MAHARAJA INSTITUTE OF TECHNOLOGY MYSORE

Belawadi, SrirangapatnaTq, Mandya-571477

DEPARTMENT OF CSE (Artificial Intelligence)

Assignment – Project Report

2025-26

Subject Name: Cloud Computing

Subject Code: M23BCS503

Semester: 5th Sem

Submitted by: Team No:

Sl. No.	Student Name	USN.	CO's Mapping					Total	Scaled to
			CO1	CO2	CO3	CO4	CO5		
1	Yashwanth Gowda KS	4MH23CA062							

Verified and Approved by:

Faculty Name: MJ Yogesh

Signature:

Date: 9/12/2025

Project GitHub Repository:

Link: https://github.com/YASHWANTHGOWDAKS5/CC_Monitoring_proj

PROJECT OVERVIEW

Introduction:

Modern cloud applications run on dynamic, scalable infrastructure where performance depends on several real-time metrics such as CPU usage, memory consumption, disk utilization, network throughput, database latency, and request rate.

To ensure reliability and cost-efficiency, continuous monitoring and performance tuning are essential. This project demonstrates a cloud monitoring and tuning system developed using a web-based dashboard. The application displays live metric graphs, generates tuning recommendations, and simulates cloud-provider behaviour (AWS, Azure, and GCP).

Key Features of the System:

✓ Real-time metric visualization

Live charts update every 3 seconds to visualize CPU, memory, disk, and network utilization.

✓ Cloud-provider selection

Users can switch between AWS, Azure, and GCP simulations.

✓ Automatic tuning recommendations

The system analyses the metrics and gives specific suggestions like scaling, caching, upgrading VM types, enabling autoscaling, etc.

✓ Manual metric input mode

Users can enter custom metric values and instantly get tuning advice for the selected provider.

✓ Status classification

The system categorizes overall health as:

Normal Warning Critical

✓ Provider-specific logic

AWS, Azure, and GCP each have their own tuning logic, aligned with real-world cloud best practices.

✓ API-based design

The dashboard communicates with the Flask backend using JSON REST calls, similar to how real cloud monitoring tools work.

Procedure to Connect the Web App With Real Cloud Metrics:

The monitoring dashboard is designed so it can easily connect to real cloud platforms such as AWS, Azure, and GCP monitoring services. Each cloud provider offers official monitoring services that allow applications to access performance information like CPU usage, memory trends, network activity, and system health. The dashboard can receive real metrics from cloud resources and display them in the charts and cards already built into the interface and get the tuning suggestions to make it efficient and well utilized.

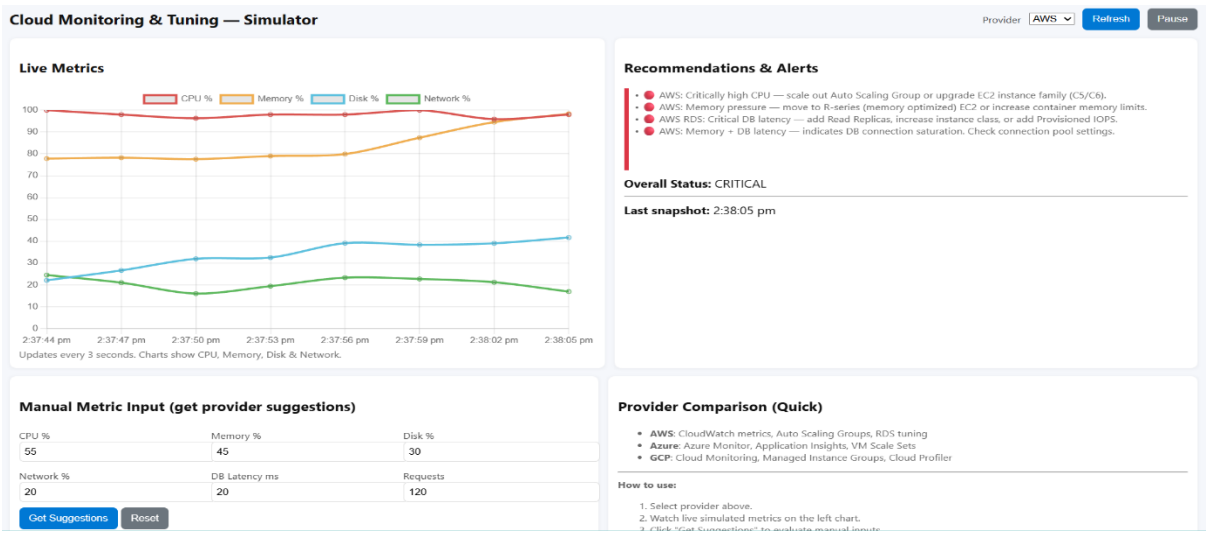
System Workflow Overview:

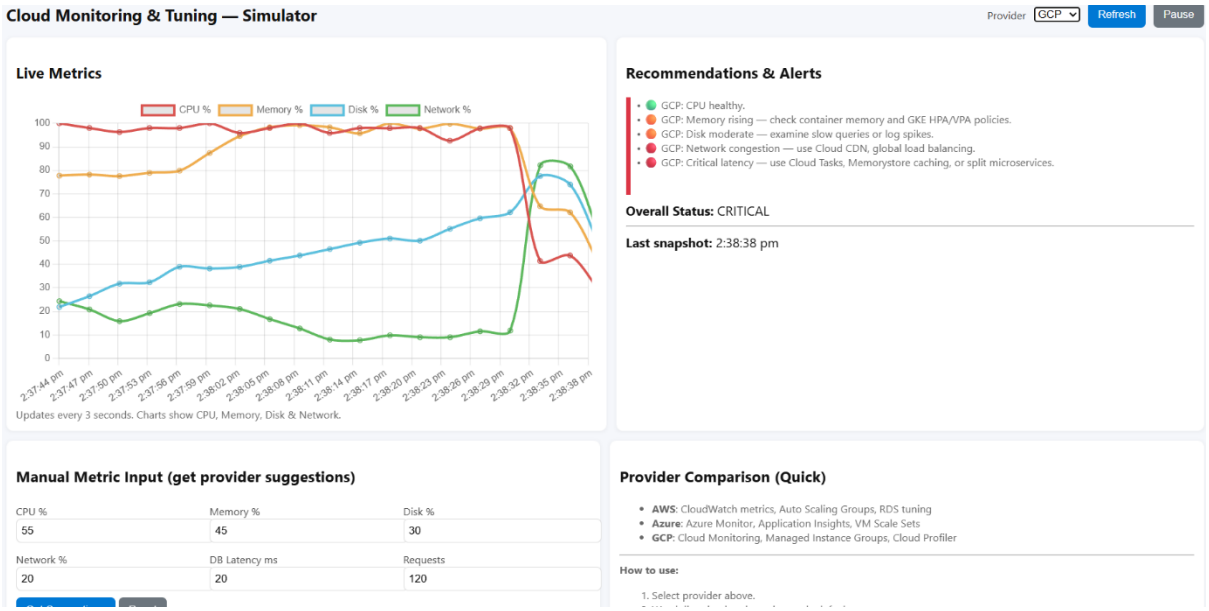
1. The user selects the cloud provider on the dashboard.
2. The frontend sends a request to the backend API.
3. The backend provides the latest performance metrics.
4. The charts and metric cards refresh automatically.
5. The user gets a clear view of the application's behaviour and Tuning Suggestions based on that cloud platform.

Tech Stack:

1. **HTML**
2. **JavaScript**
3. **Python (Flask)**
4. **Chart.js**
5. **CSS**

Outputs


Cloud Monitoring & Tuning — Simulator
Provider AWS
Refresh
Pause
Live Metrics
CPU %
Memory %
Disk %
Network %
Updates every 3 seconds. Charts show CPU, Memory, Disk & Network.
Recommendations & Alerts
AWS: Critically high CPU — scale out Auto Scaling Group or upgrade EC2 instance family (C5/C6).
AWS: Memory pressure — move to R-series (memory optimized) EC2 or increase container memory limits.
AWS RDS: Critical DB latency — add Read Replicas, increase instance class, or add Provisioned IOPS.
AWS: Memory + DB latency — indicates DB connection saturation. Check connection pool settings.
Overall Status: CRITICAL
Last snapshot: 2:38:05 pm
Manual Metric Input (get provider suggestions)
CPU %
Memory %
Disk %
55
45
30
Network %
DB Latency ms
Requests
20
20
120
Get Suggestions
Reset
Provider Comparison (Quick)
AWS: CloudWatch metrics, Auto Scaling Groups, RDS tuning
Azure: Azure Monitor, Application Insights, VM Scale Sets
GCP: Cloud Monitoring, Managed Instance Groups, Cloud Profiler
How to use:
1. Select provider above.
2. Watch live simulated metrics on the left chart.


Cloud Monitoring & Tuning — Simulator
Provider GCP
Refresh
Pause
Live Metrics
CPU %
Memory %
Disk %
Network %
Updates every 3 seconds. Charts show CPU, Memory, Disk & Network.
Recommendations & Alerts
GCP: CPU healthy.
GCP: Memory rising — check container memory and GKE HPA/VPA policies.
GCP: Disk moderate — examine slow queries or log spikes.
GCP: Network congestion — use Cloud CDN, global load balancing.
GCP: Critical latency — use Cloud Tasks, Memorystore caching, or split microservices.
Overall Status: CRITICAL
Last snapshot: 2:38:38 pm
Manual Metric Input (get provider suggestions)
CPU %
Memory %
Disk %
55
45
30
Network %
DB Latency ms
Requests
20
20
120
Provider Comparison (Quick)
AWS: CloudWatch metrics, Auto Scaling Groups, RDS tuning
Azure: Azure Monitor, Application Insights, VM Scale Sets
GCP: Cloud Monitoring, Managed Instance Groups, Cloud Profiler
How to use:
1. Select provider above.

Monitoring and Tuning in AWS

Monitoring and tuning are important to keep cloud applications healthy, fast, and reliable in AWS. Amazon provides several easy-to-use tools that help track performance and improve the efficiency of cloud resources.

1. Monitoring in AWS:

Amazon CloudWatch

CloudWatch is the main monitoring service in AWS. It shows real-time information such as:

- CPU usage
- Memory usage
- Disk activity
- Network traffic
- Application logs

2. Tuning in AWS:

Auto Scaling: Automatically adds or removes EC2 instances based on load.
This helps maintain performance during high traffic and save cost when traffic is low.
Right-Sizing: Choosing the correct instance type based on actual usage.
This avoids both over-provisioning and under-performance.
Load Balancing: AWS Load Balancers distribute traffic across multiple servers so no single server becomes overloaded.
Database Optimization:
Using features like:

- Read replicas
- Better indexing
- Proper storage types (EBS, SSD)

This improves speed and reduces delays.
Caching: Services like ElastiCache can store frequently used data to reduce database load and speed up applications.

Monitoring and Tuning in Azure

Monitoring and tuning in Azure help applications run smoothly and stay cost-effective. Azure provides built-in tools to track performance and improve resource usage.

1. Monitoring in Azure

Azure Monitor

Shows important performance data such as:

- CPU and memory usage
- Disk performance
- Network activity
- Application errors and logs

Alerts and dashboards help detect issues early.

2. Tuning Techniques in Azure

Autoscale: Adds or removes VM or App Service instances based on load to maintain performance and save cost.

Choosing the Right VM Size: Selecting the correct VM SKU based on workload patterns to avoid both underperformance and overuse of resources.

Azure Load Balancer / Application Gateway: Distributes traffic evenly so individual servers do not overload.

Optimizing Azure SQL / Cosmos DB

- Creating indexes
- Adjusting throughput
- Using the right storage tier
  Improves query speed and reduces latency.

Using Azure Cache for Redis

Stores frequently accessed data to reduce load on databases and improve app responsiveness.

Monitoring and Tuning in Google Cloud Platform

GCP provides monitoring and optimization tools that help applications run faster and handle changes in traffic efficiently.

1. Monitoring in GCP

Google Cloud Monitoring

Tracks important metrics like:

- CPU and memory usage
- Disk operations
- Network traffic
- Service logs and errors

Custom dashboards and alerts help maintain visibility.

2. Tuning Techniques in GCP

Instance Autoscaling: Automatically increases or reduces Compute Engine or GKE nodes based on CPU or request load.

Right-Sizing Recommendations: GCP suggests better VM sizes based on actual usage to avoid unnecessary cost or performance loss.

Cloud Load Balancing: Balances traffic globally across multiple regions to improve speed and reduce failures.

Database Performance Tuning (Cloud SQL / Firestore / BigQuery)

- Adding indexes
- Using read replicas
- Choosing proper storage options

  These help lower query time and improve throughput.

Using Cloud Memorystore: Provides caching to reduce database load and speed up responses for frequently used data.

Conclusion

Monitoring and tuning are essential for maintaining the performance, reliability, and efficiency of cloud applications. Through this project, a unified monitoring dashboard was developed to visualize important metrics like CPU usage, memory trends, network activity, and system health. The system provides a clear and easy interface for understanding how applications behave under different conditions and can be extended to connect with real cloud platforms such as AWS, Azure, and GCP.

By studying the monitoring and tuning techniques used in major cloud providers, it becomes clear that every platform focuses on the same goals: ensuring smooth performance, reducing delays, preventing failures, and optimizing resource usage. Features like autoscaling, load balancing, caching, and right-sizing help applications handle varying workloads while keeping costs under control.

Overall, this project helps demonstrate how cloud monitoring works in real-world environments and highlights the importance of continuously observing and improving cloud applications for better performance and reliability.

References

1. Amazon Web Services – Official Documentation
2. Microsoft Azure – Official Documentation
3. Google Cloud Platform – Official Documentation