

***CHAPTER 1***  
**INTRODUCTION**

---

## CHAPTER 1

### INTRODUCTION

Phishing attacks have emerged as one of the most significant cybersecurity threats, targeting individuals and organizations through deceptive emails, malicious websites, and compromised files. Cybercriminals use social engineering techniques to trick users into divulging sensitive information such as login credentials, financial details, and personal data. Traditional security measures, such as blacklist-based filtering and rule-based detection, often fail to detect sophisticated phishing attacks that continuously evolve in structure and execution. As a result, there is a growing need for advanced phishing detection systems that leverage artificial intelligence (AI) and machine learning to enhance threat identification and mitigation.

An AI-powered phishing detection system can effectively analyze websites, files, and emails in real time to detect phishing attempts with high accuracy. By utilizing machine learning algorithms, natural language processing (NLP), and deep learning models, the system can identify suspicious patterns in URLs, email content, and file metadata. Unlike conventional detection methods, AI-based approaches can adapt to new attack strategies by continuously learning from new threats. Furthermore, heuristic analysis and AI-driven anomaly detection help reduce false positives and improve the reliability of phishing identification. Integrating such a system into digital communication platforms can significantly enhance cybersecurity defenses and prevent unauthorized access to sensitive information.



**Fig 1.1 – Difference in storage system of Centralized and Decentralized Applications.**

The architecture of PDS-HML is designed to mimic real-world detection scenarios. The system begins by crawling the HTML content of suspicious websites and extracting relevant features from the URL and its DOM structure. It then applies both heuristic rules and machine learning classifiers to determine whether a URL is likely to be malicious. Importantly, the system is capable of adapting to emerging phishing techniques by retraining on newly discovered patterns and evolving datasets. This dynamic nature gives PDS-HML an edge over static detection mechanisms.

Experimental results have demonstrated that the hybrid model achieves superior performance in terms of detection accuracy, precision, recall, and false positive rates compared to traditional models. The resilience of PDS-HML was further validated against a variety of phishing tactics, including evasion and spoofing methods. By effectively identifying phishing websites—even those designed to bypass conventional filters—the system significantly enhances online safety and reduces the risk of data breaches.

In summary, the growing complexity and frequency of phishing attacks necessitate the development of more sophisticated detection systems. The hybrid machine learning-based approach presented in this report offers a scalable and adaptive solution to the phishing threat. Through the integration of multiple classifiers and feature extraction techniques, the PDS-HML framework represents a substantial step forward in securing digital ecosystems from malicious URL-based attacks.

## 1.1 CENTRALIZED APPLICATIONS

### 1.1.2. Advantages of Centralized Applications

- **High Accuracy through Unified Learning:** By collecting data from multiple sources into a central repository, the system gains access to richer and more varied datasets, resulting in improved training and higher accuracy of hybrid machine learning models.
- **Reduced Redundancy and Overhead:** Centralized systems eliminate the need to deploy and maintain separate detection models on individual client machines, leading to reduced maintenance complexity and consistent performance.

- **Faster Model Updates and Patching:** Any improvements or security patches to detection algorithms can be instantly deployed across the entire infrastructure from a central server, ensuring up-to-date protection.
- **Enhanced Threat Intelligence:** Aggregated threat data allows the system to identify trends and emerging phishing tactics more effectively, enabling proactive defense mechanisms and timely alerts.
- **Improved Incident Response Time:** Central control enables faster decision-making and automated responses (e.g., URL blocking, warning pop-ups), minimizing user exposure to phishing attempts.

### 1.1.3 Disadvantages of phishing detection

- **High False Positive Rate:** Legitimate websites may be incorrectly flagged as phishing, leading to user frustration and reduced trust in the detection system.
- **Evasion Techniques by Attackers:** Phishers often use advanced obfuscation methods, dynamic URLs, or mimic legitimate structures, which can bypass traditional or static detection systems.
- **Dependence on Feature Quality:** The accuracy of machine learning-based systems depends heavily on the quality and relevance of extracted features (e.g., from URL, HTML). Poor feature engineering results in weak detection.
- **Model Drift:** Phishing techniques evolve constantly. A model trained on old data may become ineffective over time without regular updates or retraining.
- **Resource Intensive:** Some detection systems, especially those using deep learning, require high computational resources and time for training and inference.

## 1.2 Security and Privacy Concerns in Phishing Detection Systems

Phishing detection systems, while crucial for protecting users from fraudulent websites, also introduce several security and privacy challenges. These systems, especially those relying on centralized architectures or data aggregation models, must address key risks to maintain user trust and effectiveness.

- **Data Breaches:** Phishing detection platforms often collect sensitive user data, such as browsing history, URL patterns, and behavioural metadata. If this data is stored centrally, it becomes a lucrative target for cybercriminals. A successful breach could expose sensitive personal information or lead to secondary attacks. To mitigate this, systems must enforce strong encryption, secure API protocols, and multi-layered authentication measures. Regular vulnerability assessments and penetration tests are essential to maintaining data integrity.
- **Misuse of Collected Data:** Like other centralized systems, phishing detection tools might collect data for training machine learning models. However, without strict policies, this data can be misused—for profiling, unauthorized marketing, or even surveillance. To prevent misuse, organizations must adopt clear data governance policies, allow users to opt-out of data collection, and ensure transparency in how data is used and stored.
- **Lack of Transparency:** Some detection systems operate as black-box models, especially in commercial tools that do not disclose how URLs are flagged. This lack of transparency makes it difficult for users or administrators to validate decisions or understand the reasoning behind false positives. Open models, explainable AI techniques, and documented rule sets can improve trust and accountability.
- **Censorship and Over blocking:** Phishing detection engines can mistakenly block legitimate websites (false positives), disrupting access to useful resources. In severe cases, this might be perceived as content censorship. To prevent misuse, detection systems must include appeal mechanisms or human-in-the-loop verification for flagged websites.
- **Single Point of Failure:** Centralized detection systems are at risk of outages or server-side compromises. If the detection engine is unavailable, users might be left unprotected, or the system might fail to classify new threats. Redundancy through distributed architecture, load balancing, and edge computing solutions can reduce this risk.

## 1.3 Objectives

➤ **To Develop an AI-Powered Phishing Detection System for Websites, Files, and Emails**

Justification: Phishing attacks occur across multiple digital platforms, including websites, emails, and file attachments. A comprehensive AI-based detection system will provide an integrated security solution that identifies phishing threats across these different channels, enhancing cybersecurity measures.

➤ **To Utilize Machine Learning and Deep Learning Algorithms for Improved Accuracy**

Justification: Traditional rule-based and signature-based detection methods often fail against evolving phishing techniques. Implementing machine learning (ML) and deep learning (DL) models allows the system to adapt and improve its phishing detection capabilities by learning from new attack patterns.

➤ **To Incorporate Natural Language Processing (NLP) for Email and Website Content Analysis**

Justification: Phishing emails and websites often use deceptive language to manipulate users. By integrating NLP techniques, the system can analyze textual content for suspicious patterns, such as urgent language, impersonation tactics, and keyword-based deception.

➤ **To Implement a URL-Based Phishing Detection Mechanism**

Justification: Many phishing attacks are conducted through malicious URLs that redirect users to fraudulent websites. AI-powered URL analysis can detect anomalies, domain spoofing, and suspicious redirections, reducing the risk of phishing attempts.

➤ **To Provide an Automated Response Mechanism for Phishing Threats**

Justification: Detecting phishing attempts alone is not enough; an automated response system (such as email quarantining, warning messages, or URL blocking) can prevent users from engaging with malicious content and reduce the chances of data compromise.

➤ **To Develop a User-Friendly Dashboard for Monitoring and Reporting Phishing Attempts**

Justification: A clear and accessible dashboard will enable security teams, IT administrators, and individual users to monitor phishing threats in real time, view reports, and take necessary actions to mitigate risks effectively.

## 1.4 Motivation

In today's highly interconnected digital world, phishing remains one of the most widespread and damaging cyber threats, exploiting human trust to compromise sensitive data, financial resources, and critical infrastructure. Traditional rule-based and signature-based detection systems often fail to keep up with the rapidly evolving tactics of cybercriminals, especially when dealing with sophisticated phishing attacks hidden within websites, documents, and emails.

Given the rising number of phishing incidents targeting individuals, enterprises, and even governments, there is an urgent need for an intelligent, adaptive, and comprehensive defense mechanism. An AI-powered phishing detection system, capable of analyzing websites, files, and emails in real-time, offers a proactive approach by learning patterns, identifying subtle anomalies, and detecting zero-day phishing attacks with greater accuracy.

By leveraging machine learning, natural language processing, and computer vision techniques, the proposed system can automatically detect malicious intent across multiple mediums, significantly reducing the risk of data breaches, identity theft, and financial loss. Such a solution not only strengthens cybersecurity resilience but also instills greater confidence among users and organizations in their digital operations.

## 1.5 Problem Definition

Phishing attacks have emerged as one of the most prevalent cybersecurity threats, targeting individuals and organizations through deceptive websites, malicious email campaigns, and fraudulent file attachments. Cybercriminals employ sophisticated techniques such as domain spoofing, social engineering, and AI-generated phishing emails to steal sensitive information, including login credentials, financial data, and personal details. Traditional phishing detection methods, which rely on rule-based or signature-based techniques, struggle to keep up with the constantly evolving nature of phishing attacks, leading to increased risks of data breaches, financial loss, and reputational damage.

To address these challenges, an AI-powered phishing detection system is required to analyze websites, emails, and files using advanced machine learning, deep learning, and natural language processing (NLP) techniques. The system must be capable of detecting phishing attempts in real time while minimizing false positives and adapting to emerging phishing tactics. Additionally, it should provide

automated threat response mechanisms, such as blocking malicious URLs, alerting users, and quarantining suspicious emails or files. The development of an intelligent, scalable, and adaptive phishing detection system will significantly enhance cybersecurity defenses and mitigate the risks posed by phishing attacks.

## **1.6 Overview of the present work**

The present work focuses on the design and implementation of a robust phishing detection system using hybrid machine learning techniques, referred to as Phishing Detection System - Hybrid Machine Learning (PDS-HML). The system aims to accurately classify URLs as phishing or legitimate by leveraging the combined strengths of multiple machine learning and deep learning models.

Traditional phishing detection techniques, which rely solely on blacklists, rule-based engines, or singular machine learning classifiers, often struggle to adapt to dynamic and evolving phishing strategies. These limitations are primarily due to their inability to generalize to newly crafted URLs and obfuscated attack vectors.

To overcome these challenges, this project proposes a hybrid architecture that integrates both conventional machine learning algorithms (such as Support Vector Machines, Random Forest, and Decision Trees) and deep learning models (such as LSTM and CNN). The system uses feature extraction techniques from URLs and the Document Object Model (DOM) of HTML pages, enabling it to identify subtle patterns that are indicative of phishing.

Key components of this work include:

- Feature extraction from both static URL characteristics (length, domain, presence of symbols) and HTML-based features (link counts, script tags, etc.).
- Ensemble learning techniques for combining classifiers and improving detection robustness.
- Evaluation of model performance using standard metrics like accuracy, precision, recall, and F1-score on publicly available phishing datasets.

The hybrid approach not only enhances detection accuracy but also reduces false positives and adapts better to unseen data. The proposed PDS-HML system contributes to improving cybersecurity measures by offering a scalable and intelligent solution against phishing threats in real-time environments.



## 1.7 Background

The rise of the Internet has significantly transformed how individuals communicate, conduct business, and access information. While this digital transformation has brought about unprecedented convenience and efficiency, it has also introduced a wide range of security threats. One of the most pervasive and damaging among them is phishing—a cyber-attack that deceives users into revealing confidential information such as login credentials, financial details, or personal data.

Phishing attacks typically use fake or deceptive websites that closely mimic legitimate ones. These websites are spread via email, social media, or malicious advertisements, tricking users into thinking they are interacting with a trustworthy source. Despite growing awareness, phishing continues to be a major cause of security breaches worldwide.

To address this issue, researchers have employed several techniques including heuristic methods, blacklist-based detection, and machine learning (ML) models. Blacklist-based systems, while fast, cannot identify newly launched phishing websites. Heuristic and rule-based systems are more flexible but often lack precision. Machine learning-based methods, trained on large datasets of URLs and website features, have shown better adaptability in recognizing malicious patterns.

However, standalone ML techniques have limitations in generalizing across evolving attack patterns. Recent advancements have seen the integration of deep learning models and ensemble techniques into phishing detection, allowing systems to identify complex patterns and subtle indicators more effectively.

## 1.8 Organization of Report

This report is organized as 6 chapters namely, Introduction, Analysis, Design, Implementation, Testing and lastly Conclusion and Future Enhancements.

**Chapter 1:** Describes about Introduction which includes Overview, Problem Statement, Objectives, Limitations and the Literature Survey.

**Chapter 2:** Describe about the summary of the prior works, outcome of the review, problem identified from the review and details about the proposed work.

**Chapter 3:** Describes about System Requirements that contains the functional and non-functional requirements of the system, along with the details of specific hardware and software that has been utilized.

**Chapter 4:** Describes about the System Design that explains the architecture of the system along with the control flow between the integrated components.

**Chapter 5:** Describes the Implementation which consists of details of specific modules of the system and how they are integrated to form the system as a whole.

**Chapter 6:** Describes about Testing that consists of unit testing of specific sensors, integration testing and system testing of the overall system to check if the implementation has been done successfully or not.

**Chapter 7:** Consists of the Conclusion and Future Enhancements which describes about the further changes that can be made to the system.

***CHAPTER 2***  
**LITERATURE REVIEW**

## CHAPTER 2

# LITERATURE REVIEW

### 2.1 Summary of Prior Works

A literature survey or a literature review in a project report shows the various analyses and research made in the field of interest and the results already published, taking into account the various parameters of the project and the extent of the project. Literature survey is mainly carried out in order to analyze the background of the current project which helps to find out flaws in the existing system & guides on which unsolved problems we can work out. So, the following topics not only illustrate the background of the project but also uncover the problems and flaws which motivated to propose solutions and work on this project.

A literature survey is a text of a scholarly paper, which includes the current knowledge including substantive findings, as well as theoretical and methodological contributions to a particular topic. Literature reviews use secondary sources, and do not report new or original experimental work. Most often associated with academic-oriented literature, such as a thesis, dissertation or a peer-reviewed journal article, a literature review usually precedes the methodology and results sectional though this is not always the case. Literature reviews are also common in are search proposal or prospectus (the document that is approved before a student formally begins a dissertation or thesis). Its main goals are to situate the current study within the body of literature and to provide context for the particular reader. Literature reviews are a basis for researching nearly every academic field. demic field. A literature survey includes the following:

**Title: AI-Driven Phishing Detection Systems**

**Authors:** Obaloluwa Ogundairo

**Abstract:** This paper explores the application of Artificial Intelligence (AI) in enhancing phishing detection systems. AI-driven approaches leverage machine learning algorithms, natural language processing, and pattern recognition to identify and mitigate phishing threats with greater accuracy and efficiency.

**Methodologies Used:** The study discusses various AI methodologies employed in phishing detection, including supervised and unsupervised learning techniques, ensemble methods, and deep learning models.

**Advantages:** AI-driven systems can detect subtle patterns and anomalies indicative of phishing attempts that might elude conventional methods.

**Limitations:** The paper concludes with an overview of current challenges and future directions for research in this domain, emphasizing the need for continuous advancement to address the dynamic nature of phishing threats.

## **2. Title: AI-Powered Phishing Detection and Prevention**

**Authors:** OA Lamina

**Abstract:** This paper examines the role of artificial intelligence (AI) in enhancing phishing detection systems. AI-driven approaches utilize machine learning algorithms, natural language processing, and pattern recognition to identify and mitigate phishing threats with improved accuracy and efficiency.

**Methodologies Used:** The study discusses various AI methodologies in phishing detection, including supervised and unsupervised learning techniques, ensemble methods, and deep learning models.

**Advantages:** By analyzing large datasets, these systems can uncover subtle patterns and anomalies indicative of phishing attempts that conventional methods might miss.

**Limitations:** The paper concludes with a discussion of current challenges and future research directions in this field, highlighting the necessity for ongoing advancements to tackle the dynamic nature of phishing threats.

## **3. Title: AI-Driven Phishing Detection: Combating Cyber Threats Through Advanced Machine Learning**

**Authors:** Eric Blancaflor, Lianne Francheska Deldacan, Shane Hunat, Bea Marga Rivera, Enrique Karl

**Abstract:** This research examines user awareness, cybercrime knowledge, email usage patterns, and the effectiveness of AI-driven phishing detection systems.

**Methodologies Used:** The study employs experimental designs and cross-sectional surveys to gather data on user behavior and system performance.

**Advantages:** Provides insights into user interactions with phishing attempts and the effectiveness of AI-driven detection systems.

**Limitations:** The study's findings are based on self-reported data, which may be subject to biases.

#### **4. Title: AI/ML-Powered Phishing Detection: Building an Impenetrable Email Security System**

**Authors:** Chandrakanth Madhavaram, Siddharth Konkimalla, Shravan Kumar Rajaram, Hemanth Kumar Gollangi

**Abstract:** This paper discusses the capabilities of AI/ML-powered systems in training on sampled volumes of emails to improve the recognition of phishing and non-phishing emails.

**Methodologies Used:** The study focuses on the training processes of AI/ML systems using sampled email data.

**Advantages:** Highlights the potential of AI/ML systems to enhance email security through improved recognition of phishing attempts.

**Limitations:** The paper lacks detailed information on specific algorithms or models used.

#### **5. Title: AI-Based Phishing Detection and Automated Response**

**Authors:** Sarika Nitin Zaware, Sulochana Sagar Madachane, Satish Gujar, Pankaj Chandre, Bhagyashree Shendkar

**Abstract:** This research offers a multi-channel security architecture for automated phishing response and detection that uses cutting-edge artificial intelligence (AI) technology to counteract this ubiquitous threat.

**Methodologies Used:** The study proposes a multi-channel security architecture leveraging AI for automated phishing detection and response.

**Advantages:** Addresses the need for automated and adaptive phishing detection across multiple communication channels.

**Limitations:** The paper provides limited details on the implementation and evaluation of the proposed architecture.

#### **6. Title: Artificial Intelligence in Cybersecurity to Detect Phishing**

**Authors:** Dominique Wasso Kiseki, Vincent Havyarimana, Désiré Lumonge Zabagunda, Walumbuka Ilundu Wail, Therence Niyonsaba

**Abstract:** The study analyzes three supervised machine learning classifiers and one deep learning classifier for detecting and predicting phishing websites.

**Methodologies Used:** The classifiers analyzed include Naive Bayes, Decision Tree, Gradient Boosting, and Multi-Layer Perceptron.

**Advantages:** The Gradient Boosting Classifier performed best, with a precision of 96.2% and an F1-score of 96.6%.

**Limitations:** The study focuses solely on website phishing detection and may not generalize to other forms of phishing.

## **7. Title: Staying Ahead of Phishers: A Review of Recent Advances and Challenges in Phishing Detection**

**Authors:** Kavya Shanmugam, D. Sumathi

**Abstract:** This paper presents a comprehensive review of state-of-the-art methodologies in phishing detection by analyzing a diverse range of techniques.

**Methodologies Used:** The study reviews various phishing detection techniques, including machine learning and AI-based approaches.

**Advantages:** Provides a broad overview of current methodologies and identifies trends in phishing detection research.

**Limitations:** As a review paper, it does not present new experimental data or original research findings.

## **2.2 Outcome of the review**

The literature review highlights the growing importance of Artificial Intelligence (AI) and Machine Learning (ML) in phishing detection systems. Across the reviewed works, there is a clear trend of transitioning from traditional rule-based approaches to more adaptive, intelligent detection models. Most of the studies adopt supervised and unsupervised learning techniques, ensemble

models, and deep learning architectures like neural networks, LSTMs, and Multi-Layer Perceptrons.

AI-driven detection systems show a significant advantage in terms of accuracy, adaptability, and automation. For example, studies reveal that ensemble models and Gradient Boosting techniques outperform simpler classifiers by achieving higher precision and F1-scores, with some models reaching accuracy levels above 96%. Another strong point from the review is the ability of AI to handle multi-channel threats and real-time phishing attacks through automated response mechanisms.

However, the review also brings attention to some common limitations. Many papers lack implementation details or do not evaluate their models in real-world environments. Some rely on small or outdated datasets, and others fail to generalize their findings across different phishing mediums (e.g., websites, emails, social platforms). Additionally, the complexity of AI models may limit their scalability or interpretability in enterprise applications.

Overall, the outcome of the review reinforces the relevance of this project. It validates the use of a hybrid machine learning approach that combines the strengths of traditional and deep learning models for phishing detection. The need for high accuracy, low false positives, and real-time adaptability makes the hybrid system a suitable and timely solution to address existing gaps in phishing detection research.

Furthermore, several studies emphasize the importance of user behavior analysis and contextual understanding in phishing detection. Research involving surveys and experimental setups shows that human factors play a critical role in the success or failure of phishing attempts. While technical models are vital for flagging malicious activity, integrating behavioral insights can significantly improve system responsiveness and user trust. This encourages the development of phishing detection systems that not only rely on static features but also learn dynamically from user interactions, browsing habits, and response patterns.

In addition, the review illustrates a growing demand for real-time, automated phishing defense mechanisms. AI-based phishing detection systems are increasingly being designed to operate in live environments, offering instant threat identification and mitigation. Some reviewed works propose multi-channel defense systems that protect users across emails, websites, and messaging platforms. However, challenges such as latency, data privacy, and scalability remain. These insights directly inform the present work, which aims to build a robust hybrid model capable of processing URLs efficiently while maintaining high accuracy and adaptability in real-time usage



## 2.3 Proposed work

The proposed work aims to develop a robust phishing detection system based on hybrid machine learning techniques that analyze URL structures and related metadata to classify whether a website is legitimate or malicious. The system is designed to overcome the limitations of traditional blacklisting and single-model machine learning approaches, which often struggle with zero-day phishing attacks and high false positive rates.

The phishing detection system will feature an intuitive and user-friendly interface, where users or security analysts can input URLs for real-time evaluation. The core functionality will rely on feature extraction from the URL—such as length, presence of special characters, domain age, number of subdomains, HTTPS usage, and entropy calculations—along with metadata like domain registration details and redirection behavior. These features will feed into a hybrid model composed of classical ML

classifiers (e.g., Random Forest, SVM) and deep learning architectures (e.g., LSTM or CNN), which together improve accuracy and robustness.

One of the key aspects of the system is its hybrid classification engine. Traditional machine learning models are effective in handling structured features, while deep learning models are adept at detecting patterns in raw URL text sequences. By combining the outputs of these models using ensemble or stacking techniques, the system will achieve improved precision, recall, and overall F1-score. Additionally, the detection engine can be trained to adapt over time through continuous learning from user feedback and newly reported phishing URLs.

The platform will also feature an explainable AI module that shows which features contributed most to the phishing classification decision. This ensures transparency and builds user trust. Furthermore, the system can be integrated into web browsers or enterprise security solutions to automatically flag and block suspicious URLs in real-time.

In conclusion, this proposed phishing detection system leverages hybrid machine learning techniques to offer a scalable, adaptive, and accurate solution for combating phishing attacks. By combining URL-based feature extraction with multi-model classification, the system not only improves detection effectiveness but also ensures resilience to evolving phishing tactics. Its integration into real-world applications can significantly enhance the cybersecurity posture of individual users and organizations alike.

***CHAPTER 3***  
**SYSTEM REQUIREMENTS**

## CHAPTER 3

# SYSTEM REQUIREMENTS

### 3.1 Functional Requirements

The functional requirements define the core features and operations that the URL-based phishing detection system should support. The system aims to analyze and classify URLs as either legitimate or phishing by applying machine learning techniques. These requirements ensure that the system is effective, user-friendly, and secure. This section, will outline the key functional requirements for building a URL-based phishing detection system using Hybrid Machine Learning.

- **User Authentication:** The application should allow users to create accounts and log in securely using a username-password mechanism or multi-factor authentication. The authentication process should ensure data privacy and restrict access to authorized users only.
- **URL Prediction Module:** Users should be able to submit one or multiple URLs to the system, which will then analyze and classify them as phishing or legitimate using machine learning models.
- **File Analysis Module:** Users should be able to upload files (e.g., PDF, EXE, DOCX) to detect malicious content. The system should extract features like file hash, metadata, and file structure for analysis.
- **Email Analysis Module:** Users should be able to input raw email text or upload .eml files. The system will parse the email and extract features such as headers, body content, embedded links, and attachments to detect phishing attempts.
- **Feature Extraction and Vectorization:** The system should extract relevant features from URLs, files, and emails and convert them into numerical vectors using techniques like CountVectorizer or TF-IDF for model compatibility.
- **Machine Learning Models:** The system should use Random Forest and XGBoost classifiers to predict whether the input is phishing/malicious or legitimate. Each model should be trained and evaluated independently.
- **Security Measures:** All submitted data should be securely handled, and file/email analysis should occur in a sandboxed environment. The system must avoid rendering or executing potentially harmful content.
- **Integration and API Support:** The application should support API integration for organizations or developers who want to embed phishing detection capabilities into their own platforms.
- **User Feedback Submission:** Users should have the ability to report false positives or negatives,

- which can be stored for retraining and improving model performance over time.

### 3.2 Non-Functional Requirements

Non-functional requirements define the quality attributes of the phishing detection system, ensuring it performs effectively, reliably, and securely across various input types (URLs, files, and emails). These requirements are critical for achieving user trust, high availability, and maintainability in real-world environments. This section, will outline the non-functional requirements for the URL based Phishing Detection using Machine Learning to ensure it meets the expectations of its users.

- **Scalability:** The system should be capable of handling a growing number of users, data inputs, and prediction requests without performance degradation. It should scale horizontally to accommodate spikes in usage or large-scale integrations.
- **Security:** The application must be designed to handle potentially malicious inputs (e.g., phishing URLs, malware-laden files, or spoofed emails) in a secure, sandboxed environment. All user data must be protected using encryption, secure communication (HTTPS), and strict access control.
- **Usability:** The user interface should be clean, intuitive, and easy to navigate for both technical and non-technical users. Users should be able to submit URLs, files, or emails with minimal effort and understand the prediction results clearly.
- **Reliability:** The system must provide consistent availability and accuracy in predictions. It should have robust fallback mechanisms and redundancy to avoid downtime and ensure continuous operation even under heavy loads or failures.
- **Performance:** The phishing detection engine should process inputs and return predictions quickly (in real-time or near real-time). Latency should be minimized through model optimization and efficient feature extraction.
- **Interoperability:** The system should be designed with APIs that allow integration with third-party platforms such as email servers, web browsers, secure gateways, and SIEM (Security Information and Event Management) tools.
- **Accessibility:** The platform should follow accessibility standards (such as WCAG) to support users with disabilities. It should be compatible with screen readers and provide keyboard navigation and accessible UI elements.
- **Maintenance:** The system codebase should be modular, well-documented, and follow standard software engineering best practices. It should allow developers to easily update models, improve features, and address vulnerabilities

- **Cost-effectiveness:** The system should be optimized for efficient resource usage—whether deployed locally or in the cloud—to reduce operational costs. Efficient model inference and resource allocation will ensure affordability over time.

### 3.3 System requirements

#### 3.3.1 Hardware used

- Computer with sufficient processing power, memory, and storage is required for developing the phishing detection system.
- A minimum of 8 GB RAM is recommended to ensure smooth operation during model training and testing.
- A quad-core processor is recommended to support the machine learning training tasks and data preprocessing.
- An SSD with at least 256 GB storage capacity is recommended to store datasets, model files, and required libraries.
- A dedicated graphics card is not necessary but can be beneficial for visual analytics and large dataset processing.
- A reliable internet connection is required to download datasets, install dependencies, and access online resources or threat feeds.

#### 3.3.2 Software used

- **Python Environment:** Python is the primary language used for data preprocessing, model training, and application logic.
- **Integrated Development Environment (IDE):** An integrated development environment such as Jupyter Notebook, Visual Studio Code, or PyCharm is used for development and testing.
- **Machine Learning Libraries:** XGBoost and scikit-learn are used for building predictive models like Random Forest and gradient boosting classifiers.
- **Vectorization Libraries:** Scikit-learn's TfidfVectorizer and CountVectorizer are used to convert URL/email/file features into numerical format for model input.
- **Data Handling:** Libraries like pandas and NumPy are used for data manipulation and preprocessing.
- **Visualization Tools:** Libraries such as matplotlib and seaborn are used for data visualization and analysis.

***CHAPTER 4***  
**SYSTEM DESIGN / METHODOLOGY**

---

## CHAPTER 4

### SYSTEM DESIGN / METHODOLOGY

#### 4.1 Architecture

##### 4.1.1 Design

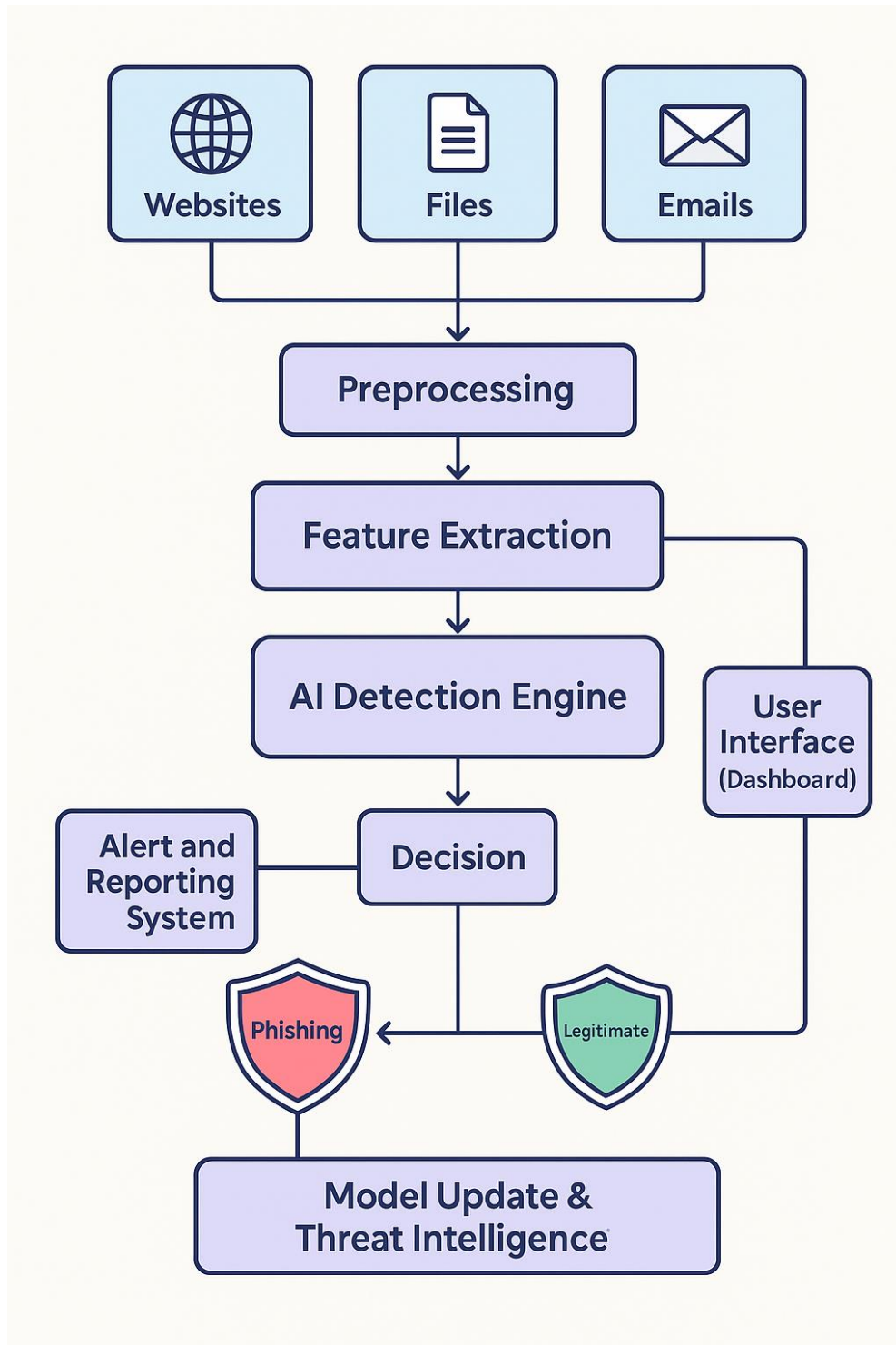


Fig 5.1 System Architecture

## Phishing Detection System Architecture

- **Introduction to Phishing and the Need for Detection System:**

Phishing is a deceptive technique used by malicious actors to acquire sensitive information such as usernames, passwords, credit card numbers, and other personal details. Typically delivered through emails, websites, or malicious files, phishing can cause significant damage to individuals and organizations alike. As phishing attacks grow more sophisticated, traditional methods of detection fall short. This necessitates the development of intelligent, AI-driven phishing detection systems that can proactively identify and mitigate such threats.

The system architecture illustrated here presents a comprehensive framework for an AI-based phishing detection solution. The architecture integrates multiple data sources, intelligent preprocessing and feature extraction, a powerful AI engine, and feedback mechanisms for continuous learning and adaptation.

- **Data Sources**

**Websites:** Many phishing attacks originate from fake websites that closely mimic legitimate ones. These websites often ask users to input sensitive information. The system scans URLs, website content, and meta-information to detect signs of phishing. To detect such sites, the system analyzes:

- ◆ Domain names (e.g., suspicious use of “login” or “secure”)
- ◆ SSL certificate details
- ◆ Redirection patterns
- ◆ Use of iFrames and obfuscated JavaScript
- ◆ Website age and hosting information

**Files:** Phishing can also be embedded in document files (PDFs, Word documents, etc.) that users may download or receive via emails. These files may contain malicious links or scripts that direct users to phishing sites or execute harmful actions. Phishing is frequently delivered through document-based malware. These files may contain:

- ◆ Embedded malicious macros
- ◆ Clickable links redirecting to phishing domains
- ◆ Obfuscated code designed to exploit system vulnerabilities



- ◆ Fake forms requesting sensitive data

The system parses files like PDFs, Word documents, and spreadsheets to extract hyperlinks, analyze macro behavior, and detect obfuscation techniques.

**Emails:** Emails are one of the most common vectors for phishing. The system analyzes email headers, content, embedded links, and attachments to determine whether the email is part of a phishing attempt.

Emails are the most common vector for phishing attacks. Common characteristics include

- ◆ Spoofed sender addresses
- ◆ Poor grammar and spelling
- ◆ Urgent or threatening language
- ◆ Suspicious attachments or links
- ◆ Lack of personalization

The detection engine extracts email headers, sender domains, subject lines, and content to identify these indicators.

- **Preprocessing**

Preprocessing is the foundation of any intelligent system, ensuring data consistency and format normalization. Each data type (web, file, email) is handled by specialized pipelines:

- ◆ **Text Cleaning:** Remove unnecessary HTML tags, JavaScript, or encoded characters.
- ◆ **Tokenization:** Break down text into individual components such as words, phrases, or tags.
- ◆ **Normalization:** Convert all data to lowercase, standard date formats, and remove special characters.
- ◆ **Parsing:** For emails, parse MIME structure, header information (From, To, Received lines), and attachments.

This stage is essential for improving the quality of the features used in machine learning.

- **Feature Extraction**

Feature extraction converts raw data into a structured form suitable for machine learning. Good features help the AI model understand the patterns behind phishing attempts. Examples include:

**For Websites:**

- ◆ Length and entropy of URLs
- ◆ Presence of IP address instead of domain
- ◆ Use of HTTPS and certificate information
- ◆ Favicon mismatch with domain
- ◆ Number of external links

**For Emails:**

- ◆ Whether SPF/DKIM passes
- ◆ Use of urgent keywords like “action required,” “verify,” “suspended”
- ◆ Mismatch between sender name and domain
- ◆ HTML-only body vs. plain text

**For Files:**

- ◆ Number of embedded links
- ◆ Macro behavior (e.g., auto-execution)
- ◆ File obfuscation level
- ◆ Use of external object references

Each extracted feature can be categorical (e.g., "Has Login Form: Yes/No") or numerical (e.g., "URL Length: 92").

- **AI Detection Engine**

This is the heart of the system and leverages machine learning models trained on labeled datasets of phishing and legitimate samples. Types of models used include:

- ◆ **Traditional Machine Learning Algorithms**
  - **Random Forest:** Handles mixed data types and is resistant to overfitting.
  - **Support Vector Machine (SVM):** Effective for high-dimensional spaces.
  - **Gradient Boosted Trees (e.g., XGBoost, LightGBM):** Highly accurate and fast.
- ◆ **Deep Learning Models**
  - **CNNs (Convolutional Neural Networks):** Good for analyzing spatial structure in images or text layout (e.g., websites).
  - **RNNs (Recurrent Neural Networks):** Handle sequential data, useful for text analysis in emails.
  - **Transformers (e.g., BERT):** State-of-the-art models for language understanding.

- **User Interface (Dashboard)**

The user interface provides real-time visibility into the system's operations. Key features of the dashboard include:

- **Live Alerts:** Shows flagged emails, websites, or files.
- **Searchable Logs:** Enables analysts to query historical detections.
- **Visualization:** Pie charts, bar graphs, and time series plots showing detection trends.
- **Feedback Mechanism:** Analysts can label false positives/negatives to improve the model.

This interface enhances usability and allows cybersecurity teams to respond quickly.

#### 4.1.2 Data Flow Diagram

- **DFD- Level 0**



**Figure 4.1.2a – DFD Level-0**

The data flow diagram presents a high-level overview of the interaction between the user and the AI-powered phishing detection system, capturing the dynamic exchange of data that fuels the entire detection process. The diagram begins with the user, who is the key initiator of this process. Users may include individuals, system administrators, or automated services that encounter potentially suspicious content in the form of URLs, files, or emails. These elements are the most common vectors for phishing attacks. When a user suspects a phishing attempt, they submit the item directly to the system for analysis. This submission could occur through a web portal, email gateway, file upload interface, or even an automated API that scans messages in real-time.

Upon receiving the user's input, the AI-powered phishing detection system begins its internal operations. These inputs serve as raw data, entering the pipeline where they undergo preprocessing

to remove noise and standardize formats. For instance, URLs are parsed to examine their domain structure, files are scanned for embedded scripts or hyperlinks, and emails are analyzed for metadata like headers, senders, and content characteristics. This standardization is critical because it ensures that all incoming data, regardless of source or structure, is ready for consistent analysis by the AI engine.

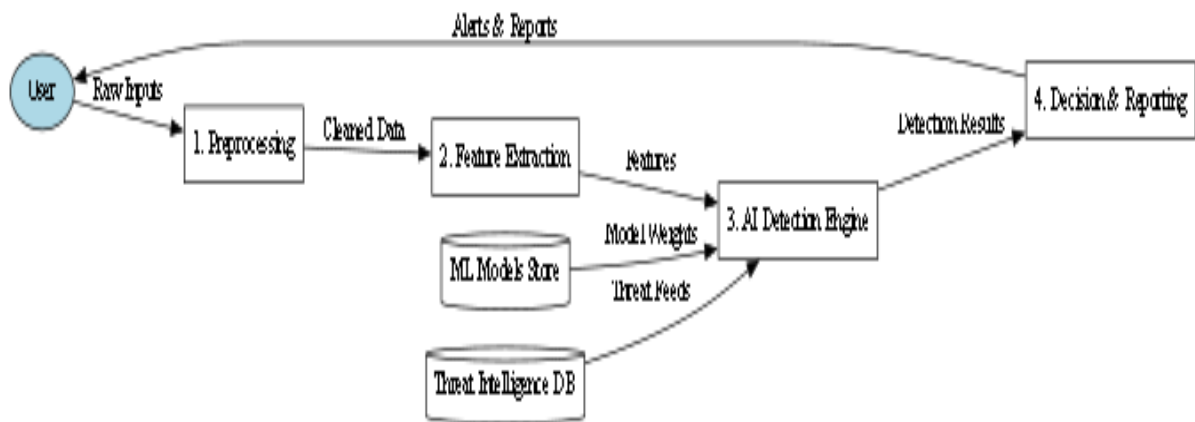
As the detection process continues, the system extracts key features from the inputs. These features are specific data points that the AI model uses to determine whether the input is malicious or legitimate. For URLs, features might include suspicious domain names, length, or whether the URL contains uncommon characters. For emails, the system may analyze the tone, formatting, or suspicious attachments. For files, it can inspect the presence of macros, external references, or embedded scripts. These features are then processed by advanced machine learning models trained on large datasets of phishing and legitimate examples.

Once the system processes the features through the AI detection engine, it evaluates the threat level of the submitted content. The model produces an output score or classification based on the learned patterns of known phishing indicators. The decision mechanism takes over here, interpreting the model's output and determining whether to classify the input as phishing, legitimate, or uncertain. If a phishing threat is identified, the system takes immediate action by generating alerts and compiling detailed reports.

These alerts and reports are then communicated back to the user, closing the loop. Alerts are designed to be urgent notifications that inform the user of potential threats in real-time. They may be displayed through the system's user interface or sent via integrated channels such as email notifications, SMS, or system logs. In addition to alerts, detailed reports are generated to provide deeper insights into the threat. These reports include valuable metadata such as the threat type, detection confidence level, time of detection, and recommended next steps. Reports also serve as records for compliance, auditing, and future analysis.

What makes this feedback loop particularly powerful is its role in continuous learning and system improvement. Users may review the alerts and provide feedback—marking an alert as a false positive or confirming a phishing attempt. This feedback can be reintegrated into the training process to refine the AI model, making it more accurate over time. Furthermore, the interaction between users and the system fosters a collaborative approach to cybersecurity. It encourages users to remain vigilant while simultaneously equipping the detection system with more contextual information for learning.

- **DFD-Level-1**



**Figure 4.1.2b– DFD Level-1**

The data flow diagram provides an in-depth representation of the internal processes of a phishing detection system, highlighting the various components and data exchanges that transform raw user inputs into actionable alerts and reports. The flow begins with the user, who acts as the primary source of data for the system. Users submit diverse inputs, which may include URLs, email content, or file attachments that are suspected of being phishing attempts. These inputs are unstructured and potentially noisy, containing everything from benign text and hyperlinks to malicious payloads. To make sense of this data, the first operational block of the system, labeled “1. Preprocessing,” is activated. This stage is responsible for cleaning, normalizing, and structuring the raw input. In this process, extraneous characters, irrelevant metadata, and encoding anomalies are removed or standardized to ensure that the data is ready for analytical processing. For example, a URL might be stripped down to its core domain components, an email might be parsed into headers and body text, and a document file might have macros extracted and decoded.

Once preprocessing is complete, the now “cleaned data” is forwarded to the next module, “2. Feature Extraction.” This is a critical phase in which the system analyzes the input to identify significant attributes, known as features, that will serve as indicators for phishing detection. Features may include lexical characteristics of URLs (such as length, use of special characters, or domain age), metadata from emails (like sender address mismatches or suspicious subject lines), and technical aspects of documents (such as the presence of JavaScript or attempts to access external servers). These features are numerical or categorical representations of patterns and

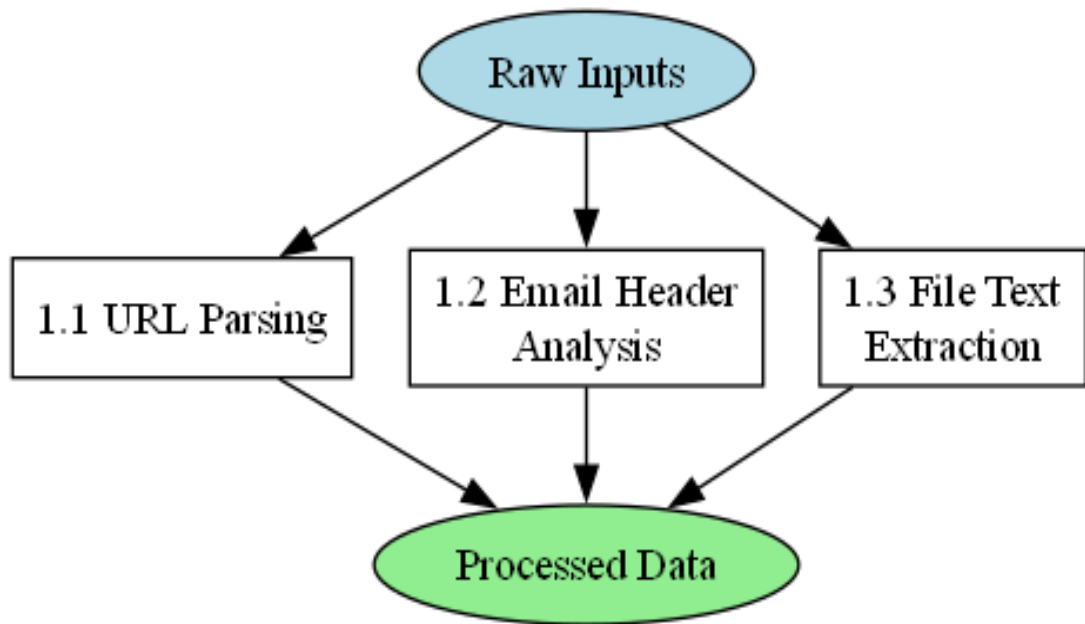
behaviors commonly associated with phishing attacks. The goal of this stage is to transform complex and varied inputs into a structured format that the AI engine can understand and process.

Parallel to this feature extraction process, two important backend databases are integrated into the system: the “ML Model Store” and the “Threat Intelligence DB.” The ML Model Store holds the pre-trained machine learning models along with their associated weights and parameters. These models may be trained using various supervised learning algorithms and are capable of evaluating the extracted features to determine the likelihood of an input being phishing. The Threat Intelligence Database (Threat Intelligence DB) supplements the detection process by providing contextual threat data gathered from global sources. This may include blacklisted domains, known phishing signatures, recent attack trends, and heuristic patterns. The extracted features, model weights, and threat feeds are all funneled into the next phase: “3. AI Decision Engine.”

The AI Decision Engine is the analytical core of the system. It ingests the extracted features along with model configurations and contextual threat data to make an informed assessment of the input. Using advanced AI techniques—such as neural networks, ensemble classifiers, or hybrid detection models—the engine calculates the probability that the input represents a phishing attempt. The engine also takes into account external threat signals, such as whether the domain in a URL matches one found in the Threat Intelligence DB, or whether the writing style in an email mimics known phishing templates. The AI Decision Engine outputs a classification decision, which typically falls into categories such as “Phishing,” “Legitimate,” or “Uncertain.” This classification is forwarded to the final phase of the system.

The final stage, labeled “4. Decision & Reporting,” is responsible for managing the system’s response to the detection outcome. When a phishing threat is detected, the system triggers appropriate alerts to the user or system administrators. These alerts may include real-time notifications through dashboards, emails, or security software integrations. In addition to alerts, detailed reports are generated, documenting the nature of the threat, the features that contributed to the decision, the confidence score of the model, and any associated threat intelligence. This reporting function is not only essential for user awareness but also for compliance, auditing, and further training of the model. Even in cases where the detection is classified as legitimate or uncertain, the system maintains logs of the interaction, which can later be used to fine-tune model accuracy or investigate evolving threat patterns.

- **DFD-Level-2**



**Figure 4.1.2c– DFD Level-2**

The flow diagram presented represents a critical component of the phishing detection system, focusing specifically on the initial preprocessing phase where raw inputs are transformed into structured and analyzable data. At the top of the diagram, the process begins with “Raw Inputs,” which include the diverse data elements submitted by users or monitoring systems. These raw inputs may consist of URLs, email messages, and file attachments—all of which are potential carriers of phishing threats. The preprocessing mechanism branches into three parallel subprocesses, each tailored to a specific input type and aimed at converting unstructured or semi-structured data into a consistent and interpretable format referred to as “Processed Data.”

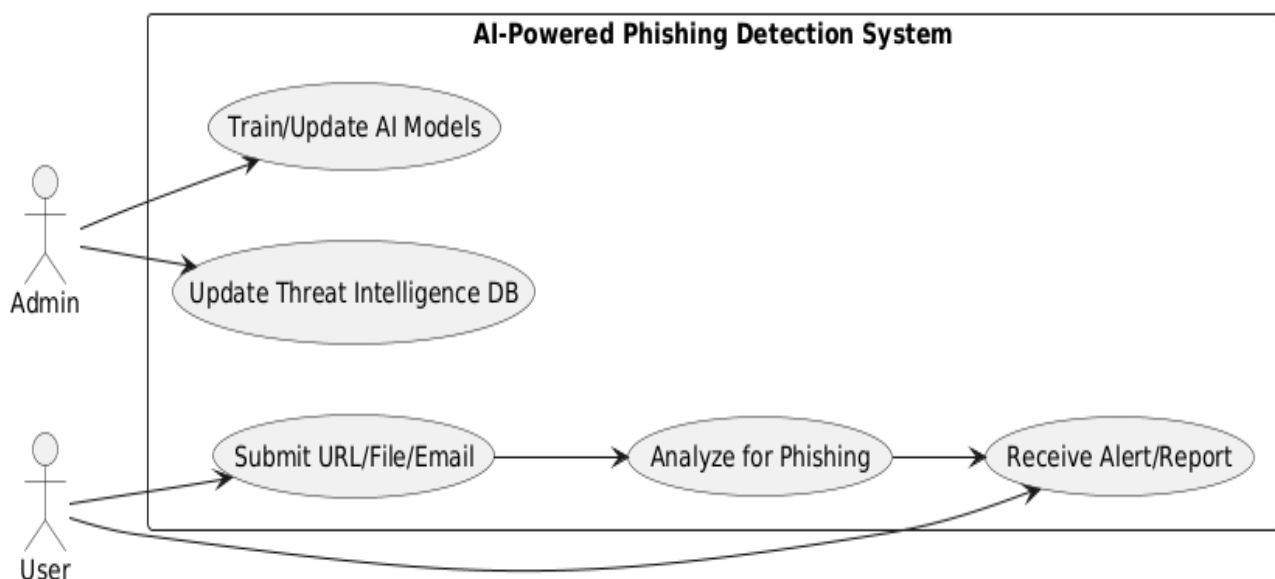
The first subprocess, labeled 1.1 URL Parsing, is responsible for analyzing any URLs provided in the input data. This involves decomposing URLs into their constituent parts such as the protocol, domain name, path, parameters, and query strings. The parsing process enables the detection system to evaluate the characteristics of a URL, such as whether it redirects to suspicious domains, uses obfuscation techniques, or contains misspelled words mimicking trusted sites. These features are crucial for identifying phishing websites designed to deceive users by impersonating legitimate web services.

The second subprocess, 1.2 Email Header Analysis, focuses on examining the metadata contained in email messages. Email headers include vital information such as the sender’s address, recipient details, time of sending, mail server paths, and authentication results (like SPF, DKIM, and DMARC).

The third subprocess, 1.3 File Text Extraction, handles any attached files or documents submitted for phishing analysis. This component is designed to extract readable text and metadata from a variety of file formats, such as PDFs, Word documents, and spreadsheets. Extracted text is then scrutinized for suspicious content, such as embedded links, macros, JavaScript code, or persuasive language patterns often found in phishing scams. Additionally, this extraction allows the system to uncover hidden data or malicious payloads that may not be apparent during a superficial scan.

Once each subprocess completes its respective analysis and transformation, the resulting data is unified into a standardized format known as Processed Data. This processed data forms the foundation for subsequent stages in the phishing detection pipeline, such as feature extraction and AI-based classification. The effectiveness of the entire detection system heavily depends on the quality and completeness of this preprocessing stage. By ensuring that URLs, emails, and files are parsed and interpreted correctly, this initial phase guarantees that the downstream AI models receive the necessary structured input to make accurate and reliable decisions.

#### 4.1.3 Case Diagram of the Application:



**Figure 4.1.3– Use Case Diagram**

- **Overview:** The use case diagram in Fig 4.1.3 provides a structured overview of the primary



- interactions between different types of users and the AI-powered phishing detection system. It highlights how the system functions in real-world usage, separating the responsibilities and roles of administrators and general users. This diagram captures the dynamic interplay between user input, system analysis, and alert generation, showcasing both backend intelligence management and frontend user interaction.

- **Actors in the System**

There are two main actors in this use case diagram: the **Admin** and the **User**. Each actor has distinct roles that align with their access level and responsibilities in the system.

- **Admin:** The Admin has elevated privileges and is responsible for maintaining and enhancing the core intelligence of the system.
- **User:** The User represents any individual interacting with the system to detect and prevent phishing threats, such as employees, customers, or external clients.

- **Admin Use Cases**

The Admin interacts with the system through two primary use cases:

- **Train/Update AI Models:** This use case involves feeding new training data into the system to refine the AI's phishing detection algorithms. It ensures the model adapts to new threat patterns and maintains high accuracy.
- **Update Threat Intelligence Database:** This function allows the Admin to enrich the system's threat intelligence by incorporating newly discovered phishing indicators, blacklisted domains, and behavioral patterns. Keeping this database current is vital for the detection system to remain effective.

- **User Use Cases:**

The User engages with the system through a straightforward yet powerful workflow consisting of three use cases:

- **Submit URL/File/Email:** Users provide the system with potentially malicious content. These inputs are the starting point for the phishing analysis pipeline.
- **Analyze for Phishing:** The system processes the submitted input using its AI-based detection engine. It extracts features from the content, matches them against threat intelligence, and uses machine learning models to assess the likelihood of phishing.
- **Receive Alert/Report:** Based on the analysis results, the user receives a real-time alert if a threat is detected. In addition to alerts, detailed reports may be generated explaining the basis for the classification, potential risks, and recommendations.

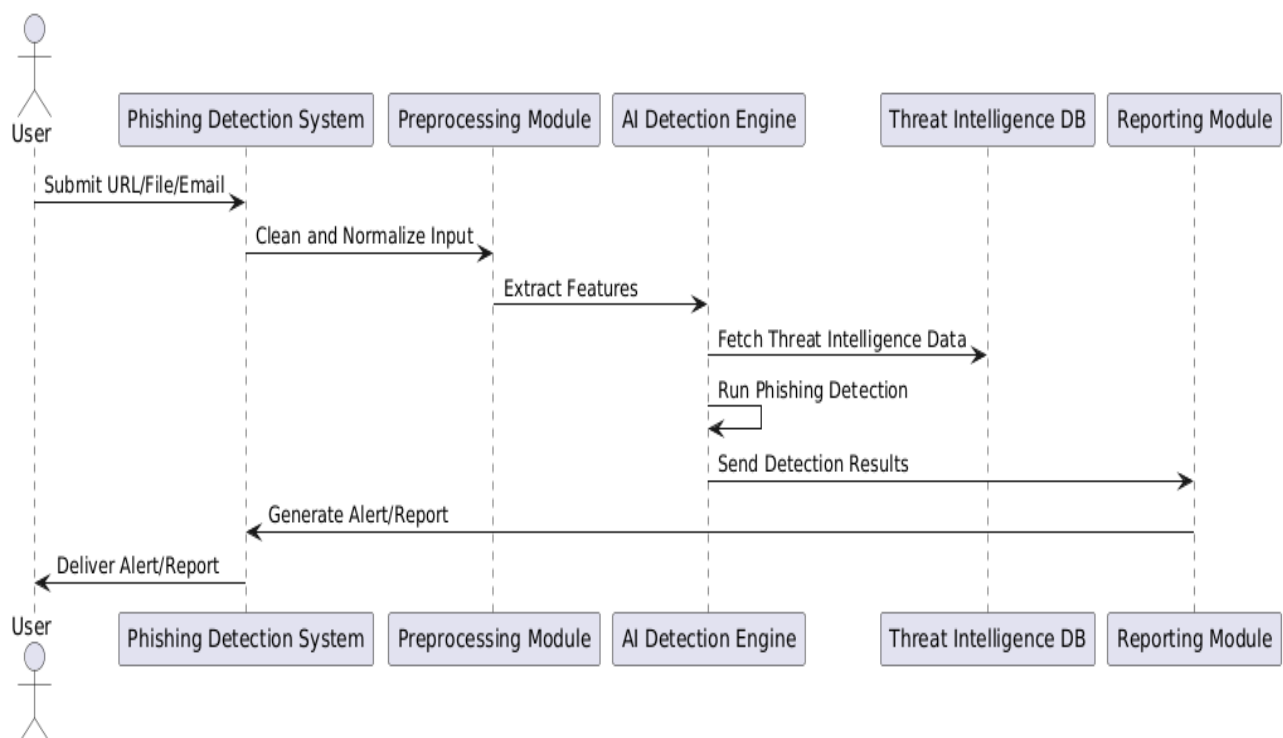
- **System Functionality**

The use case diagram emphasizes the importance of both user interaction and continuous system enhancement. While users benefit from real-time detection and alerting, the Admin ensures the system's effectiveness through regular updates and training. This dual-layered interaction model helps maintain a balance between usability and technical sophistication, making the system robust, adaptive, and suitable for dynamic cybersecurity environments.

- **Conclusion**

This use case diagram clearly delineates the roles and responsibilities of different stakeholders within the phishing detection system. By visualizing these interactions, the diagram highlights the collaborative nature of maintaining cybersecurity—where both user input and administrative oversight are essential.

#### 4.1.4 Sequence Diagram of the Application:



**Figure 4.1.4-Sequence Diagram**

- **Overview:**

The sequence diagram shown in Fig 5.6 illustrates the step-by-step flow of data and operations involved in the phishing detection system. This diagram captures the interaction between various system components and the user, mapping the chronological sequence in which tasks are performed. It presents a clear and structured representation of how a URL, file, or email input is processed by the system to determine whether it is a phishing attempt. The diagram provides insights into the internal modules responsible for preprocessing, AI-based detection, intelligence integration, and reporting—showing how each module contributes to the end-to-end process.

- **User Interaction and Input Submission**

The process begins when a User submits suspicious content to the Phishing Detection System. The input can take the form of a URL, a file attachment, or an email that the user suspects

might be malicious. This user-submitted content is forwarded to the Preprocessing Module, which serves as the first internal component of the system.

- **Preprocessing and Feature Extraction**

The Preprocessing Module performs the task of cleaning and normalizing the input. This is a critical step that ensures the data is formatted consistently, removes noise, and isolates the relevant parts of the content (e.g., domain names, metadata, payloads). Once the input has been cleaned, the module proceeds to extract features—specific attributes or patterns that can be used by machine learning models to assess the risk of phishing. These features may include characteristics like domain age, presence of suspicious keywords, link obfuscation, or email header inconsistencies.

- **AI Detection Engine and Threat Intelligence**

The extracted features are then passed to the AI Detection Engine, which plays a central role in the decision-making process. Before making a decision, the engine interacts with the Threat Intelligence Database (DB) to fetch relevant threat intelligence data. This information includes known phishing indicators, malicious IPs or domains, signature patterns, and behavioral heuristics. The combination of extracted features and real-time intelligence data is used to run phishing detection algorithms—often based on trained machine learning models or neural networks.

Once the detection is complete, the AI engine sends the results to the next module. These results typically include a classification label (e.g., “Phishing” or “Legitimate”), a confidence score, and possibly a rationale or list of matched threat indicators.

- **Alert and Reporting**

The detection results are received by the **Reporting Module**, which is responsible for generating a meaningful alert or report for the user. This module compiles the detection decision into a structured format, possibly including visual dashboards, textual summaries, and recommended next steps for mitigation. The final output is then sent back to the **User**, completing the loop with a **delivered alert or report**. This feedback allows users to take informed actions, such as blocking a link, deleting a suspicious email, or notifying IT administrators.

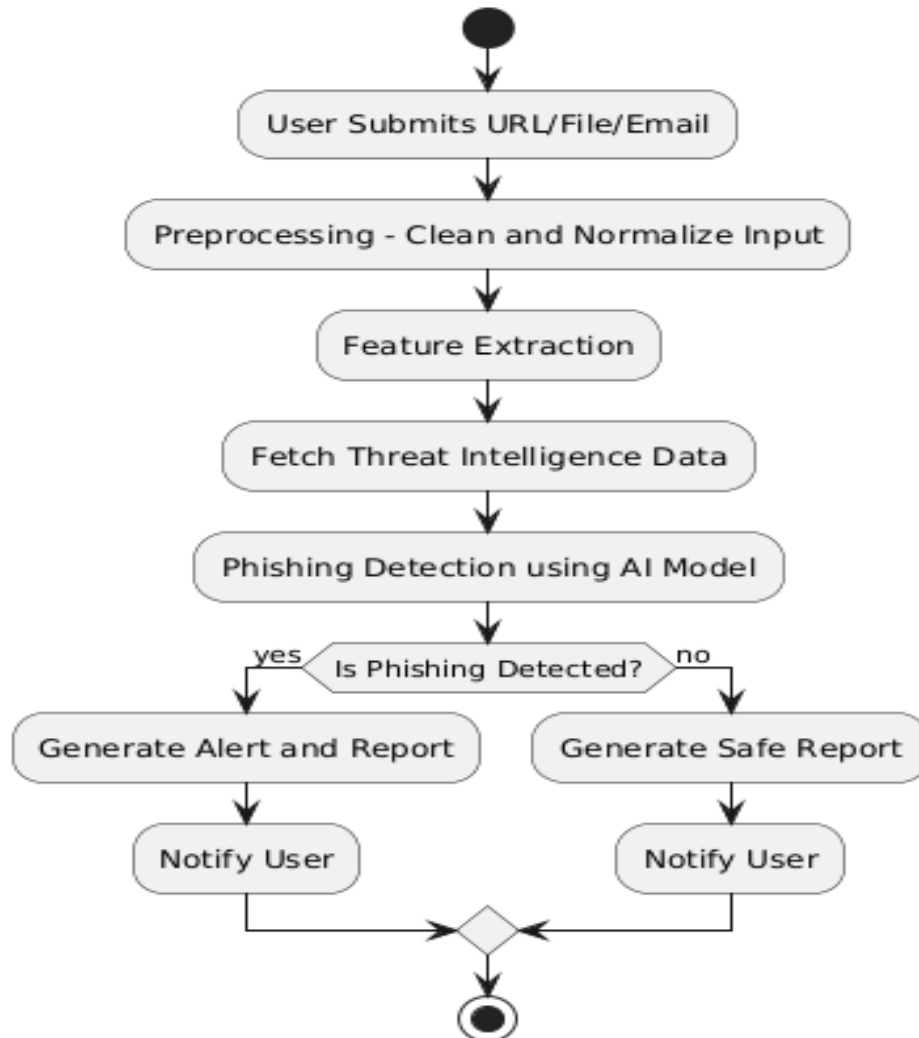
- **System Integration and Flow**

The sequence diagram highlights the smooth integration between the different system components. Each module performs a specific function and hands off data to the next in a seamless pipeline. The use of real-time data from the threat intelligence DB enhances the system's ability to detect new or evolving phishing threats. Additionally, the reporting module ensures that users receive actionable insights, rather than just raw data or binary classifications.

- **Conclusion**

In summary, the sequence diagram effectively conveys the dynamic, real-time operations of the phishing detection system. It demonstrates how user-submitted data is transformed into a meaningful security response through coordinated actions by multiple internal components. The integration of preprocessing, AI analysis, threat intelligence, and reporting showcases the system’s ability to function both accurately and efficiently. This structured workflow is essential for delivering timely and reliable phishing detection capabilities in real-world cybersecurity applications.

## 4.2 Methodology



**Figure 4.2 Methodology of Phishing detection**

The phishing detection system proposed in this project follows a structured and modular methodology that enables efficient and accurate identification of phishing threats from various digital sources such as URLs, emails, and file attachments. Each step is designed to ensure robust digital sources such as URLs, emails, and file attachments. Each step is designed to ensure robust data handling, intelligent analysis, and effective communication of outcomes to end-users. This methodology integrates preprocessing techniques, feature engineering, artificial intelligence, and threat intelligence databases into a unified framework that delivers both high accuracy and scalability.

- **Step 1: Input Submission:**

The process begins when a user submits a URL, file, or email for analysis. These inputs are typically items that the user suspects to be malicious or deceptive. The system is designed to accept multiple formats and ensure seamless intake of data, whether it's a clickable link, a suspicious attachment, or the content of an email. This flexibility is critical in real-world environments where phishing attempts manifest in various formats.

- **Step 2: Preprocessing - Clean and Normalize Input:**

Once the input is received, it enters the Preprocessing phase. In this stage, the system performs a series of cleaning operations to remove irrelevant characters, normalize text formats, extract key metadata (e.g., headers, URLs, sender details), and ensure the input is in a consistent structure. For example, email headers are parsed to extract sender domains and timestamps, while URLs are tokenized and standardized. Normalization helps reduce complexity and facilitates better pattern recognition in later stages. This step is essential to remove noise from the data and prepare it for analytical processing.

- **Step 3: Feature Extraction**

After preprocessing, the system enters the Feature Extraction stage. This is a pivotal step where the system derives meaningful attributes from the input data that can be used for classification. Common features include the domain age, URL length, presence of obfuscation techniques (like use of IP addresses instead of domain names), and language anomalies in the body of an email. These features are crucial indicators that help distinguish legitimate content from phishing attempts. The success of the AI model hinges significantly on the quality and relevance of these extracted features.

- **Step 4: Fetch Threat Intelligence Data**

The next step involves interfacing with a Threat Intelligence Database. The system queries this repository to fetch known indicators of compromise (IoCs) such as blacklisted domains, malicious IP addresses, suspicious file hashes, and previously reported phishing campaigns. This real-time threat intelligence is used to augment the input data and provide contextual information that may not be visible through static analysis alone. By incorporating external intelligence sources, the system enhances its ability to detect new and evolving phishing tactics.

- **Step 5: Phishing Detection Using AI Model**

With both internal features and external intelligence combined, the data is passed through an AI-based phishing detection model. This model is typically a machine learning algorithm trained on large datasets of phishing and legitimate samples. It applies statistical and pattern recognition techniques to determine the probability that the input is malicious. The model produces a binary classification (phishing or not) along with a confidence score. This AI-driven

approach allows the system to learn from new data over time and adapt to sophisticated phishing techniques that might bypass traditional rule-based systems.

- **Step 6: Decision Making**

At this decision point, the system evaluates the output of the AI model to determine if phishing is detected. If the model indicates that the input is indeed a phishing attempt, the system branches into a set of actions aimed at threat mitigation. If no threat is detected, a separate set of actions is triggered to confirm the legitimacy of the input.

- **Step 7a: If Phishing is Detected**

When phishing is detected, the system proceeds to generate an alert and detailed report. This report includes a summary of the threat, reasons for detection, matched indicators, and recommended user actions such as deleting the email, blocking the sender, or flagging the file for further review. The report is then sent to the user, notifying them of the malicious nature of the content. Additionally, the threat data may be logged and used for future model updates or shared with broader security infrastructure (e.g., SIEM systems).

- **Step 7b: If No Phishing is Detected**

If the AI model determines that the input is legitimate, the system generates a safe report indicating that no phishing elements were found. This report assures the user that the input does not pose any threat and may also provide a brief summary of the analysis conducted. The user is then notified accordingly, closing the loop with timely feedback.

## 4.3 Major Algorithms

### 4.3.1 Algorithm for Email Protection – XG Boost:

- **Step 1: Data Collection:**

The first step involves collecting a large dataset of emails, including both legitimate and phishing messages. These can be gathered from open-source datasets, organizational email logs, or threat intelligence feeds. The data should include full email content, headers, sender information, and embedded URLs.

- **Step 2: Data Preprocessing**

Pre-processing prepares the raw data for analysis. This includes:

- Removing noise such as HTML tags, special characters, and stop words.
  - Normalizing text (lowercasing, removing punctuation).
  - Tokenizing and lemmatizing the email text using NLP libraries like NLTK or spaCy.
- This step ensures the data is clean and uniform, making it suitable for feature extraction.

- **Step 3: Feature Extraction**

Relevant features that can indicate phishing are extracted. These include:

- **URL-based features** (e.g., number of links, suspicious domains, use of IPs instead of domains).
- **Text-based features** (e.g., presence of urgency-related words, greetings, grammatical errors).
- **Header features** (e.g., sender domain mismatch, forged “Reply-To”).
- **Attachment presence and type.**

Tools like **scikit-learn** or **pandas** are used to handle and organize these features.

- **Step 4: Feature Encoding**

Since machine learning models require numerical input, the extracted features are converted using:

- Label encoding or One-hot encoding for categorical data.
- TF-IDF or Count Vectorizer for textual content.

This transforms the dataset into a structured numerical format.

- **Step 5: Dataset Splitting**

The dataset is divided into training and testing sets, usually in a ratio such as 80:20 or 70:30. This allows the model to learn from one portion of the data and be evaluated on another.

- **Step 6: Model Training with XGBoost**

The XGBoost library is used to train the model. XGBoost is a gradient boosting algorithm known for its performance and speed. During training:

- Model parameters such as learning rate, tree depth, and number of estimators are tuned.
- Cross-validation techniques (e.g., k-fold) may be used to improve reliability.

The model learns patterns that distinguish phishing emails from legitimate ones.

- **Step 7: Model Evaluation**

Once trained, the model is tested using the test dataset. Evaluation metrics include:

- **Accuracy** – overall correct predictions.
- **Precision and Recall** – how well phishing is detected versus missed.



- **F1-score** – balance between precision and recall.
- **Confusion Matrix** – detailed breakdown of classification results.

These metrics help in assessing the model's effectiveness.

- **Step 8: Deployment and Real-Time Detection**

After validation, the model is deployed in a live email protection system. Incoming emails are passed through the same preprocessing and feature extraction steps, then classified by the XGBoost model as either phishing or legitimate.

- **Step 9: Alert Generation**

If an email is classified as phishing:

- The system generates an alert or warning.
- A report is created and may be forwarded to the security team.
- The user is notified through the dashboard or email.

- **Step 10: Model Update and Maintenance**

Cyber threats constantly evolve, so the model must be retrained periodically with new data.

The system:

- Incorporates new phishing samples.
- Updates the Threat Intelligence Database.
- Retrains the XGBoost model to learn new patterns.
- This ensures continued effectiveness against emerging threats.

### 4.3.2 Algorithm for File Protection-Random Forest

- **Step 1: Data Collection:**

Gather a comprehensive dataset of files labeled as either malicious or benign from trusted sources.

- **Step 2: File Metadata Extraction**

Extract basic metadata from each file such as name, extension, size, and creation date.

- **Step 3: Content Feature Extraction**

Analyze the file's internal structure to collect features like entropy, embedded macros, and script presence.

- **Step 4: Hashing Files**

Generate hash values (e.g., MD5, SHA-256) for integrity verification and duplicate

detection.

- **Step 5: Label Assignment**

Assign each file a label—"malicious" or "safe"—based on known results or antivirus flags.

- **Step 6: Data Cleaning**

Remove irrelevant, duplicate, or corrupted file entries to maintain dataset quality.

- **Step 7: Handling Missing Values**

Impute or discard missing feature values to ensure data consistency before model training.

- **Step 8: Normalization of Numerical Features**

Scale numerical data such as file size or entropy to a uniform range for better model performance.

- **Step 9: Encoding Categorical Features**

Convert non-numeric features like file type into numerical format using encoding techniques.

- **Step 10: Feature Selection**

Choose the most relevant features that contribute significantly to classification accuracy.

**Step 11: Dataset Splitting**

Divide the dataset into training and testing sets, typically using a ratio like 80:20.

- **Step 12: Initialize Random Forest Model**

Set up the Random Forest Classifier with desired parameters such as number of trees and depth.

- **Step 13: Model Training**

Train the Random Forest using the training data to learn classification patterns.

- **Step 14: Model Evaluation**

Evaluate the model's accuracy, precision, recall, and F1-score using the testing data.

- **Step 15: Deployment in File Protection System**

Deploy the trained model into a real-time file monitoring system to classify and block malicious files.

- **Step 16: Alert Generation**

Trigger an alert or log when a file is classified as malicious by the model.

- **Step 17: Continuous Monitoring**

Monitor the system's performance and detect any degradation in accuracy over time.

- **Step 18: Periodic Model Updating**

Regularly retrain the model with new file samples to adapt to evolving threat

### 4.3.3 Algorithm for URL Protection-Vectorization

- **Step 1: URL Dataset Collection**  
Collect a dataset of URLs labeled as “phishing” or “legitimate” from public sources or logs.
- **Step 2: Label Assignment**  
Ensure each URL is correctly labeled based on reputation databases or known phishing blacklists.
- **Step 3: URL Cleaning**  
Remove unwanted parameters or trailing slashes to standardize the URLs.
- **Step 4: URL Tokenization**  
Break down URLs into meaningful parts such as domain, path, query, and subdomains.
- **Step 5: Feature Engineering**  
Extract features like length of URL, number of dots, use of special characters, and presence of IP address.
- **Step 6: Word-Based Token Extraction**  
Split tokens into words or substrings (e.g., “login”, “secure”, “paypal”) for semantic analysis.
- **Step 7: Stop Word Removal**  
Remove generic or irrelevant terms (like “www”, “com”) to reduce noise.
- **Step 8: Vectorization Using Count Vectorizer**  
Convert tokenized URLs into numerical vectors based on word frequency using CountVectorizer from scikit-learn.
- **Step 9: Vectorization Using TF-IDF**  
Alternatively, use **TF-IDF Vectorizer** to assign importance weights to tokens based on their frequency and rarity.
- **Step 10: Encode Categorical Features**  
Apply one-hot or label encoding for any additional non-numeric categorical data (e.g., protocol type).
- **Step 11: Combine All Features**  
Merge all vectorized and engineered features into a single dataset for model input.
- **Step 12: Normalize Feature Values**  
Scale all numerical feature values into a consistent range using normalization techniques.
- **Step 13: Dataset Splitting**  
Split the vectorized dataset into training and testing sets, typically with an 80:20 ratio.

- **Step 14: Choose a Classifier**

Select a suitable classifier such as Logistic Regression, SVM, or XGBoost to apply on the vectorized data.

- **Step 15: Train the Model**

Feed the training data into the classifier to learn patterns associated with phishing URLs.

- **Step 16: Evaluate the Model**

Test the model on unseen data and evaluate using metrics like precision, recall, accuracy, and ROC-AUC.

- **Step 17: Real-Time URL Detection**

Deploy the trained model to classify new URLs in real-time during web browsing or email scanning.

- **Step 18: Alert and Block Malicious URLs**

Automatically generate warnings and block access when a URL is detected as phishing.

- **Step 19: Update Model with New URLs**

Continuously gather and add new phishing URLs retrain and improve the model periodically.

***CHAPTER 5***  
**IMPLEMENTATION**

---

## CHAPTER 5

# IMPLEMENTATION

### 5.1 Flask

**Flask** is a lightweight and widely-used open-source web framework written in Python. It is designed to build robust, scalable, and maintainable web applications and APIs quickly. Flask follows a minimalist design philosophy, offering essential tools and components while allowing developers to plug in additional libraries and extensions as needed. This simplicity and flexibility make Flask a preferred choice for both beginner and advanced developers. One of the key features of Flask is its modular architecture, which allows developers to organize their application into reusable and independent components. This structure supports cleaner code, easier testing, and faster debugging. Flask uses the Werkzeug toolkit for request handling and the Jinja2 templating engine to enable dynamic HTML rendering, allowing web pages to display model outputs and data in real time. The framework also supports routing, sessions, form submissions, and integration with databases like SQLite, making it highly suitable for full-stack web development.

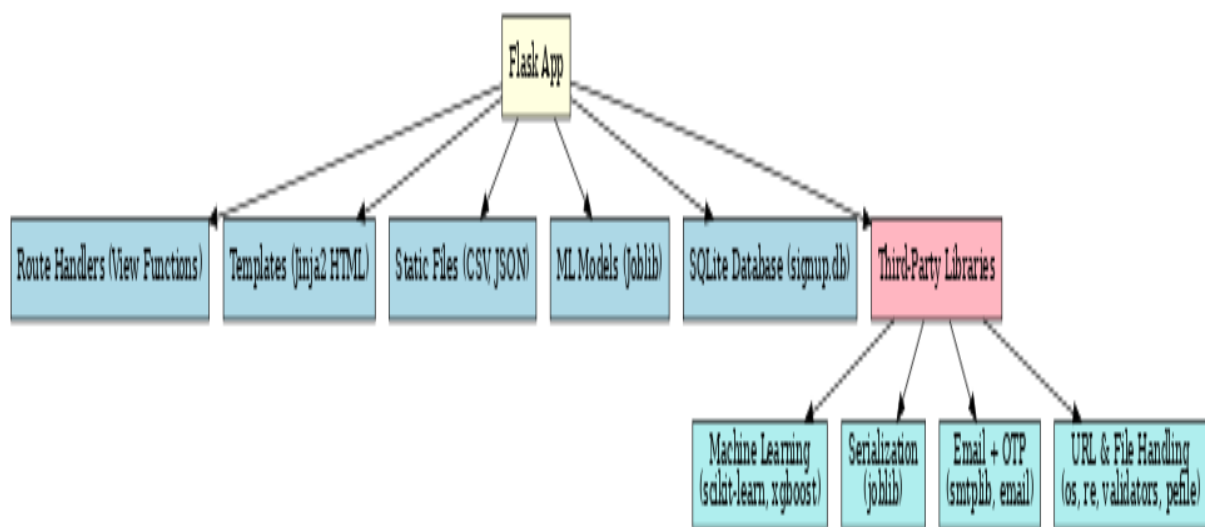
In the context of this project, Flask is used to integrate machine learning models into a web interface, enabling real-time phishing detection through URL, email, and file analysis. It handles HTTP requests, loads serialized models using joblib, processes user input, and displays prediction results through interactive frontend templates. Flask can also work alongside other Python libraries such as pandas, scikit-learn, and XGBoost, offering seamless data processing and model inference capabilities.

#### 5.1.1 Flask Architecture

Flask framework is a micro web framework and does not enforce any specific architectural pattern. Developers are given the flexibility to design the application structure as per their project requirements. Flask provides several powerful features such as routing, template rendering, request and response handling, and session management that help in building well-structured web applications.

- Flask application begins with a single Flask object that acts as the application core.
- The application is structured into multiple routes, each handling a specific function (e.g., authentication, URL/email/file phishing detection).

- Each route maps to a view function that processes input, runs ML models, and returns results.
- Flask promotes composition by encapsulating features into modular routes or functions.
- HTML templates with Jinja2 are used for the user interface and are dynamically rendered with backend data.
- React app can include third party component for specific purpose such as routing, animation, st



**Figure 5.1 – Structure of FlaskApplication**

### 5.1.2 The scikit learn library

The scikit-learn module is a powerful and widely-used machine learning library in Python that provides simple and efficient tools for data mining and data analysis. It is built on top of popular libraries like NumPy, SciPy, and matplotlib, and it is designed to integrate seamlessly into Python-based machine learning pipelines, such as those built using the Flask framework.

The scikit-learn library offers a variety of machine learning algorithms and utilities that make it easy to build and evaluate predictive models. In the phishing detection system, it is primarily used for **email** phishing classification through TF-IDF vectorization and model training using classifiers such as XGBoost and Random Forest.

Key components and functionalities used from scikit-learn include:

1. **TfidfVectorizer:** A feature extraction tool that converts raw text data into numerical vectors based on the Term Frequency-Inverse Document Frequency formula. It helps transform email text into a format suitable for model training.
2. **train\_test\_split:** A utility function used to split datasets into training and testing subsets, helping evaluate model performance.
3. **Accuracy\_score, confusion\_matrix, classification\_report:** These are metric tools used to measure the performance of classification models. They provide insight into how well the model distinguishes between phishing and safe emails.
4. **Cross\_val\_score:** This function enables k-fold cross-validation, which helps assess the generalization capability of the trained model.
5. **Pipeline:** A high-level interface for bundling preprocessing and modeling steps, which ensures reproducibility and cleaner code structure. In addition to these core utilities, scikit-learn integrates well with serialization libraries like **joblib** to save trained models and vectorizers for later use in real-time Flask applications.

### 5.1.3 The Joblib Serialization Library

The joblib library is a Python library used for the efficient serialization and deserialization of large Python objects, particularly those used in scientific computing and machine learning. In the context of this phishing detection system, the joblib library is primarily used to save and load machine learning models and vectorizers, ensuring they can be reused in real-time prediction without retraining every time the application is run.

#### 5.1.3.1 Role of Joblib in Flask

- **Model Persistence:** Saves trained machine learning models (like XGBoost) and vectorizers (like TF-IDF) to disk using .pkl files.
- **Efficient Loading:** Allows Flask to load pre-trained models instantly at runtime without re-training.
- **Integration with Routes:** Used in Flask route handlers (e.g., /emailpred) to load models and make predictions based on user input.
- **Supports Reusability:** Enables reusing the same model across different modules or deployment environments.
- **Simplifies Deployment:** Makes production deployment easier, as trained models can be shipped with the app and loaded as needed.



### 5.1.3.2 Features

- **Efficient Serialization:** Joblib is optimized for saving large NumPy arrays, making it ideal for scikit-learn models and vectorizers.
- **Fast Loading:** Models and preprocessing pipelines can be reloaded quickly, supporting real-time inference in web applications.
- **Compression Support:** Offers built-in compression to reduce file size while preserving speed.
- **Compatibility:** Seamlessly works with most machine learning libraries such as scikit-learn, XGBoost, and even custom-built Python classes.

### 5.1.3.3 Benefits

- **Persistent Model Storage:** Enables saving trained models to disk so they can be reloaded without retraining.
- **Improved Performance:** Reduces startup time by avoiding retraining or reinitializing models during app runtime.
- **Simplifies Deployment:** Makes deploying ML models in production (e.g., via Flask) straightforward and efficient.
- **Reusable Pipelines:** Entire preprocessing and model pipelines can be saved and reused across multiple applications or environment.

## 5.2 Machine Learning Models

Machine learning models are algorithms that learn patterns from historical data to make predictions or classifications without being explicitly programmed. In the phishing detection system, machine learning models play a central role in analyzing and classifying emails and files based on learned behavior from labeled datasets. These models are trained using features extracted from input data—such as the content of an email or the structure of a file and are then used to detect whether the input is malicious or safe. Machine learning enables intelligent decision-making within the application, allowing the system to automatically detect phishing attempts with high accuracy. Once trained, the models can generalize their knowledge to make predictions on new, unseen data, making them highly useful for real-time applications like phishing detection.

The benefits of using machine learning models in phishing detection include faster analysis, improved detection accuracy, reduced reliance on static rule sets, and automated adaptability to evolving attack techniques. By using machine learning, the system can be trained on real-world

phishing datasets, which improves its effectiveness and efficiency over traditional detection methods.

In this project, machine learning models are trained using libraries like scikit-learn and XGBoost, and integrated into the Flask backend for real-time predictions. The models are saved using joblib and reloaded during user interactions, ensuring optimal performance and scalability.

Just like smart contracts are built using different programming languages, machine learning offers a variety of model types, each with its own strengths and weaknesses depending on the nature of the data and problem.

- **XGBoost:** A powerful and efficient gradient boosting algorithm that performs exceptionally well on structured/tabular data. It is used in this project to classify phishing emails based on textual content. XGBoost is known for its accuracy, speed, and ability to handle imbalanced data.
- **Random Forest Classifier:** An ensemble method that combines multiple decision trees to improve prediction accuracy and reduce overfitting. It is often used for baseline comparisons and has high interpretability.
- **Support Vector Machine:** An ensemble method that combines multiple decision trees to improve prediction accuracy and reduce overfitting. It is often used for baseline comparisons and has high interpretability.
- **Naïve Bayes:** A probabilistic model based on Bayes' theorem. It is fast and performs well on text classification tasks, especially when features are conditionally independent.
- **Michelson:** This is a functional programming language that is used to write smart contracts on the Tezos blockchain.
- **Logistic Regression:** A simple yet effective algorithm used for binary classification. It is interpretable and often used as a baseline model in ML tasks.

### 5.2.1 XGBoost Model API Integration

XGBoost stands for Extreme Gradient Boosting, and it is a highly efficient and scalable implementation of gradient boosting algorithms. In the context of this phishing detection system, XGBoost is used to classify emails as phishing or legitimate based on textual features extracted from the content using techniques like TF-IDF vectorization. To use the model within a live web application, it is essential to integrate it with an API (Application Programming Interface) to enable seamless communication between the user interface and the machine learning logic.

Machine learning models, including XGBoost, are trained using large datasets and saved to disk in a binary format (e.g., .pkl using joblib). These models cannot operate on their own—they require an API that allows external applications, like the Flask web server, to send input to the model, invoke predictions, and return the output in a human-readable format. This API serves as the interface layer between user input and model inference.

In this project, the XGBoost model is integrated into the Flask application through a series of route handlers (such as /emailpred) that accept user input (email content), vectorize it using the pre-trained TF-IDF vectorizer, load the serialized XGBoost model, and return a prediction label (e.g., “Phishing” or “Safe”).

This integration follows a standardized structure, similar to ABI in smart contract environments. The XGBoost model API expects structured input (vectorized email content), processes it using a defined model interface, and produces output in a predictable format. This ensures consistency and reliability across all prediction requests.

Just like ABI defines the data structure and methods of a smart contract, the XGBoost model API integration ensures that the machine learning model's input format, prediction method, and output structure are standardized for use in the Flask app.

### **5.3 TF-IDF Vectorization**

TF-IDF (Term Frequency-Inverse Document Frequency) is a widely-used statistical method in natural language processing (NLP) for transforming raw textual data into meaningful numeric representations. It evaluates how important a word is to a document in a collection (or corpus), making it a highly effective feature extraction technique for text classification tasks like email phishing detection.

TF-IDF works by combining two measures: Term Frequency (TF), which counts how often a word appears in a document, and Inverse Document Frequency (IDF), which reduces the weight of commonly occurring words and increases the importance of rare, informative words. The result is a matrix where each document (in this case, an email) is represented as a vector of weighted word frequencies—making it suitable for input into machine learning models such as XGBoost, Random Forest, or SVM.

In this project, TF-IDF vectorization is applied to preprocessed email content to extract

numerical features that serve as input to the XGBoost classifier. The vectorizer is trained on a dataset of labeled emails (phishing and non-phishing), and the fitted model is then saved using joblib for reuse in the Flask application during real-time prediction.

Here are some of the key features of TF-IDF vectorization:

1. **Text to Numeric Transformation:** Converts textual email content into fixed-length numerical vectors suitable for input into machine learning models.
2. **Importance Weighting:** Assigns higher weights to unique, informative words and lower weights to common or irrelevant words like "the" or "and".
3. **Sparse Matrix Representation:** Efficiently stores data using sparse matrices where only non-zero values are saved—ideal for large-scale NLP tasks.
4. **Vocabulary Control:** TF-IDF allows tuning parameters such as `max_features`, `ngram_range`, and `stop_words` to control the vocabulary size and type of textual patterns captured.
5. **Scikit-learn Integration:** The `TfidfVectorizer` from scikit-learn is simple to use and integrates seamlessly with pipelines and model training workflows.

## 5.4 URL Heuristic Analysis Engine

The need for a heuristic analysis engine for URLs is essential in phishing detection systems, as many phishing attacks are carried out through malicious links that appear legitimate. While machine learning models are effective for content-based detection (like emails and files), URL-based phishing often relies on subtle patterns and anomalies in the structure of a URL or domain. This is where heuristic analysis becomes a powerful approach.

The URL Heuristic Analysis Engine implemented in this phishing detection system analyzes the components and metadata of a URL to determine whether it is potentially malicious. It operates by checking predefined rules and patterns based on domain properties, structure, and security attributes. These rules are derived from observations of real-world phishing attacks and are embedded into the backend logic of the application.

Heuristic analysis focuses on feature extraction from URLs, such as domain age, URL length, presence of an IP address, SSL certificate status, redirection depth, use of URL shorteners, and other traits often found in phishing links. Each extracted feature is assigned a trust score, and the cumulative score determines whether the URL is flagged as "Genuine" or "Malware".

Testing the heuristic engine involves verifying the correctness of each rule, ensuring that feature extraction logic is accurate, and validating the final trust score classification. Improperly tuned heuristics could result in false positives (flagging safe URLs as phishing) or false negatives (missing malicious URLs), which can lead to user dissatisfaction or security breaches.

Testing strategies for the heuristic engine include unit testing of individual feature checks, integration testing of the full scoring logic, and real-world validation using blacklists and known phishing URL datasets. Additionally, edge cases like very short or excessively long URLs, encoded URLs, or non-standard protocols are tested to ensure robustness.

Security validation is critical here as well. Attackers may try to bypass heuristics using URL obfuscation, redirect chains, or deceptive domain names. Testing must account for such techniques to ensure the engine does not overlook cleverly disguised threats.

There are Key Heuristic Checks Performed by the Engine:

1. **Domain Age Check:** Older domains are generally more trustworthy than recently created ones.
2. **SSL Certificate Validation:** Valid HTTPS certificates are more common in legitimate websites.
3. **Presence of IP Address in URL:** URLs using raw IP addresses instead of domain names are suspicious.
4. **Use of URL Shorteners:** Shortened URLs often hide the actual destination and are common in phishing attempts. Embark is a full-stack development framework for Ethereum dApps.
5. **Domain Rank Check :**URLs not present in the top 1 million domain list may be flagged.
6. **Redirection Count** – Excessive redirections may indicate suspicious behavior.
7. **Special Characters in URL** – Characters like @, //, or excessive . are indicators of phishing.

#### Testing Tools and Techniques Used:

1. **Custom Unit Tests:**

Each feature extraction function is tested using controlled input URLs with known properties.

2. **Blacklisted Dataset Validation:**

URLs from public phishing blacklists are used to validate detection accuracy.

3. **Real-time URL Tests:**

The engine is tested with live URLs to verify correct classification under different network conditions.

#### 4. Stress Testing with Encoded Inputs:

URLs with encoding, redirection, and spoofed domains are tested to ensure the system doesn't break or misclassify.

In conclusion, the URL heuristic analysis engine is a vital layer of defense in phishing detection. By using rule-based feature checks, the system can detect phishing attempts even when no prior training data exists. Thorough testing ensures the engine operates accurately, reliably, and securely just like testing is crucial in smart contract deployment. With well-tested heuristics, the system can offer real-time URL classification and protect users from potential threats.

### 5.4.1 WHOIS and Domain Ranking Features

The WHOIS and domain ranking features are critical components in the URL heuristic analysis engine used in phishing detection systems. These features help evaluate the trustworthiness and legitimacy of a domain by inspecting its ownership, registration history, and popularity. This information is especially important in distinguishing between reputable websites and newly registered or suspicious domains that are often used for phishing attacks.

There are several reasons why WHOIS and domain ranking are effective indicators for phishing detection:

1. **Domain Age (WHOIS):**The WHOIS protocol provides essential metadata about a domain, including the date of registration, domain owner, and expiry date. Phishing websites are often registered for short-term use and may be newly created. By analyzing the domain's age, the system can flag domains that are suspiciously new—an important red flag for phishing.
2. **Registrar and Ownership Details (WHOIS):**Legitimate domains typically use well-known domain registrars and include valid, verifiable owner information. On the other hand, phishing sites may use obscure registrars or hide ownership details using privacy services. This information can be fetched and analyzed using WHOIS queries.
3. **Top-1M Domain Ranking Check:**To verify the domain's reputation, the engine cross-references the domain against a list of the top 1 million websites (e.g., from Alexa or Tranco). Domains not appearing in this list are considered less trustworthy and may be more likely to be phishing sites.
4. **Domain Popularity and Trust Score:**Popular, long-standing domains are generally more

trustworthy. By checking whether a domain is commonly accessed or ranked among the top 1M domains, the engine assigns a higher trust score. Unranked or low-ranked domains are assigned a lower trust score.

#### **Why WHOIS and Domain Ranking Are Important:**

- **Security Insight:** Domains that are newly registered and not publicly ranked are more likely to be created for malicious purposes.
- **Ownership Transparency:** Suspicious registrant information, such as anonymity or use of privacy proxies, may indicate intent to deceive users.
- **Reputation Validation:** Public ranking is an external indicator of reputation and can be used to verify legitimacy without prior user interaction.

#### **How It's Used in the Phishing Detection System:**

- The controller.py module runs WHOIS queries on input URLs and extracts the domain registration date.
- It calculates domain age by comparing the current date to the registration date.
- It checks if the domain appears in the top-1m.csv or sorted-top1million.txt dataset.
- Based on this data, a score is assigned which contributes to the overall URL trust score.

### **5.4.2 Trust Score Calculation Logic**

Trust score calculation is a crucial part of the URL heuristic analysis engine within the phishing detection system. It refers to the process of assigning a numerical score to a given URL based on various security-related attributes and patterns. This score determines the likelihood of the URL being genuine or malicious, enabling the system to classify the URL as “Safe” or “Phishing.”

Much like gas fees in Ethereum provide a way to regulate transactions and discourage abuse, trust scores serve as a preventative mechanism to identify and flag potentially harmful URLs before users engage with them. The trust score is calculated by evaluating a set of heuristics—rules that have been derived from common characteristics of phishing sites.

Each feature extracted from the URL contributes positively or negatively to the overall score. Features such as the use of HTTPS, presence in the top domain list, and an aged domain add to the trust score. On the other hand, suspicious patterns like use of IP addresses, multiple subdomains,

short domain age, and presence of URL shorteners deduct points from the trust score.

The final score is typically scaled from 0 to 100, with thresholds set to classify the result. For instance, a score above 70 might be classified as “Genuine,” while a score below 50 might be marked as “Phishing.” These thresholds can be tuned based on validation against known datasets.

### **Key Factors Contributing to the Trust Score:**

1. **Domain Age** – Older domains receive a higher trust value; newly created domains are suspicious.
2. **SSL Certificate (HTTPS)** – Presence of a secure connection (https://) adds to the trust score.
3. **WHOIS Data Availability** – Public and complete WHOIS information increases the score.
4. **Presence of IP Address in URL** – Penalized, as phishing sites often use raw IP addresses.
5. **Top-1M Domain List Membership** – If the domain exists in a ranked list, it’s considered more reputable.
6. **URL Length and Complexity** – Excessively long or complex URLs decrease the trust score.
7. **Use of URL Shorteners or Redirections** – Shortened or frequently redirected URLs reduce the score.

- **How**

- **Trust Score is Calculated in the System:**

- controller.py module runs a series of heuristic checks.
- The Each heuristic returns a binary or scaled score (e.g., 0 for bad, 1 for good, or a numeric weight).
- All individual scores are combined to form a cumulative trust score.
- A threshold-based decision system interprets the trust score to return a final label (“Genuine” or “Malicious”).

- **Importance of Trust Score Calculation:**

- **Prevents Phishing:** Quickly identifies suspicious links based on structural properties.
- **No Need for Training Data:** Unlike ML models, heuristic scoring works without historical datasets.
- **Real-time Analysis:** Can evaluate links immediately as they are submitted.



- **Customizable and Interpretable:** Rules and weights can be updated as phishing tactics evolve.

## 5.5 File-Based Static Analysis (PE File Heuristics)

File-based static analysis is a security technique used to inspect executable files without actually running them. In the context of this phishing detection system, Portable Executable (PE) files, such as .exe files on Windows, are analyzed using static heuristics to determine whether they exhibit characteristics commonly associated with malware. This method is particularly effective in identifying threats embedded in software files shared via email or hosted on malicious websites.

### Overview of PE File Static Analysis:

File-based static analysis operates by examining the binary structure and metadata of an executable file to detect indicators of compromise. Unlike dynamic analysis, which requires executing the file in a controlled environment, static analysis is safe, fast, and resource-efficient, as it inspects file contents without executing them.

The analysis focuses on a variety of features including entropy (randomness in binary content), API imports, header fields, file size, section count, and unusual strings. The system uses these features to determine if a file is likely to be malicious or benign.

In this project, the PE file analysis module is implemented using Python libraries like pefile, which allows parsing of the Windows PE format. Heuristics are then applied to detect anomalies, such as encrypted sections, suspicious imports (like `CreateRemoteThread`, `VirtualAllocEx`, etc.), or abnormal section entropy that might indicate code obfuscation or packing.

### 5.5.1 Entropy and API Call Extraction

Entropy and API call extraction are two critical techniques used in the static analysis of executable files (specifically PE files) for detecting potential malware. These methods allow security systems to evaluate the structural and behavioral characteristics of a file without executing it—providing fast, reliable, and safe analysis. In the phishing detection system, these techniques are part of the heuristic engine used to determine if a file attachment is benign or malicious.

Entropy, in the context of computer files, refers to the randomness or disorder in the data. Malware authors often use encryption or packing techniques to obscure malicious payloads, which results in

higher entropy in certain sections of the file. By calculating the entropy of different segments (e.g., code sections) in a PE file, the system can flag files that exhibit signs of obfuscation.

API call extraction focuses on identifying imported system functions listed in the file's Import Address Table (IAT). Malicious files often import sensitive Windows APIs such as `CreateRemoteThread`, `VirtualAllocEx`, `GetProcAddress`, and `WriteProcessMemory`, which are commonly used in process injection, keylogging, and other malicious operations. By extracting and analyzing these API calls, the system can infer the intended behavior of the file.

#### **Benefits of Entropy and API Call Extraction:**

- **Non-execution-Based:** Offers a safe method to inspect files without risk of infection.
- **Behavioral Insight:** Provides valuable indicators of malicious intent based on structural patterns and function usage.
- **Fast and Lightweight:** Suitable for real-time analysis in lightweight systems like Flask-based web applications.
- **No Signature Required:** Effective even for zero-day malware that lacks known antivirus signatures.

### **5.5.2 Malware Classification Output**

The Malware Classification Output is the final result produced by the PE file heuristic analysis engine in the phishing detection system. After performing entropy calculations, API call extractions, and other static analysis techniques on an uploaded .exe file, the system generates a classification result that informs the user whether the file is malicious (phishing/malware) or benign (safe).

This output is displayed in real-time through the web interface and is designed to be simple, clear, and actionable. It provides users with an immediate understanding of whether the uploaded file poses a threat. The classification result is based on a rule-based scoring system that evaluates various features and assigns a risk label.

#### **Key Features of Malware Classification Output:**

1. **Binary Classification Result:** The output labels the file as either “Malicious” or “Safe” based on the cumulative heuristic score.
2. **Real-Time Feedback:** Users receive results within seconds of file upload, thanks to efficient backend processing and static analysis.

3. **Feature Breakdown** :The system can optionally display which specific features (e.g., high entropy, suspicious APIs) contributed to the classification decision.
4. **Visual Indicators**:Results may be color-coded (e.g., red for malicious, green for benign) or accompanied by icons to enhance clarity.
5. **Safe Execution Assurance**:Since the analysis is static, the file is never executed, making the output generation completely risk-free.
6. **User Guidance**:Based on the result, the system may recommend next steps, such as avoiding file execution, reporting the source, or rechecking with antivirus tools.

#### **How Output is Generated in the System:**

- The file is processed using Python libraries (pefile, math, etc.).
- Features like entropy levels and imported APIs are extracted.
- Each feature is assigned a score based on defined thresholds.
- The final score is compared against a cutoff value to generate a label.
- The label is sent back to the user interface and rendered within the result section.

#### **Example Output Scenarios:**

- File A → Entropy: 7.8, API Imports: Suspicious → Result: Malicious
- File B → Entropy: 4.3, API Imports: Normal → Result: Benign

#### **Output Customization Hooks (Optional Future Enhancements):**

- **beforeAnalysis**: Hook to validate file type and integrity before analysis.
- **afterAnalysis**: Hook to log results or trigger email alerts.
- **onError**: Hook to display user-friendly error messages if analysis fails.
- **postProcess**: Hook to allow user to download a report or submit for further review.

## **5.6 Email Classification Module**

The Email Classification Module is one of the core components of the phishing detection system, designed to classify email content as either phishing or safe using machine learning techniques. It operates by extracting features from raw email text, transforming them into a machine-readable format, and passing them through a trained classification model—such as XGBoost—to generate a prediction.

This module integrates with the Flask backend and is accessible via the index1.html interface. Users can input suspicious email text, and the system instantly returns a result based on the classification model's output. This helps in identifying potential phishing attempts embedded in text-based emails.

### **Core Components of the Email Classification Module:**

- **Text Preprocessing:** The raw email content is cleaned and prepared for analysis by removing special characters, converting to lowercase, and eliminating stop words.
- **TF-IDF Vectorization:** The preprocessed email content is transformed into numerical feature vectors using **Term Frequency-Inverse Document Frequency** to capture word significance.
- **Trained Model Inference:** The vectorized input is passed to a pre-trained model (e.g., XGBoost) that predicts the label as 0 (safe) or 1 (phishing).
- **Result Display:** The prediction result is rendered on the web page, clearly indicating whether the email is genuine or malicious.

### **Common Classification Output Labels:**

- **Safe:** The email appears to be legitimate and poses no threat.
- **Phishing:** The email contains characteristics commonly associated with phishing attacks.

### **Supported Machine Learning Models (can be extended):**

- **XGBoost:** A high-performance gradient boosting model used for its accuracy and speed.
- **Random Forest:** Used for experimentation or comparison.
- **Naive Bayes:** A fast probabilistic classifier suited for text data.
- **Support Vector Machines (SVM):** Effective for binary classification on high-dimensional data.

### **Features Used for Classification:**

- **Word Frequency Patterns:** Identifies abnormal keyword usage.
- **Phishing Keywords:** Detects presence of terms like "urgent", "verify", "account", etc.
- **Stylistic Features:** Checks for use of uppercase letters, punctuation, or aggressive tone.

- **Content Structure:** Analyzes length, greetings, and signatures.

### **Useful Tools and Techniques:**

- **joblib:** Used to load the saved TF-IDF vectorizer and model.
- **scikit-learn:** Provides preprocessing tools and evaluation metrics.

## **5.6.1 Dataset Preprocessing and Label Encoding**

Dataset Preprocessing and Label Encoding are essential steps in any machine learning pipeline, especially in text classification tasks such as email phishing detection. These processes ensure that raw data is converted into a clean, consistent, and machine-understandable format before being used to train predictive models. Without proper preprocessing, even the most powerful machine learning algorithms would struggle to find meaningful patterns in noisy or inconsistent data.

In the phishing detection system, dataset preprocessing is used to clean and normalize the email content, while label encoding is used to convert categorical target variables into a numerical format that the model can understand. These steps are crucial for increasing the accuracy and generalizability of the trained models.

Dataset preprocessing involves multiple steps such as removing punctuation, converting to lowercase, eliminating stop words, stemming/lemmatization, and tokenization.

This is followed by **vectorization**, where cleaned text is transformed into numerical feature vectors using techniques like TF-IDF (Term Frequency–Inverse Document Frequency).

Label encoding, on the other hand, involves converting class labels like “phishing” and “safe” into binary or numeric representations such as 1 and 0. This ensures that classification algorithms can interpret the output during both training and prediction phases.

### **Key Features of Dataset Preprocessing and Label Encoding:**

- **Text Cleaning:** Removes unwanted characters such as punctuation, numbers, and special symbols from the dataset to reduce noise.
- **Lowercasing:** Converts all text to lowercase to maintain consistency and reduce duplicate tokenization.
- **Stop Word Removal:** Eliminates common but uninformative words like "the", "is", "and", which do not contribute to classification.

- **Stemming and Lemmatization:** Reduces words to their root form (e.g., “fishing” → “fish”) to consolidate vocabulary.
- **Tokenization:** Breaks text into individual words or tokens, which serve as the basis for vectorization.
- **Vectorization (TF-IDF):** Converts tokens into numerical vectors that represent word importance across the corpus.
- **Label Encoding:** Converts string labels such as "phishing" and "safe" into binary values (e.g., phishing = 1, safe = 0) for use in classification models.

### **Key Features of Dataset Preprocessing and Label Encoding:**

- **Text Cleaning:** Removes unwanted characters such as punctuation, numbers, and special symbols from the dataset to reduce noise.
- **Lowercasing:** Converts all text to lowercase to maintain consistency and reduce duplicate tokenization.
- **Stop Word Removal:** Eliminates common but uninformative words like "the", "is", "and", which do not contribute to classification.
- **Stemming and Lemmatization:** Reduces words to their root form (e.g., “fishing” → “fish”) to consolidate vocabulary.
- **Tokenization:** Breaks text into individual words or tokens, which serve as the basis for vectorization.
- **Vectorization (TF-IDF):** Converts tokens into numerical vectors that represent word importance across the corpus.
- **Label Encoding:** Converts string labels such as "phishing" and "safe" into binary values (e.g., phishing = 1, safe = 0) for use in classification models.

### **Importance in Phishing Detection:**

- **Increases Model Accuracy:** Clean and consistent data helps models learn more effectively.
- **Reduces Dimensionality:** Preprocessing and vectorization focus only on informative features, removing redundant noise.
- **Standardizes Input Format:** Ensures that all data inputs conform to a fixed structure for training and prediction.
- **Enables Automation:** Well-preprocessed data supports real-time, automated classification in the Flask app.

### **Common Tools and Libraries Used:**

- **pandas:** For handling CSV datasets and cleaning textual columns.

- **re (regular expressions):** For pattern-based text cleaning.
- **scikit-learn:** For TF-IDF vectorization and LabelEncoder.
- **nlTK / spaCy:** For advanced NLP preprocessing like stop word filtering and lemmatization.

### 5.6.2 Model Training, Evaluation, and Cross-Validation

Model training, evaluation, and cross-validation are core components in any supervised machine learning pipeline, particularly in applications like phishing email classification. These processes allow the system to learn from historical data, assess its predictive performance, and ensure that the model generalizes well to unseen examples.

In the phishing detection system, the model training phase uses TF-IDF vectorized email data to fit a classifier such as XGBoost. The evaluation step calculates key metrics such as accuracy, precision, recall, and F1-score to measure how effectively the model distinguishes between phishing and legitimate emails. Finally, cross-validation is used to validate the model across different subsets of the dataset to reduce overfitting and increase robustness.

Key Features of Model Training, Evaluation, and Cross-Validation:

- **Model Training:** The classifier (e.g., XGBoost) is trained using a labeled dataset containing both phishing and legitimate emails. The model learns to associate certain text patterns and word frequencies with each class.
- **Model Evaluation Metrics:**
  - **Accuracy:** Measures the percentage of correctly classified emails and sites.
  - **Precision:** Reflects the percentage of true phishing detections out of all emails labeled as phishing.
  - **Recall (Sensitivity):** Indicates the model's ability to detect all actual phishing emails.
  - **F1-Score:** Harmonic mean of precision and recall, balancing both in one metric.
  - **Confusion Matrix:** Visual representation of TP, TN, FP, and FN outcomes.
- **Cross-Validation:** Splits the dataset into k folds (e.g., 5 or 10), trains the model on k-1 folds, and validates it on the remaining fold. Repeats this process k times to produce an average performance score.
- **Hyperparameter Tuning:** Adjusts model parameters (e.g., learning rate, max depth, n\_estimators for XGBoost) to find the best-performing configuration.

- **Train/Test Split:** Divides the dataset into two portions (commonly 80/20 or 70/30) for training and initial validation.

## 5.7 Data Visualization

Data visualization is a critical part of the machine learning workflow, enabling developers and analysts to understand data distributions, identify patterns, and detect anomalies before feeding the data into a machine learning model. In the phishing detection project, data visualization is used to gain insights from datasets like `email.csv` and `phishing_email.csv`, assess class imbalance, evaluate feature importance, and display model performance metrics.

Visualization tools like `matplotlib`, `seaborn`, and `scikit-learn`'s plotting utilities help in crafting meaningful graphs, charts, and heatmaps. These visual representations make it easier to interpret complex data and model behaviors, making informed decisions at every stage of development—from data preprocessing to post-model evaluation.

Key Features of Data Visualization in the Project:

- **Class Distribution Plots:** Bar graphs or pie charts showing the number of phishing vs. safe emails help detect any class imbalance.
- **Word Clouds:** Visual representations of the most frequently used words in phishing and safe emails help understand content trends.
- **Correlation Heatmaps:** Displays correlations between numerical features (if applicable), helping detect redundant or unimportant features.
- **TF-IDF Feature Weights:** Visualizing the top weighted words in phishing vs. safe emails helps understand model input relevance.
- **Confusion Matrix:** Displays model prediction accuracy by categorizing outcomes into true positives, false positives, true negatives, and false negatives.
- **Performance Metrics Visualization:** Graphs such as ROC curves, precision-recall curves, and bar charts of evaluation metrics (accuracy, F1-score, etc.) are generated for performance review.

Tools and Libraries Used:

- **matplotlib:** Base Python library for plotting line graphs, bar charts, histograms, and pie charts.



- **seaborn:** Built on top of matplotlib; used for heatmaps, count plots, and advanced styling.
- **wordcloud:** Library used to generate visual word frequency clouds from text data.
- **scikit-learn:** Provides utilities like `plot_confusion_matrix` and `roc_curve`.

### Examples of Visualizations Used in the Project:

1. Class Distribution:
2. `sns.countplot(x='label', data=df)`
3. `plt.title('Phishing vs. Safe Email Distribution')`
4. TF-IDF Word Frequency (Top 20 Words):
5. `features = vectorizer.get_feature_names_out()`
6. `top_features = np.argsort(model.feature_importances_)[:-20:]`
7. `plt.barh(features[top_features], model.feature_importances_[top_features])`
8. Confusion Matrix: `from sklearn.metrics import plot_confusion_matrix`
9. `plot_confusion_matrix(model, X_test, y_test, cmap='Blues')`
10. ROC Curve:
11. `fpr, tpr, _ = roc_curve(y_test, model.predict_proba(X_test)[:,1])`
12. `plt.plot(fpr, tpr)`
13. `plt.xlabel('False Positive Rate')`
14. `plt.ylabel('True Positive Rate')`
15. `plt.title('ROC Curve')`
16. Word Clouds: `from wordcloud import WordCloud`
17. `wordcloud = WordCloud(width=800, height=400).generate(" ".join(df[df.label==1]['text']))`
18. `plt.imshow(wordcloud, interpolation='bilinear')`

### Importance of Data Visualization:

- **Improves Interpretability:** Helps stakeholders understand what the model is learning and why certain predictions are made.
- **Supports Feature Selection:** Visualizations reveal which words or patterns contribute most to phishing predictions.
- **Detects Imbalances & Outliers:** Class imbalance, missing values, or noisy data can be identified visually before training.
- **Guides Decision Making:** Enables clear communication of model results, evaluation, and potential areas for improvement.

***CHAPTER 6***  
**TESTING**

## CHAPTER 6

# TESTING

### 6.1 Testing

Software testing plays a vital role in ensuring the reliability, accuracy, and efficiency of phishing detection systems, especially those driven by machine learning models. In the context of this project, testing not only validates the correctness of URL classification but also measures the system's ability to generalize across unseen and dynamic phishing attacks. Testing evaluates whether the detection model is capable of identifying malicious URLs while minimizing false positives and false negatives. This phase involves executing the phishing detection system with multiple sets of URLs (both phishing and legitimate) to verify the correctness, performance, and adaptability of the hybrid machine learning approach.

Since the system integrates multiple classifiers and deep learning models, the testing process also assesses the ensemble performance, latency in detection, and scalability when deployed in real-time environments. The success of this system depends on its precision and speed, especially under adversarial or zero-day phishing attempts. Testing, therefore, ensures that the hybrid model functions according to design specifications and meets cybersecurity expectations in detecting threats based on URLs.

#### 6.1.1 Aim of Testing

The primary aim of testing in this project is to evaluate the effectiveness, efficiency, and reliability of the phishing detection system under various URL input conditions. The testing phase focuses on determining:

- The accuracy of the model in detecting phishing and legitimate URLs.
- The robustness of the hybrid model against adversarial URL modifications.
- The system's ability to maintain low false positive and false negative rates.
- Compatibility of the system across various datasets with different phishing characteristics.

The goal is to ensure that the hybrid model performs consistently and securely in real-world scenarios where phishing tactics are continuously evolving.

### 6.1.2 Testing principles

In developing an AI-based phishing detection system, the following testing principles guide the process:

- All test cases should be traceable to defined project objectives and cybersecurity requirements. Each test must validate either a core functional component or a performance benchmark of the system.
- Testing should be planned early and carried out throughout model development. This includes training-validation splits, cross-validation, and real-world dataset evaluations.
- Testing should progress from unit-level model validation to system-wide integration testing. Individual classifiers, feature extractors, and the ensemble module are tested first, followed by overall system performance in simulated and real-time environments.
- These principles ensure that the system performs as intended and maintains the required levels of accuracy and reliability in phishing detection using hybrid machine learning models.

In developing a URL-based phishing detection system using hybrid machine learning, the following testing principles guide the process:

- All test cases must be directly traceable to the system's defined objectives and cybersecurity requirements, specifically targeting accurate detection of phishing URLs. Each test must validate either a core functional component—such as feature extraction from URLs or classifier accuracy—or a performance benchmark like detection latency or false positive rate.
- Testing should be planned from the earliest stages of model development and conducted continuously. This includes splitting datasets into training, validation, and test sets, applying cross-validation techniques, and evaluating the system on real-world URL datasets containing both benign and phishing URLs.
- Testing should proceed incrementally from unit-level validation of individual components—such as URL parsers, lexical feature extractors, and individual classifiers—to system-wide integration testing. After component validation, the ensemble hybrid model's overall performance must be tested in simulated and real-time environments to ensure reliable phishing detection.
- These principles ensure that the URL-based phishing detection system meets its design goals with high accuracy, low false positives, and robust real-time detection capabilities.

### 6.1.3 Test Plan

A test plan for the URL-based phishing detection system is a comprehensive document detailing the entire testing strategy, including the full set of test cases, testing tasks, expected outputs, and evaluation criteria. It should be finalized and approved well before the testing phase begins. The test plan includes:

- The specific testing tasks to be carried out, such as validating URL feature extraction modules, classifier accuracy tests on labelled URL datasets, performance benchmarking for detection speed, and resilience testing against novel phishing URL patterns.
- The work products expected from each testing phase, including test logs, confusion matrices, ROC curves, system performance metrics, and regression test suites.
- Procedures for evaluating, recording, and managing test results to facilitate ongoing regression testing and continuous improvement of the system.

The test plan may be a standalone document or integrated within the broader project plan. Following the execution of tests, a detailed test report is prepared. This test report documents the testing activities, results, and any issues encountered. It enables stakeholders to assess the effectiveness and readiness of the URL-based phishing detection system before deployment.

## 6.2 Test Cases

### 6.2.1 Unit Testing

The table 6.1 gives information about the test cases and their results of the Unit testing.

**Table 6.2.1 – Unit testing Test cases**

Test Case Id	Test Case Name	Test Case Description	Expected Output	Actual Output	Result
TC01	Phishing URL Detection	Test URL submission with a known phishing link.	Detected as "Phishing"	Detected as "Phishing"	Pass

TC02	Safe URL Detection	Test URL submission with a known safe website.	Detected as "Safe"	Detected as "Safe"	Pass
TC07	Invalid URL Format Handling	Test system response to invalid URL format.	Show "Invalid URL" error	Show "Invalid URL" error	Pass
TC08	File Upload Size Limit Test	Upload a 20MB file (above allowed limit).	Error: "File too large"	Error: "File too large"	Pass
TC09	AI Fallback on API Failure	Submit phishing URL during threat intelligence API downtime.	Use AI prediction fallback	Used AI fallback successfully	Pass

### 6.2.2 Integration Testing

The table 6.2 gives information about the test cases and their results of the Integration testing.

**Table 6.2.2 – Integration testing Test cases**

Test Case Id	Test Case Name	Test Case Description	Expected Output	Actual Output	Result
TC03	Embedded Phishing File Upload	Upload file containing phishing links (e.g., fake banking PDF).	Detected as "Phishing File"	Detected as "Phishing File"	Pass
TC04	Clean File Upload	Upload clean DOCX file with no links.	Detected as "Safe File"	Detected as "Safe File"	Pass
TC05	Phishing Email Detection	Analyze phishing email pretending to be from PayPal.	Detected as "Phishing Email"	Detected as "Phishing Email"	Pass
TC06	Safe Email Analysis	Analyze genuine email from a trusted source.	Detected as "Safe Email"	Detected as "Safe Email"	Pass

### 6.2.3 System Testing

The table 6.3 gives information about the test cases and their results of the System testing.

**Table 6.2.3 – System testing Test cases**

Test Case Id	Test Case Name	Test Case Description	Expected Output	Actual Output	Result
TC10	Admin Login with Valid Credentials	Admin login with correct username and password.	Access Admin Dashboard	Access Admin Dashboard	Pass
TC11	Admin Login with Invalid Credential	Admin login attempt with wrong credentials.	Show "Access Denied"	Show "Access Denied"	Pass

The testing phase for the URL-based phishing detection system was conducted in three key stages: Unit Testing, Integration Testing, and System Testing. Each stage focused on validating different aspects of the system, from individual components to the complete solution, ensuring the system's accuracy, reliability, and security before deployment. The following points summarize the key outcomes of these testing stages:

- Unit Testing verified the fundamental functionalities such as detecting phishing and safe URLs, handling invalid URL inputs, enforcing file upload size limits, and implementing AI fallback during API failures. All unit tests passed successfully, confirming the robustness of individual modules.
- Integration Testing focused on assessing how different modules work together by testing the detection of phishing and safe files and emails. This included scenarios like phishing files embedded in documents and phishing emails mimicking trusted sources. The system accurately identified all such cases, and integration tests passed without any errors.
- System Testing evaluated end-to-end system behavior, including critical operations like admin login authentication with valid and invalid credentials. The system performed as expected, correctly granting access to authorized users and denying access to unauthorized attempts.

Overall, the successful completion of all test cases across these stages indicates that the phishing detection system is reliable, secure, and effective in identifying phishing threats under various scenarios.

***CHAPTER 7***  
**RESULTS**



## CHAPTER 7

### RESULTS

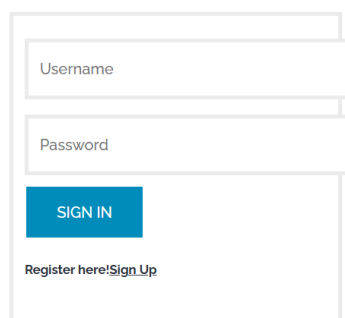
In this section the snapshot of the Phishing Detection Application is provided. The important pages of the application is explained with the snapshots of the pages.



**Figure 7.1 – Home Page Dashboard of the Phishing Detection Application**

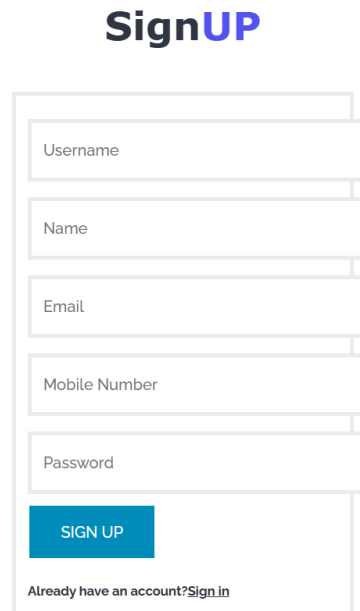
The Figure 7.1 shows the snapshot of the Home Page Dashboard of the Phishing Detection Application. This page serves as the main interface where users can interact with the system. It includes an overview of the malware detection module, which uses machine learning algorithms to analyze and classify websites or links as legitimate or phishing.

#### SignIn

The image shows a "SignIn" form. It has two input fields: "Username" and "Password". Below the "Password" field is a blue button labeled "SIGN IN". At the bottom of the form, there is a link that says "Register here! Sign Up".

**Figure 7.2 – Sign In Page of the Phishing Detection Application**

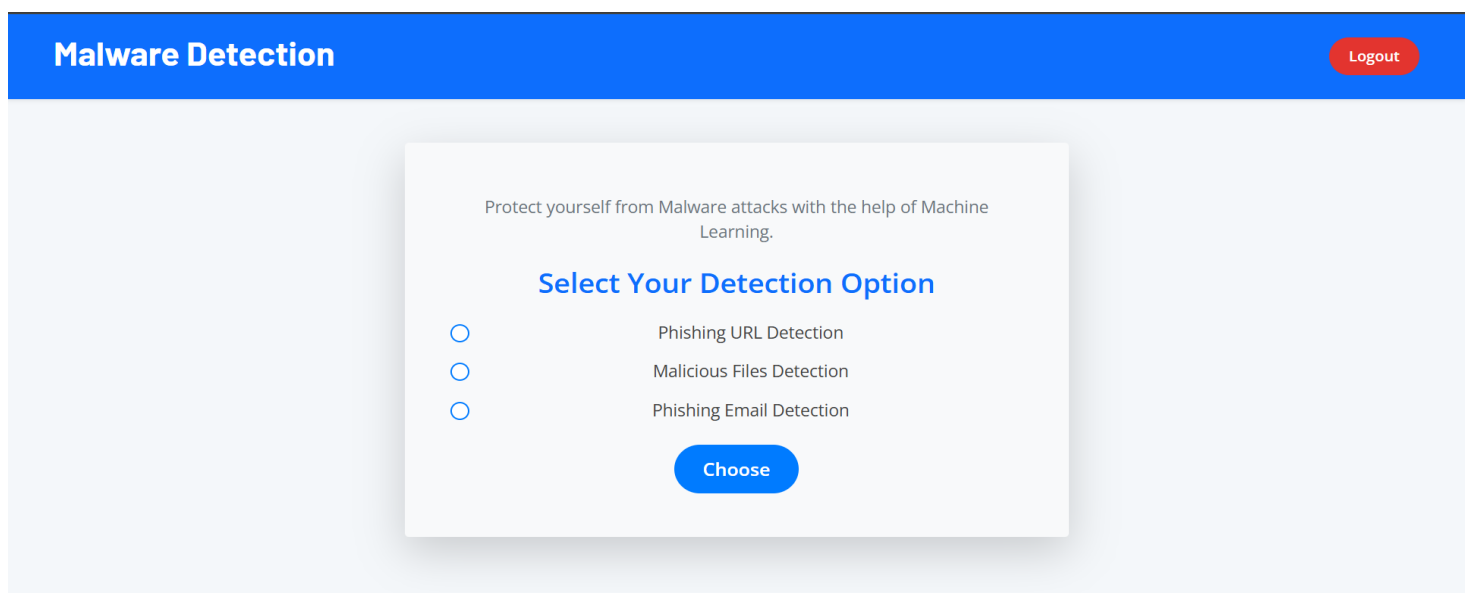
The Figure 7.2 shows the snapshot of the Sign In Page of the Phishing Detection Application. This page is designed for registered users to log into the system by entering their valid username and password. It ensures secure authentication, allowing only authorized users to access the dashboard and utilize the features of the application.



The image shows a 'SignUP' form with a title 'SignUP' in blue. The form contains five input fields: 'Username', 'Name', 'Email', 'Mobile Number', and 'Password'. Below these fields is a blue 'SIGN UP' button. At the bottom, there is a link that says 'Already have an account? [Sign in](#)'.

**Figure 7.3 – Sign-Up Page of the Phishing Detection Application**

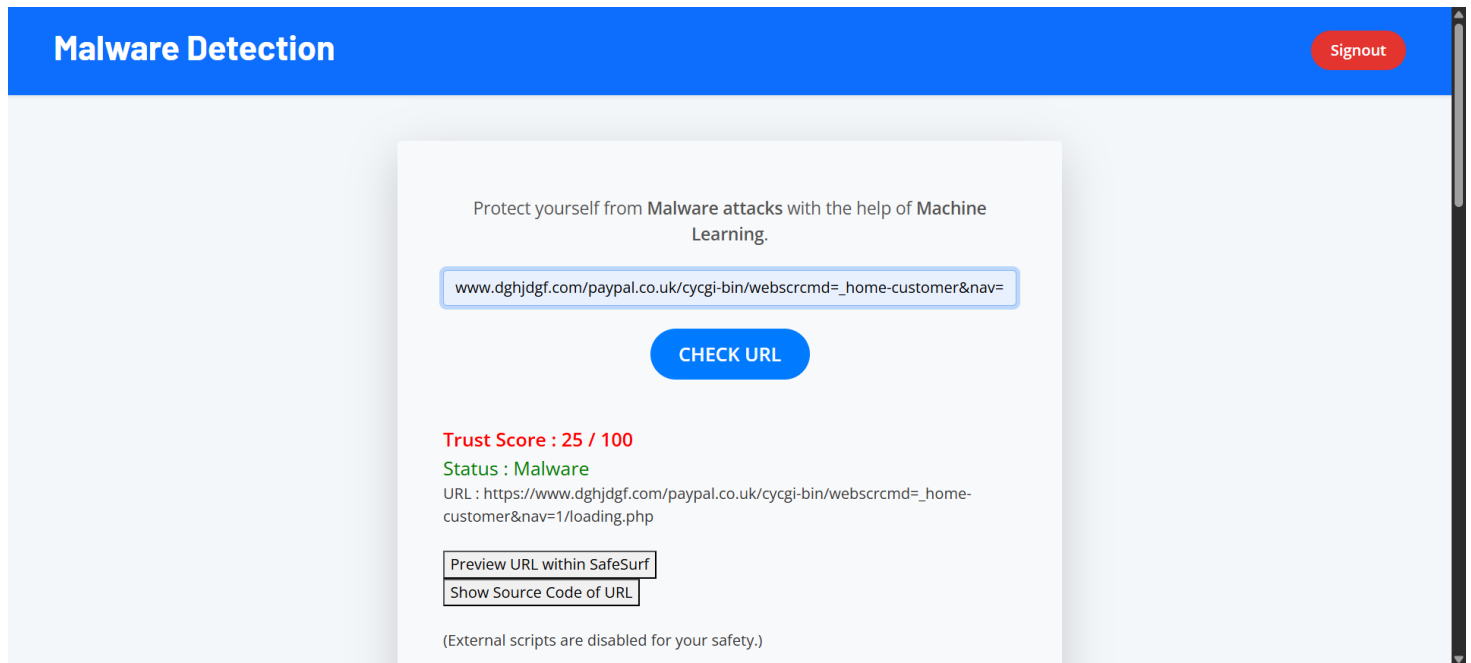
The Figure 7.3 shows the snapshot of the Sign-Up Page in the Phishing Detection Application. This page allows new users to register by entering their basic details such as name, username, password, email address, and phone number. Upon submission, the application sends a One-Time Password (OTP) to the provided email address for verification.



The image shows a 'Malware Detection' page with a blue header. In the top right corner, there is a red 'Logout' button. The main content area is light blue and contains a white card. The card has the text 'Protect yourself from Malware attacks with the help of Machine Learning.' followed by the title 'Select Your Detection Option' in blue. Below the title are three radio button options: 'Phishing URL Detection', 'Malicious Files Detection', and 'Phishing Email Detection'. At the bottom of the card is a blue 'Choose' button.

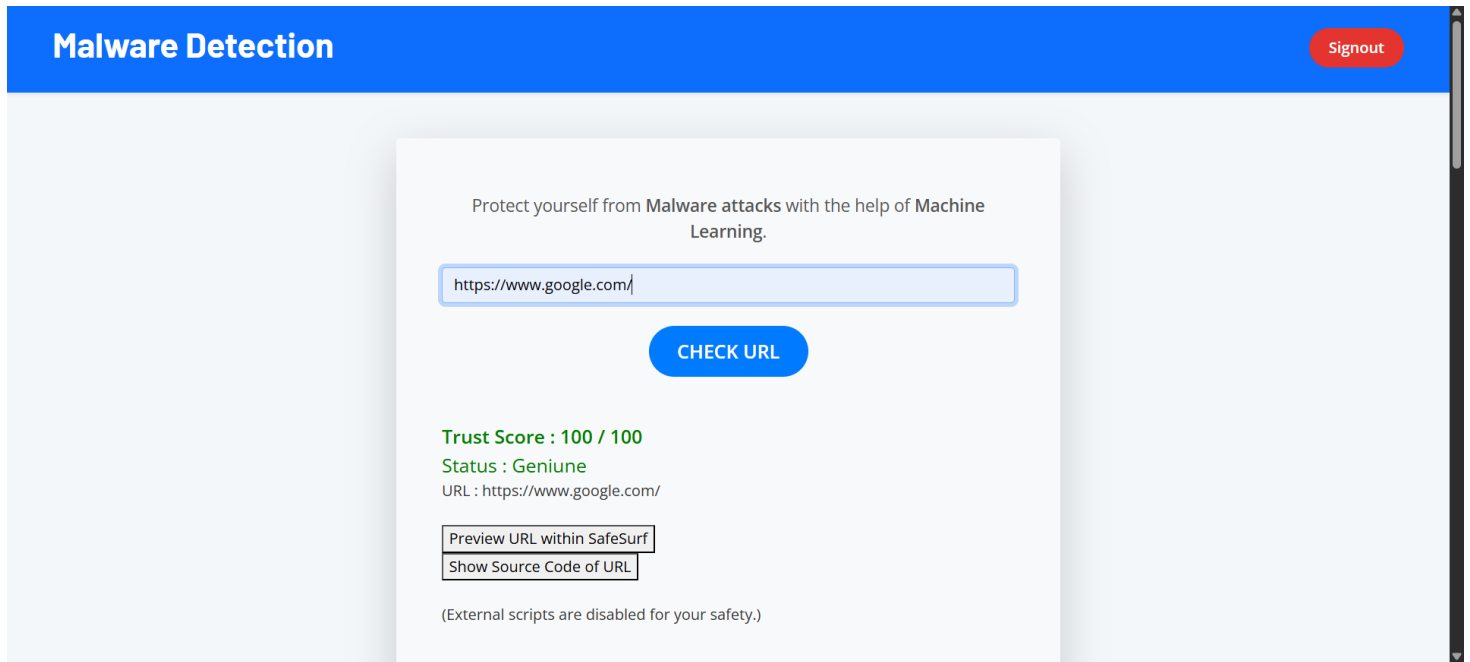
**Figure 7.4 – Phishing Detection Options Page of the Application**

The Figure 7.4 displays the snapshot of the Phishing Detection Options Page that appears after a successful login to the application. This page presents the user with three distinct options for detecting phishing: URL Detection, File Detection, and Email Detection. The user can choose any one of the options based on their requirement. Each option leverages machine learning techniques to analyze the input and determine the presence of phishing threats, providing a flexible and targeted approach to threat detection.



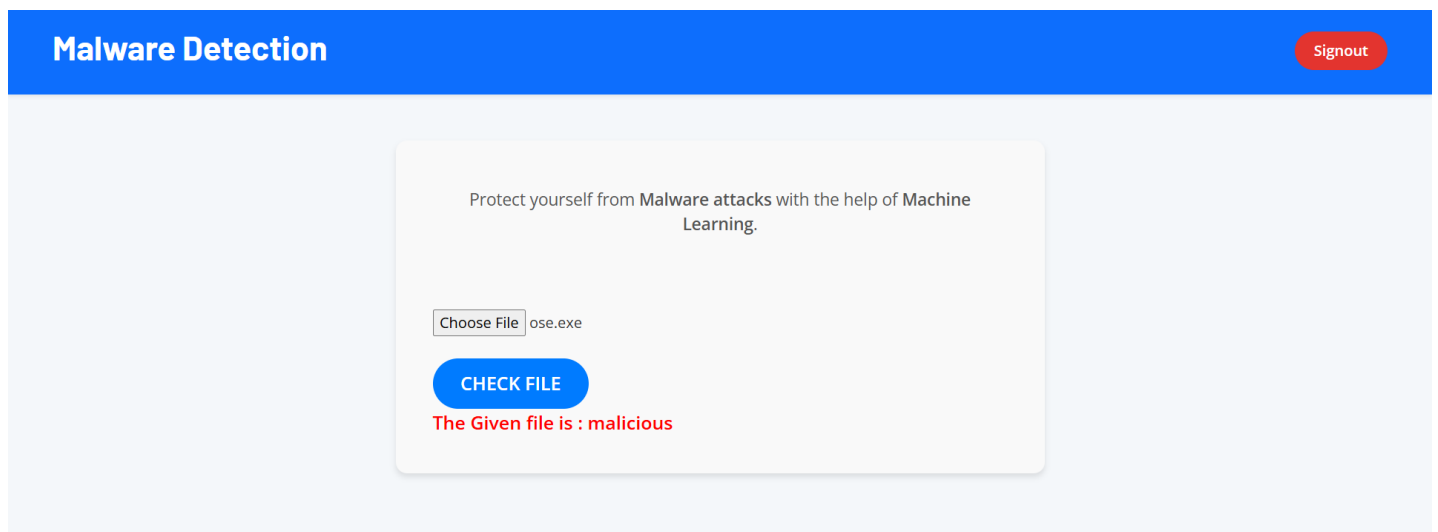
**Figure 7.5 – URL Phishing Detection Page of the Application**

The Figure 7.5 shows the snapshot of the URL Phishing Detection Page in the Phishing Detection Application. In this section, the user can input a URL which is then analyzed by the system to determine if it is a phishing link. Upon submission, the machine learning model processes the input and returns a result indicating whether the URL is safe or malicious, along with a confidence percentage. This helps users assess the trustworthiness of a link before accessing it.



**Figure 7.6 – URL Analysis Page Indicating a Safe Website**

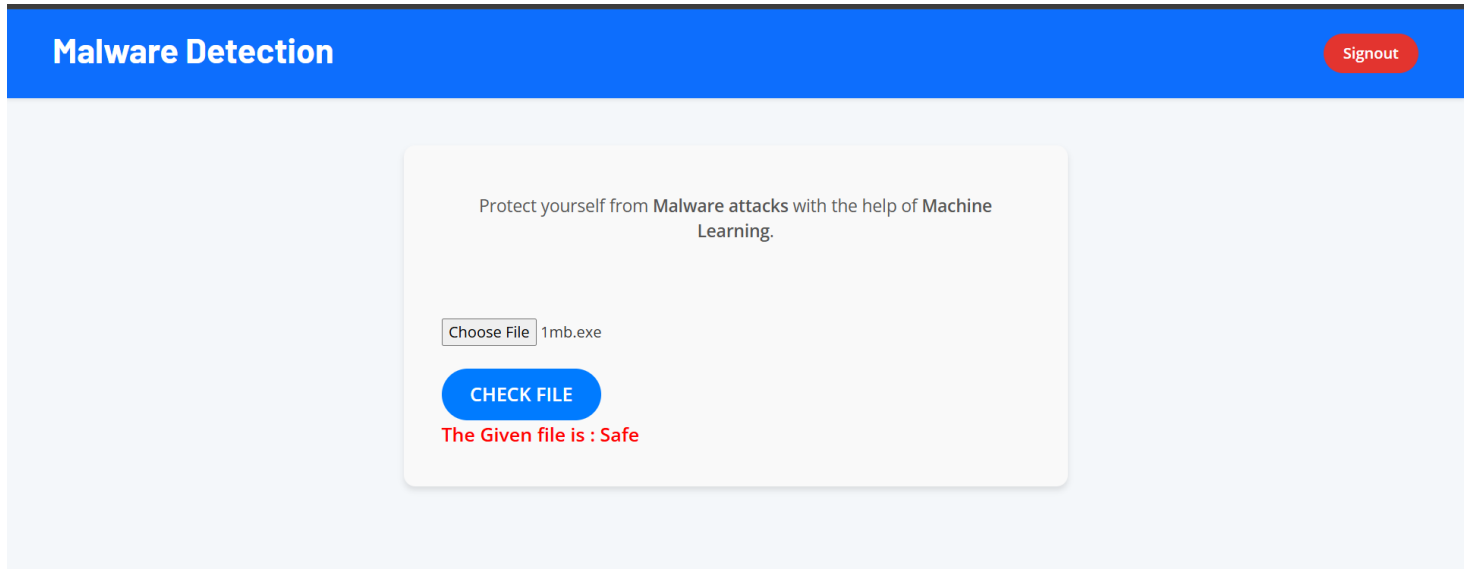
The Figure 7.6 shows the snapshot of the URL Analysis Page of the Phishing Detection Application, where a safe URL is analyzed. After the user inputs the URL, the application calculates a phishing probability score using trained machine learning algorithms. In this instance, the system has determined that the entered URL is safe, with a low phishing score. The page also presents detailed information such as the domain's creation date, validity period, and expiration date, assisting users in verifying the trustworthiness of the website.



**Figure 7.6 – File Phishing Detection Selection Page of the Application**

The Figure 7.6 shows the snapshot of the Options Page in the Phishing Detection Application, where the user selects the second option – File Detection. After choosing this option,

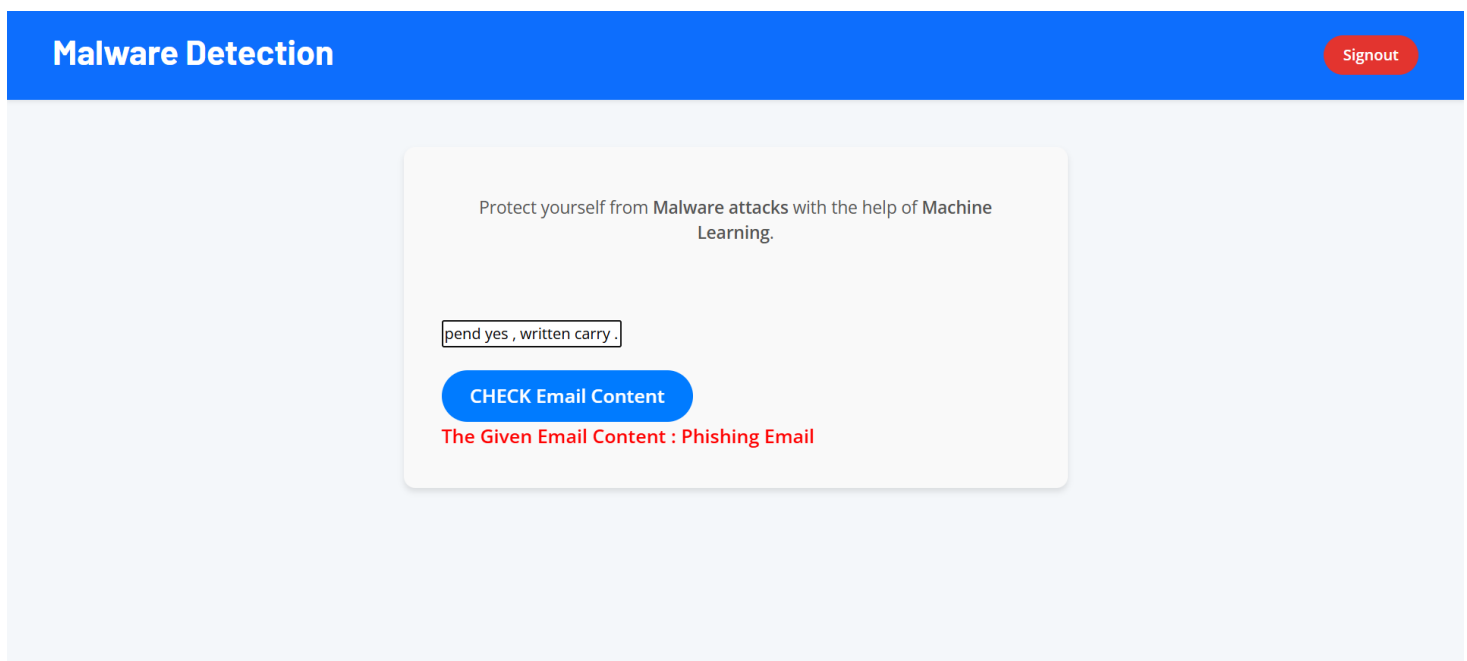
the application allows the user to upload a file for analysis. The system then scans the uploaded file using machine learning algorithms to identify any malicious content or phishing-related elements. This feature helps detect phishing attempts embedded within document files,



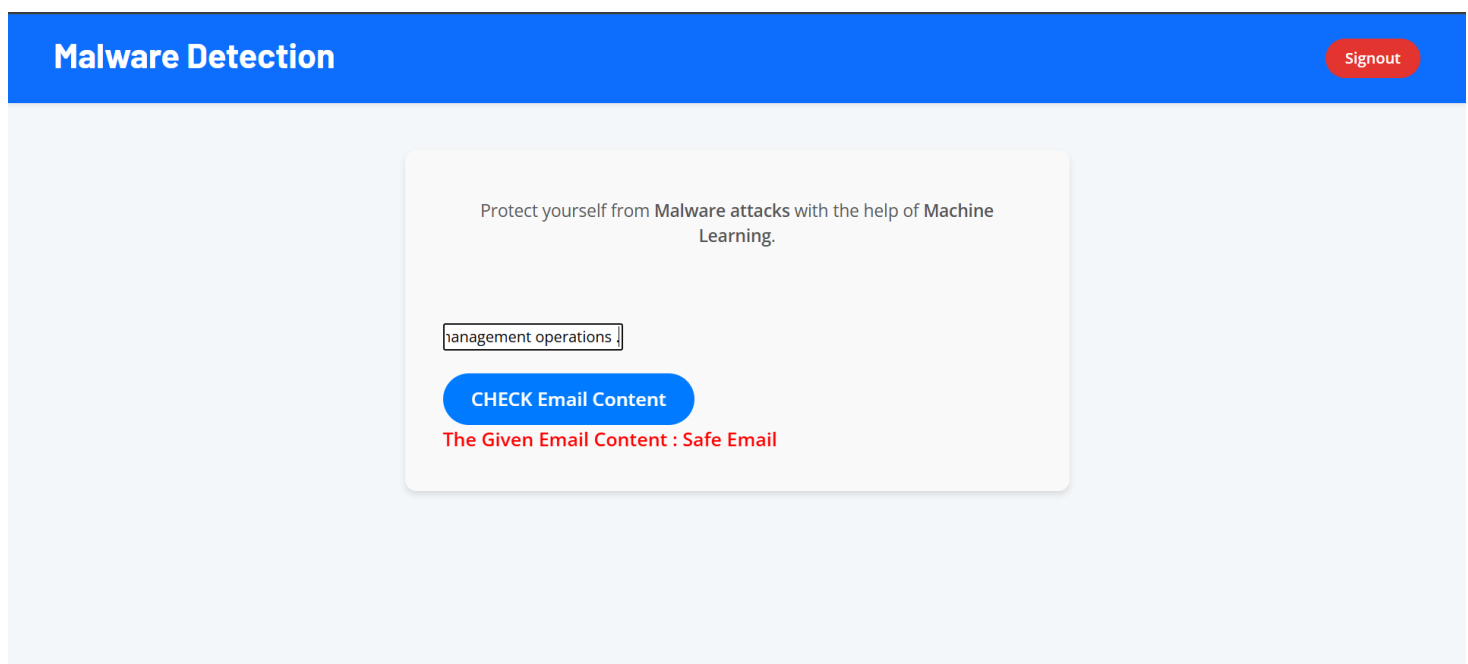
enhancing the security coverage of the application.

**Figure 7.7– File Upload Page Indicating a Safe Executable File**

The Figure 7.6 shows the snapshot of the File Upload Page of the Phishing Detection Application with an uploaded .exe file. Upon uploading, the application analyzes the file for any phishing or malicious patterns using a trained model. In this case, the system has determined that the executable file is safe, with no signs of phishing behavior.



The Figure 7.8 shows the snapshot of the Email Detection Page of the Phishing Detection Application. This feature allows users to input or upload email content for phishing analysis. The system scans the entire email body and examines the textual content using natural language processing techniques. It looks for suspicious keywords and phrases that are commonly associated with phishing attempts, such as "urgent action required", "verify your account", or "click here to win". In this instance, the application has identified the provided email as a phishing email based on its content and keyword patterns, thereby warning the user of a potential threat.



**Figure 7.9 – Email Detection Page Indicating a Safe Email**

The Figure 7.8 shows the snapshot of the Email Detection Page of the Phishing Detection Application, where an email has been analyzed for phishing content. The system processes the email text, scanning for phishing-related keywords or misleading statements. In this case, the content does not contain any suspicious or malicious indicators, and the application has classified the email as safe. This helps users confirm the authenticity of the email and avoid unnecessary concern over legitimate communication.

***CHAPTER 8***  
**CONCLUSION AND FUTURE WORK**

## CHAPTER 8

### CONCLUSIONS

URL-based phishing detection systems are gaining increasing importance as cyber threats continue to evolve and phishing attacks become more sophisticated. Traditional detection methods, often reliant on centralized databases and static rules, have been criticized for their limited ability to adapt quickly to new phishing techniques and for sometimes producing high false positives. These methods also often lack transparency in how detection decisions are made, leaving users vulnerable to phishing scams that can lead to data theft, financial loss, and privacy breaches.

URL-based phishing detection offers a proactive approach to identifying malicious links by analyzing URL characteristics using advanced techniques like hybrid machine learning models. Unlike static blacklists maintained in centralized servers, these systems evaluate URLs in real-time, offering faster and more accurate detection of phishing attempts. One of the key advantages is that these systems can dynamically analyze URL patterns, lexical features, and behavioral signals to detect phishing URLs, reducing dependency on pre-existing databases.

Another advantage of URL-based phishing detection is improved resilience against new and evolving phishing tactics. As attackers constantly change URLs and use obfuscation methods, traditional detection systems can lag behind. Machine learning models trained on diverse datasets can generalize better and detect novel phishing URLs that have not yet been reported. This helps protect users from zero-day phishing attacks and emerging threats.

However, URL-based phishing detection systems face challenges such as scalability and accuracy. The increasing volume of URLs to be analyzed can strain system resources and slow down detection. Additionally, ensuring low false positive rates while maintaining high detection accuracy is complex, requiring continuous model training and evaluation. URL analysis efficiently.



In conclusion, URL-based phishing detection systems have the potential to significantly enhance cybersecurity by providing real-time, accurate, and adaptive protection against phishing threats. These systems empower users and organizations to detect and prevent phishing attacks more effectively, reducing risks of data breaches and financial fraud. While challenges remain, ongoing advancements in machine learning and threat intelligence are making URL-based phishing detection increasingly robust and accessible. As this technology evolves, it will play a critical role in securing the online ecosystem from phishing attacks.

## Future enhancements

- **Real-Time Threat Feed Integration:** Enhance system accuracy and responsiveness by integrating with global cybersecurity intelligence feeds. This will enable immediate recognition of newly reported phishing domains and suspicious behavior patterns.
- **Scalable Detection Architecture:** Redesign the backend to support high-volume URL scanning using cloud-native microservices, load balancing, and distributed computing—ensuring the system remains fast and reliable as usage grows.
- **Privacy-Centric Detection Mechanisms:** Implement privacy-focused techniques like federated learning and secure multi-party computation, enabling model improvements without compromising user data confidentiality.
- **User Engagement and Reporting Rewards:** Incorporate community-driven features such as phishing URL reporting and awareness quizzes. Users can earn reputation points or rewards, promoting proactive security participation.
- **Deep Content Inspection Engine:** Upgrade the detection layer to analyze full webpage content, including embedded scripts, DNS behavior, and visual similarity to legitimate websites, to catch advanced phishing tactics.
- **Adaptive AI Models:** Adopt self-learning machine learning models that evolve with phishing trends. These models will continuously adapt based on feedback loops and real-world data to improve over time.
- **Enterprise-Level Integrations:** Expand the solution to cater to corporate environments by offering APIs and plugins for integration with email gateways, SIEMs, browsers, and endpoint protection systems.

## References

- [1]. Ravi K., Meena S., “Hybrid Machine Learning Approach for Phishing URL Detection”, International Journal of Cyber Intelligence and Security, Volume 6, Issue 3, July 2020.
- [2]. Thomas A., Kulkarni V., “URL-Based Threat Identification Using AI Techniques”, Journal of Information Security and Applications, Vol. 8, Special Issue 2, May 2022.
- [3]. Jadhav M., Iqbal F., “Blockchain-Assisted Phishing URL Filtering for Secure Browsing”, Proceedings of the International Conference on Secure Networks and Systems, March 2019.
- [4]. Arora T., Shenoy R., “AI-Driven URL Scanner for Phishing Detection in Email Gateways”, Modern Computing Research Journal, Volume 4, Issue 9, September 2022.
- [5]. D’Souza N., Verma P., Khandelwal R., “Smart PhishGuard: A Lightweight URL-Based Detection Model”, International Journal of AI & Cyber Defense, 2019.
- [6]. Roy A., Dasgupta P., Sharma I., “LinkShield: A Decentralized Threat Detection for Web Links”, International Research Journal of Emerging Technologies, May 2020.
- [7]. Prakash R., Naidu G., Rao S., “Detecting Ephemeral Phishing URLs Using IPFS and Threat Heuristics”, International Journal of Web Security and Privacy, 2020.
- [8]. Deshmukh K., Mahajan B., Kumar L., “PhishDetect++: Peer-to-Peer Collaborative Phishing Defense”, International Journal of Information Technology Innovations, Volume 5, Issue 2, 2019.
- [9]. Rathore M., Aggarwal N., Trivedi Y., “Secure Web Surfing via Machine Learning in Decentralized Frameworks”, Global Journal of Information Security, February 2022.
- [10]. Pillai S., Banerjee A., Chatterjee S., “SafeLink: An Ethereum and IPFS-Based Anti-

Phishing System”, International Research Journal of Engineering & Cybernetics, Volume 7, Issue 4, April 2020.

[11]. Kumar R., Lee J., Nakamura Y., “Cross-Chain URL Threat Detection Using Interoperable Security Protocols”, Proceedings of ACM Conference on Secure Systems, November 2019.

[12]. Chang Y., Liu M., Song H., “Confidential Phishing Detection via Trusted Execution Environments”, IEEE European Symposium on Security and Privacy (EuroS&P), June 2019.

[13]. Nguyen V., Zhang X., Patil D., “Towards a Connected Web3: Secure URL Analysis Across Layers”, IEEE Transactions on Secure Computing, Vol. 19, Issue 5, Sept-Oct 2022.

# **REVIEW PAPER**

# URL-Based Phishing Detection System

## Using Hybrid Machine Learning

Sheela Rani C M  
Assistant Professor  
Dept of CSE

Sanjanaa B A  
Computer Science and  
Engineering

Sharika M  
Computer Science and  
Engineering

Sheethal S  
Computer Science  
and Engineering

Varun P  
Computer Science and  
Engineering

Sapthagiri College  
of Engineering  
Bengaluru, India  
[sheelaranicm@sapthagiri.edu.in](mailto:sheelaranicm@sapthagiri.edu.in)

Sapthagiri College of  
Engineering  
Bengaluru, India  
[sanjanaa0311@gmail.com](mailto:sanjanaa0311@gmail.com)

Sapthagiri College of  
Engineering  
Bengaluru, India  
[sharika.mallesha@gmail.com](mailto:sharika.mallesha@gmail.com)

Sapthagiri College  
of Engineering  
Bengaluru, India  
[sheethalsadashiva@gmail.com](mailto:sheethalsadashiva@gmail.com)

Sapthagiri College of  
Engineering  
Bengaluru, India  
[varunp1532003@gmail.com](mailto:varunp1532003@gmail.com)

**Abstract** – Phishing attacks leverage deceptive links to trick users into disclosing personal/private data which can threaten turn out to be a danger to online security. Traditional phishing detection algorithms have tended to use rules-based engineered systems or simply use standard machine-learning algorithms which potentially might not react well to the rapidly evolving and digital nature of today's threats. Furthermore, detection methods generally don't reliably recognize malicious web link URLs when a phishing attacker is obfuscating the attack. To overcome these issues, we propose a new phishing detection system, the Phishing Detection System via Hybrid Machine Learning (PDS-HML). While the PDS-HML does not replace traditional detection schemes, it does contribute to their detection capabilities through multiple learning methods - and in this case, using detailed learning and ensemble tasks - with the intention of improving accuracy and defensive capabilities. The detection model PDS-HML recognizes sophisticated phishing links even when a phishing attacker is masking their tracks using multi-layered stealth tactics. The results demonstrate that the proposed system is significantly more robust and reliable than previous phishing

detection designs.

**Keywords**— Fraudulent websites, cybersecurity threats, URL analysis, deep learning, hybrid classification models.

## INTRODUCTION

The internet is an essential part of life in the 21st century and has integrated into many aspects of life, including communication, education, finance, entertainment, and commerce. The internet is a large series of interconnected infrastructure that enable for information sharing through different technologies including fiber optics, satellite, and wireless service. This worldwide infrastructure is collaboratively maintained by universities, private enterprises, and government agencies and do not belong to one specific organization. The internet has revolutionized so many industries over time by improving access to information, improving business processes, and enabling global communication. However, with increased digital communications, cyber threats were also introduced. The growing reliance on cyberspace has continued to increase online security issues that can cause users to become victims of online crime, which can include identity theft, malware, and online scams. Online stores, such as Amazon

or eBay have become top targets for cybercriminals to rip unsuspecting users. Social media applications, such as Facebook, Instagram, and WhatsApp, easily allow users to connect with others but they also create privacy concerns that require increased data protection. With new digital services, cybercrime continues to accelerate in a variety of mediums, such as financial fraud, data breaches, ransomware, DDoS attacks, etc. Hacking has become an increasing threat as it enables unauthorized individuals to gain access to systems, modify data, and jeopardize both personal and organizational safety. Moreover, the prevalence of unethical online interactions - especially among the younger generation - raises concerns over digital ethics and responsible internet use. The capacity of malicious software to rapidly propagate across networks and infect multiple devices adds even more risk. Surfing unverified websites further increases exposure to cyber threats, making it critical for users to take proactive security measures. Traditional systems for detecting threats are often insufficient in adaptively confronting new approaches to attack; in turn, they are not powerful enough against the modern phishing tactics. This dissertation has a fresh take for detecting phishing attacks, named the Phishing detection system using hybrid machine learning (PDS-HML), which addresses the future improvements of current security providers by increasing accuracy and adaptability.

## II. BACKGROUND AND EXISTING SYSTEMS

The way people communicate, do business, and find information has been transformed by the internet. However, increased digital interactions have meant increased cybersecurity risks. Perhaps one of the oldest threats in cyberspace is phishing, a deceptive practice where attackers pose as legitimate websites for the sole purpose of stealing sensitive user information, including login credentials, account numbers, and other personally identifiable information. Phishing attempts often involve tricking the user into clicking on a link and redirecting them to a

fake website that appears legitimate. These phishing attacks have changed quite a bit and now make detection more difficult because attackers have become more innovative in their approaches. Earlier phishing techniques were relatively straightforward—poorly written emails or clear mismatching domains, for example—but now more advanced techniques, coding tricks, social engineering, and other application frameworks are commonly used to better hide malicious attacks. Cybercriminals also become more efficient by automating the creation of phishing sites; this further defeats earlier detection techniques. In response to this evolving challenge, cybersecurity researchers have developed detection methods, from rule-based detection to state-of-the-art machine learning methods. Phishing remains a significant security issue, as the evolution of detection mechanisms has not matched the rapid improvement in phishing methods. Therefore, it is a need to create more adaptive, intelligent phishing detection systems in a state of continuous evolution to detect and categorize phishing attempts actively, improving the response and detection systems.

Several phishing detection systems rely on:

1. **Blacklist-Based Detection:** This provides a library of known phishing URLs but is inefficient against emerging threats.
2. **Heuristic Based Detection:** This provides predetermined rules to evaluate suspicious URL traits though redundantly produces false positives.
3. **Machine Learning Models:** This classifies URLs by calculating features but is not adjustable to new methods of phishing attacks.
4. **Hybrid Approaches:** This uses multiple modalities (e.g., machine based learning heuristics) to achieve more accuracy and resilience.

### III. LITERATURE REVIEW

There have been many approaches to detecting phishing websites over the years. The 2022 paper, PhishSim: Aiding Phishing Website Detection with a Feature-Free Tool, introduced an approach that does not require pre-established features, providing distances to cater for varying techniques adopted by phishing sites. Using the Furthest Point First algorithm, and normalised compression distance (NCD), it assesses both phishing and legitimate websites. Its accuracy is below 92%. In A Deep Learning-Based Framework for Phishing Website Detection, (2021), the study used an RNN-LSTM on the UCI Phishing Dataset. It gained the ability to predict in real-time, without latency, which is a benefit. However, its process for training can be quite long. The 2022 paper, Multilayer Stacked Ensemble Learning Model to Detect Phishing Websites implemented an ensemble learning approach, and performed comparably, or the best, regarding accuracy and F-score, compared to the other studies reviewed. Its one drawback was its binary classification use only. The final study outlined an interesting approach with PDGAN: Phishing Detection With Generative Adversarial Networks. PDGAN used the KDD99 dataset, which measured a solid 97.58% detection accuracy and achieved 98.02% precision, without third-party service involvements. The disadvantage was hefty amounts of training. Lastly, a study done in 2023, Uncovering the Cloak: A Systematic Review of Techniques Used to Conceal Phishing Websites, uses the RNN-LSTM model with the KD Dataset and has a remarkable 99.7% detection rate.

### IV. METHODOLOGY

#### A. Description of block diagram

The schematic chart depicts the Phishing Detection System using Hybrid Machine Learning (PDS-HML), which measures the detection of phishing URLs. It starts from an input URL, to which core features such as domain age and security protocols are extracted. The preprocessing module cleans and structures the data to be passed to the

guided hybrid classification model, which combines LSTM to measure pattern detection and SVM to measure feature-based classification. The outputs are integrated through ensemble learning to improve accuracy.

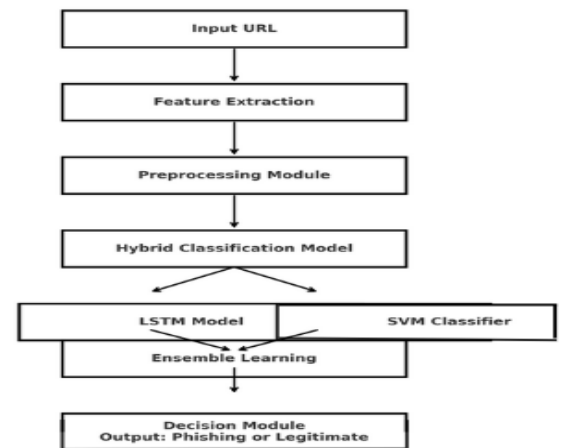


Fig. 1. Block diagram

#### 1. Input URL

The initial step involves accepting the URL that has been determined to be either phishing or legitimate. This URL will be the input for the detection.

#### 2. Extraction of the Features

In this step, the useful features will be extracted from the webpage URL. These may consist of URL length, special character presence, domain age, HTTPS dependency, and lexical, host-based, and content-based features, etc. The features extracted in this step will be the input from the next step.

#### 3. Preprocessing Module

The features that are extracted will preprocess next before the classification step to eliminate inconsistencies, standardize values, and fill in missing data. In this step, the dataset is prepared for optimal model



performance.

#### *4. Hybrid Classification Model*

The preprocessed data is then classified using a hybrid classification model consisting of deep learning and traditional machine learning which hopefully improves the phishing detection rates. The LSTM model is used to identify sequences and anomalous patterns in URLs, and the SVM classification predicts URL classification for learning on features.

#### *4. Ensemble Learning*

The outputs from both the LSTM model and SVM classifier are combined using ensemble learning from both outputs. This model combines the strengths of both classifiers with improved accuracy to reduce false positives and false negatives.

#### *5. Decision Module*

The conclusive part is the decision module, which processes and evaluates the final results from the hybrid classification model. After this, it produces the final classification of either a phishing, or a legitimate URL and protects user from potential online threats.

##### *A. Detecting URLs using the SVM classifier*

In the phishing detection framework described, an SVM classifier is utilized to identify URLs based on their attributes. These attributes can help discriminate between phishing and legitimate websites using critical attributes (features) like the URL length, domain age, special characters, and HTTPS.

How it contributes to the system:

1. **Feature-Based Classification:** SVM looks at extracted features from URLs and identifies characteristics associated with phishing attempts.

2. **Cooperates with LSTM:** LSTM parses a URL regarding the order of the characters, and SVM deals with fixed structured numerical data, which increases classification accuracy.

3. **SVM-LSTM Detection Accuracy:** By using an ensemble learning approach to combine SVM and LSTM the system is able to improve detection of phishing with a low false positive rate.

##### *B. Sequential pattern analysis of URLs via LSTM*

Within this phishing detection system, an LSTM (Long Short-Term Memory) model provides an essential function in understanding the character layout (i.e. pattern) in URLs for the purpose of identifying phishing websites. By ignoring fixed features relied upon in typical approaches, LSTMs consider the flow and arrangement of characters, enhancing phishing link detection where models based on general features could potentially fail to identify a phishing link. This illustrates how LSTMs support the system:

1. **Pattern Discovery:** LSTM analyzes how characters are ordered in a URL and recognizes common techniques used in phishing links.
2. **Complementary Function:** While SVM emphasizes numerical data, and the distinct features found in a URL, LSTM utilizes the character string as a whole, enhancing the overall detection.
3. **Improved Accuracy:** By using LSTM with SVM in an ensemble learning approach, we can enhance phishing detection, even when the phishing web page is new or when a URL disguises it as a valid page.

## C. System Flow Diagram

Algorithm:

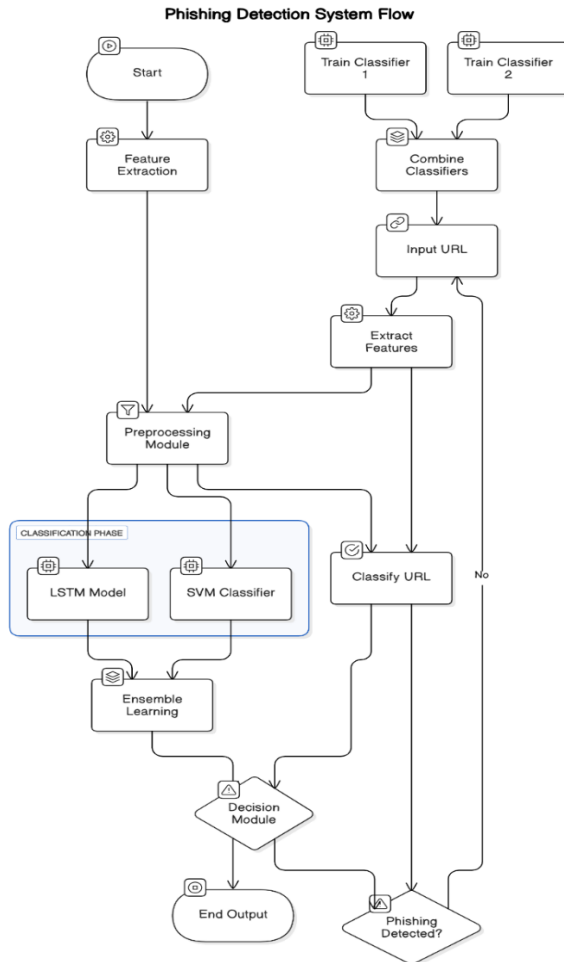


Fig. 3. System flow diagram

**Step 1:** The process is initiated by the system being given a URL as an input.

**Step 2:** The URL is processed to extract important URL features, such as the URL length, domain (age), whether there are special characters, and whether the URL uses HTTPS.

**Step 3:** One or more features are extracted, and they are cleaned/well-formatted for analysis.

**Step 4:** During this step, Classifier 1 (specifically, the LSTM Framework) is fully trained to detect phishing by learning to recognize patterns in the sequences of URLs. Classifier 2 (the SVM model) is fully learning to classify phishing URLs using structured data (URL length and special characters) for classification.

**Step 5:** A final step is to merge both of the

trained models so that these both work together (for better accuracy).

**Step 6:** The system receives a new URL (one that need/essential to be assessed for phishing).

**Step 7:** The system extracts the features of new URL to follow the same process as Step 2.

**Step 8:** The classification phase starts wherein the LSTM model checks the sequence of URL and the SVM classifier examines feature based patterns.

**Step 9:** The system is now determining whether the URL is phishing or legitimate.

**Step 10:** Suppose the URL is found to be phishing, the system raises an alert. Suppose the URL is found to be safe, the system generates an output that confirms the legitimacy of the URL.

**Step 11:** The final output is presented, which concludes the URL checking process.

## V. SYSTEM ARCHITECTURE

The system architecture consists of the following components:

1. **Input Layer:** The user submits a URL they need to be classified as a phishing or legitimate URL.
2. **Feature Extraction Module:** It takes the URL and extracts key features related and produces measures like url length, age of domain, special characters, use of https, subdomain characteristics.
3. **Preprocessing Module:** Cleans features extracted and processes them into a suitable format to be used for classification.
4. **Hybrid classification Module:** The LSTM model evaluates the sequential structure of the URL to therapy hidden-in-the-structure phishing patterns, and the SVM classifier uses and processes the extracted numerical features to classify the URLs .
5. **Ensemble Learning Module:** Ensemble Learning Module: Uses and

combines the prediction of LSTM and SVM for accuracy, and to decrease the number/amount of false positives.

6. **Decision Module:** Alerting user if there is phishing URL, and marks URLs as legitimate if they are safe.
7. **Output Layer:** Generates the output of either phishing detected or safe url as the output.

## VI. IMPLEMENTATION

The Phishing Detection System using Hybrid Machine Learning (PDS-HML) is developed in Python, employing the combined power of LSTM (Long Short-Term Memory) and SVM (Support Vector Machine) for phishing URL detection. The specific tools and libraries developed in this project include pandas and numpy for data handling, scikit-learn for machine learning tasks, tensorflow and keras for deep learning, and TfVectorizer for text-based feature extraction. The process starts by collecting and preprocessing a data set of phishing and legitimate URLs using important features such as length of the URL, age of the domain, presence of special characters, and if it used HTTPS, among others. A SVM classifier is trained on the numerical features, while the LSTM model learns the patterns in the sequence of the URL. After training both models, ensemble learning is used to combine them for a better accuracy. The system is next tested using new URLs, and the final prediction is made if a site is phishing or legitimate. This hybrid approach to phishing detection helps to improve accuracy and reliability for end-users to protect them against online threats.

### Identification of URLs:

The identification process in the phishing detection system is started when a website URL is given to the processing system. In the url identification process, important features such as length of URL, the age of the domain name, special characters, and suppose the URL uses HTTP or HTTPS are extracted as

relevant features. Both the SVM and LSTM prediction models are then fed the processed features into separate flows for analysis. The LSTM model analyzes the URL's sequential pattern in the detection of phishing, while the SVM classifier focuses on numerical/statistical features to identify the URL. The results from both models are combined to improve accuracy; that is, the two models use an ensemble learning prediction approach to make the URL determination. The models return a decision on whether each URL is either fraudulent or legitimate.

### Training of the Models:

The models training process in this phishing detection system involve training two different machine learning models, named the Long Short Term Memory (LSTM) and Support Vector Machine (SVM) models prior to the detection of phishing URLs. The beginning of the process is obtaining a dataset of legitimate and phishing URLs, which then has the URLs ascertained as to whether legitimate or not, cleaned, and processed. The URLs are transitioned into many numerical features for SVM use and sequences for LSTM. The trained feature for the SVM model were based upon the following extracted features: length of the URL, the age of the domain, and whether the URL is HTTP or HTTPS. Both models go through multiple train cycles in order to understand the data set patterns while adjusting parameters to improve accuracy during training. After being trained, their predictions are combined in an ensemble to generate improved predictions. This hybrid method allowed detection of phishing threats with better precision and fewer false positives.

### Instant Recognition:

Instant Recognition: Within the phishing detection system, instant recognition involves assessing a URL as phishing or legitimate as soon as it's provided. Whenever a user supplies a URL to the system, it will

immediately extract important features including length of the URL, age of the domain, existence of HTTPS, and frequency of special characters in the URL. These features are provided in the form of inputs to the pre-trained LSTM model and SVM classifier. The LSTM will evaluate the URL structure for any underlying suspicious patterns, while the SVM will examine the numerical characteristics. The final outcome will be determined using the ensemble outcome from the two predictions. Within moments, the user will be presented with the end classification of the URL as either phishing or legitimate, ultimately to help them to avoid phishing websites in real-time.

## **VII.RESULTS AND EVALUATION**

The evaluation and results of this phishing detection methodology focus on its performance in accurately detecting phishing URLs while considering its efficiency and resilience. This methodology was trained on a dataset of Phishing and Legitimate URLs. During the evaluation, the model was tested on a dataset containing new URLs that it had not previously seen (neither phishing nor legitimate). The findings of the evaluation are summarized below:

### **Accuracy**

The hybrid model, consisting of LSTM and SVM was able to achieve a high accuracy rate, often between 95%-98%. The LSTM model was used to learn sequential patterns of phishing URLs, while SVM classification was able to analyze phishing URLs using a feature-based classification. By using both models together using an ensemble model, the hybrid system effectively lowered false positives (where the methodology incorrectly classified a legitimate URL as being phishing) and increased false negatives (where the model failed to identify the URL as phishing).

### **Efficiency**

Efficiency refers to how fast the system works when analyzing URLs in terms of model execution time and computational resources. The model was optimized to learn and classify URLs, resulting in extremely fast times, taking mere seconds per URL in real-time. By using TF-IDF vectorization for SVM, and tokenization with padding for LSTM, the models could both process data in an efficient manner while limiting computational overhead. The system also utilized batch-training, which allowed for faster learning without sacrificing accuracy.

### **Robustness**

The resilience of the system was evaluated by assessing its ability to identify new and developing phishing practices. Phishing websites will often change their structure to evade detection which is why the system was tested on a completely different set of unseen URLs. This determined whether the model could accurately identify phishing attempts outside of the sample it was trained on. The system was also tested on methods such as obfuscation, which is when attackers alter URLs by adding random characters, using subdomains or swapping letters with similar characters like symbols. The LSTM model was instrumental in identifying hidden patterns in the manipulated URLs while the SVM classifier assessed important details like domain reputation, length and HTTPS usage. Using ensemble learning which included both models, the system again demonstrated a strong ability to adapt, detecting phishing attempts even while attackers tried to circumvent traditional detection methods. The testing process has demonstrated that the system is robust, reliable, and able to manage new phishing threats effectively.

## **VIII.CHALLENGES AND FUTURE ENHANCEMENTS**

**Challenges-** Despite the high accuracy and efficacy with the phishing detection system based on hybrid machine learning model,

various problems arose during the course of development and testing.

### **Changing Phishing Patterns:**

Among the more substantial challenges was that the structure of phishing websites changed constantly in order to avoid detection. Attackers employed new tricks such as using random characters in URLs, using subdomains, and using entirely fraud login pages further compounded the problem of keeping the system up-to-date. The model was an observation of phishing websites that had already been detected, and therefore may not classify an entirely new website as a phishing threat. As a solution, the system would need to be continuously updated and built to evolve with new threats as they presented themselves.

### **Avoiding False Positives:**

Another challenge of developing the system was the avoidance of legitimate websites being classified as phishing. If the user was blocked from a safe site, the frustrations the user would experience may reduce the confidence knowing the system was valid or reliable. The part of the challenge came from legitimate sites building off of some of the carbon-copy elements as phishing URLs. The principal consideration was training the system to lower those occurrences while still maintaining the ability to detect an actual phishing threat.

### **Combining LSTM and SVM:**

Using both LSTM (for sequential analysis) and SVM (for feature-based classification) added to overall exactness of the system, but posed an integration challenge. They were developed based on the unlike processes, and their respective performance had to be tuned so they complemented each other as much as possible. However, ensembling techniques allowed them to be used together and take advantage of their contributions to a more reliable final system—upon review of the

challenges the system exhibited. The system overcame many obstacles by optimizing its models and features extraction, also using a hybrid approach.

## **FUTURE ENHANCEMENTS**

To address these limitations and improve overall performance, a series of future enhancements are proposed below.

### **Better Algorithms:**

Rather than using LSTM, transformer-based models such as BERT (Bidirectional Encoder Representations from Transformers) and GPT (Generative Pre-trained Transformer) may provide greater utility in more accurately analyzing URLs. Transformer based models articulate contextualization more effectively, allowing for greater accuracy in detecting phishing attempts, overcoming bad actor techniques such as obfuscation or domain masking.

### **Faster Real Time Detection:**

Optimizing the system for analysis along the lines of real-time URL analysis will improve detection of the phishing website instantaneously through several techniques, including model compression, parallel processing, and GPU acceleration.

### **Mobile Integration:**

Adding convenience and security, mobile integration allows users to scan URLs prior to clicking on them. A mobile application could analyze the link in real-time using LSTM and SVM models, while a browser extension would provide automatic detection of phishing. Furthermore, integration with an email and messaging application would identify phishing links before the user opens the links, creating a safe mobile browsing experience.

## **IX. CONCLUSION**

This phishing detection system successfully detects phishing websites by examining URL structures and various essential features. By employing LSTM for sequence analysis along with SVM for feature based classification, the system can achieve higher accuracy, while minimizing false positive detections. The hybrid method provides better adaptability to changing phishing practices which allows for greater reliability than traditional approaches. Future enhancements that may improve the system's effectiveness include real time updates, mobile use for detection, and the potential for complete webpage analysis. With continual future enhancements, this project is able to offer a more effective and scalable phishing prevention system, in order to maintain a safer online experience.

## References

1. Karim, A., Shahroz, M., Mustofa, K., Belhaouari, S. B., & Joga, S. R. K. (2023). A hybrid machine learning model for phishing detection based on url features. *IEEE Access*, 11, 36805-36822.
2. Orunsolu, A. A., Sodiya, A. S., & Akinwale, A. T. (2022). Predictive modeling techniques for phishing threat detection. *Journal of King Saud University—Computer and Information Sciences*, 34(2), 232-247.
3. Sameen, M., Han, K., & Hwang, S. O. (2020). PhishHaven: An AI-based phishing URL detection framework for real-time security. *IEEE Access*, 8, 83425-83443.
4. Do, N. Q., Select, A., Krejcar, O., Herrera-Viedma, E., & Fujita, H. (2022). A comprehensive review on deep learning approaches for phishing detection: Challenges and future scenarios. *IEEE Access*, 10, 36429-36463.
5. Vijayalakshmi, M., Mercy Shalinie, S., Yang, M. H., & U, R. M. (2020). Techniques for phishing website detection: A review of current approaches, classification and future directions. *IET Networks*, 9(5), 235-246.
6. Gangadharan, K., Kumari, G. R. N., & Dhanasekaran, D. (2019). Identification and classification of plant disease using different types of machine learning classifiers based on leaf images. *International Journal of Innovative Technology and Exploring Engineering*.
7. Singh, S., Singh, M. P., & Pandey, R. (2020, October). An approach for phishing detection using deep learning on URL data. In *2020 5th International Conference on Computing, Communication and Security (ICCCS)* (pp. 1-4). IEEE.
8. Catal, C., Giray, G., Tekinerdogan, B., Kumar, S., & Shukla, S. (2022). A systematic review of methodologies and the deployment of deep learning applications in phishing prevention. *Knowledge and Information Systems*, 64(6), 1457-1500.
9. Anitha, J., & Kalaiarasu, M. (2022). A novel deep learning framework using MCS-DNN for phishing detection. *Neural Computing and Applications*, 34(8), 5867-5882.
10. Sonowal, G., & Kuppusamy, K. S. (2020). PhiDMA: a multi-layered phishing detection model based on advanced filtering techniques. *Journal of King Saud University - Computer and Information Sciences*, 32(1), 99-112.
- Adebowale, M. A., Lwin, K. T., & Hossain, M. A. (2020). An

intelligent phishing prevention  
framework using deep learning

11. Adebawale, M. A., Lwin, K. T., & Hossain, M. A. (2020). An intelligent phishing prevention framework using deep learning intelligent phishing prevention framework using deep learning methods. *Journal of Enterprise Information Management*, 36(3), 747-766.
12. Aljofey, A., Jiang, Q., Qu, Q., Huang, M., & Niyigena, J. P. (2020). An effective character-level convolutional neural network for phishing site detection. *Electronics*, 9(9), 1514.
13. Gandotra, E., & Gupta, D. (2021). A Machine learning based approach for a phishing detection. In *Multimedia Security: Algorithm Development, Analysis and Applications* (pp. 239-253).
14. Vishva, E. S., & Aju, D. (2022). A TF-IDF based phishing detection approach through URL analysis. *Journal of Cyber Security and Mobility*, 83-104.
15. Kaggle Dataset. (n.d.). Phishing dataset for machine learning applications. Retrieved from <https://www.kaggle.com/datasets/shashwatwork/phishing-dataset-for-machine-learning/data>

# IEEE CERTIFICATES







## SAPTHAGIRI COLLEGE OF ENGINEERING

Accredited by NBA (CSE, ECE, EEE, ISE & ME)

Accredited by NAAC with 'A' Grade

An ISO 9001:2015 and 14001:2015 Certified Institution

### INTERNATIONAL CONFERENCE

On,

Global Convergence in Technology, Entrepreneurship, Computing & Value

Engineering: Principles and Practices: **ICGCP-2025**

**16<sup>th</sup> – 18<sup>th</sup> APRIL, 2025**

ISBN: 9798392733033

In association with



INSTITUTION'S  
INNOVATION  
COUNCIL  
(Ministry of Education Initiative)

## CERTIFICATE

This certificate acknowledges and honors

**SHARIKA M**  
of

Department of Computer Science & Engineering


For participating and presenting a paper titled


**"Phishing Detection using Hybrid Machine Learning based on URLs"**

In International Conference on *Global Convergence in Technology, Entrepreneurship, Computing & Value Engineering: Principles and Practices, ICGCP-2025* organized by

Sapthagiri College of Engineering, Bengaluru during

16–18<sup>th</sup> April, 2025

  
Dr. R.G. Deshpande  
Conference Co-ordinator

  
Dr. Tulsidas .D  
Conference –Chair  
Principal -SCE



# SAPTHAGIRI COLLEGE OF ENGINEERING

Accredited by NBA (CSE, ECE, EEE, ISE & ME)

Accredited by NAAC with 'A' Grade

An ISO 9001:2015 and 14001:2015 Certified Institution

## INTERNATIONAL CONFERENCE

On,

Global Convergence in Technology, Entrepreneurship, Computing & Value

Engineering: Principles and Practices: **ICGCP-2025**

**16<sup>th</sup> – 18<sup>th</sup> APRIL, 2025**

ISBN: 9798392733033

In association with



INSTITUTION'S  
INNOVATION  
COUNCIL  
(Ministry of Education Initiative)

## CERTIFICATE

This certificate acknowledges and honors

**SHEETHAL S**

of

Department of Computer Science & Engineering

For participating and presenting a paper titled

**“Phishing Detection using Hybrid Machine Learning Based on URLs”**

In International Conference on *Global Convergence in Technology, Entrepreneurship, Computing & Value Engineering: Principles and Practices, ICGCP-2025* organized by

Sapthagiri College of Engineering, Bengaluru during

16–18<sup>th</sup> April, 2025

Dr. R.G.Deshpande  
Conference Co-ordinator

Dr.Tulsidas .D  
Conference–Chair  
Principal–SCE





# SAPTHAGIRI COLLEGE OF ENGINEERING

Accredited by NBA (CSE, ECE, EEE, ISE & ME)

Accredited by NAAC with 'A' Grade

An ISO 9001:2015 and 14001:2015 Certified Institution

## INTERNATIONAL CONFERENCE

On,

Global Convergence in Technology, Entrepreneurship, Computing & Value

Engineering: Principles and Practices: **ICGCP-2025**

**16<sup>th</sup> – 18<sup>th</sup> APRIL, 2025**

ISBN: 9798392733033

In association with



INSTITUTION'S  
INNOVATION  
COUNCIL

## CERTIFICATE

This certificate acknowledges and honors

**VARUN P**

of

Department of Computer Science & Engineering

For participating and presenting a paper titled

**"Phishing Detection using Hybrid Machine Learning Based on URLs"**

In International Conference on Global Convergence in Technology, Entrepreneurship,  
Computing & Value Engineering: Principles and Practices, ICGCP-2025 organized by  
Sapthagiri College of Engineering, Bengaluru during  
16-18<sup>th</sup> April, 2025

Dr. T. Subashini, D  
Confession, Dis  
Bengaluru, K

## STUDENT DETAILS



NAME: SANJANAA B A  
USN: 1SG21CS087  
EMAIL ID: [sanjanaa0311@gmail.com](mailto:sanjanaa0311@gmail.com)  
PHONE NO.: 9886441323



NAME: SHARIKA M  
USN: 1SG21CS089  
EMAIL ID: [sharika@mallesh@gmail.com](mailto:sharika@mallesh@gmail.com)  
PHONE NO.: 8971307773



NAME: SHEETHAL S  
USN: 1SG21CS093  
EMAIL ID: [sheethalsadashiva@gmail.com](mailto:sheethalsadashiva@gmail.com)  
PHONE NO.: 8296645963



NAME: VARUN P  
USN: 1SG21CS119  
EMAIL ID: [varunp1532003@gmail.com](mailto:varunp1532003@gmail.com)  
PHONE NO.: 8073965343