**BASAVARAJESWARI GROUP OF INSTITUTIONS**

**Ballari Institute of Technology & Management**

**AUTONOMOUS INSTITUTE UNDER VISVESVARAYA TECHNOLOGICAL UNIVERSITYJNANA SANGAMA, BELAGAVI 590018**

**INTERNSHIP**

**Report On**

# LOCAL SERVER MONITORING TOOL

Submitted in partial fulfilment of the requirements for the award of degree of

**Bachelor of Engineering**

**In**

**COMPUTER SCIENCE AND ENGINEERING**

**(DATA SCIENCE)**

## Submitted by

**J Nandita**

**3BR23CD031**

## Internship Carried Out By

**EZ TRAININGS & TECHNOLOGIES PVT.LTD HYDERABAD**

**Internal Guide External Guide**

**V P ANUSHYA VISHAL KUMAR**

**Asst.prof,CSE-DS Technical Trainer**

**KAMALAPADU VARSHA**

**Teaching Asst.CSE-DS**

### BALLARI INSTITUTE OF TECHNOLOGY & MANAGEMENT

NACC Accredited Institution\*

**(Recognized by Govt. of Karnataka, approved by AICTE, New Delhi & Affiliated to Visvesvaraya Technological University, Belagavi)**

**"Jnana Gangotri" Campus, No.873/2, Ballari-Hospet Road, Allipur, Ballar1-583 104 (Karnataka) (India) Ph: 08392 – 237100 / 237190, Fax: 08392 – 237197**

**2024-2025**

**BASAVARAJESWARI GROUP OF INSTITUTIONS**

## BALLARI INSTITUTE OF TECHNOLOGY & MANAGEMENT

**Autonomous institute under VISVESVARAYA TECHNOLOGICAL UNIVERSITYJNANA SANGAMA,**

**BELAGAVI 590018**

NACC Accredited Institution\*

**(Recognized by Govt. of Karnataka, approved by AICTE, New Delhi & Affiliated to** 

**Visvesvaraya Technological University, Belagavi)**

### "JnanaGangotri"Campus,No.873/2,Ballari-HospetRoad,Allipur,

**Ballar1-583 104 (Karnataka)(India)**

**Ph: 08392 – 237100 / 237190, Fax: 08392 –237197**

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

**(DATA SCIENCE)**

# CERTIFICATE

This is to certify that the Internship entitled **“ LOCAL SERVER MONITORING TOOL”** has been successfully completed by **J NANDITA** bearing USN **3BR23CD031** a bonafide student of Ballari Institute of Technology and Management,

Ballari. For the partial fulfillment of the requirementsfor the **Bachelor’s Degree in Computer**

**Science and Engineering-Data science** of the VISVESVARAYA TECHNOLOGICAL UNIVERSITY, Belagavi during the academic year 2024-2025.

**Signature of Internship Signature of HOD**

**Co-ordinator**

**V P ANUSHYA DR.AARADHANA**

**Asst.prof,CSE-DS Prof. and HOD(CSE)**

**KAMALAPADU VARSHA**

**Asst. prof,CSE-DS**

**DECLARATION**

I, **J Nandita,** second year student of Computer

Science and Engineering, Ballari Institute of Technology, Ballari, declare that Internship entitled **LOCAL SERVER MONITORING TOOL** is a part of Internship Training successfully carried out by **EZ TECHNOLOGIES & TRAININGS PVT.LTD ,Hyderabad** at “**BITM,BALLARI”.** This report is submitted in partial fulfillment of the requirements for the award of the degree, Bachelor of Engineering in Computer Science and Engineering(Data Science) of the Visvesvaraya Technological University, Belagavi.

**Date : 28/09/2024 Signature of the Student**

**Place : BALLARI**

**ACKNOWLEDGEMENT**

The satisfactions that a company the successful completion of my internship on “ LOCAL SERVER MONITORING TOOL ” would be incomplete without the mention of people who made it possible, whose noble gesture, affection, guidance, encouragement and support crowned my efforts with success. It is my privilege to express my gratitude and respect to all those who inspired me in the completion of my internship.

I am grateful to our respective coordinator **“V P Anushya (Asst.prof,CSE-DS) , Kamalapadu Varsha (Teaching.Asst.CSE-DS)”** for his noble gesture, support co-ordination and valuable suggestions given to me in the completion of Internship.

I also thank **DR. AARADHANA,** H.O.D. Department of **Computer science and engineering-Data Science** for extending all her valuable support and encouragement.

**Table of Contents**

| **Chapter No.** | **Chapter Name** | **Page No.** |
| --- | --- | --- |
| **1** | **Day to day activity(student diary extract)** | **01-02** |
| **2** | **Company Profile** | **03** |
| **3** | **Abstract** | **04** |
| **4** | **Introduction of the project** | **05** |
| **5** | **Description** | **05-07** |
| **6** | **Algorithm** | **08** |
| **7** | **Flowchart** | **09** |
| **8** | **Source Code** | **10-14** |
| **9** | **Output** | **14-16** |
| **10** | **Conclusion** | **17** |
| **11** | **References** | **17** |

**DAY TO DAY ACTIVITIES**



**Internship Program on Python for BE-3rd Sem students**

**From 9th September to 28th September (During 3rd semester vacations).**

**Student Name: J NANDITA USN No: 3BR23CD031 Branch: CSE-DS**

| **INDEX PAGE** | | | |
| --- | --- | --- | --- |
| **Day** | **Date** | **Content Covered** | **Signature of the** |
| **faculty in-charge** |
| **1** | **09.09.24** | **Introduction to Python, Setup & Installation, First Python Program, Variables, Data Types,**  **and Basic I/O** |  |
| **2** | **10.09.24** | **Control Structures: If-else, Loops, Functions and Modules** |  |
| **3** | **11.09.24** | **Lists, Tuples, and Dictionaries, File Handling** |  |
| **4** | **12.09.24** | **Exception Handling, Practice exercises on Python basics** |  |
| **5** | **13.09.24** | **Introduction to OOP, Classes, and Objects** |  |
| **6** | **14.09.24** | **Inheritance, Polymorphism, and Encapsulation** |  |
| **7** | **15.09.24** | **Abstract Classes and Interfaces** |  |
| **8** | **17.09.24** | **Practice exercises on OOP concepts** |  |
| **9** | **18.09.24** | **Introduction to DSA, Arrays, and Linked Lists** |  |
| **10** | **19.09.24** | **Stacks and Queues** |  |
| **11** | **20.09.24** | **Trees and Graphs** |  |
| **12** | **21.09.24** | **Searching and Sorting Algorithms** |  |
| **13** | **22.09.24** | **Project Building & Presentations** |  |
| **14** | **24.09.24** | **Project Building & Presentations** |  |
| **15** | **25.09.24** | **Project Building & Presentations** |  |
| **16** | **26.09.24** | **Project Building & Presentations** |  |
| **17** | **27.09.24** | **Project Building & Presentations** |  |
| **18** | **28.09.24** | **Project Building & Presentations** |  |

**COMPANY PROFILE**

**Company Name: EZ Trainings and Technologies Pvt. Ltd.**

**Introduction:**

EZ Trainings and Technologies Pvt. Ltd. is a dynamic and innovative organization dedicated to providing comprehensive training solutions and expert development services. Established with a vision to bridge the gap between academic learning and industry requirements, we specialize in college trainings for students, focusing on preparing them for successful placements. Additionally, we excel in undertaking development projects, leveraging cutting-edge technologies to bring ideas to life.

**Mission:**

Our mission is to empower the next generation of professionals by imparting relevant skills and knowledge through specialized training programs. We strive to be a catalyst in the career growth of students and contribute to the technological advancement of businesses through our development projects.

**Services:**

**College Trainings:**

* Tailored training programs designed to enhance the employability of students.
* Industry-aligned curriculum covering technical and soft skills.
* Placement assistance and career guidance.

**Development Projects:**

* End-to-end development services, from ideation to execution.
* Expertise in diverse technologies and frameworks.
* Custom solutions to meet specific business needs.

**Locations:** Hyderabad | Delhi NCR

At EZ Trainings and Technologies Pvt. Ltd., we believe in transforming potential into excellence

Abstract

The presented code implements a simple server monitoring tool in Python, enabling users to manage virtual servers effectively. The system is structured around several classes, including `Server`, `Server Manager`, `Server Health Monitor`, `Alert System`, and `Local Server Monitoring Tool`.

The `Server` class represents individual servers, encapsulating attributes such as server ID, name, and IP address. The `Server Manager` class provides functionalities to create, read, update, and delete server instances, maintaining a dictionary of servers to track their states.

The `Server Health Monitor` class periodically checks server health metrics, specifically CPU and memory usage, simulating resource consumption with random values. The `Alert System` generates alerts if usage exceeds predefined thresholds, ensuring that users are notified of potential issues.

The `Local Server Monitoring Tool` class serves as the main interface, allowing users to interact with the server management system through a command-line interface. Users can add servers, display their details, modify attributes, delete servers, and monitor health metrics.

The `main` function orchestrates user interactions, providing a looped menu for seamless management. This tool is designed to simplify server administration tasks, making it suitable for educational purposes and small-scale server management applications.

1. Introduction:

The Server Monitoring Tool is a Python application designed to help IT professionals manage and monitor server resources effectively. It allows users to create, read, update, and delete server instances, ensuring efficient management of server configurations.

Additionally, the tool includes a health monitoring feature that simulates CPU and memory usage statistics, providing alerts when resource usage exceeds specified thresholds. With a user-friendly command-line interface, this tool simplifies server management, helping teams maintain optimal performance and reliability.

2.Problem Statement:

POC: • CRUD: Server details.

\* monitor\_server\_health(server\_id): Regularly check server health metrics such as CPU and memory usage.

\* alert\_system\_downtime(alert\_details): Alert users when a server goes down.

Code: To solve the given problem of creating a Local Server Monitoring Tool Proof of Concept (POC), I'll outline a Python solution designed using Object-Oriented Programming (OOP) and Data Structures and Algorithms (DSA). This solution will consist of classes defining the server entities, methods for CRUD operations, and additional functions for monitoring server health and alerting on system downtime.

3.Objectives:

The objectives of the provided code are:

* Server Management:

Create a server: Add a new server to the system with a unique ID, name, and IP address.

Read/Show server details: View the details of a specific server based on its ID.

Update server: Modify the name or IP address of an existing server.

Delete server: Remove a server from the system by its ID.

* Health Monitoring:

Monitor server health: Simulate CPU and memory usage of servers using random values.

Set alerts: Raise alerts when CPU or memory usage exceeds certain thresholds (80% by default).

* Menu-driven Interface:

The code offers a simple text-based menu to perform the above operations interactively, allowing users to add, update, show, delete servers, or check health status.

The system will keep running until the user decides to exit.

* Integration:

The LocalServerMonitoringTool class integrates all these functionalities, allowing seamless management and monitoring of servers.

4.Technology And The Tools Used:

A real-life example of a Local Server Monitoring Tool would be a system used by IT departments or network administrators in companies to manage and monitor their internal servers. Here’s how it works:

Server Details (CRUD Operations):

IT administrators use the tool to maintain an inventory of all local servers. They can create new server entries when adding hardware, read or view details of existing servers (like IP address, operating system, and configuration), update server information (such as changing the server’s name or upgrading hardware), or delete servers that are no longer in use.

Monitor Server Health:

The tool regularly checks the health of servers by monitoring key metrics like CPU usage, memory usage, and disk space. For example, if a company has a web server running an e-commerce platform, the tool will check if the server is functioning well and not overloaded.

If the CPU usage is consistently above a certain threshold (e.g., 85%), the system could flag this as a potential issue.

Alert System Downtime:

If a server goes down, the monitoring tool sends alerts to the IT staff via email or text message. For example, if a file server fails during business hours, the tool would notify the IT team immediately so they can take action to resolve the issue.

This feature helps minimize downtime and ensures that critical services remain available, such as customer databases or internal communication systems.

5.Methodology :

5.1. Project Plan:

Objective: Develop a server monitoring tool to manage servers (CRUD operations) and monitor CPU/memory usage with alerts.

Key Features:

1. Server Management (Add, Update, Read, Delete).

2. Health Monitoring:

Simulate CPU/Memory usage.

Trigger alerts if thresholds (80% by default) are exceeded.

3. Menu-driven Interface for user interaction.

Phases:

1. Phase 1: Server Management (CRUD)

Implement classes Server and ServerManager to manage server data.

Add functionality to create, read, update, and delete servers.

2. Phase 2: Health Monitoring

Add ServerHealthMonitor to simulate CPU/Memory usage.

Implement AlertSystem to trigger alerts for high usage.

3. Phase 3: Command-Line Interface (CLI)

Create a menu for user interaction.

Integrate server management and health monitoring functions.

Continuously loop until the user chooses to exit.

Timeline:

Phase 1: Setup Server Management - 1 day

Phase 2: Implement Health Monitoring - 1 day

Phase 3: Build CLI Interface - 1 day

Testing & Debugging - 1 day

Total Time: 4 days

Future Enhancements:

Add persistent storage (e.g., database).

Real-time health monitoring.

Multi-server monitoring & alerts.

5.2.Algorithm:

1. Initialization:

Instantiate the LocalServerMonitoringTool which internally uses ServerManager to manage servers.

2. Display Menu:

Show the user a menu with the following options:

* Add Server
* Show Server
* Update Server
* Delete Server
* Check Server Health
* Exit

3. User Input Handling:

Accept the user’s choice and perform the following actions based on the input:

Add Server: Accept server\_id, name, and ip\_address from the user, and call add\_server().

Show Server: Accept server\_id and display details using show\_server().

Update Server: Accept server\_id, optional new name, and optional new IP to update server details using modify\_server().

Delete Server: Accept server\_id to remove the server using remove\_server().

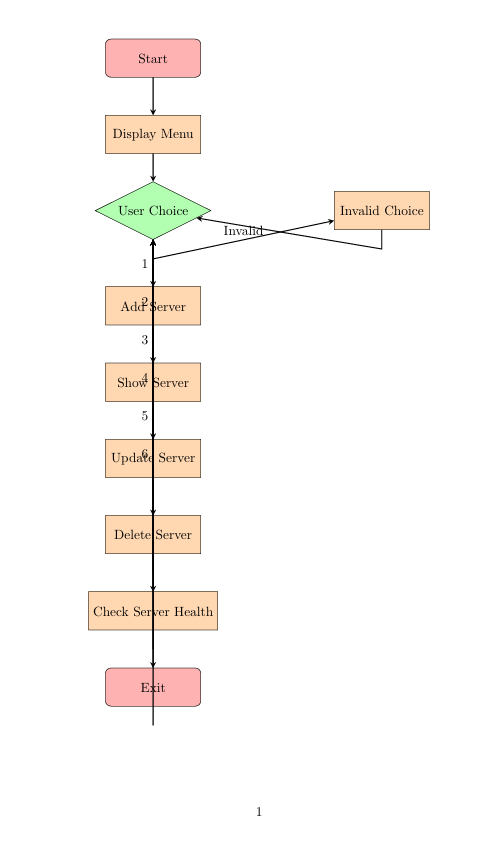
Check Server Health: Randomly generate CPU and memory usage, and raise alerts if they exceed defined thresholds using check\_health().

Exit: End the program loop.

4. Exit Condition:

If the user selects the exit option, terminate the program. Otherwise, loop back to the menu for another operation.

5.3 Flowchart



6.Source Code:

import random

class Server:

def \_init\_(self, server\_id, name, ip\_address):

self.server\_id = server\_id

self.name = name

self.ip\_address = ip\_address

def \_str\_(self):

return f"Server(ID: {self.server\_id}, Name: {self.name}, IP: {self.ip\_address})"

class ServerManager:

def \_init\_(self):

self.servers = {}

def create\_server(self, server\_id, name, ip\_address):

if server\_id in self.servers:

print("Server already exists!")

else:

new\_server = Server(server\_id, name, ip\_address)

self.servers[server\_id] = new\_server

print(f"Server {name} created.")

def read\_server(self, server\_id):

if server\_id in self.servers:

print(self.servers[server\_id])

else:

print("Server not found.")

def update\_server(self, server\_id, new\_name=None, new\_ip=None):

if server\_id in self.servers:

server = self.servers[server\_id]

if new\_name:

server.name = new\_name

if new\_ip:

server.ip\_address = new\_ip

print(f"Server {server\_id} updated.")

else:

print("Server not found.")

def delete\_server(self, server\_id):

if server\_id in self.servers:

del self.servers[server\_id]

print(f"Server {server\_id} deleted.")

else:

print("Server not found.")

class ServerHealthMonitor:

@staticmethod

def monitor\_health():

cpu\_usage = random.randint(0, 100)

memory\_usage = random.randint(0, 100)

print(f"CPU Usage: {cpu\_usage}%")

print(f"Memory Usage: {memory\_usage}%")

return cpu\_usage, memory\_usage

class AlertSystem:

@staticmethod

def alert(cpu\_threshold=80, memory\_threshold=80):

cpu\_usage, memory\_usage = ServerHealthMonitor.monitor\_health()

if cpu\_usage > cpu\_threshold:

print(f"Alert! CPU usage is over {cpu\_threshold}%")

if memory\_usage > memory\_threshold:

print(f"Alert! Memory usage is over {memory\_threshold}%")

class LocalServerMonitoringTool:

def \_init\_(self):

self.manager = ServerManager()

def add\_server(self, server\_id, name, ip\_address):

self.manager.create\_server(server\_id, name, ip\_address)

def show\_server(self, server\_id):

self.manager.read\_server(server\_id)

def modify\_server(self, server\_id, new\_name=None, new\_ip=None):

self.manager.update\_server(server\_id, new\_name, new\_ip)

def remove\_server(self, server\_id):

self.manager.delete\_server(server\_id)

def check\_health(self):

AlertSystem.alert()

def main():

tool = LocalServerMonitoringTool()

while True:

print("\nServer Monitoring Tool")

print("1. Add Server")

print("2. Show Server")

print("3. Update Server")

print("4. Delete Server")

print("5. Check Server Health")

print("6. Exit")

choice = input("Enter your choice (1-6): ")

if choice == '1':

server\_id = int(input("Enter Server ID: "))

name = input("Enter Server Name: ")

ip\_address = input("Enter IP Address: ")

tool.add\_server(server\_id, name, ip\_address)

elif choice == '2':

server\_id = int(input("Enter Server ID to show: "))

tool.show\_server(server\_id)

elif choice == '3':

server\_id = int(input("Enter Server ID to update: "))

new\_name = input("Enter new Server Name (leave blank to keep current): ")

new\_ip = input("Enter new IP Address (leave blank to keep current): ")

tool.modify\_server(server\_id, new\_name if new\_name else None, new\_ip if new\_ip else None)

elif choice == '4':

server\_id = int(input("Enter Server ID to delete: "))

tool.remove\_server(server\_id)

elif choice == '5':

tool.check\_health()

elif choice == '6':

print("Exiting program.")

break

else:

print("Invalid choice. Please try again.")

if \_name\_ == "\_main\_":

main()

7.Outputs:

Server Monitoring Tool

1. Add Server

2. Show Server

3. Update Server

4. Delete Server

5. Check Server Health

6. Exit

Enter your choice (1-6): 1

Enter Server ID: 123

Enter Server Name: s1

Enter IP Address: 450.23.12

---------------------------

Server s1 created.

---------------------------

Server Monitoring Tool

1. Add Server

2. Show Server

3. Update Server

4. Delete Server

5. Check Server Health

6. Exit

Enter your choice (1-6): 2

Enter Server ID to show: 123

Server(ID: 123, Name: s1, IP: 450.23.12)

Server Monitoring Tool

1. Add Server

2. Show Server

3. Update Server

4. Delete Server

5. Check Server Health

6. Exit

Enter your choice (1-6): 3

Enter Server ID to update: 123

Enter new Server Name (leave blank to keep current): s2

Enter new IP Address (leave blank to keep current): 456.23.546

Server 123 updated.

Server Monitoring Tool

1. Add Server

2. Show Server

3. Update Server

4. Delete Server

5. Check Server Health

6. Exit

Enter your choice (1-6): 5

CPU Usage: 86%

Memory Usage: 11%

Alert! CPU usage is over 80%

Server Monitoring Tool

1. Add Server

2. Show Server

3. Update Server

4. Delete Server

5. Check Server Health

6. Exit

Enter your choice (1-6): 4

Enter Server ID to delete: 123

Server 123 deleted.

Server Monitoring Tool

1. Add Server

2. Show Server

3. Update Server

4. Delete Server

5. Check Server Health

6. Exit

Enter your choice (1-6): 6

Exiting program.

8.Conclusion:

The provided code implements a Local Server Monitoring Tool that allows users to manage servers and monitor their health. Users can add, view, update, and delete servers, as well as check system health (CPU and memory usage). It integrates basic server management with random system health monitoring and alerts based on threshold values. This tool provides a simple and interactive command-line interface for managing and monitoring servers in a local environment.

9.Future Enhancement:

Future Enhancements:

1. Persistent Storage: Implement a database (e.g., SQLite) to store server details, so data persists across sessions.

2. Advanced Health Monitoring: Integrate real-time monitoring using system libraries (e.g., psutil) for more accurate CPU and memory metrics.

3. Server Grouping: Allow grouping of servers based on their function or location for easier management.

4. Health Log History: Maintain a history log of server health checks and alerts.

5. User Authentication: Add user authentication to restrict access to server management functions.

6. Web or GUI Interface: Develop a web or graphical user interface for easier interaction, replacing the command-line interface.

References:

<https://chatgpt.com/>

<https://www.geeksforgeeks.org/python-script-to-monitor-network-connection-and-saving-into-log-file/>