Project Report

On

“OptiWise”

Optimize your system wisely

Submitted by

1)Samruddhi Jivan Katole
2)Tejas Santosh Kale
3)Aditya Vijay Chavan

GitHub Link: https://github.com/SamruddhiKatole/ApexaiQ_Project

CONTENT:

S.R. NO.	TITLE	PAGE NO.
	ABSTRACT	03
1.	INTRODUCTION 1.1. Overview of Topic 1.2. Need of Project 1.3. Application/Use	04 04 04 04
2.	OBJECTIVES 2.1. Objectives 2.2. Technology Stack	05 05 06
3.	METHODOLOGY 3.1. Software 3.2. Structure 3.3. Block Diagram/ Flowchart	07 07 08 10
4.	RESULT 4.1. SCREENSHOTS	11 14
5.	FUTURE ENHANCEMENTS	15
6.	CONCLUSION	16
7.	REFERENCES	17

ABSTRACT

In a time when digital technologies rule the day, computer systems must operate effectively and dependably for both businesses and people. The goal of this project is to provide a comprehensive reporting and monitoring system that can provide deep insights into the performance parameters of a computer environment, which is a fundamental need. Real-time monitoring of critical factors including CPU, memory, disc space, and the existence of temporary files is the tool's main focus. This allows users to have an unmatched level of insight into the health of their system.

1. INTRODUCTION

1.1 Overview of Topic:

Robust tools are necessary in today's computing settings to efficiently monitor and control system health. The main goal of this project is to develop a flexible system that can measure important metrics continuously, giving system administrators and end users insightful information about the functionality of their systems.

1.2 Need of Project:

In the ever-changing world of technology, it is critical to keep computing systems healthy. Performance problems and unplanned outages can have serious repercussions. To meet this demand, the project offers a comprehensive monitoring and reporting solution that enables users to anticipate and resolve possible problems.

1.3 Application/ Use:

Numerous situations can benefit from the use of the system health monitoring tool, such as desktop maintenance, server management, and basic troubleshooting. It meets the various needs of the system.

2. OBJECTIVES

2.1. Objectives

The main objective of this project is to develop a system for monitoring various aspects of system health including CPU usage, memory usage, disk space utilization, and temporary file management. The key objectives are as follows:

1. Real-time Monitoring: Continuously monitor the system's CPU usage, memory usage, and disk space utilization in real-time.
2. Data Visualization: Visualize the collected data using charts to provide insights into system performance trends.
3. Temporary File Management: Identify and report temporary files present in the system.
4. Data Export: Export the collected data into CSV and Excel formats for further analysis.
5. Automation: Automate the process of generating reports and refreshing charts.
6. Web Interface: Provide a user-friendly web interface for viewing real-time system health data and generated reports.

2.2. Technology Stack:

- Python: Selected as the primary programming language due to its versatility, extensive libraries, and robust ecosystem support.
- Flask: Employed as the web framework for its lightweight nature, flexibility, and ease of integration with other components.
- Chart.js: Utilized for dynamic and interactive chart generation within the web interface. Its rich set of features enables the visualization of complex data in a user-friendly manner.
- Openpyxl: Chosen for its capability to work with Excel files, facilitating the seamless generation of detailed reports with extensive data manipulation capabilities.

3. METHODOLOGY

1. Software

The selection of software tools is a pivotal aspect contributing to the project's efficacy and success. The following tools have been meticulously chosen for their capabilities and suitability for various aspects of the project:

Python: Python serves as the cornerstone of the project, fulfilling multiple roles ranging from data collection and processing to server-side scripting. Its simplicity, versatility, and extensive library ecosystem make it an ideal choice for handling diverse requirements.

Flask: The project harnesses Flask, a lightweight and flexible web framework, to construct the user interface and manage HTTP requests. Flask's minimalist design and ease of integration align seamlessly with the project's objectives of simplicity and efficiency.

Chart.js: To enrich the user experience and facilitate intuitive data visualization, the project incorporates Chart.js, a JavaScript library renowned for its ability to create dynamic and interactive charts directly within the web interface. This empowers users to glean insights from system data with ease and precision.

Openpyxl: Leveraging Openpyxl, a Python library dedicated to working with Excel files, enables the generation of comprehensive reports encapsulating detailed system health metrics. Openpyxl's robust feature set facilitates seamless interaction with Excel spreadsheets, ensuring the creation of insightful and visually appealing reports.

Psutil: Critical system information retrieval is facilitated by Psutil, a Python library renowned for its efficiency in accessing and extracting essential system metrics. Psutil empowers the project with the capability to gather real-time data on system performance metrics, contributing to comprehensive monitoring and analysis.

2. Structure

1. Data Collection and Server-side Scripting:

- Python scripts are responsible for collecting system data in real-time, leveraging platform-specific APIs and libraries for accessing hardware and operating system metrics.
- Flask handles server-side scripting, providing endpoints for data retrieval, processing, and storage. Its lightweight nature ensures efficient handling of HTTP requests and responses.

2. Dynamic Web Interface:

- The web interface is designed using HTML, CSS, and JavaScript, with Flask serving as the backend to handle dynamic content generation and data integration.
- Chart.js is integrated into the interface to dynamically generate interactive charts based on the collected system data. Users can customize the visualization parameters and interact with the charts for deeper insights.

3. Excel Report Generation:

- Openpyxl library facilitates the creation of detailed reports in Excel format. Python scripts manipulate the collected data, generating customized reports with comprehensive insights into system performance and health.
- Reports can include various metrics, trends, and analysis summaries, providing users with actionable information for decision-making and optimization efforts.

By leveraging these technologies and architectural principles, the project aims to deliver a robust, scalable, and user-friendly system health monitoring solution, empowering users to proactively manage and optimize their system resources effectively.

3. Block Diagram/Flowchart

A meticulously crafted flowchart serves as the cornerstone of the system's architectural blueprint. This graphical representation elucidates the intricate processes underpinning data flow within the system health monitoring tool, offering stakeholders a comprehensive overview of the system's operational dynamics. Through visually intuitive delineation of data pathways, processing steps, and system interactions, the flowchart enhances comprehension and facilitates informed decision-making across all project stages.

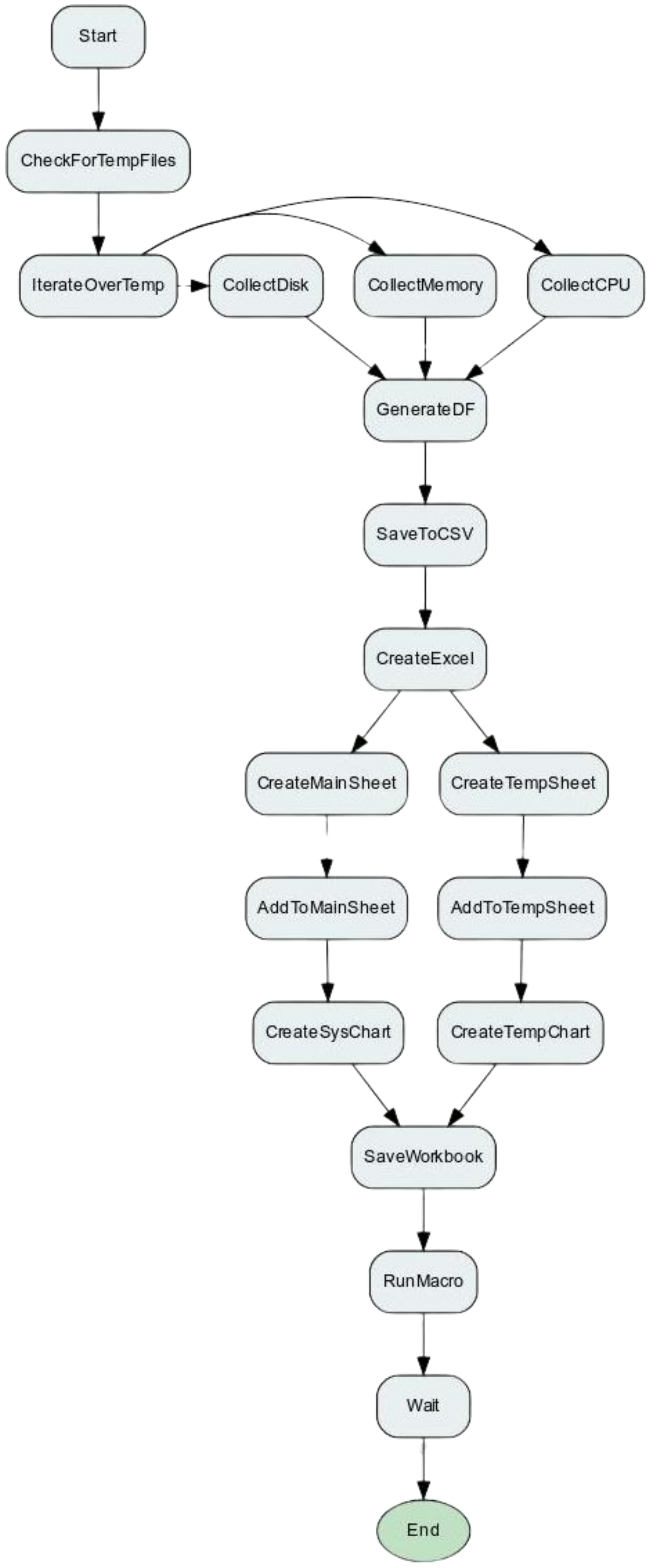
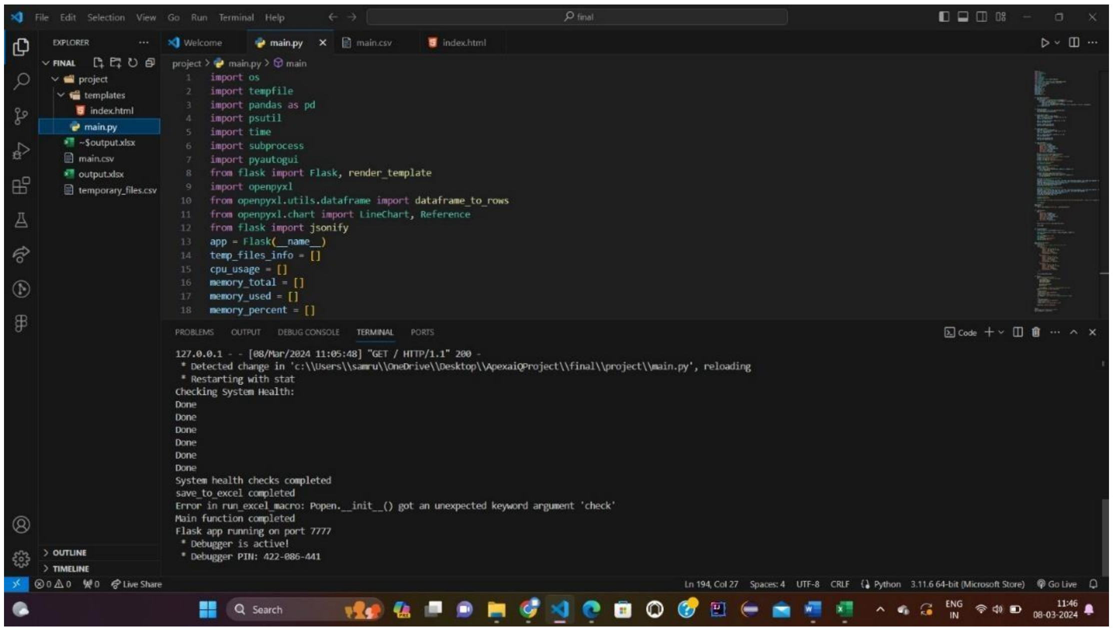


Fig. 3.3.1. Flow Chart

4. RESULT

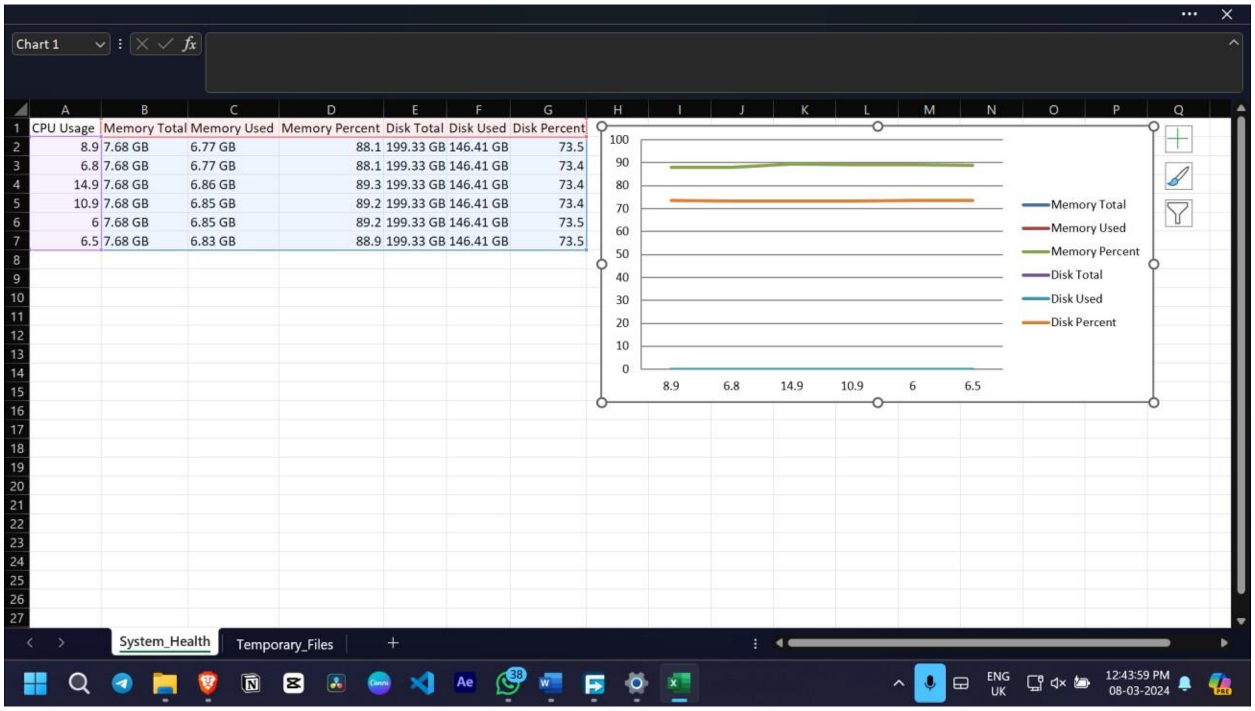
4.1. Screenshots



```
import os
import tempfile
import pandas as pd
import psutil
import time
import subprocess
import pyautogui
from flask import Flask, render_template
import openpyxl
from openpyxl.utils.dataframe import dataframe_to_rows
from openpyxl.chart import LineChart, Reference
from flask import jsonify
app = Flask(__name__)
temp_files_info = []
cpu_usage = []
memory_total = []
memory_used = []
memory_percent = []
```

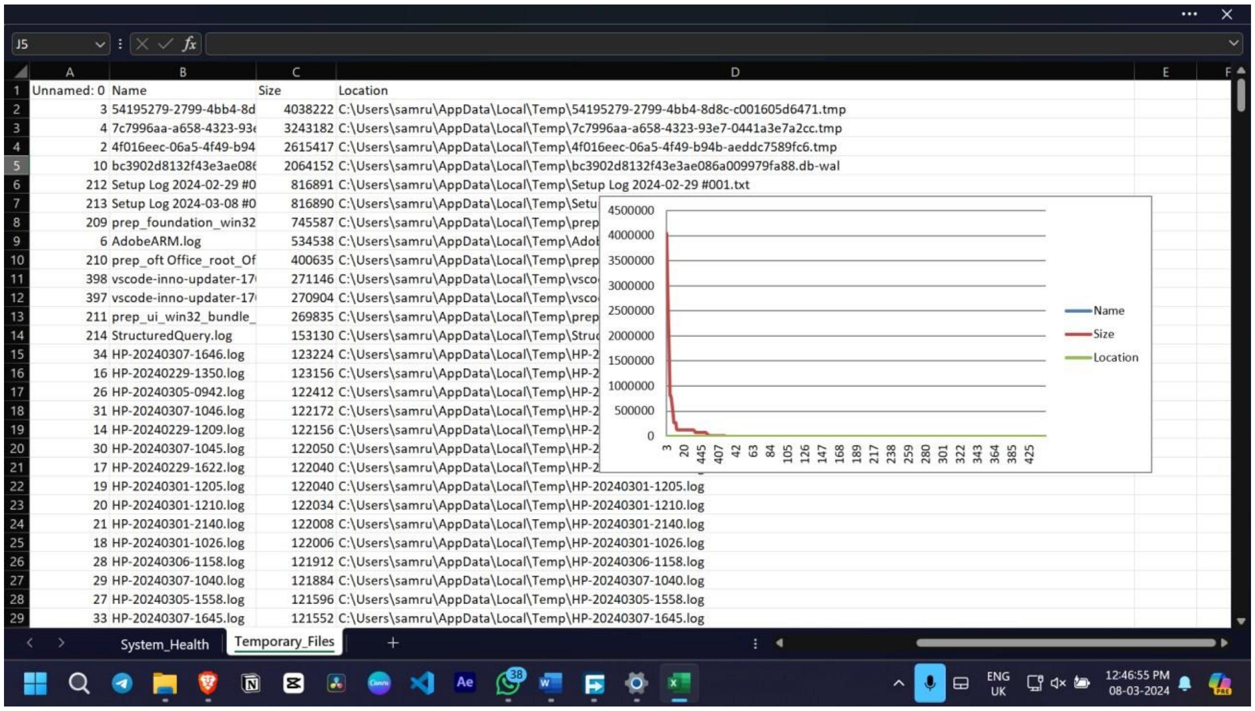
```
127.0.0.1 - - [08/Mar/2024 11:05:48] "GET / HTTP/1.1" 200 -
 * Detected change in 'c:\\Users\\samru\\OneDrive\\Desktop\\ApexaiQProject\\final\\project\\main.py', reloading
 * Restarting with stat
Checking System Health:
Done
Done
Done
Done
Done
Done
System health checks completed
save_to_excel completed
Error in run_excel_macro: Popen.__init__() got an unexpected keyword argument 'check'
Main function completed
Flask app running on port 7777
 * Debugger is active!
 * Debugger PIN: 422-086-441
```

Fig. 4.1.1 Code Execution of main.py file



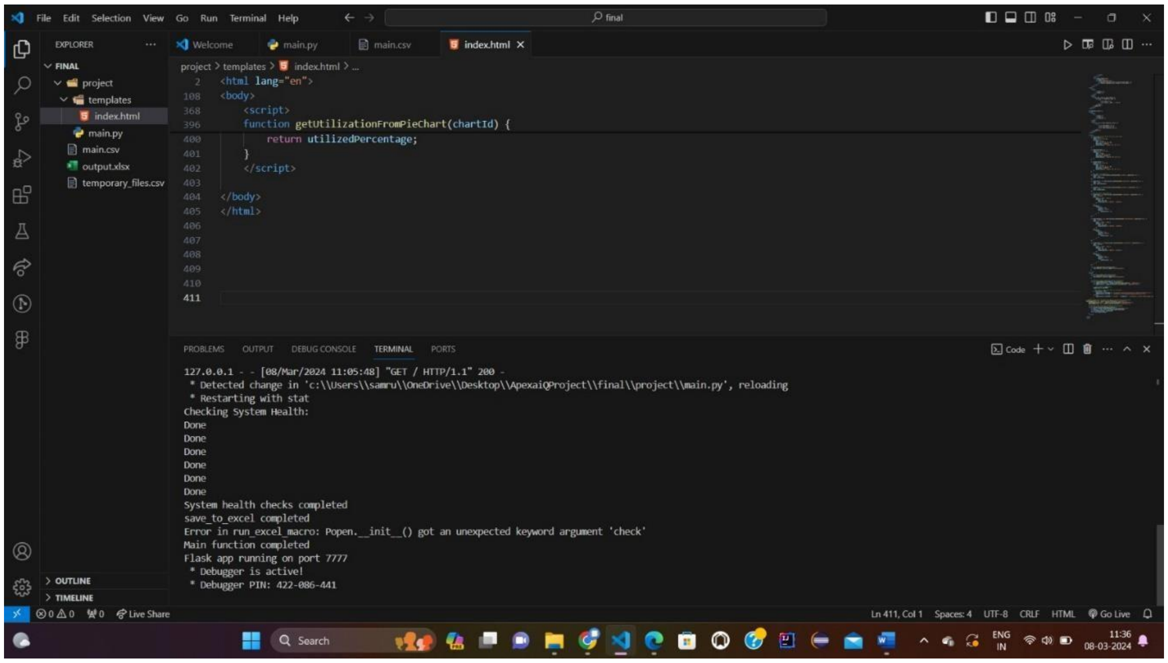
CPU Usage	Memory Total	Memory Used	Memory Percent	Disk Total	Disk Used	Disk Percent
8.9	7.68 GB	6.77 GB	88.1	199.33 GB	146.41 GB	73.5
6.8	7.68 GB	6.77 GB	88.1	199.33 GB	146.41 GB	73.4
14.9	7.68 GB	6.86 GB	89.3	199.33 GB	146.41 GB	73.4
10.9	7.68 GB	6.85 GB	89.2	199.33 GB	146.41 GB	73.4
6	7.68 GB	6.85 GB	89.2	199.33 GB	146.41 GB	73.5
6.5	7.68 GB	6.83 GB	88.9	199.33 GB	146.41 GB	73.5

Fig. 4.1.2.a Automatic opening of Excel workbook with System Health Sheet



	Unnamed: 0	Name	Size	Location
2	3	54195279-2799-4bb4-8d	4038222	C:\Users\samru\AppData\Local\Temp\54195279-2799-4bb4-8d8c-c001605d6471.tmp
3	4	7c7996aa-a658-4323-93	3243182	C:\Users\samru\AppData\Local\Temp\7c7996aa-a658-4323-93e7-0441a3e7a2cc.tmp
4	2	4f016eec-06a5-4f49-b94	2615417	C:\Users\samru\AppData\Local\Temp\4f016eec-06a5-4f49-b94b-aeddc7589fc6.tmp
5	10	bc3902d8132f43e3ae08	2064152	C:\Users\samru\AppData\Local\Temp\bc3902d8132f43e3ae086a009979fa88.db-wal
6	212	Setup Log 2024-02-29 #0	816891	C:\Users\samru\AppData\Local\Temp\Setup Log 2024-02-29 #001.txt
7	213	Setup Log 2024-03-08 #0	816890	C:\Users\samru\AppData\Local\Temp\Setu
8	209	prep_foundation_win32	745587	C:\Users\samru\AppData\Local\Temp\prep
9	6	AdobeARM.log	534538	C:\Users\samru\AppData\Local\Temp\Adol
10	210	prep_oft Office_root_Of	400635	C:\Users\samru\AppData\Local\Temp\prep
11	398	vscode-inno-updater-17	271146	C:\Users\samru\AppData\Local\Temp\vsco
12	397	vscode-inno-updater-17	270904	C:\Users\samru\AppData\Local\Temp\vsco
13	211	prep_ui_win32_bundle_	269835	C:\Users\samru\AppData\Local\Temp\prep
14	214	StructuredQuery.log	153130	C:\Users\samru\AppData\Local\Temp\Struc
15	34	HP-20240307-1646.log	123224	C:\Users\samru\AppData\Local\Temp\HP-2
16	16	HP-20240229-1350.log	123156	C:\Users\samru\AppData\Local\Temp\HP-2
17	26	HP-20240305-0942.log	122412	C:\Users\samru\AppData\Local\Temp\HP-2
18	31	HP-20240307-1046.log	122172	C:\Users\samru\AppData\Local\Temp\HP-2
19	14	HP-20240229-1209.log	122156	C:\Users\samru\AppData\Local\Temp\HP-2
20	30	HP-20240307-1045.log	122050	C:\Users\samru\AppData\Local\Temp\HP-2
21	17	HP-20240229-1622.log	122040	C:\Users\samru\AppData\Local\Temp\HP-2
22	19	HP-20240301-1205.log	122040	C:\Users\samru\AppData\Local\Temp\HP-20240301-1205.log
23	20	HP-20240301-1210.log	122034	C:\Users\samru\AppData\Local\Temp\HP-20240301-1210.log
24	21	HP-20240301-2140.log	122008	C:\Users\samru\AppData\Local\Temp\HP-20240301-2140.log
25	18	HP-20240301-1026.log	122006	C:\Users\samru\AppData\Local\Temp\HP-20240301-1026.log
26	28	HP-20240306-1158.log	121912	C:\Users\samru\AppData\Local\Temp\HP-20240306-1158.log
27	29	HP-20240307-1040.log	121884	C:\Users\samru\AppData\Local\Temp\HP-20240307-1040.log
28	27	HP-20240305-1558.log	121596	C:\Users\samru\AppData\Local\Temp\HP-20240305-1558.log
29	33	HP-20240307-1645.log	121552	C:\Users\samru\AppData\Local\Temp\HP-20240307-1645.log

Fig. 4.1.2.b Automatic opening of Excel workbook with Temporary Files Sheet


File Edit Selection View Go Run Terminal Help
final
EXPLORER
FINAL
project
templates
index.html
main.py
main.csv
output.xlsx
temporary_files.csv
Welcome
main.py
main.csv
index.html
project > templates > index.html > ...
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
OUTLINE
TIMELINE
Ln 411, Col 1 Spaces: 4 UTF-8 CRLF HTML Go Live

```
<html lang="en">
<body>
    <script>
    function getUtilizationFromPieChart(chartId) {
        return utilizedPercentage;
    }
    </script>

</body>
</html>
```

```
127.0.0.1 - - [08/Mar/2024 11:05:48] "GET / HTTP/1.1" 200 -
 * Detected change in 'c:\\Users\\samru\\OneDrive\\Desktop\\ApexaiQProject\\final\\project\\main.py', reloading
 * Restarting with stat
Checking System Health:
Done
Done
Done
Done
Done
Done
System health checks completed
save_to_excel completed
Error in run_excel_macro: Popen.__init__() got an unexpected keyword argument 'check'
Main function completed
Flask app running on port 7777
 * Debugger is active!
 * Debugger PIN: 422-086-441
```

Fig. 4.1.3 Code Execution of index.html file

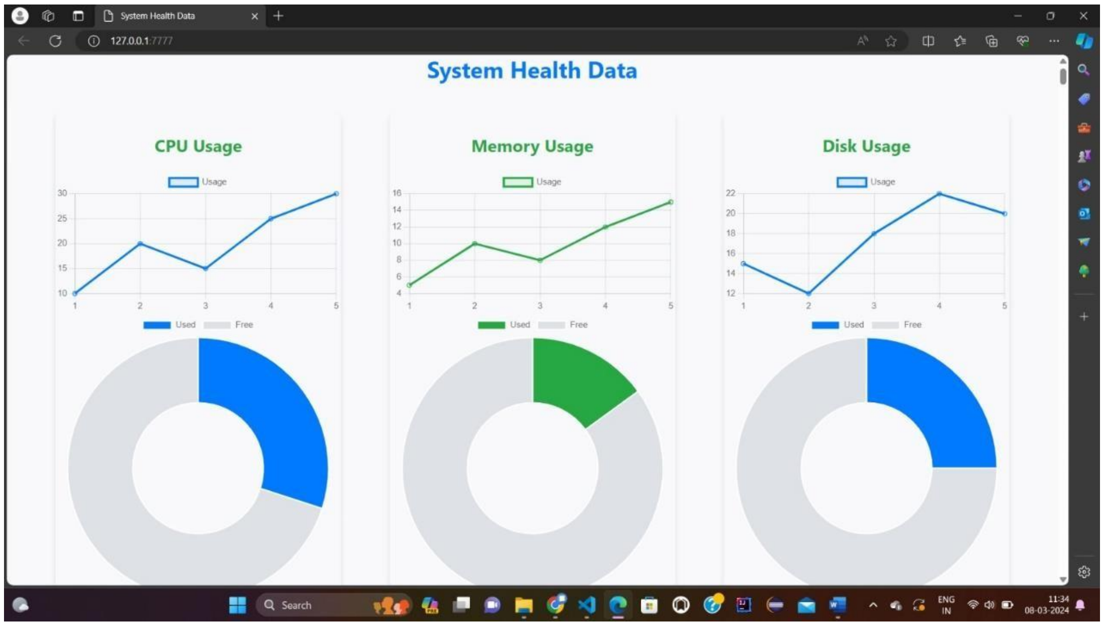
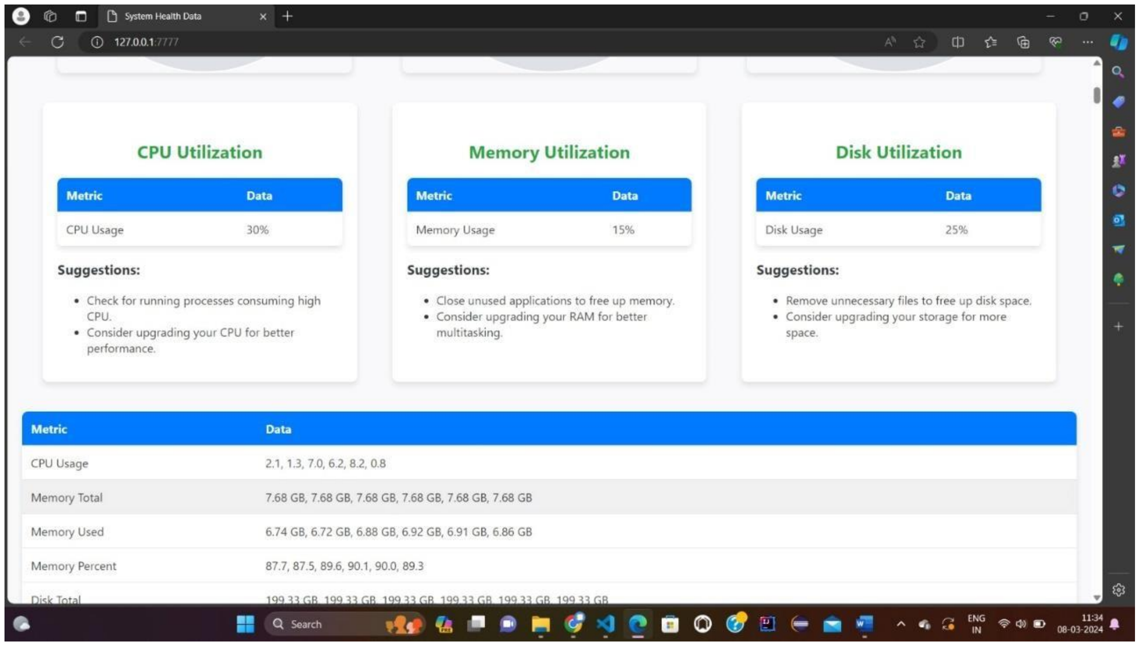


Fig. 4.1.4 Dashboard: Charts representing CPU, Memory and Disk Usage



CPU Utilization

Metric	Data
CPU Usage	30%

Suggestions:

- Check for running processes consuming high CPU.
- Consider upgrading your CPU for better performance.

Memory Utilization

Metric	Data
Memory Usage	15%

Suggestions:

- Close unused applications to free up memory.
- Consider upgrading your RAM for better multitasking.

Disk Utilization

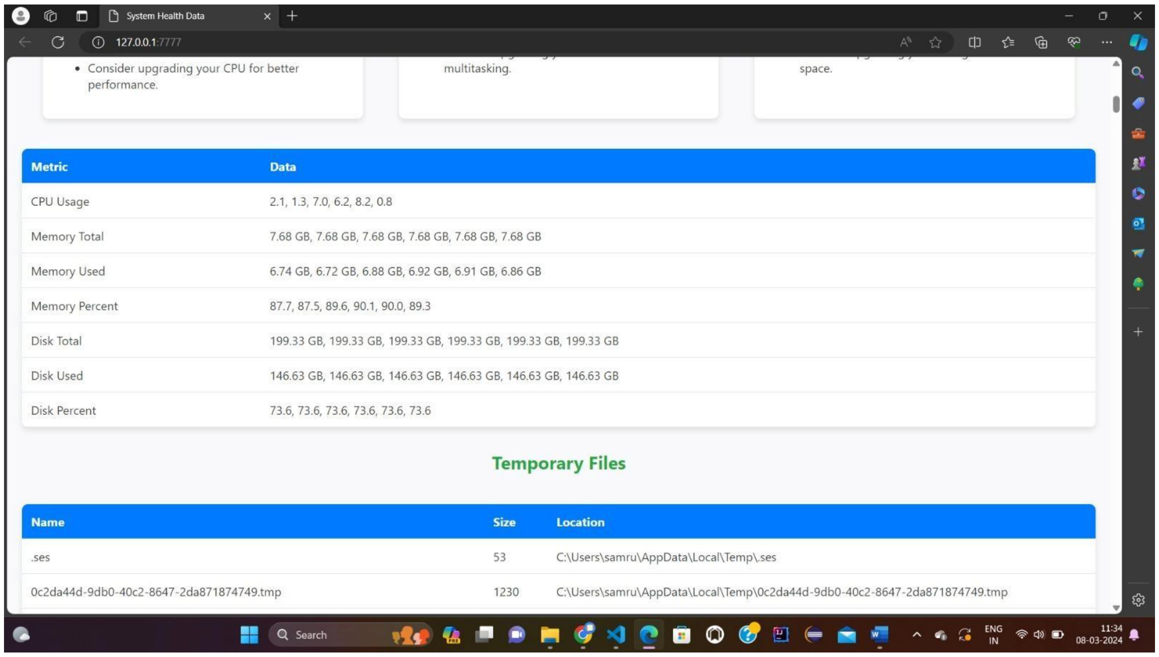
Metric	Data
Disk Usage	25%

Suggestions:

- Remove unnecessary files to free up disk space.
- Consider upgrading your storage for more space.

Metric	Data
CPU Usage	2.1, 1.3, 7.0, 6.2, 8.2, 0.8
Memory Total	7.68 GB, 7.68 GB, 7.68 GB, 7.68 GB, 7.68 GB, 7.68 GB
Memory Used	6.74 GB, 6.72 GB, 6.88 GB, 6.92 GB, 6.91 GB, 6.86 GB
Memory Percent	87.7, 87.5, 89.6, 90.1, 90.0, 89.3
Disk Total	199.33 GB, 199.33 GB, 199.33 GB, 199.33 GB, 199.33 GB, 199.33 GB

Fig. 4.1.5 Dashboard: Metrics and Suggestions on CPU, Memory & Disk Utilization



- Consider upgrading your CPU for better performance.

multitasking.

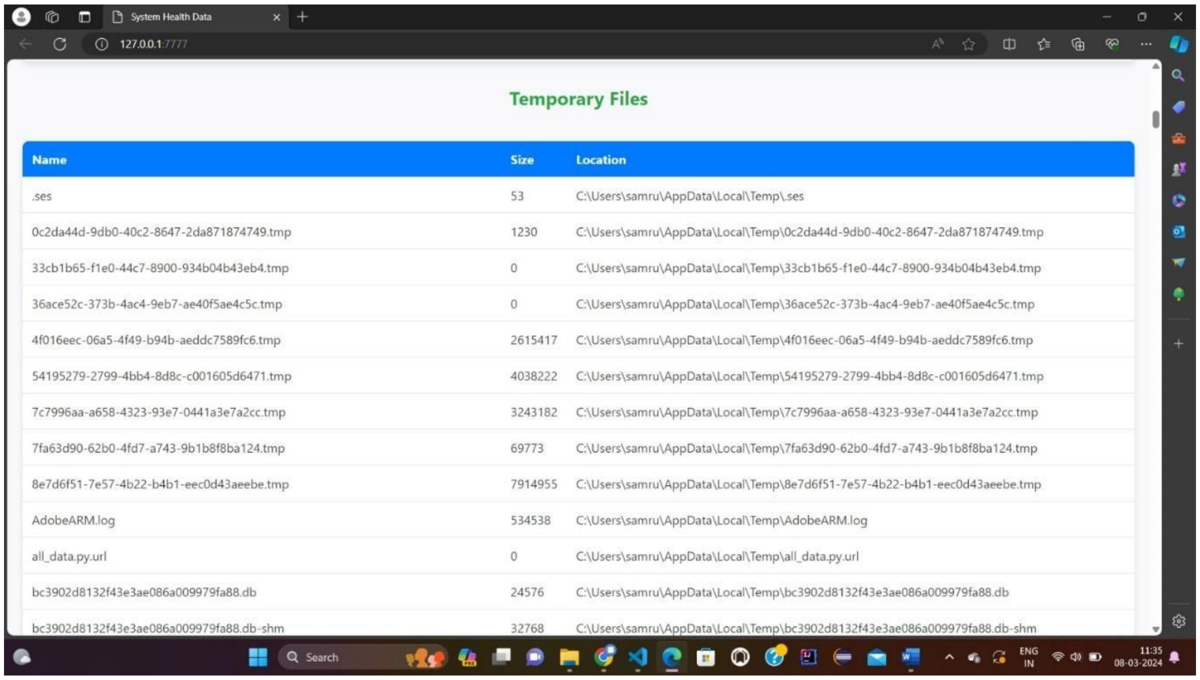
space.

Metric	Data
CPU Usage	2.1, 1.3, 7.0, 6.2, 8.2, 0.8
Memory Total	7.68 GB, 7.68 GB, 7.68 GB, 7.68 GB, 7.68 GB, 7.68 GB
Memory Used	6.74 GB, 6.72 GB, 6.88 GB, 6.92 GB, 6.91 GB, 6.86 GB
Memory Percent	87.7, 87.5, 89.6, 90.1, 90.0, 89.3
Disk Total	199.33 GB, 199.33 GB, 199.33 GB, 199.33 GB, 199.33 GB, 199.33 GB
Disk Used	146.63 GB, 146.63 GB, 146.63 GB, 146.63 GB, 146.63 GB, 146.63 GB
Disk Percent	73.6, 73.6, 73.6, 73.6, 73.6, 73.6

Temporary Files

Name	Size	Location
.ses	53	C:\Users\samru\AppData\Local\Temp\.ses
0c2da44d-9db0-40c2-8647-2da871874749.tmp	1230	C:\Users\samru\AppData\Local\Temp\0c2da44d-9db0-40c2-8647-2da871874749.tmp

Fig. 4.1.6 Dashboard: All metrics represented with the data collected



Temporary Files

Name	Size	Location
.ses	53	C:\Users\samru\AppData\Local\Temp\.ses
0c2da44d-9db0-40c2-8647-2da871874749.tmp	1230	C:\Users\samru\AppData\Local\Temp\0c2da44d-9db0-40c2-8647-2da871874749.tmp
33cb1b65-f1e0-44c7-8900-934b04b43eb4.tmp	0	C:\Users\samru\AppData\Local\Temp\33cb1b65-f1e0-44c7-8900-934b04b43eb4.tmp
36ace52c-373b-4ac4-9eb7-ae40f5ae4c5c.tmp	0	C:\Users\samru\AppData\Local\Temp\36ace52c-373b-4ac4-9eb7-ae40f5ae4c5c.tmp
4f016eec-06a5-4f49-b94b-aeddc7589fc6.tmp	2615417	C:\Users\samru\AppData\Local\Temp\4f016eec-06a5-4f49-b94b-aeddc7589fc6.tmp
54195279-2799-4bb4-8d8c-c001605d6471.tmp	4038222	C:\Users\samru\AppData\Local\Temp\54195279-2799-4bb4-8d8c-c001605d6471.tmp
7c7996aa-a658-4323-93e7-0441a3e7a2cc.tmp	3243182	C:\Users\samru\AppData\Local\Temp\7c7996aa-a658-4323-93e7-0441a3e7a2cc.tmp
7fa63d90-62b0-4fd7-a743-9b1b8f8ba124.tmp	69773	C:\Users\samru\AppData\Local\Temp\7fa63d90-62b0-4fd7-a743-9b1b8f8ba124.tmp
8e7d6f51-7e57-4b22-b4b1-eec0d43aeebe.tmp	7914955	C:\Users\samru\AppData\Local\Temp\8e7d6f51-7e57-4b22-b4b1-eec0d43aeebe.tmp
AdobeARM.log	534538	C:\Users\samru\AppData\Local\Temp\AdobeARM.log
all_data.py.url	0	C:\Users\samru\AppData\Local\Temp\all_data.py.url
bc3902d8132f43e3ae086a009979fa88.db	24576	C:\Users\samru\AppData\Local\Temp\bc3902d8132f43e3ae086a009979fa88.db
bc3902d8132f43e3ae086a009979fa88.db-shm	32768	C:\Users\samru\AppData\Local\Temp\bc3902d8132f43e3ae086a009979fa88.db-shm

Fig. 4.1.7 Dashboard: All temporary file names, sizes, and their locations

5. FUTURE ENHANCEMENTS

1. User Authentication and Authorization: Implement user authentication and authorization to restrict access to system health data and reports.

2. Email Alerts: Integrate email alerts to notify system administrators of critical system health issues or thresholds being exceeded.

3. Customizable Thresholds: Allow users to set customizable thresholds for CPU usage, memory usage, and disk space utilization, triggering alerts when these thresholds are exceeded.

4. Historical Data Analysis: Implement functionality to store historical system health data and provide tools for analyzing trends over time.

5. Improved Charting Options: Enhance charting capabilities by adding support for different types of charts and customization options.

6. Integration with Monitoring Tools: Integrate the system with popular monitoring tools such as Prometheus and Grafana for advanced monitoring and visualization capabilities.

6. CONCLUSION

The system health monitoring and reporting tool represents a significant leap forward in the realm of system administration and performance optimization. By providing a real-time, user-friendly interface and comprehensive reports, this tool equips administrators and end-users with the means to proactively manage and maintain the health of their computing systems.

In conclusion, the project successfully achieves its objectives, offering a powerful solution for system monitoring and reporting that is adaptable to diverse computing environments.

GitHub Link: https://github.com/SamruddhiKatole/ApexaiQ_Project

7. REFERENCES

1. https://umeey.medium.com/system-monitoring-made-easy-with-pythons-psutil-library-4b9add95a443#:~:text=Python's%20psutil%20library%20is%20a,few%20lines%20of%20Python%20code

2. Flask: https://flask.palletsprojects.com/en/3.0.x/

3. Python: https://docs.python.org/3/

4. psutil: https://pypi.org/project/psutil/

5. tempfile: https://docs.python.org/3/library/tempfile.html

6. subprocess: https://docs.python.org/3/library/subprocess.html