APPLIED INFORMATION TECHNOLOGY AND COMPUTER SCIENCE
e-ISSN: 2773-5141

AITCS

Vol. 7 No. 1 (2026) 1165-1185
https://publisher.uthm.edu.my/periodicals/index.php/aitcs

SpoofSentry: Email Spoofing Detection Tool Using Domain-Based Authentication

Nur Fatin Nazifa Mohd Nazlan[1], Siti Fadzlun Md Salleh[1]*

[1] *Fakulti Sains Komputer dan Teknologi Maklumat*
Universiti Tun Hussein Onn Malaysia, Parit Raja, Batu Pahat, 86400, MALAYSIA

*Corresponding Author: sfadzlun@uthm.edu.my
DOI: https://doi.org/10.30880/aitcs.2026.07.01.063

Article Info

Received: 3 August 2025
Accepted: 23 June 2026
Available online: 30 June 2026

Keywords

Email Spoofing, Cybersecurity, SPF, DKIM, DMARC, Spoofing Detection, Email Security, Validation, User-Friendly Interface

Abstract

Email spoofing poses a significant cybersecurity threat, leading to phishing attacks and business email compromise with billions in global losses. This project develops SpoofSentry, a comprehensive Email Spoofing Detection Tool that integrates Sender Policy Framework (SPF), Domain Keys Identified Mail (DKIM), and Domain-based Message Authentication, Reporting and Conformance (DMARC) with Random Forest machine learning for enhanced threat detection accuracy. The desktop application features hybrid analysis combining traditional authentication verification with intelligent pattern recognition, automated threat classification, and alert systems. The tool provides user-friendly interfaces for single email analysis and bulk processing, with automatic whitelist and blacklist management and reporting capabilities. Testing demonstrates effectiveness in identifying sophisticated spoofing, enabling organizations to implement enterprise-level email security without extensive technical expertise.

1. Introduction

Email communication is essential for modern business, with billions of messages sent daily across organizations worldwide. However, this widespread use has created security problems through email spoofing attacks that trick users by appearing to come from trusted sources. Email spoofing is a major cybersecurity threat where attackers fake email headers to look legitimate [1]. The FBI reports that business email compromise attacks have caused over $43 billion in losses globally [2], showing the urgent need for better email security tools. Current email spoofing detection has major problems. Email authentication protocols like SPF, DKIM, and DMARC provide basic security but work separately and need manual checking [3]. Most existing solutions focus on single protocols without combining different verification methods together. Manual checking creates delays and increases mistakes, while the lack of smart detection technology limits protection against new spoofing methods. This project aims to create a complete Email Spoofing Detection Tool that combines email authentication protocols with machine learning technology. The main goals include building a system that uses multiple authentication protocols with machine learning, creating a smart analysis tool with automatic threat detection, and testing how well the system works. This project focuses on building a desktop system for IT administrators who manage email security and system administrators who monitor user activities. This project expects to create important results for email security and cybersecurity tool development. The main expected result is a working SpoofSentry Tool that shows better detection accuracy by combining authentication protocols with machine learning. The system should achieve over 95% accuracy while keeping false alarms low for smooth operations. This project is also important beyond just creating one security tool, helping improve cybersecurity awareness and capabilities across different organizations. By providing an integrated approach to email spoofing detection, this project fills important gaps in current email security practices. The project shows how traditional authentication methods can

work with modern machine learning techniques to create better cybersecurity solutions that organizations with limited resources can use.

2. Literature Review

The literature review examines email spoofing detection using SPF, DKIM, and DMARC and machine learning to address phishing and data breaches. It evaluates tools like Mimecast, ProofPoint, and EmailAuth, highlighting their strengths and limitations. These insights inform the development of an automated, user-friendly tool for organizations with limited technical resources, forming the basis for the system's design and implementation.

2.1 Email Spoofing Detection Tool

Email spoofing detection tools identify and prevent spoofing attempts by analyzing email headers and performing domain-based authentication checks using SPF, DKIM, and DMARC protocols. These tools examine the sender's IP address, message routing history, and authentication records to determine if an email is legitimate or forged. The detection works by extracting email header information through efficient processing methods that focus on relevant data rather than capturing entire system memory. Popular email security solutions like Mimecast, ProofPoint, and EmailAuth incorporate these protocols to provide comprehensive protection against spoofing, phishing, and other email-based threats [4].

2.2 Email Authentication

Domain-based Message Authentication, Reporting and Conformance (DMARC) enhances SPF and DKIM by adding Identity Alignment, Policy Management, and Reporting. It protects against phishing by enabling authentication agreements, providing feedback, and guiding action on unauthorized emails. DMARC overcomes SPF and DKIM limitations, ensuring email integrity, improving sender control, and reducing spam, phishing, and spoofing risks [5].

2.3 Validation and Detection

SpoofSentry uses SPF, DKIM, and DMARC protocols combined with Random Forest machine learning to validate email authenticity and detect spoofing attacks. The tool analyzes authentication results, email content, header anomalies, and behavioral patterns through a unified threat assessment system. This hybrid approach provides automated threat classification, alerts, and comprehensive reporting to defend against sophisticated spoofing attacks.

2.3.1 Sender Policy Framework (SPF)

Is a technical standard and email authentication technique that assists with validating and authenticating an email senders' original source. SPF works by creating an SPF record, that is stored in the DNS [6], which specifies which IP address or mail server the domain owner uses to send an email. SPF authentication involves the receiving mail server verifying the domain by comparing the "envelop from" address in the email header and the IP address to ensure it matches the SPF record [7].

2.3.2 Domain Keys Identified Mail (DKIM)

DKIM is an email authentication protocol that ensures email integrity by verifying the sender's identity and detecting tampering. It adds a digital signature, generated by a Mail Transfer Agent (MTA) using hash values, to each message [8]. The receiver uses the public key in the DNS to decrypt the signature and compare it with the recalculated hash value from the email. If they match, the email is confirmed as unaltered and from the claimed domain.

2.3.3 Domain-Message Authentication Reporting and Conformance (DMARC)

DMARC enhances SPF and DKIM by enforcing policies for handling failed checks. It provides domain owners with control over email authentication and detailed reports on failures, aiding in combating spoofing. However, its effectiveness depends on correctly configured SPF and DKIM, as misconfigurations can cause delivery issues or false positives [9].

2.3.4 Random Forest Machine Learning

Random Forest is an ensemble machine learning algorithm that combines multiple decision trees to make predictions using different subsets of training data and features. For SpoofSentry, Random Forest analyzes SPF, DKIM, DMARC authentication results, email headers, content patterns, and sender behavior to distinguish legitimate emails from spoofed emails. This project demonstrates that Random Forest achieves accuracy rates

between 85% and 98% when applied to email authentication features [10], while being resistant to overfitting and capable of detecting previously unseen spoofing techniques.

2.4 Study of the Existing Tools

The following Mimecast, ProofPoint and EmailAuth are widely adopted solutions in the industry, each with clear capabilities and limitations.

2.4.1 Mimecast

Mimecast provides a cloud-based email security solution that safeguards against spoofing, phishing, and malware using SPF, DKIM, and DMARC for email validation. Its advanced threat protection is ideal for large enterprises, though its complexity and cost may be a barrier for smaller organizations [11]

2.4.2 Proofpoint

ProofPoint offers strong email security using advanced machine learning, SPF, DKIM, and DMARC, along with detailed threat analytics. It is favored by enterprises for its strong features and seamless integration with security infrastructures but can be expensive and require specialized expertise to manage [12].

2.4.3 EmailAuth

EmailAuth is a user-friendly tool designed for small to mid-sized organizations to implement SPF, DKIM, and DMARC authentication with minimal complexity and cost [13]. It offers basic threat protection but lacks the advanced features and detailed reporting of solutions like Mimecast and ProofPoint, making it less suitable for large-scale environments.

2.4.4 The Proposed System

The proposed SpoofSentry uses SPF, DKIM, and DMARC protocols combined with Random Forest machine learning to validate email authenticity and prevent spoofing attacks. With a desktop interface, it allows users to upload EML files for automated analysis that integrates authentication verification with intelligent threat assessment. Key features include classification results indicating spoofed or authentic emails, automated email alerts, bulk processing capabilities, and comprehensive reporting, enabling IT administrators to monitor and respond to threats effectively while enhancing overall email security through hybrid detection methods.

2.4.5 Comparison Table

Table 1 shows the comparison of Mimecast, ProofPoint, and EmailAuth with the proposed system. While existing tools offer comprehensive email security with advanced features like phishing protection and threat analytics, they are expensive and complex. The proposed system focuses on email spoofing detection using SPF, DKIM, and DMARC, emphasizing affordability and simplicity. With a user-friendly interface and automated analysis, it delivers actionable insights tailored to spoofing risks, which are accessible to organizations of all sizes.

Table 1 *Comparison Table of Three Existing Tools and Proposed System*

Systems	Mimecast	Proofpoint	EmailAuth	The Proposed System
Threat Detection	Yes	Yes	No	Yes
SPF Support	Yes	Yes	Yes	Yes
DKIM Support	Yes	Yes	Yes	Yes
DMARC Support	Yes	Yes	Yes	Yes
Detailed Reporting	Yes	Yes	No	Yes
Ease of Use	No	No	Yes	Yes
Suitable for Large Organizations	Yes	Yes	No	Yes
Suitable for Small to Mid-Size Organizations	Yes	No	Yes	Yes
Collect Email Data	Yes	Yes	Yes	Yes
History Maintenance	Yes	Yes	Limited	Yes
Cost	Yes	Yes	Yes	No

3. Methodology

This section outlines the development of the SpoofSentry guided by Prototyping Methodology for iterative refinement. Key phases include planning (defining objectives and scope), analysis (examining SPF, DKIM, DMARC, and existing tools), design (structuring system components), implementation (iterative testing and feedback), and finalization (integration and testing). This approach ensures a reliable, user-friendly tool for detecting email spoofing.

3.1 Prototyping Methodology

The Prototyping Model as shown in Fig. 1, a common SDLC approach, is ideal for projects with evolving requirements. It involves creating an initial prototype, refining it iteratively through user feedback, and using it as the foundation for the final product. The process starts with gathering requirements, followed by developing a basic prototype for user evaluation and improvement. This cycle continues until the prototype meets expectations, making it effective for projects like the SpoofSentry, where user needs and priorities evolve [14].

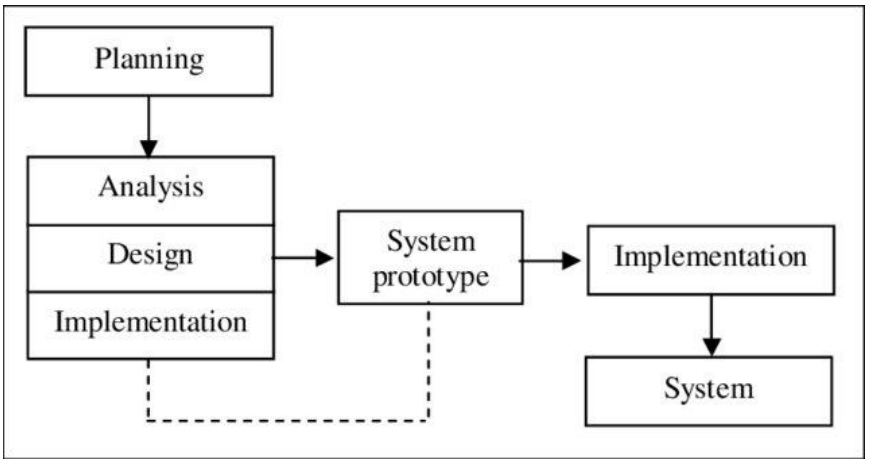


Fig. 1 *Prototyping Methodology [14]*

3.1.1 Planning Phase

The planning phase is crucial for initiating the SpoofSentry project. Key steps include selecting a supervisor specializing in information security for guidance, choosing a project title that addresses the significant cybersecurity issue of email spoofing, and presenting it in a title defense session for feedback and validation. The phase also involved defining the problem statement, objectives, and scope to ensure the project is user-friendly, effective, and achievable within available resources.

3.1.2 Analysis Phase

The analysis phase begins with thorough research on the project, focusing on the importance of detecting email spoofing and the effectiveness of SPF, DKIM, and DMARC protocols. Information is gathered from reliable sources, including articles, journals, case studies, and existing tools like Mimecast, ProofPoint, and EmailAuth, to understand advancements, challenges, strengths, and limitations.

3.1.3 Design Phase

The design phase outlines the SpoofSentry's functionality and components. Users can upload email files (.eml) through the Input Interface. The Email Parsing Module extracts key header information, while the Authentication Validation Module checks SPF, DKIM, and DMARC records for authenticity. The Result Display component shows outcomes with visual indicators, such as color coding or labels ("PASS"/"FAIL"), to highlight email legitimacy.

3.1.4 Implementation Phase

The implementation phase marks the development of SpoofSentry. A functional prototype is created and refined through iterative testing and user feedback. The initial prototype includes basic features like uploading .eml files, parsing headers, and validating protocols to detect spoofed emails. User feedback from target users, such as IT administrators, helps improve the tool's usability, interface, and accuracy.

3.1.5 System Prototype Phase

The SpoofSentry system prototype underwent continuous improvement through iterative development cycles based on user feedback and testing requirements. Initial feedback led to enhanced user interface design with professional color schemes and consistent styling across all application components. Users requested bulk processing capabilities for multiple EML files, which was integrated with progressive result display and visual feedback for better user experience. Additional improvements included automatic whitelist and blacklist management, enhanced PDF reporting with professional formatting, and email alert notifications with configurable SMTP settings. The prototype incorporated Random Forest machine learning features providing confidence scores and threat level classifications while maintaining a user-friendly desktop interface for non-technical administrators to manage email security effectively.

3.1.6 System Finalization Phase

The finalization phase integrates all system components and tests the tool for reliability in analyzing legitimate and spoofed emails, validating SPF, DKIM, and DMARC, and providing user-friendly feedback. Issues found during testing are addressed to improve performance. Comprehensive documentation, including user guides and technical support, is prepared for future maintenance. Once fully integrated and tested, the tool is ready for deployment to enhance email security.

3.1.7 Prototyping Methodology Phase

Table 2 *Prototype Methodology Phase*

Phase	Description	Outcome
Planning	• Choose a supervisor who specializes in information security • Find potential ideas • Title defense session with panel • Identify the problem statement, objectives, and scope of the project	A project Proposal and title approved during title defense session
Analysis	Conducting thorough research on the project title. Converted system requirement into Use Case Diagram, Use Case Specification (Activity Diagram) and Class Diagram	Use Case Diagram, Use Case Specification, Activity Diagram and Class Diagram
Design	Development of the system input interface, email parsing module, authentication validation module and result display interface	System interface, System output, System database and Flowchart
Implementation	Test the prototype and get user feedback	SpoofSentry basic version
System Prototype	Improve the system prototype according to the addition requirements from user feedback	SpoofSentry second version
Implementation	Test the enhanced system prototype	SpoofSentry enhanced version
System Finalization	All individual components of the system are combined into a fully functioning and cohesive system undergoes rigorous testing to confirm its reliability and effectiveness	SpoofSentry completed version

4. Analysis and Design

Focuses on the development process of the email analysis tool, covering its requirements, analysis, and design. It starts with a breakdown of functional, non-functional, user, software, and hardware requirements to ensure the system meets user needs and is technically feasible. The section then presents a use case diagram and specifications to illustrate user interactions, followed by a class diagram to define the system's structure. A flowchart visualizes the workflow, while database design is explained with a data dictionary for clarity.

4.1 System Requirements

This outlines the system requirements needed for the SpoofSentry, including functional and non-functional requirements, as well as user, software, and hardware requirements.

4.1.1 Functional Requirements

Functional requirements in Table 3 specify the core features and operations the system must perform to fulfil its purpose. These requirements define "what" the system does, focusing on tasks and services that provide value to the user. They ensure that the system behaves as expected under specified conditions and meets user needs.

Table 3 *Functional Requirements for SpoofSentry*

Module	Functional Requirements
User Authentication	The system must authenticate users using valid company name, username, and password
User Registration	The system must allow new users to register with unique credentials and company information
Email Analysis	The system should allow users to upload .eml files for individual and bulk analysis
Spoof Detection	The system must check email headers for SPF, DKIM, and DMARC records to determine if the email is spoofed
Machine Learning	The system must apply Random Forest algorithm for enhanced spoofing detection with confidence scores
Result Display	The system must display analysis results in an easy-to-read format on the graphical user interface
Report Generation	The system should allow users to save analysis results as PDF files with detailed information
Email Alerts	The system sends alert emails when spoofed emails are detected to designated addresses
Dashboard Management	The system must provide a dashboard for viewing analysis history and managing email lists
Whitelist/ Blacklist	The system must automatically manage whitelist and blacklist based on analysis results
Admin Functions	The system must provide admin capabilities for user management and system monitoring
Logout Functionality	The system must allow users to log out securely and return to the main menu

4.1.2 Non-Functional Requirements

Non-functional requirements in Table 4 focus on the overall quality and operational aspects of the system rather than specific behaviors. These requirements ensure the system is user-friendly, secure, reliable, and performs efficiently under various conditions.

Table 4 *Non-Functional Requirements for SpoofSentry*

Aspects	Non-Functional Requirements
Performance	The email analysis should be completed within 5 seconds for a single .eml file.
Usability	The user interface must be intuitive and easy to navigate, even for non-technical users.
Security	Sensitive data, such as user credentials, must be securely handled and stored using encryption.
Reliability	The system should maintain consistent functionality without crashes during normal operation
Scalability	The system must handle at least 100 email analyses per day without performance degradation.
Portability	The system should be compatible with Windows operating systems and require minimal setup.

Table 4 *(Cont.)*

Aspects	Non-Functional Requirements
Response Time	The system interface should respond to user actions within 2 seconds.
Data Integrity	All analysis results and user data must be stored accurately and remain consistent.
Maintainability	The system code should be well-documented and modular for easy maintenance and updates

4.1.3 User Requirements

User requirements in Table 5 are essential for ensuring that the system meets the needs and expectations of its end-users. These requirements describe how users will interact with the system and the key features they rely on. By addressing user requirements, the system becomes more practical, accessible, and valuable to its intended audience.

Table 5 *User Requirements for SpoofSentry*

Role	User Requirements
IT Administrator	Users must be able to register and log in using secure company name, username, and password
	Users should be able to upload email files for analysis without needing technical expertise
	The analysis results must be clearly displayed and easy to understand with visual indicators
	Users should be able to save analysis results as PDF reports for documentation purposes
	Users need a feature to send alerts for suspicious emails to designate email addresses
	Users must have access to a dashboard showing analysis history and email management features
Administrator	Admins must be able to monitor user login activities and manage user accounts
	Admins should have access to system-wide analytics and reporting capabilities
	Admins must be able to suspend user accounts and control user behaviour when necessary
	Admins should be able to view user credentials with creation dates for security purposes

4.1.4 Software Requirements

The software requirements outline in Table 6 gives the specific tools, platforms, and libraries needed to develop and run the SpoofSentry. These requirements ensure that the system functions properly on the target environment and provides the necessary features and capabilities.

Table 6 *Software Requirements for SpoofSentry*

Type	Software	Requirements
Operating System	Windows 10 or higher	To launch the tool and provide stable platform.
Programming Language	Phyton 3.9 or higher	For tool backend development and
Development Tools	Visual Studio Code	machine learning implementation
GUI Framework	tkinter	For integrated development environment for coding
Image Processing	Pillow	For graphical user interface development
Email Services	smtplib	For handling image-related functionalities in the interface
Report Generation	reportlab	For sending email alerts and notifications
Email Processing	email and re modules	For generating PDF reports with professional formatting
Machine Learning	scikit-learn	For email parsing and analyzing email content

4.1.5 Hardware requirements

The hardware requirements in Table 7 specify the minimum physical resources required to run the SpoofSentry efficiently. The hardware needs are defined to ensure that the system performs well and does not face rashes when processing email analysis tasks.

Table 7 *Hardware Requirements for SpoofSentry*

Hardware	Specifications
Processor	Intel Core i3 or equivalent AMD processor for adequate processing power
Memory (RAM)	4 GB minimum, 8 GB recommended for optimal machine learning performance
Storage	500 MB of disk space for application and database storage
Display	Minimum resolution of 1280x720 pixels for proper interface display
Input Devices	Keyboard and mouse for user interaction with the graphical interface
Network	Internet connection for sending email alerts and DNS record lookups
Graphics	Integrated graphics sufficient for GUI rendering and display

4.2 System Analysis

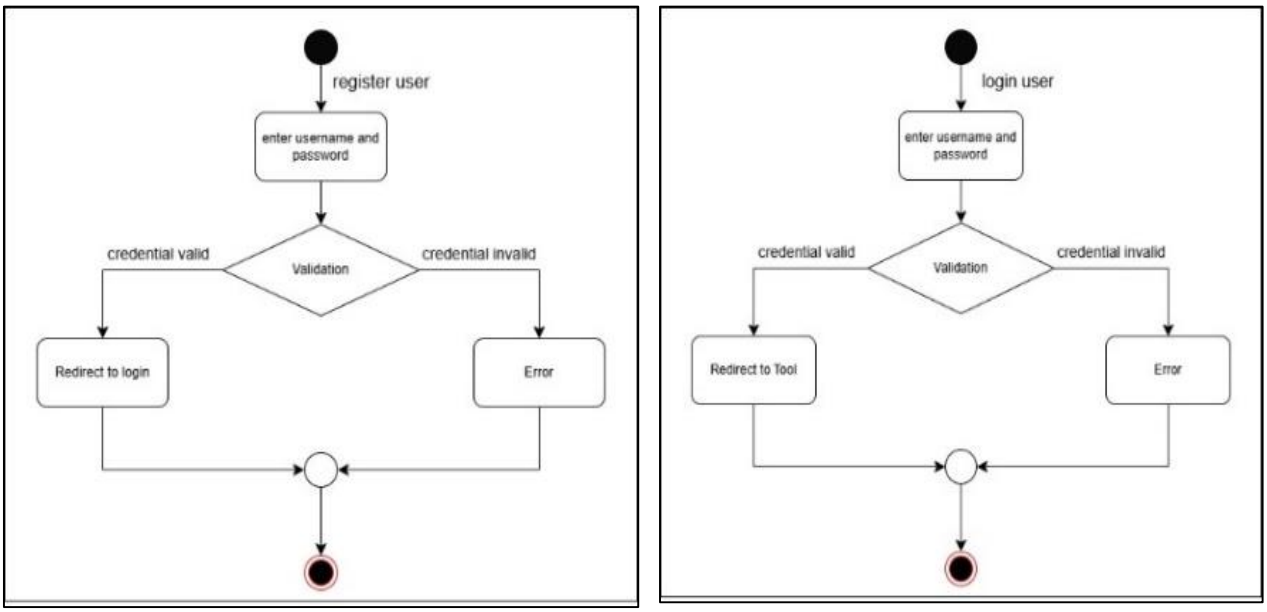
The system analysis of SpoofSentry involves understanding how the system will work, how users will interact with it, and how the various components will function together. This stage helps to design the system architecture, ensuring that all requirements are met and the tool operates efficiently.

4.2.1 Use Case Diagram

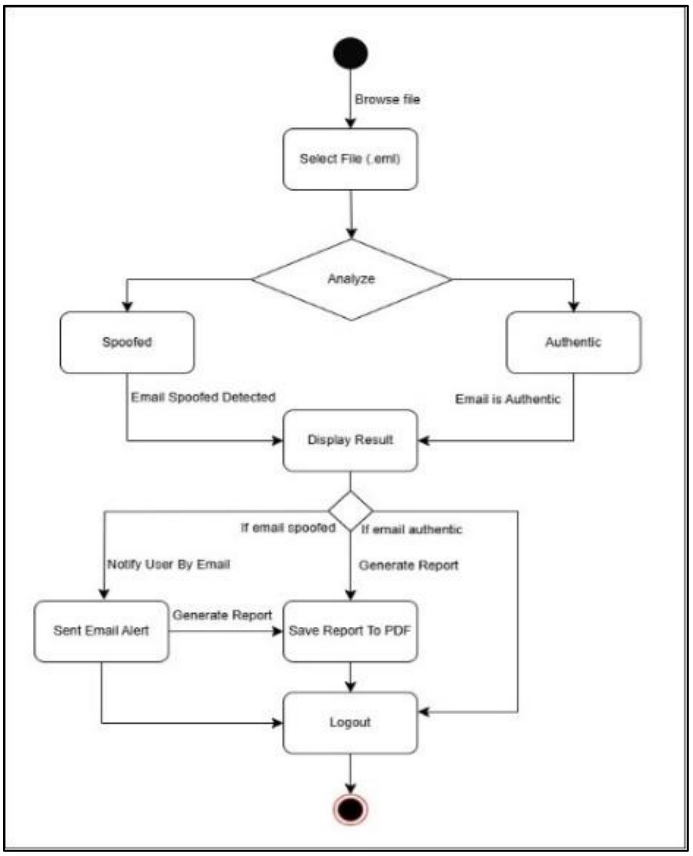
A Use Case Diagram shown in Fig. 2 in Appendix A visually represents the interactions between users and SpoofSentry, defining the system's key actions. Primary use cases include user registration and authentication, uploading EML files for analysis, performing hybrid spoofing detection through SPF, DKIM, DMARC verification combined with Random Forest machine learning classification, and displaying results with threat level assessment.

4.2.2 Use Case Specification

A Use Case Specification in Fig. 3, outlines user interactions with SpoofSentry for tasks like registration, authentication, and comprehensive email analysis. Users register with unique credentials and company information, log in with role-based access, and upload EML files for hybrid analysis combining SPF, DKIM, DMARC verification with Random Forest machine learning classification. Results display spoofed or authentic classifications with threat levels and confidence scores, while the system automatically manages whitelist/blacklist entries, generates reports, sends email alerts, and supports both single and bulk processing operations.



(a) (b)



(c)

Fig. 3 Use Case Specification of (a) Register Activity Diagram; (b) Login Activity Diagram; (c) Email Analysis Activity Diagram

4.2.3 Class Diagram

A Class Diagram shown in Fig. 4 in Appendix B represents the structure of the system by showing the different classes and their relationships. The class diagram helps to organize the system's components and their interactions, which is crucial for designing the system in an object-oriented way

4.3 System Design

The System Design phase involves creating a detailed plan for building SpoofSentry, translating requirements into a functional blueprint that ensures scalability, efficiency, and user-friendliness by organizing system components and interactions. This phase encompasses the integration of authentication protocols with machine learning capabilities, database design for storing analysis results and user management, and the desktop application architecture using Python and Tkinter. A flowchart visually outlines the system's process flow, from user authentication and email file upload to comprehensive analysis combining SPF, DKIM, DMARC verification with Random Forest classification, result display with threat level assessment, automated whitelist/blacklist management, email alert notifications, and PDF report generation.

4.3.1 Flowchart

For SpoofSentry, the flowchart shown in Fig. 5 in Appendix C outlines key steps: launching the system with automatic ML model training, user registration and authentication; uploading and validating EML email files; performing hybrid analysis combining SPF, DKIM, DMARC verification with Random Forest machine learning classification; displaying results with threat level assessment and confidence scores; automatically managing whitelist/blacklist entries; generating PDF reports and sending email alerts for spoofed emails; and secure logout. It provides a clear and systematic overview of the tool's workflow.

4.4 Database Design

The database design for SpoofSentry focuses on user authentication, storing comprehensive email analysis results, and automated list management using SQLite for local data storage. It includes a user's table for

usernames, hashed passwords, and company information ensuring secure authentication with role-based access control. The email_analysis table stores detailed metadata including SPF, DKIM, DMARC results, machine learning confidence scores, threat classifications, suspicious keyword counts, and header anomaly scores for comprehensive analysis tracking.

4.5 Interface Design

The interface design in Fig. 6 focuses on creating a user-friendly and intuitive graphical interface for the email analysis tool. It includes essential features like file selection, analysis initiation, result display, and actionable options such as generating PDF reports, sending spoofing alerts, and user authentication. Fig. 6(a) shows the Main Interface for the tool. Fig 6(b) displays the User Login Interface. User will be directed to this interface right after registering. Fig 6(c) shows the User Registration Interface. Users need to register first before login. Fig 6(d) shows the SpoofSentry Interface.

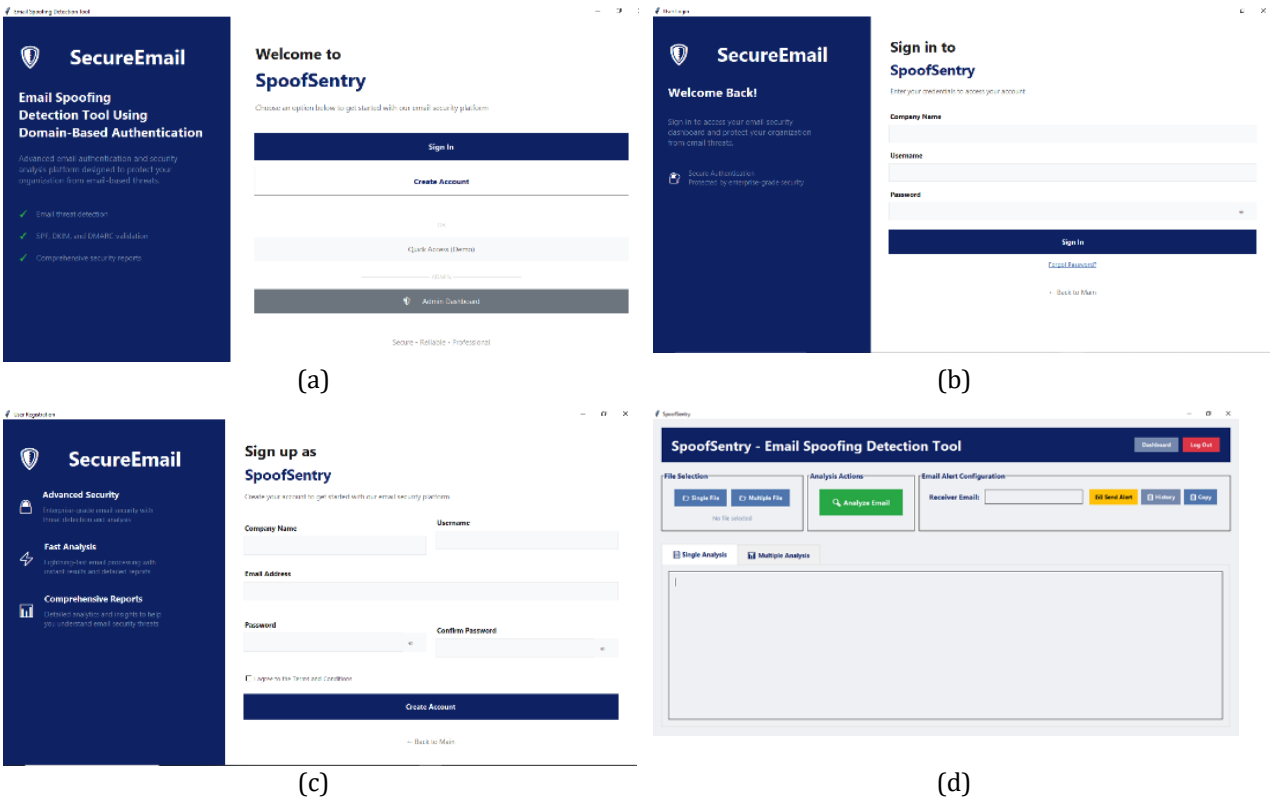


Fig. 6 *Interface Design*

5. Implementation and Testing

This section presents the implementation and testing of SpoofSentry. The implementation integrates SPF, DKIM, DMARC validation with Random Forest machine learning for enhanced spoofing detection accuracy. Comprehensive testing through unit, functional, and user acceptance testing validates that the system meets all requirements and effectively detects email spoofing attempts

5.1.1 Implementation of Security Module

The security module forms the foundation of the Email Spoofing Detection Tool Using Domain-Based Authentication, implementing robust user authentication, secure password handling, and session management to protect user data and system integrity. This module ensures that only authorized users can access the system while main training security best practices throughout the application lifecycle

5.1.2 Implementation of User Authentication System

The user authentication system in Fig. 7 implements secure login and registration functionality using bcrypt password hashing and SQLite database storage. The implementation follows NIST password guidelines and includes comprehensive input validation to prevent security vulnerabilities.

```python
def hash_password(password):
    """Hash password using bcrypt with salt"""
    salt = bcrypt.gensalt()
    hashed = bcrypt.hashpw(password.encode('utf-8'), salt)
    return hashed.decode('utf-8')

def verify_password(stored_password, provided_password):
    """Verify password against stored hash"""
    return bcrypt.checkpw(provided_password.encode('utf-8'),
                          stored_password.encode('utf-8'))
```

(a)

```python
def validate_login():
    """Validate user login credentials with security measures"""
    company_name = entry_company.get().strip()
    username = entry_username.get().strip()
    password_raw = entry_password.get()

    # Input validation and sanitization
    if not all([company_name, username, password_raw]):
        messagebox.showerror("Validation Error", "All fields are required")
        return

    # Database query with parameterized statements
    cursor.execute("SELECT id, password FROM users WHERE company_name = ? AND username = ?",
                   (company_name, username))
    user = cursor.fetchone()
```

(b)

Fig. 7 *Implementation of (a) Password Hashing and Verify User Password; (b) Login Validation*

The authentication system includes failed login attempt tracking and account lockout protection to prevent brute force attacks. Login activities are logged with timestamps and IP addresses for security monitoring and audit purposes.

5.1.3 Implementation of Password Validation

The password validation system in Fig. 8 implements NIST guidelines for secure password creation, including strength indicators and validation feedback. The system balances security requirements with user experience by providing clear guidance for password creation.

```python
def validate_password_realtime(password):
    """Real-time password validation following NIST guidelines"""
    if not password:
        return False, "Password is required"

    if len(password) < 8:
        return False, "Password must be at least 8 characters"

    # Check against common passwords
    common_passwords = ["password", "123456", "12345678", "qwerty", "abc123"]
    if password.lower() in common_passwords:
        return False, "This is a commonly used password"

    # NIST-compliant strength assessment
    strength_score = 0
    feedback = []

    if len(password) >= 12:
        strength_score += 2
    elif len(password) >= 8:
        strength_score += 1

    if any(c.isupper() for c in password):
        strength_score += 1
    else:
        feedback.append("uppercase")

    if any(c.islower() for c in password):
        strength_score += 1
```

Fig. 8 *Password Validation and NIST Compliance*

5.1.4 Implementation of Session Management

The session management system in Fig. 9 tracks user sessions and implements secure logout functionality with proper cleanup of sensitive data. The system maintains session state while ensuring security through proper resource management.

```python
def log_login_activity(user_id, username, company_name, status, failure_reason=None):
    """Log user login activity for security monitoring"""
    try:
        with get_connection() as conn:
            cursor = conn.cursor()
            cursor.execute("""
                INSERT INTO login_activity
                (user_id, username, company_name, login_time, status, failure_reason)
                VALUES (?, ?, ?, ?, ?, ?)
            """, (user_id, username, company_name,
                  datetime.now().isoformat(), status, failure_reason))
            conn.commit()
    except sqlite3.Error as e:
        print(f"Error logging login activity: {e}")

def secure_logout():
    """Implement secure logout with session cleanup"""
    # Clear sensitive data from memory
    entry_password.delete(0, tk.END)
    entry_username.delete(0, tk.END)
    entry_company.delete(0, tk.END)

    # Close database connections
    if hasattr(self, 'conn') and self.conn:
        self.conn.close()
```

Fig. 9 *Session Management Implementation*

5.1.5 Implementation of Rate Limiting and Account Protection

The rate limiting system in Fig. 10 prevents brute force attacks by tracking failed login attempts and implementing temporary account lockouts. This security feature protects user accounts from unauthorized access attempts

```python
# Rate limiting configuration
login_attempts = {}
MAX_ATTEMPTS = 5
LOCKOUT_TIME = 300  # 5 minutes in seconds

def check_rate_limit(identifier):
    """Check if user has exceeded login attempt limit"""
    current_time = time.time()

    if identifier in login_attempts:
        attempts, last_attempt = login_attempts[identifier]

        # Reset attempts if lockout period has passed
        if current_time - last_attempt > LOCKOUT_TIME:
            del login_attempts[identifier]
            return True, 0

        # Check if user is locked out
        if attempts >= MAX_ATTEMPTS:
            remaining_time = LOCKOUT_TIME - (current_time - last_attempt)
            return False, int(remaining_time)

    return True, 0
```

Fig. 10 *Rate limiting and Account Protection Implementation*

5.1.6 Implementation of Admin Security Module

The admin security module in Fig. 11 provides comprehensive administrative controls with enhanced security features for user management, system monitoring, and audit trail maintenance. This module ensures secure administrative access and comprehensive system oversight

```python
def admin_login_validation(username, password):
    """Validate admin login with enhanced security"""
    try:
        conn = get_connection()
        cursor = conn.cursor()

        cursor.execute("SELECT id, password, role, status FROM users WHERE username = ?", (username,))
        user = cursor.fetchone()
        conn.close()

        if user and user[2] == 'admin' and user[3] == 'active':
            if bcrypt.checkpw(password.encode(), user[1].encode()):
                # Log successful admin login
                log_admin_activity(user[0], "ADMIN_LOGIN", "Successful admin login")
                return True, user[0]

        # Log failed admin login attempt
        log_admin_activity(user[0] if user else None, "ADMIN_LOGIN_FAILED",
                           "Failed admin login attempt")
        return False, None

    except Exception as e:
        print(f"Admin login error: {e}")
        return False, None
```

Fig. 11 *Admin Security Module Implementation*

5.2 Implementation of Email Analysis Module

The email analysis module implements the core functionality of the Email Spoofing Detection Tool Using Domain-Based Authentication, combining domain-based authentication protocols with machine learning techniques to provide accurate spoofing detection. This module handles email parsing, authentication validation, and result presentation through an intuitive user interface.

5.2.1 Implementation of Email Parsing System

The email parsing system in Fig. 12 extracts critical information from .eml files, including headers, metadata, and content required for authentication validation and machine learning analysis. The implementation handles various email formats and encoding standards to ensure compatibility.

```python
def parse_email_file(self, file_path):
    """Parse .eml file and extract headers and content"""
    try:
        with open(file_path, 'rb') as file:
            msg = email.message_from_bytes(file.read())

        # Extract essential headers
        headers = {
            'message_id': msg.get('Message-ID', ''),
            'from': msg.get('From', ''),
            'to': msg.get('To', ''),
            'subject': msg.get('Subject', ''),
            'date': msg.get('Date', ''),
            'received': msg.get_all('Received', []),
            'return_path': msg.get('Return-Path', ''),
            'reply_to': msg.get('Reply-To', ''),
            'content_type': msg.get_content_type()
        }
```

Fig. 12 Admin Security Module

5.2.2 Implementation of Domain-Based Authentication validation

The authentication validation system in Fig. 13 implements SPF, DKIM, and DMARC protocol checking through DNS lookups and record verification. This component forms the core of the rule-based detection system.

```python
def validate_spf_record(self, sender_ip, sender_domain):
    """Validate SPF record for sender domain"""
    try:
        # Query SPF record from DNS
        spf_record = dns.resolver.resolve(sender_domain, 'TXT')
        for record in spf_record:
            record_text = record.to_text().strip('"')
            if record_text.startswith('v=spf1'):
                # Parse SPF mechanisms
                if 'include:' in record_text or 'a' in record_text or 'mx' in record_text:
                    # Simplified SPF validation
                    return True
        return False
    except Exception:
        return False
```

(a)

```python
def validate_dkim_signature(self, dkim_header, email_content):
    """Validate DKIM signature"""
    try:
        if not dkim_header:
            return False
                ctrl ⇧ Fix whitespace
        # Extract domain from DKIM signature
        domain_match = re.search(r'd=([^;]+)', dkim_header)
        if domain_match:
            domain = domain_match.group(1)
            # Query DKIM public key
            selector_match = re.search(r's=([^;]+)', dkim_header)
            if selector_match:
                selector = selector_match.group(1)
```

(b)

```python
def validate_dmarc_policy(self, sender_domain):
    """Validate DMARC policy for sender domain"""
    try:
        dmarc_domain = f"_dmarc.{sender_domain}"
        dmarc_record = dns.resolver.resolve(dmarc_domain, 'TXT')
        for record in dmarc_record:
            record_text = record.to_text().strip('"')
            if record_text.startswith('v=DMARC1'):
                # Parse DMARC policy
                if 'p=reject' in record_text or 'p=quarantine' in record_text:
                    return True
        return False
    except Exception:
        return False
```

(c)

Fig. 13 *Implementation of (a) SPF Record; (b) DKIM Signature; (c) DMARC Policy*

5.2.3 Implementation of Machine Learning Integration

The machine learning in Fig. 14 component implements Random Forest algorithm for enhanced spoofing detection, combining traditional authentication results with content analysis and behavioral patterns to improve accuracy.

```python
def extract_ml_features(self, meta, email_content=""):
    """Extract machine learning features from email metadata and content"""
    features = {}

    # Authentication protocol features
    features['spf_pass'] = 1 if meta.get("spf-record", False) else 0
    features['dkim_pass'] = 1 if meta.get("dkim-record", False) else 0
    features['dmarc_pass'] = 1 if meta.get("dmarc-record", False) else 0

    # Content analysis features
    subject = meta.get("subject", "").lower()
    features['suspicious_keywords'] = self.count_suspicious_keywords(subject + " " + email_content)
    features['subject_length'] = len(subject)
    features['has_reply_to'] = 1 if meta.get("spoofed-mail", "") else 0
    features['has_ip_address'] = 1 if meta.get("ip-address", "") else 0

    # Header anomaly features
    features['header_count'] = len([k for k, v in meta.items() if v])
    features['received_hops'] = len(meta.get("received", []))

    return features
```

Fig. 14 *Implementation of Machine Learning Random Forest*

5.2.4 Implementation of Email Alert System

The email alert system in Fig. 15 provides notifications when spoofed emails are detected, featuring professional formatting, threat level classification, and multi-provider SMTP support for reliable delivery.

```python
def send_email_alert(self, meta):
    """Send professional email alert for spoofed emails"""
    if not self.validate_alert_configuration():
        return False

    try:
        # Generate unique alert ID
        alert_id = ''.join(random.choices(string.ascii_uppercase + string.digits, k=8))

        # Determine threat level based on analysis
        threat_level = self.determine_threat_level(meta)

        # Create professional email message
        msg = MIMEMultipart('alternative')
        msg["From"] = f"Email Security Alert <{self.sender_email}>"
        msg["To"] = self.receiver_email
        msg["Subject"] = f"🚨 {threat_level} THREAT: Spoofed Email Detected - Alert #{alert_id}"
        msg["X-Priority"] = "1"  # High priority

        # Generate alert content
        text_body, html_body = self.generate_alert_content(meta, alert_id, threat_level)

        # Add both text and HTML versions
        text_part = MIMEText(text_body, "plain", "utf-8")
        html_part = MIMEText(html_body, "html", "utf-8")
```

Fig. 15 *Implementation of Email Alert System*

5.2.5 Implementation of Graphical Report Generation

The graphical report generation system in Fig. 16 creates visual analytics and charts for email security analysis trends, providing insights through professional data visualization.

```python
def generate_graphical_reports(self):
    """Generate comprehensive graphical reports with professional styling"""
    try:
        # Fetch data for visualization
        data = self.fetch_visualization_data()

        if not data:
            messagebox.showwarning("No Data", "No analysis data available for visualization")
            return

        # Create professional charts
        self.create_status_distribution_chart(data)
        self.create_protocol_analysis_chart(data)
        self.create_trend_analysis_chart(data)
```

Fig. 16 *Implementation of Graphical Report Generation*

5.2.6 User Acceptance Testing Form Results

User acceptance testing validates the system's usability, functionality, and effectiveness from the end-user perspective. Testing involved IT administrators and security professionals who represent the target user base.

5.2.7 User Interface

The forms in Fig. 17, Fig. 18, Fig. 19, and Fig. 20 utilize a scaling system ranging from one to five, where a rating one indicates a poor user interface, while rating of five signifies an excellent user interface design

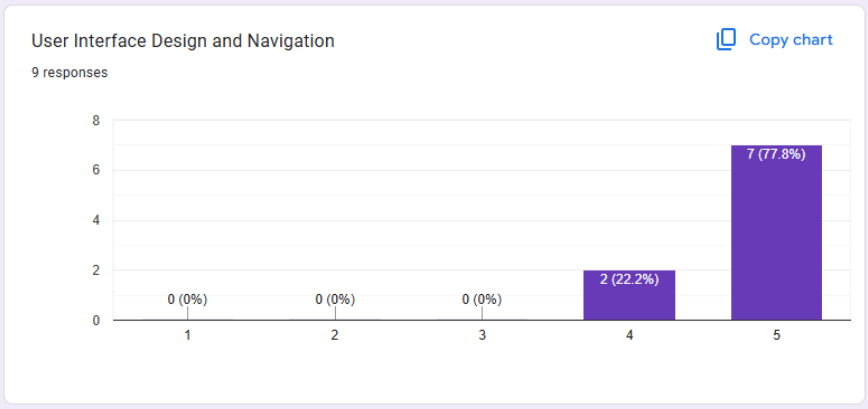


Fig. 17 *User Interface Design and Navigation*

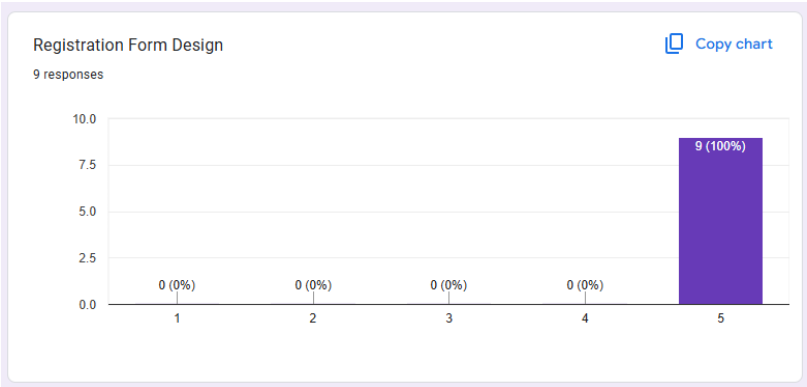


Fig. 18 *Registration Form Design*

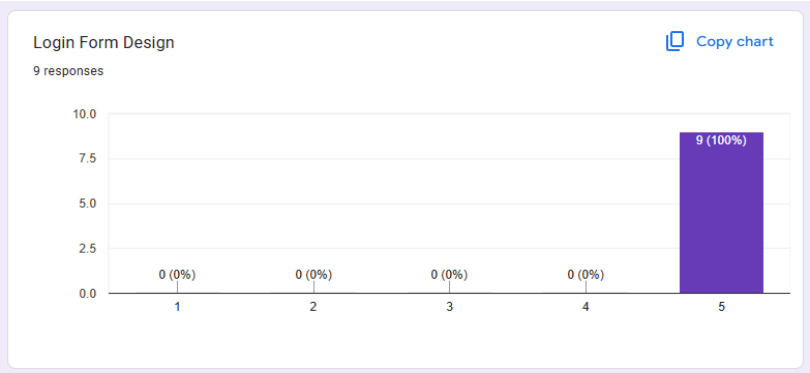


Fig. 19 *Login Form Design*

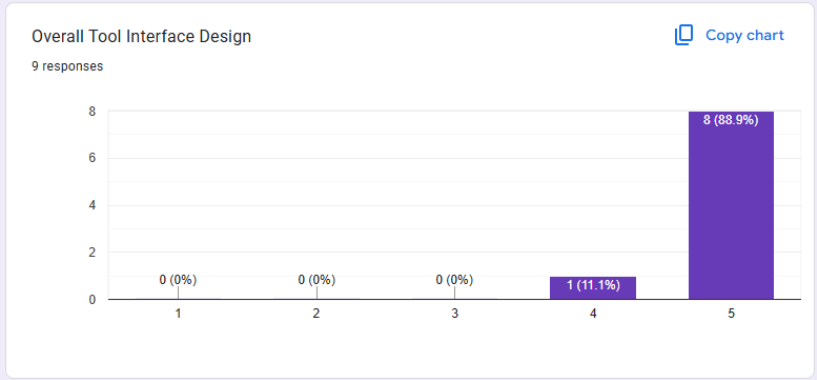


Fig. 20 *Overall Tool Interface Design*

5.2.8 User Experience

The forms in Fig. 21, Fig. 22, Fig. 23 and Fig. 24 consist of Yes or No questions. There are four questions including the user experience of SpoofSentry.

Fig. 21 *Login and Registration UX*

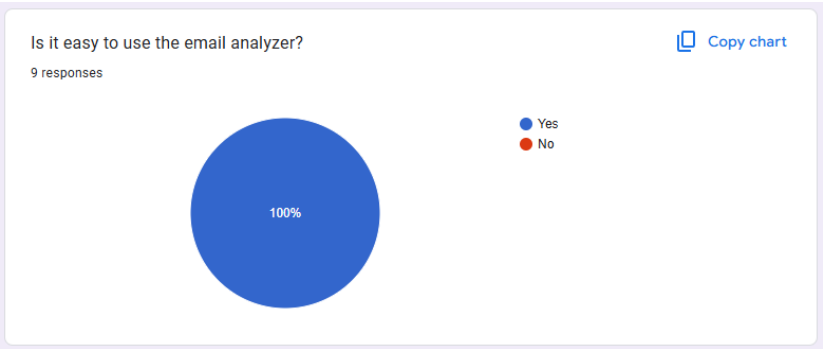


Fig. 22 *Email Analyzer UX*

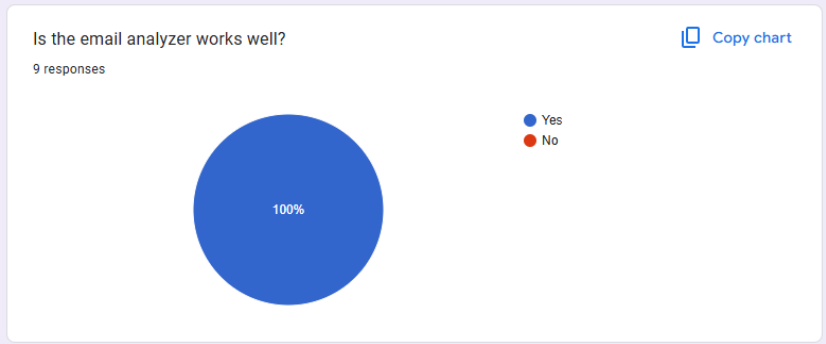


Fig. 23 *Email Analyzer Functionality*

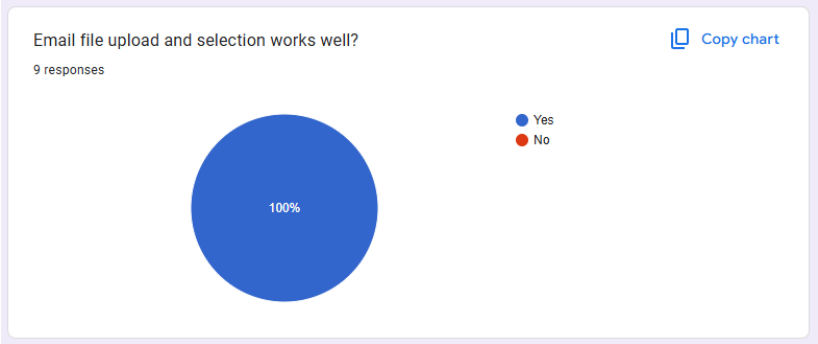


Fig. 24 *Email File Upload and Selection UX*

6. Conclusion

This project successfully developed SpoofSentry, a user-friendly email spoofing detection tool that combines SPF, DKIM, and DMARC protocols with Random Forest machine learning for enhanced threat detection. The tool demonstrated effectiveness in identifying spoofed emails through hybrid analysis, featuring automated threat classification, email alerts, bulk processing capabilities, and comprehensive PDF reporting to improve email security practices for organizations.

The findings show that integrating traditional authentication protocols with machine learning significantly improves detection accuracy while maintaining user accessibility for non-technical administrators. The tool successfully reduces manual interpretation requirements and provides automated whitelist/blacklist management, demonstrating the potential for accessible cybersecurity solutions. Future enhancements could include expanding machine learning features, supporting additional email formats, and developing cloud-based deployment options to serve larger organizational environments and diverse cybersecurity needs.

Acknowledgement

The authors would like to thank the Faculty of Computer Science and Information Technology, Universiti Tun Hussein Onn Malaysia for its support.

Conflict of Interest

Authors declare that there is no conflict of interests regarding the publication of the paper.

Author Contribution

This journal requires that all authors take public responsibility for the content of the work submitted for review. The contributions of all authors must be described in the following manner:

The authors confirm contribution to the paper as follows: ***study conception and design:*** *N. F. N. Mohd Nazlan, S. F. Md Salleh;* ***data collection:*** *N. F. N. Mohd Nazlan, S. F. Md Salleh;* ***analysis and interpretation of results:*** *N. F. N. Mohd Nazlan, S. F. Md Salleh;* ***draft manuscript preparation:*** *N. F. N. Mohd Nazlan, S. F. Md Salleh. All authors reviewed the results and approved the final version of the manuscript.*

References

[1] "Email spoofing - Wikipedia." Accessed: Jun. 12, 2025. [Online]. Available: https://en.wikipedia.org/wiki/Email_spoofing?utm_source=chatgpt.com

[2] D. Powell, "Business loses $300,000 to 'spoofed' email scam: How to protect yourself from being impersonated," *Smart Company*, Nov. 2018, Accessed: Jun. 12, 2025. [Online]. Available: https://www.smartcompany.com.au/technology/business-email-compromise-scam/

[3] M. Tariq Banday, "Effectiveness and Limitations of E-Mail Security Protocols," *International Journal of Distributed and Parallel systems*, vol. 2, no. 3, pp. 38–49, May 2011, doi: 10.5121/IJDPS.2011.2304.

[4] "Why is an Email Spoofing Tool Important? How to Detect and Prevent? | Trustifi | LOGON Software Asia." Accessed: Jun. 12, 2025. [Online]. Available: https://logon-int.com/blog/why-is-an-email-spoofing-tool-important-how-to-detect-and-prevent-trustifi/

[5] "Email Authentication Protocols – SPF, DKIM & DMARC – Networks at ITP." Accessed: Jun. 12, 2025. [Online]. Available: https://itp.nyu.edu/networks/explanations/email-authentication-protocols-spf-dkim-dmarc/

[6] A. Herzberg, "DNS-based email sender authentication mechanisms: A critical review," *Comput Secur*, vol. 28, no. 8, pp. 731–742, Nov. 2009, doi: 10.1016/J.COSE.2009.05.002.

[7] S. Görling, "An overview of the Sender Policy Framework (SPF) as an anti-phishing mechanism," *Internet Research*, vol. 17, no. 2, pp. 169–179, 2007, doi: 10.1108/10662240710737022.

[8] "(PDF) DomainKeys Identified Mail (DKIM): Using Digital Signatures for Domain Verification." Accessed: Jun. 12, 2025. [Online]. Available: https://www.researchgate.net/publication/221650803_DomainKeys_Identified_Mail_DKIM_Using_Digital_Signatures_for_Domain_Verification

[9] O. O. Rode, "Email spam protection technology based on DMARC," *Modern Information Security*, vol. 3, no. 51, 2022, doi: 10.31673/2409-7292.2022.033238.

[10] A. N. S. Kinasih, A. N. Handayani, J. T. Ardiansah, and N. S. Damanhuri, "Comparative analysis of decision tree and random forest classifiers for structured data classification in machine learning," *Science in Information Technology Letters*, vol. 5, no. 2, pp. 13–24, Nov. 2024, doi: 10.31763/SITECH.V5I2.1746.

[11] "Threat Intelligence July December 2024 | Mimecast." Accessed: Jun. 12, 2025. [Online]. Available: https://www.mimecast.com/resources/ebooks/threat-intelligence-july-december-2024/?utm_source=google&utm_medium=ppc&utm_campaign=asean-hrm-growth-brand/

[12] "Email Security Service: Threat Protection Solutions | Proofpoint US." Accessed: Jun. 12, 2025. [Online]. Available: https://www.proofpoint.com/us/products/threat-defense

[13] "Generate a DMARC record for your domain | DMARC Analyzer." Accessed: Jun. 12, 2025. [Online]. Available: https://emailauth.io/dmarc-generator

[14] P. Tavolato and K. Vincena, "PROTOTYPING METHODOLOGY AND ITS TOOL.," pp. 434–446, 1984, doi: 10.1007/978-3-642-69796-8_38.

Appendix A: Use Case Diagram

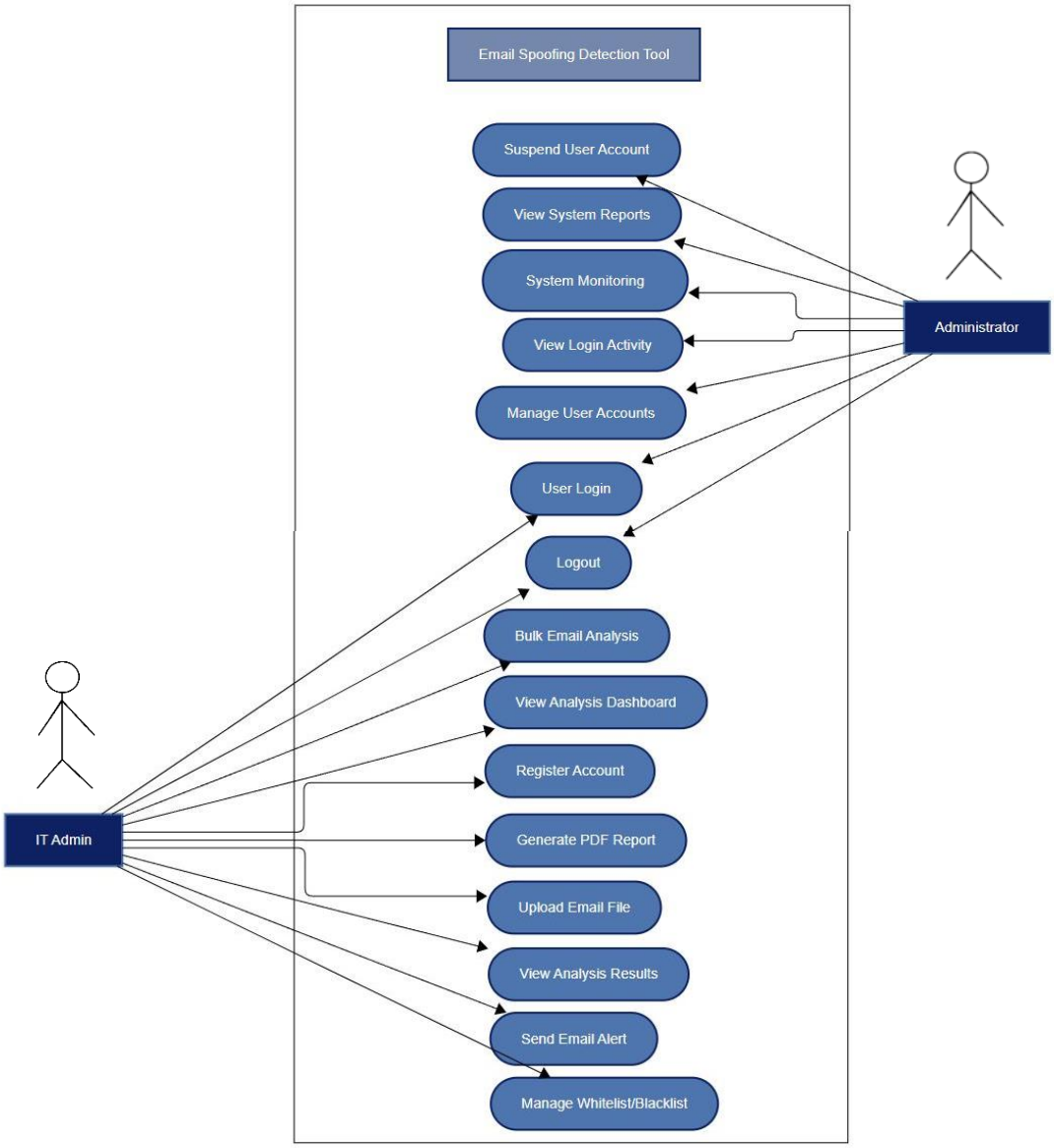


Fig. 3 *Use Case Diagram for SpoofSentry*

Appendix B: Class Diagram

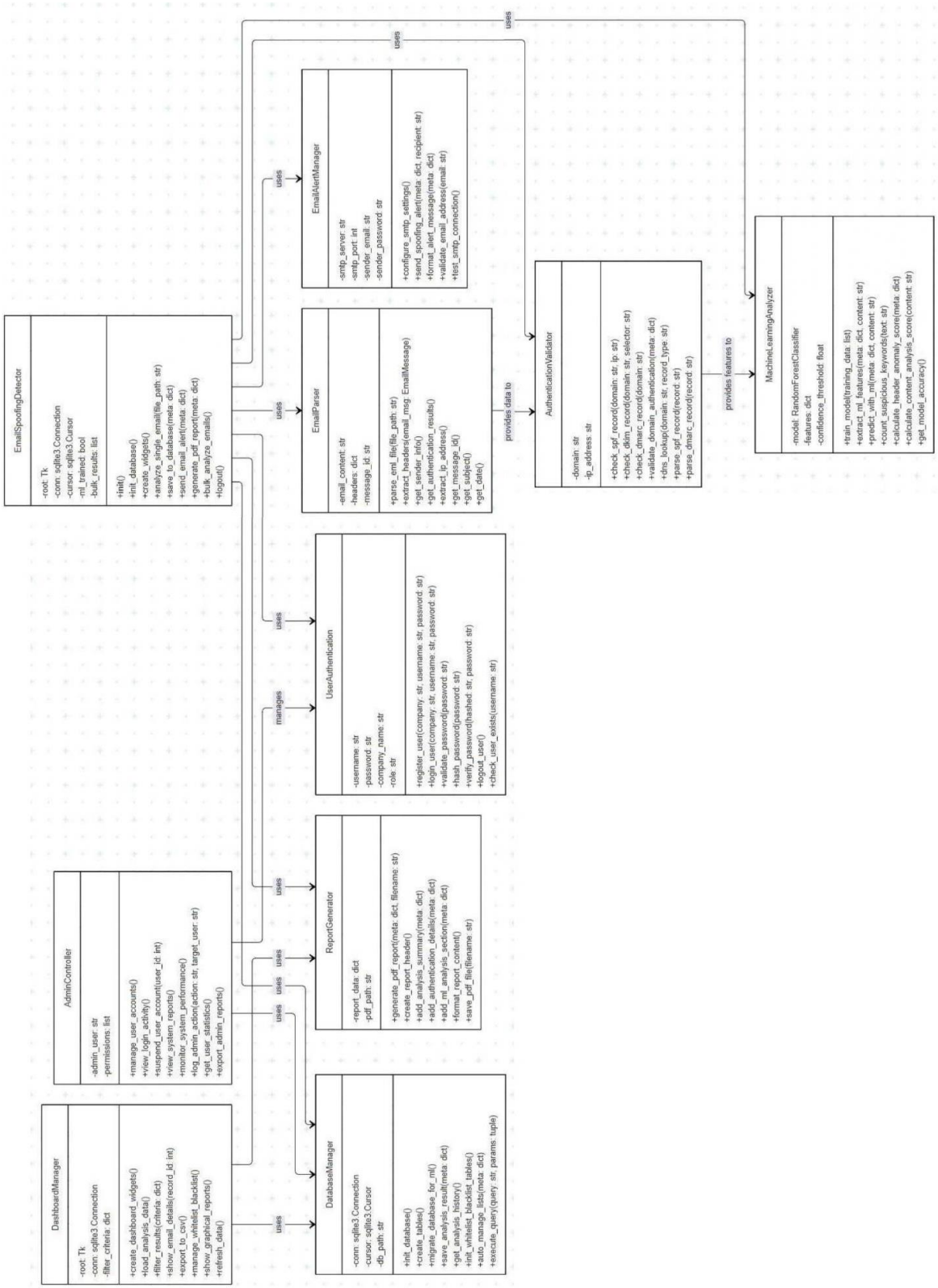


Fig. 4 *Class Diagram for SpoofSentry*

Appendix C: Flowchart

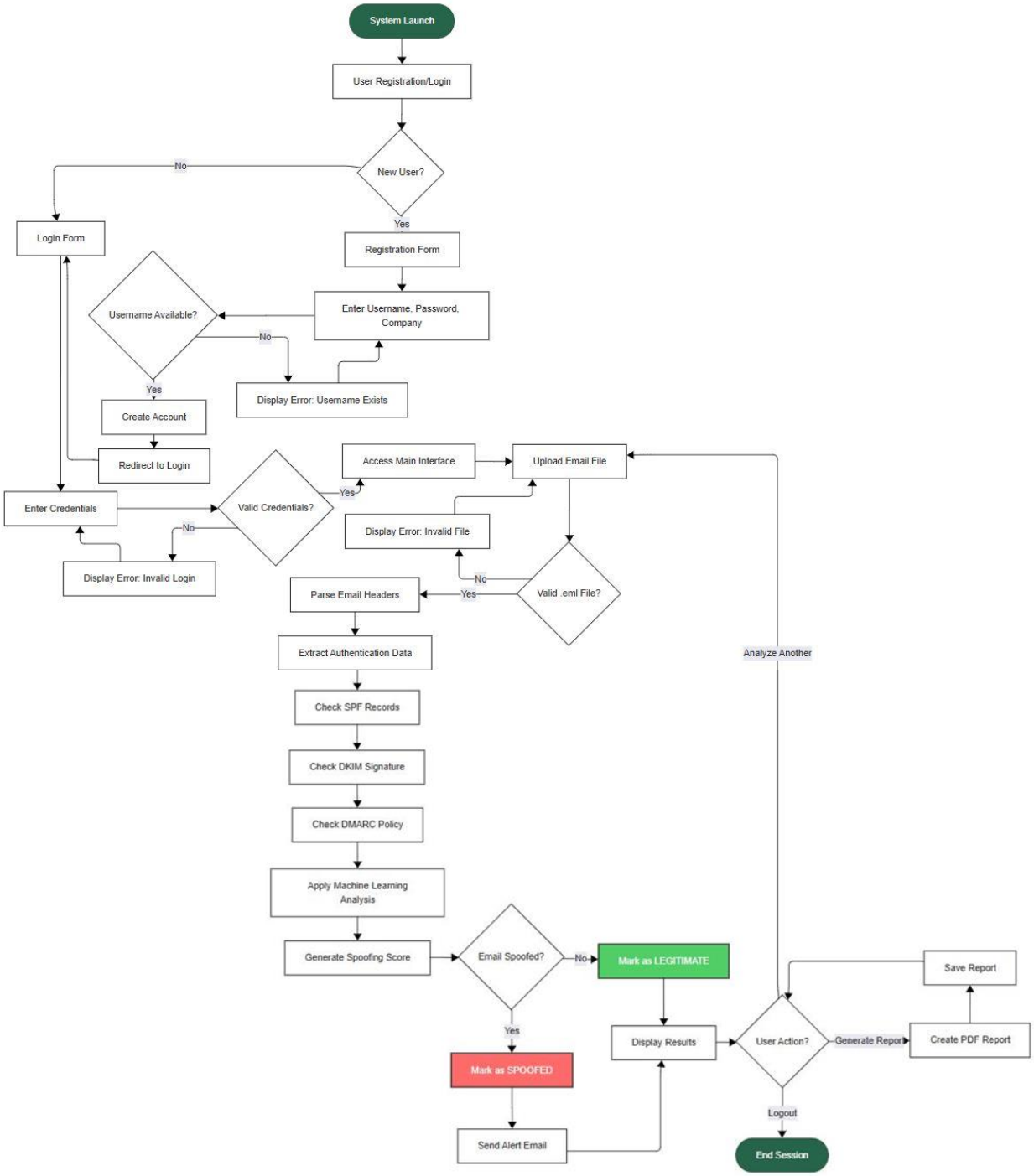


Fig. 5 *Flow Chart Diagram for SpoofSentry*