DATA USAGE ANALYZER

A MINI PROJECT REPORT
21CSC202J -OPERATING SYSTEMS

Submitted by

Jai Kushal bysani [RA211030010215]

Siva Bhadra Nalam [RA211030010214]

Karthik Mungara [RA211030010208]

Under the guidance of

Ms.V. Vijayalakshmi

Assistant Professor, Department of Networking and Communication

in partial fulfilment of the requirements for the degree

of

M.TECH INTEGRATED

COMPUTER SCIENCE ENGINEERING

with specialization in

CYBER SECURITY

DEPARTMENT OF NETWORKING AND COMMUNICATIONS
FACULTY OF ENGINEERING AND TECHNOLOGY
SRM INSTITUTE OF SCIENCE AND TECHNOLOGY
KATTANKULATHUR – 603 203
MAY 2023

BONAFIDE CERTIFICATE

Certified that this project report titled "**DATA USAGE ANALYZER**" is the Bonafide work of **"Jai Kushal [Reg No: RA2211030010215], Siva Bhadra [Reg No:RA2211030010214], Karthik [Reg No: RA2211030010208]"** who carried out the project work under my supervision. Certified further, that to the best of my knowledge, the work reported herein does not form part of anyother thesis or dissertation based on which a degree or award was conferred on an earlier occasion for this or any other candidate.

Ms.V. Vijayalakshmi	**Dr. Annapurani Panaiyappan. K**
Assistant Professor Department of Networking and Communications	HEAD OF THE DEPARTMENT Department of Networking and Communications

SIGNATURE OF INTERNAL EXAMINER

SIGNATURE OF EXTERNAL EXAMINER

TABLE OF CONTENTS

OBJECTIVE

OBJECTIVES:

The objective of this Python code is to create an application that monitors a specific process's CPU and memory usage, issuing warnings if system-wide thresholds are exceeded. This application utilizes the Tkinter library to create a user-friendly graphical user interface (GUI) and runs two separate threads for real-time monitoring of both the targeted process and the system's resource usage. The user can start and stop the analysis as needed, and the main event loop ensures that the GUI remains responsive during the monitoring process.

The code is designed to detect if the CPU and memory usage of the monitored process or the system as a whole exceeds predefined thresholds. When these thresholds are crossed, the application will issue warnings. These warnings could be in the form of notifications, pop-up messages, or any other user-defined alerts.

The main event loop of the application is designed to ensure that the GUI remains responsive during the monitoring process. This means that the user can continue to interact with the application and make changes while the monitoring threads are running.
Overall, the objective of this code is to create a user-friendly monitoring application that helps users keep track of a specific process's resource usage and the overall system's resource utilization. It provides a real-time view of these metrics and alerts the user when predefined thresholds are exceeded. This could be useful for system administrators, developers, or anyone who needs to ensure that a specific process doesn't consume too many resources or to monitor system health.

ABSTRACT

ABSTRACT:

The "Data Usage Analyzer" application is a Python program that monitors and analyzes system resource usage. Users can input a Process ID (PID), and the application launches two threads.

The first thread tracks real-time CPU and memory usage for the specified process, while the second thread issues warnings if system-wide CPU or memory usage surpasses 50%. Users can stop the monitoring processes with a "Stop Analysis" button. This tool helps users manage system resources and prevent slowdowns or crashes caused by resource overload.

INTRODUCTION

INTRODUCTION:

This Python project is a "Data Usage Analyzer" designed to monitor and analyze resource usage for a specific process. It operates by tracking CPU and memory usage in real-time. Users input a Process ID (PID), and the application initiates two threads:

1.
Monitor Process Thread: This thread continuously displays the current CPU percentage, memory usage in megabytes, and memory percentagefor the specified process.

2.
Warning Thread: This thread monitors system-wide CPU and memoryusage, issuing warnings if they surpass 50%. This tool assists users in managing system resources efficiently and preventing potential performance issues.

PROBLEM STATEMENT

PROBLEM STATEMENT:

The "Data Usage Analyzer" application fulfills a vital role in system resource management. It serves as a valuable tool for users to gain insights into the performance of specific processes running on their system. By providing real-time CPU and memory utilization data, the application empowers users to make informed decisions about their system's resource allocation.

The warning system plays a crucial role in maintaining system stability, as it issues timely alerts when CPU or memory usage crosses the 50% threshold. This proactive approach helps users prevent system slowdowns or crashes caused by excessive resource consumption. Overall, the application enhances the user's ability to monitor and manage their system's resource usage effectively through its intuitive GUI and automatic warning features.

LIMITATIONS

LIMITATIONS:

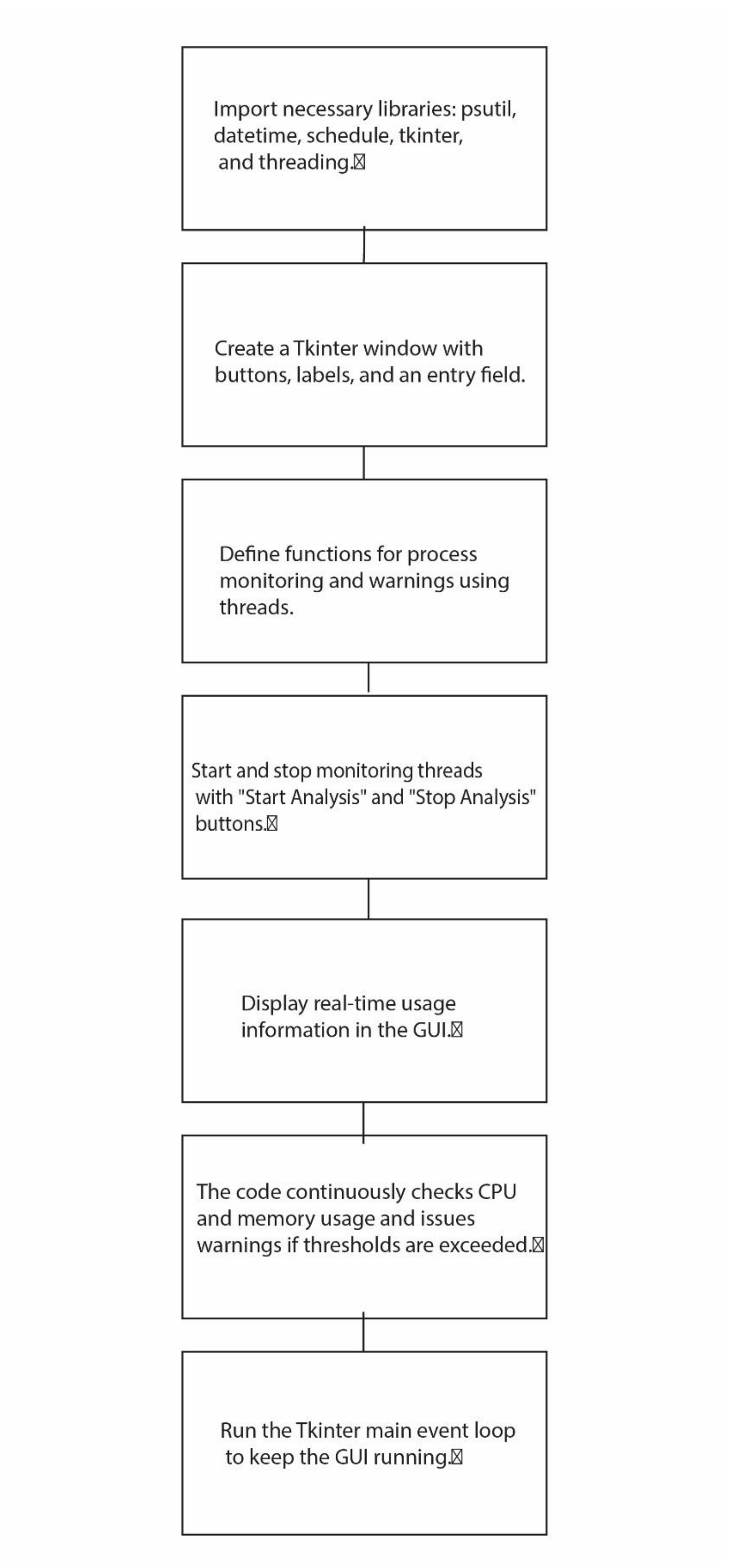
System Resource Utilization: Implementing a data usage analyzer within an operating system can consume significant system resources. This includes CPU, memory, and disk space, potentially impacting the overall performance and responsiveness of the system. This trade-off between monitoring and system efficiency can be a limitation, especially on resource-constrained devices or servers where every resource is critical.

Security and Privacy Concerns: Introducing a data usage analyzer at the operating system level can raise serious security and privacy concerns. It may require deep access to network traffic and system data, potentially exposing sensitive information. Ensuring the security of the tool and protecting user privacy is paramount but can be technically challenging, and data breaches or misuse are possible risks.

Operating System Compatibility: Developing and maintaining a data usage analyzer that is compatible with various operating systems can be a complex and resource-intensive task. Different operating systems have unique architectures, network stack implementations, and security mechanisms. Achieving consistent and accurate monitoring across diverse platforms can be challenging, leading to compatibility issues and limitations in terms of which operating systems are supported.

BLOCK DIAGRAM

BLOCK DIAGRAM:


Import necessary libraries: psutil, datetime, schedule, tkinter, and threading.
Create a Tkinter window with buttons, labels, and an entry field.
Define functions for process monitoring and warnings using threads.
Start and stop monitoring threads with "Start Analysis" and "Stop Analysis" buttons.
Display real-time usage information in the GUI.
The code continuously checks CPU and memory usage and issues warnings if thresholds are exceeded.
Run the Tkinter main event loop to keep the GUI running.

PROGRAM

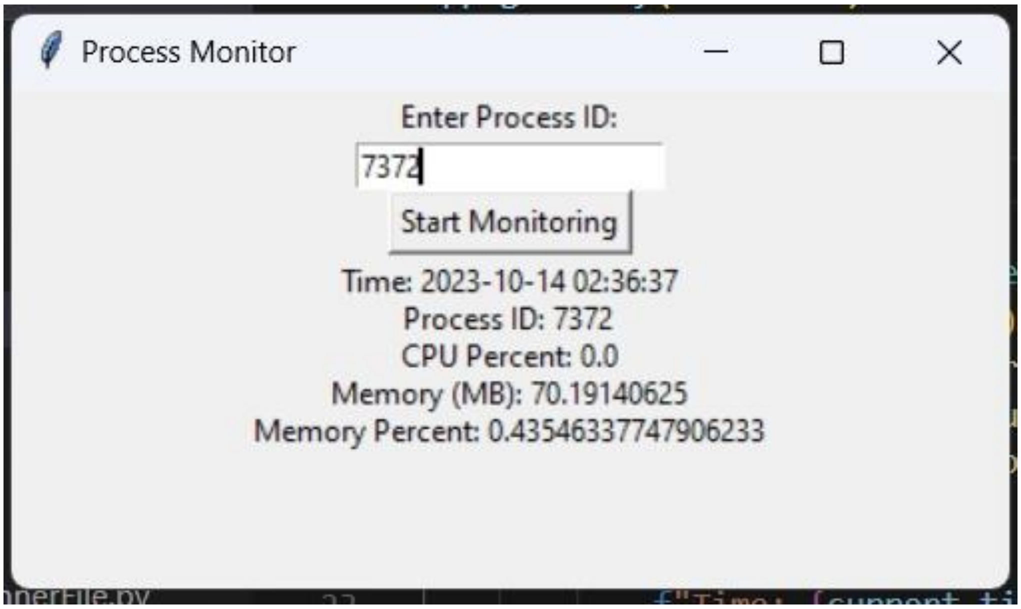
PYTHON PROGRAM(CODE):

```
import psutil
import datetime
import schedule
import tkinter as tk
from tkinter import messagebox
import threading

app = tk.Tk()
app.title("Data Usage Analyzer")
app.geometry("400x250")

def monitor_process_thread():
    pid = int(entry_pid.get())
    try:
        while monitor_enabled:
            current_time = datetime.datetime.now().strftime("%Y-%m-%d %H:%M:%S")
            p = psutil.Process(pid)
            cpu_percent = p.cpu_percent(interval=1) / psutil.cpu_count()
            memory_mb = p.memory_full_info().rss / (1024 * 1024)
            memory_percent = p.memory_percent()
            output_text.set(
                f"Time: {current_time}\n"
                f"Process ID: {pid}\n"
                f"CPU Percent: {cpu_percent}\n"
                f"Memory (MB): {memory_mb}\n"
                f"Memory Percent: {memory_percent}"
            )
    except psutil.NoSuchProcess:
        messagebox.showerror("Error", f"Process with ID {pid} not found!")


def warning_thread():
    while warning_enabled:
        cpu_usage = psutil.cpu_percent(interval=1)
        if cpu_usage > 50:
            messagebox.showwarning("Warning", f"Cpu usage is above 50%: {cpu_usage}%")

        mem_usage = psutil.virtual_memory().percent
        if mem_usage > 50:
            messagebox.showwarning("Warning", f"Memory utilization is above 50%: {mem_usage}%")


def start_analysis():
    global monitor_enabled, warning_enabled
    monitor_enabled = True
    warning_enabled = True
    process_monitor_thread = threading.Thread(target=monitor_process_thread)
    process_monitor_thread.daemon = True
    process_monitor_thread.start()

    warning_thread = threading.Thread(target=warning_thread)
    warning_thread.daemon = True
    warning_thread.start()
```

```
def stop_analysis():
    global monitor_enabled, warning_enabled
    monitor_enabled = False
    warning_enabled = False



label_pid = tk.Label(app, text="Enter Process ID:")
label_pid.pack()

entry_pid = tk.Entry(app)
entry_pid.pack()

start_button = tk.Button(app, text="Start Analysis", command=start_analysis)
start_button.pack()

stop_button = tk.Button(app, text="Stop Analysis", command=stop_analysis)
stop_button.pack()

output_text = tk.StringVar()
output_label = tk.Label(app, textvariable=output_text)
output_label.pack()

monitor_enabled = False
warning_enabled = False

app.mainloop()
```

OUTPUT

OUTPUT:



```
Time: 2023-10-14 02:36:37
Process ID: 7372
CPU Percent: 0.0
Memory (MB): 70.19140625
Memory Percent: 0.43546337747906233
```

CONCLUSION

CONCLUSION:

The Python program presented here offers an effective and practical solution for monitoring the performance of a specified process and issuing system-wide warnings when necessary. Its user-friendly graphical interface makes it accessible to a wide range of users, even those without extensive technical expertise. The inclusion of multi-threading for real-time tracking ensures that users can closely monitor resource utilization and system performance without any significant overhead. With its responsive design, this program is versatile and adaptable, catering to various needs, whether it's keeping an eye on a critical process to ensure it's running smoothly or being alerted to potential system issues before they impact overall stability. This tool empowers users to maintain a well-functioning system with ease.

In addition to its practicality, the program's ability to provide real-time monitoring and system-wide warnings makes it a valuable asset for system administrators and anyone concerned about the efficient and stable operation of their system. By offering a user-friendly interface and robust functionality, it bridges the gap between technical and non-technical users, ensuring that everyone can benefit from its features.

REFERENCES

REFERENCES:

Here are some references that you can look up for more information on fitness trackers and their functionality:

1.https://psutil.readthedocs.io/en/latest/

2.https://ieeexplore.ieee.org/Xplore/home.jsp

3.https://realpython.com/intro-to-python-threading/