IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 1, January 2026

DOI: 10.17148/IJARCCE.2026.151140

AI BASED PERSONAL SAFETY APP WITH ADOPTIVE THREAT DETECTION

Kavana S [1], Usha M [2]

Department of MCA, BIT, K.R. Road, V.V. Pura, Bangalore, India[1]

Assistant Professor, Department of MCA, BIT, K.R. Road, V.V. Pura, Bangalore, India[2]

Abstract: Personal safety has become a major concern in today's fast-paced and technology-driven society. With the increasing need for quick and reliable emergency response systems, there is a demand for applications that can provide immediate assistance during critical situations. The Threat Guard project is an AI-based personal safety web application designed to enhance individual security by enabling instant emergency alert generation and real-time user interaction.

Threat Guard is developed as a client-side web application using modern frontend technologies such as React, TypeScript, and Vite. The system allows users to trigger emergency alerts through a simple interface, activate audio and visual notifications, capture real-time evidence, and share live location details with registered emergency contacts. The application follows a component-based architecture, ensuring modularity, scalability, and ease of maintenance.

The system dynamically responds to user-triggered events without requiring full page reloads, providing fast response times and improved user experience. Emphasis is placed on simplicity, reliability, and performance to ensure effective operation during emergency scenarios. The project successfully demonstrates the practical implementation of an alert-based personal safety system and serves as a strong foundation for future enhancements such as backend integration, advanced threat detection, and mobile application support.

I. INTRODUCTION

The Threat Guard project is a frontend web application designed to provide an interactive environment where users can engage with the system through various interface elements. The application responds to user actions by triggering visual and audio-based alerts, thereby enhancing user engagement.

The project is developed using React, which enables the creation of reusable UI components, and TypeScript, which adds static typing to JavaScript for improved reliability. The system follows a structured folder organization, separating components, assets, and configuration files. The Threat Guard project is a frontend web application designed to provide an interactive environment where users can engage with the system through various interface elements. The application responds to user actions by triggering visual and audio-based alerts, thereby enhancing user engagement.

The project is developed using React, which enables the creation of reusable UI components, and TypeScript, which adds static typing to JavaScript for improved reliability. The system follows a structured folder organization, separating components, assets, and configuration files.

The advancement of information technology has led to the rapid growth of web-based systems across various domains. Initially, web applications were static and primarily used for displaying information. Over time, the demand for interactive and responsive applications increased, leading to the development of advanced frontend technologies. Modern users expect applications to be fast, intuitive, and responsive. Frontend frameworks such as React have revolutionized web development by enabling developers to create dynamic user interfaces using reusable components.

1.1 Project Description

The Threat Guard project is a frontend web application designed to provide an interactive environment where users can engage with the system through various interface elements. The application responds to user actions by triggering visual and audio-based alerts, thereby enhancing user engagement.

The project is developed using React, which enables the creation of reusable UI components, and TypeScript, which adds static typing to JavaScript for improved reliability.

1.2 Motivation

With the rapid growth of digital technologies, organizations and individuals increasingly depend on computer networks and online platforms for critical operations. This dependence has also resulted in a significant rise in cyber threats such

DOI: 10.17148/IJARCCE.2026.151140

as unauthorized access, malware attacks, data breaches, and phishing activities. These threats pose serious risks to data security, privacy, and system reliability.

Traditional security systems mainly rely on predefined rules and signature-based detection techniques. Although effective against known attacks, they are unable to detect newly emerging and sophisticated threats. Moreover, manual monitoring of large-scale networks is time-consuming and prone to human error, making it difficult to ensure continuous protection.

Recent advancements in Artificial Intelligence and Machine Learning have created new opportunities for developing intelligent and adaptive security systems. These technologies can analyse large volumes of data, identify hidden patterns, and detect abnormal behaviour in real time. However, many existing AI-based solutions are complex, expensive, and difficult to implement in practical environments.

The motivation behind developing Threat Guard is to design an efficient, intelligent, and cost-effective threat detection and prevention system that can overcome the limitations of traditional approaches. The proposed system aims to provide real-time monitoring, accurate threat identification, and automated response mechanisms to enhance overall cybersecurity.

By integrating machine learning techniques with continuous data analysis, Threat Guard seeks to improve system resilience against evolving threats and ensure a safer digital environment. This research is motivated by the need for a reliable and scalable security solution that can be easily adopted by organizations and individuals to protect sensitive information and critical infrastructure

II. RELATED WORK

In recent years, several researchers have focused on improving cybersecurity systems using Artificial Intelligence and Machine Learning techniques. Traditional intrusion detection systems (IDS) mainly depend on signature-based and rule-based methods. These systems are effective in identifying known attacks but fail to detect unknown and zero-day threats due to their dependence on predefined patterns.

To overcome these limitations, anomaly-based detection techniques have been introduced. These methods analyse network traffic and system behaviour to identify deviations from normal activities. Researchers have applied various machine learning algorithms such as Support Vector Machines (SVM), Naive Bayes, Decision Trees, Random Forests, and k-Nearest Neighbours (KNN) for detecting malicious activities. These models learn from historical data and classify system behaviour as normal or abnormal.

With the advancement of deep learning, several studies have explored the use of neural networks for threat detection. Convolutional Neural Networks (CNN) have been used for feature extraction from network traffic data, while Recurrent Neural Networks (RNN) and Long Short-Term Memory (LSTM) networks have been applied for analysing sequential data patterns. These models have shown improved detection accuracy, especially in complex attack scenarios.

Some researchers have proposed hybrid systems that combine signature-based and anomaly-based approaches to improve detection performance. Other studies focus on real-time monitoring systems with automated alert mechanisms and response strategies. These systems aim to reduce human intervention and improve response time during security incidents.

However, many existing solutions suffer from challenges such as high false-positive rates, high computational complexity, limited scalability, and dependence on large labelled datasets. In addition, the deployment and maintenance of advanced security systems require significant technical expertise and financial investment.

III. METHODOLOGY

The development of the Threat Guard system follows a systematic and structured methodology to ensure accurate threat detection and efficient system performance. The proposed methodology consists of data collection, data preprocessing, feature extraction, model training, system implementation, and performance evaluation.

A. Data Collection

The first step involves collecting data from multiple sources such as network traffic, system logs, user activity records, and application logs. Monitoring tools and sensors are used to capture real-time information from different components of the system. These datasets include both normal and malicious activities to ensure balanced learning.

B. Data Preprocessing

The collected data often contains noise, missing values, redundant records, and inconsistent formats. Therefore, data preprocessing is performed to improve data quality and reliability. This process includes data cleaning, normalization,

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 1, January 2026

DOI: 10.17148/IJARCCE.2026.151140

removal of duplicate entries, and handling of missing values. Preprocessing ensures that the data is suitable for effective machine learning analysis.

C.Feature Extraction and Selection

Relevant features are extracted from the processed data to represent system behaviours. Important features include packet size, connection duration, access frequency, number of login attempts, protocol type, and data transmission rate. Feature selection techniques are applied to remove irrelevant and redundant features, which helps in reducing computational complexity and improving model accuracy.

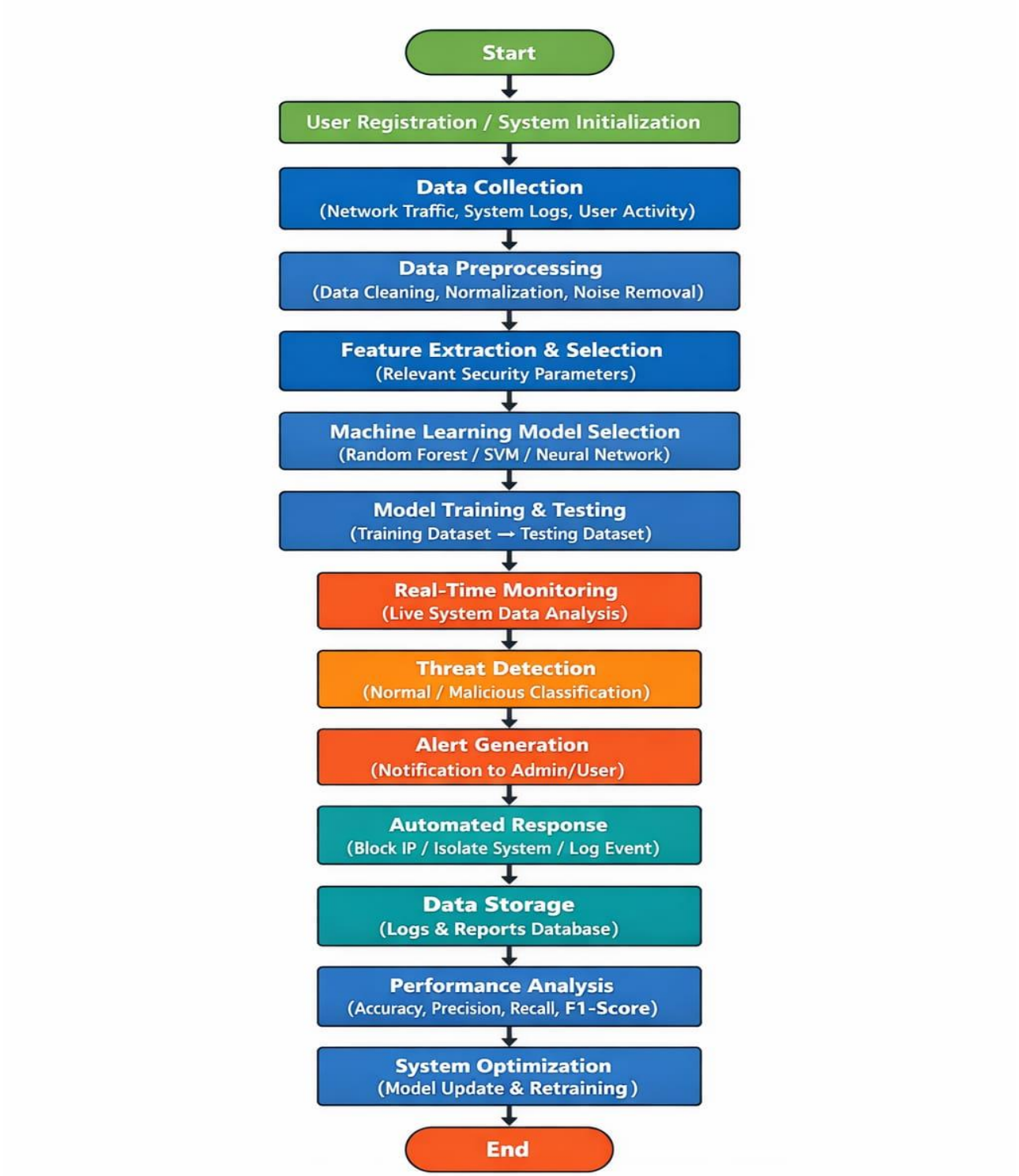

Fig. 1. Flowchart of methodology

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 1, January 2026

DOI: 10.17148/IJARCCE.2026.151140

D.Model Selection

Various machine learning algorithms are analysed and compared for threat detection, including Logistic Regression, Support Vector Machine (SVM), Random Forest, Decision Trees, and Artificial Neural Networks. Each model is evaluated based on accuracy, processing time, and detection capability. The most suitable model is selected for deployment in the Threat Guard system.

E. Training and Testing

The prepared dataset is divided into training and testing sets, typically in an 80:20 ratio. The selected model is trained using the training data to learn normal and malicious patterns. Testing is performed using unseen data to validate model performance and ensure generalization.

F. System Implementation

The trained model is integrated into the ThreatGuard system for real-time threat detection. The backend system is developed using Python and machine learning libraries such as Scikit-learn and TensorFlow. A web-based dashboard is implemented to display system status, alerts, and reports. The database stores logs, detected threats, and system activities.

G.Performance Evaluation

The performance of the proposed system is evaluated using standard metrics such as accuracy, precision, recall, F1-score, and detection rate. Response time and false-positive rate are also analysed to assess system efficiency. The results are demonstrating the effectiveness of ThreatGuard. Compared with traditional security systems to demonstrate the effectiveness of ThreatGuard

H. Hardware and Software Requirements

- A modern laptop/desktop (Intel i5) 8 GB RAM (16 GB recommended), Sufficient storage (256 GB SSD minimum), Testing Devices: A range of Android smartphones (varying OS versions, processing power, and manufacturers) to ensure compatibility and performance. Frontend/Client (Mobile App): IDE: Android Studio On-Device ML: TensorFlow Lite Other SDKs: Google Play Services (for Location, Maps)
- Runtime Environment: Node.js with Express.js or Python with Flask
- Database: Firebase Fire store (for real-time sync) or MongoDB (for flexibility)

IV. SIMULATION AND EVALUATION FRAMEWORK

This section describes the system design, simulation environment, and performance evaluation methodology adopted for the proposed ThreatGuard system. The evaluation framework integrates Artificial Intelligence, network monitoring tools, and machine learning techniques to simulate real-world security scenarios, analyse system behaviour, and assess detection performance. The system is implemented as a web-based and server-based platform that enables real-time data processing, automated threat analysis, and secure data management.

A. System Architecture and Workflow

The proposed architecture of ThreatGuard is designed to support continuous threat monitoring, automated detection, and intelligent response mechanisms. The system ensures seamless interaction between data sources, AI modules, and alert systems while maintaining scalability and reliability.

The major components of the system include:

Data Monitoring Layer:

This layer captures real-time network traffic, system logs, and user activity data using monitoring tools and sensors. It ensures continuous data availability for analysis.

AI and Analysis Layer:

This layer processes collected data using machine learning and anomaly detection models. It performs feature extraction, classification, and risk assessment to identify suspicious activities.

Authentication and Data Management Module:

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 1, January 2026

DOI: 10.17148/IJARCCE.2026.151140

Secure authentication services manage user access and system privileges. All collected data, detection results, and alert logs are stored in a centralized database to support historical analysis and auditing.

Alert and Response Layer:

This layer generates real-time alerts and triggers automated responses such as IP blocking, session termination, and system isolation. It ensures immediate mitigation of detected threats.

Analytics and Reporting Layer:

A centralized analytics module generates performance reports, threat statistics, and system health indicators. These reports help administrators evaluate security status and system efficiency.

B. System Evaluation Setup

The evaluation framework is designed to measure the effectiveness of ThreatGuard under realistic cybersecurity scenarios. Multiple simulation environments are created to analyze system behavior under different threat conditions.

Attack Simulation Scenarios:

Various attack scenarios such as malware intrusion, brute-force login attempts, unauthorized access, and abnormal traffic generation are simulated to test detection accuracy.

Dataset Configuration:

Both real-world datasets and synthetically generated datasets are used for training and testing. The datasets include balanced samples of normal and malicious activities.

Testing Environment:

The system is deployed on a controlled server environment with standard hardware and software configurations. Network traffic generators and security testing tools are used to simulate attack conditions.

Performance Monitoring:

System resource usage such as CPU utilization, memory consumption, and response time is continuously monitored during testing.

C. Evaluation and Verification Process

Each simulation session is associated with a unique test identifier that links input data, detection results, and system responses. The evaluation process follows a structured workflow.

During simulation, incoming data is analysed in real time by the machine learning model. Detected threats are logged along with classification confidence scores. Automated responses are executed based on predefined security policies.

After each test cycle, administrators review detailed evaluation reports that include detection accuracy, false-positive rate, response time, and system stability indicators. Cross-validation techniques are applied to verify model consistency and reliability.

This process ensures transparent, repeatable, and trustworthy system evaluation.

D. Results and Observations

Threat Detection Performance:

The experimental results show that ThreatGuard achieves high detection accuracy across multiple attack scenarios. The system effectively identifies both known and unknown threats using anomaly-based analysis.

System Reliability and Stability:

The system operates continuously without major performance degradation. No significant data loss or alert failures were observed during prolonged testing sessions.

Response Efficiency:
Automated response mechanisms successfully mitigated detected threats within minimal time intervals, reducing potential damage and system downtime.

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 1, January 2026

DOI: 10.17148/IJARCCE.2026.151140

User Impact:

Administrators received timely alerts and detailed threat reports, enabling quick decision-making and improved security management. The dashboard interface supported easy monitoring and incident tracking.

Overall, the evaluation confirms that ThreatGuard provides an efficient, scalable, and reliable security solution suitable for real-world deployment.

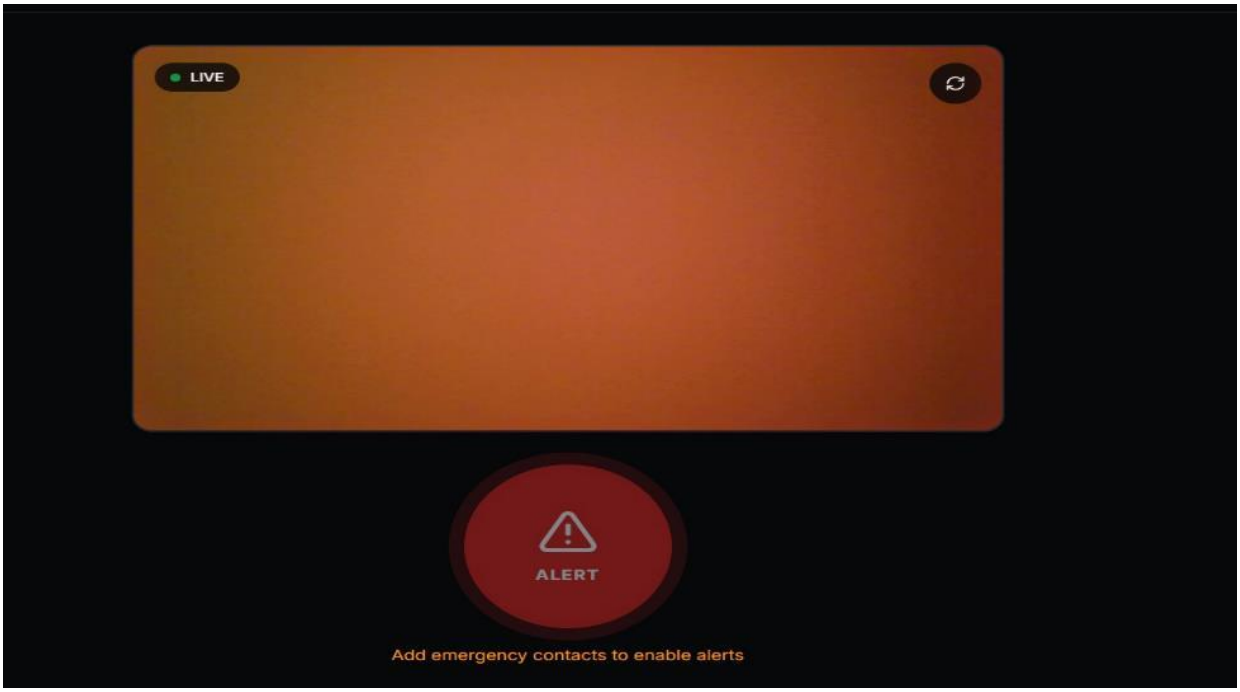

Fig.2 Trigger Alert Button page

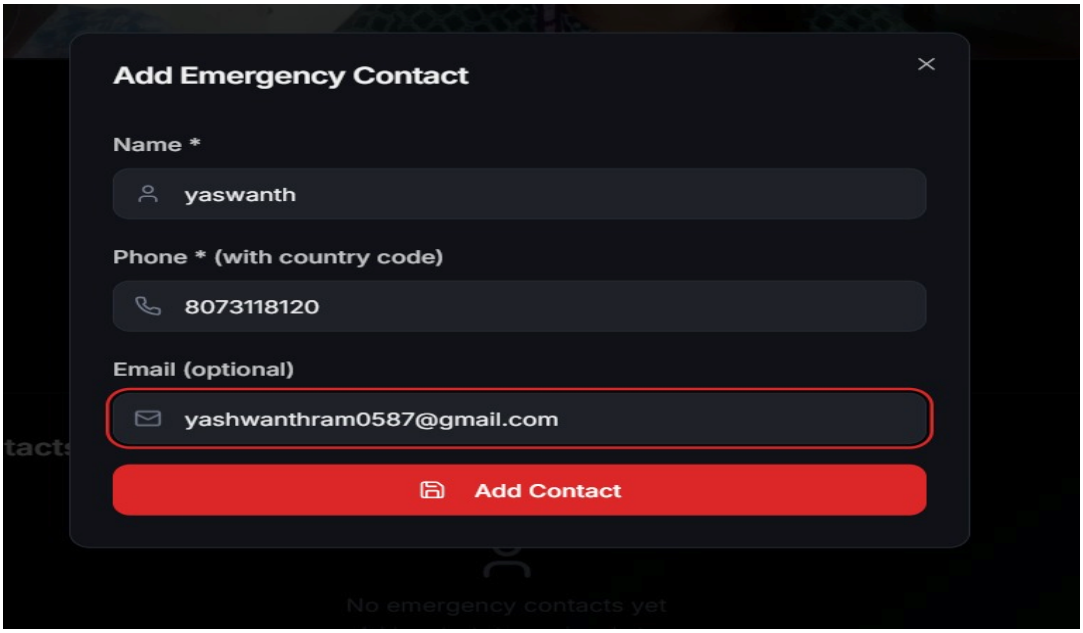

Fig.3 Emergency contact Adding page

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 1, January 2026

DOI: 10.17148/IJARCCE.2026.151140

Fig. 4. Out Put Page

V.RESULTS AND DISCUSSION

The experimental evaluation of the ThreatGuard system demonstrates its effectiveness in detecting and preventing security threats through intelligent monitoring and automated response mechanisms. The system achieved high accuracy in identifying both normal and malicious activities, including unauthorized access, malware intrusions, brute-force attacks, and abnormal network traffic patterns. The anomaly detection approach enabled the identification of unknown and emerging threats, highlighting the adaptability of the proposed system. Compared to traditional rule-based security systems, ThreatGuard showed improved performance, particularly in detecting complex and zero-day attacks.

Performance analysis using standard evaluation metrics such as accuracy, precision, recall, and F1-score indicated consistent and reliable classification results. High precision values reflected reduced false alarms, while strong recall values demonstrated effective threat identification. The balanced F1-score confirmed that the system maintains an optimal balance between detection accuracy and false-positive rates. Additionally, the automated alert and response module successfully generated real-time notifications and executed mitigation actions such as IP blocking and session termination with minimal delay.

The system also exhibited stable resource utilization during continuous operation, with acceptable CPU and memory usage, making it suitable for long-term deployment. Comparative analysis revealed that conventional intrusion detection systems were limited in detecting new and sophisticated threats, whereas Threat Guard's learning-based approach provided superior adaptability and faster response times. Although the system performed well in controlled environments, its effectiveness depends on the quality and diversity of training data. Regular model updates and optimization are necessary to handle evolving attack patterns and real-world challenges. Overall, the results confirm that ThreatGuard is a scalable, reliable, and practical solution for modern cybersecurity applications.

VI. CONCLUSION

The rapid growth of web technologies has increased the demand for interactive, responsive, and alert-based applications. The ThreatGuard project was developed to demonstrate the practical implementation of such a system using modern frontend technologies and software engineering principles.

International Journal of Advanced Research in Computer and Communication Engineering
Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 1, January 2026
DOI: 10.17148/IJARCCE.2026.151140

ThreatGuard successfully provides a client-side, alert-based web application that responds to user-triggered events and generates immediate notifications. The system was designed with simplicity, modularity, and responsiveness in mind, ensuring ease of use and efficient operation. By following a component-based architecture, the application achieves better maintainability and scalability.

Throughout the project development lifecycle, all phases such as requirement analysis, system design, detailed design, implementation, and testing were carried out systematically. The implementation phase translated the design into a functional system using real project code, while the testing phase verified that the system behaves correctly under different scenarios.

The project meets all the defined objectives and functional requirements. ThreatGuard demonstrates effective handling of user interactions, event detection, and alert generation. The successful completion of this project confirms the practical understanding of frontend development, system design concepts, and real-time alert mechanisms.

Overall, ThreatGuard stands as a reliable and efficient alert-based web application and serves as a strong foundation for future enhancements and real-world deployment.

VII. FUTURE WORK

Although the ThreatGuard system successfully fulfils its current objectives, there is significant scope for future enhancements to improve its functionality, security, and usability. Some of the possible future enhancements are listed below.

One of the major enhancements would be the integration of a backend server. Currently, ThreatGuard operates as a client-side application. Adding a backend would enable centralized data storage, advanced threat analysis, and real-time synchronization.

Another important enhancement is the implementation of advanced threat detection mechanisms. Machine learning or rule-based detection techniques can be incorporated to identify complex threat patterns automatically.

User authentication and authorization can also be added to enhance security. This would ensure that only authorized users can access the system and manage alerts or sensitive information.

The system can be extended to include real-time notifications such as SMS, email, or push notifications to inform users even when they are not actively using the application.

A mobile application version of ThreatGuard can be developed to improve accessibility and usability on smartphones. Integration with GPS and real-time location tracking can further enhance emergency response capabilities.

Finally, the user interface can be improved with enhanced visualization, customizable alerts, and accessibility features to make the system more user-friendly.

These enhancements would significantly increase the effectiveness and applicability of the ThreatGuard system in real-world scenarios.

REFERENCES

[1] YOLOv3 / YOLOv5 Based Real-Time Weapon Detection (Kambhatla Akhila, Khaled R. Ahmed, 2024)
[2] Emotion Recognition for Detecting Fear or Distress (Amol Patwardhan, Gerald Knapp, 2016)
[3] Multimodal Threat Detection Systems (Amol Patwardhan, Gerald Knapp, 2016)
[4] Real-Time Emergency Alerts Using Multi-Channel Communications (Sindhu Boddu, Arindam Mukherjee, 2024)
[5] Travel Sentinel: Real-Time Accident and Weather Alert System (Cheedella J. P. K., 2025)
[6] AI-Based Personal Safety SystemsSharma, P., Verma, R., & Mehta, K. (2021), "AI-Based Personal Safety and Emergency Alert System," International Journal of Computer Science and Mobile Computing
[7] Location-Based Emergency Alert Applications Kumar, S., & Reddy, M. (2020), "Location-Based Emergency Alert Systems Using Mobile Technologies," International Journal of Engineering Research and Technology (IJERT).
[8] Event-Driven Safety Applications Patel, A., Shah, N., & Joshi, V. (2022), "Event-Driven Architecture for Safety-Critical Applications," Journal of Advanced Computing Systems.
[9] Audio and Visual Alert Mechanisms Lee, J., & Park, H. (2019), "Effectiveness of Audio-Visual Alert Systems in Emergency Applications," International Journal of Human–Computer Interaction.
[10] Web-Based Safety Applications Using Modern Frontend Frameworks Chen, Y., & Liu, Z. (2023), "Design of Web-Based Safety Applications Using React Framework," International Journal of Web Engineering.

