|  |  |  |  |
| --- | --- | --- | --- |
| **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:  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:** |

|  |
| --- |
| **Output:** |
| Home page    List of books    Bungo stray dogs book details    Naruto book details    Creating new book    New list of books    Updating a book    New list after updating    Deleting a book    New list after deleting a book |

|  |
| --- |
| **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. |  1. 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:** |