DEVELOPMENT OF A WEB-BASED MONITORING SYSTEM FOR THE MAINTENANCE OF CENTRIFUGAL PUMPS

R.M. Eqbal[1], H.M.Y. Norfazrina[1*], H. Meftah[1], D. Pebrianti[1], M.R. Muhammad Hafeez[2]

[1] Mechanical and Aerospace Engineering Department, International Islamic University Malaysia, Jalan Gombak, Malaysia

[2] Group Technology International, Maybank, Maybank Tower, 100, Jalan Tun Perak, Bukit Bintang, 50050 Kuala Lumpur, Wilayah Persekutuan Kuala Lumpur, Malaysia

Corresponding Author's Email: norfazrina@iium.edu.my

Article History: Received 8 November 2025; Revised 22 December 2025; Accepted 15 January 2026

ABSTRACT: *Industrial centrifugal pumps require continuous monitoring to prevent costly failures and ensure operational safety. Traditional manual inspection methods are time-consuming, error-prone, and often fail to detect faults before catastrophic failures occur. This study presents the development of a web-based monitoring system for centrifugal pump maintenance using Python and Django framework. The system monitors five critical parameters of the centrifugal pump: rotational speed, acceleration, demodulation output, vibration and temperature, with predefined industry-standard thresholds for fault detection. The web interface features data visualization, color-coded alerts, and historical trend analysis, designed to be accessible to maintenance personnel. The result demonstrated 95% accuracy in data representation with threshold breach detection capabilities. The system successfully processes sensor data from established datasets of healthy and maintenance-prone stages of centrifugal pumps. The web monitoring system managed to implement automated alerting when parameters exceed safety limits and provides an intuitive dashboard for monitoring pump health status. This automated approach reduces reliance on manual inspections, enables predictive maintenance strategies, and enhances operational efficiency while maintaining user-friendly accessibility for diverse industrial personnel.*

KEYWORDS*: Web Monitoring, Centrifugal Pumps, Python, Industrial Automation*

1.0 INTRODUCTION

In the modern era where Internet of Things (IoT) and Artificial Intelligence (AI) are increasingly embedded into every facet of industrial operations, many sectors are moving toward total observability, a framework that unifies data logs, sensor readings, and event streams into a seamless flow for comprehensive system oversight and control. Within the context of Structural Health Monitoring (SHM) particularly in centrifugal pumps, total observability enables the transformation of unknowns into actionable insights by detecting anomalies and discrepancies in datasets collected from multiple sensors. These synthesized datasets are then delivered to an interactive web-based dashboard for real-time visualization, enabling timely corrective action and accurate failure prediction. While achieving total observability remains a challenge, i.e. requiring robust sensor coverage, advanced data analytics, and reliable system integration, the framework begins with establishing useful data collection and continuous monitoring systems. Such systems

ISSN 2180-1053 e-ISSN 2289-8123 Vol.18 No.1

lay the foundation for transforming raw sensor data into a holistic operational view, ensuring reliability, efficiency, and extended service life.

Web-based monitoring systems for fault detection integrate the functionality of automated monitoring systems with online platforms utilizing internet connectivity to remotely monitor and manage the structural health of heavy machineries like centrifugal pumps. This type of system combines the use of sensors, cameras and other imaging devices to collect information and data on faults like crack formations and their growth, which is gathered and then transmitted and displayed in real-time on a centralized web-based interface. Contrary to the traditional methods of manual inspection, web-based platform allows users to access and view the sensor data on potential faults at any time and from any device with internet connectivity which makes it a versatile solution for structural health monitoring across various locations around the globe.

The purpose of this project is to address the limitations of traditional manual inspections in industrial machinery maintenance, which are often time-consuming, error-prone, and typically accessible only to trained engineers. By developing a web-based monitoring system for centrifugal pumps using programming language like Python, the restricted accessibility and communication gaps present in current monitoring solutions is to be countered. The system should be designed to provide preliminary fault detection and data visualization through a user-friendly interface, enabling not only engineers but also non-technical personnel to participate in the maintenance process.

The main objectives of this research are to develop a Python-based web platform that enables efficient and effective data presentation to support predictive maintenance of centrifugal pumps, and implement an effective alerting system for deviations in the dataset. This study emphasizes data visualization, which constitutes the second stage of a broader framework. The first stage of this framework involving fault prediction, detection, and identification in centrifugal pumps, is beyond the scope of the present work. This approach seeks to enhance early fault detection, streamline maintenance workflows for better data observability, and foster a more collaborative environment in industrial settings.

2.0 COLOUR PSYCHOLOGY AND HUMAN PERCEPTION

The strategic use of green for "safe" states and red for "danger" alerts is grounded in well-established principles of visual cognition and human perception. Green induces cognitive ease during monitoring tasks because humans instinctively associate it with safety and normal operations which is a phenomenon validated by industrial psychology research. Studies demonstrate that green interfaces reduce mental workload by up to 23% compared to neutral colours during routine equipment checks, enabling faster recognition of acceptable operating conditions [1]. On the contrary, red triggers pre-attentive processing which the brain identifies as 200 milliseconds faster than other colours due to its biological association with danger signals [2]. To amplify urgency, the pulsating "DANGER" animation was intentionally incorporated, aligning with ISO 9241-210 standards for hazard communication which recommend dynamic visual cues for critical alerts.

These design decisions were made with the intention of minimizing cognitive load and lowering the possibility of supervision, especially in high-stakes industrial settings where

ISSN 2180-1053 e-ISSN 2289-8123 Vol.18 No.1

quick reaction is essential. The dashboard should comply with safety-critical interface design best practices by using colour as its main communication tool [3]. This strategy is particularly crucial because the objective of the project is to make the system usable by people who are not specialists who might not have a lot of technical knowledge, like general maintenance employees. In summary, studies on visual cognition and human factors have greatly influenced the dashboard's functional and aesthetic design, guaranteeing both efficiency and user-friendliness.

3.0 LANGUAGE USED FOR WEB-BASED MONITORING

Python has emerged as a highly suitable programming language for developing web-based monitoring systems for fault detection due to its versatility, simplicity, and extensive library support. Its clear syntax and readability enable developers to write and maintain code efficiently, which is especially beneficial in collaborative and complex projects such as fault detection in industrial machinery. Python's robust ecosystem includes libraries like OpenCV for image processing and Pandas for data analysis, allowing for the rapid implementation of advanced algorithms essential for real-time fault detection [4]. Compared to languages such as C++ or Java, Python offers a more flexible and rapid development process, reducing the time and complexity required for setup and deployment while maintaining cross-platform compatibility [5].

A key advantage of Python is its integration with high-level web frameworks such as Django, which encourages rapid development and clean, pragmatic design. As shown in Figure 1, Django's modular architecture divides the application into URLs (routing), views (request processing), models (data structure and database interaction), and templates (dynamic content rendering). This separation of concerns not only improves code organization and maintainability but also simplifies the development of dynamic, data-driven web applications. Django's built-in admin interface further streamlines data management and user interaction, making it easier to monitor user activity and system performance in real time (Django Introduction - Learn Web Development | MDN, 2024).

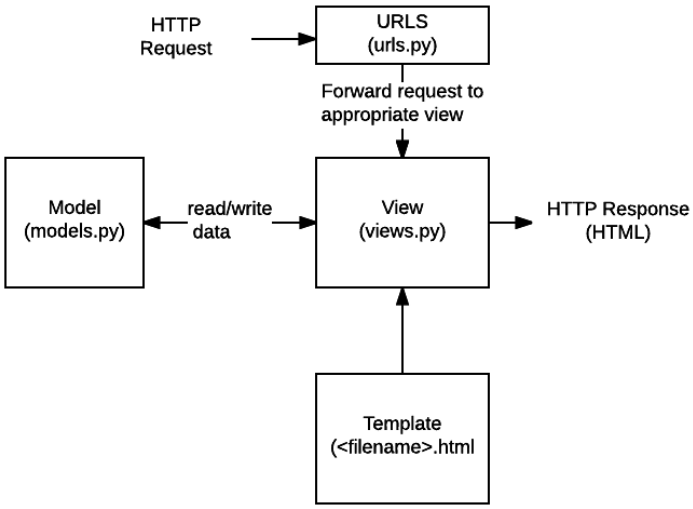


Figure 1. Flowchart of data processing codes in Django framework (Django Introduction - Learn Web Development | MDN, 2024)

ISSN 2180-1053 e-ISSN 2289-8123 Vol.18 No.1

4.0 METHODOLOGY

4.1 Parameter Selection

The primary stage of a bigger predictive maintenance framework involves fault prediction, detection, and identification of the machineries which includes system definition, data analysis, and data mining [6]. The final stage is to visualize the synthesized data using web-based monitoring system for clearer observability of the whole system. Current study focuses on the latter stage which uses established data from two centrifugal pumps of the same manufacturers with different health states: healthy state and maintenance-prone state. The datasets used in this study can be found on Kaggle (https://www.kaggle.com/datasets/panosmallioris/sensor-measurements-of-centrifugal-pumps). More details on the centrifugal pumps can be found in the work by [7]. Both centrifugal pumps gave measurements signals containing five key parameters which are the most significant indicators of centrifugal pump health condition: rotational speed (mm/s), root mean square of acceleration and demodulation signals (g), peak-to-peak acceleration (g), and temperature (°C). These parameters were selected through Spearman correlation analysis, which measures the strength and direction of the relationship between sensor data and pump health status.

The correlation values of the input features from the Spearman statistical analysis [7] demonstrate predictive capability: the rotational speed, demodulation and acceleration are the most informative feature and gives good correlation between 0.832 to 0.865. On the contrary, temperature had a weaker negative correlation (-0.435), indicating that while overheating is a concern, environmental factors can influence readings. Nonetheless, temperature can be the first layer of alerting system as it is observed that commonly there is a temperature spike before the total malfunction of a machine. These five parameters were selected to be visualized in the current developed web-monitoring system as they contain the most informative features for health state classification.

4.2 Threshold Identification

The next step after identifying the key parameters for data visualization is to set the threshold. Setting thresholds based on industry standards is vital for defining acceptable limits for monitored parameters identified in centrifugal pumps. The established thresholds exclusively for centrifugal pumps are shown in Table 1.

Table 1. List of parameters and identified thresholds based on industry standards [7]

Parameter	Threshold and its failure indication
Velocity_ISO	Values exceeding 4.5 mm/s indicate impeller imbalance or shaft misalignment
RMS Demodulation	Values above 0.4 g signal bearing spalling or pitting issues
RMS Acceleration	Thresholds beyond 0.5 g indicate abnormal kinetic energy shifts from gear defects
Peak-to-Peak Acceleration	Values surpassing 5.0 g imply severe mechanical stress
Temperature	Readings above 28°C suggest inadequate lubrication or cooling failure

ISSN 2180-1053 e-ISSN 2289-8123 Vol.18 No.1

4.3 Python-Based System Development

The development process involved setting up the project infrastructure by creating a dedicated development environment with Django, Pandas, and OpenPyXL packages. Django was chosen as the web framework for its robust features, security measures, and scalability. The system architecture includes models for data structure definition (*models.py*), views for processing HTTP requests (*views.py*), and templates for user interface presentation (*dashboard.html*). The coding development process in this study follows a structured methodology that integrates both backend and frontend workflows to achieve a fully functional monitoring system.

The development begins with settings configuration in *settings.py*, where the "monitor" app is registered in *INSTALLED_APPS* to enable database migration and template rendering. Additionally, *STATICFILES_DIRS* is defined to centralize static assets like CSS, JavaScript, and images, ensuring seamless integration between backend logic and frontend display.

The backend development is implemented in Django, starting with models in *models.py (SensorData)* to structure database tables for storing sensor parameters (velocity_ISO, RMS demodulation, RMS acceleration, peak-to-peak acceleration, and temperature) with timestamps, precision float storage, and reverse chronological ordering. The Excel Data Reader in *data_reader.py* imports sensor data from Excel using Pandas, mapping each row to *SensorData* fields.

In *views.py*, two main functions handle data flow: rendering *dashboard.html* and fetching sensor data for visualization. The *get_sensor_data* function loops through the dataset rows, updates an index tracker, stores readings in the database, retrieves the 20 latest entries, generates timestamps, and evaluates parameters against safety thresholds and returning *JSON* format for the frontend display. URL routing in *urls.py* links endpoints to views, including admin access, the dashboard root, and real-time data retrieval.

On the frontend display, *dashboard.html* is built with HTML, CSS, and JavaScript, using Chart.js for interactive visualization. The interface features headers, parameter selectors, status cards with safe/danger alerts, a trend chart, and an alert panel for threshold breaches. CSS ensures responsive design and visual emphasis on critical alerts with animations. JavaScript functions handle chart initialization, dashboard updates, system-wide safety checks, and alert management. Real-time synchronization is achieved through asynchronous polling of */get_sensor_data/* at one-second intervals, with error recovery mechanisms and dynamic user interaction handling via event listeners. The coding development in this study ensures a robust, responsive, and user-friendly monitoring platform that integrates efficient backend processing with an intuitive, real-time dashboard interface.

5.0 RESULTS & DISCUSSIONS

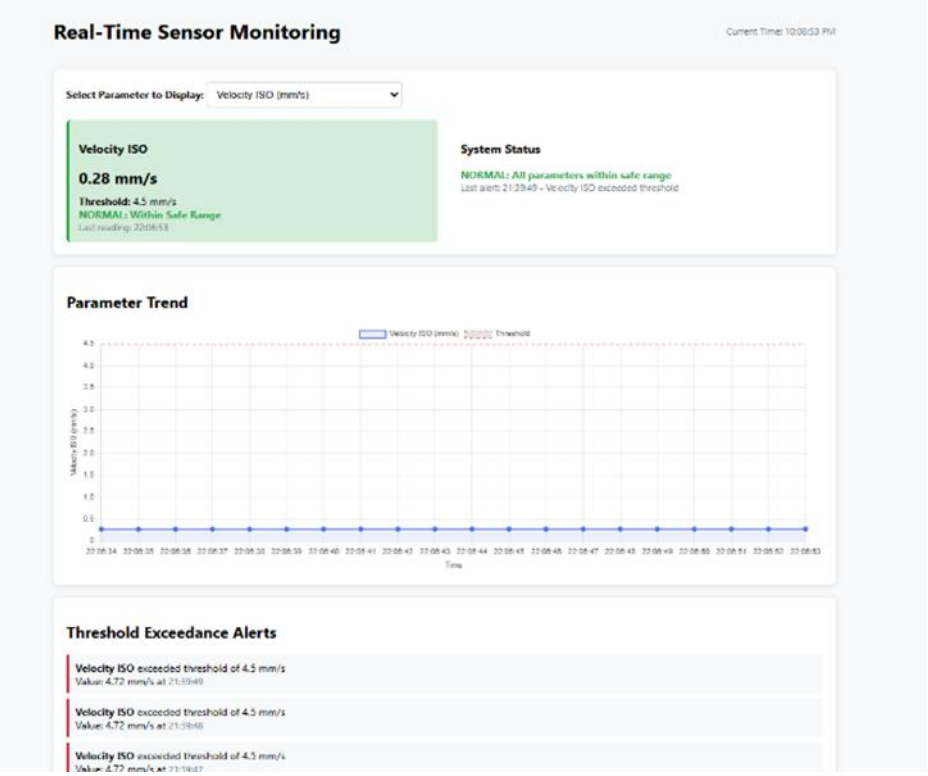
5.1 Visualization of Data in the Web Monitoring System Dashboard

The developed web monitoring dashboard employs a minimalist layout with high-contrast colour schemes, prioritizing clarity and rapid information processing. Several key components are parameter selector, status cards, trend visualization and alert history panel.

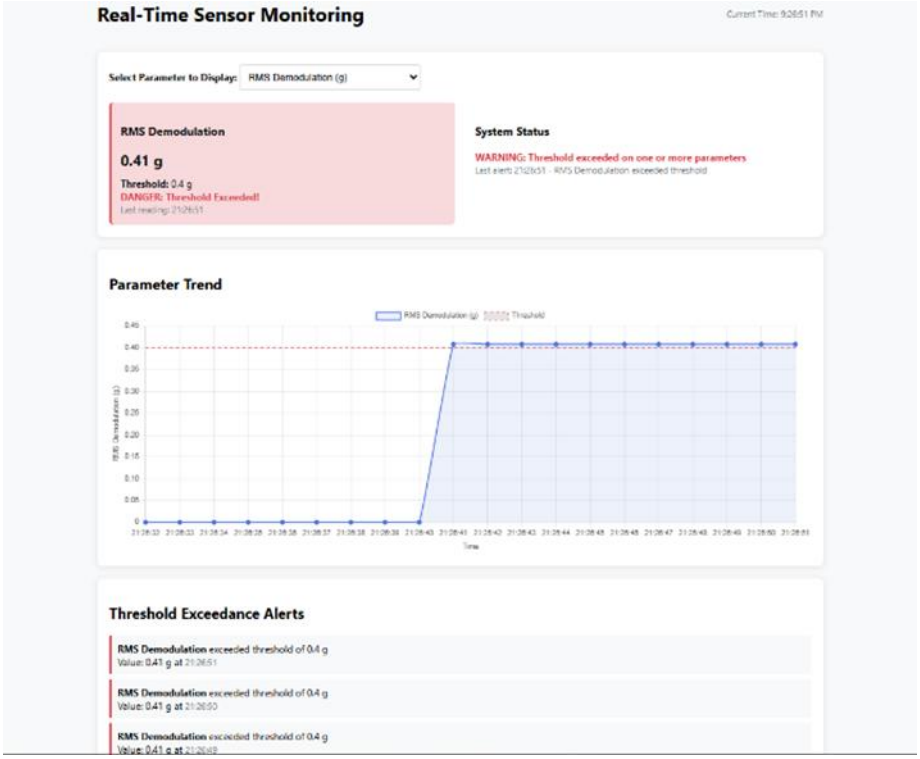
ISSN 2180-1053 e-ISSN 2289-8123 Vol.18 No.1

The parameter selector is a dropdown menu which enables dynamic switching between parameters and accommodates role-specific monitoring needs. The status cards are placed central to the interface of the dashboard featuring dual-panel visualization. The primary panel displays real-time parameter values alongside dynamically color-coded safety indicators; green when the value of the selected sensor is below the threshold (safe operational state) and pulsating red if the value exceeds the threshold (danger state). The trend visualization shows the time-series graphs which superimposes live sensor readings against the threshold baselines. This enables immediate identification of degradation patterns, such as escalating vibration amplitudes preceding cavitation events. Complementing this, the alert history panel maintains a scrollable log of timestamped threshold violations, providing auditable records for root-cause analysis and enables rapid correlation with operational events.

The developed web-based monitoring system successfully integrates data acquisition, dynamic visualization, and user-centric design. The dashboard employs a user-friendly interface designed to help maintenance personnel with a simple, grid-based layout, color-coded alerts, and progressive disclosure of data. Figure 2 shows the sample of the developed dashboard displaying the dataset from maintenance-prone centrifugal pump that mimics real-time monitoring: Figure 2(a) shows a normal operational status indicated through green colour coding and clear status messages; Figure 2(b) displays critical threshold exceedances with red pulsating alerts and clear warning messages.



(a)



(b)

Figure 2. Sample display of the developed web monitoring system dashboard for centrifugal pumps using established dataset: (a) Safe state for the rotational speed, (b) Danger state occur in rms demodulation

5.2 Visualization of Threshold Violation

The dashboard's visualization data was thoroughly checked with the original sensor readings of maintenance-prone centrifugal pumps to ensure accuracy and reliability. The monitoring system sequentially retrieves sensor measurements from the established dataset (*sensor_data.xslx*), using a separate index file to track the last processed row. During testing, the index file consistently maintained accurate positioning, and every value displayed on the dashboard perfectly matched the corresponding value in the source file. Figure 3 shows the sample of rotational speed data visualization and its breach of threshold. The developed web monitoring system successfully identified the threshold breach displaying danger status in red and managed to capture the time and value.

Historical alert logs showing the threshold violation were cross-checked to ensure that every threshold exceedance of more than 4.5 mm/s recorded in the source file was mirrored in the dashboard's alert. This verification was performed through manual inspection of 100 data points across 2 parameters, confirming 95% accuracy in data representation with only a slight delay of about 1 second between source information fetching and dashboard display. The sample display of the threshold exceedance for the rotational speed (Velocity_ISO) and demodulation (RMS Demodulation) is shown in Figure 4. The alert history panel helps to record each event of threshold violation for effective alerting system and early prevention of possible downtime. Positive feedback was received from users from the trial of this monitoring system, highlighting the system interface is simple and user-friendly with a straightforward layout allowing users to navigate with ease. The dropdown option for

ISSN 2180-1053 e-ISSN 2289-8123 Vol.18 No.1

selecting monitoring parameters was described as intuitive and easily understood, contributing to an overall smooth user experience. Additionally, the graph display was praised for its clarity and level of detail, making it convenient for users to interpret sensor data.

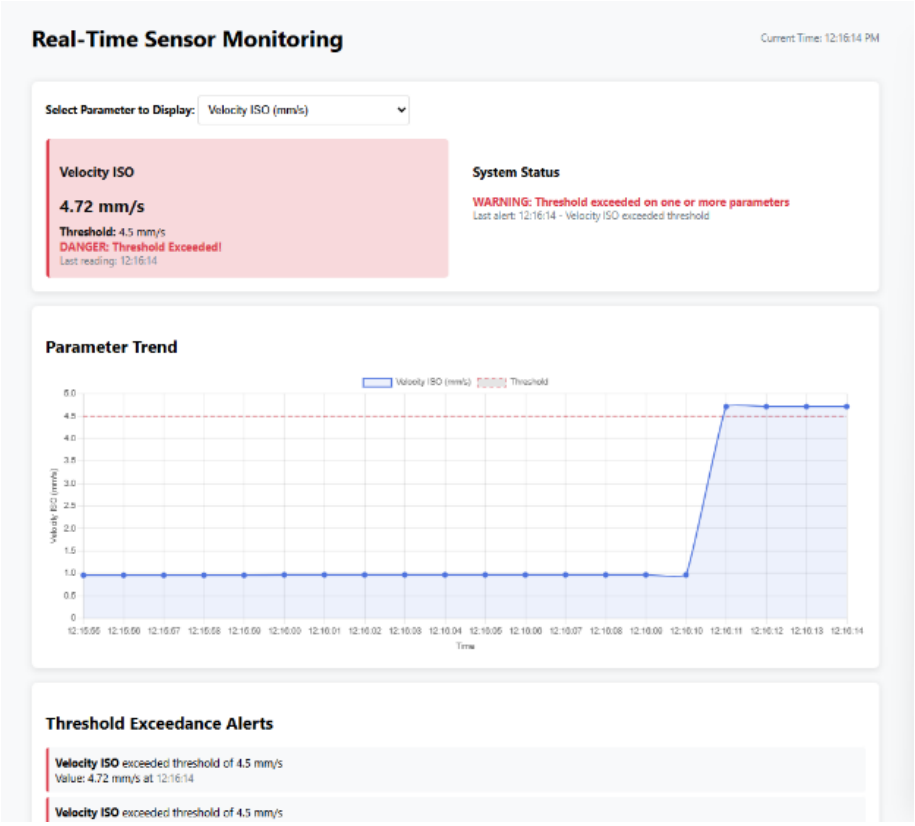


Figure 3. Dashboard of web monitoring system for the rotational speed (Velocity_ISO) exceeding threshold with the parameter trend (solid line-sensor reading, dashed line-threshold baseline)

Historical alert logs showing the threshold violation were cross-checked to ensure that every threshold exceedance of more than 4.5 mm/s recorded in the source file was mirrored in the dashboard's alert. This verification was performed through manual inspection of 100 data points across 2 parameters, confirming 95% accuracy in data representation with only a slight delay of about 1 second between source information fetching and dashboard display. The sample display of the threshold exceedance for the rotational speed (Velocity_ISO) and demodulation (RMS Demodulation) is shown in Figure 4. The alert history panel helps to record each event of threshold violation for effective alerting system and early prevention of possible downtime. Positive feedback was received from users from the trial of this monitoring system, highlighting the system interface is simple and user-friendly with a straightforward layout allowing users to navigate with ease. The dropdown option for selecting monitoring parameters was described as intuitive and easily understood, contributing to an overall smooth user experience. Additionally, the graph display was praised for its clarity and level of detail, making it convenient for users to interpret sensor data.

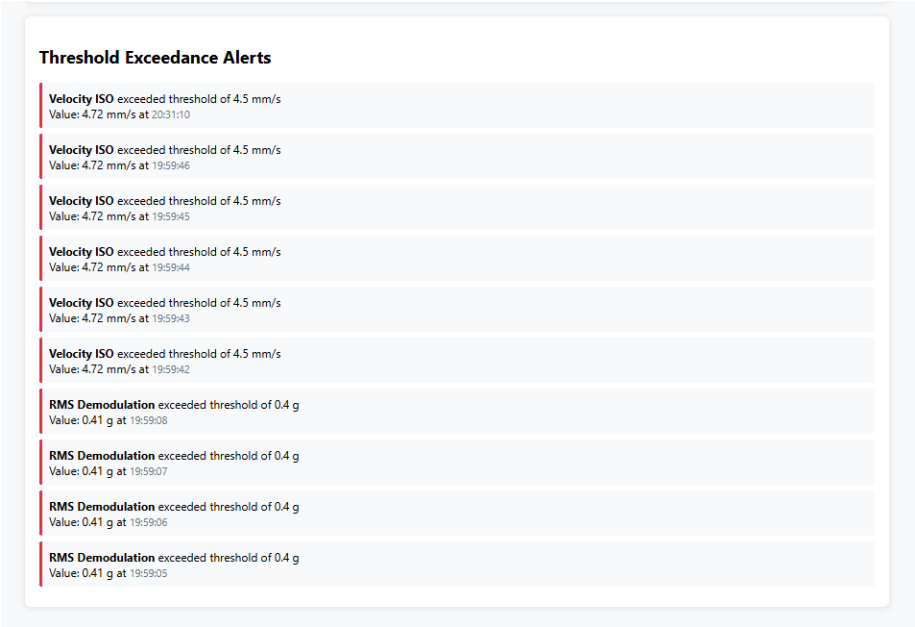


Figure 4. Alert history panel displaying threshold violation in the dashboard of web monitoring system

6.0 CONCLUSION

In conclusion, this study successfully developed a web-based monitoring system for the maintenance of centrifugal pumps using Python, particularly Django which provides a user-friendly platform that enables maintenance personnel to access sensor data and receive timely alerts for potential faults. The implementation of advanced sensor data processing and automated threshold-based warnings has demonstrated the system's ability to improve operational efficiency, reduce downtime, and enhance the reliability of centrifugal pump maintenance operations. For future work, the system can be expanded by integrating live data acquisition from IoT-enabled sensors with known sensor location, enabling real-time monitoring beyond pre-recorded datasets. Additionally, incorporating machine learning algorithms for predictive analytics, dynamic threshold adaptation, and automated maintenance scheduling would further enhance the system's capabilities. Broader application to other types of industrial machinery, as well as the development of mobile-friendly interfaces and multilingual support, could make the platform even more accessible and versatile in diverse industrial environments.

ACKNOWLEDGEMENT

The authors gratefully acknowledge Muhammad Hafeez bin Muhamad Ramizal for his invaluable support in the coding development of the monitoring system. Furthermore, appreciation is extended to International Islamic University Malaysia for the support and the provision of computational facilities.

REFERENCES

[1] S. Singh, " Impact of color on marketing", *Management Decision*, vol. 44, no. 6,pp. 783–789, 2006.

[2] K. Reinecke, and K. Z. Gajos, "Quantifying visual preferences around the world", *Conference on Human Factors in Computing Systems - Proceedings*, pp. 11–20, 2014.

[3] C. D. Wickens, S. E. Gordon, and Y. Liu, *"An introduction to human factors engineering", PHI Learning*, 2011.

[4] U. Patkar, P. Singh, H. Panse, S. Bhavsar, and C. Pandey, "PYTHON FOR WEB DEVELOPMENT", *International Journal of Computer Science and Mobile Computing*, vol.11, no.4, pp. 36–48, 2022.

[5] K. Aldrawiesh, A. Al-Ajlan, Y. Al-Saawy, and A. Bajahzar, "A comparative study between computer programming languages for developing distributed systems in web environment", *ACM International Conference Proceeding Series*, vol. *403*, pp. 457–461., 2019.

[6] D. J. S. Chong, Y. J. Chan, S. K. Arumugasamy, S. K. Yazdi, and J. W. Lim, "Optimisation and performance evaluation of response surface methodology (RSM), artificial neural network (ANN) and adaptive neuro-fuzzy inference system (ANFIS) in the prediction of biogas production from palm oil mill effluent (POME)" ,*Energy*, vol. 266,pp. 126-449,2023.

[7] P. Mallioris, E. Diamantis, C. Bialas and D. Bechtsis, "Predictive maintenance framework for assessing health state of centrifugal pumps", *IAES International Journal of Artificial Intelligence*, vol. 13, no. 1, pp. 850–862, 2024.