St. Francis Institute of Technology, Mumbai-400 103

A.Y. 2023-24

Class: SE-ITA/ITB, Semester: IV

Subject: **Python Lab**.

Experiment – 15: Building a simple REST API using flask.

1. **Aim:** To build a REST API using flask
2. **Prerequisite:** knowledge of the command line, Python, and web concepts .
3. **Objective:** To understand API.
4. **Requirements:** Personal Computer (PC), Windows /Linux Operating System, IDLE 3.6 for Python3.
5. **Pre-Experiment Exercise: Theory:**

Flask a web framework for Python, meaning that it provides functionality for building web applications, including managing HTTP requests and rendering templates.

REST API stands for Restful API that allows integrating applications or interaction with RESTful web services. It is now growing as the most common method for connecting components in a microservice architecture. APIs will enable you to get or send data to a website and perform some action to get your task done over a web service. Each website uses different types of API, like stock market trading websites integrating with Sensex or Nifty to get a current price and ups-down. Ticket booking apps use a desired single portal API to keep updated data at a familiar interface.REST stands for REpresentational State Transfer and is an architectural style used in modern web development. It defines a set or rules/constraints for a web application to send and receive data.Flask is a popular micro framework for building web applications. Since it is a micro-framework, it is very easy to use and lacks most of the advanced functionality which is found in a full-fledged framework. Therefore, building a REST API in Flask is very simple.

There are two ways of creating a REST API in Flask:

1. Using Flask without any external libraries
2. Using flask_restful library

Libraries required:

flask_restful can be installed via the pip command:

```
sudo pip3 install flask-restful
```

6. **Laboratory Exercise**

A. Procedure:

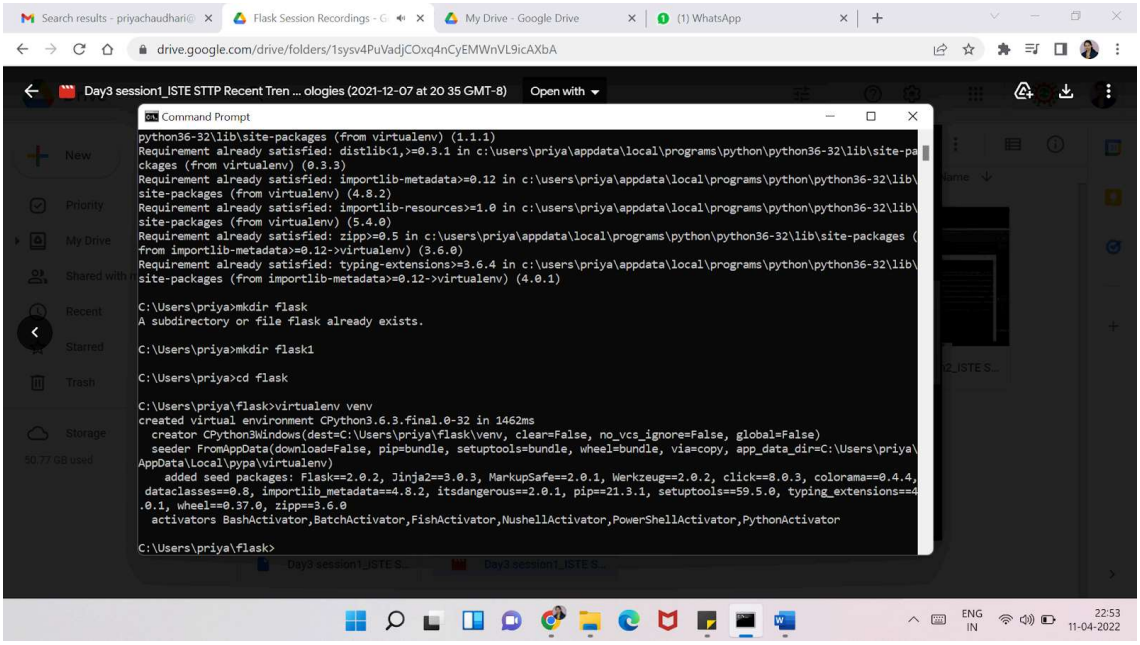
we will build a REST API in Python using the Flask framework. Flask is a popular micro framework for building web applications. Since it is a micro-framework, it is very easy to use and lacks most of the advanced functionality.

All steps of installation and execution:

a. pip install virtualenv

Create new folder for the environment

I. mkdir flask1
II. cd flask1
III. virtualenv venv



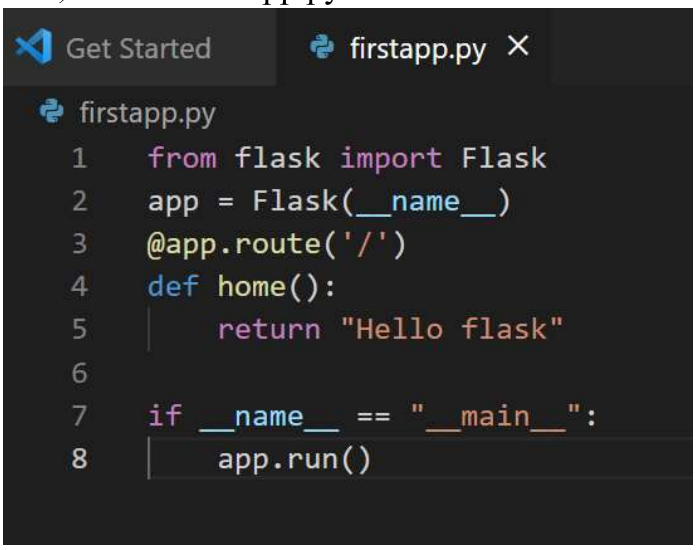
1. activate the envioronemnt
venv\scripts\activate

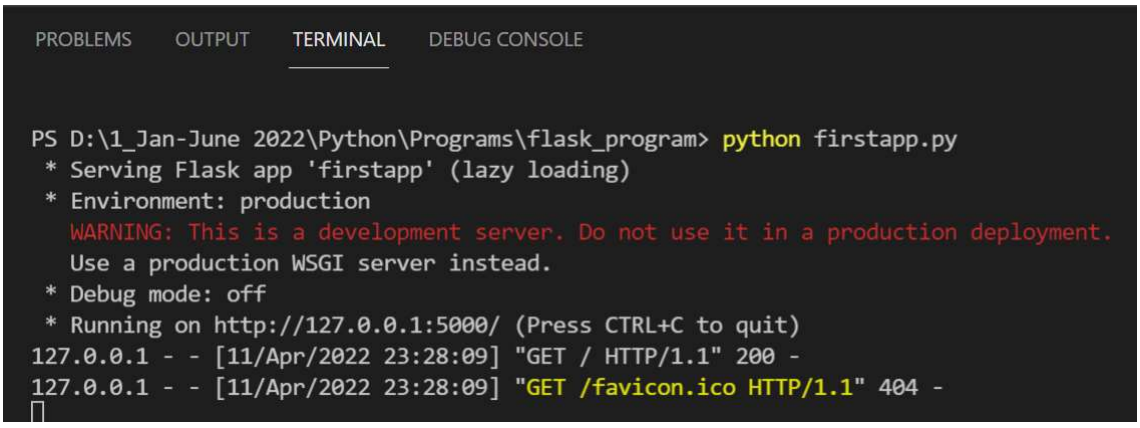
```
C:\Users\priya\flask>venv\scripts\activate

(venv) C:\Users\priya\flask>
```

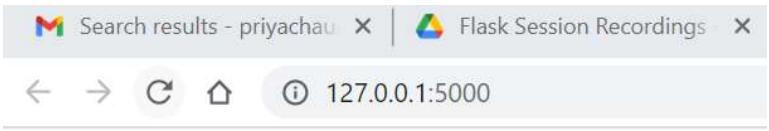
1. pip install flask

```
(venv) C:\Users\priya\flask>pip install flask
Requirement already satisfied: flask in c:\users\priya\flask\venv\lib\site-packages (2.0.2)
Requirement already satisfied: itsdangerous>=2.0 in c:\users\priya\flask\venv\lib\site-packages (from flask) (2.0.1)
Requirement already satisfied: Werkzeug>=2.0 in c:\users\priya\flask\venv\lib\site-packages (from flask) (2.0.2)
Requirement already satisfied: click>=7.1.2 in c:\users\priya\flask\venv\lib\site-packages (from flask) (8.0.3)
Requirement already satisfied: Jinja2>=3.0 in c:\users\priya\flask\venv\lib\site-packages (from flask) (3.0.3)
Requirement already satisfied: colorama in c:\users\priya\flask\venv\lib\site-packages (from click>=7.1.2->flask) (0.4.4
)
Requirement already satisfied: importlib-metadata in c:\users\priya\flask\venv\lib\site-packages (from click>=7.1.2->fla
sk) (4.8.2)
Requirement already satisfied: MarkupSafe>=2.0 in c:\users\priya\flask\venv\lib\site-packages (from Jinja2>=3.0->flask)
(2.0.1)
Requirement already satisfied: dataclasses in c:\users\priya\flask\venv\lib\site-packages (from Werkzeug>=2.0->flask) (0
.8)
Requirement already satisfied: typing-extensions>=3.6.4 in c:\users\priya\flask\venv\lib\site-packages (from importlib-m
etadata->click>=7.1.2->flask) (4.0.1)
Requirement already satisfied: zipp>=0.5 in c:\users\priya\flask\venv\lib\site-packages (from importlib-metadata->click>
=7.1.2->flask) (3.6.0)
```

1) Create a folder
2) Open python idle and import module
3) Create firstapp.py file.

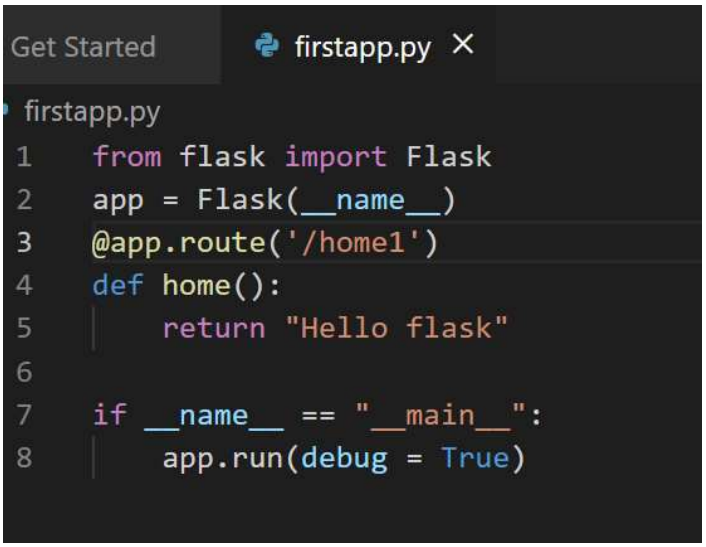






Hello flask

```python
from flask import Flask
app = Flask(__name__)
@app.route('/')
def home():
    return "Hello flask"

if __name__ == "__main__":
    app.run(debug = True)
```

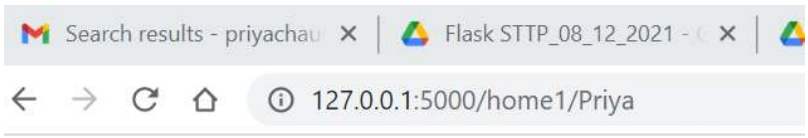


Hello flask

Dynamic URLs

firstapp.py

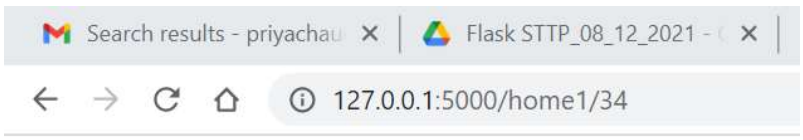
```
1   from flask import Flask
2   app = Flask(__name__)
3   @app.route('/home1/<name>')
4   def home(name):
5       return "Hello flask "+name
6
7   if __name__ == "__main__":
8       app.run(debug = True)
```



Hello flask Priya

firstapp.py

```
from flask import Flask
app = Flask(__name__)
@app.route('/home1/<int:age>')
def home(age):
    return "age: %d"%age

if __name__ == "__main__":
    app.run(debug = True)
```



age: 34

B. Program code with comments:
Write commands.

7. Post-Experiments Exercise

a. **Extended Theory:**

Understanding HTTP Request through Flask RESTApi

b. **Questions/Programs:**

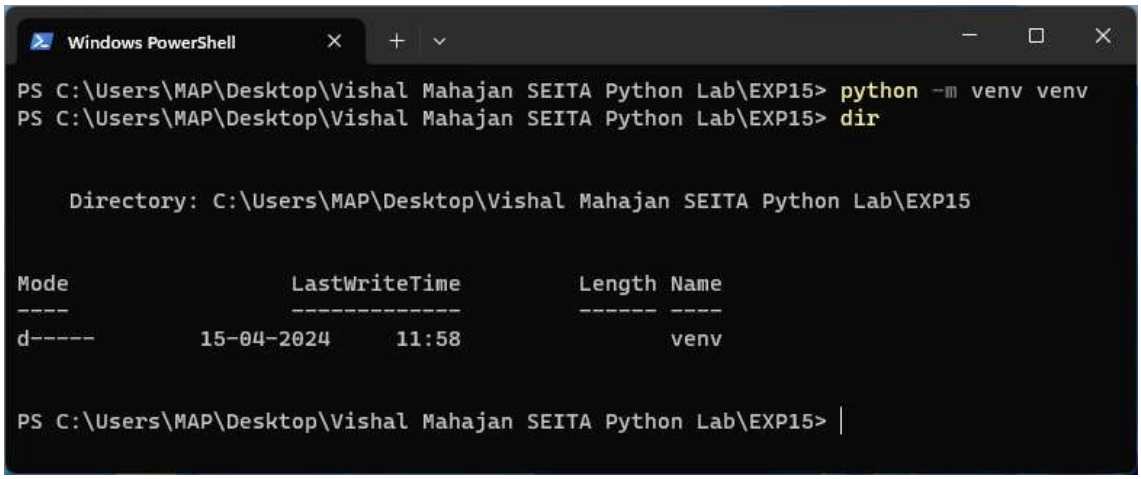
c. **Conclusion:** Conclude what you understood after performing this experiment.

d. **References**

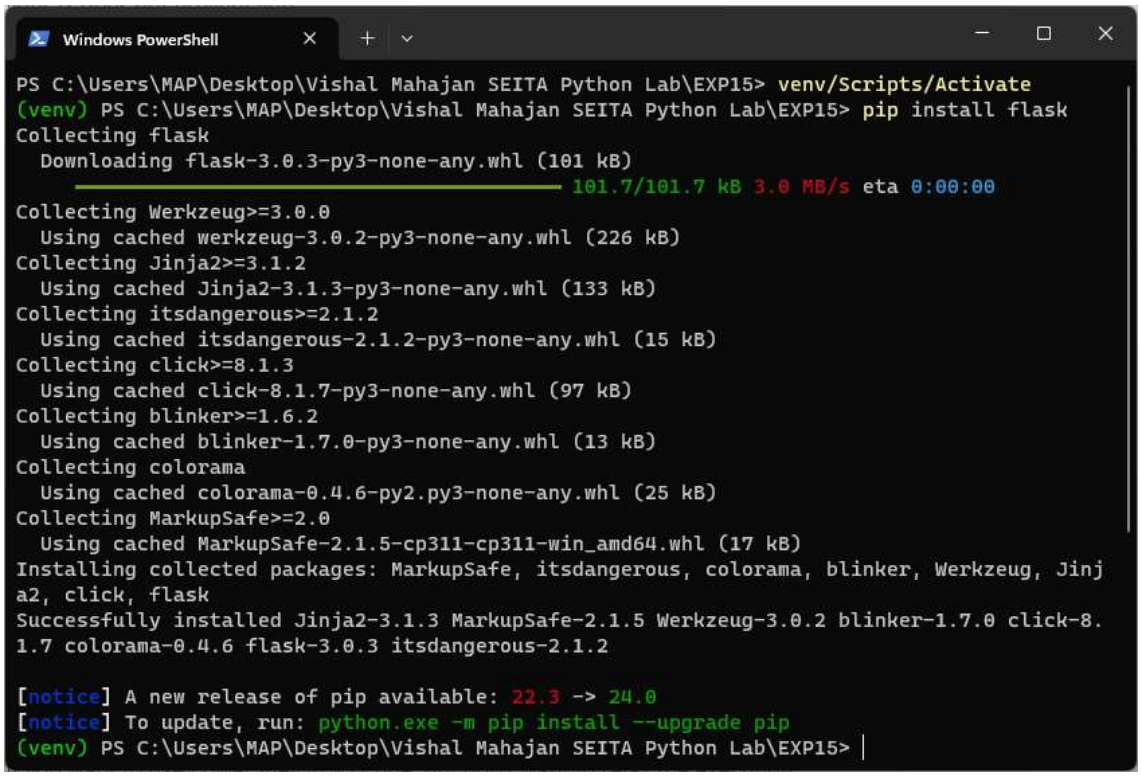
1. https://towardsdatascience.com/creating-restful-apis-using-flask-and-python
2. **https://www.analyticsvidhya.com/blog/2022/01/rest-api-with-python-and-flask/**
3. **https://www.tutorialspoint.com/flask/index.htm**

In-Lab Exercise:

1. Creating a Virtualenv



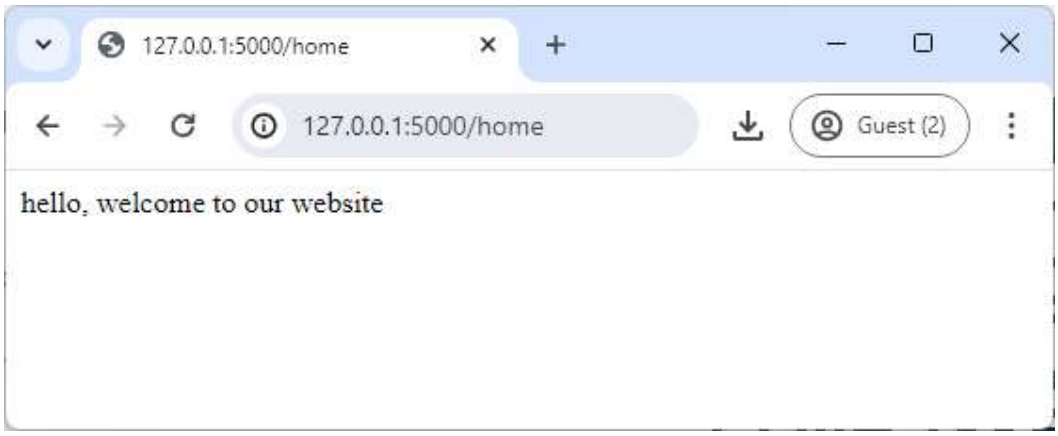
2. Installing Flask



3. Creating a /home route in flask to return a basic page

```python
from flask import Flask
app = Flask(__name__)

@app.route('/home')
def home():
    return "hello, welcome to our website";

if __name__ =="__main__":
    app.run(debug = True)
```

Output:



4. Creating a Login Form using HTML

```
<html>
   <body>
      <form action = "http://localhost:5000/login" method = "post">
         <table>
        <tr><td>Name</td>
        <td><input type ="text" name ="uname"></td></tr>
        <tr><td>Password</td>
        <td><input type ="password" name ="pass"></td></tr>
        <tr><td><input type = "submit"></td></tr>
    </table>
      </form>
   </body>
</html>
```

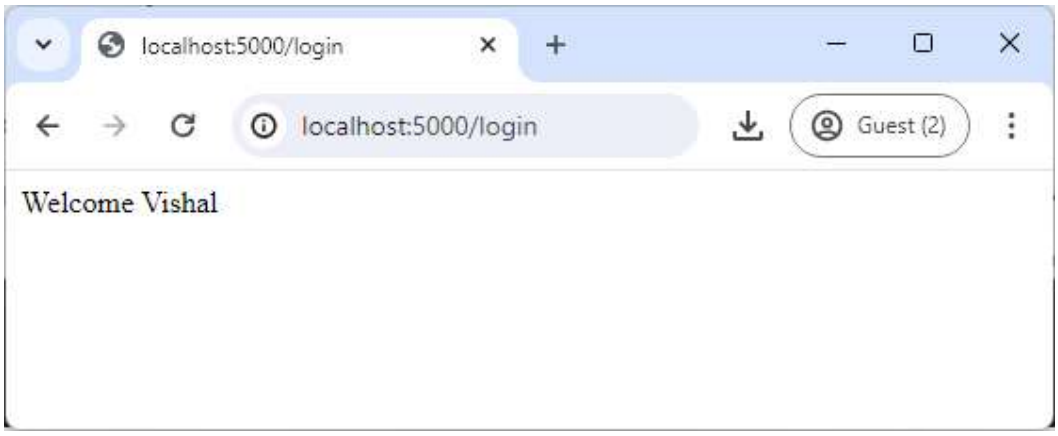
Form:

5. Using a Post method to Post Form data to a Flask “\login” endpoint

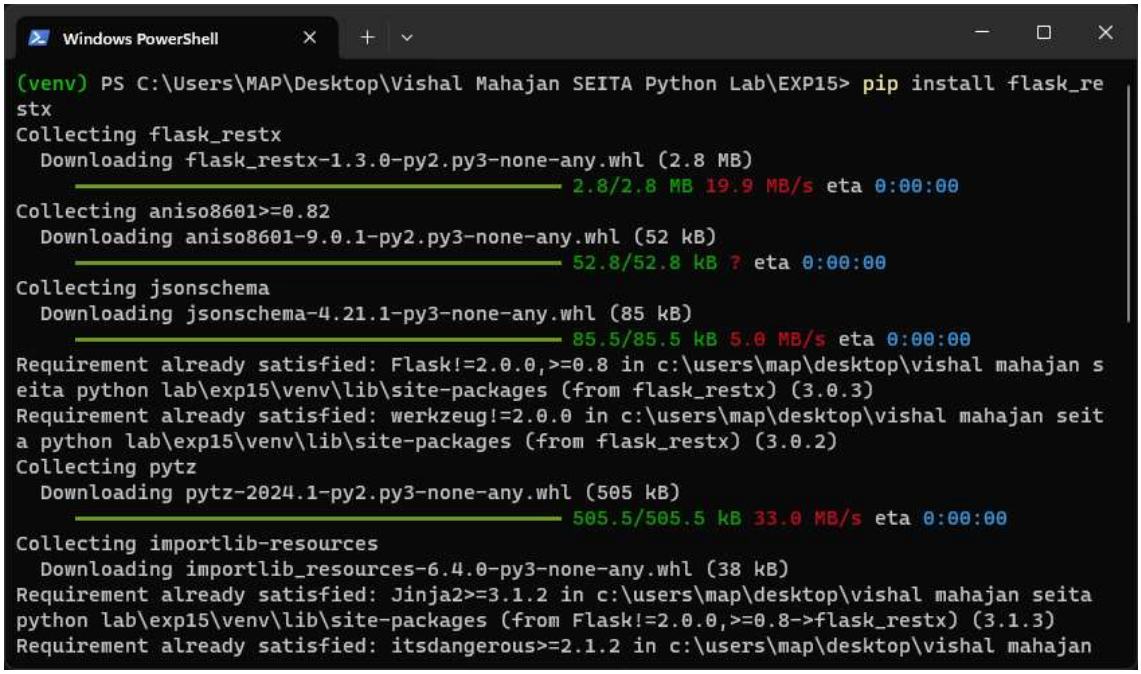
```
from flask import *
app = Flask(__name__)

@app.route('/login',methods = ['Post'])
def login():
      uname=request.form['uname']
      passwrd=request.form['pass']
      if uname=="Vishal" and passwrd=="Mahajan":
          return "Welcome %s" %uname

if __name__ == '__main__':
   app.run(debug = True)
```

Post Form Data [uname= Vishal and passwrd = Mahajan]

Redirect to /login with Welcome Vishal



6.Installing a Flask_restx Library to for resource and API



7. Creating a API Testing Page using flask_restx (Swagger)

```python
from flask import Flask, request, jsonify
from flask_restx import Resource, Api

app = Flask(__name__)

api = Api(app)

class Helloworld(Resource):

    def __init__(self):

        pass

    def get(self):

        return {

            "Hello": "World"
```

```
        }

api.add_resource(Helloworld, '/')

if __name__ == '__main__':

    app.run(debug=True)
```

Output:

We can test endpoint using execute as below: