PHISHING DETECTION TOOL

A project report submitted in partial fulfilment of the requirement for the award of the degree of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE AND ENGINEERING

By

K. Tejasri (22NM1A4619)
P. Lavanya (22NM1A4637)
P. Neelima (23NM5A4604)
K. Jyothi (22NM1A4623)

Under the guidance of

Dr.D.Manendra Sai
Professor

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

VIGNAN'S INSTITUTE OF ENGINEERING FOR WOMEN (A)
(Approved by AICTE, New Delhi & Affiliated to JNTU- GV, Vizianagaram) Estd. 2008
Kapujaggarajupeta, VSEZ (Post), Visakhapatnam – 530049.

2026

VIGNAN'S INSTITUTE OF ENGINEERING FOR WOMEN (A)

(Approved by AICTE, New Delhi & Affiliated to JNTU- GV, Vizianagaram) Estd. 2008
Kapujaggarajupeta, VSEZ (Post), Visakhapatnam – 530049.

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

CERTIFICATE

This is to certify that project report title **"PHISHING DETECTION TOOL "** is being submitted to Department of Computer Science and Engineering by **K.Tejasri (22NM1A4619), P.Lavanya(22NM1A4637), P.Neelima (23NM5A4604), K.Jyothi (22NM1A4623**) during the study of **IV B.Tech I Semester of Bachelor of Technology in Computer Science & Engineering(Cyber Security),** during the period **May 2025-July 2025**, in partial fulfilment of the requirements for the award of the **Degree of Bachelor of Technology in Computer Science & Engineering.** This project report has not been previously submitted to any other University/Institution for the award of any other degree.

INTERNAL GUIDE **HEAD OF THE DEPARTMENT**

EXTERNAL EXAMINER 1 **EXTERNAL EXAMINER 2**

DECLARATION

We hereby declare that this project entitled "**PHISHING DETECTION TOOL**" is the original work done by us in partial fulfilment of the requirements for the award of the degree of Bachelor of Technology in Computer Science & Engineering, Jawaharlal Nehru Technological University, Vizianagaram. This project report has not been previously submitted in any other university/Institution for the award of any other degree.

K. Tejasri (22NM1A4619)
P. Lavanya (22NM1A4637)
P. Neelima (23NM5A4604)
K. Jyothi (22NM1A4623)

ACKNOWLEDGEMENT

We are deeply grateful to our project guide, **Dr.D.Manendhra Sai, Professor**, Department of Computer Science & Engineering, Vignan's Institute of Engineering for Women, whose valuable guidance and coordination were instrumental to the project's success.

Our heartfelt thanks to Dr. P. Vijaya Bharati, Professor & HOD, Department of Computer Science & Engineering, for providing the necessary resources and guidance throughout the project timeline.

We thank our esteemed Principal, Dr. B. Arundhati, for her support. We also appreciate the assistance provided by all faculty members during our project journey.

We express our sincere gratitude to everyone who supported us throughout this project.
The experience has been both rewarding and educational.

K. Tejasri
P. Lavanya
P. Neelima
K. Jyothi

ABSTRACT

In the digital age, phishing attacks have emerged as a significant threat to individuals and organizations, leading to substantial financial losses and data breaches. This paper presents a comprehensive phishing detection tool designed to identify and mitigate phishing attempts effectively. Leveraging advanced machine learning algorithms, natural language processing, and heuristic analysis, our tool analyses various attributes of URLs, email content, and website characteristics to classify potential phishing threats. Phishing is one of the most prevalent cybersecurity threats, exploiting human trust to gain unauthorized access to sensitive information through deceptive emails, SMS messages, phone calls, and social media interactions. This mini project presents a lightweight, rule- based Phishing Detection and Awareness Tool designed to identify and classify different types of phishing attacks, including Email Phishing, Spear Phishing, Whaling, Smishing/Vishing, and Angler Phishing. The tool allows users to input suspicious messages, which are analyzed based on predefined phishing indicators such as urgency, impersonation, suspicious URLs, and psychological manipulation tactics. The system then detects the likelihood of phishing and categorizes the attack type, while also offering safety tips and awareness examples to help users recognize and avoid similar threats in the future. Built using Python (Flask) for backend processing and HTML/CSS/JavaScript for a user- friendly frontend, this project serves as an educational platform and practical cybersecurity aid for everyday users.

Keywords:

Phishing Detection, Cybersecurity, Machine Learning, Natural Language Processing (NLP), URL Analysis, Email Security, Feature Extraction, Classification Algorithms, Real-Time Threat Detection, Data-Driven defence.

INTRODUCTION

1.1 Background of the Problem

In the digital age, the widespread adoption of internet services has significantly increased the volume and sensitivity of online communications and transactions. Alongside these advancements, cyber threats have grown in sophistication and frequency, with phishing attacks emerging as one of the most prevalent and damaging forms of cybercrime.

Phishing is a fraudulent practice in which attackers impersonate trustworthy entities often through emails, websites, or messages to deceive individuals into revealing sensitive information such as usernames, passwords, or banking credentials. These attacks exploit human vulnerability rather than system flaws, making them especially challenging to combat using traditional security mechanisms like firewalls and antivirus software.

According to industry reports, phishing attacks account for more than 80% of reported security incidents and continue to evolve, leveraging social engineering and obfuscation techniques to bypass conventional detection tools. This escalating threat landscape highlights the urgent need for intelligent, adaptive systems capable of identifying phishing attempts in real-time with high accuracy.

"PhishDetect" was conceptualized as a response to this challenge. By integrating advanced technologies such as machine learning, natural language processing, and real-time data analysis, the project aims to develop a robust and scalable phishing detection system. The solution is envisioned to not only improve cybersecurity defenses but also to contribute to the broader mission of making the digital ecosystem safer for users worldwide.

1.2 Problem Statement

The goal is to develop an intelligent phishing detection system that can automatically analyze user-submitted messages and identify potential phishing attempts in real time. With phishing attacks becoming increasingly sophisticated and difficult to distinguish from legitimate communication, the system must go beyond basic keyword matching to intelligently assess the intent and risk level of incoming messages.

Key aspects of the problem include:

- Heuristic and Rule-Based Detection: The system should be capable of identifying phishing indicators such as suspicious links, impersonation patterns, requests for sensitive data, and psychological manipulation tactics (e.g., urgency, fear, rewards), using a robust set of detection heuristics.

- User-Friendly Interface: It must provide a simple and accessible web-based interface, allowing non-technical users to input suspicious messages and receive immediate feedback about their safety.

- Real-Time Analysis: The system should analyze user input on-the-fly and deliver near-instant phishing risk assessments to prevent users from engaging with malicious content.

- Adaptability to Attack Variants: The tool should generalize well across different forms of phishing (e.g., email phishing, smishing) and adapt to evolving attack strategies by allowing easy updates to the detection rules.

- False Positive Minimization: To maintain user trust and effectiveness, the system should minimize false positives by clearly distinguishing between harmless content and actual phishing threats, using contextual analysis and risk indicators.

- Scalability and Lightweight Deployment: The system must be efficient and lightweight enough to scale for integration into various platforms (e.g., browsers, email clients, mobile apps) without heavy resource usage.

- Awareness-Oriented Output: Beyond detection, the tool should also offer short, educational insights on why a particular message was flagged, helping users build awareness and resist future phishing attempts.

1.3 Objectives of the project

The primary objective of this project is to develop a web-based phishing detection tool that can intelligently analyze user-submitted messages and identify potential phishing content. The system aims to assist users in recognizing phishing attempts and preventing data breaches by providing real-time feedback and educational insights. The specific objectives are:

- To detect phishing attempts based on content analysis: Analyze user-submitted messages using predefined heuristic rules to identify common phishing traits such as suspicious links, urgency language, and requests for personal information.

- To provide a lightweight and user-friendly web interface: Build a simple web application using Flask that allows users to input suspicious text and receive immediate detection results in a clear and understandable format.

- To increase user awareness of phishing threats: Educate users by highlighting risky elements in messages and offering safety tips, thereby promoting better decision-making and cyber hygiene.

- To enable fast and real-time detection: Ensure that the system responds quickly to user input, providing real-time or near-real-time phishing assessment without delays.

- To reduce false positives and ensure accurate results: Implement intelligent rules and context-aware filtering to minimize misclassification of legitimate messages as phishing, improving reliability.

- To design a modular system for future scalability: Structure the detection engine in a modular way to allow easy integration of future enhancements like machine learning models, URL scanning, or email header analysis.

METHODOLOGY USED

Chapter 1: Approach

The project used a hybrid phishing detection approach that combines **rule-based heuristics** and **machine learning classification** for more accurate and real-time analysis.

- **Rule-based analysis** detects phishing through specific keyword patterns, suspicious phrases (like "verify your account"), and link characteristics (e.g., use of shortening services or IP addresses in URLs).

- **Machine learning analysis** uses a trained Random Forest model to classify URLs as phishing or legitimate based on extracted features such as length, domain format, HTTPS usage, and more.

The application is designed to analyze **user-submitted text or links in real time**, returning results immediately through a web-based interface.The frontend sends data to the backend, which processes the input, runs both detection methods, and responds with a **classification result and explanation**.

Chapter 2: Tools Used

Development Environment

- **Visual Studio Code (VS Code)**: Used as the main IDE for writing and organizing Python, HTML, CSS, and JavaScript files.

- **Python Extension for VS Code**: Enabled linting, debugging, and virtual environment support directly in the editor.

- **Terminal in VS Code**: Used to install dependencies (Flask, Scikit-learn, etc.) and run the Flask development server.

Backend Technologies

- **Python**: Core language used for building the backend logic and machine learning model.

- **Flask**: A lightweight web framework for creating APIs and routing user input for analysis.

- **Flask-CORS**: Used to handle Cross-Origin Resource Sharing to allow communication between frontend and backend.

Machine Learning Tools

- **Scikit-learn**: Used to train and evaluate a **Random Forest Classifier** using features extracted from phishing and legitimate URLs.

- **Joblib**: Used to serialize the trained model into .pkl format for loading and prediction during runtime.

- **Pandas / NumPy**: Used for data preprocessing and manipulation (in the model training phase).

Frontend Technologies

- **HTML**: For the structural layout of the web interface.
- **CSS**: For styling and UI presentation.
- **JavaScript**: For handling user input, API calls, and dynamic result rendering.
- **Bootstrap 5**: For responsive layout and grid design.
- **Font Awesome**: For adding visual icons to enhance UI feedback.

Chapter 3: Techniques and Implementation

a. Extracted over 60+ features from each URL such as:

  - Length of URL and hostname
  - Use of IP address instead of domain
  - Number of special characters (@, -, =, etc.)
  - Presence of HTTPS and known shortening services
  - Count of subdomains, file extensions, and suspicious TLDs
  - Feature extraction logic was written in Python to ensure compatibility with the trained model.

b. Machine Learning Model

- Trained a **Random Forest Classifier** using the structured dataset from Kaggle.
- Model trained to classify URLs as **phishing (1)** or **legitimate (0)**.
- Serialized into a .pkl file using joblib and loaded into the Flask backend for real-time predictions.
- Model outputs were mapped to user-friendly labels like “Phishing (ML Classified)” or “Legitimate (ML Classified)”.

c. Rule-Based Detection

- Implemented keyword-based and pattern-based detection logic for analyzing:
  - Email content (e.g., “verify account”, “claim now”, “reset password”)
  - Suspicious URL patterns (e.g., IP-based URLs, shorteners, homoglyphs)
  - Impersonation of trusted domains (e.g., xn--pple.com, typosquatting)

d. Frontend Integration

- User interface built using HTML and styled with Bootstrap and custom CSS.

- JavaScript (script.js) used to:
    - Handle user input submission
    - Communicate with the backend via fetch() (POST request)
    - Display results dynamically, including status, reasons, and safety tips
    - Maintain user scan history using localStorage

e. Backend Routing (Flask)

- One main route /analyze is exposed to receive user input.
- Backend performs:
    - Rule-based analysis (text or URL)
    - ML-based analysis (only for URLs)
    - Returns a JSON response with:
        - Classification
        - Reasons for detection
        - Suggested safety precautions

f. Testing and Validation

- The system was tested using a mix of:
    - Known phishing URLs (from dataset)
    - Real-world phishing messages (scrubbed samples)
    - Safe and legitimate URLs for false positive checks
- The final output to users includes:
    - A clear status message ("Safe" or "Suspicious")
    - Reasons why the content was flagged (e.g., suspicious keywords, ML classification)
    - Suggested precautions (e.g., "Do not click links", "Verify sender independently")

OUTCOMES AND PERCENTAGE OF WORK COMPLETED

The development of the PhishDetect system has resulted in a functional, interactive, and intelligent phishing detection tool that successfully integrates rule-based logic with machine learning. The outcomes are evaluated based on functionality, performance, and user experience.

Key Outcomes Achieved

- **Real-Time Phishing Detection Engine**
  - Built and deployed a Flask-based backend capable of analyzing both URLs and text inputs.
  - Integrated a trained Random Forest model (phishing_model.pkl) that accurately classifies URLs as phishing or legitimate.
  - Rule-based analysis handles content-based phishing (e.g., job scams, login baits, financial fraud).
- **Feature Extraction for ML Integration**
  - Developed a feature engineering function that replicates the training process to ensure consistent input to the model.
  - Extracts over 60+ structural and lexical URL features (e.g., presence of IPs, special characters, length ratios).
- **User-Friendly Web Interface**
  - Designed a clean, mobile-responsive frontend using HTML, CSS, and JavaScript.
  - Interface includes input field, real-time analysis button, and dynamic result presentation.
  - Clear color-coded result feedback: green for safe, red for suspicious.
- **Interactive Feedback with Explanations**
  - Provides reasons behind detection (e.g., suspicious keywords, Punycode usage, ML prediction confidence).
  - Offers tailored safety tips and awareness-building information to the user.
- **History Tracking Feature**
  - Implemented local history tracking using browser storage to let users review past inputs and results.
- **Modular and Extensible Architecture**
  - Clean separation between frontend, backend, rules, and model logic allows for future expansion (e.g., mobile apps or browser plugins).

Completion Percentage Summary

Component	**Status**	**Completion (%)**
Requirement Analysis	Completed	100%
Rule-Based Detection Module	Completed	100%
Machine Learning Model Integration	Completed	100%
Feature Extraction Engine	Completed	100%
Frontend UI (HTML, CSS, JS)	Completed	100%
Flask Backend & API Integration	Completed	100%
Testing & Validation	Completed	100%
Deployment (Local)	Completed	100%
Documentation	Completed	100%

Overall Completion: 100%

FUTURE SCOPE OF THE PROJECT

The current version of PhishDetect successfully delivers a lightweight, real-time phishing detection system with a clean user interface and integrated machine learning capabilities. However, there is significant potential to expand its reach, accuracy, and impact in future development phases. One of the key areas of advancement lies in enhancing the machine learning model. While the current implementation uses a Random Forest classifier, more sophisticated algorithms such as gradient boosting, deep learning, or hybrid ensemble models could be employed to increase accuracy and better handle subtle phishing variations. Additionally, incorporating natural language processing (NLP) techniques would allow the system to understand the context and intent behind messages something that simple rule-based systems often miss making it more resilient against social engineering attacks.

To improve usability and real-world applicability, PhishDetect can be extended beyond the browser into a standalone mobile application or a browser extension. This would enable users to detect phishing attempts directly within their emails or web pages in real time, without needing to visit an external site. Integration with popular platforms such as Gmail, Outlook, or enterprise email clients would allow seamless analysis of incoming messages. On the infrastructure side, deploying the solution as a cloud- based API service or integrating it into enterprise security pipelines via webhooks would make it scalable and enterprise-ready.

Furthermore, the rule-based anomaly detection system, currently tailored for Docker log monitoring, can evolve into a broader log intelligence platform. Future enhancements could include real-time alerting systems via Slack, email, or SMS, correlation of threats across services, and linking phishing detection with broader security incidents. With support for log filtering, user roles, secure authentication, and alert prioritization, PhishDetect could serve as a foundational security tool in both personal and enterprise settings.

Finally, connecting the system with external threat intelligence databases (like VirusTotal or AbuseIPDB) would allow for automatic blacklist updates and risk scoring of new domains. Coupled with continuous learning, explainable ML outputs, and a scalable deployment model via Docker or Kubernetes, PhishDetect has the potential to grow into a comprehensive phishing and anomaly detection platform that is not only technically advanced but also accessible and educational for users of all backgrounds.

REFERENCES

- **Abdelhamid, N, Ayesh, A,&Thabtah,F.(2014)**

  Phishing detection: A recent intelligent machine learning comparison based on models content and features.In 2014 IEEE International Conference on Information Technology (pp. 1-8). IEEE. https://doi.org/10.1109/ICIT.2014.46

- **Kaggle. (n.d.).**

  Phishing Website Detector Dataset.

  Retrieved from https://www.kaggle.com/datasets (Include your specific dataset link here)

- **Scikit-learn Developers. (2023).**

  scikit-learn: Machine Learning in Python – Random Forest Classifier Documentation.

  Retrieved from https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html

- **CERT-In (Indian Computer Emergency Response Team). (2024).**

  Advisories and Guidelines on Phishing and Cybersecurity Awareness.

  Retrieved from https://www.cert-in.org.in

- **Mozilla Developer Network (MDN). (2023).**

  HTML, CSS, and JavaScript Web Docs.

  Retrieved from https://developer.mozilla.org/

- **OWASP Foundation. (2023).**

  OWASP Phishing Guide – Phishing Detection and Prevention Best Practices.

  Retrieved from https://owasp.org/www-community/phishing

Extending the Project as a Final Year Main Project

- To extend PhishDetect into a Final Year Major Project follows a structured development plan can be followed to enhance its technical depth, usability, and real-world relevance. The first step would involve integrating more advanced machine learning and deep learning models such as XGBoost, Support Vector Machines (SVM), or Long Short-Term Memory (LSTM) networks to improve detection accuracy, particularly for complex phishing attempts. These models could be evaluated using metrics like precision, recall, and F1-score for better comparative analysis and reliability.
- Beyond URL analysis, expanding the system to handle full email-based phishing detection would greatly increase its effectiveness. This involves analyzing raw email content, headers, subject lines, and detecting impersonation tactics. Real-time threat intelligence integration through APIs like VirusTotal or Google Safe Browsing could further enhance detection by cross-verifying links and IPs with known blacklists. To make the system multi-user and production-ready, a user dashboard with authentication, scan history, and admin control for rule management could be developed using secure login framework.
- The project can also be extended through the development of browser extensions or mobile apps to allow phishing detection directly during user browsing. REST APIs can be used to connect these clients with the backend Flask server. A real-time alerting system could also be implemented to notify users via email or Slack when a high-risk message is detected. Finally, the entire system could be containerized using Docker and deployed to a cloud platform like AWS or Heroku with CI/CD pipelines for automatic deployment and updates, transforming PhishDetect into a full-fledged, scalable security solution suitable for enterprise or academic use.
- PhishDetect project detecting phishing using rule-based logic and machine learning have already been successfully achieved. Extending this system further would mainly involve minor enhancements rather than offering new learning opportunities. At this stage, I am more interested in exploring different domains such as Artificial Intelligence, Computer Vision, or the Internet of Things (IoT), where I can apply and expand my skills in new ways. Repeating the same project would not reflect growth in my technical abilities or demonstrate new concepts. Since PhishDetect already performs effectively and fulfills its intended purpose, I have decided to conclude this project and begin work on a new topic that will offer broader experience and diversity. The knowledge gained from this project will continue to support my development in future projects.