D. Y. PATIL COLLEGE OF ENGINEERING & TECHNOLOGY, KOLHAPUR

(An Autonomous Institute)

DEPARTMENT OF CSE (DATA SCIENCE)

A

Project-III Report

on

"Proactive IT Support System"

Submitted by

Name	Roll No.
Mr. Tejas Vaibhav Kevate	11
Mr. Afif Sharif Sayyad	13
Mr. Amey Uday Yarnalkar	15
Mr. Basavaraj Sawanta Mali	67

Under the guidance of

Mr. S. K. Patil

Final Year B. Tech. CSE (Data Science)

Academic Year 2024-25

D. Y. PATIL COLLEGE OF ENGINEERING & TECHNOLOGY, KOLHAPUR

(An Autonomous Institute)

DEPARTMENT OF CSE (DATA SCIENCE)

CERTIFICATE

This is to certify that,

Roll No.	Unique ID	Student Name	Exam Seat No.
11	EN21130679	Mr. Tejas Vaibhav Kevate	1684
13	EN21149993	Mr. Afif Sharif Sayyad	1558
15	EN21160358	Mr. Amey Uday Yarnalkar	1563
67	EN21223398	Mr. Basawaraj Savanta Mali	1584

have successfully completed the Project-III work entitled,

"Proactive IT Support System"

In partial fulfilment for the curriculum of **Final Year B. Tech. CSE (Data Science)**. This is the record of their work carried out during academic year 2024-2025.

Date: 28/12/2024 **Place:** Kolhapur

Mr. S. K. Patil	**Prof. DR. G. V. Patil**	**Prof. DR. S. D. Chede**
Project Guide	HOD Data Science	Principal

External Examiner

INDEX

INTRODUCTION

In today's rapidly evolving digital landscape, IT systems are the backbone of organizational operations. The complexity and interconnectivity of these systems have increased, making them more susceptible to various issues ranging from hardware failures to software malfunctions. Traditional IT support models are primarily reactive, addressing problems only after they occur, often resulting in significant downtime, productivity loss, and increased operational costs.

The Proactive IT Support System is designed to address these challenges by shifting the focus from reactive problem-solving to proactive issue prevention. By leveraging advanced AI/ML technologies, the system can continuously monitor IT environments, analyse data for patterns and anomalies, and predict potential issues before they escalate. This proactive approach not only reduces downtime but also optimizes resource allocation and enhances the overall efficiency of IT operations.

The proposed system aims to provide a comprehensive solution that integrates real-time monitoring, predictive analytics, automated alert generation, and support ticket management into a single platform. This holistic approach ensures that organizations can maintain high levels of system availability and reliability, ultimately leading to improved business continuity and user satisfaction.

- **Problem Statement:**

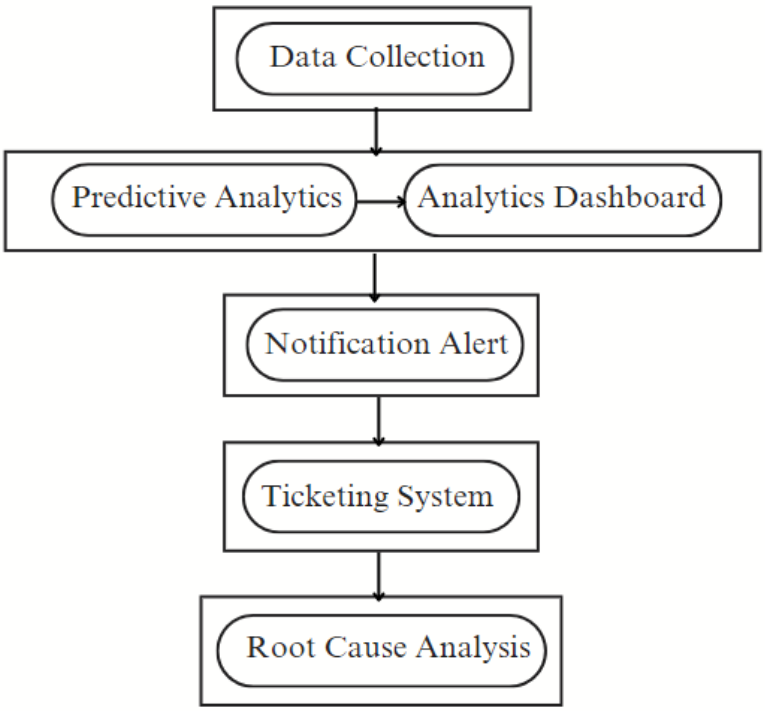
" To develop a Proactive IT Support System using predictive analytics, real-time monitoring, and automated root cause analysis to reduce system downtime, time-consuming helpdesk ticket resolution process, and an enhanced user experience for the end users."

- **Objectives:**

1. To implement a predictive analytics system to forecast potential IT issues using system the logs and the performance metrics.
2. To design an integrated dashboard for real-time display of the system metrics and the alerts.
3. To develop the user-friendly ticket management system for tracking IT support tickets raising process.
4. To set up a multi-channel notification system for real-time alerts on critical the events and predicted the issues.
5. To introduce the basic root cause analysis using predefined rules for the diagnosing recurring IT problems.

- **Design Approach**

The system is designed to integrate multiple modules, each addressing a specific aspect of IT support:



1. **Data Collection Module:**

- Functionality: Gathers information from system logs, network traffic, and performance metrics. Predictive analytics and real-time monitoring leverage input information from this data.
- Components: Log Aggregator, Performance Monitor, Network Traffic Analyzer.

2. Predictive Analytics Module:

- Use case: Detects potential IT issues by accessing the collected data via machine learning models. Predicts errors and generates alerts.
- Components: Machine Learning Model, Prediction Engine, Alert Generator.

3. Integrated Dashboard Module:

- Utility: It shows you what your system is doing in real-time and alerts/notifications. Central console for IT staff to observe system health.
- Components: Real-Time Metrics Display, Alert Notification Center, User Interface.

4. Ticket Management Module:

- Functionality: Create, update, and view support tickets. Supports the IT issue lifecycle from reporting to resolution.
- Components: Create Ticket Interface, Tracking System to create tickets, Status Update.

5. Notification Module:

- Features: Notifies IT staff about critical events using in-app alerts and email.
- Components: Notification Engine, Multi-Channel Delivery System, Event Handler.

6. Root Cause Analysis Module:

- Functionality: Provides rudimentary root cause diagnosis based on preconfigured heuristics or rules. Helps identify the root causes of repeat IT issues.
- Components: Rule-Based Analyzer, Heuristic Engine, Report Generator.

- ## Methodology:

The development process adheres to a structured, systematic approach to ensure the successful delivery of the Proactive IT Support System. Each phase focuses on leveraging technical best practices for robust and efficient system development.

1. Requirement Analysis:

- Detailed assessment of functional and non-functional requirements to align system goals with organizational needs.
- Includes stakeholder consultations and analysis of system specifications like predictive analytics, real-time monitoring, and dashboard integration.
-

2. System Design:

- Creation of a modular architecture defining interactions between key components like Data Collection, Predictive Analytics, and Notification Modules.
- Design of data flow diagrams (DFDs), use case diagrams, and database schemas to establish system workflows and data dependencies.

3. Implementation:

- Development of individual modules using modern technologies like Python for machine learning and React for the user interface.
- Integration of APIs and libraries for real-time data streaming, monitoring, and alert generation.

4. Testing:

- Execution of unit tests for individual components to validate logic, functionality, and performance.
- Conducting integration testing to ensure seamless communication between modules and adherence to requirements.

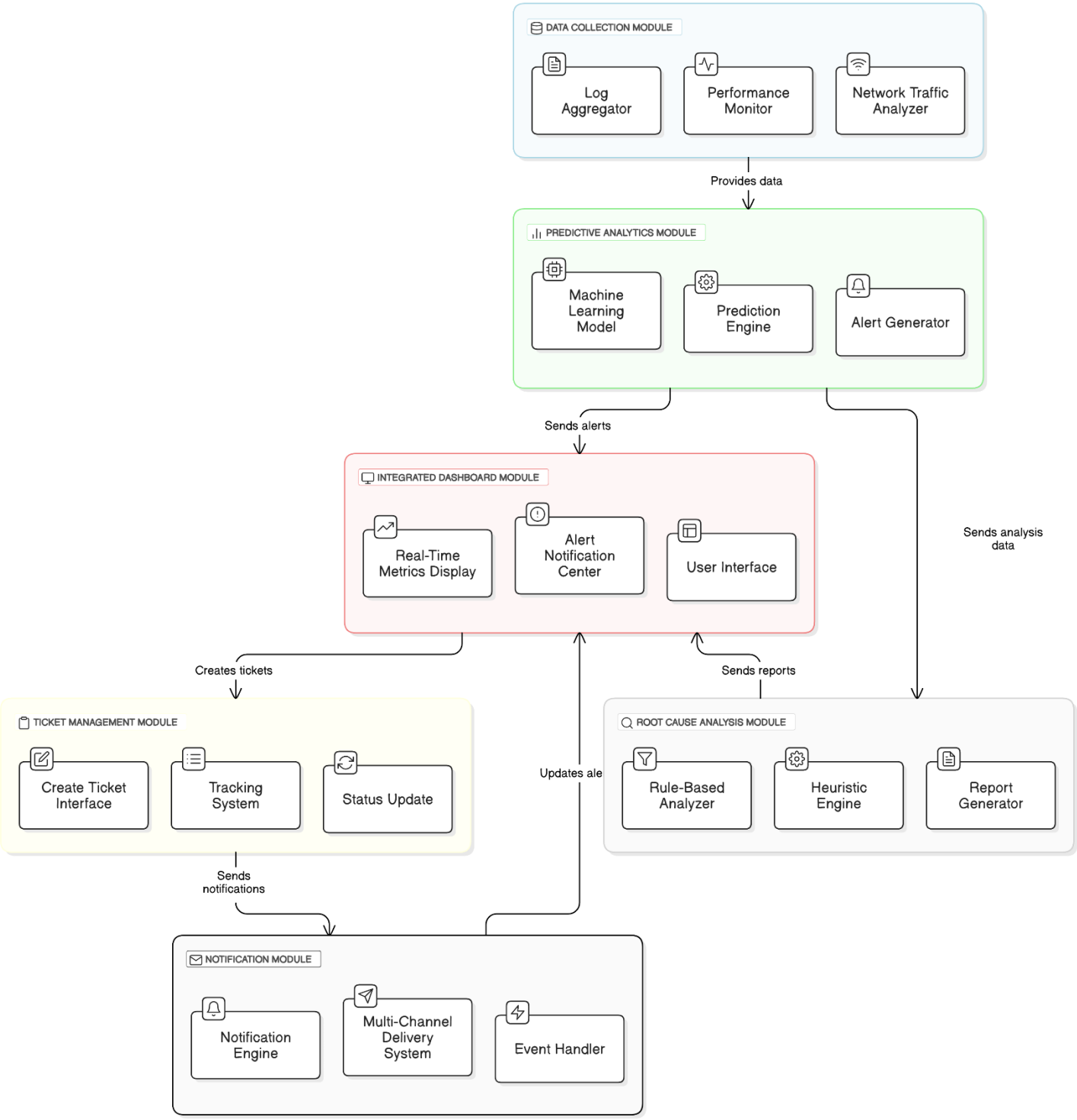
5. Deployment:

- System deployment in a controlled environment with secure infrastructure configurations, including database setup and server configurations.
- Implementation of monitoring tools to track performance and error rates post-deployment.

6. Evaluation:

- Performance evaluation through stress testing, uptime analysis, and feedback from end-users.
- Iterative refinement of system components to optimize efficiency, scalability, and user experience.

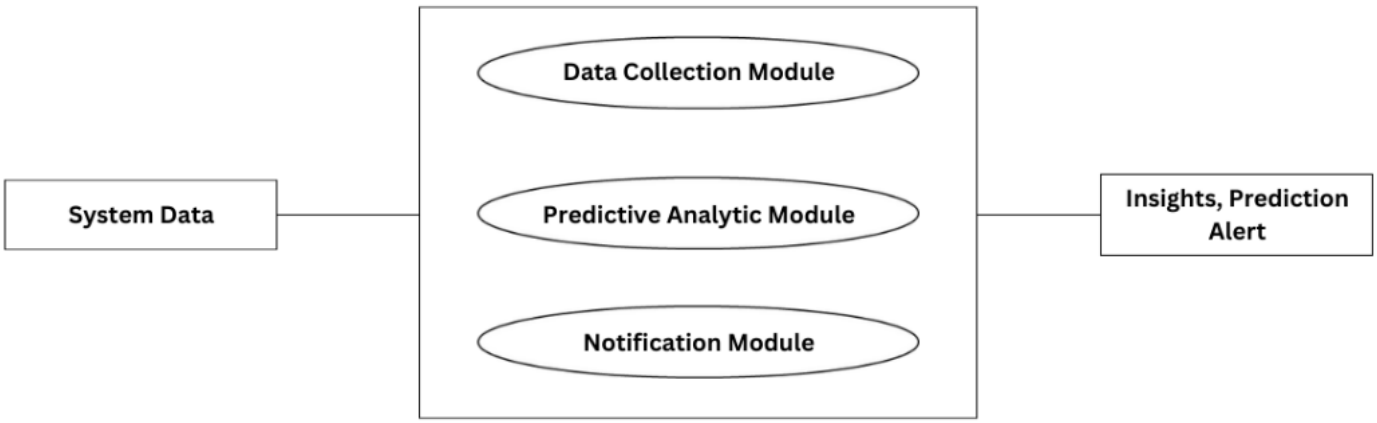
SYSTEM ARCHITECTURE



- **Functional Design**
  - **DFD Level – 0:**

The Level-0 Data Flow Diagram offers a high-level view of the system, emphasizing the core input, processing, and output. **System Data** (logs, performance metrics, and network traffic) is processed by the **IT Support System** to generate actionable outcomes. The primary output includes **Insights**, **Predictions**, and **Alerts**, providing users with key information to take corrective action.

  - **DFD Level – 1:**



Level-1 DFD elaborates on the primary process by breaking it into core modules. The system accepts **Raw Data** (logs, metrics, and traffic) which flows through key processes like **Data Collection**, **Predictive Analytics**, and **Notification Modules**. The output at this level includes **Alerts**, **Predictions**, and **Tickets**, ensuring users receive timely updates and actionable recommendations.

- **Use Case Diagram**

- ## Non-Functional Design

1. Performance Requirements

- The system should process and analyze **real-time logs and metrics** within **2 seconds** to generate actionable insights.
- Ensure a throughput of **1000 tickets/hour** in the Ticket Management Module under peak conditions.
- Dashboard updates must reflect **live system metrics** with a maximum latency of **500 milliseconds**.

2. Scalability Requirements

- The system must support a **10x increase in data volume** (logs, metrics, and alerts) as the user base scales.
- Enable horizontal scaling of modules (e.g., Data Collection and Predictive Analytics) using **cloud-based microservices** architecture.

3. Reliability Requirements

- The system should provide **99.9% uptime** for critical modules like Notification and Ticket Management.
- Implement **redundant failover mechanisms** to ensure uninterrupted operations during hardware or network failures.
- Predictive Analytics must maintain a **95% accuracy rate** under all conditions.

4. Security Requirements

- Encrypt all data in transit using **TLS 1.3** and at rest with **AES-256**.
- Implement **role-based access control (RBAC)** to prevent unauthorized access to system dashboards and tickets.
- Detect and block any **malicious log injection attempts** in real-time.

5. Maintainability Requirements

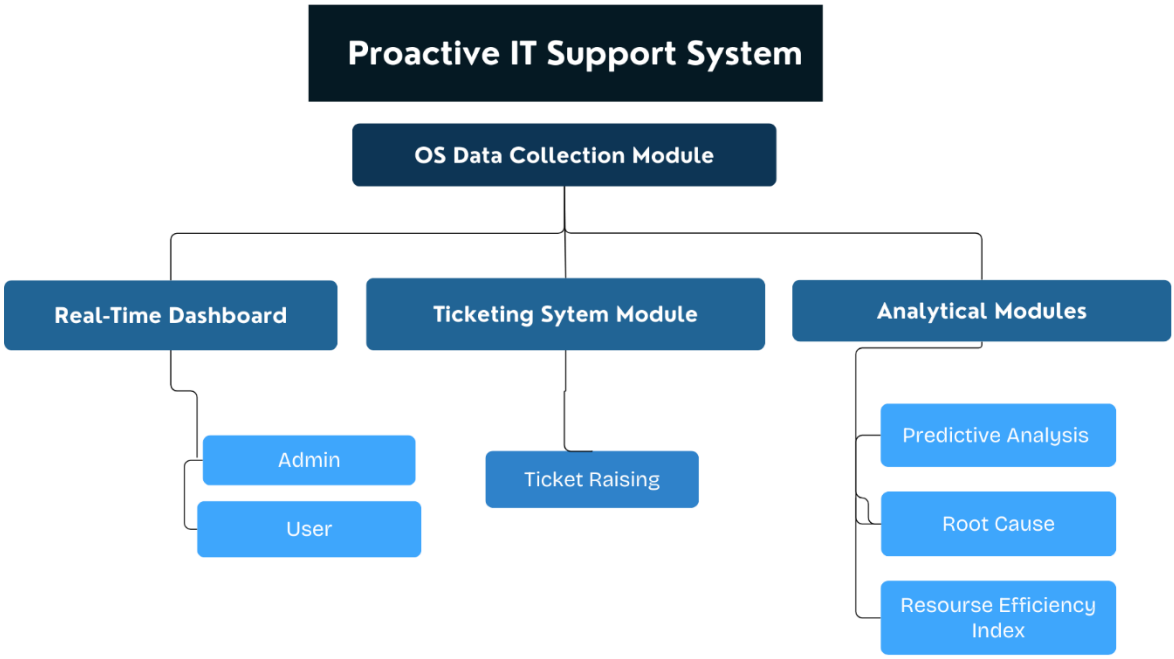
- The system should allow for **hotfix deployments** with minimal downtime (<30 seconds).
- All modules should adhere to **modular programming principles**, enabling easier updates and debugging.
- Ensure the system logs every activity for **audit and debugging purposes** with a log retention period of **6 months**.

Technology/Tools Used:

- **Programming Languages**: Python, JavaScript
- **Operating System**: Linux/Windows Server
- **Frameworks:** React for frontend, Flask for backend, Streamlit
- **Databases:** MongoDB for data storage, MySQL for relational data
- **Machine Learning Libraries**: Scikit-learn, TensorFlow for predictive modeling
- **Monitoring Libraries:** Prometheus for monitoring system performance
- **Logging:** Python's built-in logging module for tracking application behavior.

IMPLEMENTATION DETAILS

Implementation of all modules through logging and multithreading:



1. Project Setup

1.1 Environment Setup

Tools Required:

- Python 3.x
- MongoDB for data storage
- Streamlit for the dashboard
- Required Python libraries: psutil, pymongo, plotly, scikit-learn, numpy, pandas, flask, twilio, concurrent.futures

1.2. Project Directory

Create a structured directory for the project:

```
/Proactive_IT_Support
├── /Modules
│   ├── __init__.py
│   ├── data_collection.py
│   ├── predictive_analytics.py
│   ├── root_cause_analysis.py
│   ├── ticketing_system.py
│   ├── notification_alert.py
│   ├── resource_efficiency.py
│   ├── dashboard.py
│   └── utils.py
├── /Logs
│   ├── data_collection.log
│   ├── analytics.log
│   ├── root_cause.log
│   └── general.log
├── requirements.txt
└── main.py
```

2. Multithreading and Logging

2.1 Multithreading:

1. Implemented parallel execution of modules using concurrent.futures.
2. Optimized response times for prediction and notifications by distributing tasks across threads.

2.2 Logging:

1. Configured structured logging for each module using Python's logging library.
2. Centralized logs for analysis using ELK Stack (Elasticsearch, Logstash, Kibana).

3. Data Collection Module

3.1 Script: data_collection.py

Purpose: Collect system metrics (e.g., CPU, memory, disk usage) and store them in MongoDB.

Steps:

1. **Library Imports:** Import psutil for metrics, pymongo for database interactions, and logging for activity tracking.
2. **Logging Configuration:** Set up logging with RotatingFileHandler to log the data collection processes into /Logs/data_collection.log.
3. **MongoDB Connection:** Use pymongo.MongoClient to connect to the MongoDB database and define a collection (e.g., system_metrics).
4. **Data Fetching:** Implement a get_system_metrics() function that fetches system stats like CPU usage, memory, disk activity, etc., using psutil.
5. **Data Storage:** Write a store_metrics_in_db() function to insert collected metrics into the MongoDB collection with timestamps.
6. **Main Loop:** Use a while True loop to collect and store metrics every 10 seconds with a sleep timer to throttle the process.

4. Predictive Analytics Module

4.1 Script: predictive_analytics.py

Purpose: Analyze collected data and predict system anomalies using machine learning models.

Steps:

1. **Library Imports:** Import numpy, pandas, scikit-learn, and joblib for ML model training and prediction.
2. **Data Preprocessing:** Fetch data from MongoDB, clean it using pandas, and normalize features for model input.

3. **Model Training:** Train an anomaly detection model (e.g., Random Forest or LSTM) on historical data. Save the trained model using joblib.
4. **Real-Time Prediction:** Load the saved model and pass new data through it to predict potential anomalies.
5. **Logging:** Log predictions and anomalies into /Logs/analytics.log for traceability.

5. Analytical Dashboard Module

5.1 Script: dashboard.py

Purpose: Visualize system metrics, predictions, and tickets in real-time using a web-based dashboard.

Steps:

1. **Library Imports:** Import streamlit, plotly, and flask for visualizations and web integration.
2. **Data Retrieval:** Fetch system metrics and predictions from MongoDB. Use APIs for real-time updates.
3. **Visualizations:** Implement interactive plots for CPU, memory usage trends, and anomaly predictions using plotly.
4. **Ticket Overview:** Integrate with the ticketing module to display ticket status and resolutions.
5. **User Interaction:** Enable user filters to customize views (e.g., time range, metric types).

6. Notification Alert Module

6.1 Script: notification_alert.py

Purpose: Send real-time alerts for anomalies or critical issues.

Steps:

1. **Library Imports:** Import twilio, smtplib, and RabbitMQ for message delivery.

2. **Priority-Based Notifications:** Define severity levels (e.g., High, Medium, Low) for different alerts.
3. **Message Delivery:** Use Twilio API for SMS alerts and SMTP for email notifications.
4. **Real-Time Delivery:** Queue alerts using RabbitMQ for reliable and asynchronous delivery.

7. Ticketing System Module

7.1 Script: ticketing_system.py

Purpose: Automate the generation and management of IT support tickets.

Steps:

1. **Library Imports:** Import flask for API creation and PostgreSQL for ticket storage.
2. **API for Ticket Management:** Develop APIs for creating, updating, and closing tickets.
3. **Database Integration:** Use SQLAlchemy to manage ticket data in a relational database.
4. **Ticket Escalation:** Implement escalation workflows based on the severity of issues.

8. Root Cause Analysis Module

8.1 Script: root_cause_analysis.py

Purpose: Perform diagnostics to identify underlying issues causing anomalies.

Steps:

1. **Library Imports:** Import numpy and decision-tree algorithms for rule-based analysis.
2. **Feature Extraction:** Extract meaningful patterns from system logs and metrics.
3. **Correlation Analysis:** Use correlation algorithms to link symptoms to probable causes.
4. **Diagnostics Results:** Output potential root causes with confidence scores.

9. Resource Efficiency Index Module

9.1 Script: resource_efficiency.py

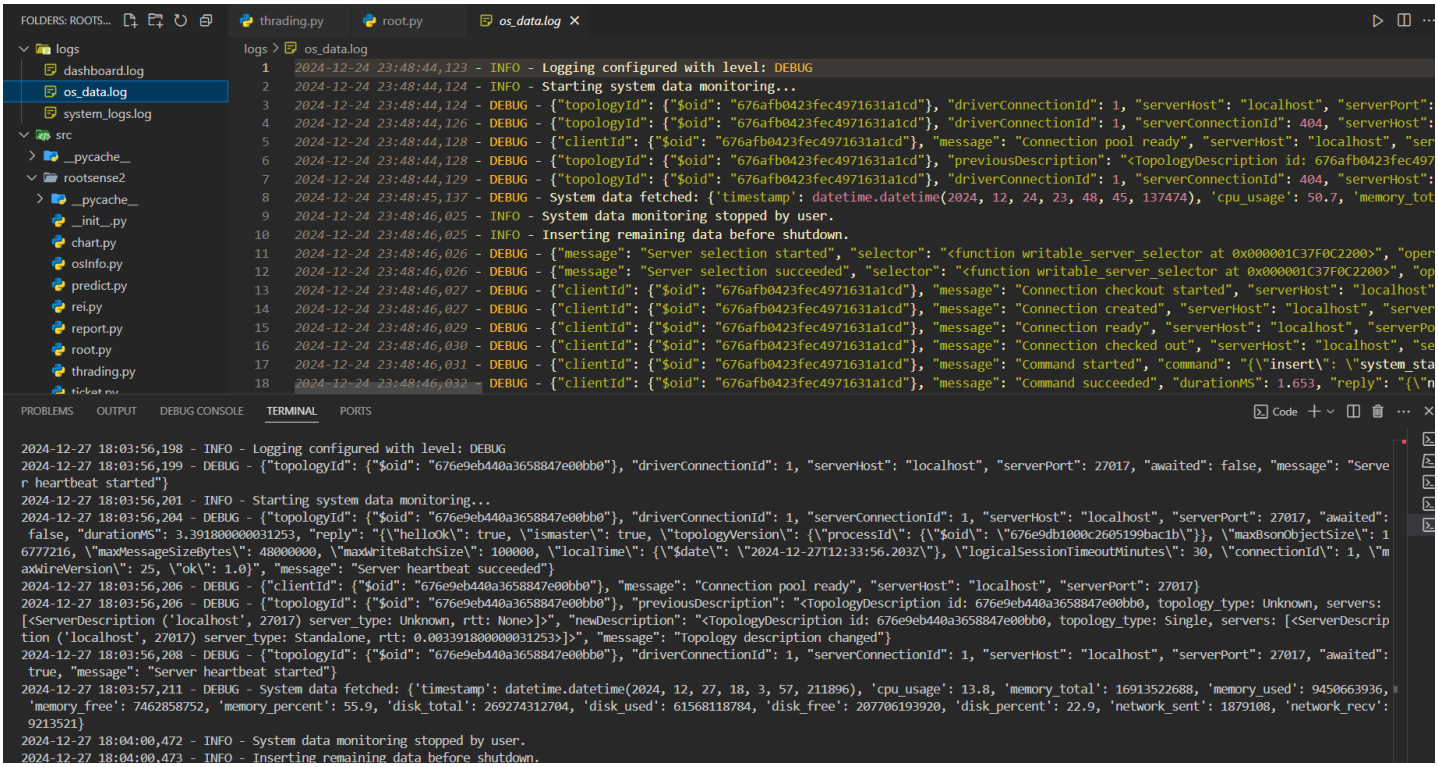
Purpose: Measure system resource utilization efficiency.

Steps:

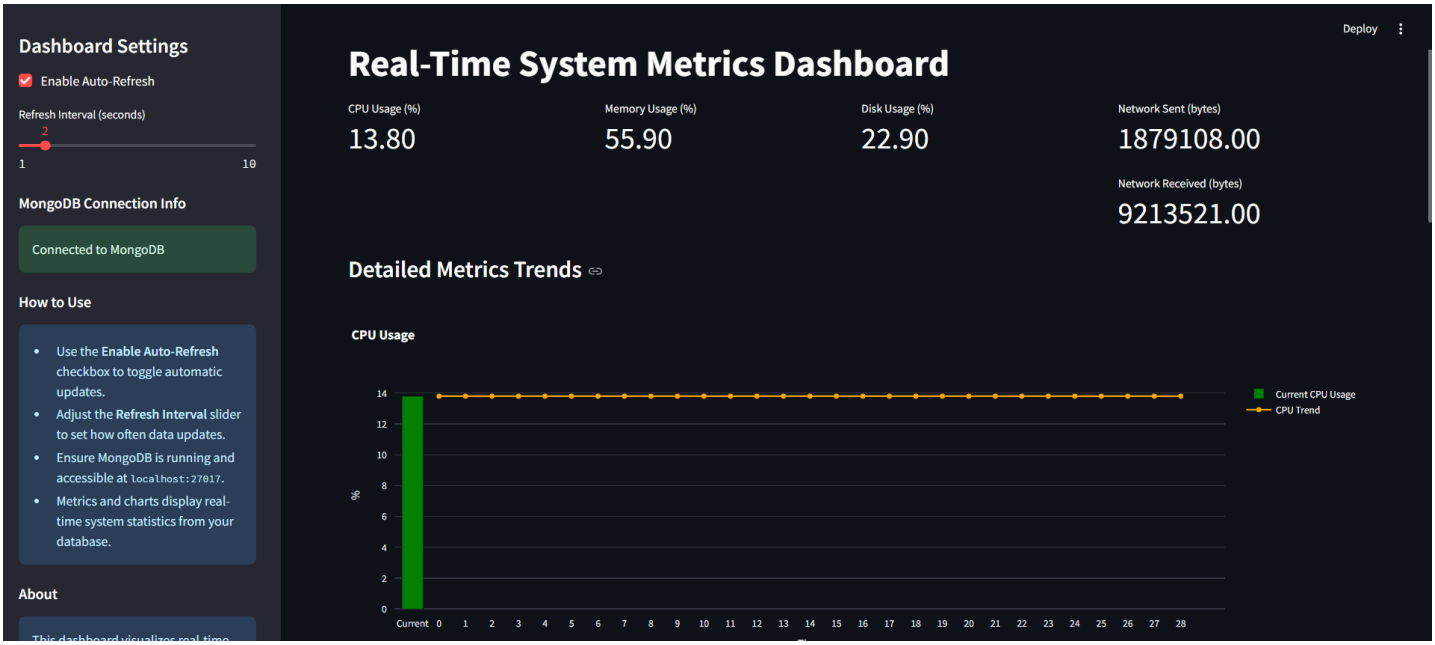
1. **Metrics Collection:** Collect CPU, memory, and disk stats using psutil and store them in MongoDB.
2. **Index Calculation:** Compute the efficiency index as a weighted average of normalized resource metrics.
3. **Visualization:** Display the index on the dashboard with trend graphs.

RESULT ANALYSIS

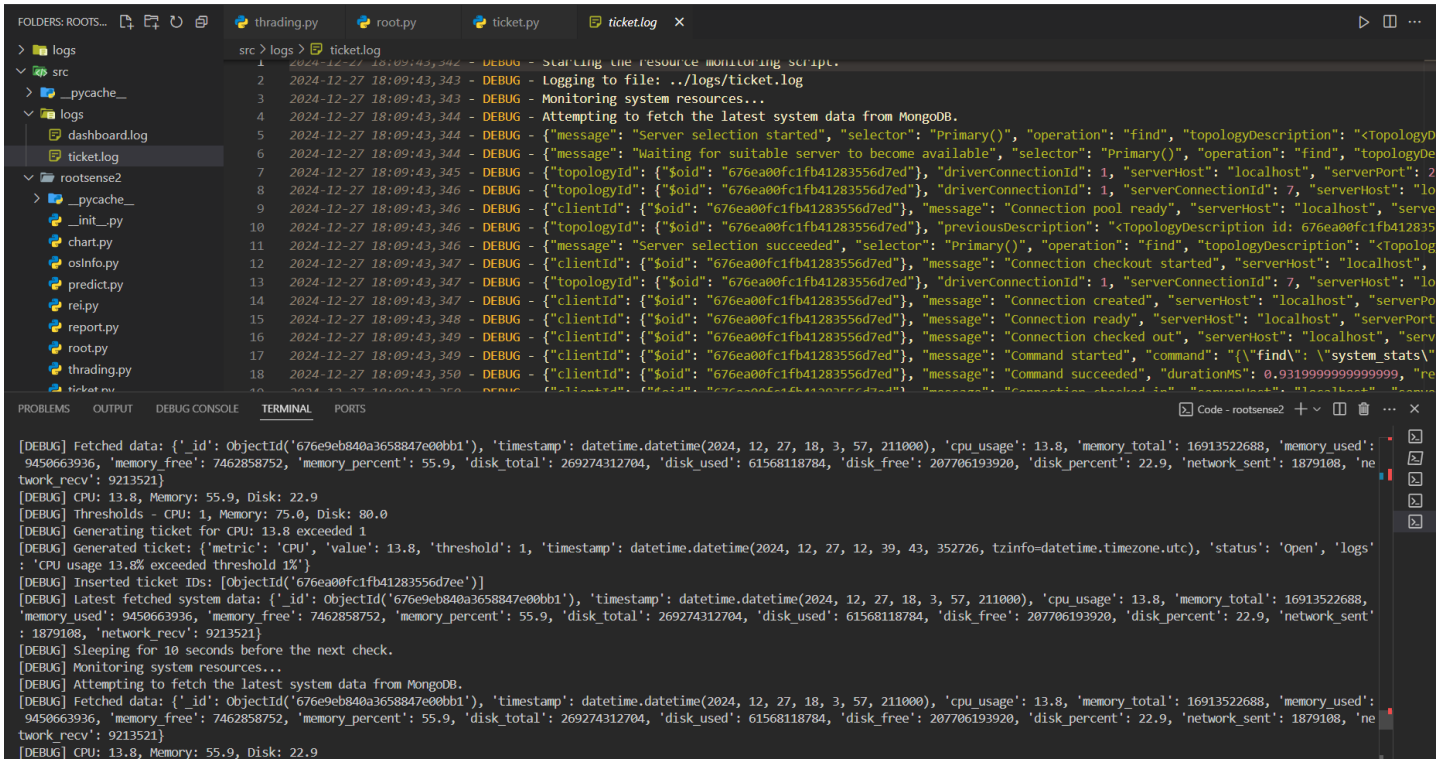
1) OS INFO collection:



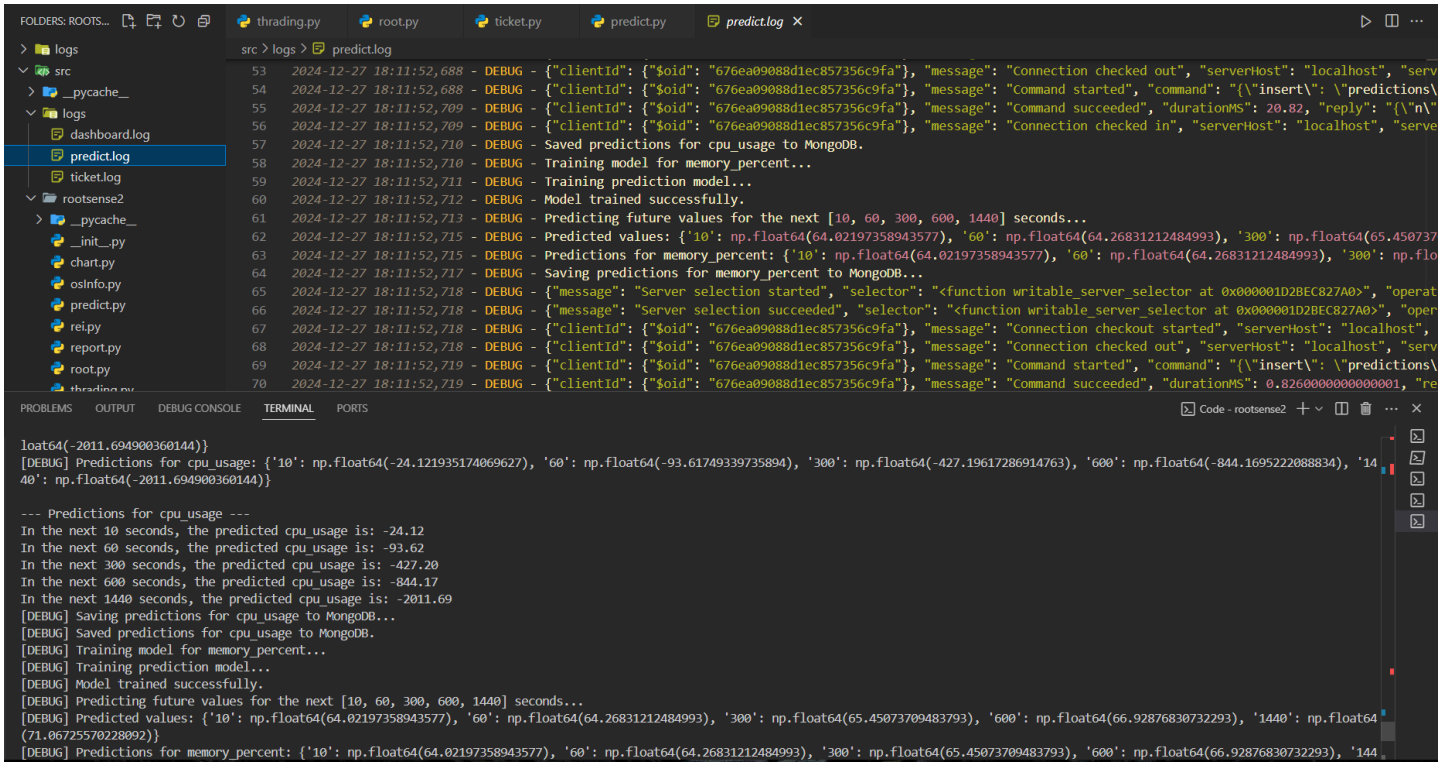
2) Realtime Dashboard:



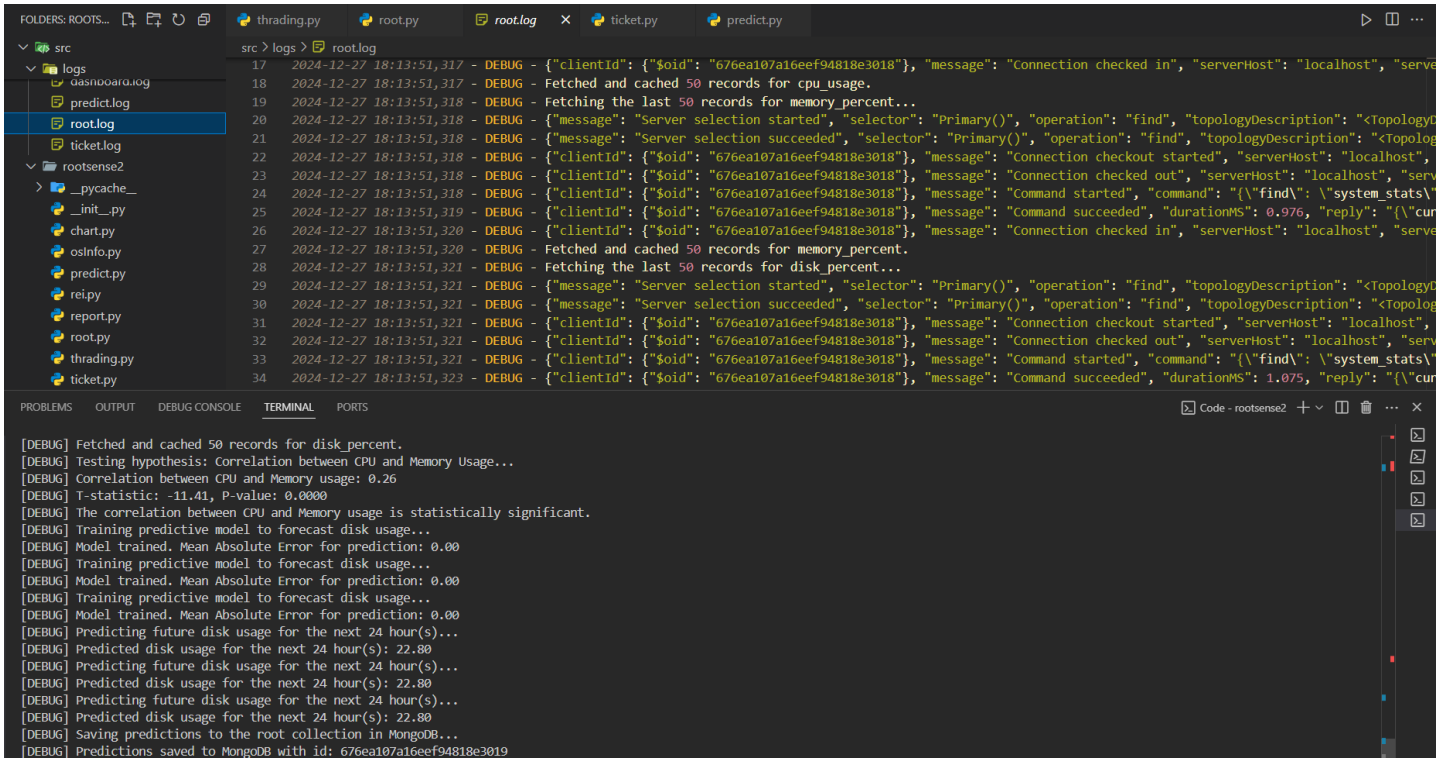
3) Ticket Creation:



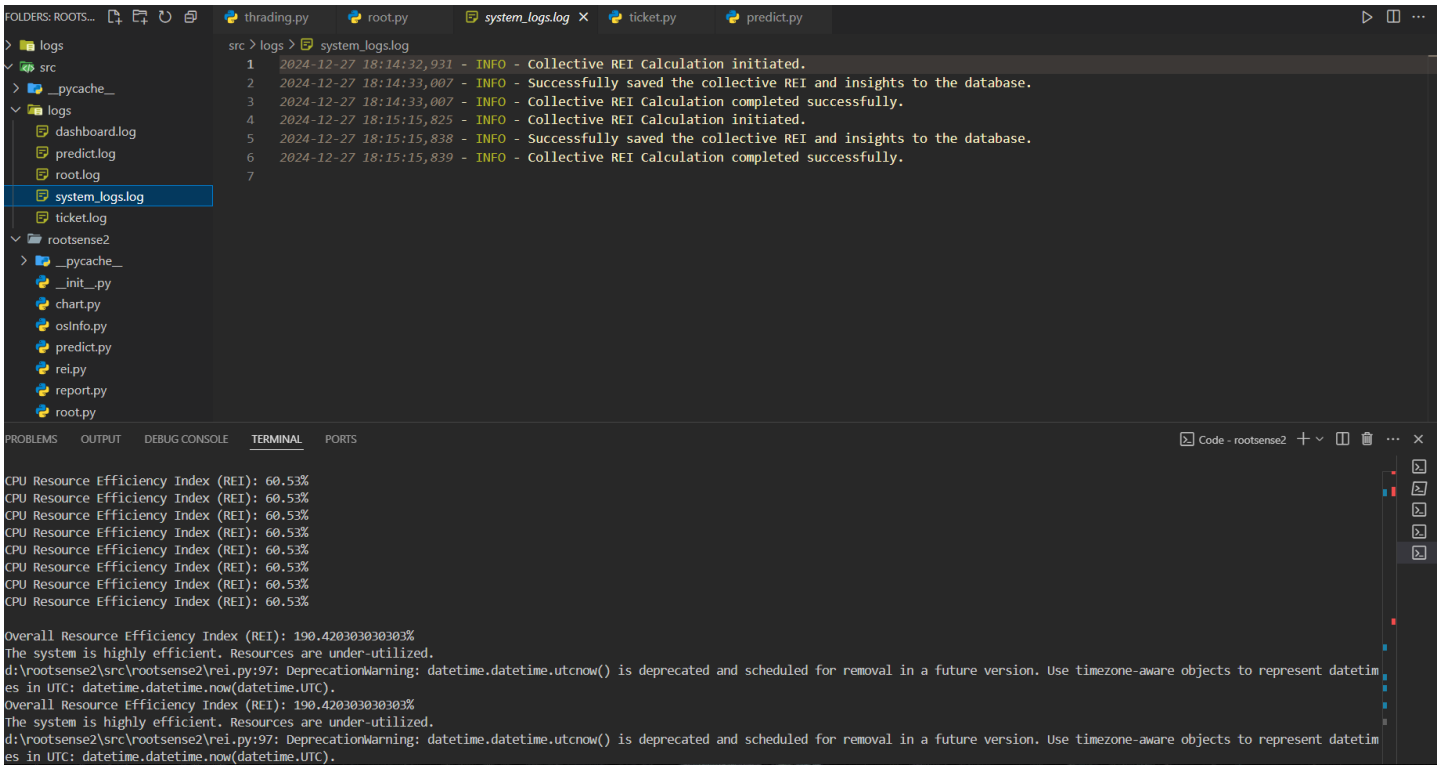
4) Prediction:



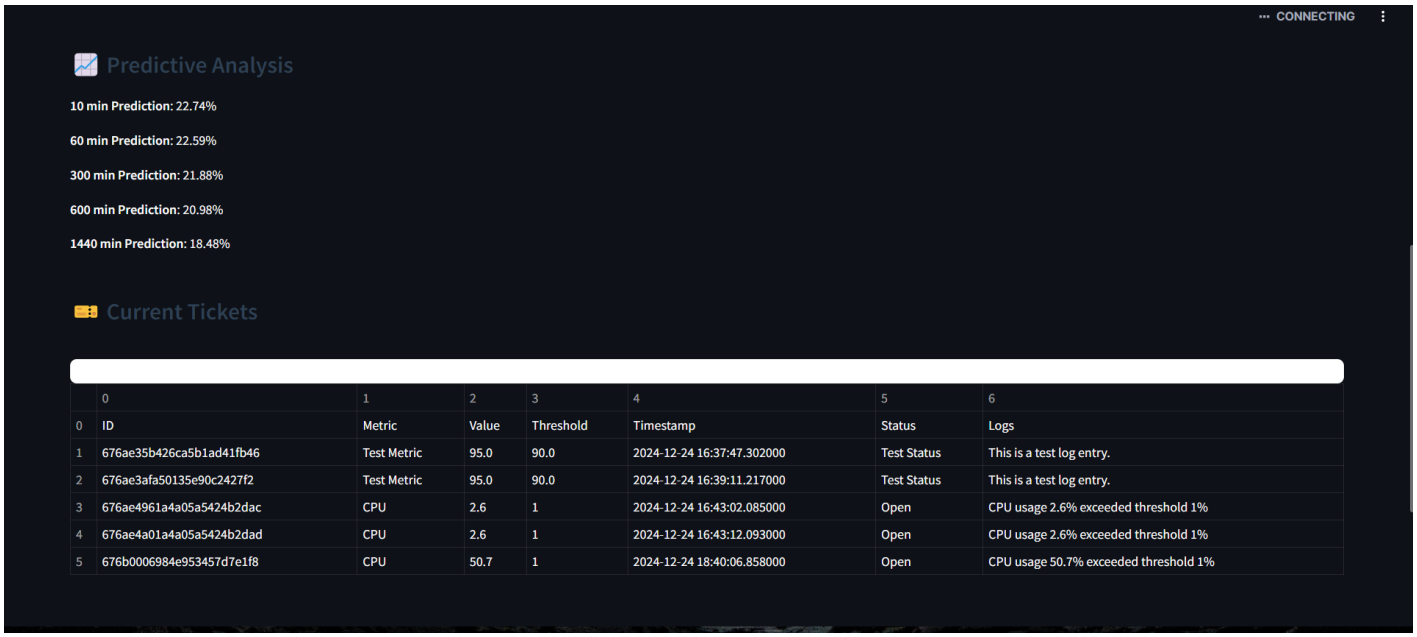
5) Root Cause Analysis:



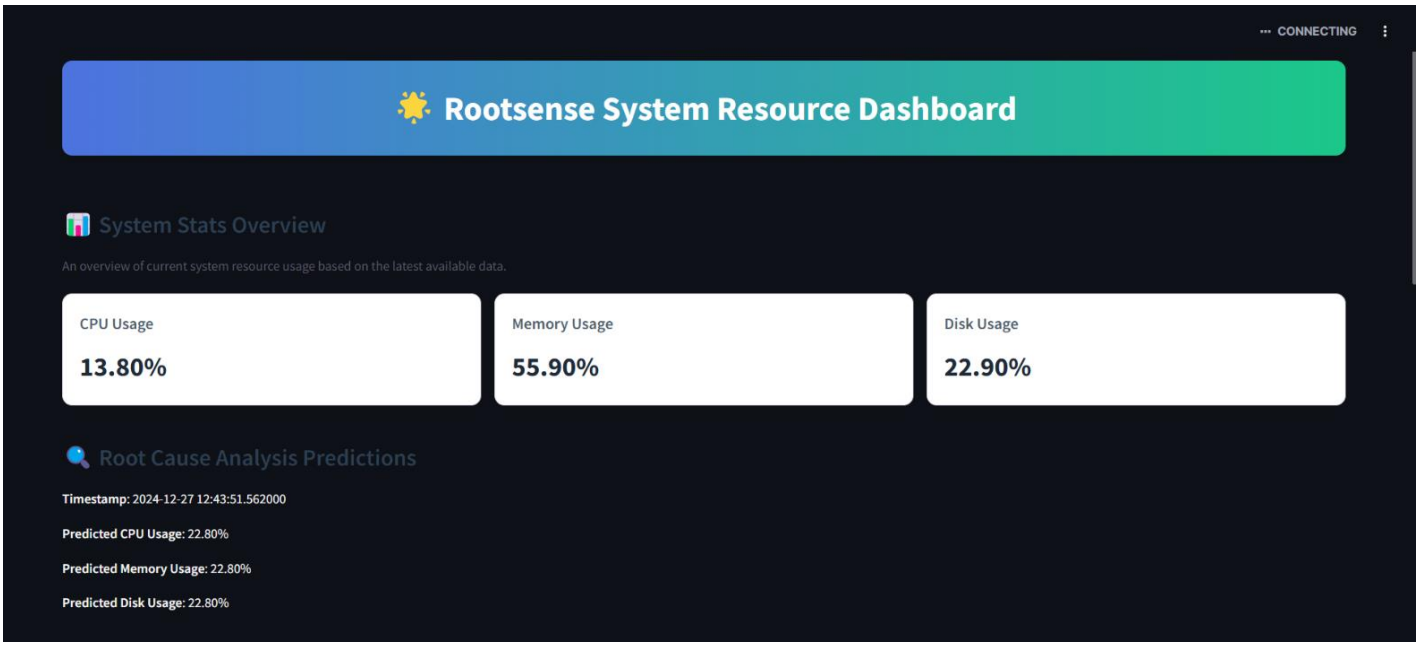
6) Resource Efficiency Index:

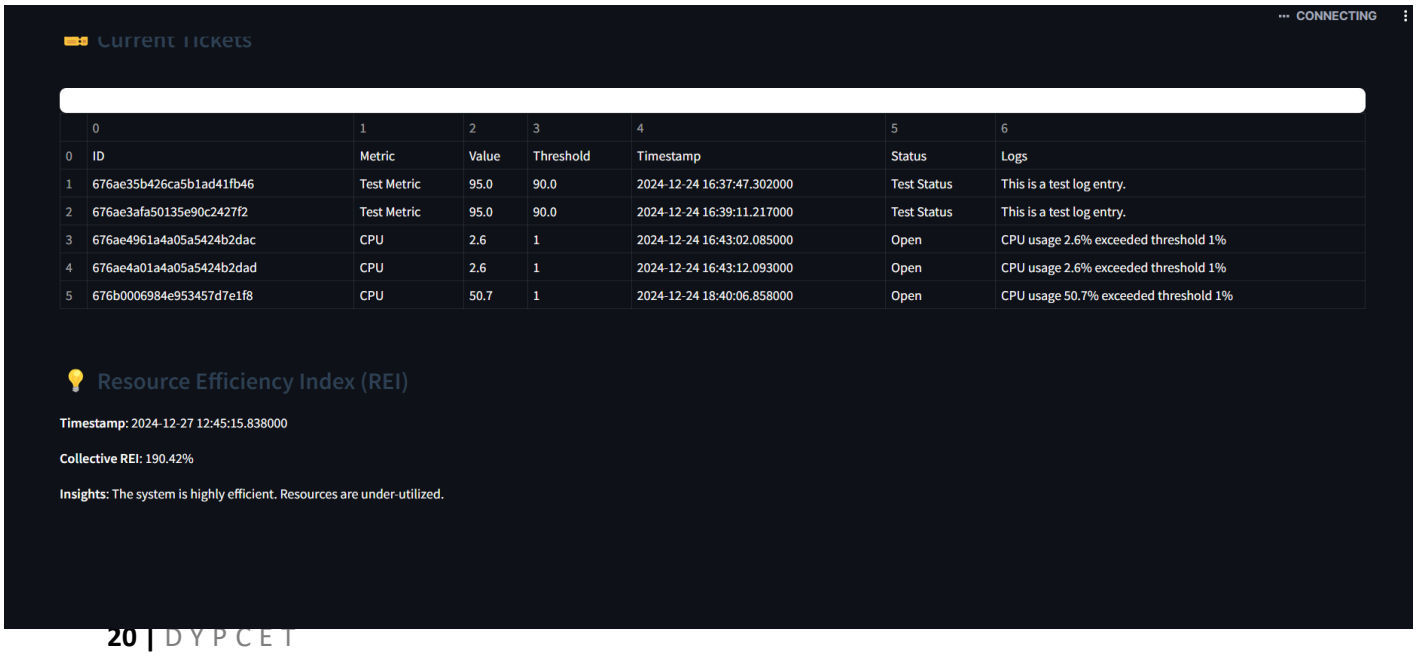


7) Report:



	0	1	2	3	4	5	6
0	ID	Metric	Value	Threshold	Timestamp	Status	Logs
1	676ae35b426ca5b1ad41fb46	Test Metric	95.0	90.0	2024-12-24 16:37:47.302000	Test Status	This is a test log entry.
2	676ae3afa50135e90c2427f2	Test Metric	95.0	90.0	2024-12-24 16:39:11.217000	Test Status	This is a test log entry.
3	676ae4961a4a05a5424b2dac	CPU	2.6	1	2024-12-24 16:43:02.085000	Open	CPU usage 2.6% exceeded threshold 1%
4	676ae4a01a4a05a5424b2dad	CPU	2.6	1	2024-12-24 16:43:12.093000	Open	CPU usage 2.6% exceeded threshold 1%
5	676b0006984e953457d7e1f8	CPU	50.7	1	2024-12-24 18:40:06.858000	Open	CPU usage 50.7% exceeded threshold 1%





	0	1	2	3	4	5	6
0	ID	Metric	Value	Threshold	Timestamp	Status	Logs
1	676ae35b426ca5b1ad41fb46	Test Metric	95.0	90.0	2024-12-24 16:37:47.302000	Test Status	This is a test log entry.
2	676ae3afa50135e90c2427f2	Test Metric	95.0	90.0	2024-12-24 16:39:11.217000	Test Status	This is a test log entry.
3	676ae4961a4a05a5424b2dac	CPU	2.6	1	2024-12-24 16:43:02.085000	Open	CPU usage 2.6% exceeded threshold 1%
4	676ae4a01a4a05a5424b2dad	CPU	2.6	1	2024-12-24 16:43:12.093000	Open	CPU usage 2.6% exceeded threshold 1%
5	676b0006984e953457d7e1f8	CPU	50.7	1	2024-12-24 18:40:06.858000	Open	CPU usage 50.7% exceeded threshold 1%

Conclusion:

The Proactive IT Support System successfully integrates predictive analytics, real-time monitoring, and automated support processes. It effectively reduces system downtime, enhances IT support efficiency, and improves user satisfaction.

Future Development Plan:

- **Advanced Root Cause Analysis**: Incorporate machine learning techniques for more accurate diagnostics.
- **Enhanced User Interface**: Improve the dashboard for better user experience.
- **Scalability Enhancements**: Optimize the system to handle larger infrastructures.
- **Integration with Third-Party Tools**: Enable seamless interaction with other IT management tools.
- **Automated Remediation**: Develop capabilities for the system to automatically resolve identified issues.

REFERENCES

[1] James Smith, “Real-Time Monitoring and Alerts in IT Support Systems, ” *Computers & Security*, Vol. 42, no. 1, pp. 98–110, 2023.

[2] Zheng Shun, “Automated Data Collection for IT Support Systems, ”*Journal of Information Systems Management (JISM)*, Vol. 34, no. 2, pp. 123–134, 2022.

[3] Brown Andrew, “Root Cause Analysis in IT Support: Challenges and Solutions, ” *IT Systems Journal*, Vol. 25, no. 5, pp. 112–125, 2022.

[4] Rajeev Kumar, “A Review on Interactive Dashboards for IT Support, ” *International Journal of IT Management*, Vol. 27, no. 3, pp. 45–60, 2021.

[5] Chen Christopher, “Efficiency of IT Support Ticketing Systems, ” *Journal of Service Management*, Vol. 18, no. 4, pp. 67–78, 2020.

[6] Maria Johnson, “Predictive Analytics in IT Systems: Improving Efficiency and Reliability, ” *Journal of Data Science Applications*, Vol. 15, no. 6, pp. 200–215, 2023.