CS 504 Software Engineering

**HOS03A –Restful API** **using Flask**

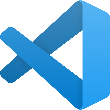
03/10/2023 Developed by Ajay Naik

08/25/2024 Reviewed by Lisa Schultz

09/09/2024 Reviewed by Yogesh Goel

10/16/2024 Reviewed by Naveena Moddu

School of Technology & Computing (STC) @ City University of Seattle (CityU)



**Before You Start**

* Version numbers may not match the most current version at the time of writing. If given the option to choose between the stable release (long-term support) or the most recent, please select the stable release rather than the beta-testing version.
* This tutorial targets Windows users and MacOS users.
* There might be subtle discrepancies along the steps. Please use your best judgment while going through this cookbook-style tutorial to complete each step.
* For your working directory, use your course number. This tutorial may use a different course number as an example.

**Instructions**

1. Complete all the steps described in this document.
2. At the end of the assignment, follow the instructions to commit your work to GitHub.

**Learning Outcomes**

* Flask Microservice
* CRUD Operation

**Resources**

* Loubser, N. (n.d.). Software Engineering for Absolute Beginners: Your Guide to Creating Software Products. In *O'Reilly Online Learning*. <https://learning.oreilly.com/library/view/software-engineering-for/9781484266229/>
* <https://flask.palletsprojects.com/en/2.2.x/>
* <https://www.postman.com/product/what-is-postman/>

**Introduction**

In this HOS, we will build a small project using python. We will use a framework called Flask. Flask is a microframework used to create REST APIs. We will use REST APIs, CRUD operations, and Postman.

**What is REST?**

REST, or RESTful, is a popular architectural pattern that allows a system to serve data to an external source, like a front end, reliably and predictably. It enables different systems, written in entirely different technologies, to communicate with your system. It is programming language-agnostic, but you must adhere to standards to make the interoperability between different systems dependable. REST stands for Representational State Transfer. In simple terms, REST is an HTTP, preferably an HTTPS, request like requesting a normal website, but with specific rules allowing for dependable and understandable data transfer.

Diagram

Description automatically generated

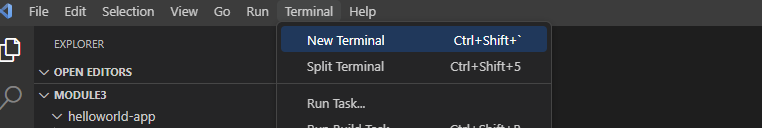
**Creating of Micro Service**

**Initializing The Project**

Create a Module 03 Folder.

Open your Visual Studio Code in the Module 03 folder

Go to Terminal and click on New Terminal.



Note: If you face any permission issue, windows users open a command prompt as Admin, and Linux/Mac users use sudo before the command.

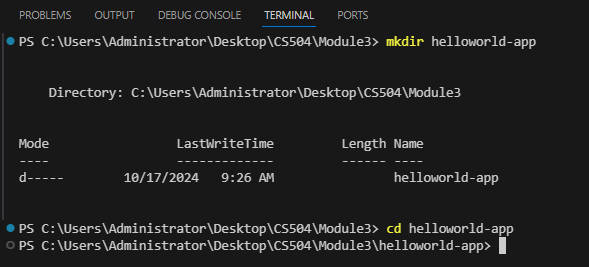
Create a helloworld-app folder and change the current working directory to it.

**Command**

mkdir helloworld-app

cd helloworld-app

**Output**



The above commands create a helloworld -app folder and change the current working directory to it.

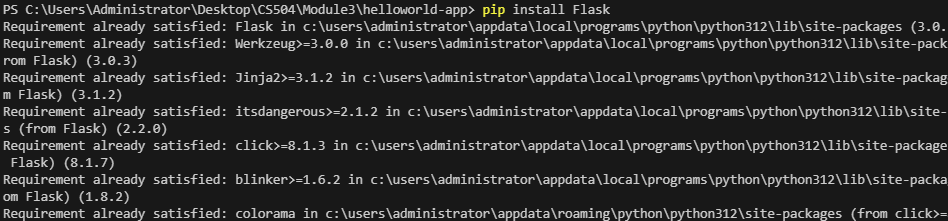
**Installing Flask**

Note: If you are using an older version or somehow pip is not installed, you will need to install it first, follow <https://pip.pypa.io/en/stable/installation/>

**Command**

pip install Flask

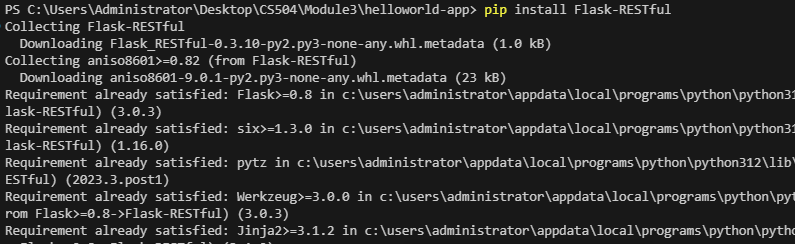
**Output**



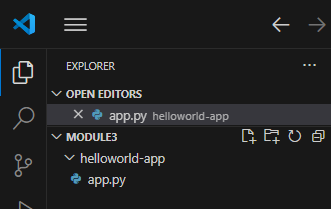
**Command**

pip install Flask-RESTful

**Output**



Create a New File named app.py inside helloworld-app



Add the following code in the app.py file

from flask import Flask

from flask import request

import json

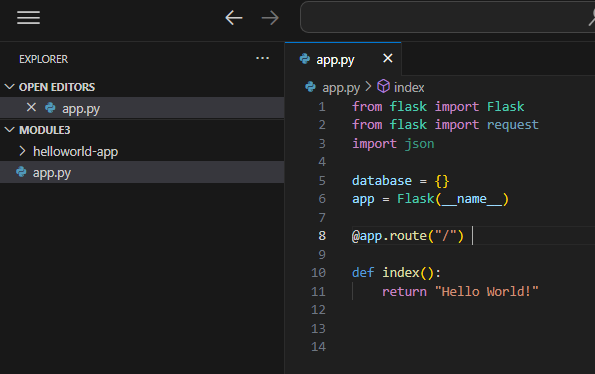
database = {}

app = Flask(\_\_name\_\_)

@app.route("/")

def index():

    return "Hello World!"



**Creating a CRUD Service**

An HTTP verb is something you use implicitly with every page load and every form submission on the Internet. It is a mandatory part of every HTTP call. In REST, you use the verb explicitly to tell the back-end system what the action will be. We are going to explore four HTTP protocols.

* GET retrieves data from the system.
* POST adds new data to the system.
* PUT replaces/modifies an entry entirely in the system.
* DELETE removes an entry from the system.

We are creating a CRUD Rest Service for students. In this CRUD operation, we will only be storing names and ages. Please note HTTP POST and PUT are interchangeable terms, and both protocols can serve the purpose of creation and updation.

To create a new Student Record in the system, add the code below to the app.py file.

@app.route('/students', methods=['POST'])

def post\_students\_details():

    try:

        data = request.json

        dict\_json = json.loads(json.dumps(data))

        database[dict\_json["name"]] = dict\_json["age"]

        return 'Success', 200

    except Exception as e:

        print("Error during saving object ", e)

        return 'Failed', 400

This code accepts the HTTP post request with the path 'Students.' It parses the student Json request and stores it in a temporary data structure.

**Updating Existing Records**

Updating/Modifying the existing records is the common request, and HTTP PUT request is the most used protocol.

Use the code below to update existing records.

**Code**

@app.route('/students', methods=['PUT'])

def put\_students\_details():

    try:

        data = request.json

        dict\_json = json.loads(json.dumps(data))

        database[dict\_json["name"]] = dict\_json["age"]

        return 'Success', 200

    except Exception as e:

        print("Error during saving object ", e)

        return 'Failed', 400

**Returning Existing Records**

The GET method retrieves information from the given server using a given URI. Requests using GET should only retrieve data and should have no other effect on the data.

Use the following code to get records

**Code**

@app.route('/students/<Student\_name>', methods=['GET'])

def get\_students\_details(Student\_name):

    try:

        name = database[Student\_name]

        if name == None:

            return 'Record Not Found', 404

        else:

            return 'Record Found ' + Student\_name + ' age is ' + str(name), 200

    except KeyError:

        return 'Record Not Found', 404

**Deleting Records**

An HTTP request of this type is sent to remove the target resource from the server. Precisely what happens on the server's side, such as whether the space is reclaimed or the resource is instead just taken offline, is under the server's control, and how its environment has been configured.

Use the following code

**Code**

@app.route('/students/<Student\_name>', methods=['DELETE'])

def delete\_students\_details(Student\_name):

    try:

        name = database[Student\_name]

        database.pop(Student\_name)

        return 'Record deleted successfully', 200

    except KeyError:

        return 'Record Not Found', 404

    except Exception as e:

        print("Error while removing record ", e)

        return 'Error while removing record', 400

**Starting Flask**

Run the following commands from the helloworld-app directory

Windows:

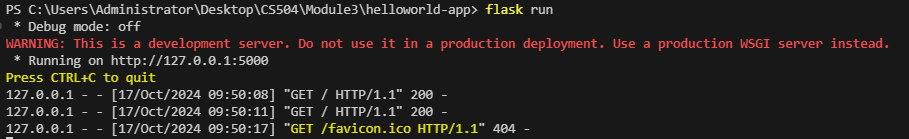
set FLASK\_APP=app.py

Mac:

export FLASK\_APP=app.py

Followed by:

flask run



Note: If you receive an error message indicating the term ‘flask’ is not recognized, try using this code:

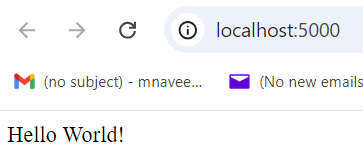
python –m flask run

**Testing the application**

Navigate to <http://localhost:5000/>

**Expected**

You should see Hello World in the browser

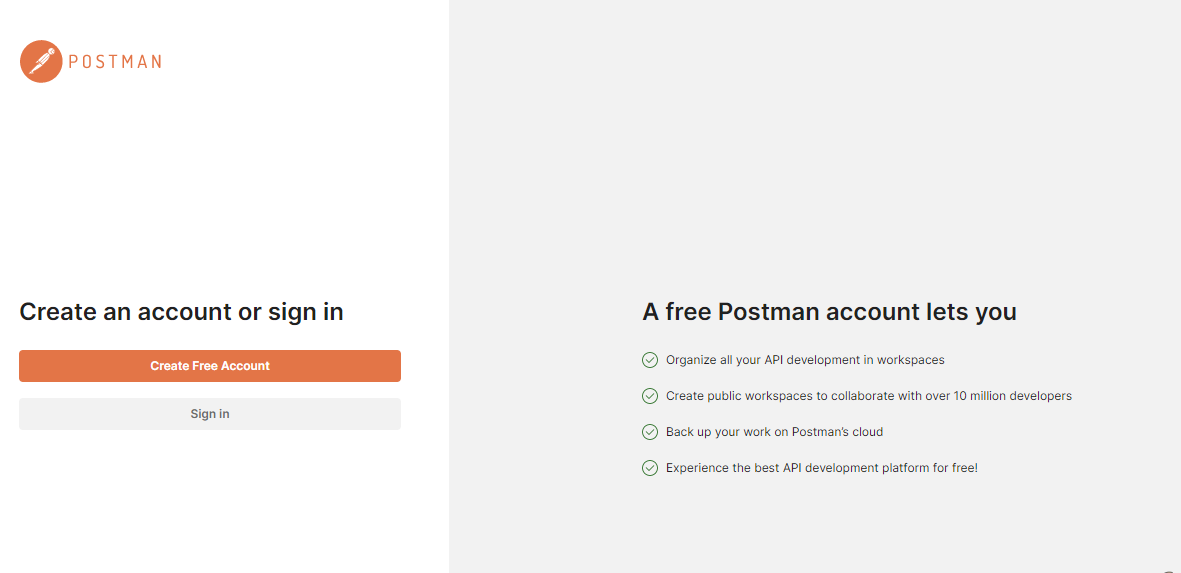


**Testing Crud Operations**

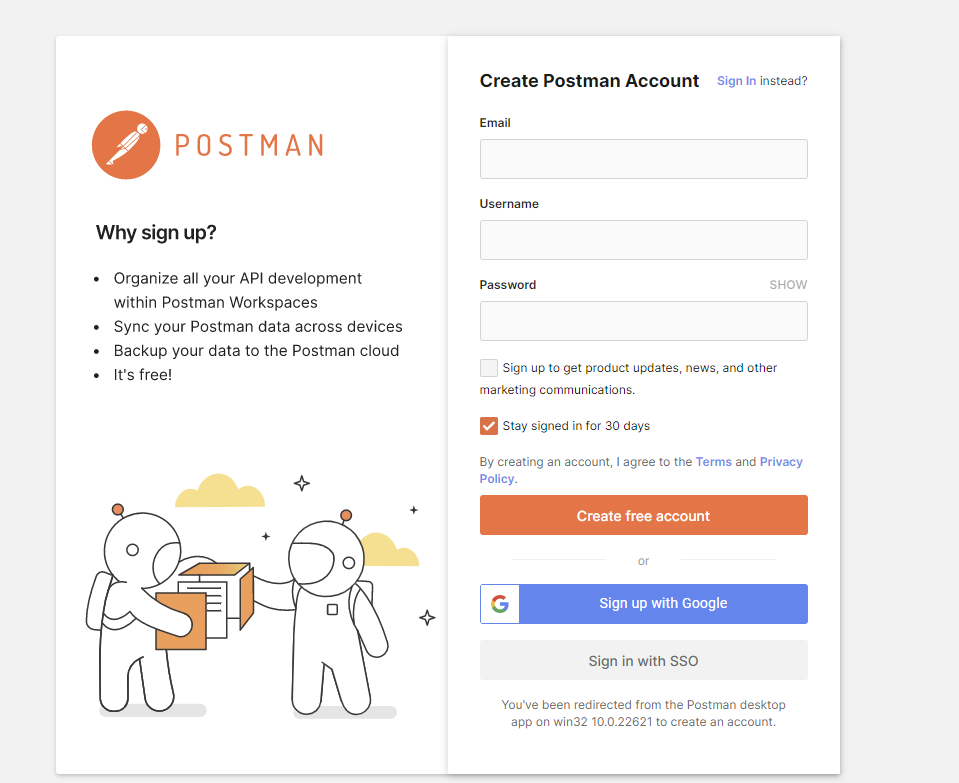
Download Postman from here:

<https://www.postman.com/downloads/>

Open Postman

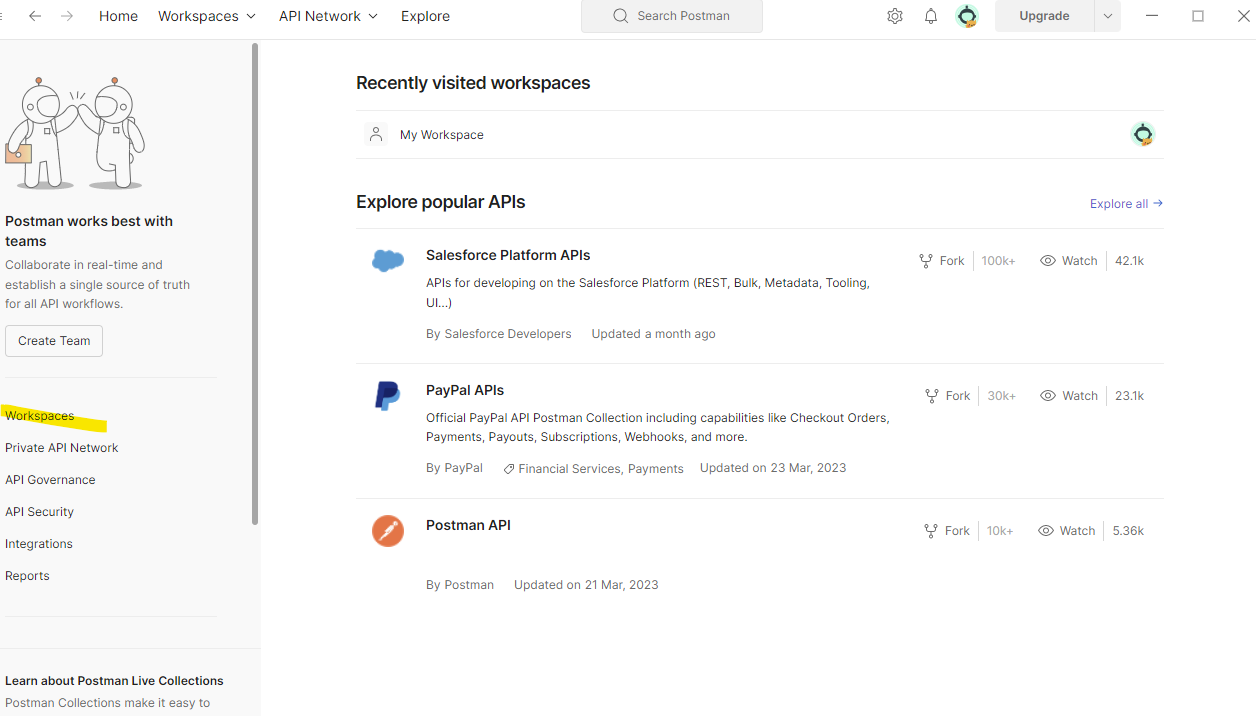


Click on Create Free Account.

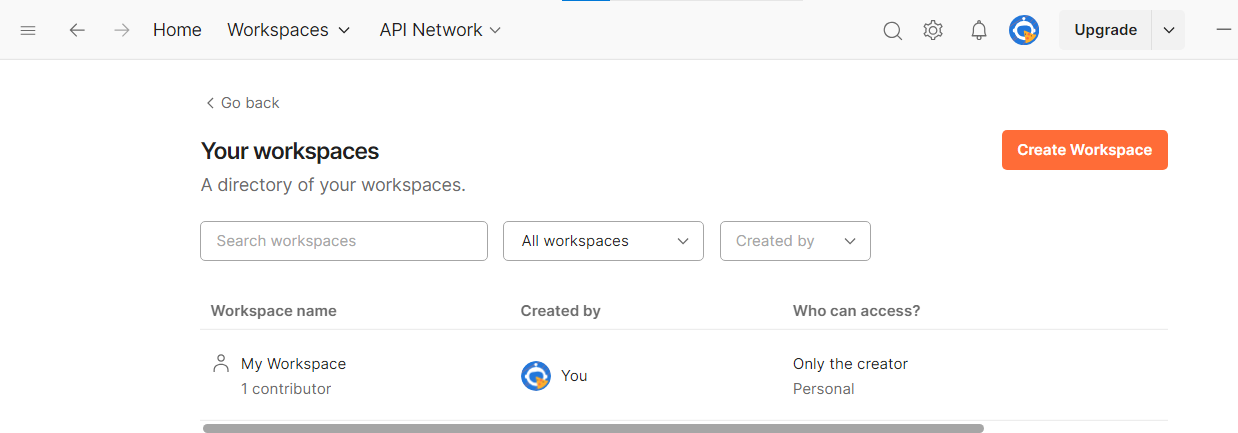


Postman will give two options either by entering credentials or by Sign Up with Google.

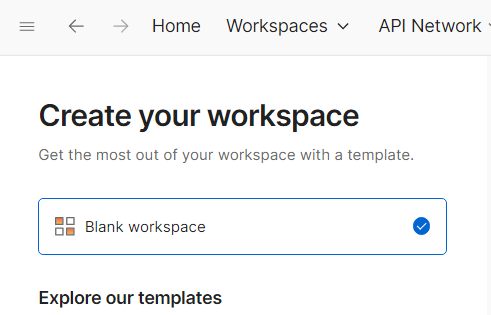
Click on Sign Up with Google and choose your account, Postman will redirect to home screen.

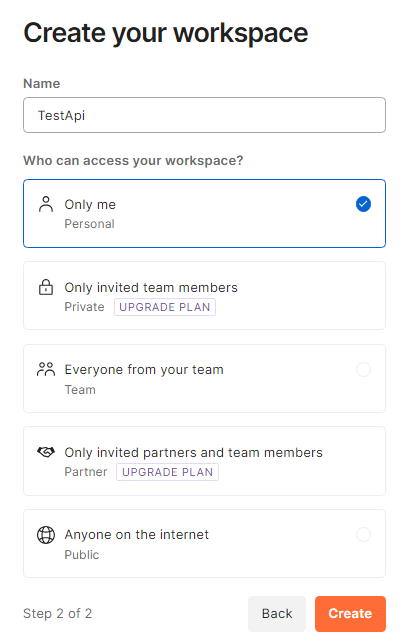


Click on create Workspace

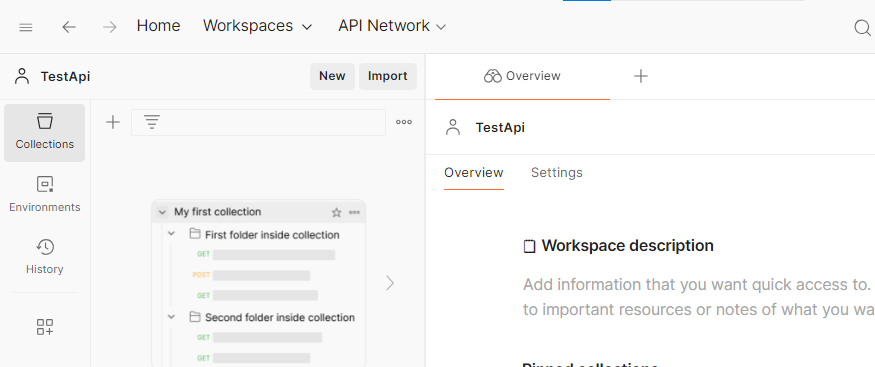


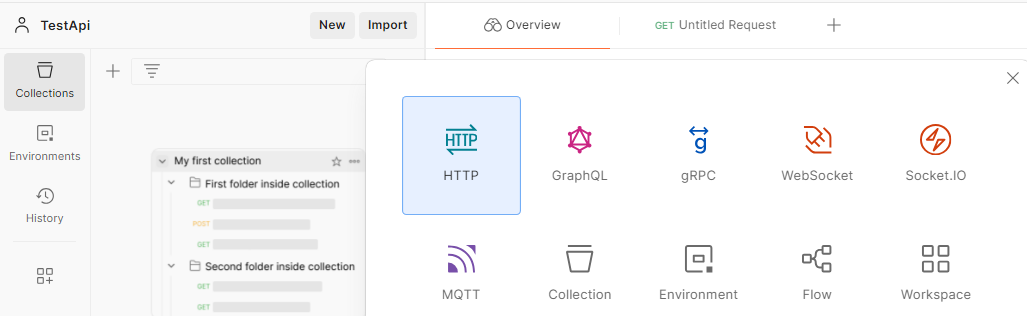
Select Blank Workspace. Enter Name as TestApi and click on Create.

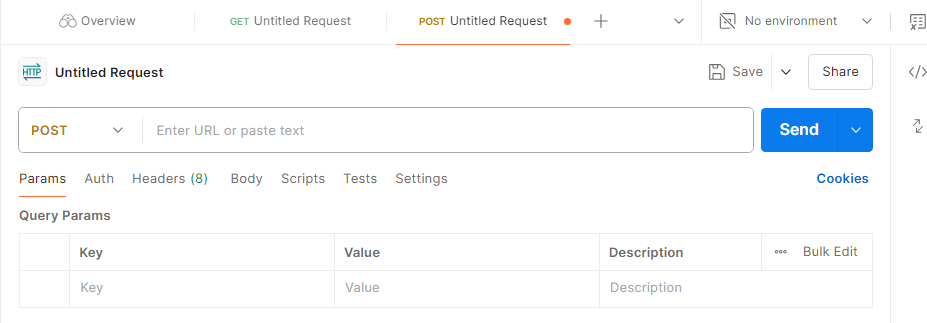




Click on “New” next to TestApi and select HTTP.



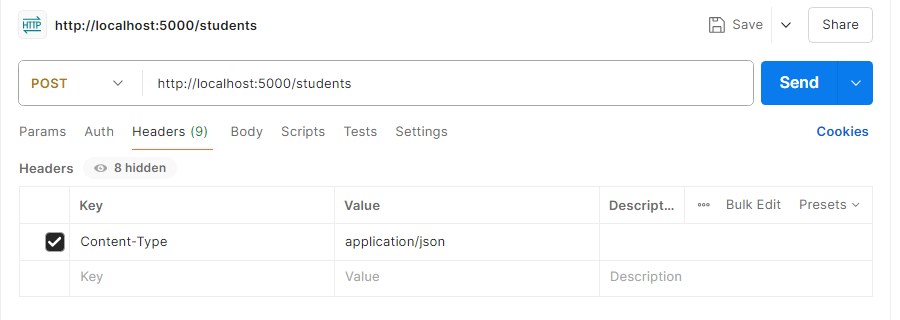




Select Post

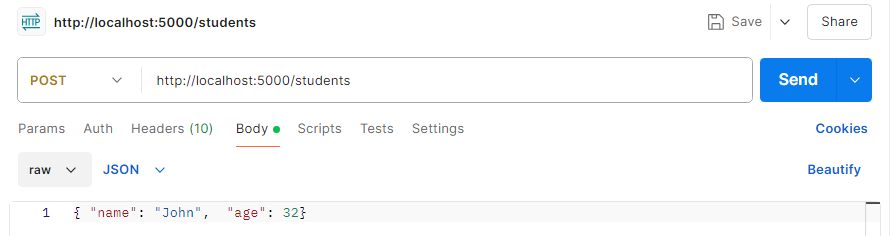
Enter the URL: http://localhost:5000/students

Under headers, add Content-Type -> application/json



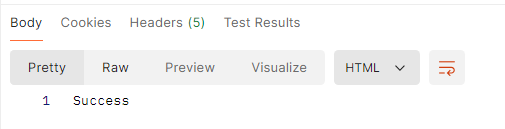
Under the body, select raw and enter the following json

{ "name": "John", "age": 32}



Click Send

The following should be displayed.



**Modifying records**

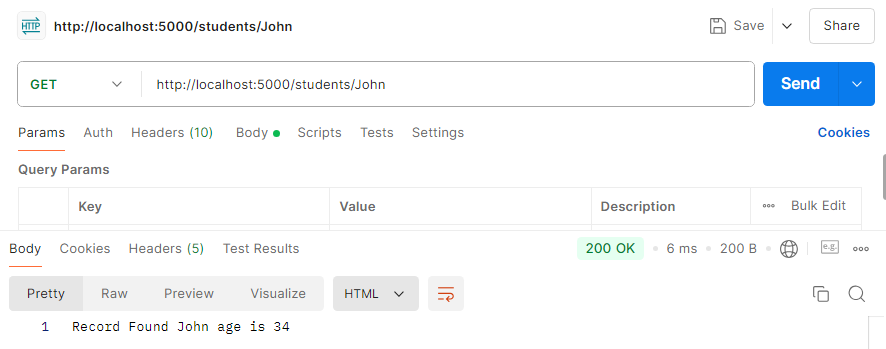
Follow the steps before, but this time use the PUT method and the json below. This will modify the record for John to 34

{ "name": "John", "age": 34}

**Retrieving records**

In Postman, use the method GET and the URL http://localhost:5000/students/John

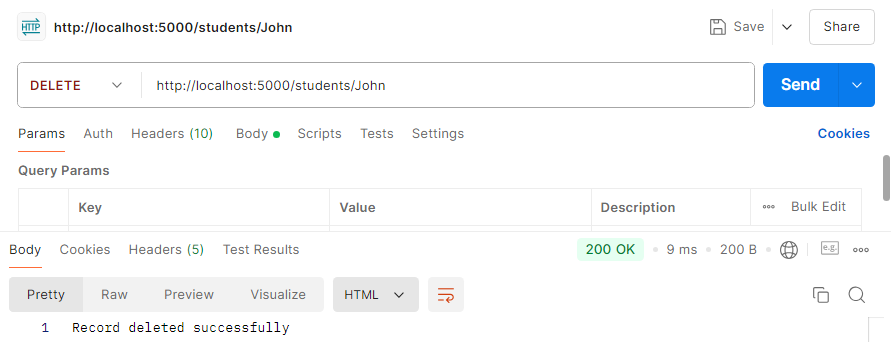
Postman should display the following.



**Deleting Records**

In Postman, select DELETE and use the URL <http://localhost:5000/students/John>

Postman should display the following:



**Push your work to GitHub**

Open the Terminal from the VS Code by hitting the control + ~ key and typing the following command:

Run the following commands to push your work to the GitHub repository:

git add .

git commit -m "Submission for Module 3"

git push