iJRASET
International Journal For Research in Applied Science and Engineering Technology

# INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

**Volume:** 13 **Issue:** IV **Month of publication:** April 2025

**DOI:** https://doi.org/10.22214/ijraset.2025.69772

www.ijraset.com

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

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
*ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538*
*Volume 13 Issue IV Apr 2025- Available at www.ijraset.com*

# AI Based Web Approach System for Detecting Malicious URLs and Preventing Cyber Fraud

Prof. Chethana R.M[1], Shaheeda Kasim Pinjar[2], Sneha[3], Tisha Prakash Kavri[4], Yashaswini S[5]

***Abstract:*** ***Describehowadvancesindeeplearning,especially As the internet becomes more essential in our daily lives, cyber fraud—especially through harmful links—has become a serious issue. This project introduces a smart, web-based system that detects and classifies dangerous URLs in real time, such as phishing, malware, or defacement links. It uses machine learning models trained on a mix of safe and harmful URLs by analyzing features like link structure, special characters, and keywords.***
***Built using Flask, the system provides a simple interface where users can check URLs. It also includes a feedback option, so users can help improve accuracy by confirming or correcting results, which helps the system learn and improve over time. The system can also scrape live webpage content and display the main text in a clean, readable format to help users understand what the page is about. A whitelist of trusted domains helps avoid unnecessary checks on safe websites. The design is light, fast, and easy to expand in the future with features like scanning multiple links, analyzing content tone, or auto-flagging suspicious content. Overall, this system offers a smart, user-friendly, and effective way to fight cyber threats using AI and real-time analysis.***

## I. INTRODUCTION

Cyber fraud has become a major concern in today's digital world, affecting individuals, businesses, and governments. As internet use grows, attackers are finding new ways to trick users—often through malicious URLs that appear safe but lead to harmful websites. These links are spread through emails, messages, ads, and social media, and can be used for phishing, spreading malware, or website defacement.

Traditional security methods like blacklists are often slow, outdated, and ineffective against new or unknown threats. To tackle this, smarter and faster systems are needed that can detect harmful URLs before they cause harm.

This project presents an AI-based web system that uses machine learning to analyze URLs and classify them as safe or malicious. It learns from URL patterns and user feedback, improving over time. The system also uses web scraping to check live content and includes a feature to show clean, readable webpage content to help users decide if a link is safe.

Built with the Flask framework, the system offers a simple interface for users and includes a trusted domain whitelist to reduce false alerts. Overall, it provides a smart, real-time solution to help protect users from online threats.

## II. PROBLEM STATEMENT

Cybercriminals increasingly use malicious URLs for phishing, spam, and malware attacks. Traditional detection methods fail against evolving threats like redirection and shortened links.

*1)* Malicious URLs are used in cyber attacks like phishing and malware, Attackers often disguised through redirection, shortened links, or trusted domains to avoid detection.
*2)* Traditional methods like blacklists and signature-based detection are not effective against new, smart threats.
*3)* Cyber fraud through malicious URLs can impact all sectors—such as finance, healthcare, education, and e-commerce—by compromising sensitive data, disrupting services, and causing financial and reputational losses.
*4)* An AI-powered system using machine learning and NLP can accurately classify URLs in real time, helping prevent cyber fraud and protect users during online activities.

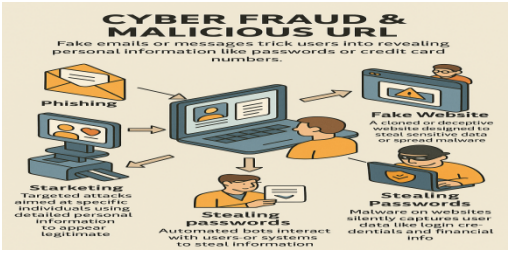


Figure1 Cyber Fraud

## III. OBJECTIVES

1) The system analyzes structural features of URLssuch as length, special characters, and suspicious keywordsto detect if a URL is safe or harmful. This helps in identifying threats before users interact with them.
2) Instead of just labeling URLs as malicious, the system classifies them into specific categories like phishing, malware, or defacement.
3) A feedback mechanism allows users to confirm or correct the system's predictions. This input is used to retrain the model, helping it learn from real-world interactions and improve over time.
4) The system scrapes live webpage content to analyze what the URL actually displays. This dynamic check helps identify threats that may not be obvious from the URL structure alone.
5) A content extraction module presents the main readable parts of a webpage in a clean format. This helps users judge if a URL leads to a trustworthy page without opening potentially harmful sites.
6) The user interface is built using Flask, making the system lightweight, responsive, and easy to use. Users can easily enter URLs, view results, and give feedback in a seamless environment.

## IV. LITERATURE REVIEW

The detection of malicious URLs and prevention of cyber fraudare critical areas in cybersecurity research. Numerous systems and methodologies have been proposed and implemented over the years. These systems largely fall into four broad categories: blacklist based systems, heuristic based approaches, traditional machine learning models, and web scraping baseddetection mechanisms. Despite their widespread usage, these systems face significant challenges in terms of real time detection , adaptability to evolving threats, and usability.

### A. *Existing Systems*

1) Blacklist based systems :these system use statics databases ofknown malicious websites, such as Google Safe Browsing , Phish Tank and URL Haus but struggle with new threats, While they offer fast detection for previously identified threats, they are ineffective against new, unknown or rapidly changing phishing and malware URLs.
2) Heuristic-based Systems: These systems analyze URLs for malicious patterns but are
3) vulnerable to evasion techniques. Heuristic systems detect suspicious URLs by analyzing known patterns, like specific keywords ("login", "secure"), unusual domain structures, or the use of IP addresses instead of domain names. However, attackers can bypass these rules using domain obfuscation or legitimate-looking URLs.
4) Machine Learning based Systems (Traditional): Machine learning models classify URLs using features like length, special characters, and domain data. Although they perform better than static methods, they rely on fixed datasets and lack adaptability to new attack types
5) Web Scraping for Malicious URL Detection: Analyze live webpage content for threats. Often lack clear data presentation and struggle with dynamic content.

### B. *Key Findings*

1) Blacklist-based Systems: These use static blacklists of known malicious websites but struggle with new threats.
2) Heuristic-based Systems: These analyze URLs for malicious patterns but are vulnerable to evasion techniques.
3) Machine Learning based Systems (Traditional): Some use machine learning for URL classification but may lack adaptability.
4) Web Scraping for Malicious URL Detection: Some systems use web scraping but may not provide effective content extraction

### C. *Limitations*

Existing systems often suffer from

1) These systems struggle to detect newly emerging malicious sites or dynamic phishing techniques.
2) Traditional machine learning models often cannot adapt to evolving attack strategies.
3) Most systems do not allow users to provide direct feedback that can be used to improve the system
4) Insufficient content analysis: Many systems only classify URLs but do not provide a comprehensive analysis of website content to aid in verification

## V. PROPOSED SYSTEM

The proposed AI-based Web Approach System for Detecting Malicious URLs and Preventing Cyber Fraud addresses the limitations of existing systems by introducing several novel features:

*1) Dynamic Machine Learning Classification:*

- The system uses a Random Forest Classifier trained on URL features (e.g., length, dots, slashes, digits, keywords) to classify sites as phishing, malware, defacement, or safe. It continuously adapts to new threats through ongoing user feedback.
- Using Feedback mechanism :The system includes a user feedback mechanism for its predictions. Users can indicate whether a site is correctly classified as benign or malicious. This feedback is collected and stored for future use. It helps retrain the model, improving its accuracy and adaptability over time.

*2) Web Scraping for Live Webpage Analysis:*

Unlike most systems that only check the URL, this system also looks at the actual content of the webpage. It uses tools like `requests` and `BeautifulSoup` to scrape live content from the site. This helps the system see what's really on the page in real time. It then checks if the content looks suspicious or matches known patterns of phishing or fraud.

3) *Readable Content Extraction:*

The system has a feature that pulls out the main text from a webpage using tools like `readability-lxml` and `BeautifulSoup`. It removes unnecessary code and clutter to show clean, readable content like articles or key information. This helps users easily understand what the site is about. By seeing the actual content, users can better decide if the website is real or suspicious.

*4) Flask-based User Interface:*

- The system is built using Flask, a lightweight web framework. The interface is simple, intuitive, and accessible, providing users with:
- A URL input field to test suspicious URLs.
- A feedback form to rate the system's predictions (correct or incorrect).
- A content extraction page that fetches and displays live webpage content for further analysis.
- The web interface ensures ease of use and real-time interaction with the system

*5) Scalability and Customization:*

- The system is designed to grow and can include extra features like checking the tone of the content, spotting smarter scams, or supporting multiple languages.
- It can also be improved using advanced machine learning methods like deep learning or combining several models for better accuracy.

### *A. System Architecture*

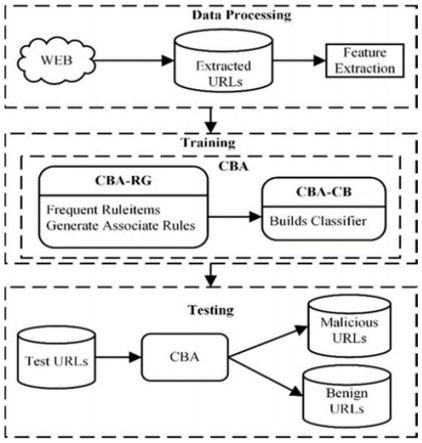



Figure 2 System Architecture of proposed system

1) Data Processing: URLs are collected from the web. Important features like length, symbols, and keywords are extracted to help identify patterns.
2) Training:
- The system uses CBA (Classification Based on Association).
- It finds common patterns in the data and builds rules.
- These rules are then used to create a classifier that can tell if a URL is safe or malicious.

3) Testing: New URLs are tested using the trained model, and then classified as either malicious or benign (safe).

*B. Dataset And Preprocessing*

1) Dataset Source: This project relies on clean, high-quality data to accurately detect malicious URLs. The dataset, titled "Malicious and Benign URLs," was sourced from Kaggle and includes a large number of labeled URLs with different types of threats. It was carefully cleaned and preprocessed to train a reliable machine learning model that can perform well in real-world situations.
2) Labels

Each URL in the dataset is associated with one of the following labels:
- Benign – Safe URLs that do not pose any security threat.
- Phishing – URLs designed to deceive users into revealing personal or confidential information.
- Malware – URLs that lead to the installation or distribution of malicious software.
- Defacement – URLs targeting websites for unauthorized content modification, often for ideological or reputational attacks

3) Preprocessing Steps:
- URL Cleaning: Irrelevant characters and noise were removed by trimming whitespace, standardizing formatting, and discarding incomplete or malformed URLs.
- Feature Engineering:

Instead of feeding raw URLs into the machine learning model, structured features were extracted to capture meaningful patterns. These features include:
- URL length
- Number of dots, slashes, and hyphens
- Presence of special characters like '@', '?', '='
- Count of numeric digits
- Use of suspicious keywords such as "login", "verify", "secure", etc.
- Presence of IP addresses
- Whether the URL uses HTTPS
- Number of subdomains

This structured representation helps the model learn the characteristics commonly
found in malicious versus benign URLs

4) Label Encoding:

Since machine learning models operate on numerical data, the categorical labels (e.g., benign, phishing) were encoded into integer values using LabelEncoder. This allows the model to interpret the output class for training and prediction.

5) Train-Test split

The dataset was divided into training (80%) and testing (20%) subsets. This split ensures that the model is trained on a substantial portion of the data while still being evaluated on unseen data to validate its generalization capability

*C. Flask Integration*

This project uses Flask, a simple Python web tool, to build the website and connect it with the AI model. It helps users easily interact with the model and get real-time results.

*1) Backend functionality*

- URL Input Handling: Users enter a URL through a web form, and Flask sends it to the model which tells if it's safe or harmful.
- Prediction and Trusted Domain Logic: The system first checks if the URL is from a trusted site; if not, it uses the model to predict if it's safe or a threat.
- User Feedback Mechanism: After getting the result, users can confirm if it was right, and this feedback is saved to help improve the model later.
- Content Extraction Endpoint: There's also a feature that shows readable content from the URL, helping users decide if the site is trustworthy.

*2) Front-end Design*

- Index.html:

The main Landing page where users can:

- Submit a URL for classification
- View the predicted result
- Submit feedback (Yes/No) to confirm if the prediction was correct
- Access a link to view the readable content of the entered URL
- Content_html: A secondary page dedicated to displaying the extracted content from the URL. This allows users to visually inspect the site's main body text, enabling them to make informed decisions before clicking on suspicious links.

*D. Technologies Used*

- Flask: Core backend framework for routing and model integration.
- HTML/CSS: For creating responsive, user-friendly interfaces.
- Jinja2: Flask's templating engine used to dynamically display results and content.
- CSV Module: Used for storing user feedback data.

## VI. METHODOLOGY

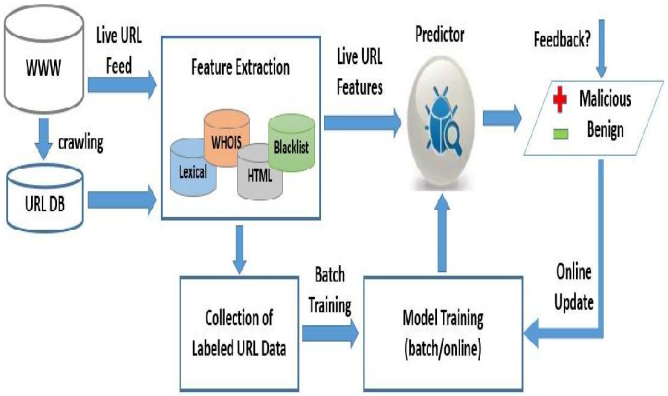



Figure3 Overview of Malicious URL detection with feedback driven

*1)* Data Collection: URLs are collected from live feeds and databases. Selenium (gather.py) captures dynamic content, while scrape.py extracts static HTML. All data is stored in a
*2)* centralized data lake.
*3)* Feature Extraction: Features are derived from URL structure, WHOIS data, blacklist status, and HTML content. These include length, special characters, domain age, and
*4)* suspicious patterns.
*5)* Model Training: A Random Forest Classifier is trained on labeled data (benign, phishing, malware, defacement) due to its accuracy, robustness, and interpretability.
*6)* Real-Time Prediction: The model is deployed in a Flask web app, allowing users to input URLs and get immediate classification results.

7) User Feedback: Users confirm or correct predictions. Feedback is stored and used for periodic model updates, improving future accuracy.
8) Content Extraction: Readable webpage content is extracted using readability-lxml and BeautifulSoup to help users assess site legitimacy safely

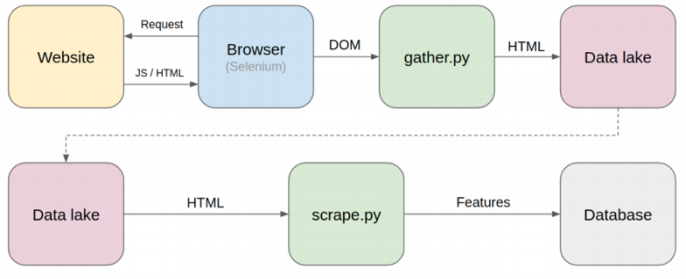



Figure 4 Steps to collect and process website data into a database

## VII.RESULTS AND DISCUSSION

A strong machine learning model was built and carefully tested to classify URLs as safe or harmful, using key evaluation methods and helpful tools to guide the process.

### A. *Model Used:*

A Random Forest Classifier was used because it combines many decision trees to make accurate predictions and avoid overfitting. It works well with the kind of structured data taken from URL features

Training Process:

1) Feature Input: The model was trained using a set of features from each URL, like its length, number of dots, slashed and suspicious words, to help identify if it's safe or harmful
2) Label Encoding: Categorical labels (benign, phishing, malware, defacement) were encoded into numerical values, providing a quantitative basis for model training.
3) Data Splitting: The dataset was split 80% for training and 20% for testing to make sure the model learns well and its tested on new, unseens URLs for better accuracy in real use

### B. *Evaluation Metrics:*

1) Accuracy (89%): Accuracy measures the overall correctness of the model across all classes,indicating that 89% of the URL predictions made by the model were correct
2) F1-Score (Macro Average: 86%): The macro-average F1-score balances precision and recall in multiclass classification. An 86% score shows the model performs well across all URL types.
3) Confusion Matrix: A confusion matrix was created to show how well the model classified each type of URL. It confirmed that the model clearly separated safe URLs from different malicious ones like phishing, malware, and defacement.

### C. *Comparative Analysis of Traditional Approaches and the Proposed Random Forest Based Solution*

Traditional methods like blacklists, pattern matching, and basic machine learning have been used to detect harmful URLs, but they often miss new or hidden threats and lack adaptability. In contrast, this project uses a smarter, AI-based system with a Random Forest model that analyzes detailed URL features, performs live content checks, and learns from user feedback. It also provides an easy-to-use web interface built with Flask for real-time URL checking and improved protection.

### D. *Evaluating Random Forest as optimal classifier*

The Random Forest Classifier was chosen for its high accuracy (89%) and strong performance across different types of URLs. It combines many decision trees to reduce errors and avoid overfitting, making it reliable and adaptable. It also shows which features matter most, helping build trust in its decisions especially important in cybersecurity. Plus, it handles messy data well and works smoothly with the system's feedback loop, allowing the model to keep improving without starting over.

*E. Result Screenshot*

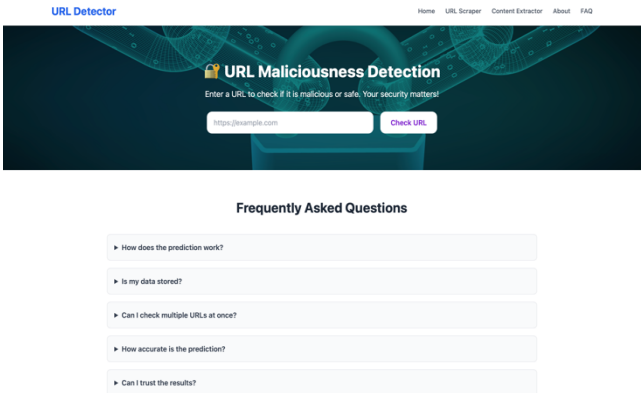



Figure 5 Screenshot of the Project Result

This webpage is part of our project called **Malicious URL Detection**, built to help users check if a URL is safe or harmful. It lets you enter any link and instantly tells you if it's **benign**, **phishing**, **malware**, or **defacement** using a trained machine learning model. In addition to detecting threats, it also offers **web scraping** and **content extraction** tools that show the readable content of a webpage. This helps users confirm if a website is genuine before they visit or interact with it. Everything is built with a user-friendly design so anyone can quickly and easily stay safe online.

*F. Overview Of Additional Feature: Web Scrapping*

Web scraping is used to look at the actual content of a webpage, not just the URL. This helps catch hidden threats like fake login forms, phishing messages, or suspicious code that might not be clear from the link alone.

*1)* User Interaction: Users can enter any URL, and the system will fetch and analyze its content to extract useful information.
*2)* Technologies Used: The system uses `requests` to get the webpage's HTML and `BeautifulSoup` to extract readable content from it.
*3)* Process Flow:

- User enters a URL on the scraping interface.
- The system sends a request to the URL and receives its HTML response.
- The HTML is parsed using BeautifulSoup to isolate the visible and meaningful content.
- The extracted content is then displayed back to the user for inspection.

Benefits and Use case:

- Allows users to check if a website is trustworthy by looking at its live content.
- It works alongside prediction tools to give more detailed insights about a website
- Great for checking websites that aren't on existing blacklists or databases yet, allowing users to make quick decisions.

## VIII. ADVANTAGES

*1)* Continuous user feedback helps the system get better at detecting new threats.
*2)* The system doesn't just classify URLs; it looks at the website's content to give a clearer picture of its trustworthiness
*3)* The Flask-based interface makes it simple for all users, whether they're tech-savvy or not.
*4)* The system can grow with new machine learning models and features for deeper analysis as threats evolve.

## IX. LIMITATIONS

AI-based web systems for detecting malicious URLs have limitations, including the risk of false positives and negatives, where safe URLs may be flagged as harmful or malicious URLs might be missed. They also rely on large, up-to-date datasets, and may struggle with new threats until they are trained. The system can be resource-intensive, requiring powerful servers, and some malicious sites may use techniques to bypass detection.

## X. CONCLUSION

This project presents a smart and user-friendly AI-based web system that helps detect and prevent cyber fraud by analyzing URLs in real-time. Using machine learning, web scraping, and content extraction, the system can accurately classify URLs as safe or malicious (like phishing or malware) and show users clear, readable webpage content to help them make better decisions. A feedback feature lets users improve the system's accuracy over time. Overall, it's a scalable and interactive solution that enhances online safety and lays the groundwork for future upgrades like multilingual support and integration with other security tools.

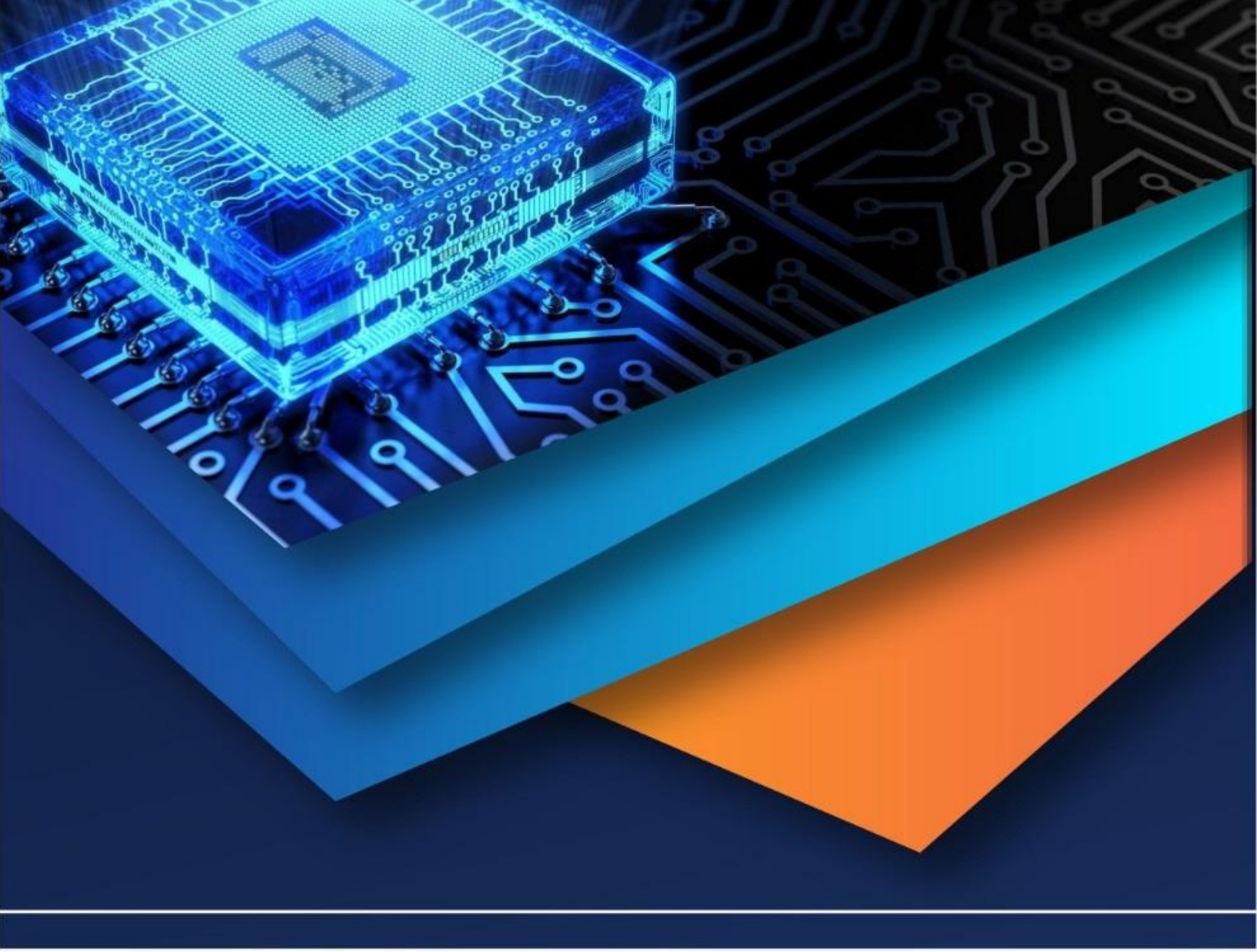
## XI. FUTURE SCOPE

*1)* Natural Language Analysis: Applying sentiment analysis or keyword detection on the extracted content to flag potentially harmful language or intent.
*2)* Language Detection and Translation: Enabling multilingual support for globalaccessibility.
*3)* Screenshot Previews: Adding visual context through automated screenshots of thewebpage.

## REFERENCES

[1] Shumail, A., & Iqbal, Z. (2021)A Complete Review of How URLs Are Classified to Detect Phishing. Published in the International Journal of Computer Applications., 176(4), 7-13.
[2] Singh, D., & Sahu, N. (2018).Detection of PhishingWebsites Using URL-BasedFeatures. Proceedings of the International Conference on Information Technology
[3] Finkel, H., & Rodriguez, S. (2017). A Review on the Techniques for Website Content Extraction. Journal of Web Engineering, 16(3), 120-135.Manning, C. D., & Schütze, H. (1999). Foundations of Statistical Natural Language Processing. MIT Press.
[4] Zhang, Z., & Guo, Y. (2020). Web Scraping and Data Mining for Online Security Applications. Springer.
[5] Python Software Foundation. (2023). BeautifulSoup Documentation. Retrieved from https://www.crummy.com/software/BeautifulSoup/ Python Software Foundation. (2023). Requests Documentation. Source: https://docs.python-requests.org/en/master/Readability-lxml Documentation. (2023). Readability-lxml - A Python Library for
[6] Readability. Source: https://readability-lxml.readthedocs.io/en/latest/ Scikit-learn Documentation. (2023). Scikit-learn - Machine Learning in Python. Retrieved from https://scikit-learn.org/stable/
[7] Faizan, A. (2024). Guardians of the Digital Realm: Navigating the Frontiers of Cybersecurity. Integrated Journal of Science and Technology
[8] Malatji, M., & Tolah, A. (2024). Artificial intelligence (AI) cybersecurity dimensions: a comprehensive framework for understanding adversarial and offensive AI. AI and Ethics, 1-28
[9] Liu, R., Wang, Y., Xu, H., Qin, Z., Liu, Y., & Cao, Z. (2023). Malicious URL Detection via Pretrained Language Model Guided Multi-Level Feature Attention Network. arXiv preprint arXiv:2311.12372
[10] Abad, S., Gholamy, H., & Aslani, M. (2023). Classification of malicious URLs using machine learning. Sensors, 23(18), 7760.
[11] Aljabri, M., Altamimi, H. S., Albelali, S. A., Al-Harbi, M., Alhuraib, H. T., Alotaibi, N. K., ... & Salah, K. (2022). Detecting malicious URLs using machine learning techniques: review and research directions. IEEE Access, 10, 121395-121417.
[12] Reyes-Dorta, N., Caballero-Gil, P., & Rosa-Remedios, C. (2024). Detection of malicious URLs using machine learning. Wireless Networks, 1-18.

*ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538*
*Volume 13 Issue IV Apr 2025- Available at www.ijraset.com*



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

# INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY