



Usman Institute of Technology

Department of Computer Science

Course Code: SE308

Course Title: Software Design and Architecture

Fall 2022

Lab 10

OBJECTIVE: Introduction to API and its implementation using Python

- To component communication.
- To implement API using flask framework in python programming language.

Student Information

Student Name	
Student ID	
Date	

Assessment

Marks Obtained	
Remarks	
Signature	

1 THEORY

1.1 What is API

An API (Application Programming Interface) is a set of programming code that enables data transmission between one software product and another. It also contains the terms of this data exchange.

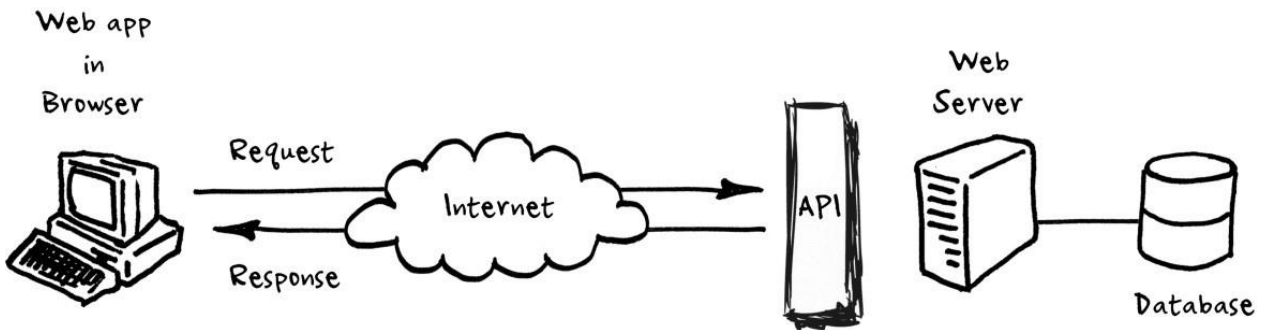


Figure 1. API, how its work

Implementation

Step 1: Open VSCode / PyCharm or any other IDE and create a new project, make sure that you have added the desire packages (Flask) on your newly created project, by using python package installer (pip command)

Step 2: Create a new file named app.py and write flask skeleton code shown as below.

```
from flask import Flask, jsonify, render_template, request
app = Flask(__name__)

// all code goes here including route
if __name__ == '__main__':
    app.run(debug=True)
```

Step 3: Now add new route for index(home) page contain the following line

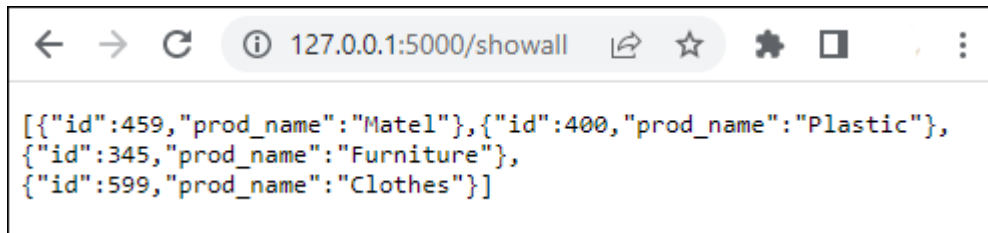
```
@app.route('/')
def index():
    return jsonify({'message': 'Hello World'})
```

Exercise: Below is the code snippet to create an API contain student data in JSON format (python dictionary) using flask web development framework

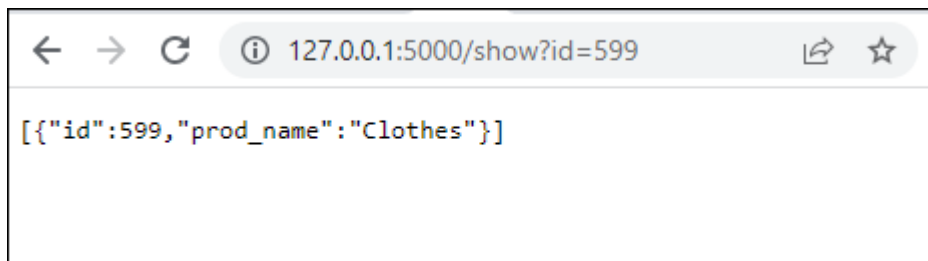
Note: Make sure that your newly created python project in your IDE must include Flask library.

```
1  from flask import Flask, jsonify, request, render_template
2  import json
3
4  app = Flask(__name__)
5  #product = json.load(open('data.json'))
6  product = [{"id":459, "prod_name":"Matel"}, {"id":400, "prod_name":"Plastic"},
7             {"id":345, "prod_name":"Furniture"}, {"id":599, "prod_name":"Clothes"}]
8  @app.route("/")
9  def hello():
10     return "Hello, Flask!"
11
12  @app.route("/showall")
13  def showall():
14     return jsonify(product)
15
16  @app.route("/show", methods=['GET'])
17  def show():
18     d=[]
19     id = int(request.args['id'])
20     for i in product :
21         if int(i['id'])==id:
22             d.append(i)
23             break
24     return jsonify(d)
25     #return render_template('show.html', title=i['title'], category=i['category'], i
26
27  if __name__ == "__main__":
28     app.run(debug=True)
```

Outputs



A screenshot of a web browser window. The address bar shows the URL `127.0.0.1:5000/showall`. The browser content displays a JSON array of product objects: `[{"id":459,"prod_name":"Matel"}, {"id":400,"prod_name":"Plastic"}, {"id":345,"prod_name":"Furniture"}, {"id":599,"prod_name":"Clothes"}]`.



A screenshot of a web browser window. The address bar shows the URL `127.0.0.1:5000/show?id=599`. The browser content displays a JSON array containing only the product with id 599: `[{"id":599,"prod_name":"Clothes"}]`.

2 What is REST

REST stands for REpresentational State Transfer. REST is a web standards-based architecture and uses HTTP Protocol for data communication. It revolves around resources where every component is a resource and a resource is accessed by a common interface using HTTP standard methods. REST was first introduced by Roy Fielding in year 2000. In REST architecture, a REST Server simply provides access to resources and the REST client accesses and presents the resources. Here each resource is identified by URIs/ Global IDs. REST uses various representations to represent a resource like Text, JSON and XML. JSON is now the most popular format being used in Web Services.

2.1 HTTP Methods

The following HTTP methods are most commonly used in a REST based architecture.

- GET - Provides a read only access to a resource.
- PUT - Used to create a new resource.
- DELETE - Used to remove a resource.
- POST - Used to update an existing resource or create a new resource.
- OPTIONS - Used to get the supported operations on a resource

2.2 RESTful Web Services

A web service is a collection of open protocols and standards used for exchanging data between applications or systems. Software applications written in various programming languages and running on various platforms can use web services to exchange data over computer networks like the Internet in a manner similar to inter-process communication on a single computer. This interoperability (e.g., between Java and Python, or Windows and Linux applications) is due to the use of open standards. Web services based on REST Architecture are known as RESTful Web Services.

These web services use HTTP methods to implement the concept of REST architecture. A RESTful web service usually defines a URI (Uniform Resource Identifier), which is a service that provides resource representation such as JSON and a set of HTTP Methods

2.3 HTTP Response Codes

- 200 (OK) - the request has been processed successfully on server
- 201 (Created) - one or more new resources have been successfully created on server.
- 202 (Accepted) - the request has been accepted for processing, but the processing has not been completed
- 204 (No Content) - indicates that the server has successfully fulfilled the request and that there is no content to send in the response.
- 301 (Moved Permanently) - indicates that the resource requested has been permanently moved to the URL given

2.4 Introduction Flask

Flask is a lightweight WSGI (web server gateway interface) web application framework. It is designed to make getting started quick and easy, with the ability to scale up to complex applications. It began as a simple wrapper around Werkzeug and Jinja and has become one of the most popular Python web application frameworks.

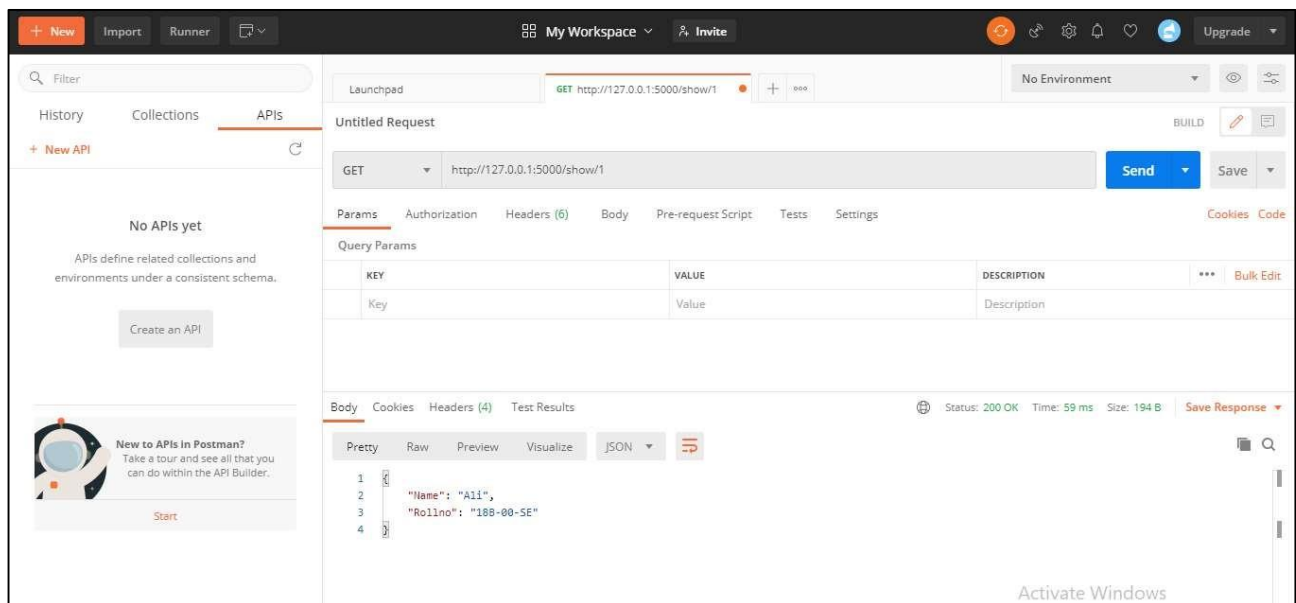


We use Flask as web development platform for our RESTful API implementation, for this we need to add Flask-RESTful library in-order to quickly building REST APIs

2.5 Introduction POSTMAN tool

Postman is a popular API client that makes it easy for developers to create, share, test and document APIs. This is done by allowing users to create and save simple and complex HTTP/s requests, as well as read their responses. The result - more efficient and less tedious work.

Postman is a great tool when trying to dissect RESTful APIs made by others or test ones you have made yourself. It offers a sleek user interface with which to make HTML requests, without the hassle of writing a bunch of code just to test an API's functionality.



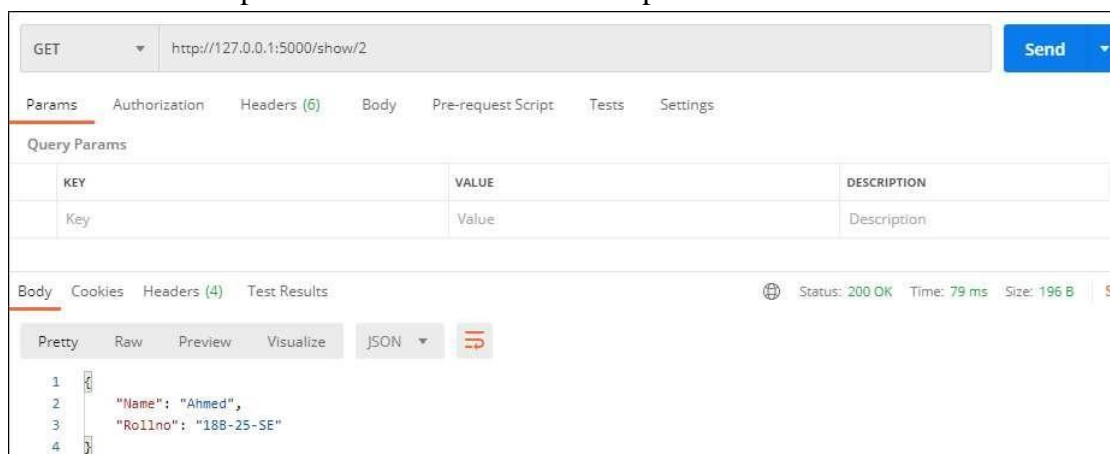
2.4 Creating RESTful Web Service

Below is the code snippet to create a web service contain student data in JSON format (python dictionary) with all the HTTP Methods mention above in 2.1

Note: Make sure that your newly created python project in PyCharm include Flask & Flask-RESTful library.

```
main.py x
1  from flask import Flask
2  from flask_restful import Api, Resource, reqparse, abort
3
4  app = Flask(__name__)
5  api = Api(app)
6
7  std_put = reqparse.RequestParser()
8  std_put.add_argument("Name", type=str)
9  std_put.add_argument("Rollno", type=str)
10
11  students = {1: {"Name": "Ali", "Rollno": "18B-00-SE"},
12             2: {"Name": "Ahmed", "Rollno": "18B-25-SE"},
13             } #dictionary
14
15  def notFound(std_id):
16      if std_id not in students:
17          abort(404, message="Student id not found!")
18
19  class std(Resource):
20      def get(self, std_id):
21          notFound(std_id)
22          return students[std_id]
23
24      def put(self, std_id):
25          newStd = std_put.parse_args()
26          students[std_id] = newStd
27          return students[std_id]
28
29      def delete(self, std_id):
30          del students[std_id]
31          return '', 204
32
33  api.add_resource(std, "/show/<int:std_id>") # represent route
34
35  if __name__ == '__main__':
36      app.run(debug=True)
```

Below are the HTTP responses that has been shows via postman tool



Student Tasks.

Implement a Restful API with all HTTP methods, using python flask framework having your current semester transcript contains at least 5 courses that contain a JSON format data of (5 key-value pairs)

Your solution file contains

- Complete code snippet of your Restful API
- All routes output screens (in postman tool)
- All server responses screens (clearly shown GET/PUT/DELETE requests and respective HTTP codes)

DEADLINE: Sunday 18th December 2022 23:59