# NETWORK MONITORING DASHBOARD



## A PROJECT REPORT

***Submitted by***

**DHARUNRAJ P (9517202109015)**

**HARIHARAN VIGNESH**  **K (9517202109021)**

**JAYARAM N (9517202109022)**

***In partial fulfillment for the award of the degree***

### of

**BACHELOR OF ENGINEERING**

### in

**ARTIFICIAL INTELLIGENCE AND DATA SCIENCE**

### MEPCO SCHLENK COLLEGE OF ENGINEERING, SIVAKASI-626123

**(An Autonomous Institution) ANNA UNIVERSITY: CHENNAI 600 025**

**October 2023**

**MEPCO SCHLENK ENGINEERING COLLEGE, SIVAKASI**

**AUTONOMOUS**

**DEPARTMENT OF ARTIFICIAL INTELLIGENCE AND DATA SCIENCE**

## https://lh5.googleusercontent.com/zNka7zD4qVeBnvWlplpLRs0AhIfyJjynl0Y2sPhYAkYB_kgqY56FgxjF1WVSLdzKUNV1mSRE1xG8FJnbkZZ529aoZw5spPvW0R-MyUTvZzc5MNA6XyrD5FI-pmidFP5Fo-JuhJaom5SYt9aa2d89-bhUVc1HPBPTS704MiLgcRAmaFCTlQGTRmeON30JVQC6VpjQ25gmNw

## BONAFIDE CERTIFICATE

Certified that this project report“**NETWORK MONITORING DASHBOARD**” is the bonafide work of **DHARUNRAJ P(9517202109015),HARIHARAN VIGNESH K(9517202109021), JAYARAM N (9517202109022)** who carried out the project work under my supervision.

**SIGNATURE SIGNATURE**

**Mrs.P.Swathika,M.E.,(Ph.D) Dr.J.Angela Jennifa Sujana, M.E.,Ph.D**

**Assistant Professor(SG) Professor & Head**

Artificial Intelligence and Data Science Artificial Intelligence and Data Science Mepco Schlenk Engineering College,Mepco Schlenk Engineering College Sivakasi- 626 005,Virudhunagar Sivakasi- 626 005,Virudhunagar

The project report submitted for the viva voce held on ………….

**INTERNAL EXAMINER EXTERNAL EXAMINER**

### TABLE OF CONTENTS

**Chapter 1: Introduction**

1.1 Introduction………………………………………………………....Page No. 5

1.2 Objectives…………………………………………………………...Page No. 6

1.3 Scope of the project…………………………………………………Page No. 6

1.4 Function used………………………………………………………..Page No. 6

1.5 Report Summary……………………………………………………Page No. 8

**Chapter 2: Implementation**

2.1 Program Coding……….……………………………………………Page No. 9

2.2 Output……………………...………………………………………Page No. 16

**Chapter 3: Conclusion**

**ABSTRACT**

The program aims to provide a graphical user interface for monitoring various aspects of a computer system, including fetching system information, displaying device status, analyzing network traffic, showing performance metrics, and allowing user access to view data. The dashboard includes buttons for different functionalities such as fetching data, displaying device status, analyzing traffic, showing metrics, user access, and exiting the program.

The `checkUserAccess` function is responsible for validating user credentials, and if the entered username and password match the predefined values, access is granted to view data. The `fetchInput`, `displayDeviceStatus`, `displayTrafficAnalysis`, and `showPerformanceMetrics` functions utilize system commands to retrieve and display information related to the operating system, CPU, memory, network, and disk usage. The `viewData` function lists the contents of the user's home directory and a specified directory in terms of file size and modification time.

The GTK library is used to create a graphical user interface with buttons that trigger the corresponding functionalities. Each button is associated with a callback function that performs the specified action when the button is clicked. Overall, this program provides a basic foundation for a network monitoring tool with a user-friendly interface.

Moreover, the program employs a combination of system commands and file operations to gather and present real-time information on crucial system parameters. By tapping into systeminfo, ipconfig, netstat, and other commands, the application dynamically fetches and displays data, providing an up-to-date snapshot of the system's health. The use of popen for command execution and GTK for the graphical interface showcases a synergistic integration of low-level system interactions and high-level user interaction, offering a versatile solution adaptable to diverse user preferences and requirements. In essence, this network monitoring dashboard not only facilitates efficient data retrieval but also prioritizes user interaction and accessibility in its design.

**CHAPTER 1**

**INTRODUCTION**

**1.1 INTRODUCTION**

The "Network monitoring dashboard" project is designed to create a network monitoring dashboard with a graphical user interface using the GTK library. The program encompasses a range of functionalities aimed at system information retrieval, device status display, traffic analysis, performance metrics, and user access control. The graphical user interface is constructed with buttons that trigger specific actions, such as fetching system data, displaying device status, analyzing network traffic, showing performance metrics, granting user access, and exiting the application.

The code is organized into various functions, each dedicated to a specific aspect of the network monitoring dashboard. Functions like `fetchInput`, `displayDeviceStatus`, `displayTrafficAnalysis`, `showPerformanceMetrics`, and `viewData` handle distinct tasks related to data retrieval and presentation. Additionally, event-handling functions such as `onFetchDataClicked`, `onDisplayStatusClicked`, etc., are associated with specific buttons in the graphical user interface and execute corresponding actions upon user interaction.

User access control is implemented through a hardcoded username and password (`admin` and `admin123`), with the `checkUserAccess` function verifying the provided credentials. In the case of valid access, the program proceeds to display user-specific data using the `viewData` function. This code serves as a foundational framework for a network monitoring tool, offering a user-friendly interface for efficient navigation and interaction with various system-related functionalities.

**1.2.OBJECTIVE OF THE PROJECT**

### The objective of the "Optimal Distance" project is to create a network monitoring dashboard with a graphical user interface using the GTK library.The dashboard presents various functionality buttons, each triggering specific system and network monitoring actions. These actions include fetching and displaying system information, showing device status, analyzing network traffic, presenting performance metrics, and allowing users to view data from their home directory. The GTK framework is utilized for constructing the graphical interface, and system commands are employed to gather and display relevant information about the system and network. The program aims to provide a comprehensive yet user-friendly tool for monitoring and understanding key aspects of the computer system and network.

### 1.3.SCOPE OF THE PROJECT

The "Network monitoring dashboard" project aims to aim at providing various functionalities related to fetching system information, displaying device status, analyzing network traffic, showing performance metrics, and viewing data.

### 1.4.FUNCTION USED:

**1.GET CAPTION**

**wmic cpu get caption(**)-Using `wmic cpu get caption` fetches information about your CPU,specifically the "caption" property. This will give you the name or description of your CPU.

1. **GET CAPACITY**

**wmic memorychip get capacity()**-The `wmic memorychip get capacity` command used to retrieve information about the capacity of memory chips (RAM modules) installed on a Windows system.

**3.GET LOAD PERCENTAGE**

**wmic cpu get loadpercentage()**-The `wmic cpu get loadpercentage` command is a Windows Management Instrumentation Command-line (WMIC) query that retrieves the current CPU load percentage. When you run this command in the Command Prompt, it provides information about the CPU utilization at the moment.

**4.FREE PHYSICAL MEMORY**

**wmic os get freephysicalmemory()**-The `wmic os get freephysicalmemory` is a Windows Management Instrumentation Command-line tool that retrieves information about the operating system, and in this case, it fetches the amount of free physical memory.

**5.GET NAME**

**wmic nic get Name()**-The`wmic nic get Name` is a Windows Management Instrumentation Command-line (WMIC) tool. It retrieves information about network interface controllers (NICs) on your system. Specifically, it gets the names of the NICs.

**6.NETCONNECTION STATUS**

**wmic nic get NetConnectionStatus()**- The`wmic nic get NetConnectionStatus` is a command used in Windows to retrieve the network connection status of network interface cards (NICs) on a computer. In the context of `NetConnectionStatus`, it retrieves the current connection status of the NICs.

**7.IPCONFIG**

**ipconfig()**-The `ipconfig` is a command-line utility in Windows used to display the current TCP/IP network configuration on a system. It provides information about the computer's IP address, subnet mask, default gateway, and other network-related details.

**8.TASKLIST**

**tasklist()**-The `tasklist()` function is actually a command in Windows that displays a list of all running processes on your computer. It shows the process name, process ID (PID), and other information.

**9.NETSTAT**

**netstat -an()**-The `netstat -an` is a command-line utility used to display network connections, routing tables, interface statistics, masquerade connections, and multicast memberships.

**10.PING**

**ping ()**-The `ping` is a network utility used to test the reachability of a host on an Internet Protocol (IP) network. It also measures the round-trip time for messages sent from the originating host to a destination computer.

When you use the `ping` command followed by a specific IP address or domain name, your computer sends a small packet of data to the target, and the target responds. The response time is measured, and the utility reports various statistics, including the round-trip time, packet loss, and other details.

### 1.5.REPORT SUMMARY

The project successfully demonstrates the implementation of basic network monitoring dashboard with features like system information retrieval, device status display, traffic analysis, performance metrics, and data viewing. Overall, it provides a foundation for a network monitoring tool.

**CHAPTER-2**

**IMPLEMENTATION**

**2.1 PROGRAM CODING**

#include <stdio.h>

#include <gtk/gtk.h>

#include <stdio.h>

#include <stdlib.h>

#include <stdbool.h>

#include <dirent.h>

#include <sys/types.h>

#include <sys/stat.h>

#include <unistd.h>

#include <string.h>

#include <time.h>

const char \*validUsername = "admin";

const char \*validPassword = "admin123";

bool checkUserAccess(const char \*username, const char \*password)

{

return (strcmp(username, validUsername) == 0 && strcmp(password, validPassword) == 0);

}

void fetchInput()

{

g\_print("Fetching system information...\n");

FILE \*osInfo = popen("systeminfo | findstr /B /C:\"OS Name\" /C:\"OS Version\"", "r");

char buffer[256];

while (fgets(buffer, sizeof(buffer), osInfo) != NULL) {

g\_print("%s", buffer);

}

pclose(osInfo);

FILE \*cpuInfo = popen("**wmic cpu get caption**", "r");

while (fgets(buffer, sizeof(buffer), cpuInfo) != NULL)

{

g\_print("CPU: %s", buffer);

}

pclose(cpuInfo);

FILE \*memoryInfo = popen("**wmic memorychip get capacity**", "r");

while (fgets(buffer, sizeof(buffer), memoryInfo) != NULL)

{

g\_print("Memory: %s", buffer);

}

pclose(memoryInfo);

g\_print("Network Status: Connected\n");

}

void displayDeviceStatus()

{

g\_print("Displaying device status...\n");

system("echo 'Router Status:'");

system("**ipconfig**");

system("echo 'Server Status:'");

system("**tasklist**");

}

void displayTrafficAnalysis()

{

printf("'Traffic Analysis:'\n");

system("**netstat -an**");

printf("'Bandwidth Usage:'\n");

system("**ping -n 5 google.com**");

printf("Identifying Potential Bottlenecks/Anomalies:\n");

FILE \*cpuUsageFile = popen("**wmic cpu get loadpercentage** | findstr /r /v '^$'", "r");

int cpuUsage;

fscanf(cpuUsageFile, "%d", &cpuUsage);

pclose(cpuUsageFile);

printf("CPU Usage: %d\n", cpuUsage);

FILE \*diskUsageFile = popen("fsutil volume diskfree C: | findstr /r /v '^$'", "r");

unsigned long long freeBytes, totalBytes;

fscanf(diskUsageFile, "%\*s %\*s %llu %llu", &freeBytes, &totalBytes);

pclose(diskUsageFile);

double diskUsage = ((totalBytes - freeBytes) / (double)totalBytes) \* 100.0;

printf("Disk Usage: %.2fbytes\n", diskUsage);

}

void showPerformanceMetrics()

{

g\_print("Showing performance metrics...\n");

FILE \*cpuUsage = popen("wmic cpu get loadpercentage 2>&1", "r");

if (cpuUsage == NULL)

{

g\_print("Error opening CPU usage command.\n");

return;

}

char buffer[256];

while (fgets(buffer, sizeof(buffer), cpuUsage) != NULL)

{

g\_print("CPU Usage: %s", buffer);

}

if (ferror(cpuUsage) != 0)

{

g\_print("Error reading from CPU usage command.\n");

}

if (pclose(cpuUsage) == -1)

{

g\_print("Error closing CPU usage command.\n");

}

FILE \*memoryUsage = popen("**wmic os get freephysicalmemory**, totalvisiblememorysize | findstr /r /v '^$'", "r");

double freeMemory, totalMemory;

fscanf(memoryUsage, "%\*s %lf %\*s %\*s %lf", &freeMemory, &totalMemory);

fclose(memoryUsage);

double memoryUsagePercentage = ((totalMemory - freeMemory) / totalMemory) \* 100.0;

g\_print("Memory Usage: %.2f%%\n", memoryUsagePercentage);

FILE \*networkThroughput = popen("**wmic nic get Name, NetConnectionStatus**, Speed 2>&1", "r");

if (networkThroughput == NULL)

{

g\_print("Error opening network throughput command.\n");

}

else

{

while (fgets(buffer, sizeof(buffer), networkThroughput) != NULL) {

g\_print("Network Throughput: %s", buffer);

}

if (ferror(networkThroughput) != 0)

{

g\_print("Error reading from network throughput command.\n");

}

if (pclose(networkThroughput) == -1)

{

g\_print("Error closing network throughput command.\n");

}

}

FILE \*diskUsage = popen("**wmic logicaldisk get Name**, **FreeSpace**, Size 2>&1", "r");

if (diskUsage == NULL) {

g\_print("Error opening disk usage command.\n");

}

else

{

while (fgets(buffer, sizeof(buffer), diskUsage) != NULL)

{

g\_print("Disk Usage: %s", buffer);

}

if (ferror(diskUsage) != 0)

{

g\_print("Error reading from disk usage command.\n");

}

if (pclose(diskUsage) == -1)

{

g\_print("Error closing disk usage command.\n");

}

}

}

void viewData()

{

g\_print("Viewing data...\n");

g\_print("Data Source: User's Home Directory\n");

system("ls -l ~/");

const char \*directoryPath = "C:\\";

g\_print("Viewing data from directory: %s\n", directoryPath);

DIR \*dir = opendir(directoryPath);

if (dir)

{

struct dirent \*entry;

struct stat file;

struct stat fileStat;

int count = 1;

while ((entry = readdir(dir)) != NULL)

{

char filePath[256];

snprintf(filePath, sizeof(filePath), "%s%s", directoryPath, entry->d\_name);

if (stat(filePath, &fileStat) == 0)

{

g\_print("%d. %s\n", count, entry->d\_name);

g\_print(" Size: %lld bytes\n", (long long)fileStat.st\_size);

g\_print(" Last Modified: %s", ctime(&fileStat.st\_mtime));

count++;

}

}

closedir(dir);

}

else

{

g\_print("Error: Unable to open the directory.\n");

}

}

void onExitClicked(GtkWidget \*widget, gpointer data)

{

g\_print("Exiting the program...\n");

gtk\_main\_quit();

}

void onFetchDataClicked(GtkWidget \*widget, gpointer data)

{

fetchInput();

}

void onDisplayStatusClicked(GtkWidget \*widget, gpointer data)

{

displayDeviceStatus();

}

void onAnalyzeTrafficClicked(GtkWidget \*widget, gpointer data)

{

displayTrafficAnalysis();

}

void onShowMetricsClicked(GtkWidget \*widget, gpointer data)

{

showPerformanceMetrics();

}

void onUserAccessClicked(GtkWidget \*widget, gpointer data)

{

GtkWidget \*dialog;

GtkWidget \*contentArea;

GtkWidget \*grid;

GtkWidget \*labelUsername, \*labelPassword;

GtkWidget \*entryUsername, \*entryPassword;

dialog = gtk\_dialog\_new\_with\_buttons("User Access", NULL, GTK\_DIALOG\_MODAL, "OK", GTK\_RESPONSE\_OK, "Cancel", GTK\_RESPONSE\_CANCEL, NULL);

contentArea = gtk\_dialog\_get\_content\_area(GTK\_DIALOG(dialog));

grid = gtk\_grid\_new();

gtk\_container\_add(GTK\_CONTAINER(contentArea), grid);

labelUsername = gtk\_label\_new("Username:");

gtk\_grid\_attach(GTK\_GRID(grid), labelUsername, 0, 0, 1, 1);

entryUsername = gtk\_entry\_new();

gtk\_grid\_attach(GTK\_GRID(grid), entryUsername, 1, 0, 1, 1);

labelPassword = gtk\_label\_new("Password:");

gtk\_grid\_attach(GTK\_GRID(grid), labelPassword, 0, 1, 1, 1);

entryPassword = gtk\_entry\_new();

gtk\_entry\_set\_visibility(GTK\_ENTRY(entryPassword), FALSE); gtk\_grid\_attach(GTK\_GRID(grid), entryPassword, 1, 1, 1, 1);

gtk\_widget\_show\_all(dialog);

gint result = gtk\_dialog\_run(GTK\_DIALOG(dialog));

const char \*enteredUsername = gtk\_entry\_get\_text(GTK\_ENTRY(entryUsername));

const char \*enteredPassword = gtk\_entry\_get\_text(GTK\_ENTRY(entryPassword));

if (result == GTK\_RESPONSE\_OK)

{

if (checkUserAccess(enteredUsername, enteredPassword))

{

g\_print("Access granted!\n");

viewData();

}

else

{

g\_print("Access denied. Invalid username or password.\n");

}

}

gtk\_widget\_destroy(dialog);

}

int main(int argc, char \*argv[])

{

GtkWidget \*window;

GtkWidget \*grid;

GtkWidget \*fetchButton, \*statusButton, \*trafficButton, \*metricsButton, \*dataButton, \*accessButton,\*exitButton;

gtk\_init(&argc, &argv);

window = gtk\_window\_new(GTK\_WINDOW\_TOPLEVEL);

gtk\_window\_set\_title(GTK\_WINDOW(window), "Network Monitoring Dashboard");

g\_signal\_connect(window, "destroy", G\_CALLBACK(gtk\_main\_quit), NULL);

grid = gtk\_grid\_new();

gtk\_container\_add(GTK\_CONTAINER(window), grid);

fetchButton = gtk\_button\_new\_with\_label("Fetch Data");

statusButton = gtk\_button\_new\_with\_label("Display Status");

trafficButton = gtk\_button\_new\_with\_label("Analyze Traffic");

metricsButton = gtk\_button\_new\_with\_label("Show Metrics");

dataButton = gtk\_button\_new\_with\_label("View Data");

accessButton = gtk\_button\_new\_with\_label("User Access");

exitButton = gtk\_button\_new\_with\_label("Exit");

g\_signal\_connect(fetchButton, "clicked", G\_CALLBACK(onFetchDataClicked), NULL);

g\_signal\_connect(statusButton, "clicked", G\_CALLBACK(onDisplayStatusClicked), NULL);

g\_signal\_connect(trafficButton, "clicked", G\_CALLBACK(onAnalyzeTrafficClicked),

NULL);

g\_signal\_connect(metricsButton, "clicked", G\_CALLBACK(onShowMetricsClicked),

NULL);

g\_signal\_connect(accessButton, "clicked", G\_CALLBACK(onUserAccessClicked),

window);

g\_signal\_connect(exitButton, "clicked", G\_CALLBACK(onExitClicked), NULL);

gtk\_grid\_attach(GTK\_GRID(grid), fetchButton, 0, 0, 1, 1);

gtk\_grid\_attach(GTK\_GRID(grid), statusButton, 0, 1, 1, 1);

gtk\_grid\_attach(GTK\_GRID(grid), trafficButton, 0, 2, 1, 1);

gtk\_grid\_attach(GTK\_GRID(grid), metricsButton, 0, 3, 1, 1);

gtk\_grid\_attach(GTK\_GRID(grid), accessButton, 0, 4, 1, 1);

gtk\_grid\_attach(GTK\_GRID(grid), exitButton, 0, 10, 1, 1);

gtk\_widget\_set\_hexpand(fetchButton, TRUE);

gtk\_widget\_set\_hexpand(statusButton, TRUE);

gtk\_widget\_set\_hexpand(trafficButton, TRUE);

gtk\_widget\_set\_hexpand(metricsButton, TRUE);

gtk\_widget\_set\_hexpand(accessButton, TRUE);

gtk\_widget\_set\_hexpand(exitButton, TRUE);

gtk\_widget\_set\_halign(fetchButton, GTK\_ALIGN\_FILL);

gtk\_widget\_set\_halign(statusButton, GTK\_ALIGN\_FILL);

gtk\_widget\_set\_halign(trafficButton, GTK\_ALIGN\_FILL);

gtk\_widget\_set\_halign(metricsButton, GTK\_ALIGN\_FILL);

gtk\_widget\_set\_halign(accessButton, GTK\_ALIGN\_FILL);

gtk\_widget\_set\_halign(exitButton, GTK\_ALIGN\_FILL);

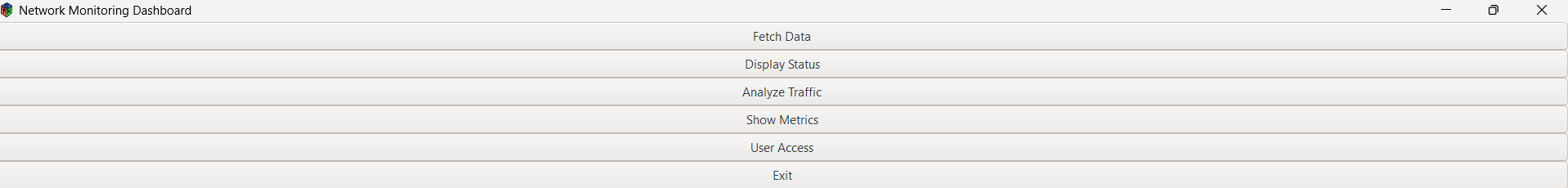
gtk\_widget\_show\_all(window);

gtk\_main();

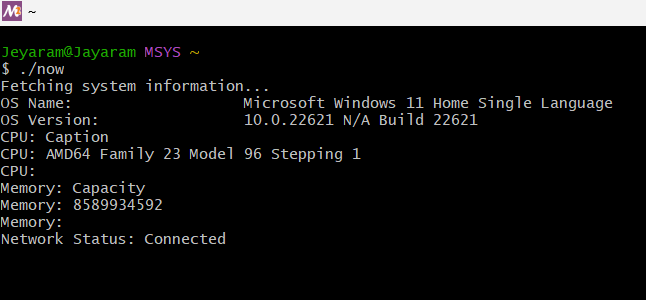
return 0;

}

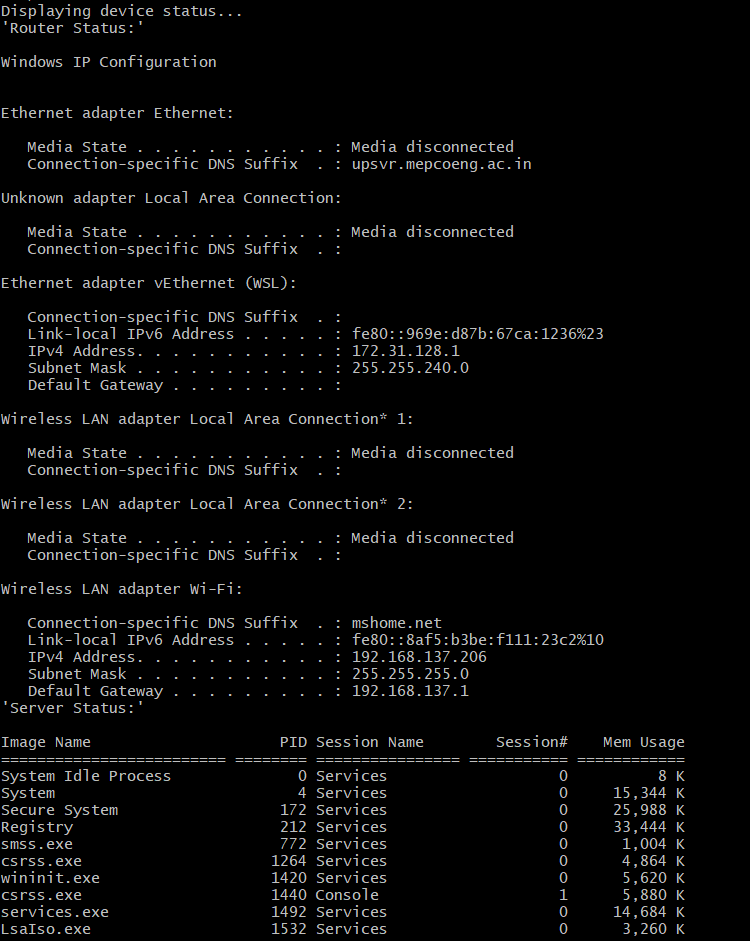
**2.2 OUTPUT**

****

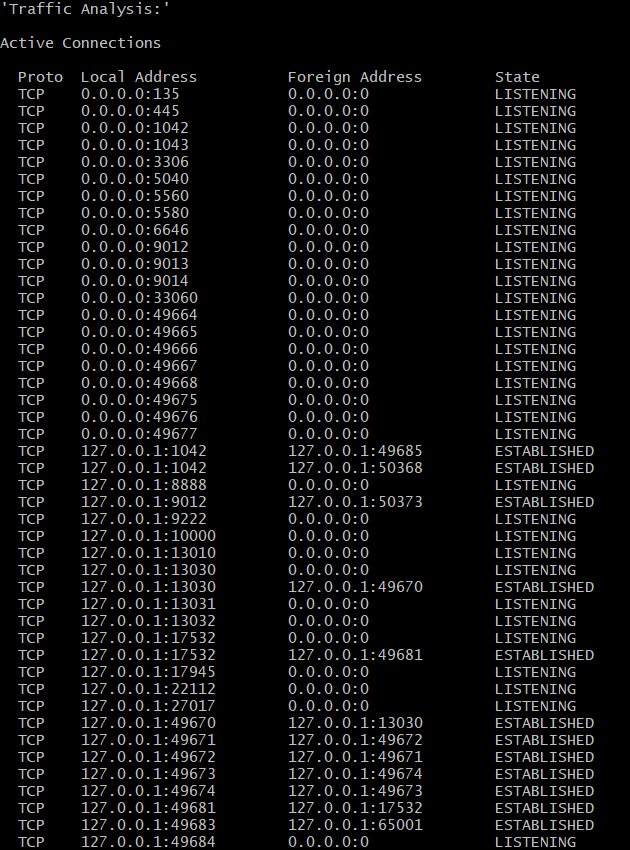
**Fig-2.1 GUI DASH-BOARD**

****

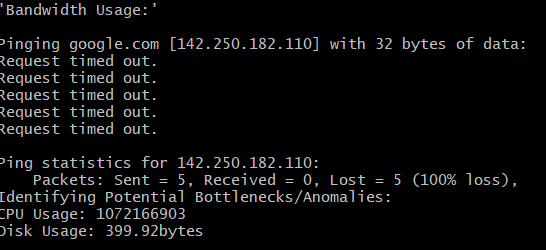
**Fig-2.2 Fetching System Information**

****

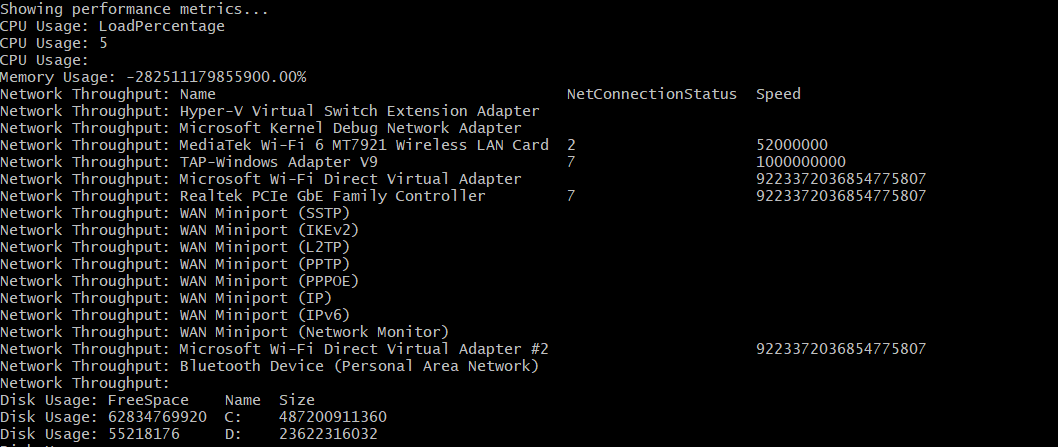
**Fig-2.3 Display status**

****

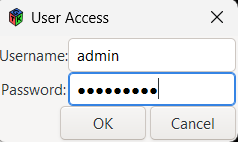
**Fig-2.4 Traffic Analysis**

****

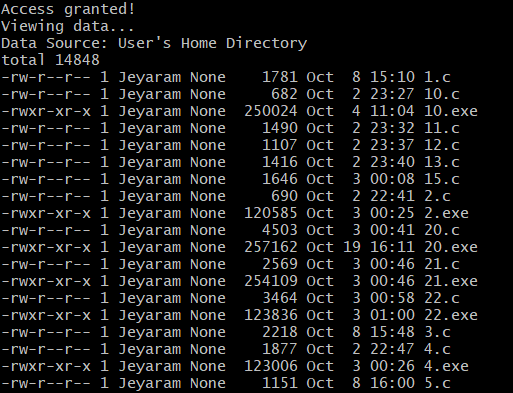
**Fig-2.5 Bandwidth Usage**

****

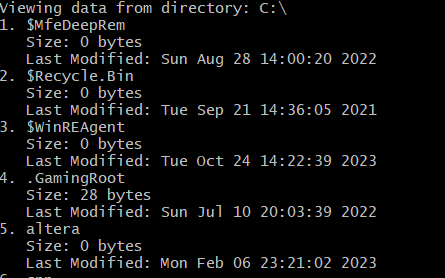
**Fig-2.6 Performance Metrics**

****

**Fig-2.7 User Access Login**

****

**Fig-2.8 Viewing Home Directory files**

****

**Fig-2.9 Viewing Drive Files**

**CHAPTER 3**

**CONCLUSION**

**3.1 CONCLUSION**

The code establishes a basic network monitoring dashboard using GTK for a graphical interface. The network management dashboard, implemented in C with GTK, serves as a foundational tool for monitoring system metrics and network status. While providing essential functionalities such as data fetching, device status display, traffic analysis, and user access control, it could benefit from improved security measures like input validation. Additionally, a more modular and organized structure would enhance scalability and code maintenance in the long run.