Network utility GUI

Project description: This project is a graphical user interface (GUI) application built using the Tkinter library in Python. The GUI provides a user-friendly interface for executing various network-related commands and actions. It includes functionalities for DNS commands, network tracing, protocol testing (HTTP and FTP), and TCP/UDP server-client interactions.

Software and technologies used to write and run the code:

1. Windows 10
2. Visual Studio code IDE
3. Ubuntu wsl version 20.04
4. Xming -> x11 display server

How to run "network_utility.py" code?

1. You need the latest version of linux based Operating system, e.g **Ubuntu**
2. Install a x11 display server https://sourceforge.net/projects/xming/
3. On your linux based OS:
    a) sudo apt install python3.8-tk
    b) sudo apt-get install python3-requests
    c) nano ~/.zshrc
    d) **Paste** **"**export DISPLAY=$(awk '/nameserver / {print $2; exit}' /etc/resolv.conf 2>/dev/null):0 export LIBGL_ALWAYS_INDIRECT=1**"**
    e) Save and exit the file

4. Make sure that all VcXsrv and Xming X Server are green in Inbound Rules by entering **wf.msc** to your Windows search bar

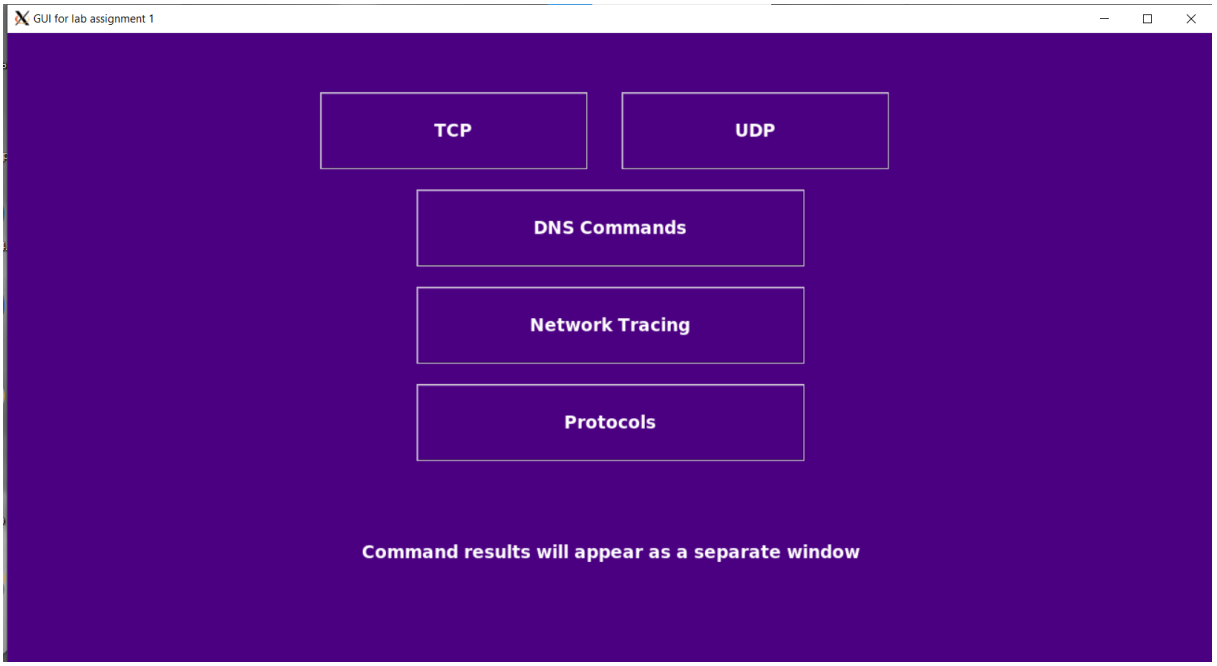
5. Run **Xming** server, check "Disable access control". More on how to install and run this server: https://www.cs.mtsu.edu/~rwsmith/share/VcXsrv/VcXsrv.html

6. On your linux based OS:
    a) source ~/.zshrc
    b) python3 network_utility.py

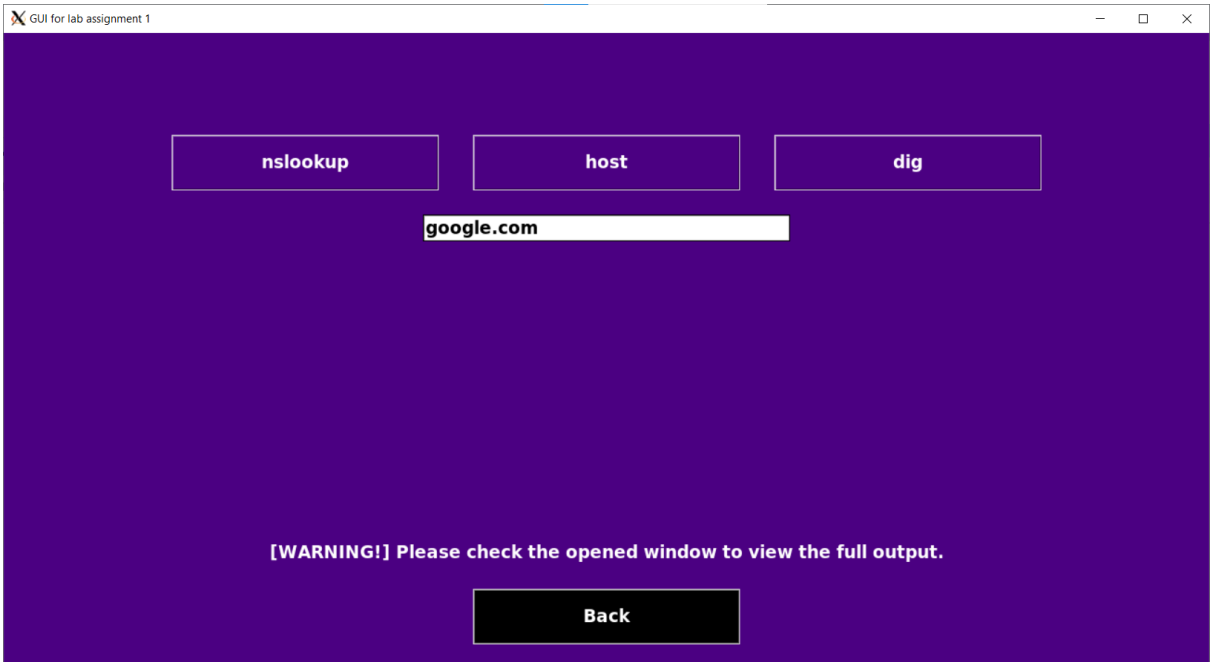
*Repeat steps 6a and 6b when running this file next times

GUI usage demonstration:

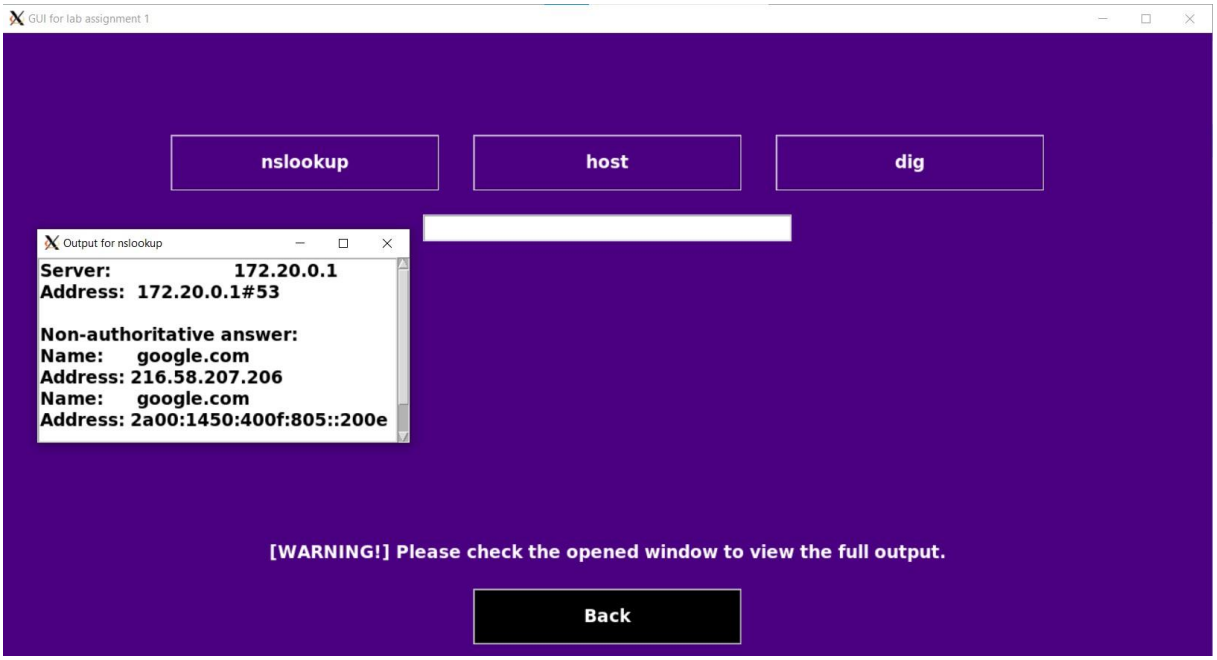
1. **Main page**



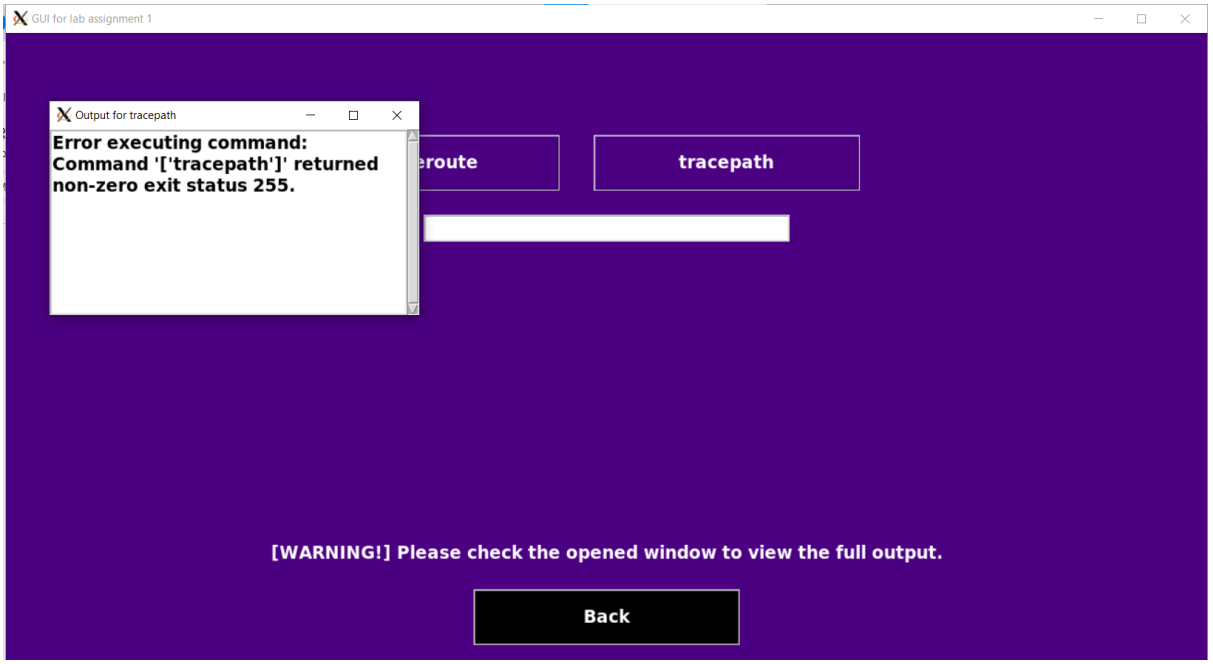
2. **Pressing on nslookup button after entering the domain name**



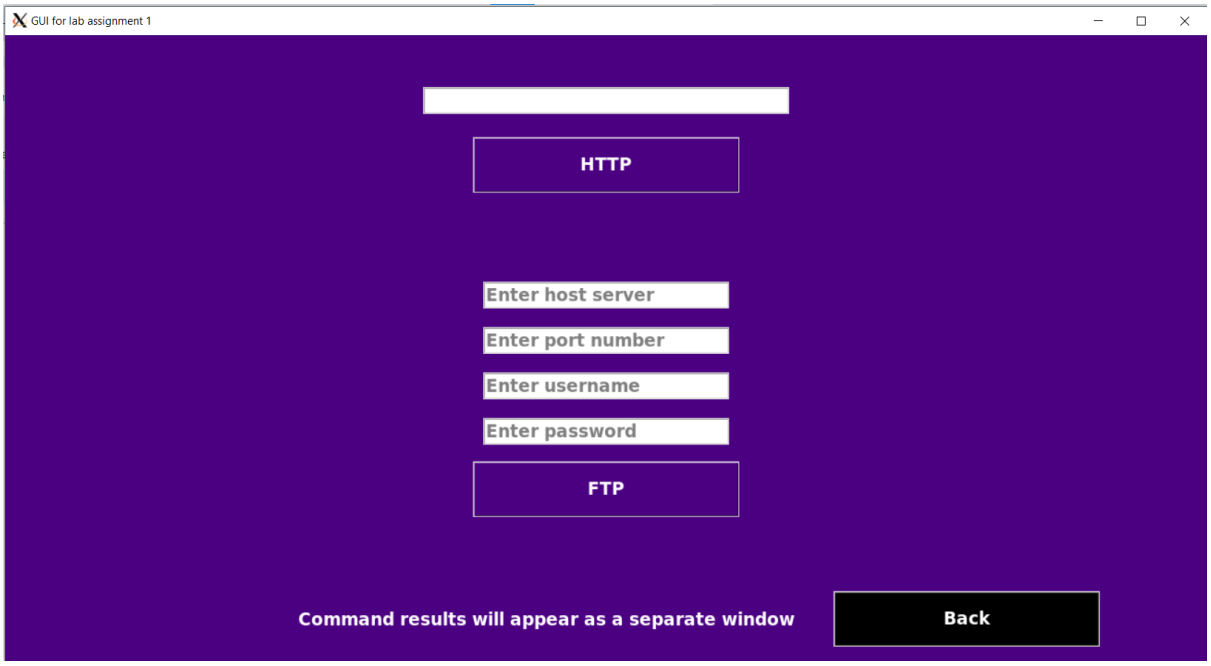
3. **Output: A separate window appears with the output**



4. **Error handling: sending empty domain name to the tracepath**



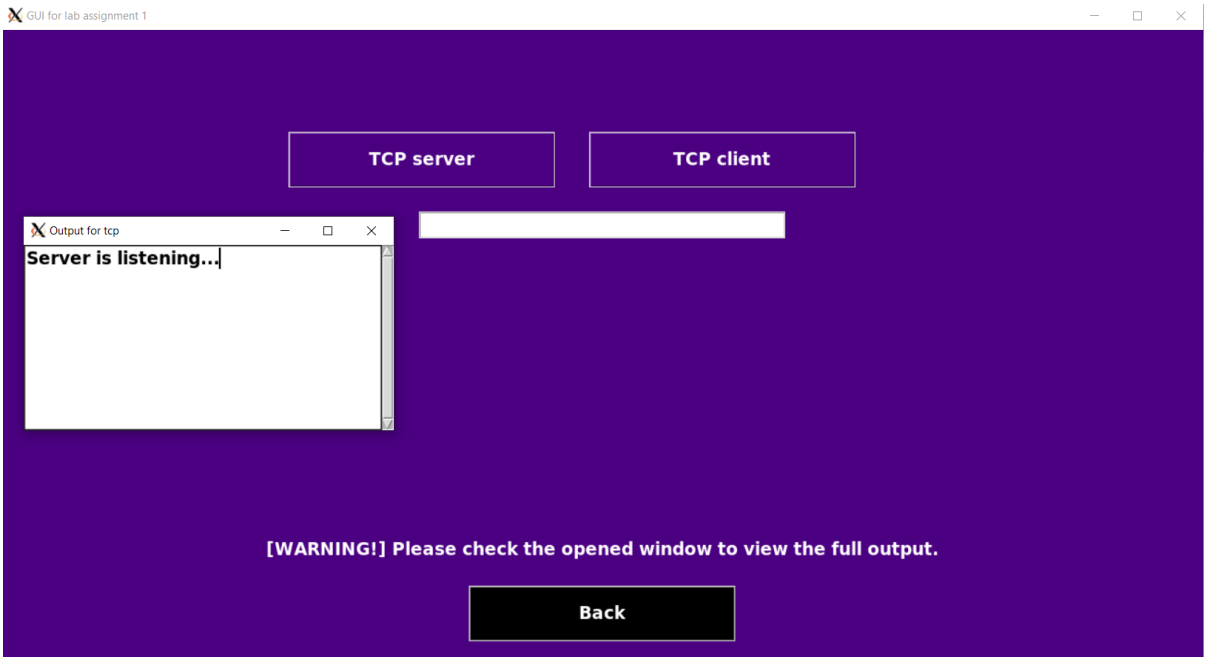
5. **Network tracing page: First entry widget is for pasting URL and sending HTTP request. Other entry widgets display hints on the expected input from the user.**



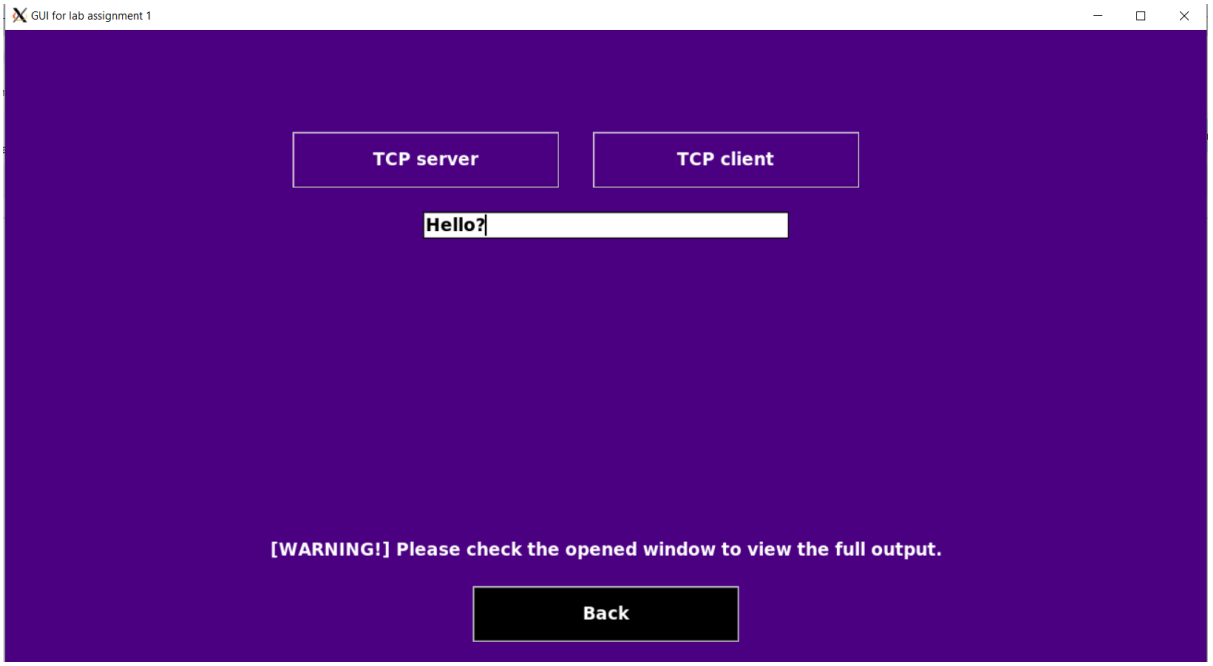
6. **Output example for HTTP, if opening the output window:**

How to use TCP and UDP connection?

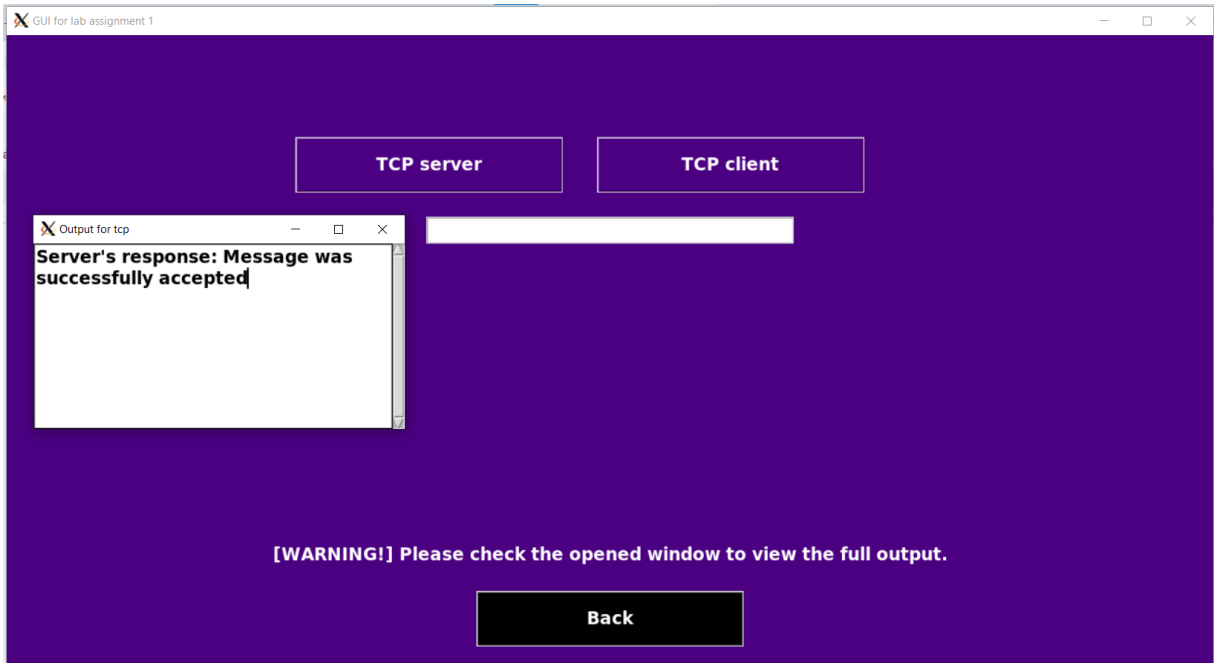
1. **Press on TCP server or UDP server to start the server**



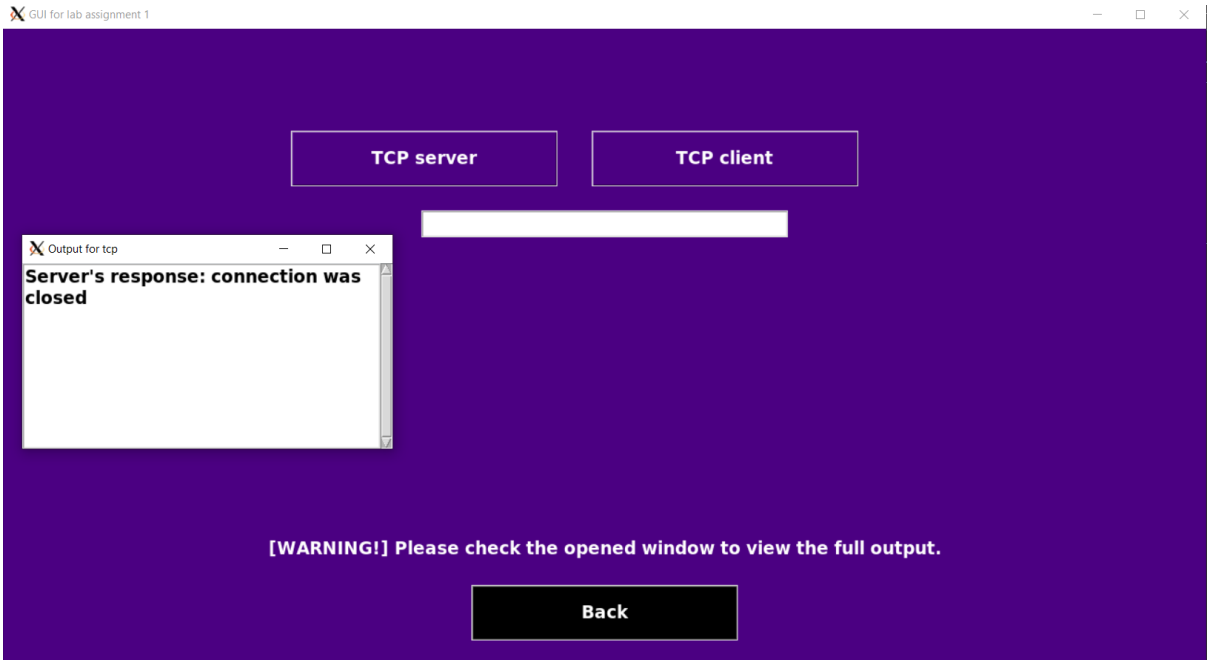
2. **Enter your message**



3. **Press on TCP client or TCP client button**



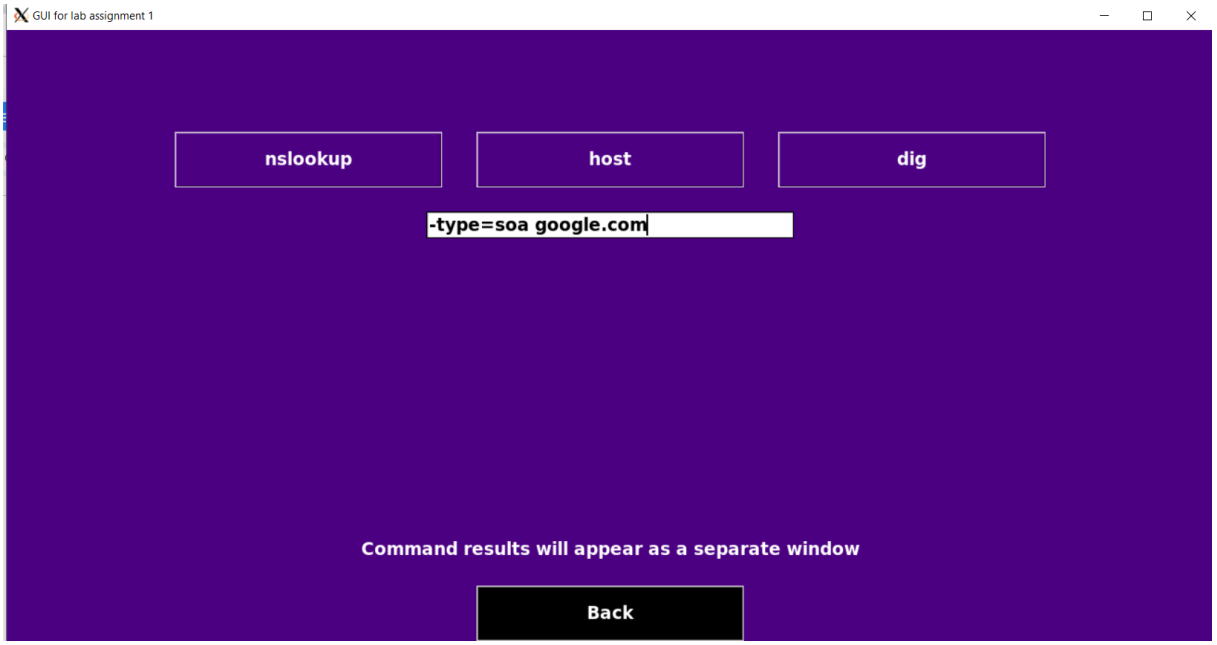
4. **Enter “finish” and press on TCP or UDP client to close the connection**



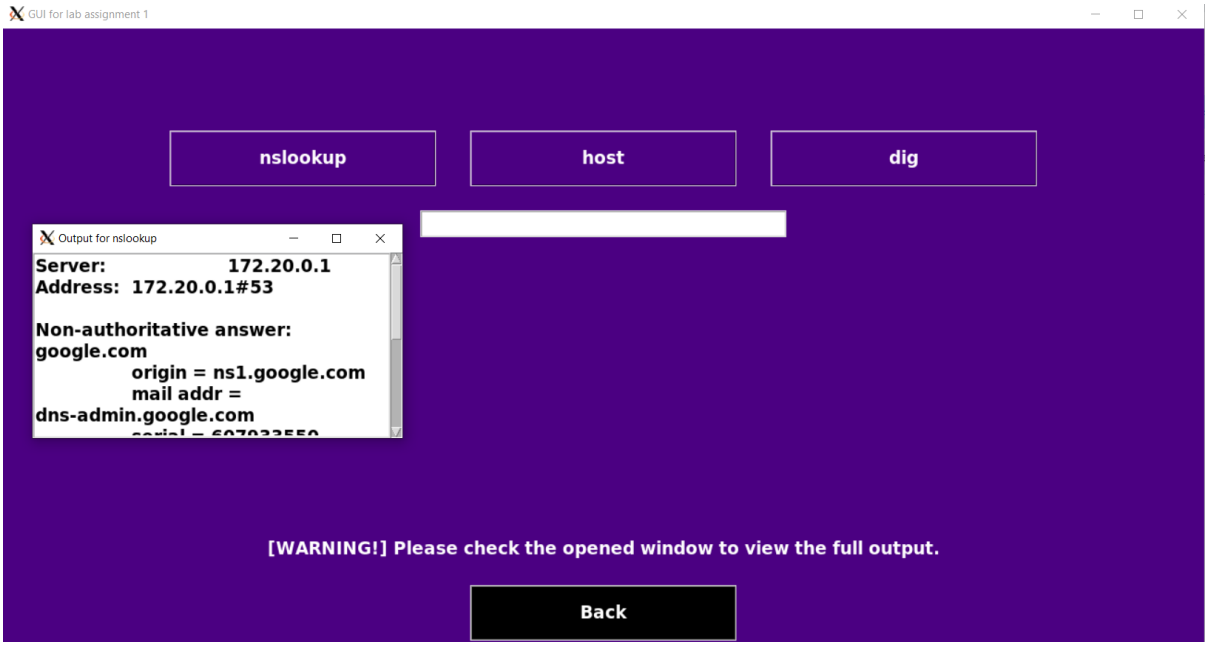
Important notes:

1. All errors are gracefully handled by printing them out on a separate window, which pops out automatically
2. The GUI is capable of executing the DNS commands and Network trace commands with any arguments. It is important to have a space between the argument and the domain name, and no additional spaces within the argument. For example:
    a) Valid: -type=soa google.com
    b) Invalid: - type = soa google.com
    c) Invalid: -type=soagoogle.com

Example:

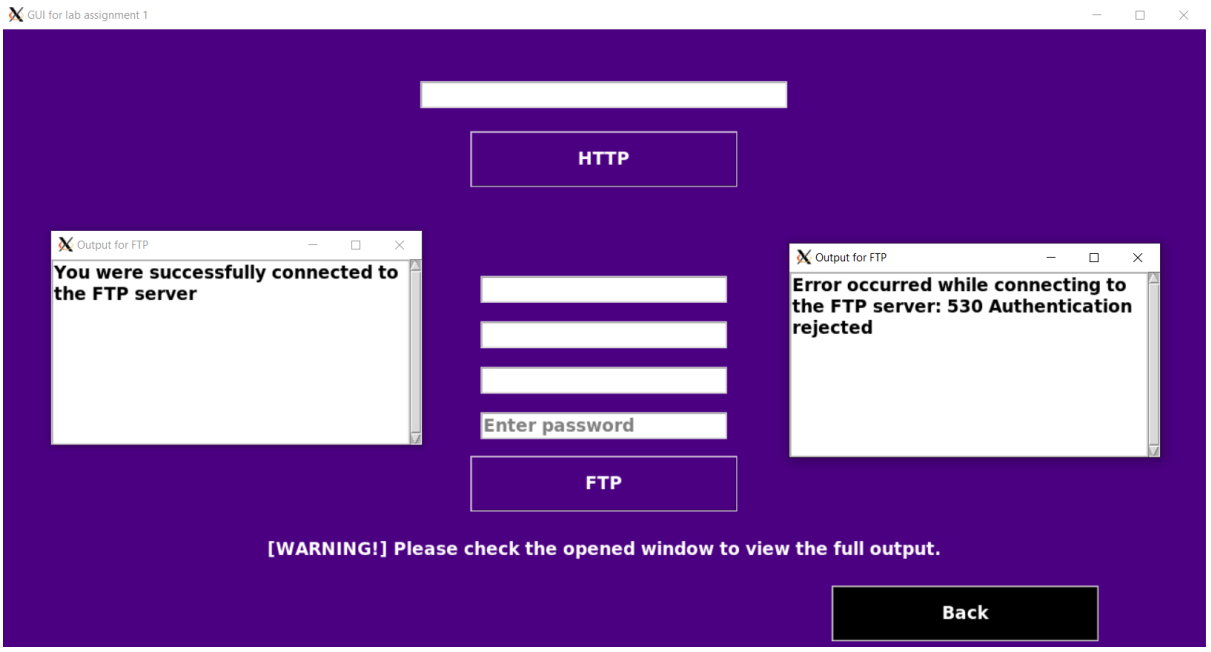


Output:



3. Network trace commands (**traceroute, tracepath)** take some time to execute
4. For testing FTP https://test.rebex.net/ was used.
   Server: test.rebex.net
   Port: 21
   Username: demo
   Password: password

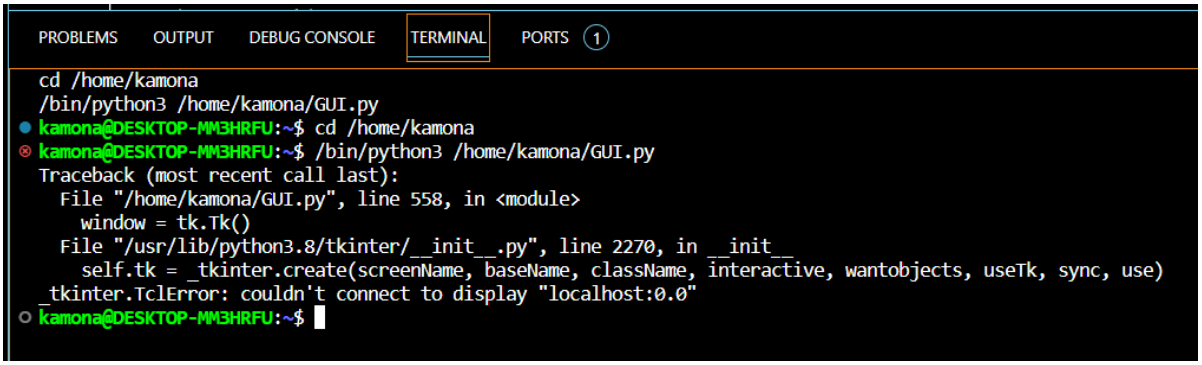
Two possible outputs for FTP result:



5. For TCP and UDP, press the SERVER button first to start the server, then enter your message and press the CLIENT button.
    a) To close the connection, send “finish” message

Encountered difficulties:

1. I was struggling a lot with running this code from my linux based operating system. I am still not sure what was the problem, but I kept getting the following error:



I have tried all possible solutions from the internet, and eventually the following post helped me to solve this error: https://stackoverflow.com/questions/70170987/wsl2-error-tkinter-tclerror-couldnt-connect-to-display-127-0-0-10-0

The content of this post became a guide on how to run my networ_utility.py, which can be seen at the beginning of this pdf file.

2. Another difficulty that I encountered while completing this lab assignment was enabling the client to send an infinite number of messages to the server before sending a “finish” message. My code would only get one message from the client and output a “Bad pipe” and “Connection refused” errors. The problem was with the “client” button, as every time a user would press it to

send their message, it would restart the client connection. I solved this problem by configuring the client button with a function inside the client function after the first message is received from the user.