Звіт

про виконання завдання з лабораторної роботи №1

з курсу “Комп'ютерні мережі”

тема “Розробка програми моніторингу мережі”

студентом Антонюком Олександром Андрійовичем (група КН-21)

в 2023-2024 навчальному році

за індивідуальним варіантом даних №1

Мета: Метою цієї лабораторної роботи є навчання студентів розробці програми моніторингу мережі з використанням мови програмування Python та створення інтерфейсу для відстеження стану обладнання та мережевого трафіку. В рамках цієї роботи студенти отримають навички роботи з мережевими бібліотеками, створенням графічного інтерфейсу користувача та аналізом мережевих даних.

```python
import tkinter as tk

from helpers import update_values
from labels import initialize_labels
from treeview import initialize_treeview, initialize_treeview_values

if __name__ == "__main__":
    root = tk.Tk()
    root.title("System Monitor")
    root.resizable(width=False, height=False)
    tree = initialize_treeview(root)
    initialize_treeview_values(tree)
    labels = initialize_labels(root)
    update_values(tree, labels)
    root.mainloop()
```

Рис. 1 - Зображення головного скрипту

На рис.1 можна побачити, що програма створює вікно за допомогою бібліотеки Tkinter. Після цього вона встановлює заголовок програми та відключає масштабування вікна. Далі відбувається ініціалізація деревоподібного віджету (TreeView) за допомогою функції initialize_treeview. Цей віджет використовується

для відображення інтерфейсів мережі та показу обсягу відправленого та отриманого мережевого трафіку.

Після ініціалізації деревоподібного віджету його заповнюють даними. Для цього також викликається функція initialize_labels, що створює набір міток (labels). Далі програма викликає функцію update_values. Ця функція відповідає за оновлення назв інтерфейсів мережі, обсягу відправленого та отриманого мережевого трафіку, використання CPU, оперативної пам'яті та дискового простору у відсотках.

```
def initialize_treeview(root):
    tree = ttk.Treeview(root, columns=("Column 1", "Column 2", "Column 3", "Column 4"),
                        show="headings")
    tree.heading("#1", text="ID")
    tree.heading("#2", text="Interface")
    tree.heading("#3", text="Bytes Sent")
    tree.heading("#4", text="Bytes Received")
    return tree
```

Рис. 2 - Зображення функції initialize_treeview

Функція initialize_treeview створює об’єкт деревоподібного віджету з допомогою конструктора ttk.Treeview з чотирма колонками ID, Interface, Bytes Sent, Bytes Received та вказує, що відображатись будуть лише заголовки колонок.

```
def initialize_treeview_values(tree):
    num = calc_number_of_interfaces()
    step = 1

    for j in range(num):
        values = [f"Value{i}" for i in range(step, step + 3)]
        tree.insert("", "end", text="Parent", values=values)
        step += 3

    tree.pack(expand=True)
```

Рис. 3 - Зображення функції initialize_treeview

Функція initialize_treeview_values визначає кількість мережевих інтерфейсів за допомогою функції calc_number_of_interfaces і використовує це значення для налаштування кількості клітинок та рядків у деревоподібному віджеті Treeview.

```
def calc_number_of_interfaces():
    network_stats = psutil.net_io_counters(pernic=True)
    return len(network_stats.values())
```

Рис. 4 - Зображення функції calc_number_of_interfaces

```
def initialize_labels(root):
    labels = []
    labels.append(ttk.Label(root, text=f"CPU Usage"))
    labels.append(ttk.Label(root, text=f"Memory Usage"))
    labels.append(ttk.Label(root, text=f"Disk Usage"))

    for label in labels:
        label.pack()

    return labels
```

Рис. 5 - Зображення функції initialize_treeview

Функція initialize_labels створює список, що містить об’єкти міток, що відображають значення використання CPU, оперативної пам'яті та дискового простору у відсотках та розміщає кожну мітку на вікні користувача.

```
def update_values(tree, labels):
    update_treeview(tree)
    update_labels(labels)

    tree.after(1000, update_values, tree, labels)
```

Рис. 6 - Зображення функції initialize_treeview

Функція update_values відповідає за оновлення значень деревоподібного віджета Treeview за допомогою функції update_treeview та оновлення значень міток за допомогою функції update_labels.

```
def update_treeview(tree):
    list_of_interfaces = get_list_of_interfaces()
    for index, interface in enumerate(list_of_interfaces):
        new_values = []
        new_values.append(interface.get_index())
        new_values.append(interface.get_name())
        new_values.append(interface.get_bytes_sent())
        new_values.append(interface.get_bytes_received())

        tree.item(tree.get_children()[index], values=new_values)
```

Рис. 7 - Зображення функції update_treeview

Функція update_treeview отримує дані про мережеві інтерфейси за допомогою функції get_list_of_interfaces та замінює старні дані на отримані в деревоподібному віджеті Treeview

```
def get_list_of_interfaces():
    list_of_interfaces = []
    network_stats = psutil.net_io_counters(pernic=True)
    for index, key in enumerate(network_stats.keys()):
        value = network_stats[key]
        interface = Interface()
        interface.set_index(index)
        interface.set_name(key)
        interface.set_bytes_sent(value.bytes_sent)
        interface.set_bytes_received(value.bytes_recv)
        list_of_interfaces.append(interface)
    return list_of_interfaces
```

Рис. 8 - Зображення функції get_list_of_interfaces

Функція get_list_of_interfaces використовує функцію net_io_counters з бібліотеки psutil, щоб отримати дані про мережеві інтерфейси. На основі цих даних вона формує список, де кожен елемент є екземпляром класу Interface, який зберігає інформацію про індекс, назву, надіслані та отримані байти для кожного мережевого інтерфейсу.

```python
def update_labels(labels):
    cpu_percent, memory_usage, disk_usage = get_data()
    labels[0].config(text=f"CPU Usage: {cpu_percent}%")
    labels[1].config(text=f"Memory Usage: {memory_usage}%")
    labels[2].config(text=f"Disk Usage: {disk_usage}%")
```

Рис. 9 - Зображення функції update_labels

Функція update_labels отримує дані, за допомогою бібліоткеи psutil, про використання CPU, оперативної пам'яті та дискового простору за допомогою функції get_data та записує їх в мітки.

```python
def get_data():
    cpu_percent = psutil.cpu_percent()
    memory_usage = psutil.virtual_memory().percent
    disk_usage = psutil.disk_usage('/').percent
    return cpu_percent, memory_usage, disk_usage
```

Рис. 10 - Зображення функції update_labels

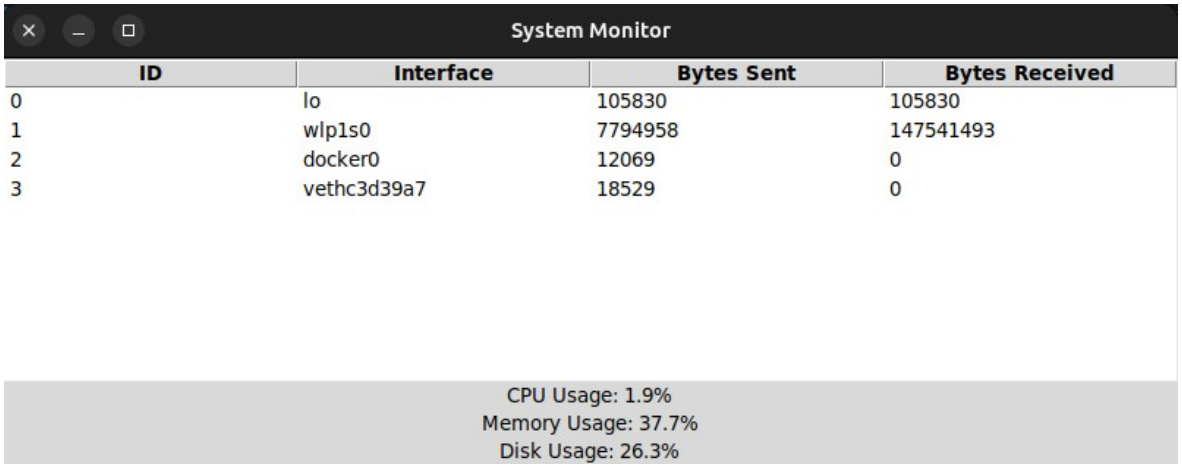

Рис. 11 - Зображення запущеної програми

Висновок: під час лабораторної роботи було створено програму моніторингу мережі за допомогою мови програмування Python та використано

бібліотеку psutil для отримання даних про стан системи та мережевих інтерфейсів. Крім того, був розроблений інтерфейс користувача для відстеження стану обладнання та мережевого трафіку з використанням бібліотеки tkinter.