

TCP/IP Implementation using Python Socket Programming

LAB # 11



Spring 2025

Submitted by: **Mohsin Sajjad**
Registration No: **22pwsce2149**

Class Section: A

“On my honor, as student of University of Engineering and Technology, I have neither given nor received unauthorized assistance on this academic work.”

A handwritten signature in black ink that reads "Mohsin Sajjad".

Student Signature: _____

Submitted to:

Dr. Yasir Saleem Afridi

Month Day, Year (30 05, 2025)

Department of Computer Systems Engineering
University of Engineering and Technology, Peshawar

CSE 303L: Data Communication and Computer Networks

Credit Hours: 1

Demonstration of Concepts	Poor (Does not meet expectation (1)) The student failed to demonstrate a clear understanding of the assignment concepts	Fair (Meet Expectation (2-3)) The student demonstrated a clear understanding of some of the assignment concepts	Good (Exceeds Expectation (4-5)) The student demonstrated a clear understanding of the assignment concepts	Score 30%
Accuracy	The student mis-configured enough network settings that the lab computer couldn't function properly on the network	The student configured enough network settings that the lab computer partially functioned on the network	The student configured the network settings that the lab computer fully functioned on the network	30%
Following Directions	The student clearly failed to follow the verbal and written instructions to successfully complete the lab	The student failed to follow the some of the verbal and written instructions to successfully complete all requirements of the lab	The student followed the verbal and written instructions to successfully complete requirements of the lab	20%
Time Utilization	The student failed to complete even part of the lab in the allotted amount of time	The student failed to complete the entire lab in the allotted amount of time	The student completed the lab in its entirety in the al	20%

TCP/IP Implementation using Python Socket Programming

Objectives of Lab

- To understand the implementation of TCP/IP using Python socket programming.
- To establish a basic client-server communication system.
- To explore the functionality of Python's socket library.

Introduction to Python Programming

Python is a high-level, interpreted programming language known for its simplicity and readability. It supports multiple programming paradigms, including procedural, object oriented, and functional programming. Python is widely used in web development, data analysis, artificial intelligence, scientific computing, and network programming. Due to its vast standard library and community support, Python is ideal for rapid application development.

Introduction to Python Socket Library and Its Various Functions

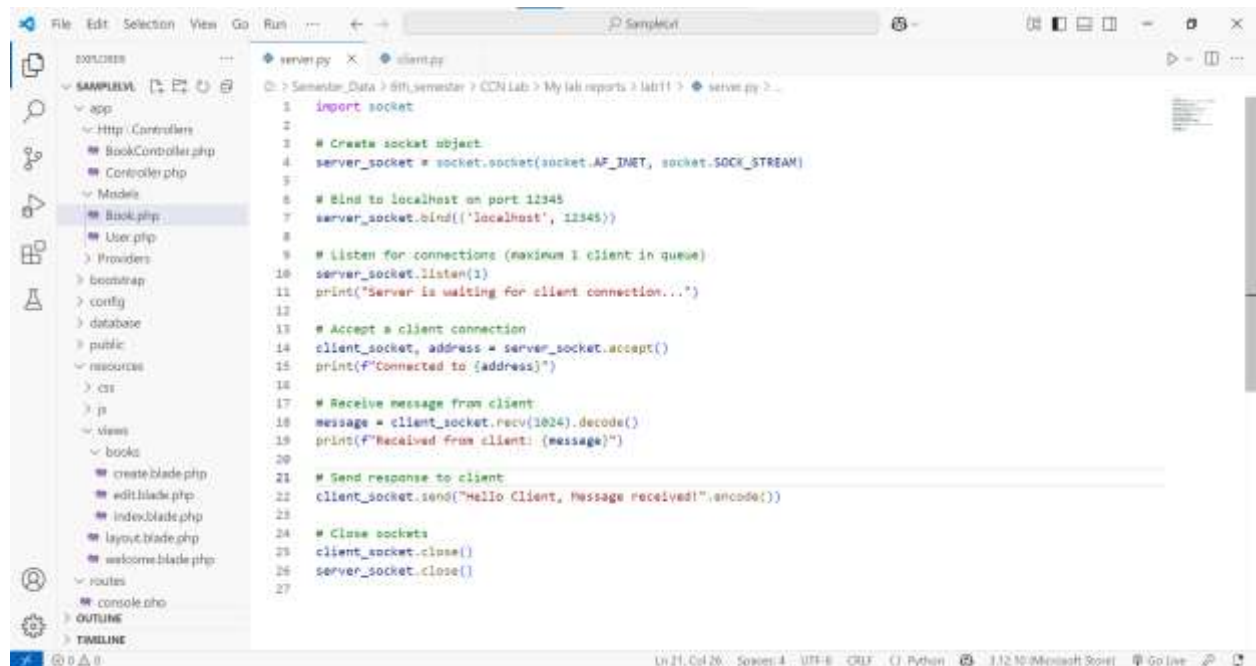
Python's socket module provides a standard way of networking in Python and is used for implementing clients and servers. It supports both TCP and UDP protocols.

Key functions used in this lab:

- `socket.socket()` –
 - Creates a new socket object.
- `bind()` – Associates the socket with a specific IP and port.
- `listen()` – Enables a server to accept connections.
- `accept()` – Accepts a connection request from a client.
- `connect()` – Connects a client to a server.
- `send()` / `sendall()`
 - Sends data from the client to the server.
- `recv()` – Receives data from the connection.
- `close()` – Closes the socket.

CODE:

Server:



The screenshot shows a Visual Studio Code editor with a file explorer on the left and a code editor on the right. The file explorer shows a project named 'SAMPLELVL' with a directory structure including 'app', 'controllers', 'models', 'providers', 'bootstrap', 'config', 'database', 'public', 'resources', 'css', 'js', 'views', and 'books'. The code editor displays the 'server.py' file, which is a Python script for a server. The script imports the 'socket' module, creates a 'server_socket' object, binds it to 'localhost' on port 12345, listens for connections, accepts a client connection, receives a message from the client, sends a response, and closes the sockets.

```
1 import socket
2
3 # Create socket object
4 server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
5
6 # Bind to localhost on port 12345
7 server_socket.bind(('localhost', 12345))
8
9 # Listen for connections (maximum 1 client in queue)
10 server_socket.listen(1)
11 print("Server is waiting for client connection...")
12
13 # Accept a client connection
14 client_socket, address = server_socket.accept()
15 print(f"Connected to {address}")
16
17 # Receive message from client
18 message = client_socket.recv(1024).decode()
19 print(f"Received from client: {message}")
20
21 # Send response to client
22 client_socket.send("Hello Client, Message received!".encode())
23
24 # Close sockets
25 client_socket.close()
26 server_socket.close()
27
```

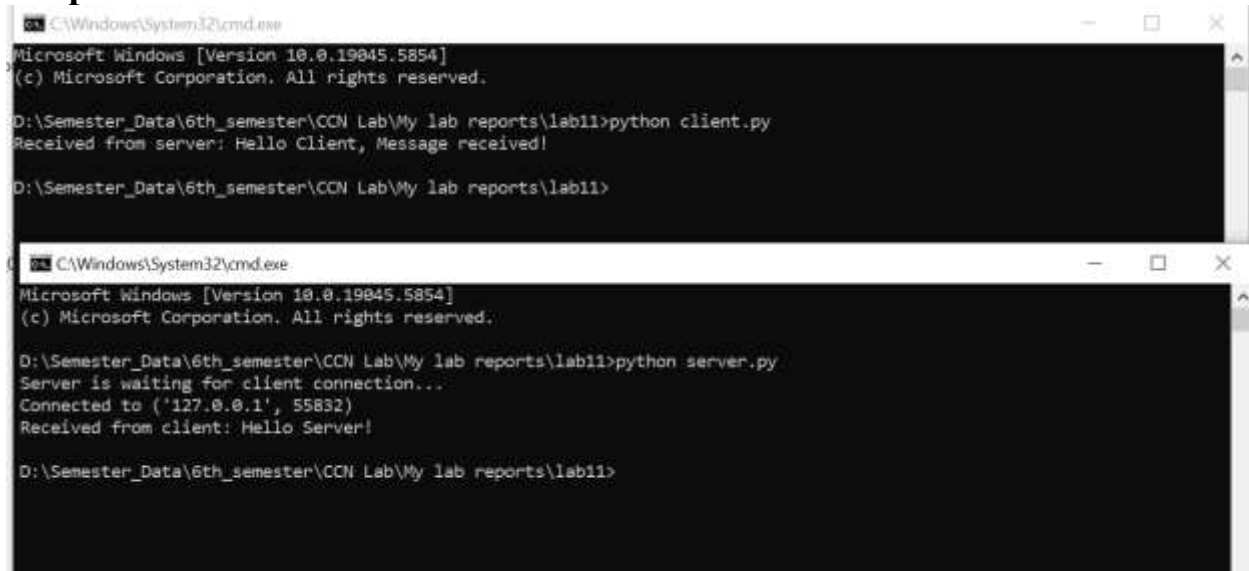
CLIENT:



The screenshot shows a Visual Studio Code editor with a file explorer on the left and a code editor on the right. The file explorer shows the same project structure as the previous screenshot. The code editor displays the 'client.py' file, which is a Python script for a client. The script imports the 'socket' module, creates a 'client_socket' object, connects it to 'localhost' on port 12345, sends a message to the server, receives a response, and closes the socket.

```
1 import socket
2
3 # Create socket object
4 client_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
5
6 # Connect to server at localhost on port 12345
7 client_socket.connect(('localhost', 12345))
8
9 # Send message to server
10 client_socket.send("Hello Server!".encode())
11
12 # Receive response from server
13 response = client_socket.recv(1024).decode()
14 print(f"Received from server: {response}")
15
16 # Close socket
17 client_socket.close()
18
```

Output:



The image displays two overlapping Windows command prompt windows. The top window shows the execution of a client script, and the bottom window shows the execution of a server script. Both windows are titled 'C:\Windows\System32\cmd.exe' and show the standard Windows version and copyright information.

```
C:\Windows\System32\cmd.exe
Microsoft Windows [Version 10.0.19045.5854]
(c) Microsoft Corporation. All rights reserved.

D:\Semester_Data\6th_semester\CCN Lab\My lab reports\lab11>python client.py
Received from server: Hello Client, Message received!

D:\Semester_Data\6th_semester\CCN Lab\My lab reports\lab11>
```

```
C:\Windows\System32\cmd.exe
Microsoft Windows [Version 10.0.19045.5854]
(c) Microsoft Corporation. All rights reserved.

D:\Semester_Data\6th_semester\CCN Lab\My lab reports\lab11>python server.py
Server is waiting for client connection...
Connected to ('127.0.0.1', 55832)
Received from client: Hello Server!

D:\Semester_Data\6th_semester\CCN Lab\My lab reports\lab11>
```

Conclusion:

This lab successfully demonstrated the fundamentals of TCP/IP communication using Python's socket programming. We implemented a basic client-server model where a client sends a message to a server and receives a confirmation in response. This lays the foundation for understanding more advanced network communication systems.