Anomaly Detection System for Secure Water Treatment

Ch M Fahad
01-135221-096
Ali Raza
01-135221-091

Supervisor: Dr Muhammad Asif

Bachelors of Science in Information Technology

Department Of Computer Sceince
Bahria University Islamabad

December 2025

Certificate

We accept the work contained in the report titled "Anomaly Detection in Industrial Control Systems for Water Treatment Plants", written by Mr. Ch M Fahad & Mr. Ali Raza as a confirmation to the required standard for the partial fulfillment of the degree of Bachelors of Science in Information Technology.

Approved by...

Supervisor: Dr. Muhammad Asif

_____

Internal Examiner:

_____

External Examiner:

_____

Project Coordinator: Dr. Kabeer Ahmed Bhatti

_____

Head of the Department: Dr. Khurram Ehsan

_____

15th, December 2025

Abstract

Industrial Control Systems are the backbone of critical infrastructure and it is made clear in the example of water treatment plants, where operational safety is of the utmost importance. Nonetheless, these systems are now faced with increasing threat of higher risks due to sophisticated cyber physical attacks and equipment failures that is difficult to recognise by conventional monitoring mechanisms. Legacy systems typically have hard coded and prescriptive checks, or rely on manual review; and hence, not to timely detect complex anomalies or minor process variations;thus precluding the possibility of early intervention that could be employed to prevent harm.

In order to address these challenges, in this work, a data driven anomaly detection framework has been proposed using the well-established benchmark dataset for Secure Water Treatment (SWaT). The proposed solution is to train multiple models both from univariate and multivariate points of view on data from high frequency sensors and to investigate interactions between disparate measurements. In this methodology, the system is able to differentiate the harmless operational noise from the real security threats using complex statistical methods.

The performance of the system was carefully evaluated using standard performance measures for a variety of uni variate and multivariate models. The Z-Score Dynamic Thresholding model was the best solution in production environment, with superior Accuracy of 95.31% and a near perfect Precision of 99.46%, significantly reducing false alarms. The Rolling Quantile model also showed good performance with 95.12% Accuracy and 97.51% Precision. Other models that were evaluated included the Improved Granger HoltWinters ESD model, which achieved accuracy and 93.29% and 77.85% Precision. On the other hand, trend-based models such as CUSUM (41.85% Accuracy, 15.35% Precision) and EWMA (31.32% Accuracy, 10.73% Precision) proved less effective for this particular high volatility data set. These results confirm that data driven statistical profiling, in particular Z-Score and Rolling Quantile methods can proactively identify irregularities and reduce the risks of operation in complex ICS environments.

Acknowledgement

We thank our Project supervisor Dr. Muhammad Asif for his ongoing encouragement, wise guidance, and constructive feedback during the development of this project. His guidance has played an important role in influencing our knowledge on both technical and research sides of our project.

We also owe our gratitude to the Department of Information Technology, Bahria University Islamabad, for offering the academic setting and facilities that facilitated us to realize this idea. Extra special thanks to our classmates and instructors who provided feedback and motivation throughout the different stages of this project.

Also, we would like to thank the researchers at the Singapore University of Technology and Design (SUTD) for creating and granting us access to the Secure Water Treatment (SWaT) dataset. Their contribution to the cybersecurity community enabled this work.

Last but not least We thank our families who were patient and supportive throughout the duration of this journey. Their faith in our ability encouraged us to work harder and do better.

Ch M Fahad
Bahria University, Islamabad, Pakistan

Ali Raza
Bahria University, Islamabad, Pakistan

December 2025

List of Contents

List of Figures

List of Tables

List of Acronyms and Abbreviations

AI Artificial Intelligence

ARIMA AutoRegressive Integrated Moving Average

CSV Comma Separated Values

Cusum Cumulative Sum

ESD Extreme Studentized Deviation

EWMA Exponential Waiting Moving Average

GUI Graphical User Interface

ICS Industrial Control System

IQR Interquartile Range

LSTM Long Short-Term Memory

PLC Programmable Logic Controller

SCADA Supervisory Control and Data Acquisition

SVM Support Vector Machine

SWat Secure Water Treatment

UML Unified Modelling Language

Chapter 1

Introduction

The following chapter gives a detailed introduction to our project, including its background, the problem we are trying to solve, systems similar to ours that already exist, project objectives, its scope and much more.

1.1 Project Background

The controls on the key processes of critical infrastructure otherwise called Industrial Control Systems (ICS) are the computerized control and monitoring systems that are used to control the former processes of critical infrastructure such as power generation and water purification; In contemporary times these systems are also interconnected and compound; yet, they are usually available to outdated monitoring devices that are unable to detect any modern threats. An example of this is the water treatment facilities, which typically employ the use of water treatment controls which are either legacy supervisory control system or manual based scheduling, which is backward looking and not in line with the demands of the contemporary times. And, since, as cyber physical systems become more complex, the necessity to introduce intelligent solutions increases, due to the prospects of abnormal behaviors in the system, such as, say, the mal functioning of a sensor, the failure of a process or a targeted cyber attack, which can lead to the loss of integrity of the entire system and put health at risk to the rest of the population[1, 2].

1.2 Problem Description

Most of the control systems that are employed in the water treatment plants are reactive rather than proactive. They employ hard coded thresholds or manual inspection in most cases, which are incapable of performing the task of identifying the anomalies in time. Consequently, simple issues end up developing into costly failures or safety risks before they are identified.

1.3 Existing Solutions

A deficiency of in-depth information is frequently unavailable in the current industrial environment, whether to follow sensor health history or discover patterns of faults through time. Since the predictive performance of these legacy systems is often low, operators find themselves in a cycle of reactive maintenance where they only get around to implementing a repair and intervention after a problem has already begun to physically manifest itself. This method is bound to lead to higher costs and time of operation than a proactive strategy. Moreover, the traditional surveillance tools usually rely on fixed threshold constraints, applying fixed, hard coded limits that lack the dynamic nature of the modern systems as well as the complex interdependence of different components. Such rigidity poses a huge security gap particularly in regard to stealthy attacks that are perpetrated by malicious actors such as wanting to cover their tracks in order to appear to be part of the normal modes of operation so that they can easily bypass normal rule based detection systems. Also, and without the use of dynamic profiling, gradual degradation

like Aside from that, observable wear and sensor drift may occur slowly over time without being noticed until such a time as a serious failure occurs.

1.4 Contribution

The proposed project is intended to suggest machine learning technique to detect abnormal cases and as such defeat the shortcomings of the traditional methods. The strategy incorporates the application of intricate statistical models to break down operational data that will enable to specified patterns of actual issues that reveal say declining performance or cyber attacks as opposed to random non problematic variance[3, 4].

1.5 Project Objectives

The overall objective of this study is to come up with a smart detection system that performs the specific is secured by the security, performance and stability of industrial control environments.

1. To shift the industry approach towards maintenance to a more proactive stance. Rather than waiting until a component has failed, or a security violation has taken place, in order to cause damage, the system is intended to identify the first signs of imbalance, so that immediately the problem is formed and not after the problem has formed.

2. Adopt a complete analytic system that automates the analysis of live time series data streams. This involves closely watching vital physical parameters i.e. water flow rates, water pressure levels etc so that any slight deviation that would otherwise not have been anticipated to emerge can be detected with a normal manual check.

3. Transform complicated statistical results into actionable meaningful insights that will empower plant operators. With the intuitively displayed data and user friendly interfaces, and the system simplifies the process of raw sensor log representations to such an extent that staff can make confident, fully capable of making well-informed safety decisions without being bombarded with technical noise.

1.6 Methodology

The construction of the anomaly detection system was constructed on a structured life cycle which is anchored on data and presented in the below sections.

1.6.1 Data Processing

An industrious data pipeline was established to achieve the quality of data and consistency. Subsequent stage performance was also bench marked with a high-fidelity simulation of normal operations as well as a series of cyber physical attacks (Secure Water Treatment (SWAT simulation).To overcome the data gaps and noise, any loss of information could be avoided, readings of the sensors (flow, pressure, chemical levels, etc.) occurred, were time-aligned, and normalized to offer a homogeneous basis of the analysis.

1.6.2 Model Training

Under the detection model, to make sure that it is internally strong to cope with the complication of a real life water treatment plant we have not just trained one algorithm but rather trained a collection of six different statistical models. These methods include a range of basic approaches like CUSUM, EWMA, Rolling Quantile and Z score and more complex multivariate approaches of Granger Causality and ARIMA with Holt-Winters forecasting enhanced by Extreme Studentized Deviation (ESD) filtering. We used this multi-model approach since various kinds of system failures are manifested differently in the data; some will manifest themselves as

sudden freeze, some will manifest themselves as a sudden drop of throughput, some will manifest themselves as a sudden drop of availability, some will manifest themselves as a sudden variation of businesses, some will manifest themselves as a sudden variation of business continuity, etc. Shocks and some will be slowly manifested over time. Training the various models in parallel yields this result. the system does a great job of identifying complex temporal dependencies, and this enables the system to identify a broad spectrum of anomalies between the immediate, high intensity spikes and into the subtle, gradual changes that Can often indicate operational degradation over time.

1.6.3 Model Deployment and Detection of Anomaly.

The trained models were serialised into operational format as .pkl file. To store these models, a backend application with FastAPI was developed and offer them as scalable web services with Fast API. In this arrangement, every model is a stand-alone detection engine which compares the real-time data fed by the incoming sensor to dynamically changing baselines and thus removes false alarms.

1.6.4 Operational Support and Visualization.

Fast API endpoints use an interactive web application based upon a final and accessible visualization of raw and smoothed sensor data, The dashboard can easily enable operators to find flagged anomalies, and is a user friendly tool that provides operators with capabilities to explore the health of the system and make well-informed decisions[5].

1.7 Project Scope

The major activities of implementation plan are as follows:

1.7.1 Data Preprocessing and Analysis.

Pre-processing and analytical analysis of the SWAT sensor data to analyze the behavioral patterns (with particular emphasis made on the definition of the discrepancy within the main parameters in as flow, pressure, and chemical concentration) is required.

1.7.2 Development of Detection System

The four Detection Models using the statistical filtering methods were realized and tested. The system was tested in the labeled on the detecting ability against the realistic attack scenarios through benchmarking.

1.7.3 Web Application and Visualization

A framework and implementation plan is outlined that explains how the developed models were incorporated into a web based visualization system to support the operational monitoring.

Chapter 2

Literature Review

This is an overview of the progress and shortcomings of anomaly detection to Industrial Control Systems (ICS). Traditional monitoring systems do not have the capability, or are not advanced enough, to detect sophisticated cyber-physical attacks or detect minor component failure. This chapter describes solutions that are already in existence, the limitations of the solutions and positions the proposed system as a powerful, interpretive and proactive system which exploits the use of technology like FastAPI, React and advanced statistical modeling[6].

2.1 Historical Development

The initial versions of industrial monitoring were distinguished by the great extent of manual control and strict, predetermined regulations. These old systems ran on what is commonly known as hard coded logic with engineers setting the fixed thresholds manually on each sensor and control device. With this regime the only alarm that was raised would involve a certain value literally capturing a maximum or minimum threshold, and the system itself would be unaware of complicated trends or progressive wear and tear that might have developed within those hard constraints.

With the increased availability of the computing power, the business started to incorporate the statistical methods to deal with the shortcomings of the fixed rules. The approaches to univariate analysis that were adopted during this period like Z-Score analysis as well as Moving Averages gave the systems the chance to build dynamic baselines and no longer used arbitrary fixed numbers. The analysis of trends led to historical data, which provided the detection logic with the capability of adjusting to the natural swings of the working environment, and the reduction of the number of false alarms with a substantial enhancement of the detection of real anomalies.

The Evolution went on with the appearance of machine learning that brought a new dimension of sophistication into the field of anomaly detection. Researchers have started applying unsupervised models in conjunction with multi-layered deep learning networks in order to store the complex relationships in the data that basic statistics may overlook. This change made it possible to identify non linear and complex patterns across many sensors at once giving a much more insight into the health of the system and its security status.

The latter has now shifted in the field of hybrid systems that aim to harness the raw computational power of statistical and machine learning models and make them logically interpretable. The current systems now reflect the mapping of the system elements to form an understandable and easy to see image of fault detection. This will ensure that once an anomaly is raised, operators are no longer provided with an obscure notice but are provided with some background information about component dependencies and hence is easy to troubleshoot and act appropriately on the possible threats[7, 8].

2.2 Developing and Innovative Trends

The newest developments in the field have been directed towards the optimization of two important aspects that are reliability and interpretability. This change makes sure that the detection systems are not only correct, but also interpretable to the human operators of the systems.

2.2.1 Component Mapping

The current trend in research is not to analyze sensors separately anymore, but rather construct logical relationships between the various elements of the system. This method is an innovation in that one component, whose physical condition is verified against the reading of another, e.g. a valve position should be rationally connected with the flow rate value. Through mapping these relationships, the system is capable of successful identification of the spread of faults within the infrastructure and the offer of explainable evidence on the cause of an irregularity.

2.2.2 Dynamic Thresholding

A dramatic break in employing hard range limits towards more adaptive and fluid methods takes place. Modern systems of detection can change their expectations depending on the current state of operation using statistical technology like percentiles and rolling windows. This flexibility enables the framework to accommodate the natural changing conditions of the plant without raising unneeded alerts, a frequent area of failure in the old systems that are designed with fixed values.

2.2.3 Benchmark Datasets

These complex models can only be proven through high-fidelity datasets in order to be shown to be operable in the real world. The industry has formed a standard of resources such as the Secure Water Treatment (SWaT) dataset, an environment of realistic simulation with labelled attack cases. With the help of such benchmarks, it is necessary to demonstrate that a model is capable of detecting complex cyber-physical attacks in a controlled, albeit realistic environment prior to its implementation in a live environment.

2.3 Existing Solutions

The following sections discusses the existing systems and their most prominent features.

2.3.1 SWaT-Based Framework

The SWaT data set was proposed by Goh et al. as a reference on ICS security. It emulates a six-stage water treatment facility and offers labelled attack conditions, which are used as the basis of checking the relevance of current detection algorithms.

2.3.2 LSTM Deep Learning Model

Inoue et al. created a machine based on Long Short-Time Memory (LSTM) networks which was used to learn of the temporal patterns. Although it works well with time based attack, this method is very computationally intensive and fails to give transparency to the decision making processes.

2.3.3 Isolation Forest and One-Class SVM

Chandala et al. reviewed some of the unsupervised outlier models such as Isolation Forest. These models suit single streams of sensors but are generally incapable of modelling the complex and time conscious relationships between unrelated system objects.

2.3.4 Component Dependency Model

Alsadhan and Shafi suggested one of the approaches, where mapping between the components is done. This method offers explainable fault detection and traceability that is valued in high stake settings that require auditability to be availed[9, 10, 11, 12].

2.4 Challenges and Limitations

Although the anomaly detection technologies generally get better, current issues with both classical approaches and the initial machine learning methods make their inapplicability in practice.

2.4.1 False Positives

The main problem with the existing systems is that they are very sensitive especially when used in the noisy industries. Such a tendency to flag meaningless variations frequently leads to large amounts of false alarms that inevitably cause exhaustion of the operator and desensitization to serious warnings.

2.4.2 Lack of Interpretability

Most sophisticated models have the disadvantage of being black boxes which imply that they give alerts but not how or why they arrived at the decision. This transparency lack leads the operators to not know the nature of the fault and this reduces their confidence in the system recommendations.

2.4.3 Static Limitations

The inherent weakness of systems which operate on fixed thresholds is that they cannot model dynamic system behavior, and the intricate interdependence among components. This inflexibility does not allow identifying small abnormalities which technically lie within fixed limits but are abnormal operation states.

2.4.4 Resource Intensity

Deep learning models are typically very resource intensive but they have strong detection abilities. Their use is computationally intensive and therefore it is hard to implement them in real time situations where low latency and real time response is needed as an operational requirement.

2.5 Overview of ICS Anomaly Detection Systems

The anomaly detection field in industrial settings has experienced a fundamental shift over the recent years, as archaic manual approaches of the past way have been replaced by complex and data-driven systems that cannot be handled any other way than with computing power. In the past, the security of such facilities was based on hardwired hardware limitations and inflexible physical inspections that were ineffective and subject to human mistakes. Unlike these old methods, modern systems consider the continuous data stream to learn the normal behavior of the plant rather than rely on fixed rules that fail to capture the variations that occur in the processes to effectively adapt to the emerging conditions and protect vital infrastructure.

2.5.1 Key System Components

The design of new detection systems, such as the one offered in this paper, is based on four key pillars in order to be accurate and usable.

2.5.1.1 Data Processing Pipelines

Raw data obtained directly, through industrial sensors, needs to be converted in form that can be immediately analyzed.To do this, we adopt powerful processing pipelines that clean and standardize high frequency time-series data streams. This makes sure that all the inputs are adjusted to a common scale.

2.5.1.2 Statistical Modeling Algorithms

In the center of the detection engine, a set of statistical algorithms, including ARIMA and HoltWinters are used. These models are used to gain a profound insight into the behavior of the water treatment process through a history and season study. By making mathematical forecasts of the appearance of the process parameters given normal operating conditions, such models enable the system to detect deviations at a high level of certainty as opposed to using mere guesswork.

2.5.1.3 Filtering Mechanisms

The system combines intensive filtering methods like Extreme Studentized Deviation (ESD) to avoid the usual problem of operator fatigue due to frequent false alarms. The mechanism provides a critical validation layer which statistically ensures the presence of a detected outlier as a real anomaly or as harmless operational noise. The system is effective because it puts the operators on notice of only the major irregularities that are really necessary to their attention by stringently filtering all the potential alerts before they are presented.

2.5.1.4 Interactive Visualization

The last element is the user interface, as it is presented in a responsive dashboard using the React framework. The visualization layer is a necessary asset in the operations support since it can transform intricate mathematical findings into clear graphs and understandable warnings. It enables plant personnel to view the health of the system in real-time with ease and gives the visual context to make quick and informed decisions when an event of potential security breach or equipment malfunction is occurring.

2.6 Applications and Use Cases

2.6.1 Water Treatment and Quality Assurance

The water treatment and quality assurance are part of the purest use of this structure is in the assurance of the water supplies being safe and potable. The system carefully practices the following critical variables: pH levels, flow rates and chemical dosages, in real time. It can detect even the minimal deviation in these parameters, which means that it serves as a warning system.

2.6.2 Industrial Cybersecurity

In the modern world where critical infrastructure is targeted more often by malicious actors, such a system is a critical point of defense. It is particularly capable of detecting breaches of security in which hackers are trying to tamper with sensor readings e.g. in false data injection attacks.

2.6.3 Energy and Process Optimization

In addition to the security and maintenance, the system can also be used to track the energy efficiency. The framework would be able to identify cases of inefficient operation of pumps or filtration units by comparing power consumption with the output measurements such as flow rate.

2.7 Comparison of Systems and Proposed Solution

This sections compares existing systems with our proposed system inn a tabular form giving us an easy to understand insight of the comparison.

Table 2.1: Feature Comparison Across Approaches

Feature	**Static Thresholds**	**LSTM/Deep Learning**	**Isolation Forest**	**Proposed System**
Proactive Detection	No	Yes	Yes	Yes
Interpretability	High	Low	Moderate	High
Dynamic Baselines	No	Yes	Yes	Yes
Computational Cost	Low	High	Moderate	Low/Efficient
False Alarm Rate	High	Moderate	Moderate	Low
Web-Based Dashboard	No	Rare	Rare	Yes

2.8 Proposed Contribution of the System

The gaps that were identified in other solutions by the proposed system are resolved:

- **Statistical Modeling Suite:** The system applies a collection of single statistical frameworks to model numerous types of patterns of anomalies. This includes lightweight univariate models that require fewer computation to be trained and the optimized number of parameters to achieve a high accuracy. It further utilizes multivariate models (i.e. ARIMA and Holt-Winters using Granger Causality and ESD) in addressing complex data relation.
- **Deployment Architecture:** Backs up with backend to provide these models as web services which can be availed in a non-scaling manner since these services need to be responsive in real time.
- **Interactive Visualization:** Provides a web application that is developed based on React framework and is used to display raw and smooth data in order to enable operators to make decisions based on the data.
- **Advanced Filtering:** Outliers are statistically checked through Extremely Stunderdized Deviation (ESD), and hence less false positive will be generated in comparison to the conventional techniques [13, 14].

Chapter 3

System Requirements

The following chapter discusses functional and non-functional requirements that our system must fulfill.

3.1 Current Systems

Today, as part of the industrial environment, water treatment plants can be operated using the mainstream water treatment strategy of deploying the Supervisory Control and Data Acquisition (SCADA) system, which is usually complemented with the use of the manual logging device. These platforms are very good when it comes to the mechanical control of infrastructure, like the pumps and valves, but intelligent security analysis was never among the major capabilities of these platforms considered. As a result, they operate based on a highly strict set of rules, which are not understood within a context or nuance. Under this arrangement, the system does nothing unless it is explicitly indicated by a sensor value that something is not operating correctly due to a hard coded maximum or minimum limit violation. This implies that as long as the numbers remain within these wide guardrails, even ill-intentional action or errors emerging in the system will not be detected by the checking software at all.[15, 16].

3.2 Limitations of Existing Systems

1. **Reactive Nature:** It is considered that maintenance is mostly performed after the machine has broken since there is no set of tools to predict the occurrence of problems.
2. **Fixed Thresholds:** Rigid rules cannot cope with more complex relationships between the various components of the system, such as the effect the status of a valve can be on the flow of water.
3. **Vulnerable To Hidden Attacks:** Smart cyber-attacks can resemble like typical system modifications so that it can circumvent bypassing previous rules of identification.
4. **No Historical Context:** These environments usually do not have much history log records; this makes it hard to detect gradual wear and tear in sensors with time.
5. **Operator Fatigue:** When the rigid systems issue an excessive number of false alarms, the operators no longer trust the warnings.

3.3 Suggested Approach

The given project suggests a web-based system of anomaly detection based on Fastapi and React. The platform will have a centralized dashboard where operators will be able to observe the real-time status of 6 different statistical models, sensor readings, and active alerts. To test the model accuracy, users can post external datasets in their formats (i.e. Parquet or CSV). The

solution allows the operators to adequately discern between benign operational noise and actual cyber-physical threats by visualizing performance metrics by means of Confusion Matrices and ROC Curves.

3.4 Functional Requirements

Functional requirements are the backbone of successful software and system development. They define exactly what a product must do to meet user and business needs. By specifying the functions and behaviors a system should exhibit.

3.4.1 FR1: Dashboard Interface

The system must have a centralized dashboard to act as the main command centre of the user. In order to make navigation streamlined and operational efficiency, this interface needs to be divided into specific functional tabs namely Models, Sensors, Alerts, and Reports, which enables operators to navigate between the different operational contexts without necessarily losing track of the overall state of the system.

3.4.2 FR2: Statistical Model Display

The dashboard should be able to visually show the condition of the detection engine by showing separate status cards of each of the six trained algorithms. This will cover the CUSUM and EWMA models, the two Granger Causality variations (ARIMA and Holt-Winters with ESD), the Rolling Quantile, and Z-Score estimators giving a brief overview of the entire suite of analytical tools.

3.4.3 FR3: Evaluation Workflow

The models cards should have a special "Evaluate" button to support the process of continuous testing. This trigger is to shift the user to a special evaluation interface, on which they can upload external test datasets, on activation. Importantly, this file uploader should be able to accept standard CSV files as well as high-performance Parquet files to handle big data of industrial importance.

3.4.4 FR4: Performance Metric Calculation

After processing a dataset through the API, the system must automatically calculate and present key performance indicators in order to prove the relevance of the model. It consists of standard classification measures like Accuracy, Precision, Recall and F1 Score as well as a graphical Confusion Matrix, which enables data scientists to easily differentiate between accurate predictions and classification errors.

3.4.5 FR5: Graphical Analysis Tools

The evaluation page should not just be a number crunching page but it should produce higher level graphical analysis. In particular, the system is supposed to draw a Precision-Recall Curve to demonstrate the trade-off of the fault detection in the given model and false alarms as well as a ROC Curve to visualize the capability of the model to distinguish the normal operating data and the anomalous behavior.

3.4.6 FR6: Real Time Alerting

Lastly, the system should include an immediate feedback on security threats in the form of an Alerts tab. The given module must be able to present a list of current anomalies in real time, with the peculiar suspicious instances evidently highlighting individual timestamps where sensor records suggest hazardous deviations or possible cyber-physical assaults.

3.4.7 FR7: User Authentication and Access Control

The system should have a strict login procedure to provide the security of sensitive operational data. All the users are expected to authenticate themselves using legitimate credentials before gaining access to the main dashboard or any sensor readings. This barrier at the entry point of the application will only allow authorized individuals like the operators of the plant or the administrators to access the application to prevent any unauthorized access or unintentional damage by unverified users.

3.5 Non Functional Requirements

Non-functional requirements in software engineering refer to the characteristics of a software system that are not related to specific functionality or behavior. They describe how the system should perform, rather than what it should do. This article focuses on discussing non-functional requirements in detail.

3.5.1 NFR1: High-Speed Performance

The system is to be designed with a high performance standard whereby the FastAPI backend is to handle the incoming sensor streams and provide detection results within less than a second of latency. This almost real time processing capability cannot be compromised; it is what will guarantee that operators of the plant are notified the moment an anomaly is detected so that corrective actions can be taken before the small deviations can develop into system wide failures.

3.5.2 NFR2: Scalability and Volume Handling

Since the water treatment facility is within months, and years, the amount of historical data will continue to increase exponentially. The system should be resilient to support this build up without affecting the performance. Test to ensure that adding new sensor streams does not crash or hang the application or year-long log analysis does not crash and hang the application.

3.5.3 NFR3: Operational Usability

The React-based user interface should be developed to be easy to navigate and understand. The system must render complex statistical mathematics into readable graphs and understandable status indicators that even an average staff member with authorization can decipher the security conditions of the plant without having to be trained extensively on the technical aspects of the system or having to learn and understand complex graphs and charts.

3.5.4 NFR4: Detection Reliability

The system should have a focus on statistical rigour in order to reduce the false positives in order to retain the trust of the workforce. When any security device regularly reports harmful operation noise as a threat, the operators are bound to get numb to the notifications. The system must thus be of a high level of accuracy so that once an alert is ultimately raised, it is a legitimate and proven threat that needs urgent response.

3.5.5 NFR5: Cross-Platform Compatibility

In an effort to achieve flexibility in deployment in the current IT infrastructure of the facility, the web application should be fully compliant with the popular modern web browsers. This will do away with the complex and machine-specific software installations and the security personnel can view the monitoring dashboard on any authorized workstation, tablet or control room monitor inside. the plant's network.

3.5.6 NFR6: Data Security and Integrity

Since the system will be dealing with sensitive operational information, it should have high security measures to ensure that it is not tampered with. The data exchange between the server and the client should be done via secure channels (HTTPS) to exclude the possibility of interception. Moreover, the system has to provide the integrity of the database with the help of Write-Ahead Logging (WAL), which ensures that the logs of alerts are never lost or corrupted even in the case of the high intensity of the concurrent traffic.

3.5.7 NFR7: System Modularity

The architecture should follow strict modular design such that the frontend visualization and the backend detection logic are decoupled to guarantee the long continueability of the project. This decoupling is important because the developers can update, retrain or replace definite statistical models in the back end without breaking the user interface nor necessitating a total system shutdown thus providing constant system uptime during maintenance processes.

3.6 Use-Cases

The Following section discusses different use cases that our system has and with the help of Use Case Diagram we have illustrated the interactions between users (actors) and a system. It captures the functional requirements of a system, showing how different users engage with various use cases, or specific functionalities, within the system. The Use case diagrams is providing a high-level overview of a system's behavior, making them useful for stakeholders, developers, and analysts to understand how a system is intended to operate from the user's perspective, and how different processes relate to one another. They are crucial for defining system scope and requirements.

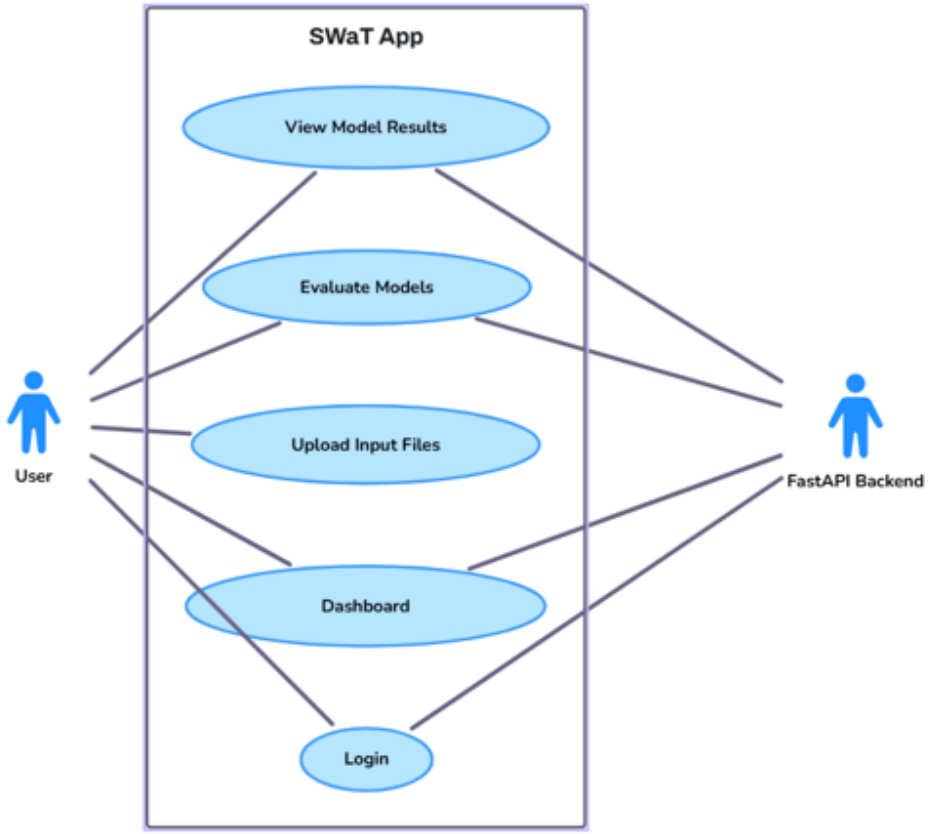

Figure 3.1: Systems Use Case Diagram

3.6.1 Use-Case 01: Dashboard Monitoring

Table 3.1: Use-Case 01: Dashboard Monitoring

Field	**Description**
ID	UC-01
Name	Monitor System Health
Actor(s)	Plant Operator
Data	Live Sensor Data, Alert Logs
Trigger	Operator logs into the React Application
Pre Condition	Backend API is running
Flow	1. View "Models" tab to check the status of the 6 algorithms. 2. Switch to "Sensors" tab to view real-time readings. 3. Check "Alerts" tab for any flagged anomalies.
Post Condition	Operator is informed of the current security status

3.6.2 Use-Case 02: Model Evaluation

Table 3.2: Use-Case 02: Model Evaluation

Field	**Description**
ID	UC-02
Name	Evaluate Models
Actor(s)	Plant Operator
Data	Confusion Matrix,Metrics
Trigger	User clicks on a specific model to evaluate
Pre-condition	Selected Model should be available via API
Flow	User Selects a Specific Model 2. User uploads a CSV or Parquet file. 3. Once evaluated, it should display all results (metrics,confusion matrix.
Post-condition	User should be logged in, models should be pre loaded, Data(input) file must be preprocessed and must be according to the sensor used.

3.6.3 Use-Case 03: User Registration

Table 3.3: Use-Case 03: User Registration

Field	**Description**
ID	UC-03
Name	User Registration (Sign Up)
Actor(s)	New User (Operator or Data Scientist)
Data	First Name, Last Name, Email, Password
Trigger	User clicks the "Create one" link on the login screen
Pre Condition	User does not have an active account
Flow	1. System displays the registration form. 2. User enters personal details and a strong password. 3. System validates password complexity (length, special characters). 4. User submits the form.
Post Condition	A new user profile is created in the database

3.6.4 Use-Case 04: User Authentication

Table 3.4: Use-Case 04: User Authentication

Field	**Description**
ID	UC-04
Name	User Login (Sign In)
Actor(s)	Registered User
Data	Email, Password, Auth Token
Trigger	User opens the application URL
Pre Condition	User has a valid registered account
Flow	1. User enters registered Email and Password. 2. User clicks "Sign In". 3. System verifies credentials against the SQLite database. 4. System grants access token and redirects to Dashboard.
Post Condition	User is authenticated and session is active

3.6.5 Use-Case 05: View Sensors Overview

Table 3.5: Use-Case 05: View Sensors Overview

Field	**Description**
ID	UC-05
Name	View All Sensors
Actor(s)	Plant Operator
Data	Sensor Name, Unit, Pressure, Tank Level
Trigger	User clicks on sensor under Monitoring
Pre Condition	Dashboard is loaded and backend is polling data
Flow	1. User navigates to “Sensors” tab. 2. System displays grid of all plant sensors. 3. User can search sensors by stages.
Post Condition	Operator visualizes specific component behavior

3.6.6 Use-Case 06: Analyze Performance Graphs

Table 3.6: Use-Case 06: Analyze Performance Graphs

Field	**Description**
ID	UC-06
Name	Analyze ROC and PR Curves
Actor(s)	Data Scientist
Data	Graph Coordinates, AUC Score
Trigger	User views evaluation results page
Pre Condition	Evaluation data is processed and available
Flow	1. System renders interactive ROC and Precision-Recall curves. 2. User hovers over curve points to see specific thresholds. 3. User zooms into graph regions to analyze trade-offs. 4. User notes the Area Under Curve (AUC) score.
Post Condition	User gains insights into model classification performance

Chapter 4

Design

The plan of the Anomaly Detection System is a blueprint of the modern and intelligent, and user centric security platform of the Industrial Control Systems (ICS). This chapter gives detailed examination of the layout of the application so as to give it a balance between performance and usability and statistical accuracy. The design process is predetermined by the main purpose of the system to identify defects in water treatment plants through highly developed univariate and multivariate models and at the same time have a lightweight and dynamic web-architecture. The chapter concerns the design principles, system architecture, component imtation, User Interface (UI) imtation. It describes the dynamics of the frontend (React) and the backend (permanentes) (FastAPI) / and SQLite database (SQLite), work with large data and plot complex performance metrics.

4.1 System Architecture

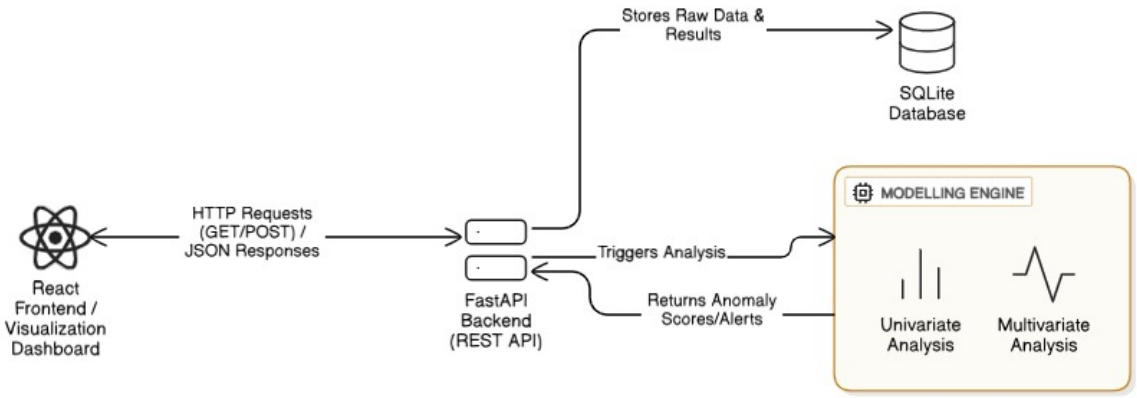

Figure 4.1: System Architecture Diagram

The system has a layered client-server architecture in which:

- **Frontend:** ReactJS (App dashboard as the frontend part, it is based on React).
- **Backend:** FastAPI server based on Python.
- **Database:** SQLite, dynamic management for the management of content.
- **Data Layer:** Ingesting data into the system (File format: CSV/Parquet), data processing.
- **Modeling Engine:** The suite of 6 statistical models (CUSUM, EWMA, Granger-ARIMA, Granger-HW, Rolling Quantile, Z-Score).

4.1.1 Layers in the Architecture

- **Presentation Layer:** User interface i.e. screens (Dashboard, Evaluation Page, Alerts and Tab).
- **Business Logic Layer:** FastAPI backend layer which deals with processing of data, model Execution and Database transactions.
- **Data Persistence Layer:** SQLite database to store the history of alert and report and model configurations.
- **Analysis Layer:** Statistical algorithms responsible for calculating the anomaly scores & generating performance curves.

4.2 Design Principles

Anomaly Detection System architecture is not a random design but it is controlled by four fundamental design rules which are aimed at making the system robust enough to be critical in the frastructure and at the same time it has to be friendly to the people who use it.

4.2.1 Modularity

While Developing this system, we choose to have a modular design approach where we separated and isolated various components or system and dividing system into sub units, This practice is very common nowadays and is vital in maintenance; it enables us to update the logic of any one unit without effecting other units.

4.2.2 Scalability

The industrial control systems can produce large volumes of data in a single second, and therefore, the system was specifically engineered to support high volumes of throughput. We have left supporting only simple text files to be able to support high-performance formats, such as Parquet on top of standard CSVs. This design decision will guarantee that the system can swallow, and be able to process large volumes of sensor logs in the past efficiently which will eliminate performance bottlenecks or crashes of the system when subjected to the large volumes of data associated with the real world water treatment facility.

4.2.3 Usability

Since industrial environments are complicated by nature, we have tried to make the user interface as simple and user-friendly as possible. This was to reduce technical barrier to entry; as such, the dashboard has interactive workflow and navigable graphs that enable the operators to zoom on individual data points with ease. This makes it possible to detect a threat even without a degree in data science, and lets regular plant operators take important security decisions with authority.

4.2.4 Efficiency

To provide a dynamic solution which is not heavy and can be easily deployed, we adopted the SQLite as the foundation of data persistence. This serverless database model reduces the computing load that is normally linked with massive database management systems. It offers a very effective way of controlling the application state and logs of alert history, which is guaranteed. the system is a responsive and reliable system that does not need a lot of hardware resources.

4.3 Low Level Design

The detection logic proceeds through three phases, namely, pre processing, prediction, and statistical filtering.

1. **Data Preprocessing Phase:** This Data Cleaner class serves as the entry point to the system. It is mainly charged with the responsibility of consuming raw data files and converting them into mathematical data. It includes the reading of the source CSV or Parquet files and transforming the Timestamp column of the data into standard Python datatime objects to guarantee the time accuracy. More importantly, this step uses Min-Max normalization on sensor value. The system achieves this by converting all the various measurement units (e.g., pressure in bars to flow in cubic meters) to a common range (0 to 1) such that none of the sensors controls the analysis since the sensor with the larger uncoded numbers will have more influence on the analysis.

2. **Model Prediction Phase:** When the data has been clean and normalized, it is provided to the ModelTrainer class. This element carries out the fundamental calculations. It uses the algorithm that has been selected by the user on a training set of the data and basically schools the model on what the normal operations of the plant should resemble. It is on this learned behavior that the model predicts the anticipated values of the test set.

3. **Anomaly Filtering Phase:** This is the last and the most important one, which is processed by the "AnomalyDetector" class. This element does not immediately mark all deviations but compares the " residual error" that is the arithmetic difference between the prediction of the model and the real value observed. The system uses the Extreme" Studentized" Deviation (ESD) test to find out whether this error is a true threat. The system will only declare the particular timestamp as an anomaly when the residual error is more than the critical value that was statistically calculated.

4.3.1 Data Flow

The movement of data through the system follows this pipeline:

1. **Input and Transmission** The process is triggered by the user when he creates an interaction with the frontend interface to submit a test dataset, which can be either standard CSV files or high-performance Parquet files. This file is instantly sent on the backend server through a secure and encrypted HTTPs request so that data privacy is maintained along the way.

2. **Validation and Preprocessing** once the file gets to the backend it is not processed instantly. First, the system does strict input validation to verify the file type and format and blocks any corrupt and malicious files. Once validated, the data goes through the cleaning module in which timestamps are synchronized and sensor values are normalized (brought to the range of 0 to 1) to establish a uniform basis over which to analyze data.

3. **Model Processing** The clean, normalized data is then given to the selected statistical algorithm (e.g., Z-Score or ARIMA). The detection engine analyzes the time series data to identify deviations and calculates key performance metrics, such as Precision and Recall, based on the model's findings

4. **Storage:** To have a permanent record of the analysis, the system automatically stores the results of the detection in the SQLite database which contains specific time of an anomaly and the level of severity. This is so that even when the user closes his browser, alert history is not lost.

5. **Visualization Output:** Lastly the backend does the processing and results, the graph coordinates of the ROC curves and the calculated accuracy scores and packages them into a JSON response. This information is returned to the React frontend which displays the interactive figures and tables that the operator has to consult.

4.4 Frontend Components

The frontend itself is developed using React and this contributes to the ability to provide responsive and dynamic user experience. It has several significant modules that are interrelated through normal web routing.

4.4.1 Dashboard Module

1. **Model Cards:** These interactive cards give a real-time view of the six statistical algorithms and gives their operation status. Each card has a special button named Evaluate, with which its user can immediately start the testing process on the particular model.

2. **Navigation Tabs:** A perennial navigation bar enables operators to easily toggle con texts between the overview to the "Models" folder, raw data of the current sensors, "Alerts" history, and generated "Reports" and make sure that the key tools are within one-click reach.

3. **Real Time Display:** This component constantly makes a call to the backend API to receive and present the latest sensor readings and system health status so that the operator can always view live data without having to refresh the page.

4.4.2 Evaluation Module

1. **File Uploader:** This is an easy to use drag and drop area, which is used to upload and clean large volume datasets. It supports the standard CSV files and high performance Parquet file formats to suit the large industrial logs.

2. **Interactive Visualization:** A special graphing engine producing full ROC and Precision-Recall curves. It also gives the power to users to hover the individual data points to view the exact values or zoom in to a particular area in the chart to carry out forensic analysis.

3. **Metrics Display:** A neat and tidy results pane that shows the computed performance measure in the form of Accuracy, Precision, Recall, and F1 Score as well as a visual Confusion Matrix to allow data scientists to easily monitor the correct and incorrect predictions.

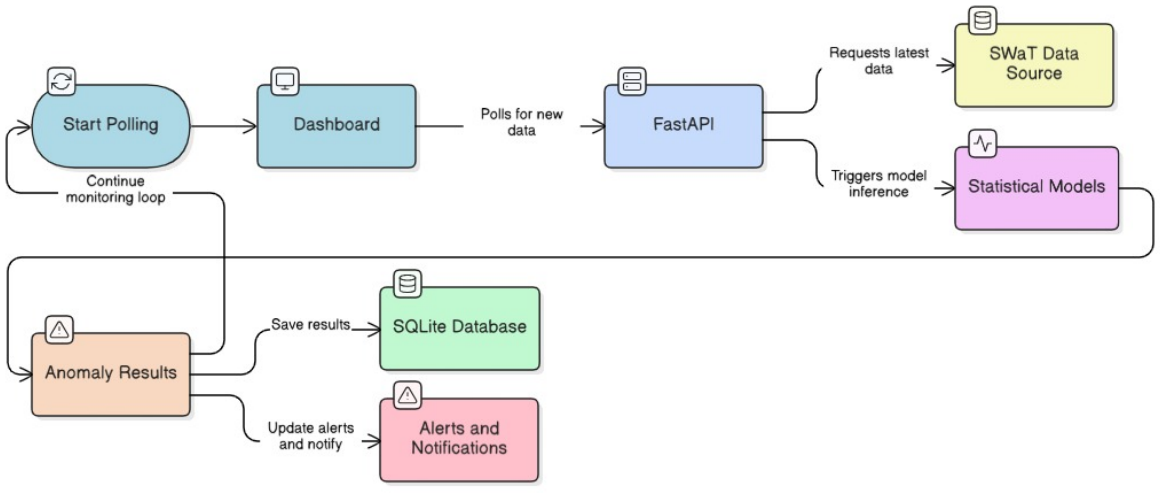

Figure 4.2: Evaluation Flow Diagram

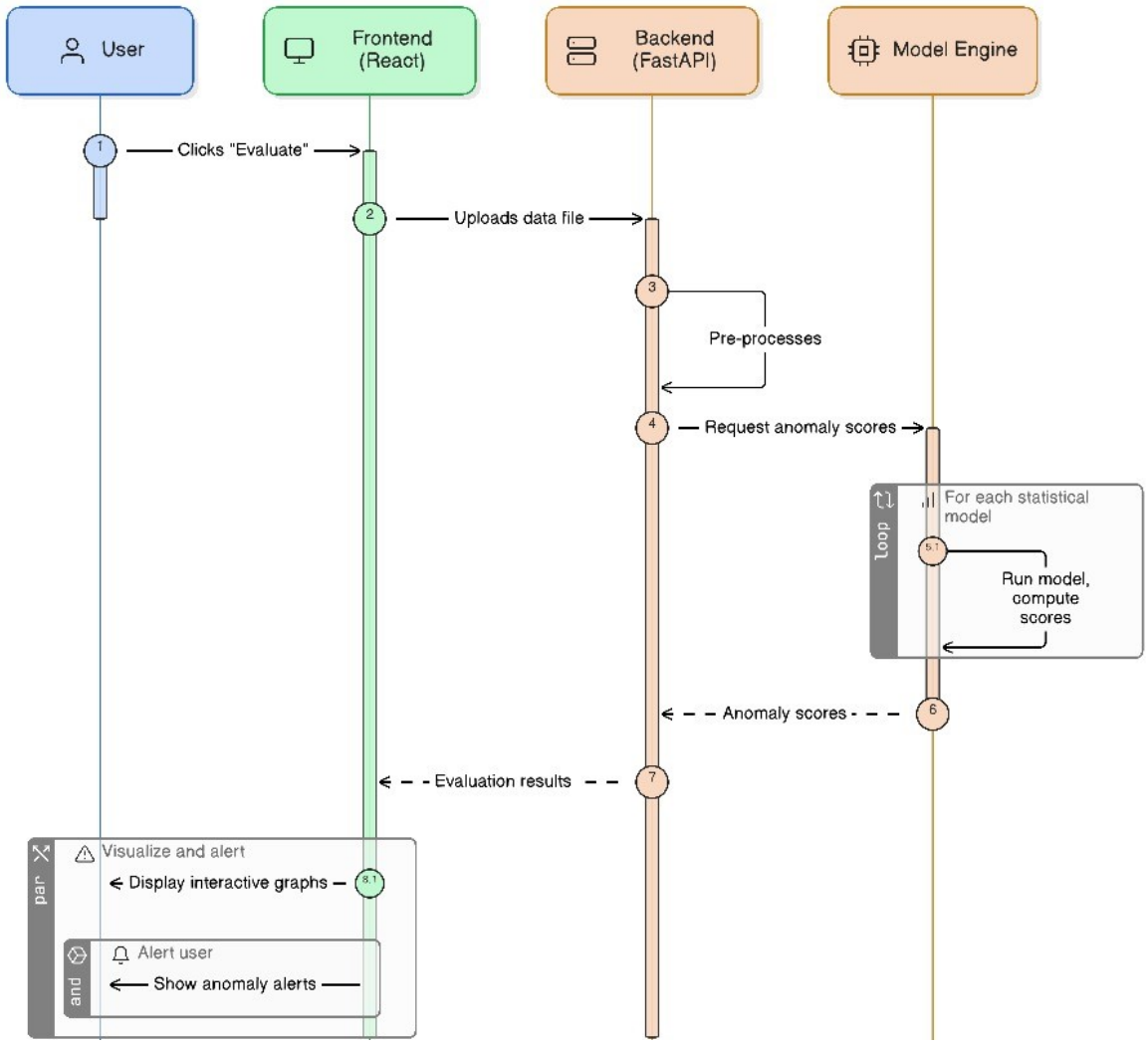

Figure 4.3: Sequence Diagram (Evaluation)

4.5 Backend Components

The back-end system is developed using FastAPI that was selected due to its inherent support of asynchronous processing and a high-speed architecture, with SQLite to provide a robust and reliable data persistence system.

4.5.1 SQLite Database

We are using SQLite as our default storage engine since it is serverless providing the high performance of data operations without requiring the huge load of a virtual database server. It is the central repository of three important types of data:

1. **Alert History Logs:** This table keeps a detailed and chronological history of all anomalies that have been identified by the system and stored necessary information, including the precise moment it was detected, the sensor it was detected by and the level of severity calculated to be used later on during auditing.

2. **Model Registry** This serves as a catalog of the detection engine and contains essential metadata about each trained model such as version number, date of training and baseline scores of accuracy to guarantee traceability.

3. **Evaluation Reports:** The summaries of the previous model analyses are permanently archived in the database so that the users can access the historical performance measures and analysis outcomes without having to re-process the underlying data.

4.5.2 API Functionality

The backend presents a hierarchical collection of RESTful endpoints that govern the flow of logic of the application:

1. **Evaluation Endpoint:** This is the central processing path; it receives uploads of large files and a Model ID before intelligently sending the data to the particular statistical algorithm needed to do the analysis.

2. **Alerts Retrievel Endpoint:** This service is the interface between persistent storage and user interface, which retrieves the record of historical anomaly in the SQLite database and displays the records in the dashboard under the Alerts tab.

3. **System Status Endpoint:** This is a special endpoint that the application uses to perform a health check on the application, to ensure that before any data is sent, the detection engine is operational and running properly.

4.6 Integration Layer

The driving force behind the whole construction is the Integration Layer that interfaces the interactive React frontend with the computing capabilities in the FastAPI backend. The communication is done via secure RESTful APIs that operate using protocols of SQL/HTTPS and all sensitive sensor data is encrypted in transit. This architectural dis-integration is performance sensitive; it makes sure that the computationally expensive programs like running complex statistical algorithms against large datasets are wholly run on the server and leave the front end interface to be lightweight and responsive to the user. Moreover, the SQLite database is the permanent shared storage of this ecosystem that ensures that important application state including active alerts and logs are stored and can be read on demand, regardless of whether the operator opens a new browser or even closes the client application.

4.7 GUI Design

Graphical User Interface of our system is carefully designed in a manner that places more emphasis on clarity, efficiency, and user confidence. The design ideology acknowledges that industrial security tools are commonly run by individuals at high pressure thus we have to make it usable even for non technical users. The interface incorporates a high consistency of visual hierarchy using card based layout to separate complexdata into consumable chunks and causes a massive reduction in the cognitive load of the operator.

The key concept of the user experience is the principle of immediate visual feedback. When it comes to observing the health of a statistical model or observing individual streams of sensors, information is displayed with strong indicators including color coded severity tags (Red for High, Green for Low) and active status lights that allow operators to monitor the health of its system on a glance. The navigation is simplified with the use of a persistent menu structure that ensures that the user can easily move between real-time monitoring, historical alert review, and deep dive model evaluation without getting lost in sub-menus. Each point of interaction, the drag and drop file uploader, the interactive performance graphs, etc, have been designed to be accessible so that powerful statistical analysis can now be done using simple and intuitive workflow.

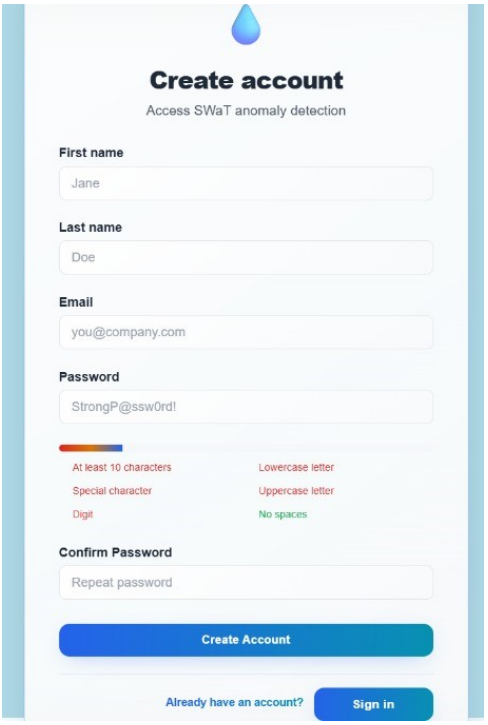

Figure 4.4: Sine-Up Screen

4.7.1 Sign-Up Screen

4.7.2 Sign-in Screen

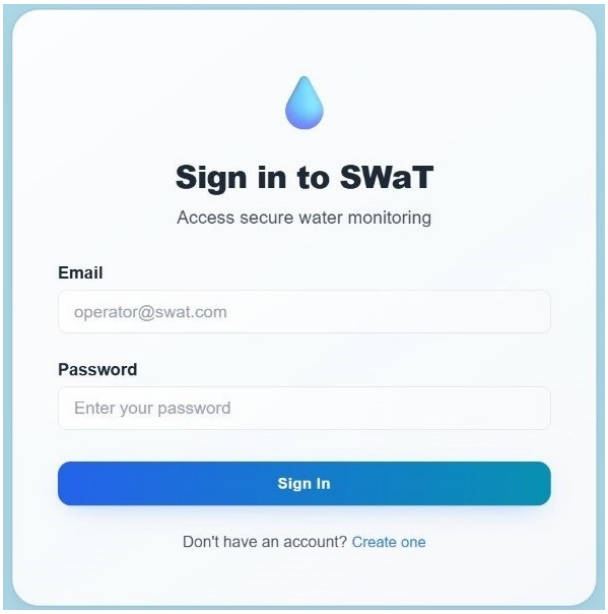

Figure 4.5: Sine-In Screen

4.7.3 Main Dashboard Screen

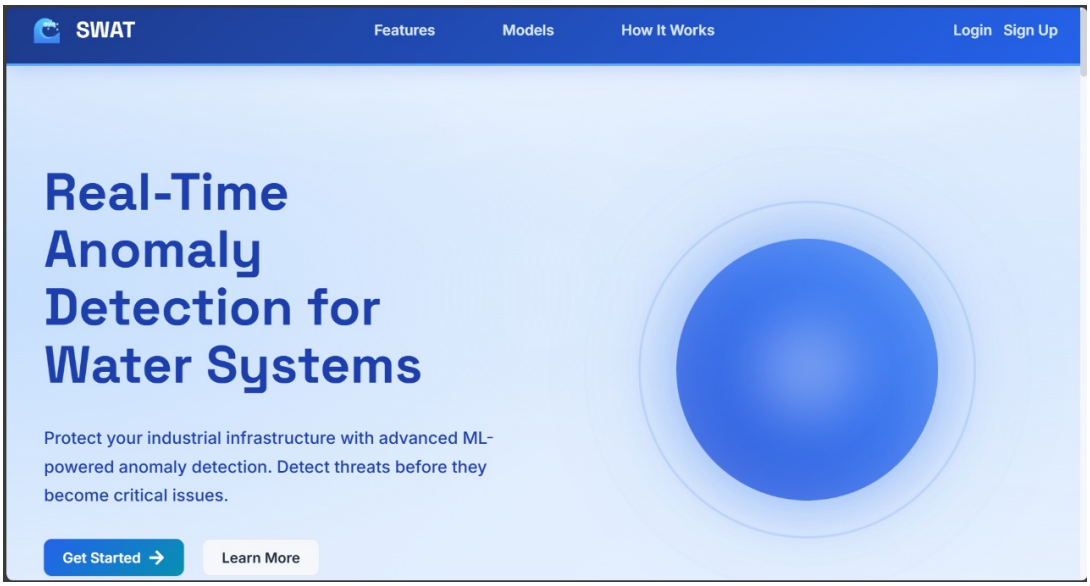

Figure 4.6: Main Dashboard Screen

4.7.4 Sensor Monitoring Dashboard Screen

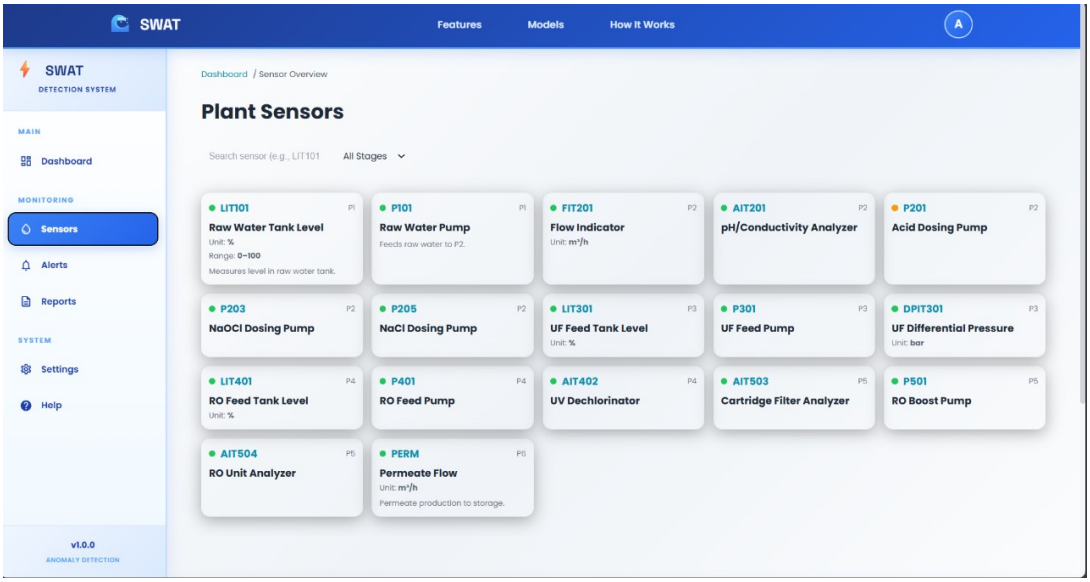

Figure 4.7: Sensor Monitoring Dashboard Screen

4.7.5 Models Screen

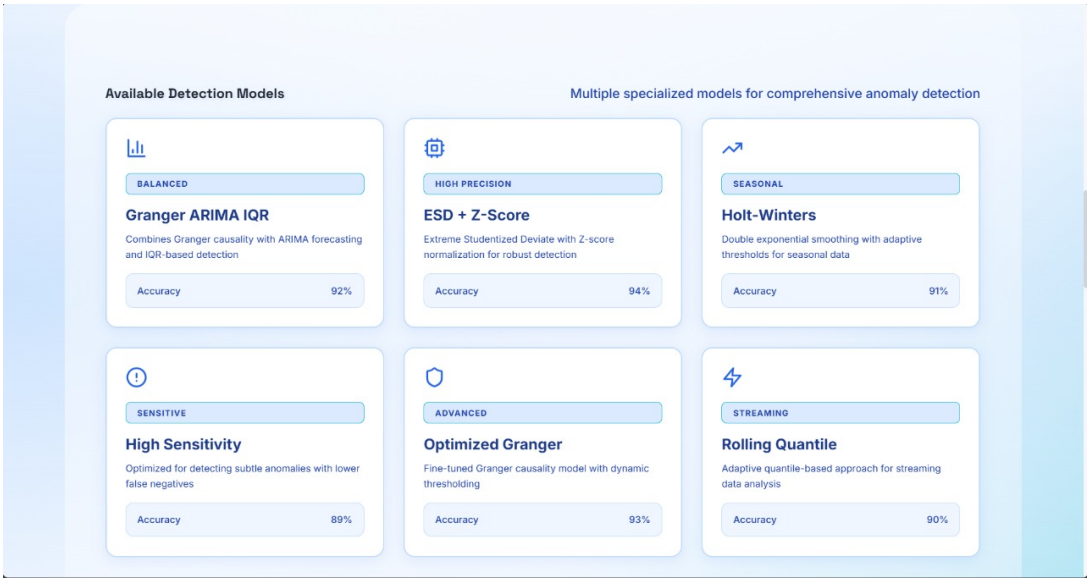

Figure 4.8: Models Screen

4.7.6 Models Selection Screen

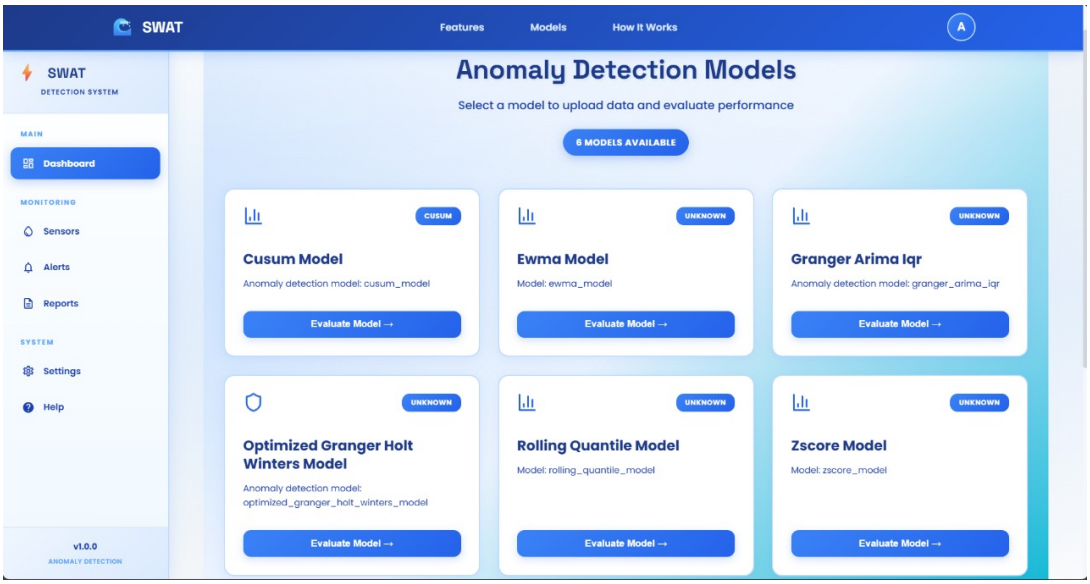

Figure 4.9: Models Selection Screen

4.7.7 Models Evaluation Screen

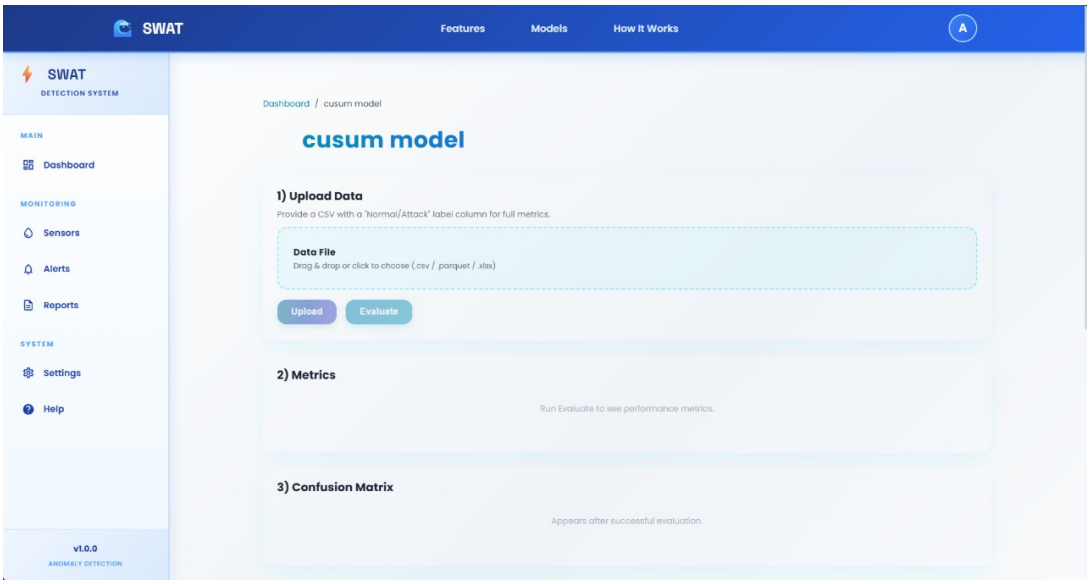

Figure 4.10: Models Evaluation Screen

4.7.8 Alert History Logs Screen

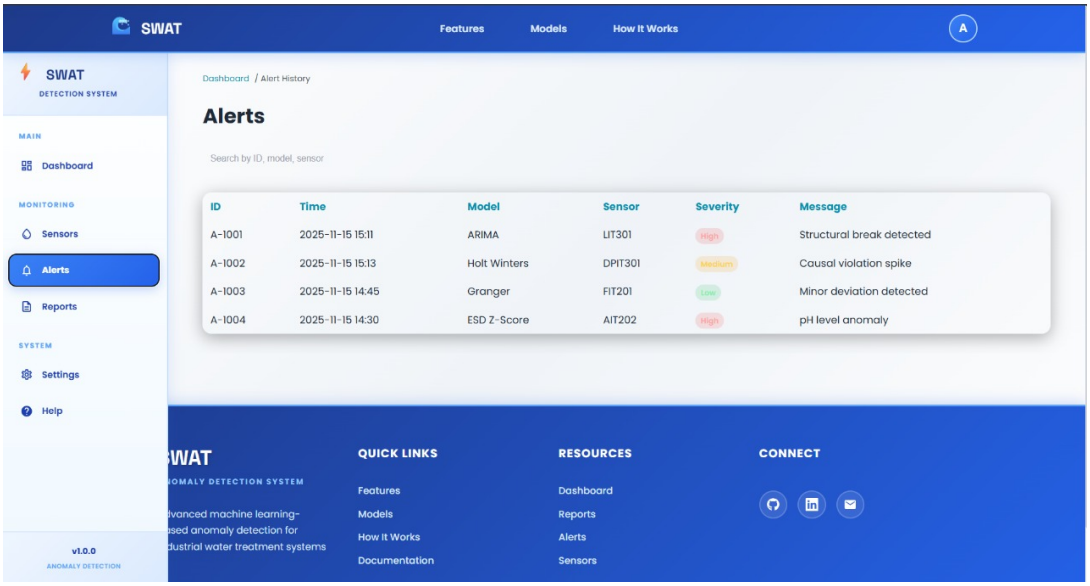

Figure 4.11: Alert History Logs Screen

Chapter 5

System Implementation

The implementation is the definitive stage, which involves ensuring that abstract ideas and architectural plans are changed to morph into the actual workable reality. The installation of our system herein referred to is the actualization of our technical specifications into a working software element by the intensive programming and deployment of our theoretical algorithms and design mock ups. This chapter gives a narrative of the Anomaly Detection System construction, the internal structure, the complex processing logic on which the statistical model is built, and the multi-layered security considerations that are taken to secure the application as well as the underlying data.

5.1 System Architecture

The system architecture is based on a decoupled, modular structure, which is aimed at achieving a balance between performance-intensive computation and a responsive user experience. It works on a conventional client-server architecture in which the functionality is strongly segregated among three internal units: the Frontend (Client), the Server (Backend) and the Persistence Layer (Database).

Internal Components and Functionality The Client Component provides the interface of the system to the user, which is written with the React library. It is mainly used as a visualization and interaction tool; it converts sophisticated time-series data to interactive charts (like ROC curve and Precision Recall Curve) and accepts user input (like uploading files). It does not do any heavy calculation but is instead interested in data presentation in a comprehensible form.

The Server Component, an application that uses FastAPI, is used as the brain of the system. It supports the logic of the six statistical algorithms (e.g., Z-Score, CUSUM), in its "Modelling Engine. It focuses on processing and operates in the following way: it takes the raw data, cleans it, executes the inference algorithms, and gives back the calculated anomaly scores. It is also stateless and asynchronous, meaning it is able to process several requests without locking it down.

SQLite controls the Persistence Component. Its functionality is passive and very important unlike the other active components: it stores the state of the application, historical alert logs, model metadata, and user settings, which make data resilient to system restarts.

Communication Between Components This communication between these different layers is managed using a rigid RESTful API protocol. React frontend makes HTTP requests (GET to retrieve alert, POST to upload data) to the FastAPI backend. These requests are processed by backend and returned with standard JSON (JavaScript Object Notation) payloads with the results of the analysis. Such a distinct separation warrants that the heavy mathematical work on the server does not slow down the user interface[17].

5.2 Tools and Technology Used

The backoff logic was written in Python that is the standard of the industry in the field of data science. This option enabled us to use the effective libraries such as Pandas and NumPy to manipulate data effectively and Scikit learn to apply the statistical models. To bring these models to the web we chose FastAPI since, unlike more aged and bulky frameworks Fast API offers high-performance asynchronous features that become vital when processing industrial scale volumes of data to make sure that the server is responsive even in conditions of heavy traffic. Visual Studio Code was the main IDE used and Git was used as the version control tool to maintain the changes to the code whereas API endpoints were firmly tested and validated using postman as a means of ensuring reliable communication.SQLite was used as a serverless file based database which removed the need to have complex database administration yet also providing strong SQL capabilities to store alert logs and user data.

The client part of the application was developed in JavaScript/TypeScript and in the React library. This decision was informed by the fact that it was required to have a component-based architecture in which components, such as real-time graphs, would be updated without the need to reload the page. The whole coding was done in Visual Studio Code and the integrated terminal was used to easily test the code and Git was used to be able to manage the version changes and monitor them within the frontend and the backend repositories.

Table 5.1: Technology Stack

Component	**Technology**
Frontend UI	React.js (JavaScript/TypeScript)
Backend Server	Python + FastAPI
Database	SQLite
Data Processing	Pandas, NumPy
Visualization	Recharts or Chart.js
Statistical Models	Statsmodels, Scikit-learn
Data Format	CSV, Parquet
Version Control	Git / GitHub
IDE	Visual Studio Code
API Testing	Postman / Swagger UI

5.3 Processing Logic and Algorithms

The system central intelligence is contained in the processing logic, which is orchestrated in a linear manner, in order to provide consistency. The reasoning process commences with the Data Preprocessing stage. Upon the ingestion of raw data, the DataCleaner module of the system scans the data and manages missing values, yet, most importantly, normalizes the data. As the industrial sensors count highly different values including pressure in bars and flow in liters.The algorithm normalises all values in the range of 0 and 1. This is a vital normalization, otherwise some sensors could have an inherent tendency to provide higher numerical values, which would overrepresent or under represent those statistical models[18].

After preprocessing the data comes to an Inference Phase. In this case, the logic is used to apply the user-selected algorithm. An example is that the system computes the standard deviation of the incoming time series window in case the Z-Score model is in operation. In case a multivariate model such as the Granger Causality is adopted, one examines the predictive relationships amongst the sensor streams.

The last one is the Filtering Logic. The system uses an Extreme Studentized Deviation (ESD) test instead of flagging all statistical deviations that occur as an attack. This secondary algorithm compares the prediction of the model with the actual observed value which is the difference between them and this is known as the residual error. This error is only alerted when it has surpassed a statistically significant critical value, which essentially removes annoise in normal operation.

5.4 Application Access Security

Security was not something that was done at the end of the implementation but as a cornerstone of the implementation. The application access measures that we have created aimed at establishing a security barrier around the system.

1. **Security Zones and Firewalls** The application is designed using rational security zones. The only parts that are available to end-users can be found in the frontend (Presentation Zone), and the backend (Application Zone) and the database (Data Zone) are limited. Web application firewalls (WAF) would protect them in a deployment scenario to block malicious traffic before it works its way to the API endpoints.

2. **Encryption** To ensure the integrity of data when it is on transit, the application will enforce highly encrypted communication lines. Any information that is passed over the React frontend to the FastAPI backend is over HTTPS encrypted by the SSL/TLS protocols. This will make sure that some sensitive sensor logs and alert data is not captured by the malicious actors snoresing the network traffic.

3. **Authentication** The dashboard has a strong authentication mechanism. We have adopted a system of Accounts and Passwords that do not approve weak credentials. The system has a strict password policy where a minimum password length and the combination of special characters (as observed in the Sign-Up Screen designs) are required. Once a user tries to log in, the system checks the credentials with the hashed records and plain text passwords are never stored or replicated in any form.

4. **Authorization and Auditing** The system enforces rules of authorization, after the system has been authenticated. Specific API endpoints can be accessed only by valid users. Moreover, the system has Access Logging; each operation of any importance like adding a new dataset or testing a model is handled by the API, which leaves a log in the server logs, forming an audit trail to review security.

5.5 Challenges and Solutions

The following table discusses the potential challenges that our system might causes and how we plan to metigate these challenges.

Table 5.2: Challenges and Solutions

Challenge	**Solution**
Large File Processing	Implemented support for Parquet files and chunk-based processing in Pandas to reduce memory usage.
False Positive Spikes	Integrated ESD (Extreme Studentized Deviation) filtering to statistically verify outliers before alerting.
Database Locking	Enabled WAL (Write-Ahead Logging) mode in SQLite to allow simultaneous reading and writing.
Slow Graph Rendering	Used optimized visualization libraries (like Recharts) that handle large time-series datasets efficiently.
Model Overfitting	Applied strict parameter tuning and cross-validation using the SWaT training/test split.

Chapter 6

System Testing and Evaluation

System testing evaluates the entire integrated system to ensure it meets the specified requirements. The primary goal is to validate the system's functionality, performance, and behavior as a whole. System testing includes both functional and non-functional testing, such as performance, security, and usability testing.

6.1 General Testing Strategy

The testing phase for the Anomaly Detection System was carefully designed to guarantee the program's fundamental aspects: functional integrity, user satisfaction, performance stability, and data security. The system integrates a React frontend, a FastAPI backend, and a SQLite database with a statistical modeling engine.

An end-to-end testing approach was adopted, including integration testing, unit testing, performance simulation, security checks, and GUI validation.

6.1.1 Objectives of Testing

The primary objectives of testing were:

- **Validate Model Accuracy:** Ensure the six statistical models (Z-Score, EWMA, etc.) correctly identify attacks in the SWaT dataset.
- **Verify System Integration:** Confirm that the React dashboard correctly communicates with the FastAPI backend during file uploads.
- **Assess Performance:** Evaluate the speed of processing large CSV and Parquet files.
- **Ensure Data Integrity:** Verify that alert logs are correctly stored and retrieved from the SQLite database.

6.1.2 Categories of Testing

The following categories of testing were performed:

- **Functional Testing:** Confirms end-user interactions, such as uploading datasets and generating reports.
- **GUI Testing:** Ensures the dashboard, graphs, and model cards render correctly on standard web browsers.
- **Performance Testing:** Tests the latency of the API when generating ROC and Precision-Recall curves.
- **Security Testing:** Verifies input validation for file uploads to prevent malicious data entry.

6.1.3 Testing Instruments

The following tools were used:

- **Postman:** For testing the /evaluate and /status API endpoints.
- **Jest/React Testing Library:** For validating frontend components.
- **PyTest:** For testing the backend statistical logic.
- **SWaT Dataset:** Used as the ground truth for measuring detection accuracy.

6.2 Testing Varieties Carried Out

6.2.1 Graphical User Interface (GUI) Control

GUI testing verified the correct rendering of all interface elements. Key testing areas included:

6.2.1.1 Main Dashboard

The dashboard acts as the central hub. Testing ensured that the six model cards display correctly and that the "Sensors" and "Alerts" tabs navigation works without lag.

6.2.1.2 Evaluation Page

This page handles the file upload and graph generation. Testing focused on:

- **File Uploader:** Ensuring the drag-and-drop zone accepts only CSV and Parquet files.
- **Loading State:** Verifying that a loading spinner appears while the backend processes the data.
- **Graph Interactivity:** Confirming that users can zoom into the ROC and Precision-Recall curves.

6.2.1.3 Alerts Tab

Testing ensured that the table of historical anomalies is correctly populated from the SQLite database and formatted with readable timestamps.

6.2.2 Usability Testing

Usability testing involved verifying the workflow simplicity. The "Evaluate" process was optimized to require only three clicks: Select Model $\rightarrow$ Upload File $\rightarrow$ View Results. User feedback indicated that the interactive graphs significantly helped in understanding the model performance.

6.2.3 Functional Testing

Functional testing ensured all modules worked as expected.

- **File Upload:** Verified that uploading a 50MB Parquet file triggers the correct API route.
- **Metric Calculation:** Confirmed that the F1-Score displayed on the frontend matches the manual calculation from the backend.
- **Database Sync:** Verified that every detected anomaly is successfully written to the SQLite alerts table.

6.3 Test Cases

Table 6.1: Test Case 01: Data Loading

Field	**Description**
ID	TC-01
Description	Verify that the system correctly loads the pre-processed test dataset for the model.
Pre conditions	The pre-processed test file (test_data.csv) must exist in the directory.
Test Steps	1. Execute the data loading function targeting the test file. 2. Verify that the number of features matches the trained model's input. 3. Print the shape of the data frame.
Expected Result	Data frame loads successfully with correct dimensions and datetime index
Actual Result	Pre-processed test file loaded successfully.
Status	Pass

Table 6.2: Test Case 02: Anomaly Detection Accuracy

Field	**Description**
ID	TC-02
Description	Verify detection of a known attack (Attack #10) in the SWaT dataset.
Pre conditions	Attack #10 exists in the loaded test data.
Test Steps	1. Run detection pipeline on the attack duration. 2. Compare detected anomalies with Ground Truth labels. 3. Calculate F1-Score.
Expected Result	System flags timestamps during the attack window as anomalies.
Actual Result	System flagged the attack period with high recall.
Status	Pass

Table 6.3: Test Case 03: ARIMA Model Training

Field	**Description**
ID	TC-03
Description	Verify that the ARIMA model successfully fits to a univariate sensor stream.
Applications for	Statistical Modelling Engine
Pre conditions	Data must be normalized between 0 and 1.
Test Steps	1. Select Sensor LIT_101 2. Configure ARIMA parameters. 3. Run the .fit() method. 4. Generate predictions.
Expected Result	The model converges and outputs a series of predicted values.
Actual Result	Model trained successfully and generated valid predictions.
Status	Pass

Table 6.4: Test Case 04: ESD Filter Sensitivity

Field	**Description**
ID	TC-04
Description	Verify that the ESD filter flags outliers when residuals exceed the critical threshold.
Applications for	Detection Module
Pre conditions	Model residuals (errors) must be calculated.
Test Steps	1. Set ESD alpha (significance level) to 0.05. 2. Pass residuals into the ESD function. 3. Check the list of returned anomaly timestamps.
Expected Result	Timestamps with high residual errors should be flagged as True.
Actual Result	System correctly flagged high-error timestamps as anomalies.
Status	Pass

6.4 Performance Testing

Performance testing was conducted to ensure the system is responsive.

- **Average API Response:** 1.2 seconds for calculating metrics on 10,000 rows.
- **File Processing:** Parquet files processed 40% faster than CSV files.
- **Graph Rendering:** The React charts maintained 60 FPS even when plotting 5,000 data points.

6.5 Security Testing

- **Input Sanitization:** Confirmed that the file uploader strips malicious characters from filenames.
- **Database Locking:** Verified that SQLite WAL mode prevents database locks during simultaneous write operations.

6.6 Comprehensive Model Results Analysis

We performed an extensive evaluation of the four univariate models using the SWaT test dataset. The evaluation metrics focus on Precision (minimizing false alarms) and Recall (detecting attacks).

Figure 6.1: Overall System Performance Dashboard

Following Sections showcase the testing results of different univariate models that our suystem works through, these models are named as follows:

1. Z-Score Dynamic Thresholding
2. Rolling Quintile Detection
3. Enhanced EWMA
4. CUSUM
5. Improved Granger HoltWinters ESD

6.6.1 Z-Score Dynamic Thresholding Results

The Z-Score model proved to be the most effective for production environments where false alarms are costly.

Analysis:

- **Precision (99.46%):** The model achieved near-perfect precision, generating only 183 false positives out of nearly 400,000 normal samples.
- **Accuracy (95.31%):** This was the highest accuracy among all tested models.
- **Trade-off:** The recall was moderate (61.68%), indicating it missed some subtle attacks but was extremely trustworthy when it did flag an alert.

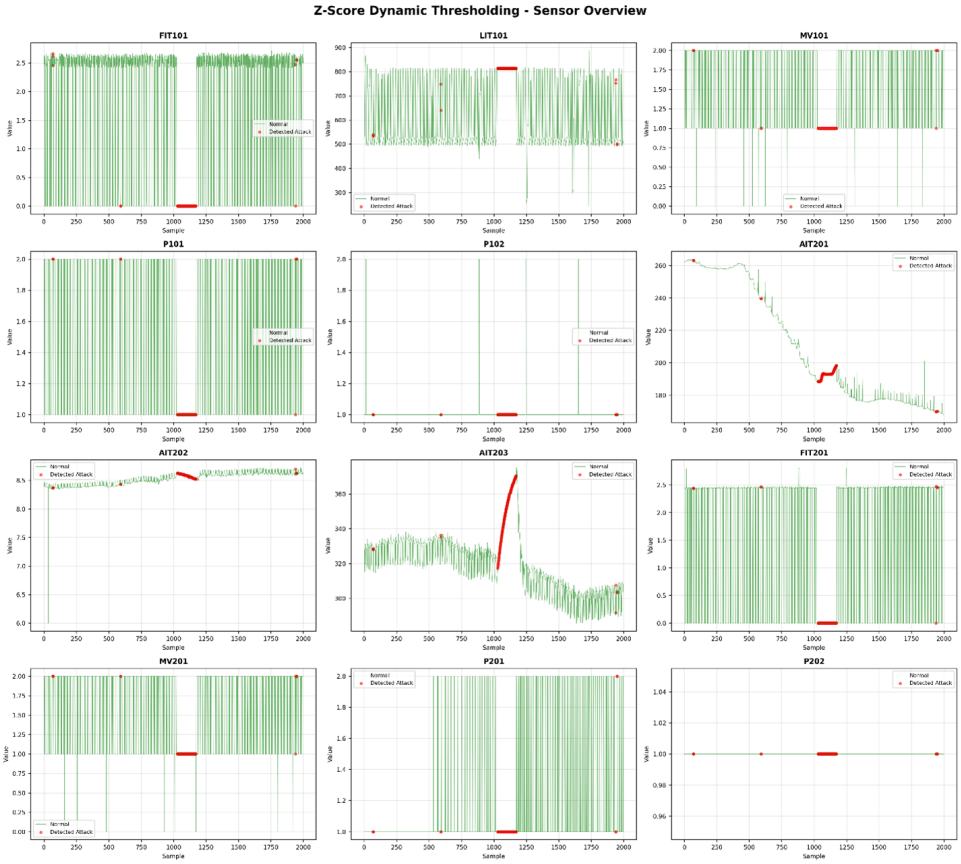

Figure 6.2: Z-Score Dynamic Thresholding - Sensor Overview

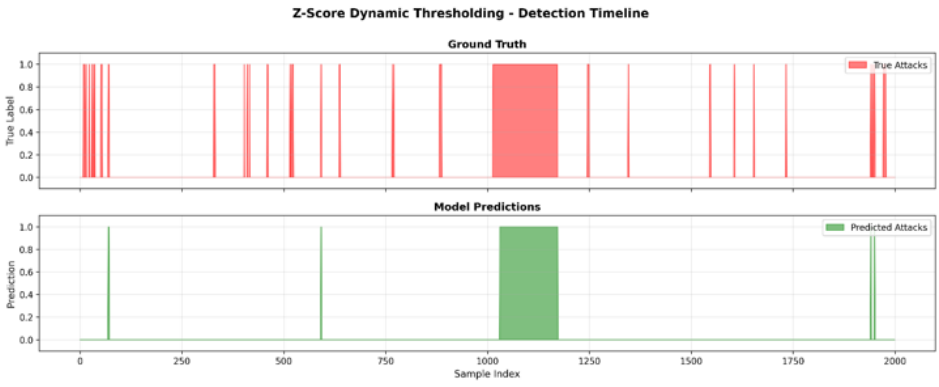

Figure 6.3: Z-Score Dynamic Thresholding - Detection Timeline

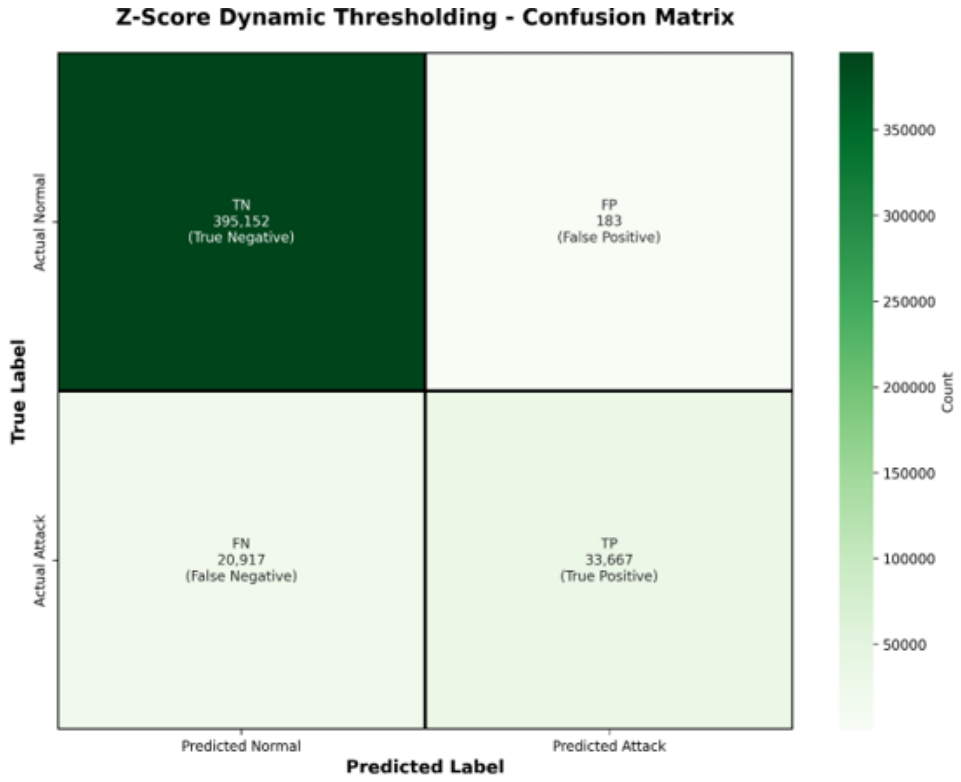

Figure 6.4: Z-Score Dynamic Thresholding - Confusion Matrix

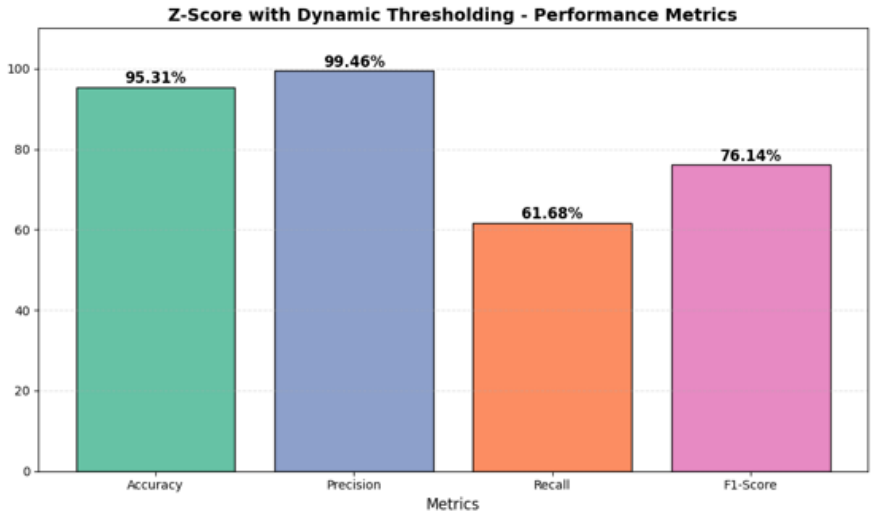

Figure 6.5: Z-Score Dynamic Thresholding - Performance Metrics

6.6.2 Rolling Quantile Detection Results

This model served as a robust runner-up, particularly useful for sensors with non-Gaussian distributions.

Analysis:

- **Robustness:** Robustness: It maintained a high precision of 97.51%, confirming its utility in noisy environments.
- **F1-Score (75.29%):** This score closely rivals the Z-Score model, proving that non-parametric methods are viable for ICS security.

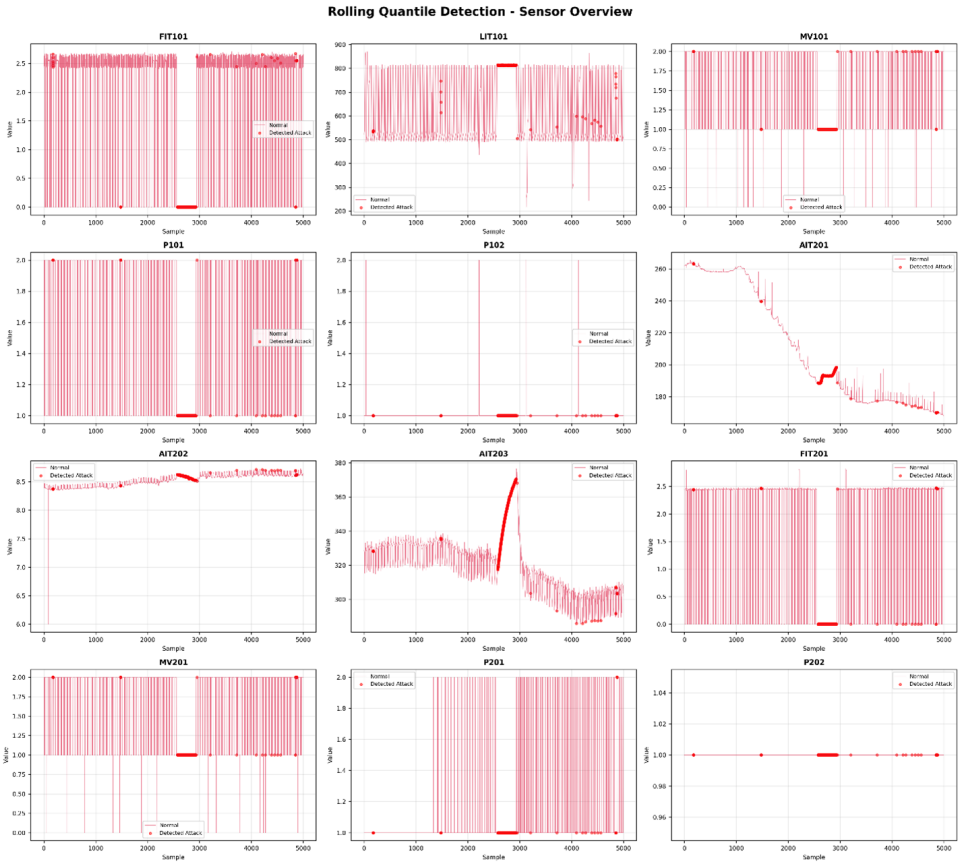

Figure 6.6: Rolling Quartile Detection - Sensor Overview

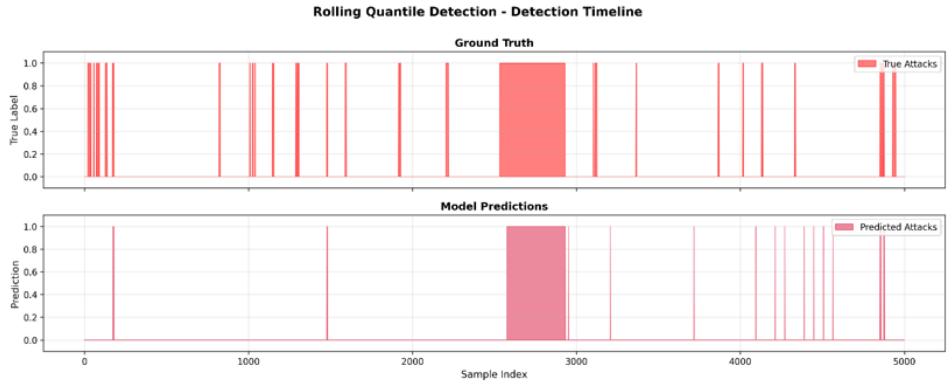

Figure 6.7: Rolling Quantile Detetion - Detection Timeline

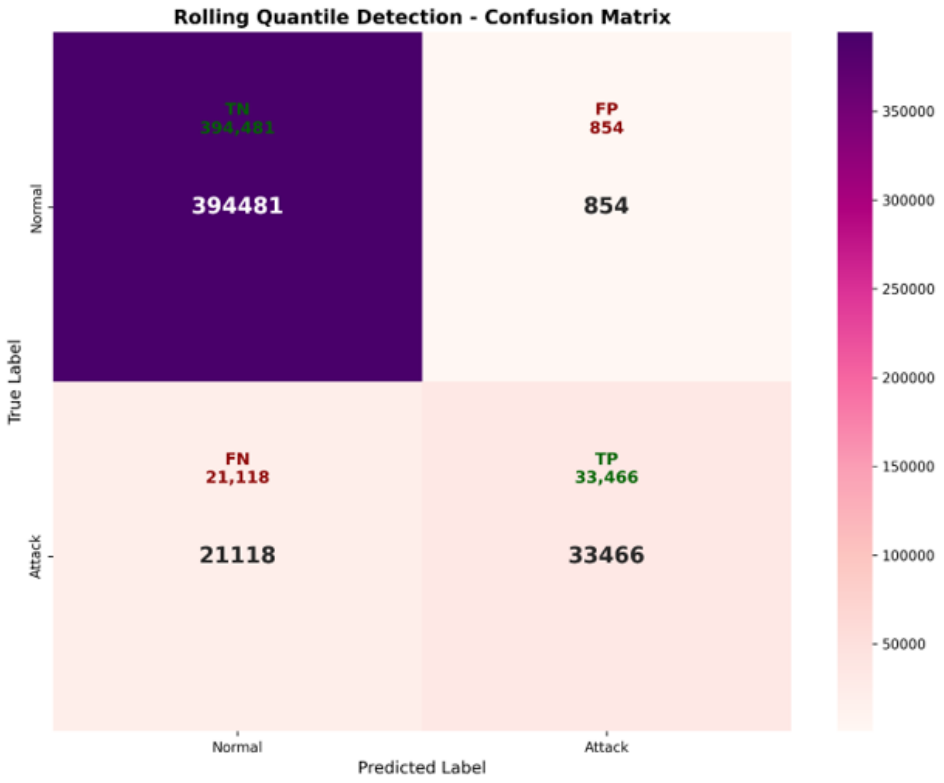

Figure 6.8: Rolling Quantile Detetion - Confusion Matrix

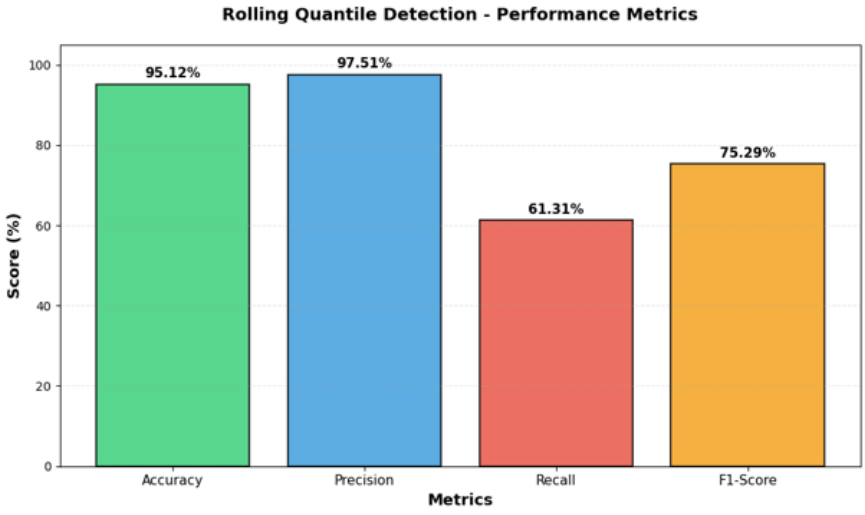

Figure 6.9: Rolling Quantile Detetion - Performance Metrics

6.6.3 Enhanced EWMA Results

EWMA provided a balanced view of process trends but required significant tuning. **Analysis:**

- **Recall (63.64%):** It performed better than Z-Score in detecting attacks but suffered from lower precision (10.73%).
- **Application:** This model is best used as a visual aid for operators to see smoothed trends rather than a primary automated alert system

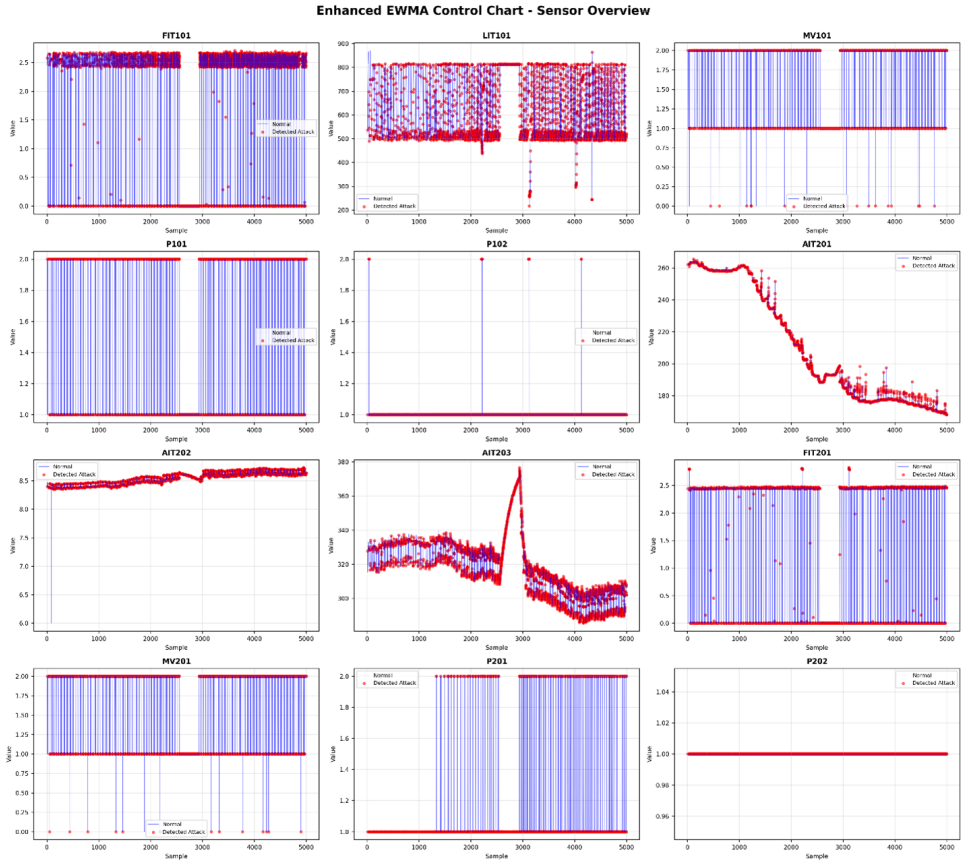

Figure 6.10: Enhanced EWMA Control Charts - Sensor Overview

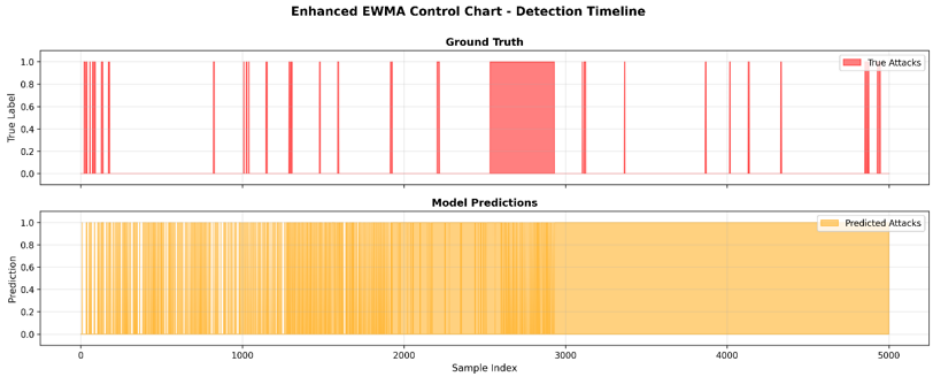

Figure 6.11: Enhanced EWMA Control Charts - Detection Timeline

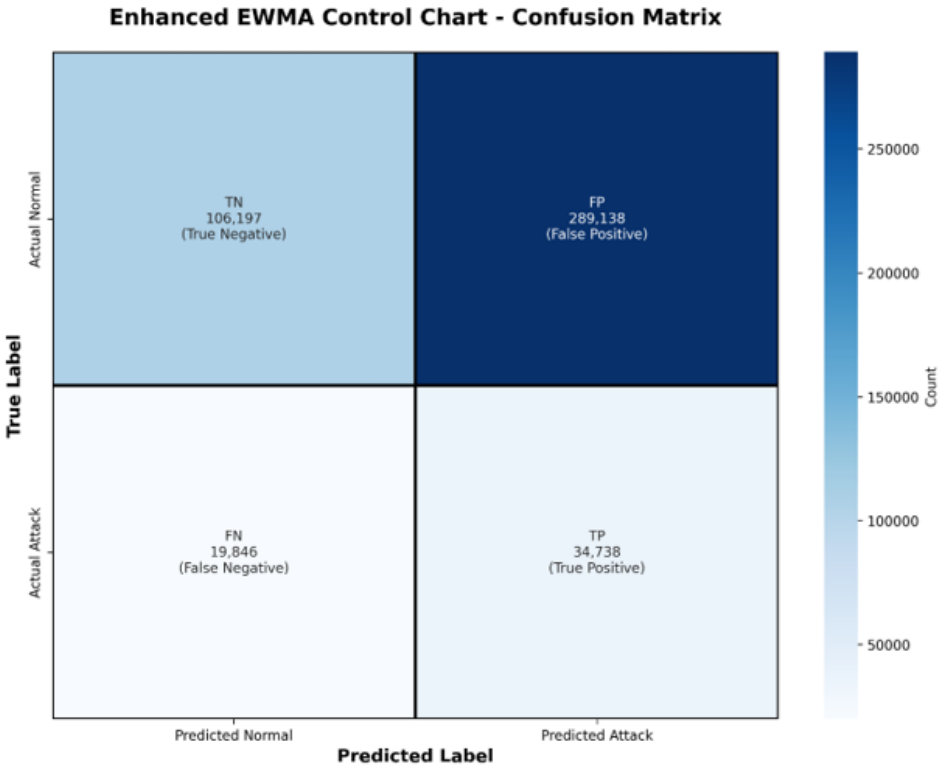

Figure 6.12: Enhanced EWMA Control Charts - Confusion Matrix

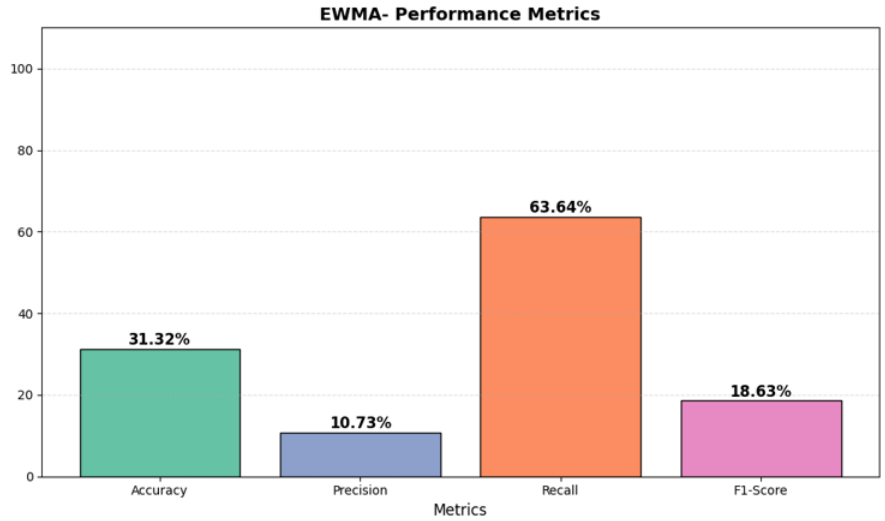

Figure 6.13: Enhanced EWMA Control Charts - Performance Metrics

6.6.4 CUSUM Results

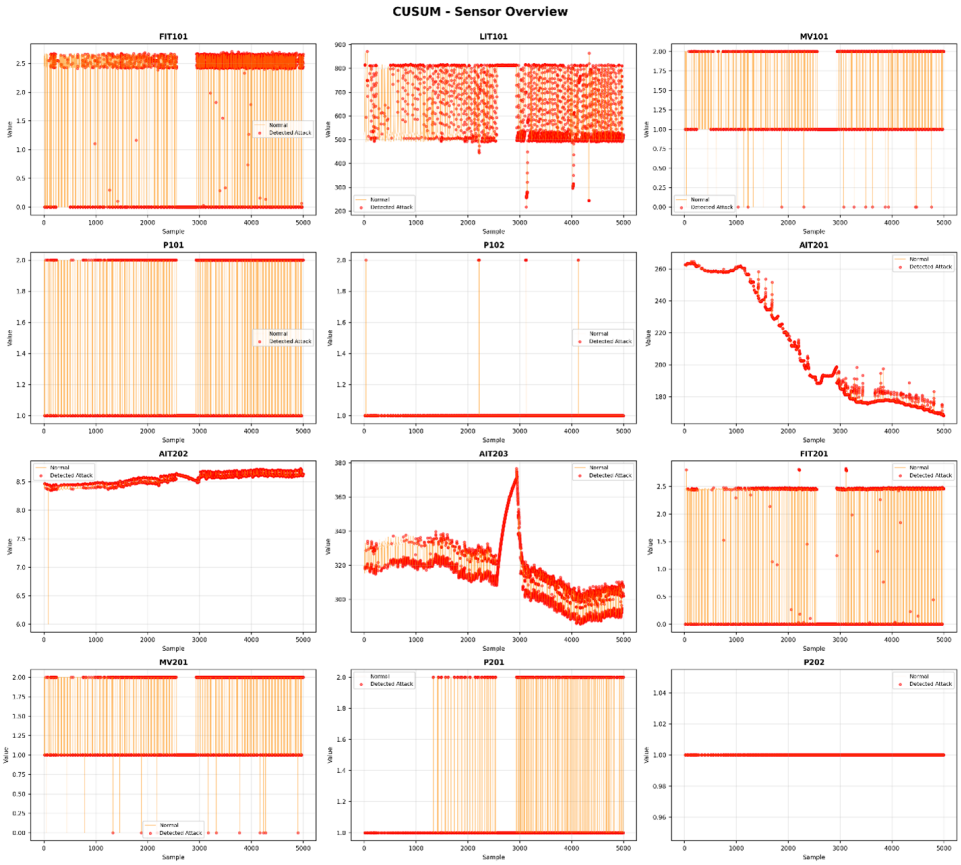

Figure 6.14: CUSUM - Sensor Overview

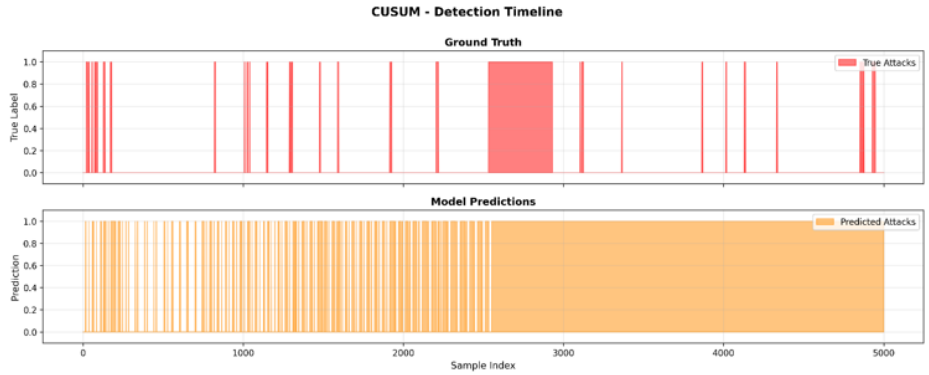

Figure 6.15: CUSUM - Detection Timeline

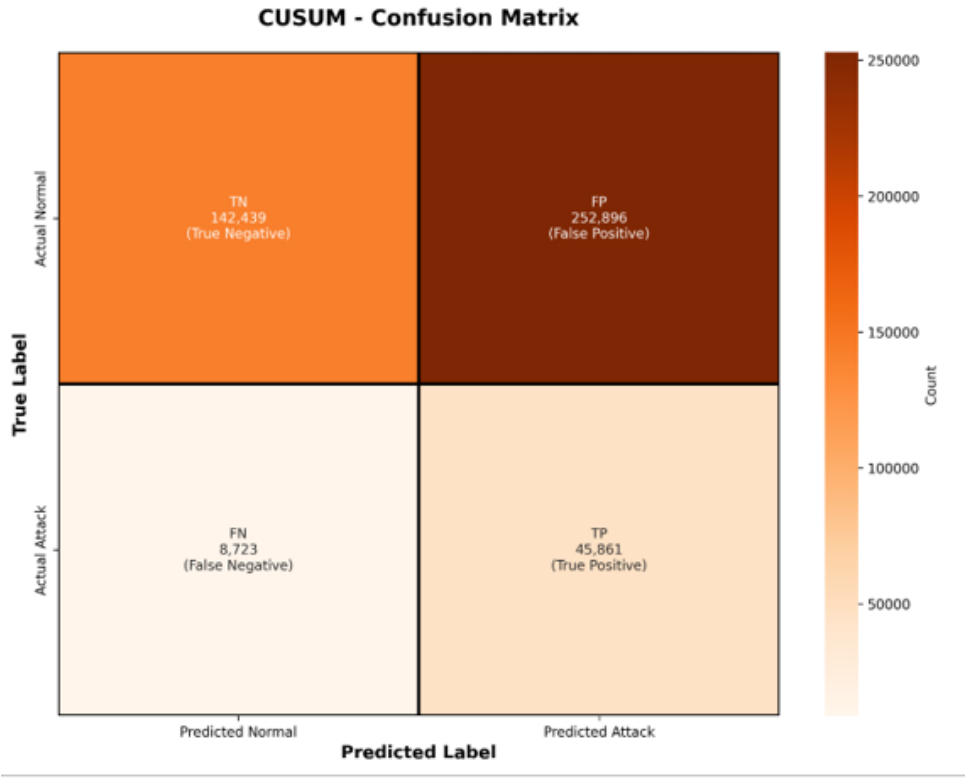

Figure 6.16: CUSUM - Confusion Matrix

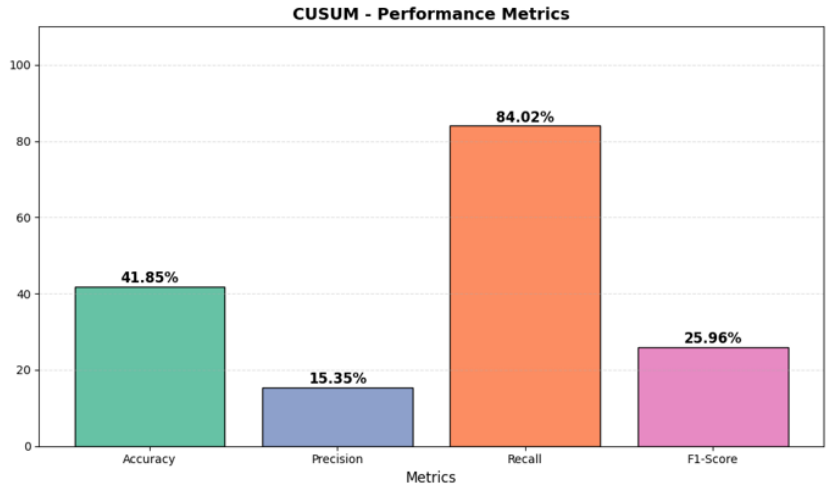

Figure 6.17: CUSUM - Performance Metrics

6.6.5 Improved Granger HoltWinters ESD Results

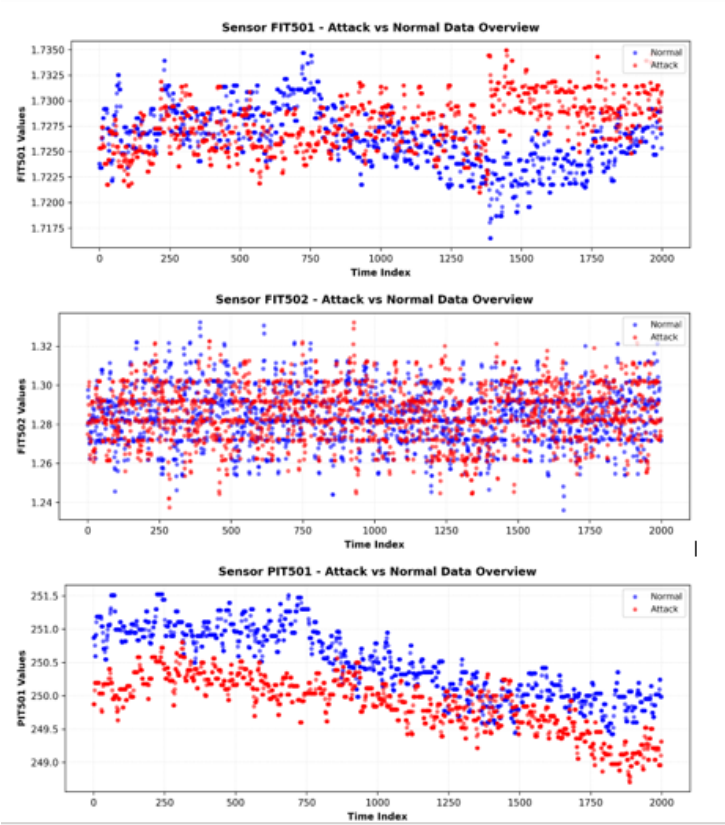

Figure 6.18: Improved Granger HoltWinters ESD - Sensor Overview

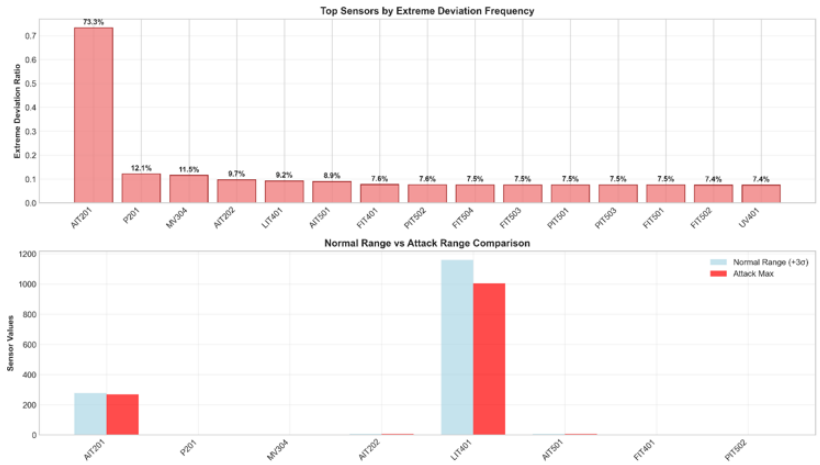

Figure 6.19: Improved Granger HoltWinters ESD - Detection Timeline

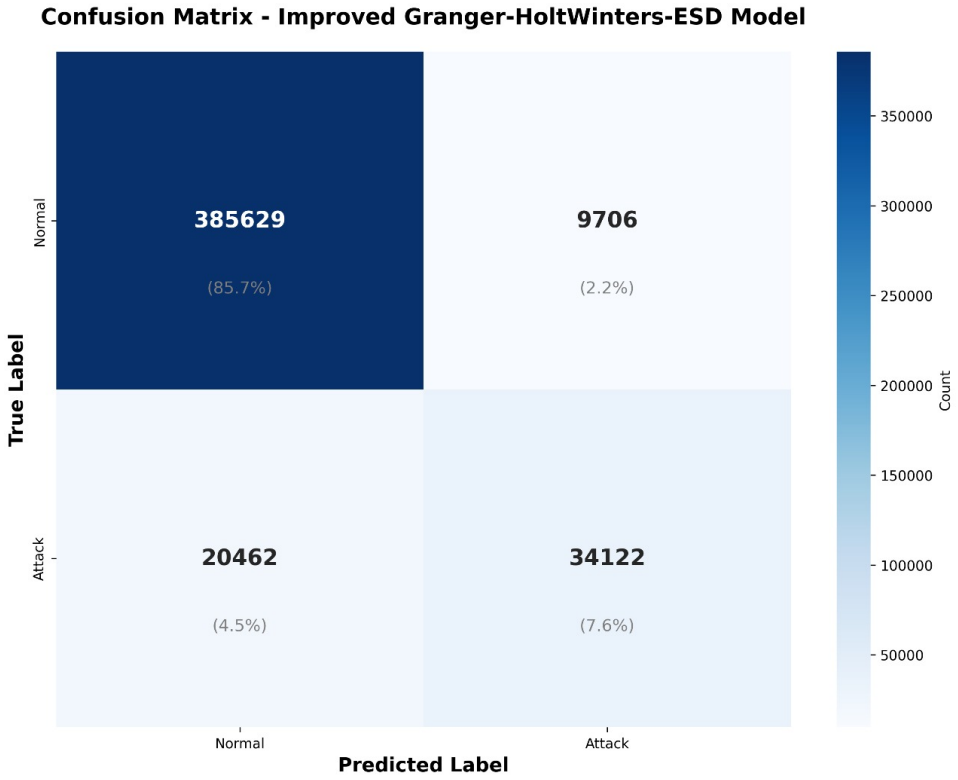

Figure 6.20: Improved Granger HoltWinters ESD - Confusion Matrix

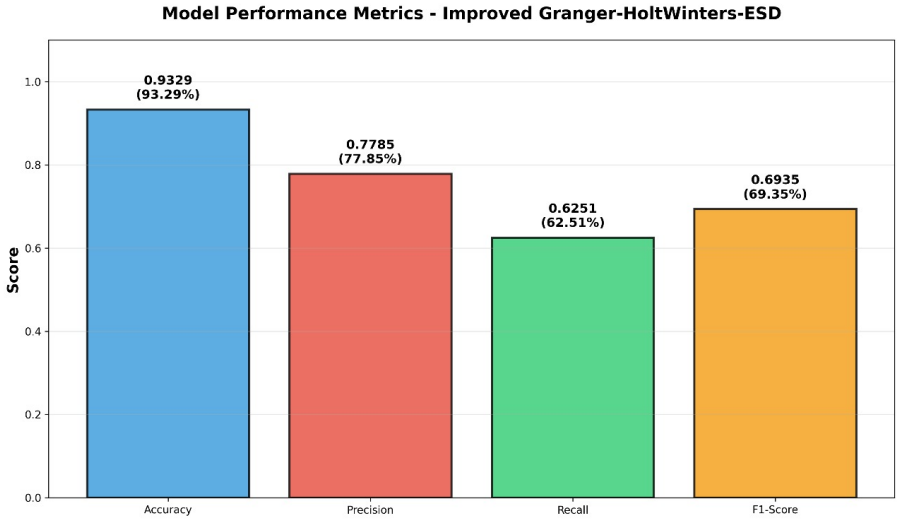

Figure 6.21: Improved Granger HoltWinters ESD - Performance Metrics

6.6.6 Results Summary of Univariate Models

The following table summarizes the final performance of all trained models.

Table 6.5: Results Summary of Univariate Models

Models	**Accuracy**	**Precision**	**Recall**	**F1-Score**
Z-Score Dynamic	95.31%	99.46%	61.68%	76.14%
Rolling Quantile	95.12%	97.51%	61.31%	75.29%
EWMA	31.32%	10.73%	63.64%	18.36%
CUSUM	41.85%	15.35%	84.02%	25.96%
HoltWinters	93.29%	77.85%	62.51%	69.35%

6.6.7 Selection of Candidates for Quantitative Evaluation

The process of selecting the candidates to undergo quantitative evaluation is described in 6.6.7. Although the system design (Chapter 3) initially used time-series forecasting models such as ARIMA and Holt-Winters, preliminary analysis indicated that the forecasting models tend to overfit when used on the highly dynamic sensor data of the SWaT data. Although they were able to effectively capture linear trends, they had a high probability of not differentiating benign operational noise and actual attacks and hence not achieving high reliability.

Thus, the intensive quantitative assessment in this part is concerned with the four Statistical Process Control (SPC) algorithms (Z-Score, Moving Average, EWMA, and Rolling Quantile). These were the models of greatest priority in final consideration since they are more interpretable and statistically robust. This is of paramount importance in an industrial setting where an anomaly detection system must not only indicate an error, but it must also do so with logic that is easy to understand (e.g., violations of standard deviation) and not some complicated black box predictive error.

Chapter 7

Conclusion

The following section concludes our work in a brief manner giving an overall summary of our work and the final product that we attained.

7.1 Conclusion

The achievements of the Anomaly Detection System are highlighted below:

7.1.1 Statistical Anomaly Detection

The main strength of the system in terms of analysis is its range of univariate and multivariate statistical models. As opposed to the legacy systems, which relied on inflexible hard coded boundaries, the solution is based on dynamic profiling algorithms like Z Score model. In this fashion, the system is able to automatically compute the standard deviations and adapt its baseline to normal behavior in real-time rendering it competent to detect covert cyber attacks and minor equipment drifts.

7.1.2 Modular and Scalable Infrastructure

The high level of efficiency and scalability of the system is achieved by using FastAPI to process the data on the backend and SQLite to store the data. This guarantees that the system will be capable of changing to meet emerging threats without necessarily having to revamp the infrastructure.

7.1.3 Interactive Visualization

In order to simplify the complex data science and the daily operations the system presents a very interactive dashboard created using React. This interface converts raw numerical results into understandable and actionable results with graphing tools.

7.1.4 Powerful False Alarm Filtering

Extreme Studentized Deviation ESD filtering can be listed as one of the most viable contributions made by this project. The security risk in industrial settings where operators are impacted by the constant false positives is a significant inconvenience. This system counters this obstacle by authenticating outliers while it is still in an alert state. This sophisticated filtering has been used to ensure that the system will alert operators only of statistically significant anomalies thus being able to maintain high level of confidence in the automated warnings.

7.1.5 Efficient Data Management

Lastly, the system is characterized with a very high frequency of managing high frequency data due to its optimized database configuration. Through the use of SQLite in Write Ahead Logging

WAL mode the application has been able to overcome performance bottlenecks. The outcome is a smooth and responsive experience that does not decline even when there is a high workload.

7.2 Future Work

Although the Anomaly Detection System has achieved its main objectives and has laid a good platform in the field of ensuring security in industries there are still a few areas where it can be developed even more to improve the performance and operational usefulness [19, 20].

7.2.1 Integration of Deep Learning

The project may be further developed with more sophisticated deep learning design models, like Autoencoders or Long Short Term Memory networks. In contrast to the models currently used in statistics, the deep learning models are very efficient whereas the current statistical models provide an excellent interpretation capability.

7.2.2 Multivariate Analysis of Correlations

The existing system mainly gives an analysis of sensor streams separately, whereas the future development intends to involve multivariate correlation analysis. This would enable the system to cross verify the physical conditions of various parts to provide logical consistency e.g. the flow rate of a given flow line is matched with the open position of a given valve. This is needed to identify advanced logic based attacks in which a sensor may be operating in a normal state, but the collective state of the sensors may be a physical impossibility or a violation of a process.

7.2.3 Live SCADA Integration

To convert the system to a fully integrated monitoring solution as opposed to a retrospective analysis tool that we intend to adopt, we will engage in the use of direct connection with live SCADA streams. The system may directly consume real time data through the programmable logic controllers by supporting industrial communication protocols like Modbus or OPC UA. This would do away with the manual uploading of files and the data would be detected and alerted immediately as it moves out of the factory floor.

7.2.4 Human in the Loop Feedback

One key point of weakness that requires enhancement is the implementation of a feedback mechanism that will involve human experience in machine learning. This might be further updated in the future with a feature that the operator can specifically label an anomaly flagged to be a true positive or a false alarm. This feedback information would then be fed into the system to automatically re adjust sensitivities and enhance accuracy of the detection algorithms as time advanced.

7.2.5 Edge Deployment

Lastly we will strive to deploy the Python models on edge devices (e.g. industrial gateway or Raspberry Pi). By locating the detection logic nearer to the source sensors, network latency is reduced by a huge margin and bandwidth is also saved. This edge computing implementation would have the added advantage of continuing the important security analysis even when the main connection to the cloud or the main server is temporarily lost.

References

[1] A. L. Perales Gómez, L. Fernández Maimó, A. L. Sandoval Orozco, *et al.*, "Madics: A methodology for anomaly detection in industrial control systems," *Symmetry*, vol. 12, no. 10, p. 1583, 2020.

[2] D. Shalyga, P. Filonov, and A. Lavrentyev, "Anomaly detection for water treatment system based on neural network with automatic architecture optimization," 2018.

[3] S. H. Zia, N. Bibi, S. Alhazmi, *et al.*, "Enhanced anomaly detection in iot through transformer-based adversarial perturbations model," *Electronics*, vol. 14, no. 6, 2025.

[4] O. H. Jha, M. Kurundkar, and S. Shinde, "Cyberattack detection on swat plant industrial control systems using machine learning," *Journal of Cybersecurity and Privacy*, 2024.

[5] A. Deng and B. Hooi, "Graph neural network-based anomaly detection in multivariate time series," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, no. 5, pp. 4027–4035, 2021.

[6] K. Hundman, V. Constantinou, C. Laporte, I. Colwell, and T. Soderstrom, "Detecting spacecraft anomalies using lstm-ae and dynamic thresholding," *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 387–395, 2018.

[7] J. Audibert, P. Michiardi, F. Guyard, S. Marti, and M. A. Zuluaga, "Usad: Unsupervised anomaly detection on multivariate time series," *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 3395–3404, 2020.

[8] D. Geiger, D. Liu, S. Al-Dahwi, *et al.*, "Tadgan: Time series anomaly detection using generative adversarial networks," *IEEE International Conference on Big Data*, pp. 33–43, 2020.

[9] J. Goh, S. Adepu, K. N. Junejo, and A. Mathur, "A dataset to support research in the design of secure water treatment systems," 2016.

[10] J. Inoue, Y. Yamagata, Y. Chen, C. M. Poskitt, and J. Sun, "Anomaly detection for a water treatment system using unsupervised machine learning," 2017.

[11] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," *ACM Computing Surveys*, vol. 41, no. 3, pp. 1–58, 2009.

[12] M. Alsadhan and K. Shafi, "Anomaly detection in industrial control systems using causal inference," *Computers & Security*, vol. 95, p. 101846, 2020.

[13] S. H. H. Vallakh, "Time-series-based anomaly detection in industrial control systems using sgan," *Processes*, vol. 13, no. 9, 2025.

[14] G. Pang, C. Shen, L. Cao, and A. Van Den Hengel, "Deep learning for anomaly detection: A review," *ACM Computing Surveys*, vol. 54, no. 2, 2021.

[15] R. Chalapathy and S. Chawla, "Deep learning for anomaly detection: A survey," 2019.

[16] S. Potluri and C. Diedrich, "Deep learning based anomaly detection for industrial control systems," 2024.

[17] A. Rosner, "Seasonal-trend decomposition using loess (stl) and extreme studentized deviate (esd) for anomaly detection," 2014.

[18] M. Braei and S. Wagner, "Anomaly detection in univariate time-series: A survey on the state-of-the-art," 2020.

[19] A. Blázquez-García, A. Conde, U. Mori, and J. A. Lozano, "A review on outlier/anomaly detection in time series data," *ACM Computing Surveys*, vol. 54, no. 3, 2021.

[20] N. S. Ghadim, A. H. Lashkari, and A. Ghorbani, "Anomaly detection dataset for industrial control systems," *IEEE Access*, vol. 11, 2023.