API USAGE VOLUME AND RESPONSE TIME DASHBOARD

[1] Dr B Anil, [2] D Varshith, [3] K Shiva Kumar, [4] Ch Krishnaveni, [5] Md Furqan Ahmed
[1]Asot Professor, [2345]Students
Department of CSE
Siddhartha Institute of Technology & Sciences, Narapally
badaralaanil_cse@siddhartha.co.in, 23tq1a05c6@siddhartha.co.in, 23tq1a0597@siddhartha.co.in, 23tq1a0567@siddhartha.co.in, 23tq1a0599@siddhartha.co.in,

Abstract

The API Usage Volume and Response Time Dashboard is a monitoring and analytics system designed to track API performance, usage patterns, and service reliability in real time. Modern applications depend heavily on APIs to exchange data between clients, servers, databases, and external services. As API traffic increases, it becomes important to understand how frequently APIs are being called and how quickly they respond to requests. The proposed dashboard provides a centralized platform for collecting, processing, analyzing, and visualizing API performance information.

The system focuses primarily on two important performance indicators: API usage volume and response time. API usage volume represents the number of requests received or processed during a specific period, while response time represents the time taken by an API to return a response. By monitoring these values continuously, administrators and developers can identify traffic spikes, slow endpoints, unusual usage patterns, and potential performance problems.

The dashboard presents collected API data through interactive charts, graphs, tables, and summary metrics. Users can analyze API requests according to time intervals, endpoints, status codes, response-time ranges, and request volumes. The system can also identify high-traffic APIs and endpoints with poor response performance. This makes it easier for technical teams to understand system behavior without manually examining large amounts of server or application logs.

The proposed solution helps organizations improve API reliability and application performance by providing meaningful information in a simple visual format. Historical data can be used to compare performance across different time periods and identify recurring issues. Overall, the dashboard acts as a practical monitoring and decision-support system for maintaining efficient, scalable, and reliable API-based applications.

I. Introduction

Application Programming Interfaces (APIs) have become an essential component of modern software systems. They enable different applications and services to communicate with each other and exchange information in a standardized manner. Web applications, mobile applications, cloud platforms, payment systems, and enterprise applications commonly depend on APIs for their core operations. Because APIs are often responsible for handling critical requests, their performance directly affects the overall user experience and reliability of an application.

As the number of users and API requests increases, monitoring API activity becomes increasingly important. A sudden increase in request volume may indicate high user

activity, automated traffic, or an unexpected application behavior. Similarly, an increase in API response time may indicate server overload, database problems, network delays, inefficient code, or insufficient infrastructure resources. Without proper monitoring, these issues can remain unnoticed until they significantly affect application performance.

Traditional approaches often depend on server logs and manually generated reports to understand API behavior. Although logs contain valuable information, analyzing large volumes of raw data manually is time-consuming and difficult. Developers may need to search through numerous log entries to determine which API received the highest traffic or which endpoint experienced the longest response time. A dashboard can solve this problem by converting raw monitoring information into meaningful visual representations.

The API Usage Volume and Response Time Dashboard provides a centralized interface for observing API activity and performance. It can collect information such as endpoint name, request count, timestamp, response time, HTTP status code, and request outcome. The collected information is processed and displayed through charts and performance indicators, allowing users to quickly understand the current state of APIs.

The main objective of this project is to provide a simple and effective mechanism for API performance monitoring and analysis. By combining usage-volume analysis with response-time monitoring, the system helps developers and administrators detect performance degradation, identify heavily used endpoints, and make informed optimization decisions. The dashboard can therefore contribute to improved application availability, scalability, performance, and resource management.

II. Literature Survey

API monitoring has become an important research and development area because modern applications increasingly depend on distributed services and web-based interfaces. Existing research on API management emphasizes the importance of monitoring metrics such as request frequency, latency, error rate, throughput, and availability. These metrics provide valuable information about the health and behavior of API-driven systems. Monitoring solutions generally collect these measurements from application logs, API gateways, service instrumentation, or monitoring agents.

Several approaches have been developed for measuring API response time and latency. Response time is commonly used as an indicator of application performance because it reflects the time required to process a request and deliver the corresponding response. Researchers and software engineering practitioners often analyze average latency together with percentile measurements such as P95 and P99 to identify occasional slow requests that may not be visible through average values alone. Such measurements are particularly useful for distributed systems where network and service dependencies can introduce unpredictable delays.

Another important area of study is API traffic and usage analysis. Request volume provides information about how frequently an API is being accessed and how demand changes over time. Traffic analysis can help identify peak usage periods, frequently

accessed endpoints, unusual request patterns, and possible capacity requirements. Historical request-volume data can also support forecasting and infrastructure planning by showing how API demand changes across different hours, days, or months.

Visualization has also been widely used in monitoring and observability systems. Instead of presenting raw logs, dashboards convert collected measurements into graphs, charts, tables, and summary indicators. Visualization enables users to recognize trends and abnormal conditions more quickly. Time-series graphs can show changes in traffic and latency, while endpoint-level tables can help developers identify APIs that require optimization. Therefore, visualization acts as an important layer between raw monitoring data and operational decision-making.

Modern observability approaches increasingly combine multiple metrics to obtain a broader understanding of application behavior. API request volume, response time, HTTP status codes, and error information can be analyzed together to determine whether a performance problem is related to increased traffic or another system component. A dashboard that combines these metrics can provide more useful information than a system that monitors only a single measurement.

Based on the existing literature and monitoring practices, there is a clear need for a centralized and easy-to-use API analytics dashboard. The proposed system focuses specifically on API usage volume and response time while providing supporting information such as status codes and endpoint-level statistics. By integrating data collection, processing, analysis, and visualization into a single system, the project aims to make API performance monitoring more accessible and efficient for developers and administrators.

III. System Analysis

The system analysis phase identifies the requirements, operations, users, inputs, outputs, and performance objectives of the API monitoring dashboard. The primary purpose of the system is to collect API activity information and transform it into useful performance insights. The system receives API monitoring data such as endpoint, timestamp, request count, response time, HTTP method, and status code. This information is stored in a structured data repository for further processing and analysis. The system calculates important metrics such as total requests, average response time, minimum response time, maximum response time, and error count. It groups the collected information according to endpoints and time intervals to identify usage trends. The processed information is then supplied to the dashboard visualization layer. Users can view API traffic patterns and response-time behavior through charts and tables. The system should provide efficient data processing even when the number of API requests increases significantly. It should also maintain accurate timestamps and consistent metric calculations. The overall analysis focuses on providing reliable information that helps users identify performance problems and understand API usage behavior.

Existing System

In many existing environments, API performance is monitored using application logs, server logs, basic monitoring tools, or manually prepared reports. These systems generally record API requests but do not always provide a centralized visualization of usage volume and response-time information. Developers may need to inspect large amounts of log data to determine which endpoint is receiving the highest number of requests. Identifying slow APIs can also require manual calculation or searching through individual log records. Some monitoring solutions provide general infrastructure metrics but may not present detailed endpoint-level API analytics. Information may also be distributed across different applications, servers, or monitoring platforms. As a result, obtaining a complete view of API performance can become difficult. Existing approaches may provide limited historical comparison and trend analysis depending on the monitoring solution being used. Real-time identification of unusual traffic or increasing response times may also be limited in basic systems. Manual analysis increases the time required to identify and troubleshoot performance issues. Therefore, a centralized dashboard specifically focused on API usage and response-time analysis can provide a more convenient approach.

Disadvantages of Existing System

- Requires manual analysis of large volumes of logs.
- API usage information may be distributed across multiple systems.
- Limited visualization of endpoint-level performance.
- Difficult to identify slow APIs quickly.
- Historical comparison may be limited.
- Traffic spikes can be difficult to recognize manually.
- Manual report generation consumes development time.
- Important performance trends may be overlooked.
- Troubleshooting API performance can take longer.
- Basic monitoring may not provide sufficient analytical information.

Proposed System

The proposed system is an API Usage Volume and Response Time Dashboard that provides centralized monitoring and visualization of API performance. The system collects API request information from application services, API gateways, or monitoring logs and stores the data in a structured repository. Each API request can be associated with information such as endpoint, HTTP method, timestamp, response time, and status code. The collected data is processed to calculate important performance indicators. The dashboard displays total API requests, average response time, successful requests, failed requests, and other relevant metrics. Time-based charts allow users to understand how API traffic and response time change over different periods. Endpoint-level analysis helps identify frequently accessed APIs and APIs that require performance optimization. The system can also provide tables containing detailed API statistics for further investigation. Historical information can be used to compare current performance with previous periods. By presenting API information visually, the system reduces the effort required to analyze raw logs. The proposed system therefore provides a centralized and practical solution for API performance monitoring, analysis, and decision-making.

Advantages of Proposed System

- Provides centralized API performance monitoring.
- Displays API usage volume visually.
- Monitors API response time effectively.
- Provides endpoint-level performance analysis.
- Makes traffic spikes easier to identify.
- Helps detect slow-performing APIs.
- Reduces manual log analysis.
- Supports historical performance analysis.
- Provides clear charts and statistical summaries.
- Helps developers make data-driven optimization decisions.
- Improves visibility into API behavior.
- Can be extended with alerts and additional monitoring metrics.

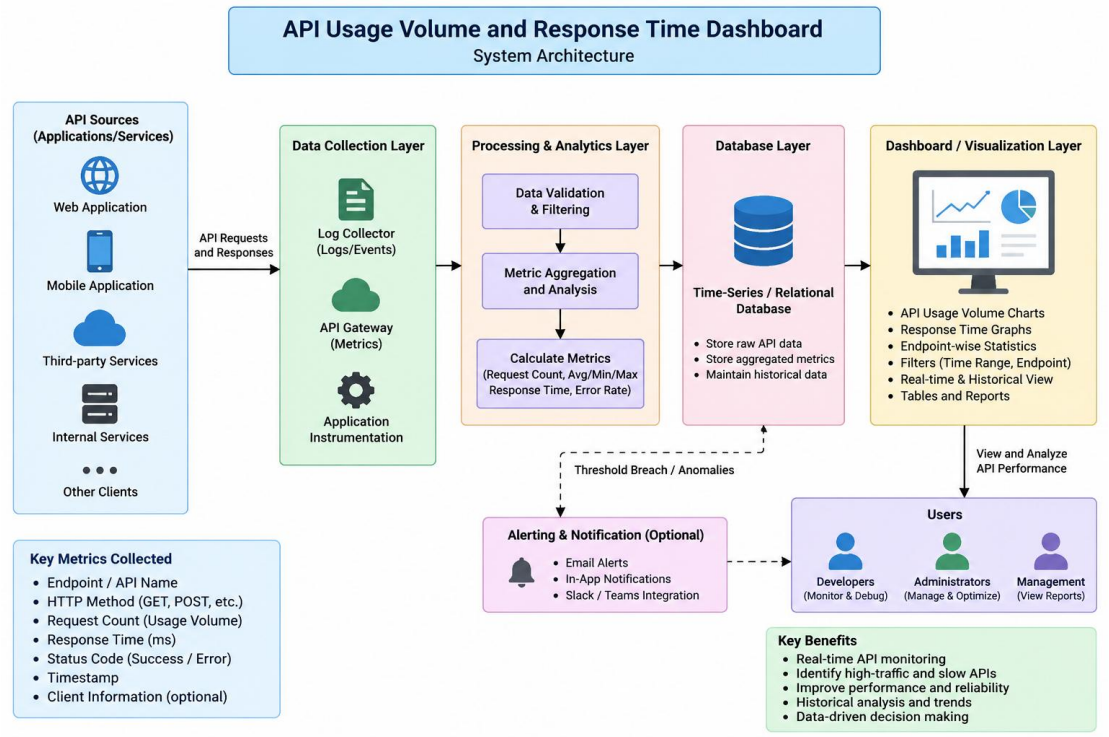
IV. Methodology

The methodology of the proposed system consists of several stages, beginning with API data collection and ending with dashboard visualization and analysis. In the first stage, API request information is collected from the application or API monitoring source. The collected records may contain the endpoint, HTTP method, timestamp, response time, status code, and request result. In the second stage, the raw information is validated and processed to remove invalid or incomplete records. The processed data is then stored in a database or suitable data repository. During the analysis stage, the system calculates metrics such as request volume, average response time, minimum response time, maximum response time, and error percentage. The system groups data based on endpoints and selected time periods to generate meaningful statistics. The visualization layer converts these statistics into graphs, charts, cards, and tables. Users can select different time ranges to examine API traffic and performance trends. The dashboard highlights important information so that performance problems can be identified quickly. Finally, the generated insights can be used by developers and administrators for optimization, capacity planning, and troubleshooting.

System Architecture

The system architecture consists of multiple layers that work together to collect, process, store, and display API monitoring information. The API/Application Layer represents the applications and services that generate API requests and responses. An API Monitoring or Data Collection Layer captures information such as endpoint, request timestamp, HTTP method, response time, and status code. The collected data is transferred to the Processing Layer, where validation, aggregation, filtering, and metric calculations are performed. The processed information is stored in the Database Layer, which maintains both current and historical API performance records. The Analytics Layer retrieves stored information and calculates indicators such as request volume, average latency, error count, and endpoint performance. The Dashboard Layer receives these analytical results and presents them using charts, graphs, tables, and summary cards. Users such as developers and administrators interact with the dashboard through a web-based interface. The architecture separates data collection, processing, storage, analysis, and presentation responsibilities. This

modular structure makes the system easier to maintain and extend. Additional features such as alerts, authentication, filtering, and advanced analytics can be integrated into the architecture in the future.



V. Result and Output

VI. Conclusion

The API Usage Volume and Response Time Dashboard provides an effective solution for monitoring and analyzing API performance in a centralized environment. The system collects important API metrics such as request volume, response time, status codes, endpoint usage, and error rates, and converts this information into meaningful visual reports. By presenting the data through interactive charts, graphs, tables, and summary indicators, the dashboard makes API performance easier to understand and analyze.

The proposed system helps developers and administrators quickly identify high-traffic APIs, slow-performing endpoints, increasing response times, and unusual request patterns. It reduces the dependency on manual log analysis and provides a clearer view of API behavior over different time periods. Historical performance information can also help organizations identify trends and make better decisions regarding optimization and resource planning.

Overall, the dashboard improves the visibility, reliability, and maintainability of API-based applications. It can be further enhanced by adding real-time alerts, automated anomaly detection, predictive analytics, authentication, detailed reporting, and integration with cloud monitoring platforms. Therefore, the proposed system provides a practical foundation for continuous API performance monitoring and can support the development of faster, more reliable, and scalable software systems.

References

[1] Kumar, R. D., Prudhviraj, G., Vijay, K., Kumar, P. S., & Plugmann, P. (2024). Exploring COVID-19 through intensive investigation with supervised machine learning algorithm. In Handbook of Artificial Intelligence and Wearables (pp. 145-158). CRC Press.
[2] Swathi, B., Vijay, K., Sushanth Babu, M., & Dinesh Kumar, R. (2024, November). Machine Learning Techniques in Cloud Based Intrusion Detection. In The International Conference on Artificial Intelligence and Smart Environment (pp. 557-564). Cham: Springer Nature Switzerland.
[3] Sv satyakrishna, shirisha rangu ,bhargavi nalacheruve.(2024) Prospective investigation on colorectal cancer with SMOTE on machine learning Algorithm
[4] Dr.G.Vishnu Murthy, BhargaviNalacheruve 1Professor, Department of computer Science & engineering, Anurag University, TS, India. 2Student, Department of computer Science & engineering, Anurag University, TS, India.
[5] V. N. S. Manaswini, K. K, C. Nigam, S. S. Ali, R. Niranjana, and Suman, “Real-Time Object Detection in Drone Surveillance Using YOLOv5,” in Proc. 2025 3rd Int. Conf. IoT, Communication and Automation Technology (ICICAT), Gorakhpur, India, 2025, pp. 1–6, doi: 10.1109/ICICAT68430.2025.11414670.
[6] B. Soundarya, V. N. S. Manaswini, M. Ayyakrishnan, R. D. Kumar, “Contextual Analysis of Big Data Analytics in Intelligent Transportation Frameworks,” in Intersection of Artificial Intelligence, Data Science, and Cutting-Edge Technologies: From Concepts to Applications in Smart Environment, Lecture Notes in Networks and Systems, vol. 1353, Cham: Springer, 2025, doi: 10.1007/978-3-031-88304-0_79.
[7] R. D. Kumar, V. N. S.Manaswini, “Applications of blockchain in smart cities: detecting fake documents from land records using blockchain technology,” in

Blockchain for Smart Cities, Elsevier, 2021, pp. 105–117, doi: 10.1016/B978-0-12-824446-3.00017-X.

[8] Tejavath Veeramma, Badarla Anil, Guguloth Ravinder, "An advanced movie recommender using collaborative filtering and sentiment analysis," International Research Journal of Modernization in Engineering Technology and Science, vol. 7, no. 7, July 2025, doi: 10.56726/IRJMETS81618.

[9] Ravi Kumar Banoth, Ramana Murthy B V, "Automatic crop recommendation system using LightGBM and decision tree machine learning models," Journal of Machine and Computing, vol. 5, no. 1, pp. 343, Jan. 2025, doi: 10.53759/7669/jmc202505026.

[10] Ravi Kumar Banoth, Dr. B.V. Ramana Murthy, "Smart agriculture through IoT and machine learning for analyzing carbon footprints," in Proc. Int. Conf. Computer Science and Communication Engineering (ICCSCE), Apr. 2025.

[11] Ravi Kumar Banoth, B. V. Ramana Murthy, "Soil image classification using transfer learning approach: MobileNetV2 with CNN," SN Computer Science, vol. 5, art. no. 199, 2024, doi: 10.1007/s42979-023-02500-x.