Page No.
Date / /

Assignment No. - 07

Aim :- Create a simple web service and write any distributed application to consume the web service.

Objective :- To understand web services, and how distibuted applications can be developed to consume web services.

Infrastructure :- Python environment.

Software Requirements :- Python 3.0, Flask, request library.

Theory :-

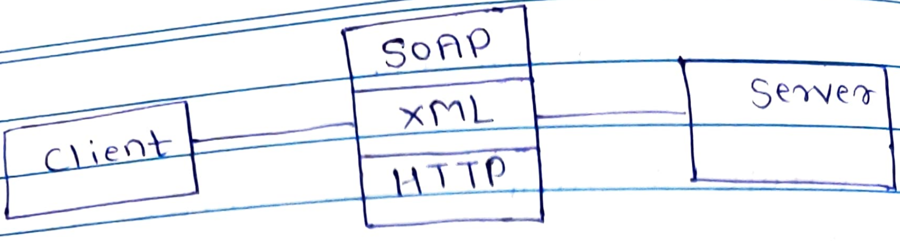
What are Web Services?

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 & running on various platforms can use web services to exchange data over computer networks like Internet in a manner similar to inter-process communication on a single computer.

Components of Web Services :-

The basic web services platform is XML + HTTP. All the standard web services work using the following components :-

- SOAP (Simple Object Access Protocol)
- UDDI (Universal Description, Discovery & Integration)
- WSDL (Web Services Description Language)



How Does a Web Service Work?

A web service enables communication amo various applications by using open standards s HTML, XML, WSDL, and SOAP. A web service ta help of :-

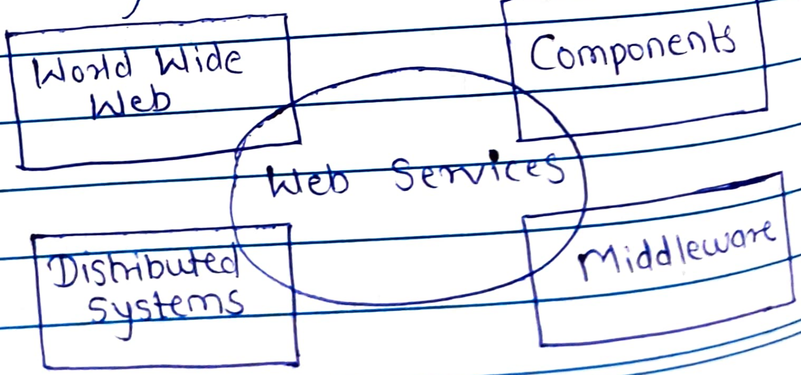
- XML to tag the data
- SOAP to transfer a message.
- WSDL to describe the availability of service

Types of Web Services :-

There are mainly two types of web services
(I) SOAP web services.
(II) RESTful web services.

Distributed Systems and Web Services :-

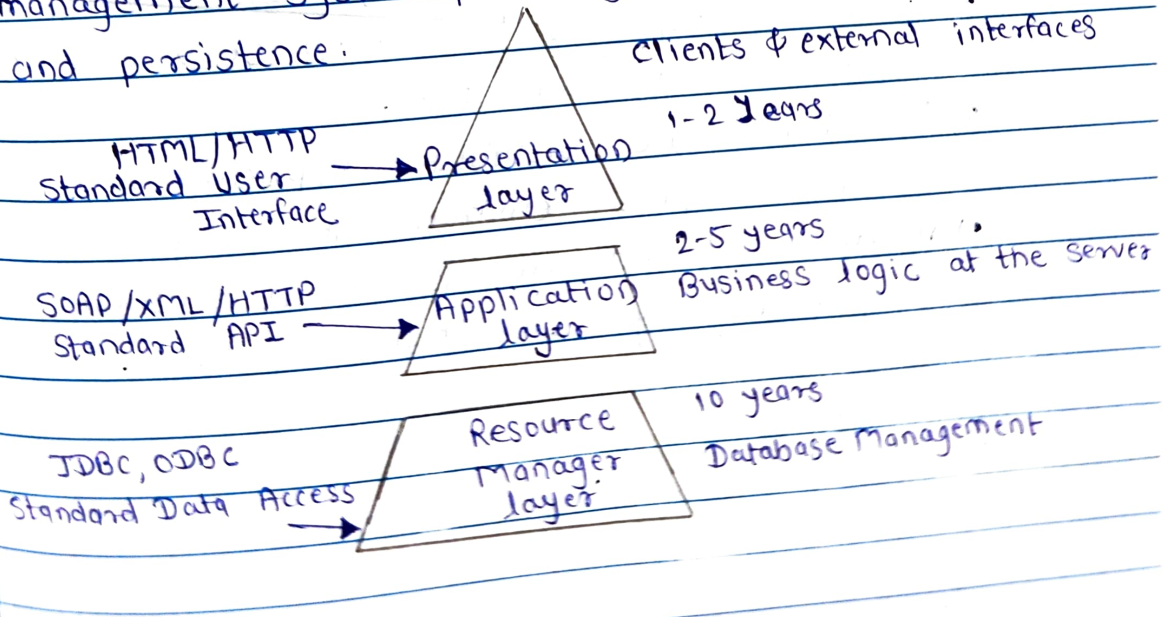
- Web services provide standards for developing scale distributed system.
- Web services on the path of success while distributed objects failed (This is nothing techn only a matter of widespread industry accepta



Page No.
Date

Layers in Distributed Systems:

- Client is any user or program that wants to perform an operation over system. To support a client, the system needs to have a presentation layer through which the user can submit operations and obtain a result.
- The application logic establishes what operations can be performed over system & how they take place. It takes care of enforcing business rules & establish the business processes. The application logic can be expressed & implemented in many different ways: constraints, business processes, server with encoded logic.
- The resource manager deals with organization (storage indexing & retrieval) of the data necessary to support the application logic. This is typically a database but it can also be a text retrieval system or any other data management system providing querying capabilities and persistence.



Steps involved in development of Web Service
distributed application to utilize this web se
1) Setting up web service :-
• Choose a programming language & framewor
web service.
• Define functionality & endpoints of your we
simplicity, let's assume you want to create
calculator API with two endpoints : POST /ad
2) Deploying Web Service :-
You can deploy the web service on any cl
Alternatively, you can run a local server of
3) Building the Distributed Application :-
• Define the functionality of your distributed
In this case, you'll create an application t
calculator API endpoints.
• Implement the logic to make HTTP requests
service endpoints. You can use libraries like
JavaScript or requests in Python to send HT
• Parse the responses from web service & ha
any errors that may occur.
4) Test & Run the Distributed Application :-
• Set up development environment for your dist
• Run distributed application & ensure it consumes
• Debug & fix any issues that may arise.

Conclusion :-
We learnt :- • how the web service works
• how to use web service in a distributed app
• Implementation of web and di
application.

```
NAME: PALLAVI K. CHOPADE
ROLL NO.: 14
PRN NO.: 72036169K
BE (IT)
LP - V

#app.py

from flask import Flask, request, jsonify

app = Flask(__name__)

@app.route('/add', methods=['POST'])
def add():
    data = request.get_json()
    num1 = data['num1']
    num2 = data['num2']
    num3 = num1 + num2
    return jsonify({"result": num3})

@app.route('/multiply', methods=['POST'])
def multiply():
    data = request.get_json()
    num1 = data['num1']
    num2 = data['num2']
    num3 = num1 * num2
    return jsonify({"result": num3})

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

#client.py

import requests

url = 'http://127.0.0.1:5000/'

def add_num(num1, num2):
    endpoint = url + 'add'
    data = {"num1": num1, "num2": num2}
    response = requests.post(endpoint, json=data)
    result = response.json()['result']
    return result

def multiply_num(num1, num2):
    endpoint = url + 'multiply'
    data = {"num1": num1, "num2": num2}
    response = requests.post(endpoint, json=data)
    result = response.json()["result"]
    return result

state = True

while state:
    try:
        print("Enter the first number:")
        num1 = int(input())
        print("Enter the second number:")
        num2 = int(input())

        print("Do you want to:\n1. Add\n2. Multiply\n3. Exit")
        choice = int(input(""))

        if choice == 1:
            print(add_num(num1, num2))
            print("Do you wish to continue? (Yes, No)")
            if input().lower() == "no":
                state = False
    elif choice == 2:
        print(multiply_num(num1, num2))
        print("Do you wish to continue? (Yes, No)")
        if input().lower() == "no":
            state = False

    elif choice == 3:
        print("Thank you for using the service")
        state = False

    else:
        print("Invalid Input")

    if state:
        print("New Calculation")
        print("_" * 10, end="\n")

except Exception as e:
    print("Encountered Error:", str(e))
    print("Restarting interface", end="\n")
```

Assignment No.- 07

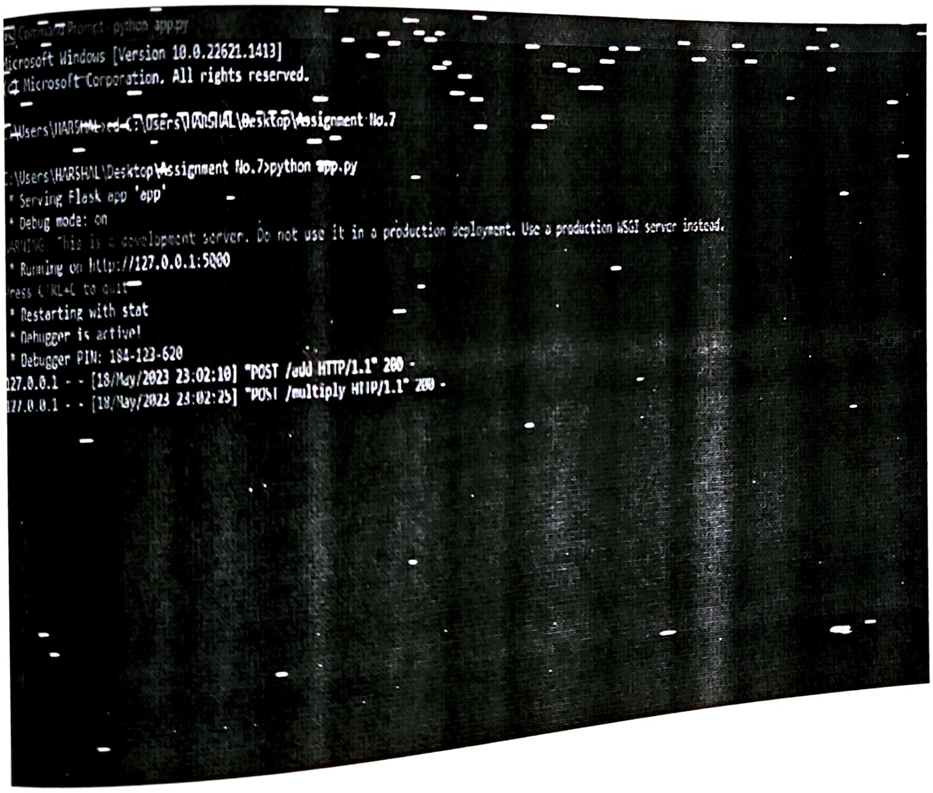
Name: - Pallavi Kamlakar Chopade.

Roll No.: - 14

PRN No.: - 72036169K

Subject: - Distributed Systems

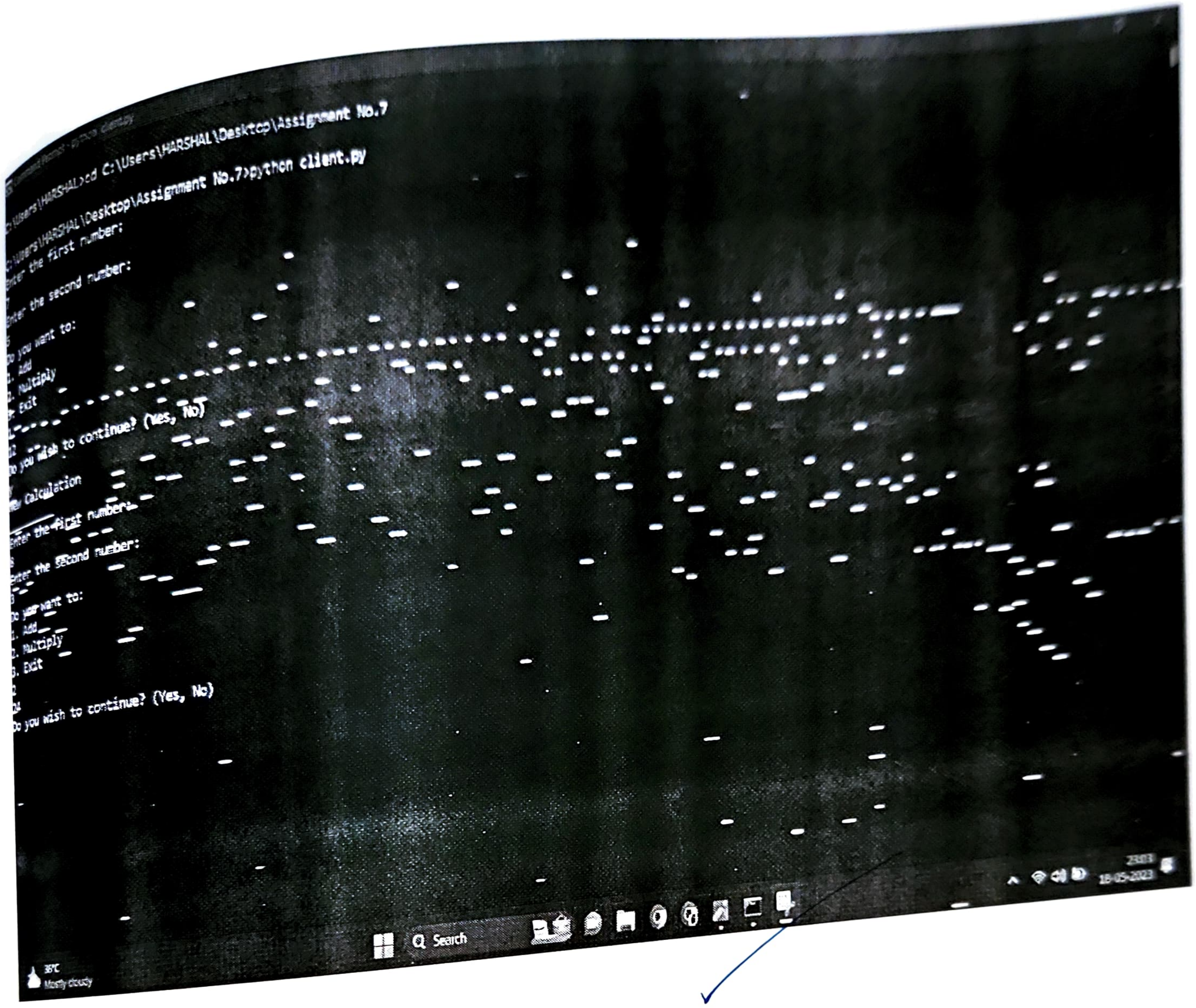
Class: - BE(IT)



```
Command Prompt - python app.py
icrosoft Windows [Version 10.0.22621.1413]
c) Microsoft Corporation. All rights reserved.

:\Users\HARSHAL>cd C:\Users\HARSHAL\Desktop\Assignment No.7

:\Users\HARSHAL\Desktop\Assignment No.7>python app.py
 * Serving Flask app 'app'
 * Debug mode: on
ARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on http://127.0.0.1:5000
ress CTRL+C to quit
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 184-123-620
127.0.0.1 - - [18/May/2023 23:02:10] "POST /add HTTP/1.1" 200 -
127.0.0.1 - - [18/May/2023 23:02:25] "POST /multiply HTTP/1.1" 200 -
```



```
C:\Users\HARSHAL>cd C:\Users\HARSHAL\Desktop\Assignment No.7

C:\Users\HARSHAL\Desktop\Assignment No.7>python client.py
Enter the first number:
7
Enter the second number:
5
Do you want to:
1. Add
2. Multiply
3. Exit
1
12
Do you wish to continue? (Yes, No)
y
New Calculation
Enter the first number:
8
Enter the second number:
3
Do you want to:
1. Add
2. Multiply
3. Exit
2
24
Do you wish to continue? (Yes, No)
```