# Project 1 Report

Mehroz Akhtar - 922734865
Abdul Nawab - 917412765

## Part 2 - Proxy server

### Files

proxy_server[mehrozakhtar]_[922734865]_[abdulnawab]_[917412765].py
client[mehrozakhtar]_[922734865]_[abdulnawab]_[917412765].py
server[mehrozakhtar]_[922734865]_[abdulnawab]_[917412765].py

### How to Run Code

Open three terminals, and do the following in order:

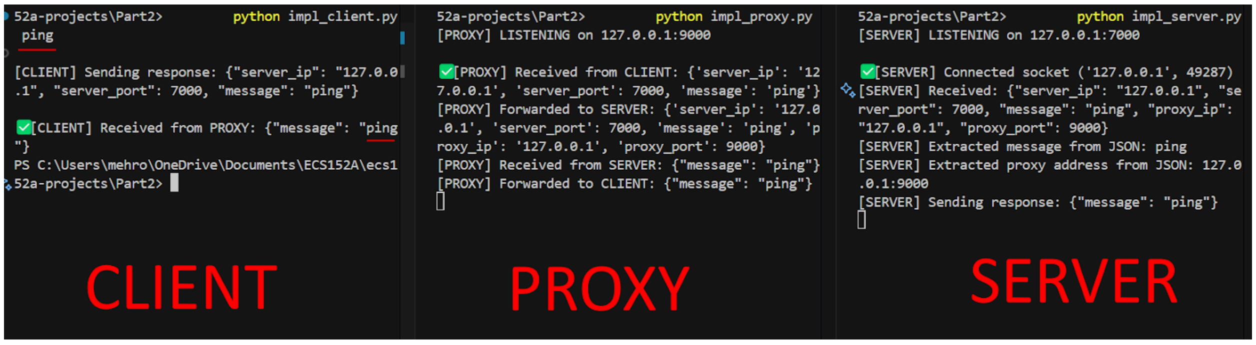
1) Make sure each terminal's directory is in **Part2**

2) Run the **server** in a terminal:
   **- python "**server[mehrozakhtar]_[922734865]_[abdulnawab]_[917412765].py"

3) Run the **proxy** in another terminal:
   **- python "**proxy_server[mehrozakhtar]_[922734865]_[abdulnawab]_[917412765].py"

4) Run the **client** with a 4 character message in another terminal
   **- python "**client[mehrozakhtar]_[922734865]_[abdulnawab]_[917412765].py" **<4 char msg>**

Example:
Running: **python "**client[mehrozakhtar]_[922734865]_[abdulnawab]_[917412765].py" **ping**



```
52a-projects\Part2>         python impl_client.py
 ping

[CLIENT] Sending response: {"server_ip": "127.0.0
.1", "server_port": 7000, "message": "ping"}

✅[CLIENT] Received from PROXY: {"message": "ping
"}
PS C:\Users\mehro\OneDrive\Documents\ECS152A\ecs1
52a-projects\Part2>
```

```
52a-projects\Part2>         python impl_proxy.py
[PROXY] LISTENING on 127.0.0.1:9000

✅[PROXY] Received from CLIENT: {'server_ip': '12
7.0.0.1', 'server_port': 7000, 'message': 'ping'}
[PROXY] Forwarded to SERVER: {'server_ip': '127.0
.0.1', 'server_port': 7000, 'message': 'ping', 'p
roxy_ip': '127.0.0.1', 'proxy_port': 9000}
[PROXY] Received from SERVER: {"message": "ping"}
[PROXY] Forwarded to CLIENT: {"message": "ping"}
```

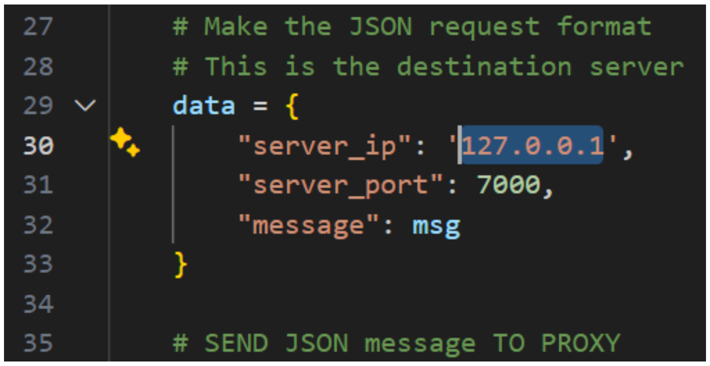
```
52a-projects\Part2>         python impl_server.py
[SERVER] LISTENING on 127.0.0.1:7000

✅[SERVER] Connected socket ('127.0.0.1', 49287)
[SERVER] Received: {"server_ip": "127.0.0.1", "se
rver_port": 7000, "message": "ping", "proxy_ip":
"127.0.0.1", "proxy_port": 9000}
[SERVER] Extracted message from JSON: ping
[SERVER] Extracted proxy address from JSON: 127.0
.0.1:9000
[SERVER] Sending response: {"message": "ping"}
```

### Test Blocked IP's

- Our blocked IPs are **127.0.0.4** and **127.0.1.4**
- Our working default IP: **127.0.0.1**

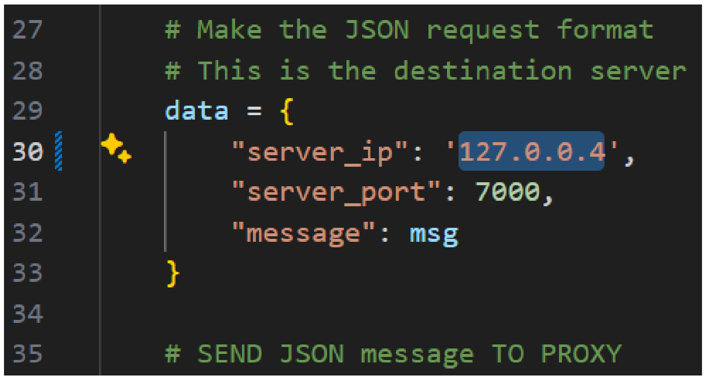
To test the Blocked IPs, change **line 30** in **client[..].py** to one of the blocked IP’s (127.0.0.4 or 127.0.1.4).

**Valid IP example (default):**



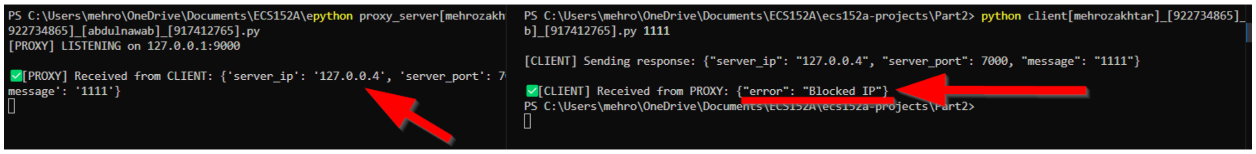
```
27      # Make the JSON request format
28      # This is the destination server
29      data = {
30          "server_ip": '127.0.0.1',
31          "server_port": 7000,
32          "message": msg
33      }
34
35      # SEND JSON message TO PROXY
```

**Blocked IP example (127.0.0.4):**



```
27      # Make the JSON request format
28      # This is the destination server
29      data = {
30          "server_ip": '127.0.0.4',
31          "server_port": 7000,
32          "message": msg
33      }
34
35      # SEND JSON message TO PROXY
```

Output:



```
PS C:\Users\mehro\OneDrive\Documents\ECS152A\epython proxy_server[mehrozakh
922734865]_[abdulnawab]_[917412765].py
[PROXY] LISTENING on 127.0.0.1:9000

✅[PROXY] Received from CLIENT: {'server_ip': '127.0.0.4', 'server_port': 7
message': '1111'}
```

```
PS C:\Users\mehro\OneDrive\Documents\ECS152A\ecs152a-projects\Part2> python client[mehrozakhtar]_[922734865]_
b]_[917412765].py 1111

[CLIENT] Sending response: {"server_ip": "127.0.0.4", "server_port": 7000, "message": "1111"}

✅[CLIENT] Received from PROXY: {"error": "Blocked IP"}
PS C:\Users\mehro\OneDrive\Documents\ECS152A\ecs152a-projects\Part2>
```

## Initial Attempts

Initially, we used the code from Part 1 as a guide to create the proxy and client scripts. The initialization of the variables like the IP and port were similar to Part 1. Likewise, error checking for running the client script was done in a similar manner, where the number of argument parameters and message length (4 characters) were verified prior to creating the socket.

We first made sure that the client and server could connect to each other properly through TCP and bounce messages back and forth. Once this was successful, the proxy was then created to act as an intermediary between the client and the server. Instead of going directly from client to server, the messages would be forwarded to the proxy instead. We used this guide to create the proxy, and this worked well. However, unnecessary code like connecting to a domain was removed, and replaced with the client and server address instead.

## Description of Expected Results

First, the user enters a 4 letter message which they would like to send to the server. This is done through the **client** program. Then, the client program connects to the **proxy** server with a JSON formatted message, which has the server IP, port, and the user's message. The client prints the JSON and sends it to the proxy.

After, the **proxy** receives the request from the client. The proxy first prints the request it got. Then, it checks if the server IP is blocked, and if it is, then it will send back an error message to the client and not forward the request to the server. Otherwise, the proxy will add its IP & port to the JSON data, print the JSON, and forward the JSON request to the **server**.

The **server** gets the JSON request and prints it. Then, it extracts the message and proxy address from the JSON and prints it. After, it sends back the message to the **proxy** server and prints the sent data.

The **proxy** receives the JSON from the server and prints it. Then, it sends the JSON data message back to the client.

The **client** receives the JSON data from the proxy and finally prints the message.

## Issues Faced

Some issues faced was finding out how the JSON data could be transmitted from the client to the proxy/server, and vice versa. This part was confusing to us because it was our first time working with JSON in python. To solve this problem, we found a JSON library in python that was able to send user data from client to proxy to server, and vice versa. We looked at the python json documentation which went over a json.dumps(*data*) method. This was used to convert the initial dict *data* into a JSON format which could then be sent over across the network. Another problem was adding the proxy IP into the json data. The server had to somehow extract the proxy address. To do this, we simply added a new field into the JSON request in the proxy program, which allowed the proxy address to be extracted from the data in the server side.