KALINGA INSTITUTE
OF INDUSTRIAL TECHNOLOGY
Deemed to be University U/S 3 of UGC Act, 1956

Computer Networks

[IT-3009]

NETWORK ALCHEMIST

[A network analyser and a troubleshooter]

Prepared by:

Muhit Khan	**21052943**
Anushka Pandey	**21051287**
Sangbartika Saha	**21051334**

Table of Contents

1. Introduction

1.1 Purpose

This document has been meticulously crafted to delineate the comprehensive specifications for a sophisticated desktop application poised to optimize the intricate process of network troubleshooting. The core essence of this application is to empower users with the capability to effortlessly input and execute Windows Command Line Interface (CLI) commands while presenting the outcomes in an intuitively designed Graphical User Interface (GUI). This seamless fusion of technology leverages the skilfulness of renowned Python libraries to deliver a refined and user-centric experience.

1.2 Product Scope

The scope of this application is squarely aimed at the domain of network troubleshooting. The application will be used for network troubleshooting, processing Windows CLI commands in the background and displaying the output in a user-friendly GUI.It serves the distinct purpose of alleviating the challenges associated with network troubleshooting by rendering it more accessible and comprehensible to a broader user base.

1.3 References

1. Ideas on Troubleshoot Commands:

https://blog.invgate.com/network-troubleshooting-tools .

2. Tkinter Documentation:

Official Tkinter documentation: https://docs.python.org/3/library/tkinter.html

3.GUI Programming with Python and Tkinter:

"Python GUI Programming with Tkinter" by Alan D. Moore.

This book offers in-depth coverage of Tkinter and practical examples to help you get started with GUI development in Python.

2. General Description

2.1 Product Functions:

The application will support the following commands:

- ipconfig /all
- ping "URL"
- traceroute "URL" or, tracert "URL"
- nsslookup "URL"
- ipconfig /flushdns
- ipconfig /release
- netsh int ip reset
- netsh winsock reset

2.2 User Classes and Characteristics

The end-users of this specialized application are anticipated to possess a foundational understanding of network troubleshooting principles and a degree of familiarity with the Command Line Interface (CLI) commands germane to the realm of network diagnostics. The emphasis on user proficiency underscores the application's commitment to catering to a discerning audience who have an inherent grasp of network intricacies.

The underlying premise of this project is to provide a powerful yet user-friendly tool that augments the proficiency of network administrators and facilitates smoother troubleshooting.

2.3 Operating Environment

Python boasts a myriad of GUI frameworks, yet Tkinter stands alone as the sole framework seamlessly integrated into the Python standard library. Tkinter exhibits several commendable attributes. It offers cross-platform compatibility, enabling code to run flawlessly on Windows, macOS, and Linux systems. This is achieved by rendering visual elements using native operating system components, ensuring that applications developed with Tkinter seamlessly blend with the platform on which they are deployed.

Nevertheless, Tkinter is not immune to criticism. One noteworthy critique revolves around the perception that Tkinter-based graphical user interfaces (GUIs) may appear somewhat antiquated. For those seeking a polished, contemporary aesthetic, Tkinter may not be the ideal choice.

Despite this critique, Tkinter shines in terms of its lightweight nature and user-friendliness, particularly when juxtaposed with alternative frameworks. Consequently, Tkinter remains a compelling choice for Python developers seeking to build GUI applications. It is particularly advantageous for situations where modern aesthetics take a back seat to expeditious development and the imperative requirement of cross-platform compatibility.

3. System Features:

3.1 Functional Requirements

1. Command Input: The application is mandated to offer a well-defined input field that accommodates users in inputting their chosen Command Line Interface (CLI) commands. This input field should be intuitively designed and readily accessible to ensure user convenience and facilitate seamless interaction with the application.

2.Command Execution: An imperative functionality of the application is its ability to execute the CLI commands entered by the user. This execution process should operate discreetly in the background, preserving the Graphical User Interfaces (GUI) responsiveness. By effectively multitasking, the application enhances the user experience, making it both efficient and user-friendly.

3.Output Presentation: The application bears the onus of presenting the results of the executed commands in a comprehensible and visually appealing manner within the GUI. This presentation should prioritize legibility, adhering to modern design principles to ensure that users can readily interpret the information presented.

4.Error Handling: An application of this calibre must possess a robust error-handling mechanism. It is paramount that the application adeptly addresses any anomalies or errors that may arise during the execution of CLI commands.

3.2 Performance Requirements

The application should process commands and display output within a reasonable time frame, ensuring a smooth user experience.

i) Efficient Command Processing: The application is expected to process CLI commands efficiently, prioritizing speed and responsiveness.
ii) Timely Output Display: The application is committed to presenting the output of executed commands within an acceptable timeframe.

3.3 Design and Implementation Constraints

Design Constraints:

User Interface Consistency*:* While Tkinter's GUI elements are native to the platform, achieving a consistent and modern user interface may be a constraint. Careful design and customization may be required to overcome this limitation.

Security: The application may need to execute potentially sensitive CLI commands. Ensuring secure command execution and input validation is a design constraint.
Resource Utilization: Keeping resource utilization (CPU, memory) low is essential for a responsive application. Design should consider minimizing resource usage.

Implementation Constraints:

Python and Tkinter Compatibility: The use of Tkinter constrains your choice of programming language to Python. The application's design and implementation should adhere to Python best practices and the Tkinter framework.

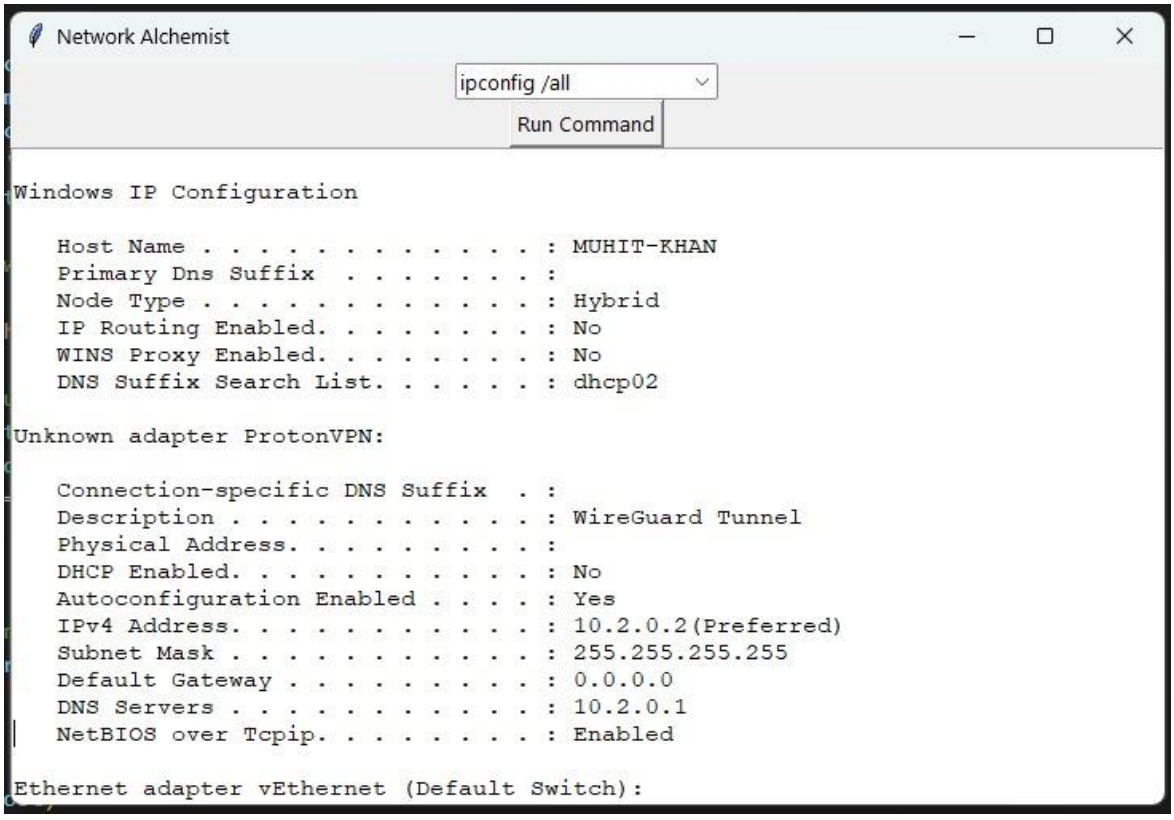
Network Environment: The application's functionality may be constrained by the specifics of the network environment in which it operates, such as firewalls, access rights, and network configurations.

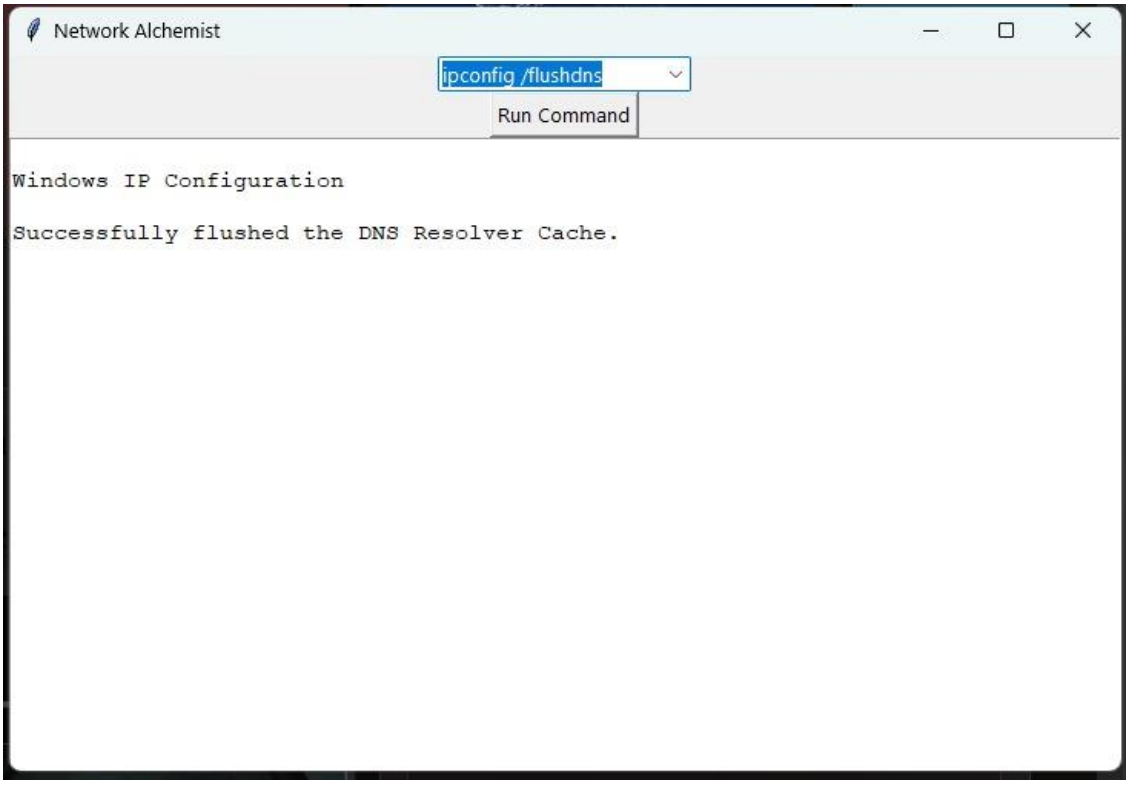
User Permissions: Proper design should consider user permission constraints, especially if the application requires administrative privileges to execute certain commands.

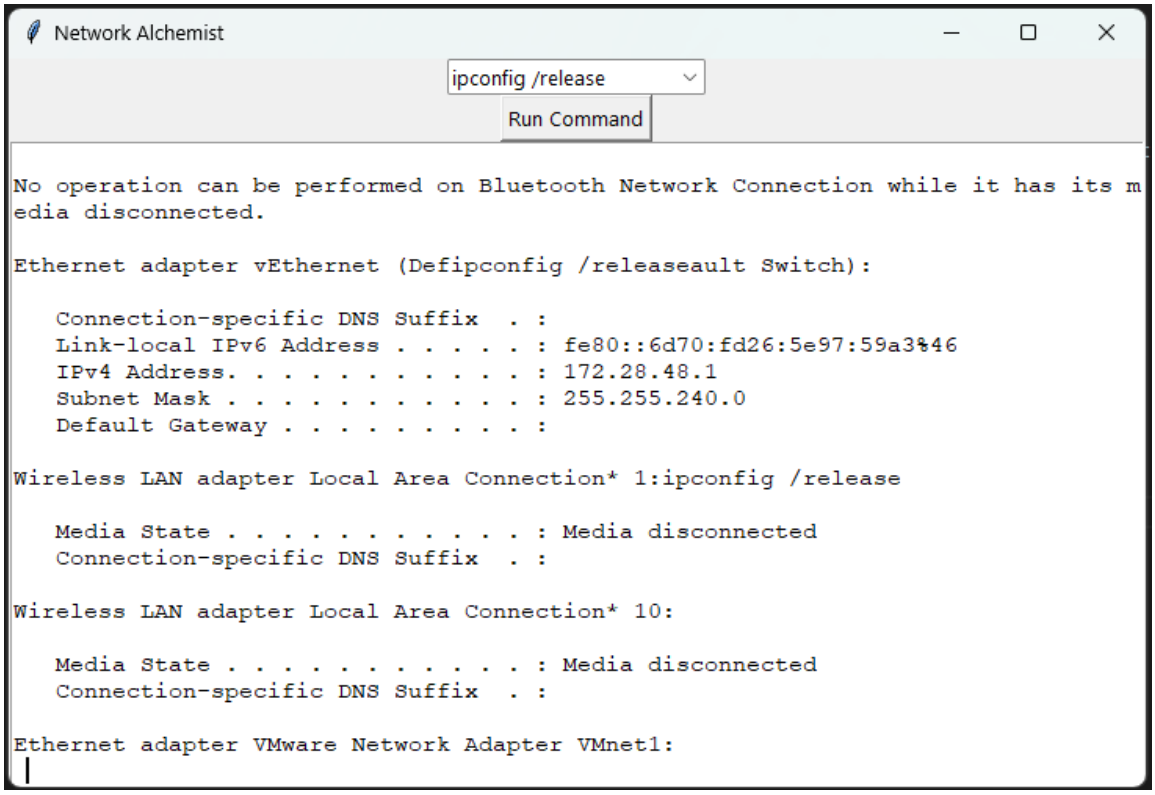
User Feedback: Design constraints may relate to the collection and handling of user feedback for improvements and issue resolution.

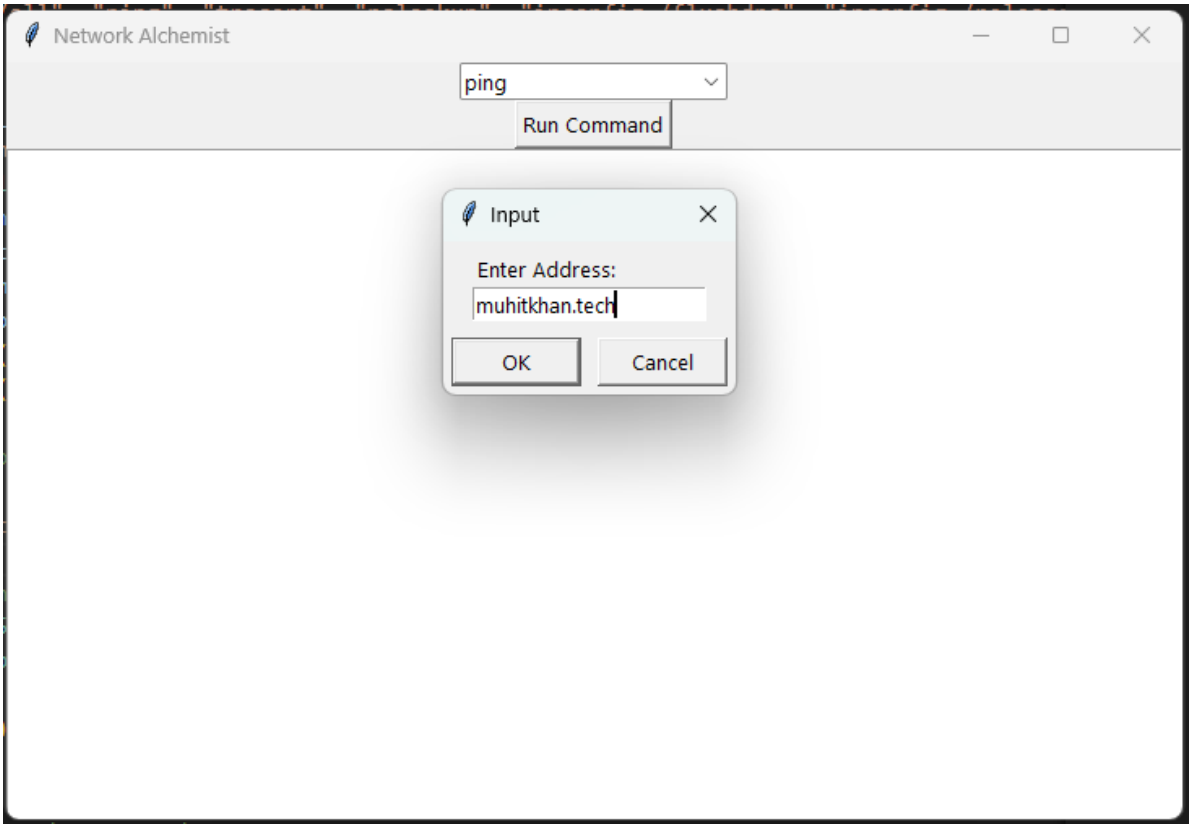
4. External Interface Requirements

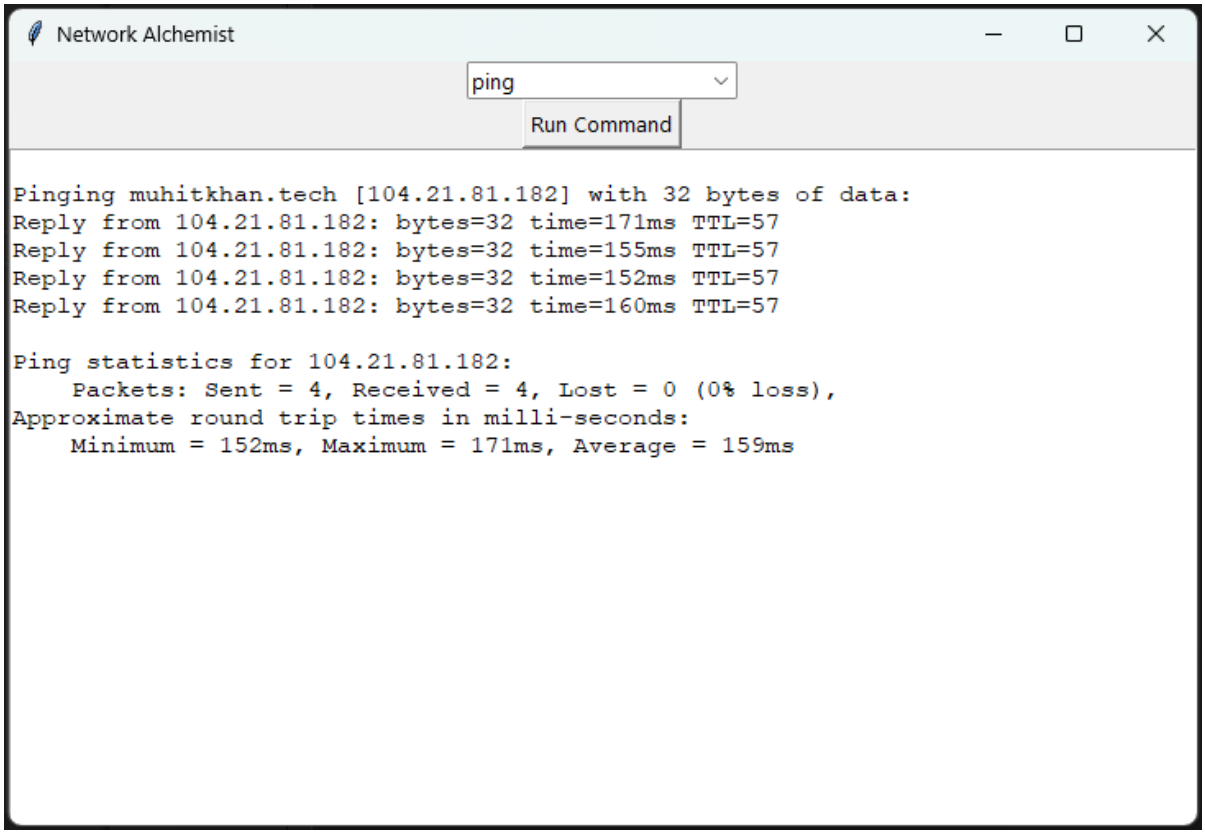
4.1 User Interfaces

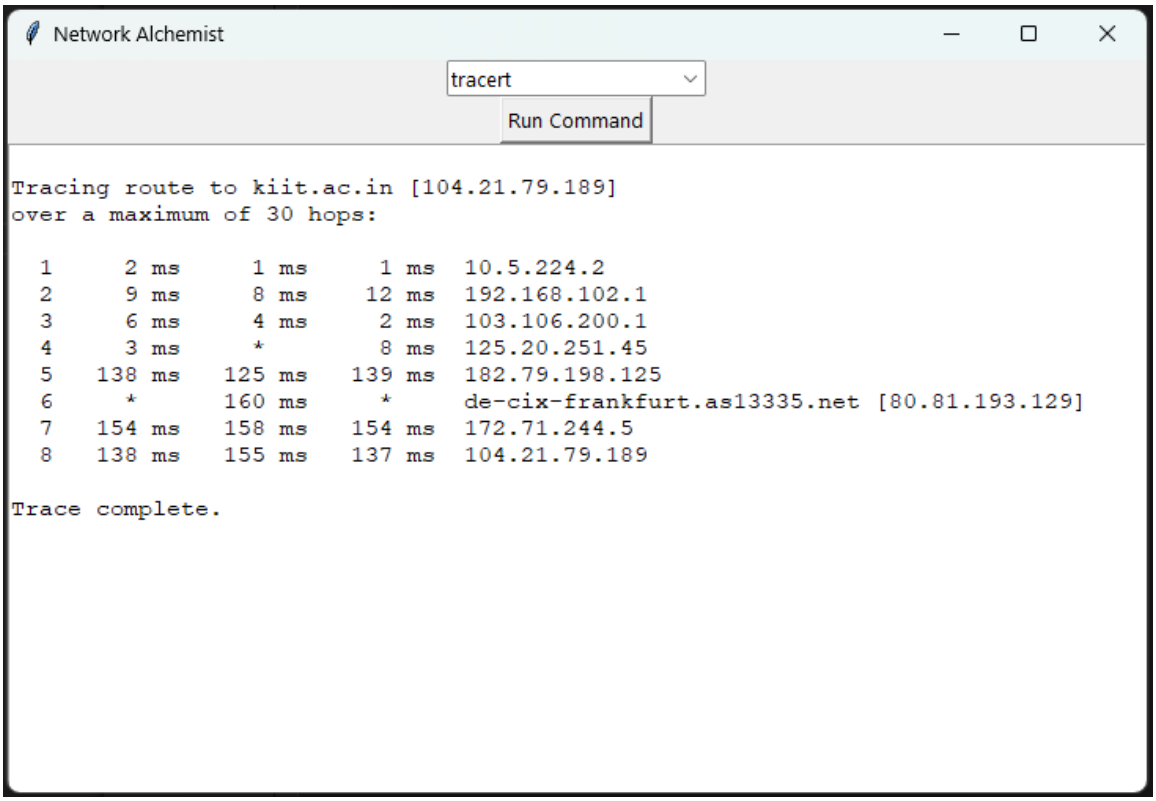


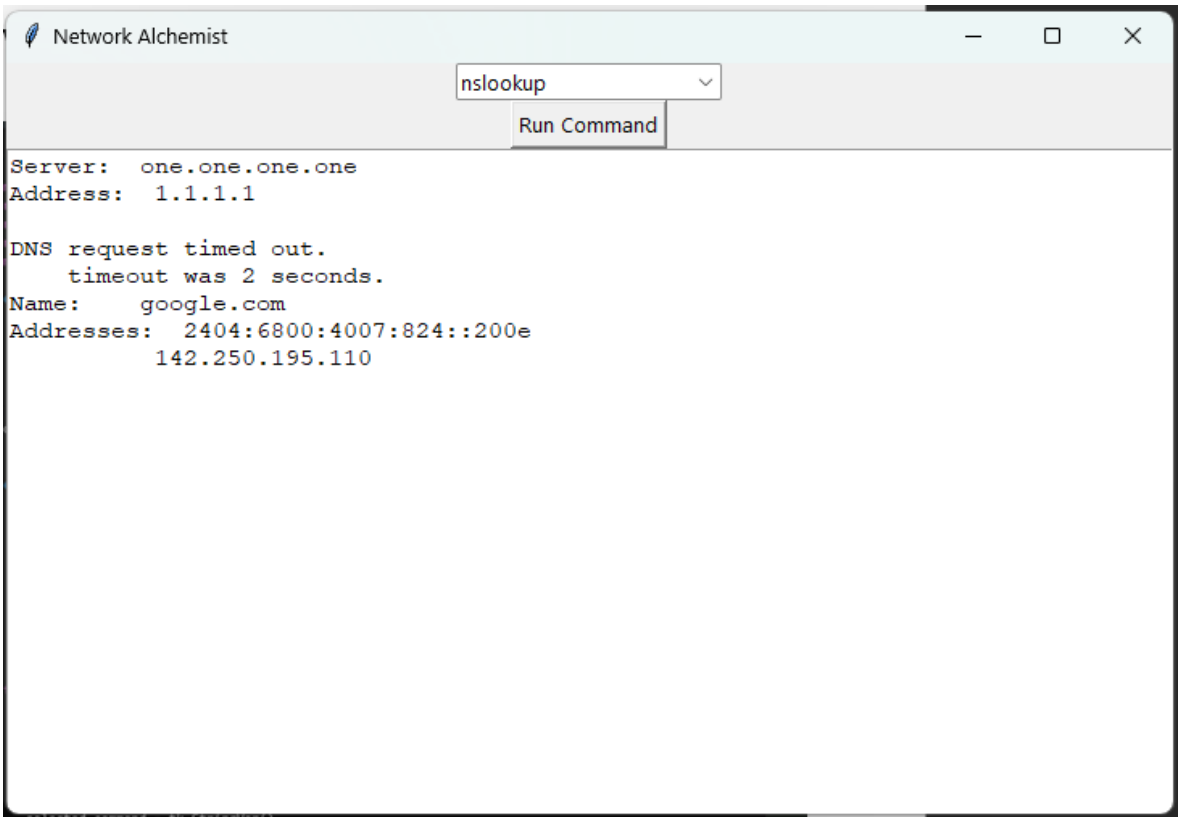

Network Alchemist
ipconfig /flushdns
Run Command
Windows IP Configuration
Successfully flushed the DNS Resolver Cache.


Network Alchemist
ipconfig /release
Run Command
No operation can be performed on Bluetooth Network Connection while it has its m
edia disconnected.
Ethernet adapter vEthernet (Defipconfig /releaseault Switch):
Connection-specific DNS Suffix . :
Link-local IPv6 Address . . . . . : fe80::6d70:fd26:5e97:59a3%46
IPv4 Address. . . . . : 172.28.48.1
Subnet Mask . . . . . : 255.255.240.0
Default Gateway . . . . . :
Wireless LAN adapter Local Area Connection* 1:ipconfig /release
Media State . . . . . : Media disconnected
Connection-specific DNS Suffix . :
Wireless LAN adapter Local Area Connection* 10:
Media State . . . . . : Media disconnected
Connection-specific DNS Suffix . :
Ethernet adapter VMware Network Adapter VMnet1:


Network Alchemist
ping
Run Command
Input
Enter Address:
muhitkhan.tech
OK
Cancel


Network Alchemist
ping
Run Command
Pinging muhitkhan.tech [104.21.81.182] with 32 bytes of data:
Reply from 104.21.81.182: bytes=32 time=171ms TTL=57
Reply from 104.21.81.182: bytes=32 time=155ms TTL=57
Reply from 104.21.81.182: bytes=32 time=152ms TTL=57
Reply from 104.21.81.182: bytes=32 time=160ms TTL=57
Ping statistics for 104.21.81.182:
Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
Minimum = 152ms, Maximum = 171ms, Average = 159ms


Network Alchemist
tracert
Run Command
Tracing route to kiit.ac.in [104.21.79.189]
over a maximum of 30 hops:
1 2 ms 1 ms 1 ms 10.5.224.2
2 9 ms 8 ms 12 ms 192.168.102.1
3 6 ms 4 ms 2 ms 103.106.200.1
4 3 ms * 8 ms 125.20.251.45
5 138 ms 125 ms 139 ms 182.79.198.125
6 * 160 ms * de-cix-frankfurt.as13335.net [80.81.193.129]
7 154 ms 158 ms 154 ms 172.71.244.5
8 138 ms 155 ms 137 ms 104.21.79.189
Trace complete.


Network Alchemist
nslookup
Run Command
Server: one.one.one.one
Address: 1.1.1.1
DNS request timed out.
timeout was 2 seconds.
Name: google.com
Addresses: 2404:6800:4007:824::200e
142.250.195.110

4.2 Software Interfaces

```python
import tkinter as tk
from tkinter import ttk
from subprocess import Popen, PIPE
from tkinter import simpledialog
import ctypes, sys

def is_admin():
    try:
        return ctypes.windll.shell32.IsUserAnAdmin()
    except:
        return False

# Define the commands
commands = ["ipconfig /all", "ping", "tracert", "nslookup", "ipconfig /flushdns",
            "ipconfig /release", "netsh int ip reset", "netsh winsock reset"]

def run_command():
    command = selected_command.get()
    if command in ["ping", "tracert", "nslookup"]:
        address = simpledialog.askstring("Input", "Enter Address:", parent=root)
        if address is not None:
            command = command + " " + address
    process = Popen(command, stdout=PIPE, stderr=PIPE, shell=True)
    output, error = process.communicate()
    output_text.delete('1.0', tk.END)
    output_text.insert(tk.END, output)

if is_admin():
    # User is admin
    # Create the main window
    root = tk.Tk()
    root.title("Network Alchemist")  # Set the window title

    # Create a dropdown menu
    selected_command = tk.StringVar()
    command_menu = ttk.Combobox(root, textvariable=selected_command)
    command_menu['values'] = commands
    command_menu.current(0)
    command_menu.pack()

    # Create a button to run the command
    run_button = tk.Button(root, text="Run Command", command=run_command)
    run_button.pack()

    # Create a text box for the output
    output_text = tk.Text(root)
    output_text.pack()

    # Start the main loop
    root.mainloop()
else:
    # Re-run the program with admin rights
    ctypes.windll.shell32.ShellExecuteW(None, "runas", sys.executable, " ".join(sys.argv), None, 1)

```

5. Benefits

Enhancing Network Troubleshooting

Streamlined Troubleshooting Process: The core objective of this application is to streamline the network troubleshooting process, thereby eliminating complexities and inefficiencies.

Universal Accessibility: In the realm of network troubleshooting, accessibility is paramount. This application caters not only to seasoned network administrators who possess in-depth knowledge of CLI commands but also to individuals who are relatively new to network diagnostics. For the latter group, the CLI can often appear as an insurmountable barrier. The GUI offered by this application levels the playing field, offering an approachable entry point to network troubleshooting.

Enhanced Efficiency: Beyond accessibility, this application seeks to optimize the efficiency of network troubleshooting endeavours. For experienced professionals, it provides a tool that significantly expedites the process.

In conclusion, this application redefines network troubleshooting by not only rendering it more accessible to the novice but also by empowering the experts to work with unparalleled efficiency.

6. Future Scope

The network troubleshooting application, as envisioned, is poised to serve as a foundational cornerstone in the domain of network management.

Some key areas for future enhancements include:

6.1. Persistent Command Output Storage

A prospective enhancement of paramount significance is the incorporation of a feature that enables users to save command outputs for future reference. This evolution aligns with the imperatives of knowledge retention and historical analysis.

6.2. Integration of Advanced Network Troubleshooting Tools

The relentless evolution of network technologies necessitates a corresponding evolution of troubleshooting methodologies. Future iterations of the application could encompass the seamless integration of advanced network troubleshooting tools.

6.3. Integration of Machine Learning and Automation

Looking ahead, the application's potential could extend to the integration of machine learning algorithms and automation capabilities. This can usher in predictive analysis, anomaly detection, and proactive troubleshooting, ultimately enhancing the efficacy of network management.

7. Conclusion

This Software Requirements Specification (SRS) provides a foundational framework for the network troubleshooting application, offering a high-level overview of its requirements and objectives. As the application matures, it remains poised to adapt, innovate, and continue serving as an invaluable asset in the realm of network troubleshooting.
