Vidya Vikas Education Trust's
Universal College of Engineering, Kaman Road, Vasai – 401208
Accredited A Grade by NAAC

Experiment 2

Name of the Student: - Rutuja Rajesh Kini
Roll No:- 53
Date of Practical Performed: -28/1/25 Staff Signature with Date & Marks

Aim: Write a program to perform Client server using RPC/RMI.

Theory:

RPC (Remote Procedure Call) and **RMI (Remote Method Invocation)** are two popular communication methods used in distributed systems, where a **client** can request services or methods from a **server** as if they were local procedures or methods, even though they may be running on different machines in the network.

1. **Remote Procedure Call (RPC) -** It is a protocol that allows a client to execute a procedure (or function) on a remote server. The client sends a request to the server, which processes it and returns a response. This "remote" interaction is abstracted so the client doesn't have to worry about the details of communication over a network.

How it works:

- **Client Side**: The client calls a local proxy function that looks like the server-side function. The proxy handles the network communication and sends the request to the remote server.
- **Server Side**: The server listens for requests, executes the requested procedure, and sends back the result.
- **Stubs and Skeletons**: These are intermediary components that facilitate communication between client and server:
  - **Client Stub**: It acts as a local proxy for the server function. When the client calls the remote function, the stub handles marshalling (packing) the arguments, sending them over the network, and returning the response.
  - **Server Skeleton**: It receives the call from the client, unpacks the arguments, calls the actual server function, and sends the result back to the client.

Example Use Case:

- A **client application** might request data or processing from a **remote server**, like fetching user details or performing a computation.

- **Java RMI** is an example of an RPC-based system.
- When working with **heterogeneous systems** or different programming languages (e.g., Python to Java communication).
- When you need simple **procedure-based** communication.

Code:

Server-

```
from xmlrpc.server import SimpleXMLRPCServer

# Function that the client will call remotely
def add(x, y):
    return x + y

# Create an XML-RPC server
server = SimpleXMLRPCServer(('localhost', 8001))

# Register the function so the server can process it
server.register_function(add, 'add')

print("Server is running...")
server.serve_forever()
```

client-

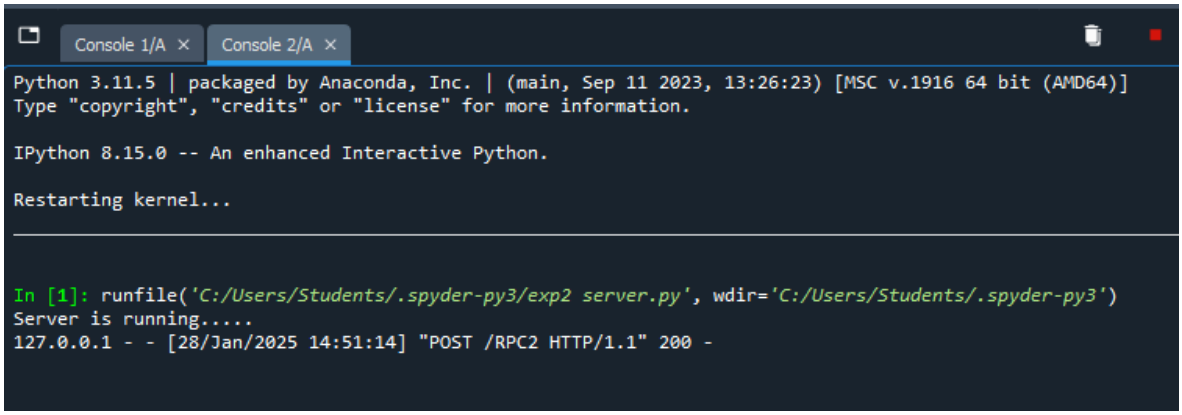
```
#client code
import xmlrpc.client

# Create a connection to the server
server = xmlrpc.client.ServerProxy('http://localhost:8001')

# Call the remote function to add two numbers
result = server.add(6, 5)

print(f"The result of adding two no's: {result}")
```

Output:



Conclusion: Therefore, we have learned and understood communication between client-servers using RPC(Remote Procedure Call).