IMPLEMENTING WEB APPLICATION

Project Date	**September 2022**
Team ID	**PNT2022TMID13786**
Project Name	**Containment Zone Alerting Application**

Create APIs in Flask:

Step 1: Flask Installation and Server Setup

We are assuming that you have already installed Python, and it's up to date. So let's set up our project and set up a virtual environment.

So now open the terminal type the below command:

```
pip3 install virtualenv
```

Once it's installed, let's create a directory or folder for your project. We are using `flask-test` as the folder name, but you can pick any name for your project.

```
mkdir flask-test
```

Then, change the directory:

```
cd flask-test
```

Now it's time to create a virtual environment for our project so that the dependencies don't mess up the global package folder. Execute the below command now:

```
virtualenv .
```

And then,

```
source bin/activate
```

Explanation: What we are doing here is we are telling the module that the current folder can be used for the virtual environment, and then it is activating the virtual environment in the second step.

Now once you have activated the virtual environment, let's install the Flask package in that environment.

Now run the below command:

```
Python3 -m pip install Flask
```

So far, we have created our project folder, installed and created a virtual environment for our project, and installed Flask in the environment. Let's head towards Step 2.

Step 2: Let's Write Some Code

Now create a file `app.py` and paste the below code:

```
from flask import Flask app

= Flask(__name__)




@app.route('/') def

hello_world():

    return 'This is my first API call!'
```

Make sure to save your `app.py` file to the current directory.

Code Explanation: First, we are importing the `flask` module into our application and then defining the API route. The `@` is used for defining the API route. We're passing `/`, which means this is our base route

Step 3: Running the Server and Making the First API Call

Once you're done with the coding part, it's time to run our Flask server and make our first API call.

To run the server, execute the below command:

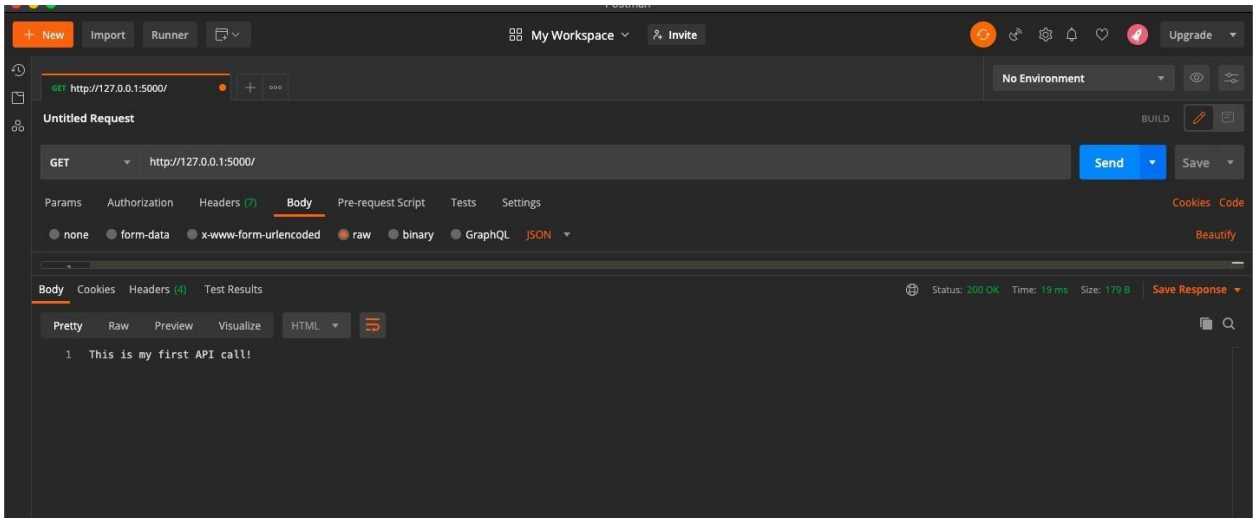
```
flask run
```

You should see the below output on the terminal:

```
* Environment: production

   WARNING: This is a development server. Do not use it in a production deployment.

   Use a production WSGI server instead.

* Debug mode: off

* Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```

Now open your favorite API testing tool. We'll be using Postman in the tutorial below, but you can pick any of the tools mentioned here.

Now copy and paste the URL printed on the terminal and make a GET request. You should see the below output:



Step 4: POST APIs

Flask makes it very easy to program different commands for various HTTP methods like POST, GET, PUT, etc. In the above code, you can see there's the function `route`. The second parameter passed to this function is actually the method of the route. If nothing is passed then, it is GET.

But we also have to import two additional modules named `request` and `jsonify` used to fetch the params and JSON conversion.

Now lets define another route for our POST requests. So open the `app.py` file and replace the existing code with the below code:

```
from flask import Flask, request, jsonify

 app =

Flask(__name__)



@app.route('/') def

hello_world():

    return 'This is my first API call!'


@app.route('/post', methods=["POST"]) def

testpost():

     input_json = request.get_json(force=True)
dictToReturn = {'text':input_json['text']}
```

```
    return jsonify(dictToReturn)
```

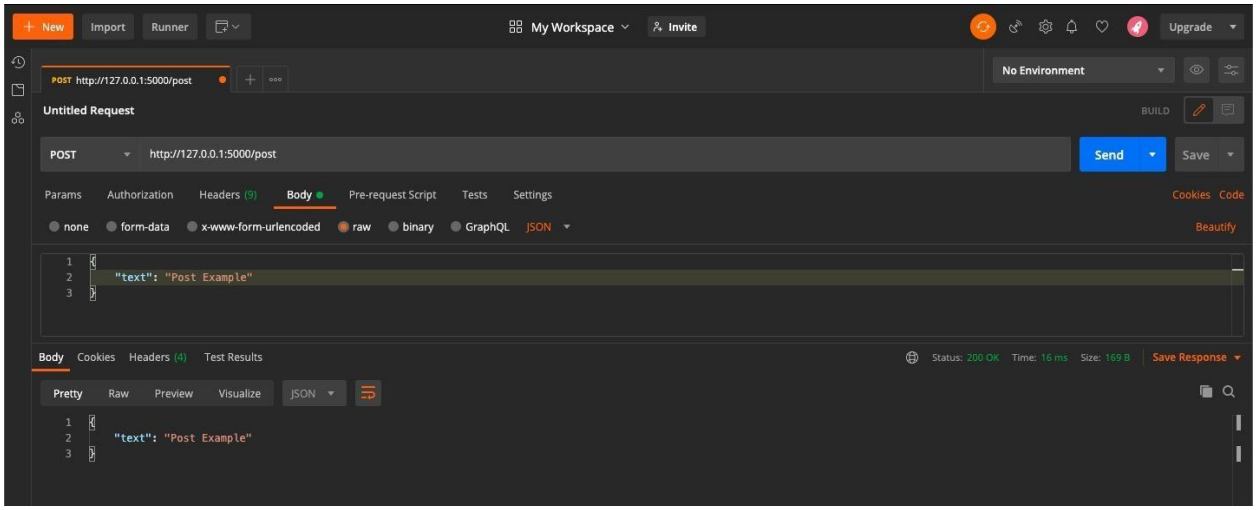
Code Explanation: Here, we have imported some more modules from Flask, like `request` and `jsonify`. We are using `request` to get the data which the user is sending, and we're using `jsonify` for converting dictionaries to JSON. We have added one more route that is `/post` and also passing `POST` as a list and returning back what the user is sending in the parameters.

Now once again head over to the API testing tool and hit URL: http://127.0.0.1:5000/post with parameters:

```
{
    "text": "Post Example"
}
```

Since we edited `app.py`, before you run a `POST` API call, you will need to restart the virtual server. To do so, (Press CTRL+C to quit), and then enter `flask run` into the Terminal again.

You should see the below response:



A Basic API With Flask

Now we're done with the most basic Flask tutorial with one GET and one POST API endpoint. This tutorial was just a gist so that you can understand the basics of Flask.

Flask is a very powerful tool, and on an advanced level, you can achieve a lot of things with it. For example, you could add authentication with JWT or OAuth. You can also connect the code with a MySQL backend and perform CRUD operations.

I hope you learned something new from this article, and hopefully, I made it easier for you to understand Flask and API creation basics.