
        },
        {
            "name": "Jujustu Kaisen",
            "price": 70
        },
        {
            "name": "Naruto",
            "price": 60
        }
    ]
}
```

Post Lab Subjective/Objective type Questions:

1. What is difference between API and RESTFull API

ANS:

	REST API	**RESTful API**
Definition	Develops APIs to enable client-server interaction.	Follows REST architecture for web applications, promoting system interoperability.
Working	Uses web services based on request and response.	Completely based on REST architecture principles.
Protocol	Strong protocol with built-in security layers.	Less secure and relies on a transport protocol.
Format of Data	Format is primarily based on HTTP.	Supports multiple formats like HTTP, JSON, and text.
Cache	Represents cacheable and non-cacheable data.	Allows access to cacheable data anytime, anywhere.

2. List and explain various methods of HTTP module

- GET:

Its purpose is to retrieve data from the server.
It is used to request resources without modifying anything.
Example: Fetching a list of books.

- POST:

Its purpose is to send data to the server to create a new resource.
It is used to submit forms or upload data.
Example: Creating a new book entry.

- PUT:

Its purpose is to update or replace an existing resource.
It is used when modifying an entire resource.
Example: Updating all details of a specific book.

- DELETE:

Its purpose is to remove a resource from the server.
It is used to delete resources.
Example: Deleting a book by its ISBN.

Conclusion:
We have successfully understood how to implement a RESTful API using Flask for CRUD operations, how to use HTTP methods such as GET, POST, PUT, and DELETE to manage resources, and how socket-based communication supports web services for inter-process communication.

Signature of faculty in-charge with Date: