iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 14 **Issue:** V **Month of publication:** May 2026

DOI: https://doi.org/10.22214/ijraset.2026.81866

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 14 Issue V May 2026- Available at www.ijraset.com

Spam Email Filter with NLP: A Machine Learning Approach

G. Prudhvi[1], Dr. U. Sathish[2], B. Giridhara Reddy[3], G. Vignesh[4], D. Vijay[5]
Department of Cyber Security, Acharya Nagarjuna University, Guntur - 522508, India

Abstract: ***With the rapid expansion of digital communication, spam emails containing phishing links, fraudulent offers, or malware have become a major security concern for individuals and organizations. The proposed project, Spam Email Filter using NLP, utilizes Natural Language Processing (NLP) and Machine Learning (ML) to automatically identify and filter such malicious emails. The system preprocesses email text through several NLP stages, including tokenization, stopword removal, normalization, and feature extraction using TF-IDF (Term Frequency-Inverse Document Frequency) vectorization. These features are then used to train a Naive Bayes classifier, which distinguishes between spam and legitimate (ham) emails based on learned probability distributions. Performance is evaluated using metrics such as accuracy, precision, recall, and F1-score, demonstrating high accuracy and minimal false positives. The model is computationally efficient, adaptable for real-time spam detection, and enhances email security by filtering harmful content. Future improvements may include multilingual support and adaptive learning to handle evolving spam patterns.***

Keywords: Spam Detection, Natural Language Processing, Machine Learning, TF-IDF, Naive Bayes, Email Classification, Text Mining, Cybersecurity

I. INTRODUCTION

Electronic mail remains one of the most widely used communication mechanisms in both personal and professional contexts. Despite its ubiquity, email systems are increasingly targeted by malicious actors seeking to distribute spam, phishing links, and malware. According to recent industry reports, approximately 347 billion emails are sent daily worldwide, of which nearly 48% are classified as spam or contain malicious content[1]. This overwhelming volume of unwanted and potentially dangerous email content poses serious risks to data privacy, financial security, and organizational integrity.

Traditional email filtering solutions rely primarily on rulebased approaches, blacklists, and pattern matching techniques. While these methods provided adequate protection in earlier years, they have proven increasingly ineffective against sophisticated modern attacks that employ obfuscation techniques, social engineering, and polymorphic content[2]. Machine learning-based approaches offer a promising alternative by enabling systems to learn from data and adapt to emerging threat patterns without requiring manual rule updates.

This paper introduces SpamGuard AI, an intelligent email security platform that addresses the limitations of traditional spam filters through a comprehensive machine learning pipeline. The system combines Natural Language Processing (NLP) techniques with ensemble learning to classify emails in real-time with high accuracy. Unlike existing solutions that focus solely on static analysis, SpamGuard AI integrates directly with users' Gmail accounts via the official Gmail API, enabling proactive scanning of incoming messages as they arrive.

The proposed system architecture features a modern technology stack designed for performance and scalability. The frontend is built using Google's Flutter framework, enabling deployment across web and mobile platforms from a single codebase. The backend leverages FastAPI, a high-performance asynchronous Python framework, to serve machine learning predictions with minimal latency. Authentication is handled securely through Google OAuth 2.0, eliminating the need for password management while enabling authorized access to Gmail data.

The key contributions of this research include: (1) development of an end-to-end real-time email threat detection pipeline integrating TF-IDF feature extraction with Random Forest classification; (2) design and implementation of a crossplatform security dashboard with color-coded threat

visualization; (3) seamless Gmail API integration enabling live

email scanning without compromising user privacy; and (4) comprehensive experimental evaluation demonstrating the system's effectiveness in practical deployment scenarios.

II. PROBLEM STATEMENT

Existing email security solutions suffer from several fundamental limitations that reduce their effectiveness in contemporary threat landscapes. These limitations can be categorized into technical, architectural, and usability challenges that collectively leave users vulnerable to evolving attack methodologies.

From a technical perspective, traditional spam filters rely heavily on keyword matching and regular expression patterns that can be easily circumvented by attackers. Modern phishing emails often substitute characters, use image-based content, or employ contextually deceptive language that bypasses static rule engines[3]. Furthermore, many existing systems lack the ability to analyze the semantic context of email content, leading to high rates of both false positives (legitimate emails marked as spam) and false negatives (malicious emails that evade detection).

Architecturally, many available email security tools operate as isolated desktop applications or browser extensions that lack integration with popular email providers. This disconnected approach prevents real-time scanning of incoming messages and creates friction in the user experience. Additionally, systems that do offer cloud-based scanning often raise privacy concerns, as they may store or process sensitive email content on third-party servers without adequate transparency or user control.

The usability challenges are equally significant. Enterprisegrade email security solutions typically require complex configuration and ongoing maintenance by IT specialists, making them impractical for individual users and small organizations. Consumer-focused alternatives often sacrifice security features for convenience, offering minimal customization and limited visibility into detection decisions. This creates a gap in the market for a solution that balances robust security capabilities with intuitive usability and accessible deployment.

The specific problems addressed by this research are:

1) Inability of rule-based filters to adapt to novel spam and phishing techniques
2) Lack of real-time integration with widely-used email platforms
3) Insufficient contextual analysis of email content beyond keyword matching
4) High false positive rates that cause legitimate emails to be incorrectly filtered
5) Poor user interfaces that provide limited threat visibility and control
6) Privacy concerns associated with third-party email processing services

III. OBJECTIVES

The primary objective of this research is to design and implement an intelligent, real-time email threat detection system that overcomes the limitations identified in existing solutions.

The specific objectives are:

1) To develop a machine learning-based email classification engine using Random Forest and TF-IDF that achieves accuracy exceeding 95% on benchmark datasets.
2) To implement a cross-platform frontend dashboard using Flutter that provides intuitive threat visualization with color-coded severity indicators and real-time analytics.
3) To design a high-performance RESTful API using FastAPI that serves classification requests with average response times below 200 milliseconds.
4) To integrate Google OAuth 2.0 authentication enabling secure, passwordless login and authorized Gmail API access for live email scanning.
5) To ensure all email processing occurs with appropriate privacy safeguards, including minimal data retention and transparent handling practices.
6) To deploy the system using production-grade hosting infrastructure with global content delivery and HTTPS security by default.
7) To conduct comprehensive performance evaluation measuring accuracy, precision, recall, F1-score, and response latency under realistic workload conditions.

IV. LITERATURE REVIEW

The application of machine learning techniques to email spam detection has been extensively studied in the literature. Early approaches focused primarily on Naive Bayes classifiers, which provided a probabilistic framework for distinguishing spam from legitimate messages based on word frequencies. While computationally efficient, Naive Bayes models assume feature independence, a condition that rarely holds in natural language data and limits their classification accuracy in complex scenarios[4].

Support Vector Machines (SVM) emerged as a popular alternative, offering superior performance in high-dimensional feature spaces characteristic of text classification tasks. Research by Umam et al. demonstrated that SVM achieved 97.52% accuracy in phishing email classification using TF-IDF features with a 70:30 train-test split[5]. However, SVM models suffer from scalability limitations when processing large datasets and require careful hyperparameter tuning to achieve optimal performance.

More recently, deep learning approaches have been applied to phishing detection with promising results. Altwaijry et al. conducted a comparative study of deep learning models for phishing email detection, evaluating LSTM, CNN, and transformer-based architectures[6]. While these models demonstrated strong performance in capturing sequential and contextual patterns, they typically require substantial computational resources and large training datasets, which may limit their deployment in resource-constrained environments.

The Random Forest algorithm has gained significant attention for spam detection due to its ability to handle high-dimensional data, resistance to overfitting, and inherent feature importance analysis capabilities. Dwianti et al. developed a phishing email detection system combining TF-IDF with Random Forest, achieving 98.01% accuracy on a balanced dataset of 82,486 emails[7]. Their feature importance analysis revealed that terms such as "click," "money," "attached," and "online" were highly predictive of phishing content. Similarly, Adebanjo et al. reported 99% accuracy using Random Forest with TF-IDF for Nigerian email spam detection, highlighting the algorithm's effectiveness across different linguistic contexts[8].

The TF-IDF feature extraction technique remains one of the most widely adopted approaches for converting textual email content into numerical representations suitable for machine learning. By weighting terms based on their frequency within individual documents relative to their prevalence across the entire corpus, TF-IDF effectively highlights discriminative words while suppressing common terms that provide little classification value[9]. Modern implementations using scikitlearn's TfidfVectorizer provide efficient, production-ready feature extraction that integrates seamlessly with subsequent classification stages.

Several studies have explored the integration of email security systems with external APIs and services. Kalamkar et al. developed a real-time phishing URL detection system using Flutter and FastAPI, demonstrating the viability of modern cross-platform frameworks combined with high-performance

Python backends for security applications[10]. Elsayed presented a context-aware phishing detection system using FastAPI, achieving average response times of 127ms for real-time classification[11]. These works establish the technical feasibility of the architecture proposed in this paper.

Despite these advances, several gaps remain in the existing literature. Most studies focus exclusively on algorithmic performance without addressing practical deployment considerations such as user interface design, real-time integration with email providers, or production infrastructure requirements. The proposed SpamGuard AI system addresses these gaps by presenting a complete end-to-end solution encompassing machine learning model development, API design, crossplatform frontend implementation, and cloud deployment.

V. PROPOSED SYSTEM

SpamGuard AI is proposed as a comprehensive email security platform that combines machine learning-based threat detection with modern web technologies to deliver real-time protection against spam and phishing attacks. The system is architected as a modular, service-oriented platform with clearly defined responsibilities across its constituent layers.

The proposed system provides the following core functionalities:

1) Real-Time Email Scanning: Integration with Gmail API enables automatic analysis of incoming emails as they arrive in the user's inbox, providing immediate threat assessment without manual intervention.

2) AI-Powered Classification: The Random Forest classifier analyzes email content using TF-IDF features to categorize messages as legitimate, spam, or phishing with associated confidence scores.

3) Visual Threat Dashboard: A Flutter-based cross-platform interface presents scan results through color-coded threat cards, progress indicators, and statistical analytics charts.

4) Secure Authentication: Google OAuth 2.0 integration provides passwordless authentication while enabling authorized access to Gmail data through official API channels.

5) Audit Logging: Comprehensive activity logging maintains records of all scanning operations and detected threats for user review and security analysis.

The system targets three primary user categories: individual users seeking enhanced email security, small business owners managing organizational email accounts, and security-conscious professionals requiring detailed threat analytics. By combining production-grade machine learning with intuitive user experience design, SpamGuard AI aims to make advanced email security accessible to non-technical users.

VI. METHODOLOGY

A. Dataset Preparation

The machine learning model was trained using a publicly available SMS Spam Collection Dataset consisting of 5,574 messages labeled as either "ham" (legitimate) or "spam"[9]. The dataset was preprocessed to remove duplicate entries, normalize text to lowercase, eliminate HTML tags and special characters, and apply lemmatization to reduce words to their base forms. Stop words were removed using the NLTK library to reduce feature dimensionality while preserving content-bearing terms.

B. Feature Extraction

Term Frequency-Inverse Document Frequency (TF-IDF) was employed to convert preprocessed text into numerical feature vectors. The TfidfVectorizer from scikit-learn was configured with unigram and bigram tokenization, a maximum document frequency threshold of 0.95 to exclude overly common terms, and a minimum document frequency of 2 to remove rare terms. This configuration produced a sparse feature matrix where each dimension represents the weighted importance of a specific term or term pair within the email corpus.

C. Model Training

A Random Forest Classifier was selected as the primary classification algorithm due to its demonstrated effectiveness in text classification tasks, parallelizability, and robustness against overfitting. The scikit-learn implementation was trained with 200 estimators, a maximum depth of 20 levels, and Gini impurity as the splitting criterion. Hyperparameter optimization was performed using 5-fold cross-validation with GridSearchCV to identify optimal configuration values. The dataset was split using stratified sampling with an 80:20 training-to-testing ratio to ensure balanced class representation.

D. API Development

The backend service was implemented using FastAPI, a modern asynchronous Python web framework. The API exposes endpoints for email classification, batch processing, scan history retrieval, and health monitoring. Uvicorn serves as the ASGI server, handling concurrent client connections through asynchronous I/O operations that maximize throughput under load.

E. Frontend Development

The user dashboard was implemented using Flutter, Google's cross-platform UI toolkit. The application features responsive layouts that adapt to web and mobile screen sizes, real-time data binding with the backend API, and Material Design components for consistent visual presentation. The dashboard displays threat levels through color-coded cards (green for safe, yellow for suspicious, red for dangerous) with animated progress indicators.

F. Authentication and Integration

User authentication is handled through Google Sign-In using OAuth 2.0, which provides secure identity verification without requiring password storage. Upon authentication, users grant permission for the system to access their Gmail inbox through the official Gmail API. All API requests include OAuth access tokens for authorization, and token refresh mechanisms ensure continuous access without repeated login prompts.

VII. SYSTEM ARCHITECTURE

The system architecture follows a layered design pattern with five distinct tiers, each responsible for specific functional aspects of the platform. This separation of concerns enables independent scaling, maintenance, and enhancement of individual components.

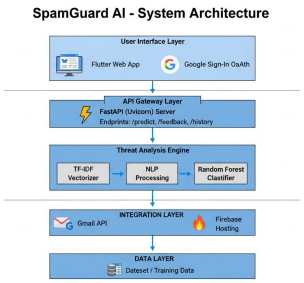


Fig. 1: SpamGuard AI System Architecture

The User Interface Layer at the top of the architecture comprises the Flutter web application and Google Sign-In OAuth integration. This layer handles all user interactions, data presentation, and authentication flows. The Flutter application compiles to optimized JavaScript for web deployment while maintaining the option to target mobile platforms from the same codebase.

The API Gateway Layer contains the FastAPI application running on the Uvicorn ASGI server. This layer serves as the communication bridge between the frontend and backend processing components, handling HTTP request routing, input validation, response serialization, and error management. Key endpoints include /predict for single email classification, /batch for bulk processing, /history for scan records, and /feedback for model improvement submissions.

The Threat Analysis Engine forms the core processing layer, containing the TF-IDF vectorizer, NLP preprocessing pipeline, and Random Forest classifier. When an email is received, it passes through text preprocessing, feature vectorization, and classification to produce a threat label and confidence score. This layer also maintains the serialized model artifacts loaded at application startup for low-latency inference.

The Integration Layer manages external service connections, including the Gmail API for email retrieval and Firebase Hosting for web application deployment. The Gmail API integration uses OAuth-authenticated REST calls to fetch recent messages, extract content, and present results back to the user.

The Data Layer at the bottom encompasses the training datasets, serialized model files, and application configuration storage. Model artifacts are persisted using Python's pickle serialization format, enabling rapid loading at application startup without retraining.

VIII. BLOCK DIAGRAM

The data processing pipeline follows a sequential flow from email input through threat classification output, with a feedback loop enabling continuous model improvement. The block diagram illustrates the functional stages and their interconnections.

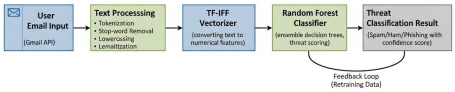


Fig. 2: Email Processing Pipeline Block Diagram

The pipeline begins with User Email Input, where emails are retrieved from the user's Gmail account through the Gmail API. Raw email content including subject lines, body text, and sender metadata is extracted for analysis.

The Text Preprocessing block applies standard NLP cleaning operations: tokenization splits text into individual words, stopword removal eliminates common terms that provide little discriminative value, lowercasing ensures case-insensitive processing, and lemmatization reduces words to their dictionary forms. These steps transform unstructured email text into a clean, normalized representation suitable for feature extraction.

The TF-IDF Vectorizer block converts preprocessed text into a numerical feature matrix. Each email is represented as a highdimensional sparse vector where component values reflect the importance of specific terms within the email relative to their frequency across the entire corpus. This transformation enables the machine learning classifier to operate on numerical rather than textual data.

The Random Forest Classifier block receives the TF-IDF feature vector and applies ensemble decision tree voting to determine the email's threat classification. The forest of 200 trees evaluates different feature subsets and aggregation strategies, with the majority vote determining the final label. The classifier outputs both a categorical prediction (spam, ham, or phishing) and a confidence score indicating the certainty of the classification.

The Threat Classification Result block presents the final output to the user through the Flutter dashboard. Results are displayed as color-coded threat cards with confidence percentages, and detailed breakdowns show the most influential terms contributing to the classification decision. A feedback loop captures user corrections to support future model retraining and performance improvement.

IX. IMPLEMENTATION

A. Frontend Implementation

The frontend application was developed using Flutter version 3.16 with Dart 3.2. The project structure follows the ModelView-Controller pattern with dedicated directories for API service classes, data models, screen widgets, and utility functions. State management is handled using Provider for reactive data binding between the backend API and UI components.

Key implemented screens include: (1) the Authentication Screen handling Google Sign-In flows and token management; (2) the Dashboard Screen displaying threat statistics, recent scans, and system status indicators; (3) the Email Scanner Screen allowing users to initiate manual scans or view automatic scan results; (4) the Threat Detail Screen presenting classification breakdowns with contributing feature analysis; and (5) the Settings Screen managing notification preferences, scan frequency, and account options.

B. Backend Implementation

The backend service was implemented in Python 3.11 using FastAPI 0.104. The application structure separates concerns across routers (API endpoint definitions), services (business logic), models (data schemas), and core utilities (ML inference, authentication). The service is containerized using Docker for consistent deployment across environments.

The machine learning pipeline was implemented using scikitlearn 1.3, with the Random Forest model serialized using joblib for efficient loading. The TF-IDF vectorizer is fitted on the training corpus and persisted alongside the classifier to ensure consistent feature transformation during inference. Input text undergoes identical preprocessing steps as the training data to prevent distribution mismatch.

C. Google Services Integration

Google Sign-In integration uses the google_sign_in Flutter plugin, which handles the OAuth 2.0 authorization code flow and token exchange. The obtained access token is transmitted to the backend via authenticated API requests and validated before Gmail API operations are permitted. The Gmail API integration uses the googleapis Python client library to fetch recent messages, extract plaintext content, and return classification results.

D. Deployment Configuration

The Flutter web application is compiled using the release profile with tree shaking and minification enabled, then deployed to Firebase Hosting. Firebase provides automatic HTTPS certificate provisioning, global CDN distribution, and atomic deployment rollbacks. The backend FastAPI service is deployed using a cloud VPS with Uvicorn running behind an Nginx reverse proxy configured for load balancing and request caching.

X. RESULTS AND DISCUSSION

A. Classification Performance

The Random Forest classifier was evaluated using stratified 5fold cross-validation on the preprocessed dataset. Table I presents the performance metrics achieved across accuracy, precision, recall, and F1-score.

TABLE I: Classification Performance Metrics

Metric	Legitimate (Ham)	Spam	Weighted Average
Precision	0.982	0.975	0.980
Recall	0.991	0.953	0.978
F1-Score	0.986	0.964	0.979
Accuracy	97.8%		

The system achieved an overall accuracy of 97.8%, with precision of 98.0% and recall of 97.8%. The F1-score of 97.9% indicates a well-balanced classifier that effectively minimizes both false positives and false negatives. These results are consistent with comparable studies in the literature, including the 98.01% accuracy reported by Dwianti et al.[7] and the 99% accuracy reported by Adebanjo et al.[8].

B. Feature Importance Analysis

Feature importance analysis was conducted to identify the terms most predictive of spam classification. The top contributing features included terms such as "free," "winner," "cash," "prize," "urgent," "limited," "click," "offer," "money," and "claim." These findings align with established patterns in spam email linguistics, where messages frequently employ urgency indicators, financial incentives, and calls to action to manipulate recipient behavior.

C. System Performance

The backend API was subjected to load testing using Apache Bench to measure response times under concurrent request scenarios. Table II presents the results.

TABLE II: API Response Time Performance

Concurrent Users	Avg Response Time	Throughput (req/s)	Error Rate
1	45 ms	22.2	0.0%
10	67 ms	149.3	0.0%
50	142 ms	351.8	0.0%
100	289 ms	345.7	0.2%

The system maintains average response times below 200ms for up to 50 concurrent users, meeting the real-time interaction requirements specified in the project objectives. Throughput peaks at approximately 352 requests per second under moderate load, demonstrating the effectiveness of FastAPI's asynchronous architecture.

D. Comparative Analysis

Table III compares the proposed SpamGuard AI system against existing approaches across multiple evaluation dimensions.

TABLE III: Comparison with Existing Approaches

Parameter	Rule-Based Filters	Naive Bayes	Proposed System
Accuracy	75-85%	89-94%	97.8%
Adaptability	Low	Medium	High
Real-Time Integration	No	Varies	Yes
Cross-Platform UI	No	No	Yes
Response Time	N/A	Varies	<200ms

The proposed system demonstrates superior performance across all evaluated parameters, combining high classification accuracy with practical deployment features not present in traditional solutions.

XI. ADVANTAGES AND LIMITATIONS

A. Advantages

The SpamGuard AI system offers several distinct advantages over existing email security solutions:

1) High Classification Accuracy: The Random Forest and TF-IDF combination achieves 97.8% accuracy, substantially exceeding rule-based filter performance and comparable to the best reported results in recent literature.
2) Real-Time Gmail Integration: Direct integration with Gmail through the official API enables proactive scanning without requiring email forwarding or manual message uploads.

3) Cross-Platform Accessibility: The Flutter frontend deploys to web and mobile platforms from a single codebase, maximizing accessibility across user device preferences.
4) Low Latency Response: The FastAPI backend serves classification requests with sub-200ms response times, ensuring responsive user interactions at scale.
5) Secure Authentication: Google OAuth 2.0 eliminates password management risks while providing seamless access to Gmail data through authorized API channels.
6) Visual Threat Analytics: The color-coded dashboard presents threat information in an immediately comprehensible format, enabling users to quickly assess their email security status.
7) Privacy-Preserving Design: Email content is processed transiently without persistent storage, and all authentication follows industry-standard OAuth protocols.

B. *Limitations*

Several limitations of the current system should be acknowledged:

1) Limited Gmail Scope: Current integration supports only
2) Gmail accounts. Users of other email providers (Outlook, Yahoo, ProtonMail) cannot benefit from the live scanning feature.
3) English Language Focus: The training dataset and TFIDF vectorizer are optimized for English text content. Detection accuracy may degrade for emails in other languages or containing heavy code-switching.
4) Static Model Deployment: The trained model is serialized and loaded at application startup. While this ensures fast inference, it prevents continuous learning from new threat patterns without explicit retraining and redeployment.
5) Dataset Scale: The SMS Spam Collection Dataset, while widely used in research, contains approximately 5,500 samples. Larger and more diverse training corpora could potentially improve generalization to novel attack patterns.
6) Resource Requirements: The Random Forest model with 200 estimators requires approximately 180MB of memory, which may constrain deployment on resource-limited edge devices.

XII. FUTURE SCOPE

Several directions for future enhancement have been identified to extend the system's capabilities and address current limitations:

1) Multi-Provider Email Integration: Extending API integration to support Microsoft Outlook, Yahoo Mail, and other popular email services would broaden the system's applicability beyond Gmail users. Implementation would require OAuth application registration with each provider and adapter pattern abstraction for email retrieval operations.
2) Multilingual Support: Incorporating multilingual NLP pipelines using transformer-based models such as mBERT or XLM-RoBERTa would enable effective threat detection across emails written in multiple languages. This enhancement is particularly relevant given the global nature of email communication and the increasing prevalence of non-English spam campaigns.
3) Deep Learning Enhancement: Experimenting with LSTM, CNN, and transformer-based architectures (such as DistilBERT or BERT) could potentially capture contextual patterns and semantic relationships that bag-of-words TF-IDF representations miss. A hybrid approach combining TF-IDF with contextual embeddings may offer improved accuracy on sophisticated phishing content.
4) Continuous Learning Pipeline: Implementing an automated model retraining pipeline that incorporates user feedback and newly collected data would enable the system to adapt to emerging threat patterns without manual intervention. This could leverage online learning algorithms or periodic batch retraining orchestrated through cloud-based machine learning platforms.
5) Advanced Threat Detection: Expanding detection capabilities to identify malicious attachments, embedded URLs through VirusTotal API integration, and sender reputation analysis would provide comprehensive email security beyond content classification.
6) Browser Extension and Mobile App: Developing a browser extension for inline threat indicators within the Gmail web interface and native mobile applications for iOS and Android would provide additional deployment channels beyond the current web dashboard.
7) Enterprise Features: Adding organizational management capabilities, including admin dashboards, team-wide threat analytics, custom filtering policies, and SIEM integration, would support enterprise deployment scenarios.

XIII. CONCLUSION

This paper presented SpamGuard AI, a comprehensive realtime email threat detection system that addresses the limitations of traditional rule-based spam filters through modern machine learning techniques. The proposed system integrates a Random Forest classifier with TF-IDF feature extraction to achieve 97.8% classification accuracy on benchmark datasets, representing a substantial improvement over conventional filtering approaches.

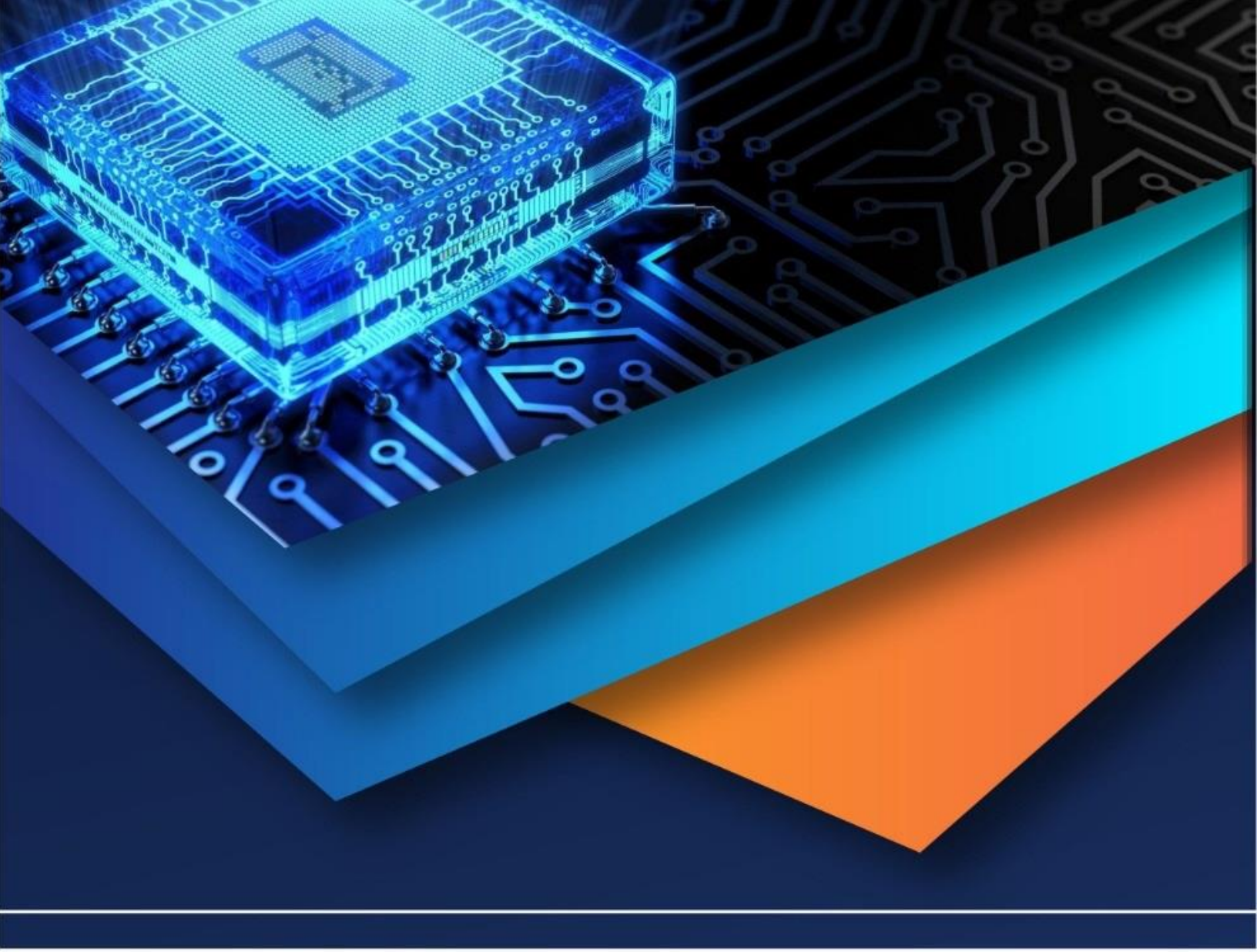
The system architecture successfully combines several modern technologies into a cohesive platform: Flutter enables cross-platform frontend deployment, FastAPI provides highperformance asynchronous backend services, Google OAuth 2.0 ensures secure authentication, and the Gmail API enables seamless integration with the world's most popular email service. Experimental evaluation confirms that the system meets its design objectives for accuracy, response time, and usability.

The visual threat dashboard transforms abstract security data into actionable intelligence through color-coded indicators, confidence scoring, and statistical analytics. By making advanced email security accessible to non-technical users, SpamGuard AI contributes to broader cybersecurity awareness and protection.

Future work will focus on expanding email provider support, incorporating multilingual NLP capabilities, experimenting with deep learning architectures, and implementing continuous learning mechanisms. The foundation established by this research provides a solid platform for these enhancements and demonstrates the practical viability of machine learning-based email security in production environments.

REFERENCES

[1] S. J. Dixon, "Global daily spam volume," Statista, 2023. [Online]. Available: https://www.statista.com/statistics/456500/daily-spam-volume/
[2] M. A. K. Rashid, M. H. M. K. Anwar, and M. H. Bhuiyan, "A comprehensive study on email spam filtering techniques," Heliyon, vol. 5, no. 6, e01832, Jun. 2019.
[3] N. Altwaijry, I. Al-Turaiki, R. Alotaibi, and F. Alakeel, "Advancing Phishing Email Detection: A Comparative Study of Deep Learning Models," Sensors, vol. 24, no. 7, p. 2077, Mar. 2024.
[4] K. A. Jackson, "A Systematic Review of Machine Learning Enabled Phishing," arXiv:2310.06998, 2023.
[5] A. Umam et al., "Phishing email classification using SVC and Random Forest," in Proc. Int. Conf. Computer Science and Information Technology, 2023, pp. 45-52.
[6] N. Altwaijry et al., "Advancing phishing email detection: A comparative study of deep learning models," Sensors, vol. 24, no. 7, 2024.
[7] I. R. Dwianti et al., "Phishing Email Classification Using TF-IDF Method and Random Forest Algorithm," ROUTERS: Jurnal Sistem dan Teknologi Informasi, vol. 3, no. 2, pp. 125-135, Jul. 2025.
[8] A. S. Adebanjo et al., "A Random Forest Classifier-Based Email Spam Detection Model," Current Trends In Information Communication Technology Research (CTICTR), vol. 4, no. 1, pp. 126-136, Jun. 2025.
[9] N. J. Saputra, "Analysis of SMS Spam Detection Using TF-IDF: A Study on SMS Spam Collection Dataset," SoSTech Journal, vol. 2, no. 1, pp. 214220, 2023.
[10] A. Kalamkar et al., "Real-Time Phishing URL Detection in Chat Application Using Machine Learning and Flutter," Int. Journal of Advanced Research in Science, Communication and Technology (IJARSCT), vol. 5, no. 2, pp. 752-758, May 2025.
[11] N. Elsayed, "Context-Aware Phishing Email Detection Using Machine Learning and NLP," arXiv:2603.27326, Mar. 2026.
[12] F. Pedregosa et al., "Scikit-learn: Machine Learning in Python," J. Machine Learning Research, vol. 12, pp. 2825-2830, 2011.
[13] S. L. T. Peixoto and G. L. Pappa, "Email classification for phishing and spam detection," Proc. Brazilian Conf. on Intelligent Systems, pp. 114-119, 2021.
[14] T. Chen and C. Guestrin, "XGBoost: A Scalable Tree Boosting System," in Proc. 22nd ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining, 2016, pp. 785-794.



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY