DEVELOPMENT OF A REAL-TIME PHISHING WEBSITE DETECTION SYSTEM

BY

AMADI CHIBUIKEM FRANCIS

GOU/U22/CSC/948

PROJECT REPORT SUBMITTED TO THE DEPARTMENT OF COMPUTER SCIENCE AND MATHEMATICS, FACULTY OF COMPUTING AND INFORMATION TECHNOLOGY, GODFREY OKOYE UNIVERSITY, ENUGU STATE

IN PARTIAL FULFILMENT OF THE REQUIREMENTS FOR AWARDING THE DEGREE OF BACHELOR OF SCIENCE IN COMPUTER SCIENCE

SUPERVISOR:

ENGR. CAMILLUS NJOKU

MAY, 2026

APPROVAL PAGE

This research, titled "**DEVELOPMENT OF A REAL-TIME PHISHING WEBSITE DETECTION SYSTEM"** has been assessed and approved by the Department of Computer Science and Mathematics Committees in the Faculty of Computing and Information technology, Godfrey Okoye University, Enugu, Nigeria.

Engr. Camillus Njoku **.....**
.....

Project Supervisor Signature Date

External Examiner **.....**
.....

External Examiner Signature Date

Dr. Frank Okebanama **.....**
.....

HOD, Department of Signature Date

Computer Science and

Mathematics

Prof.Dr. I A Ajah **.....**
.....

Dean, Faculty of Computing Signature Date

And Information Technology.

CERTIFICATION

I certify that I completed the research included therein and that it was primarily based on my observations and investigation. Consequently, I will fully accept the University's decision if I am found to have cheated on the assignment.

_____

AMADI CHIBUIKEM FRANCIS

GOU/U22/CSC/948

DEDICATION

This project is dedicated to God Almighty for His grace and guidance, to my family for their unwavering support.

ACKNOWLEDGMENTS

First and above all, I give thanks to God for the strength to keep going when the work felt endless, for clarity when nothing made sense, and for seeing me through to this point. None of this would exist without His grace.

To my parents, thank you for never once making me feel like this was too much to pursue. Your support through the long nights, the pressure, and the uncertainty was the kind that cannot be repaid, only carried forward.

To my project supervisor, Engr. Camillus Njoku, thank you not just for the guidance, but for the patience. You asked the right questions, pointed out what I had missed, and never let me settle for less than my best. This project is sharper because of you.

To my Head of Department, Dr Frank Okebanama, I am grateful for the direction and academic environment that made this project possible. Your commitment to the department sets a standard that pushes students to take their work seriously.

To the lecturers who taught me over these years: Dr Ajah (Dean), Dr Okebanama (HOD), Prof Okereke, Engr Kingsley and others whose names I cannot mention here, I did not always appreciate it at the moment, but looking back, the foundation you built shows up in every part of this work. Thank you.

And to everyone else who played even a small part in this project, whether through advice, a conversation, or just encouragement at the right time, it meant more than you know.

ABSTRACT

Phishing has emerged as a massively common type of cyber threat, relying on misleading websites and emails to obtain confidential user information. Most traditional phishing detection techniques such as blacklists and rules-based detectors have trouble keeping pace with new and emerging threats in real-time. In this paper we describe the design and development of a machine learning-based, real-time phishing detection system (RTS) that greatly enhances both the accuracy and speed of detecting phishing attacks. The RTS detects phishing through a random forest classification model trained on datasets of email messages and URLs marked as either phishing or legitimate. Features related to phishing were extracted from the input URLs and email messages through a feature extraction module, and the input URLs were geolocated through an embedded geolocation module to provide additional geographic intelligence of the originating domain/email sender. The RTS was implemented as a web-based application developed using Python, Flask (Web Application Framework), HTML, CSS and JavaScript, thereby providing a user-friendly interactive interface for users to conduct real-time phishing attacks. Evaluating the RTS using standard machine learning metrics, the RTS achieved a 97% accuracy rate, 95% precision, 97% recall, and 96% F1-Scores. These evaluation results demonstrate that the RTS is a viable solution to detect phishing attacks and to provide additional context to the detected threat from the geolocation analysis. The study concludes that integrating machine learning with real-time web technologies offers a practical and reliable solution for mitigating phishing attacks.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER ONE

INTRODUCTION

1.1 BACKGROUND OF THE STUDY

Phishing is a social engineering approach, and is used to refer to the computer crime involving deceiving people/organisations to divulge sensitive details. They achieve this by presenting them with links to apparently credible websites, typically via email and text messages. Despite the fact that today phishing belongs to the number of rather well-known cybercrime attacks, it became more widely recognised by the population only a decade(10) after its introduction. Phishing is loosely based on the words fishing for victims and the fact that ph is used instead of f, so that it can be related to the original hackers, who were called Phreaks (Matecas et al., 2025).

The first record of the term 'phishing' being used was in January 1995 from the cracking toolkit AOHell, which was used to exploit the weaknesses in AOL. It was mentioned in a Usenet newsgroup called AOHell. Notably, AOL or American Online would later become the platform where the first signs of a major cybercrime would take place. Back when AOL was the most popular online service provider, it was a nest to several underground communities, including hacker forums and pages for sharing pirated software - it was called the warez community (Anupama, 2022). This community was the first to make moves towards conducting phishing attacks. They did this by illegally acquiring users' passwords and using an algorithmic software to generate fake credit card numbers, which were in turn used to create AOL accounts. However, in 1995, AOL employed security measures which eventually stopped the generation of arbitrary credit cards.

Consequently, with their credit card business shut, it didn't take long before phishers devised a new means to steal information from users online. They'd send messages through the AOL

messaging system, impersonating AOL employees and ask for sensitive information. This method proved to be especially efficient since it was a technique that had never been done before, and the users were always so trusting since they really didn't have a reason not to. It escalated to the point of AOL attaching warning messages to their email and Messenger clients to prevent their clients from falling victim to the malicious attacks.

Although phishing originated from the underground communities of AOL it gradually evolved to focus on online payment, starting with an unsuccessful attack on E-Gold (Tanti, 2024). It further evolved in 2003 to the cloning of popular sites that offer financial services and spamming clients' emails, prompting them to update their payment information. This is the format most commonly used today. By 2004, it expanded to banking sites and over time became a part of the black market, normally employed with specialised software to commit organised cybercrime on a global scale (Osamor et al., 2025)

More importantly, phishing attacks have left a lasting impact on the World Wide Web, as in 2015, Ukraine experienced possibly the worst instance of a phishing attack when its power grid system was compromised. The build-up started a month before the security incident by spamming phishing emails to the system operators and administrators in order to gain administrator privileges to the system. As a result of this breach, citizens were left in a power blackout during a cold winter night. Similarly, between 2013 and 2015, Evaldas Rimasauskas managed to deceive large companies like Google and Facebook, sending fake invoices that led to over $120 million in unauthorised payments. His arrest in 2017 highlighted the necessity for rigorous training and security sensitisation in financial departments to recognise and prevent such fraudulent activities (Abraham et al., 2025).

Therefore, a thorough analysis of the impact of phishing on today's society would lead anyone to draw a conclusion that superior methods or techniques should be adopted to curb the dynamic landscape of phishing techniques being developed. Phishing detection techniques like

URL-based, Content-based, and Blacklist-based approaches are among the most effective and successful in practice (Abuzuraiq & Alkasassbeh, 2019). However, for the sake of this study, we are interested in using the Heuristic/Machine Learning approach.

Machine Learning (ML) is a subdiscipline of AI that allows systems to acquire patterns from collecting and processing data, apply the learned patterns to new tasks, make judgments or predictions based on experience, and also continue improving with time without being programmed to do so (Davies et al., 2025). ML is best for this investigation owing to its ability to adapt to evolving phishing methods and to enable real-time detection in the proposed system, which is particularly valuable in dealing with zero-day attacks (Wilk-Jakubowski et al., 2025), and therefore is the adopted approach for detecting phishing attacks.

1.2 STATEMENT OF PROBLEM

Although existing phishing detection systems have made tremendous impact in curtailing phishing attacks and providing security to the Confidentiality, Integrity, and Availability of organisation's systems, there are still areas that need further improvement:

1. Limited Cross-Platform Integration - Existing systems are only available on web browsers, leaving a large percentage neglected.

2. Unengaging User Onboarding: Many systems have bland or unappealing user interfaces. This extends into features and functionalities such as the onboarding process and on-site navigation which can discourage potential users from actively using them

3. Lack of User Education: Current systems focus solely on detection without educating users on phishing tactics. This makes them reliant on automated protection without knowing what exactly they are being protected against.

4. No Sandbox Preview Control – Users cannot safely inspect suspicious links in an isolated environment before scanning, limiting situational awareness and trust in the results.

1.3 AIM AND OBJECTIVES OF STUDY

This research aims to develop a real-time phishing link detection system using machine learning algorithms. The specific objectives are to:

1. Review existing phishing-detection systems to analyse their key features.
2. Collect and preprocess an appropriate dataset for phishing detection.
3. Design an enhanced system architecture that integrates all necessary components for the system.
4. Implement the proposed solution by training the design model architecture and developing core modules.
5. Test the system against defined evaluation criteria.

1.4 SIGNIFICANCE OF STUDY

The development of the proposed system, a single and cross-platform solution for detecting phishing links has far-reaching consequences for various stakeholders. Significance to:

i. Every Internet User

The proposed system provides protection that is seamless across the browsers and from browser to desktop applications, which mitigates the phishing risk to individuals. Real-time sandbox preview and contextual education promote safer browsing habits and cut down on attack vectors that are usually used by phishers.

ii. Data-Driven Enterprises and Organisations

Those that process sensitive information about customers or employees are well-protected. The ensemble detection algorithms and centralized history tracking part of the proposed system can reduce successful compromises, aid adherence to compliance and minimise incident-response expenses.

iii. Security Teams (SOC Analysts)

Security Operations Centers will have a flexible and integrable framework to monitor and alert on phishing. The API has a low latency, filters can be customized and the dashboard metrics speed up the triage process, eliminate false positives and bring protection to the mobile and desktop environments.

iv. Application Developers and Platform Providers

The proposed system illustrates an Electron-based overlay and RESTful integration approach, providing a template for the integration of phishing defenses into current products, such as browser extensions, native applications or messaging.

v. Researchers and Future Innovators

The information, patterns and evaluation approach developed in this project can be adopted as a reference in the further evolution of anti-phishing research. Understanding how well hybrid ML can perform, and the feasibility of sandbox integration, along with education effectiveness for users, will drive next generation detection methods and user-centric security products.

1.5 SCOPE OF THE STUDY

This research is based on an accurate phishing detection system which can help to protect the user's data on various digital platforms. The system will be mitigating the attack surface of the phishers, lowering the risk of phishing attacks.

The research will include an educational component in the system's interface to enhance security and user experience. This will actively educate users on phishing techniques and will provide real-time detection that will lead to safer behaviors online.

Additionally, this research will study how to incorporate a real time threat analysis through machine learning that makes the system adaptable to new phishing strategies and decrease false alarms. The project will develop a solution for phishing risk mitigation that is dynamic, adaptive and user-friendly**.**

1.6 LIMITATIONS OF THE STUDY

1. Cross-Platform Integration Challenges: There will be difficulties in developing the system to work across multiple platforms due to constraints like compatibility issues, and the nature of phishing attacks may come across multiple platforms.

2. False Positives and False Negatives: Since no system will be right all the time, there will be false positives (legitimate sites flagged as phishing sites) and false negatives (phishing sites failed to be detected as such) detected occasionally.

3. Resource Constraints: Essential resources like electricity, adequate development time, accurate and large dataset, continuous updates and maintenance will be need to optimize the system for maximum efficiency.

4. User Compliance: The system is being developed with the intention that users will utilize it correctly but that will not always be the case. Users may ignore security warnings and visit flagged phishing sites or they may disregard the educational security tips.

5. Evolving Nature of Phishing Attack: The ever-changing landscape of phishing and cybercrime brings rise to the possibility of the proposed system eventually fading into obsolescence.

1.7 OPERATIONAL DEFINITION OF TERMS

Phishing: It is a social engineering tactic that is used to describe the cybercrime of tricking individuals/organisations into revealing sensitive information.

Phishing Link: This is a URL(Uniform Resource Locator) that hosts a phishing site. These links are sent to unsuspecting user to trick them into providing that confidential information on the phishing site.

Phishing Sites: These are fraudulent sites that mimic legitimate or secure site with the purpose of tricking users into giving up their sensitive information.

Phishing Detection System: This is a system that detects and flags potential phishing attempts based on a variety of detection algorithms

Machine Learning: a branch of artificial intelligence that enables systems to learn patterns from gathering and analysing data, apply the learned patterns to new situations, and make decisions or predictions based on experience, often improving their performance over time without being explicitly programmed.

Dataset: A collection of structured and organised information used to train and evaluate a model.

Accuracy: it is the systems ability to correctly detect phishing attempts. It is measured by the number of correct predictions divided by the total numbers of prediction and expressed as a percentage.

SSL/TLS: SSL (Secure Sockets Layer) and TLS (Transport Layer Security) are cryptographic protocols designed to secure communication over the internet.

CHAPTER TWO

LITERATURE REVIEW

2.1 Introduction

This chapter presents a thorough analysis of the literature that is most relevant to the research top, as it aims to depict for the researcher and the reader the background, progress and weaknesses of this field. It also sets the tone for the importance of the current research to show what has already been achieved and what is still to be answered. In this section, students will research and analyze papers from the scientific literature, research papers, and project presentations from other years, to identify the advantages and disadvantages of the methods used and to identify gaps in the knowledge. In this way it lays the groundwork for the study provided and ensures that the new research does not duplicate the previous research.

2.2 Theoretical Background

It explains the principles of the modern phishing link detection based on hybrid machine learning approach. First we define phishing and how it has developed over the years: from bulk email to spear phishing, PhaaS and AI generated attacks; and then look at traditional defenses (blacklists, heuristics, sandbox previews, gateway filters). We then review the basics of ML techniques including supervised (decision trees and SVMs), unsupervised (clustering and autoencoders), and deep learning (CNNs and RNNs), as well as their benefits. We are introducing feature engineering for URL strings, domain and certificate metadata, HTML content and sandbox behaviours. Finally, hybrid pipelines and ensemble models that fuse these in order to gain high accuracy and robustness to varying threats are discussed. The following subsections delve into the main areas relevant to the research:

2.2.1 Evolution of Phishing Attacks

Phishing started as a hammer, a mass mailed email pretending to be a bank or lottery organiser, and using fear or greed to get users to turn over their credentials. As time passed, the attackers

evolved in their tactics. Spear phishing attacks use publicly available information (social media accounts, company directories) to create a message that is very customized. Today, non-technical criminals are able to hire phishing kits and phishing infrastructure via so-called Phishing as a Service (PhaaS) platforms. The latest kind of AI phishing is based on large language models that create grammatically correct, contextually relevant phishing messages on a large scale. Many of the signs of phishing links have also changed:

i. IDN homographs replace characters of other alphabets (e.g. “appIe.com” instead of “apple.com”).

ii. Typo squatting involves the registration of domains that either have a character different from the original or have a dot (.) that is in the wrong position.

iii. If you use HTTPS with a self signed or expired certificate you will see a padlock icon.

iv. URL shorteners and redirection chains hide the end page. v. Asset cloning is copying real page images/text to trick on the surface.

2.2.2 Conventional Detection Methods

i. **Blacklist/Whitelist**: Keeps up-to-date lists of known malicious or safe sites in near real-time for rapid lookup. Extremely low latency after being cached, but anything new or not reported to the solution is not known until it's manually added, and maintenance on the list can be a strain on security staff.

ii. **Heuristic Rules**: Applies heuristic rules such as human designed patterns, URL length thresholds, special character frequency, subdomain depth and suspicious keywords to score links. Provides fast adaptation to new threats, but attackers may be able to escape rules by making small changes to their URL, and rules can be tuned to be more sensitive to false positive/false negative trade-offs is difficult.

iii. **Sandbox Previewing**: Runs urls under a different browser with sandbox mode enabled, recording everything from outbound network calls to redirect chains, DOM mutations, as well as the execution traces of JavaScript code. Detects hidden payloads and time delayed payloads not detected by static filters, but each analysis adds seconds of delay and CPU/Memory usage.

iv. **Browser- and Gateway-Based Filters**: They are installed at the endpoint or network edge, and feature signature matching, light HTML structure verification, and third party domain/IP reputation services to block threats on the fly. They ensure protection for entire user populations in a central manner, but require current signatures and can produce false positives for modern, dynamic web applications. This involves using machine learning to detect phishing attempts. This includes implementing machine learning for phishing detection.

2.2.3 Machine-Learning in Phishing Detection

Complex patterns are discovered across large datasets of benign and malicious URLs using machine learning models to automate the discovery. Feature Engineering is a key element to their success, in that it turns into raw data into informative inputs:

i. Lexical features are features of the string of the URL itself (e.g., length, number of digits, number of special characters, distribution of tokens).

ii. Host based features check domain age (WHOIS creation date), DNS TTL values, and registrar metadata to identify newly registered domains and/or suspicious domains.

iii. Content based features are used to break apart HTML artefacts, to count the number of iframes, and to compute the number of requests for resources from outside the page.

iv. Certificate based features include determining if a certificate is valid, is issued by a trusted CA, or is self signed, and determining the validity period of the certificate.

v. With these features, various model classes excel:

vi. Supervised algorithms (decision trees, support vector machines, logistic regression) learn simple decision boundaries from labeled examples. They're readable, but need to be regularized carefully to prevent overfitting.

vii. Unsupervised approaches (clustering, one class SVM, autoencoders) are able to identify anomalies without the use of labeled data which can be used for identifying new phishing strategies, but require careful tuning of the threshold to ensure that they are both sensitive and specific.

viii. Raw URL sequences of characters, or snapshots of web pages can be fed to deep learning models, based on convolutional or recurrent neural networks, which can be used to automatically learn hierarchical representations; such models are able to absorb subtle sequential or visual clues, but require large training sets and a lot of computing power..

2.3 Review of Related Works

There have been a number of systems that this research is related to in one way or another. Some of these systems will be discussed in detail here to show their relevance and relation to the research topic:

Catal et al. (2022) investigated phishing detection based on deep learning with the complex pattern recognition of deep learning to identify phishing. They employed a deep neural network method, which provided high accuracy at the expense of large datasets and high computation. Their method enhanced detection but was not straightforward to implement in real time due to the high computation demands. This study is consistent with my exploration of machine learning-based phishing detection but suggests a trade-off between accuracy and ease of implementation.

Castaño et al. (2021) studied phishing detection techniques, distinguishing between content-based and hybrid approaches and reviewing the text, images, and URL style used. The hybrid method provided better phishing detection; however, the method encountered problems owing to changing content of the websites.. In this respect, this research supports my arguments about the advantages of multi-feature analysis but mentions the limitations of a static method in adapting to new methods of phishing.

Abuzuraiq et al. (2020) presented an intelligent phishing detection system based on fuzzy logic and machine learning. Their solution aimed at enhancing the accuracy of classification, considering the nature of uncertainties of phishing data, i.e., slightly modified URLs or misleading content. The system showed encouraging performances, especially in the low false negative rate, yet the authors declared that it was still susceptible to advanced obfuscation techniques and needed to be refined further to minimise the false positive rate. This work can be applied to my research, as this paper talks about the difficulty of having a high precision and recall in phishing detection. One of the possible solutions to these difficulties is the application of fuzzy logic, and I will consider this point in my work.

Nguyet Quang Do et al. (2022) conducted a systematic literature review of deep learning-based phishing detection, establishing a taxonomy of existing methods and challenges. Although their study provided useful information about deep learning applications, it lacked experimental validation. The review pertains to my study in the sense that it explains gaps in the integration of multiple deep learning methods and the need for integrated frameworks.

Thakur et al. (2023) reviewed deep learning techniques for phishing email detection, contrasting the performance of CNN and LSTM models among others. They presented strengths and weaknesses of current techniques but identified real-world application and variability of datasets as constraints. Their study is applicable to my research, and it indicates the need for adaptable models to thwart dynamic phishing activities.

Çatal et al. (2022) performed a systematic review of the literature on deep learning for phishing detection, analysing 43 journal articles. They identified common algorithms and frameworks but also the lack of semi-supervised and unsupervised research. This study supports my project with its comprehensive summary of deep learning techniques and the highlighting of the promise of hybrid approaches.

According to Reddy et al. (2023), they devised a hybrid phishing detection model using machine-learning and behavioural analytics mechanisms. The proposed model makes detection more efficient by integrating user behaviours, such as typing speeds and mouse activities, into the investigation. The authors of the research report that their model is superior to previous technologies. Nevertheless, they claim that the efficiency of their model mainly depends on the quality of the behavioural data, and the technology has to be tested in various conditions. This piece of research is suitable for my particular project due to its focus on the incorporation of behavioural analytics into machine-learning detection models. The findings will be used to study hybrid systems based on different data types used for phishing identification.

Finally, according to Ahmadi et al. (2024), they have presented an extensive review of artificial intelligence-based phishing identification systems, having analyzed over 130 studies. They have noticed that the topic requires further investigation. The researchers conducted a thorough review; however, the researchers did not provide information on practical implementation. This review will be instrumental in my work because it highlights the necessity of scalable AI systems.

Sahingoz et al. (2024) suggested DEPHIDES, a deep learning-based phishing detection system, with an accuracy of 98.74% using CNNs. It was a specialist in URL-based analysis, but lacked content and behavioral features of webpages. This study helped with my project in two ways: 1) it shows results on how effective CNNs can be and 2) it offers a big dataset to train the models.

In this research, Kyaw et al. (2024) provided an overview of deep learning techniques for detecting phishing emails, creating a taxonomy of techniques, and examining their flexibility. They focused their survey on the importance of good quality datasets and identified a lack of existing research. This survey is helpful in my research because it highlights approaches and techniques for modelling resilience and improving the quality of data sets.

Ji and Lee (2024) tested phishing detection systems that are based on images and discovered that these systems struggle against smart attacks. They showed that those systems are exposed to small changes in images and need much computation power. The article is useful to learn about image-based detection and gives an idea on how to implement adversarial training to improve the resilience of systems.

Prajapati et al. (2023) used machine learning algorithms like Random Forest and SVM for phishing email detection using TF-IDF and n-grams for feature extraction. Their work demonstrated the success of supervised learning but lacked discussion on adversarial or image phishing. Their contribution is aligned with my contribution and proposes to incorporate deep learning models to improve detection.

Azeez et al. (2021) suggested a whitelist-based method of phish detection that shows high precision and low data generalization. Their work provided an innovative concept of efficient computational methods that can be utilized to increase the precision of my detecting system.

Ozker & Sahingoz (2020) suggested a machine learning model with content-based phishing detection system considering both textual and visual features. Complete feature analysis was selected as a priority with the assertion of computational overhead. The research will benefit my project because it expounds on how content analysis can be incorporated within existing frameworks.

A hybrid mobile app phishing detection mechanism was proposed by Alzamil et al. (2020) via a combination of blacklist and whitelist techniques. Their method was capable of decreasing

the false positives but depended on updating the lists frequently. The research confirms that my work is accurate in demonstrating that hybrid solutions are possible and adaptive algorithms are required.

Akande et al. (2023) suggested SMSPROTECT, a rule-based machine learning mobile phone application to identify smishing. They relied on the user's discretion to make final decisions, which can be wrong. The current study can be applied to my research problem of mobile phishing detection, and this section will provide recommendations on how the user effort can be reduced.

Ujah-Ogbuagu et al. (2024) suggested a CNN-LSTM hybrid model with high accuracy and computational complexity in detecting URLs in real-time. Their work shows that sequence learning is a vital part of phishing detection and guides my research with regard to optimisation of computational efficiency.

Do et al. (2024) suggested a hybrid system of deep learning classifiers and pre-trained transformers for the detection of phishing URLs. Their model did better with state-of-the-art models but could not generalise well to new models. According to this study, hybrid models and contextual analysis hold promise in my study.

Sivakumar Nagarajan (2024) investigated AI-based phishing detection techniques, proposing CRF+LDA for feature extraction and AdaBoost for classification. Their research was on hybrid AI techniques, but without regard to real-time performance. This research is aligned with my interest in feature engineering and model interpretability.

Shahrivari et al. (2020) benchmarked a number of machine learning models for phishing detection, with significant features for classification. Although their study was limited by dataset size and the interpretability of neural networks, their work informs my research by highlighting successful models and the potential for hybrid approaches.

2.4 Summary of Literature Review

Author (Year)	Methodology Used	Contribution	Limitation
Catal et al. (2022).	Deep learning	Leverages deep learning to capture complex phishing patterns.	Requires large datasets; high computational demands.
Castaño et al. (2021).	Content-based analysis & hybrid techniques	Uses content analysis (text, images, URL structure) for phishing detection.	Vulnerable to dynamically changing website content.
Abuzuraiq et al. (2020).	Machine learning with fuzzy logic	Achieves high classification accuracy using fuzzy logic.	Suffers from false positives and struggles with obfuscation.
Nguyet Quang Do et al. (2022)	Systematic literature review analysing 81 selected papers. Development of a taxonomy for phishing detection methods based on deep learning.	Provides a comprehensive survey of deep learning techniques applied to phishing detection. Proposes a taxonomy categorising existing methods and identifies current challenges in the field. Suggests future research directions to enhance phishing detection mechanisms.	Lacks experimental validation for the proposed future directions. Does not present a unified framework integrating various deep learning approaches for phishing detection.
		Provides a comprehensive review of deep learning techniques for	

		phishing email detection. Evaluates the effectiveness of algorithms like CNNs and LSTMs in identifying phishing attempts. Identifies strengths, weaknesses, and areas for future research in the field.	
Thakur, K., Ali, M. L., Obaidat, M. A., & Kamruzzaman, A. (2023)	Systematic literature review analysing existing studies on deep learning applications in phishing detection.	Conducts a systematic review of 43 journal articles on deep learning applications in phishing detection. Identifies commonly used algorithms, data sources, and frameworks. Highlights challenges and proposes future research directions.	Does not include experimental validation of discussed techniques. Limited focus on real-world application challenges and dataset diversity.
Çatal, Ç., Giray, G., Tekinerdogan, B., Kumar, S., & Shukla, S. (2022)	Systematic literature review analysing existing studies on deep learning applications in phishing detection.	Provides a comprehensive survey of deep learning techniques applied to phishing detection. Proposes a taxonomy categorising existing methods and identifies current challenges in the field. Suggests future research directions to enhance phishing detection mechanisms.	Limited to supervised learning approaches; minimal exploration of semi-supervised or unsupervised methods. Lack of experimental validation for proposed future directions.
Reddy, P. S.,	Integration of	Proposes a comprehensive anti-	Specific details on the

Bansal, A., Ali, M. K., & Jha, A. K. (2023)	machine learning algorithms with behavioural analysis to detect and prevent phishing attacks.	phishing system combining machine learning techniques with behavioural analysis. Aims to enhance the detection and prevention of phishing attacks by leveraging advanced technologies.	implementation and performance metrics of the proposed system are not provided in the available summary.
Ahmad et al. (2024)	Systematic literature review analysing studies on AI applications in phishing detection.	Conducts a comparative analysis of over 130 articles on AI-based phishing detection models. Identifies challenges and gaps in existing literature. Provides a roadmap for future research in phishing detection.	Focuses primarily on literature review without experimental validation. Limited discussion on real-world implementation challenges of AI models in phishing detection.
Sahingoz, O. K., Buber, E., & Kugu, E. (2024	Implementation and evaluation of deep learning models (ANNs, CNNs, RNNs, BRNNs, attention networks) for phishing URL detection.	Developed DEPHIDES, a phishing detection system utilising five deep learning models. Compiled and shared a dataset of ~5 million labelled URLs. Achieved 98.74% detection accuracy using CNNs.	Focuses solely on URL analysis; does not incorporate webpage content or behavioural data. Real-world applicability and performance may vary.
Kyaw, P. H.,	Systematic	Conducted a systematic literature	Limited to the analysis

Gutiérrez, J. A., & Ghobakhlou, A. (2024)	literature review following PRISMA guidelines to	review of 33 papers on DL-based phishing email detection. Developed a taxonomy of DL methods, analysing their effectiveness and limitations. Identified adaptability issues in current detection models to evolving phishing behaviours.	of existing literature without experimental validation. Emphasises the need for large, high-quality datasets, highlighting a gap in current research resources.
Ji, F., & Lee, K. (2024)	Uses deep learning techniques to analyse website visuals. Conducts adversarial testing to assess model robustness.	Investigates phishing detection using visual similarity analysis**.** Evaluates how well deep learning models can identify phishing sites based on images and layout. Tests robustness against adversarial attacks that manipulate website visuals.	Visual similarity models can still be deceived by subtle image modifications**.** Requires high computational power, making real-time detection challenging.
Prajapati, P. et. al (2023)	Implements Random Forest, SVM, Naïve Bayes, and Decision Trees. Uses TF-IDF and n-grams for text-based feature extraction.	Evaluates multiple machine learning algorithms for phishing email detection. Uses TF-IDF and n-grams for feature extraction. Measures performance using accuracy, precision, recall, and "F1" score.	Relies on static textual features, which can be bypassed by sophisticated phishing techniques. Lacks exploration of adversarial attacks or image-based phishing.

Azeez, N. A., Misra, S., Margaret, I. A., Fernández-Sanz, L., & Abdulhamid, S. M. (2021)	Detailed analysis comparing visual and actual links.	Introduces an automated whitelist-based method for phishing detection. Achieves high accuracy (average 96.17%) and a true positive rate of 95.0%. Efficient in computational resource utilisation.	Accuracy varies across different datasets, indicating challenges in generalisation.
Ozker, U., & Sahingoz, O. K. (2020)	Machine learning; Content feature extraction from web pages	Presents an anti-phishing system based on machine learning that examines content of web pages. Implements several models including decision trees, SVM, and neural networks. Focuses on both textual and visual content properties for classification.	Potential high computational overhead. Detection effectiveness dependent on training data quality.
Alzamil, Z. A., Aljurayyad, A., Almajally, M., & Alfreddi, B. (2020)	Analyses URLs accessed via mobile apps. Cross-references URLs against dynamically updated blacklists and whitelists.	Describes a hybrid approach in phishing detection which uses both blacklist and whitelist methods for mobile applications. Aims at improving accuracy of detection while lowering the amount of false positives.	Inherits limitations of blacklist (ineffectiveness against zero-day attacks) and whitelist (need for continuous updates) methods.
Akande, O. N.,	Rule-based	Developed a mobile application	Effectiveness may be

Gbenle, O., Abikoye, O. C., Jimoh, R. G., Akande, H. B., Balogun, A. O., & Fatokun, A. (2023)	approach	(**SMSPROTECT**) for detecting and preventing smishing attacks. Utilises a rule-based machine learning model to classify SMS messages.- Provides users with analysis results and decision-making capabilities regarding message retention.	limited against new smishing techniques not covered by existing rules. Relies on user judgment for final message disposition, which could introduce errors.
Ujah-Ogbuagu, B. C., Akande, O. N., & Ogbuju, E. (2024)	Hybrid Deep Learning Approach	Developed a hybrid CNN-LSTM model for real-time detection of malicious URLs. Achieved high detection accuracies using publicly available datasets. Demonstrated the effectiveness of combining CNN and LSTM for phishing detection.	Potential challenges in real-time deployment due to computational complexity. Dependence on the quality and diversity of training datasets.
Do, N. Q., Selamat, A., Fujita, H., & Krejcar, O. (2024)	Deep Learning Technique	Introduces an integrated model combining RasNet, TCMA, and MPNet for phishing URL detection. Achieves high detection accuracy across multiple datasets, outperforming existing models.	Increased computational complexity may hinder real-time deployment.
Sivakumar Nagarajan	Multi-stage AI (CRF, LDA,	Proposed CRF+LDA for phishing feature extraction;	Limited evaluation of real-time performance;

(2024)	AdaBoost); Association Rule Mining for Wi-Fi phishing.	introduced AdaBoost for classification. Highlighted Wi-Fi phishing via Association Rule Mining.	AdaBoost may struggle with noisy data.
Shahrivari et al. (2020)	Machine Learning; Comparison approach	Evaluated multiple ML models for phishing detection; identified key features for classification.	Small dataset; neural networks were slow and lacked interpretability.

2.5 Research Gap

Considering the advances in deep learning applied to phishing detection (Çatal et al., 2022) and hybrid ML behavioral approaches (Reddy et al., 2023), still many challenges exist. Most of the successful systems heavily depend on extensive and homogenous datasets that are rarely available in practice; thus, it cannot be generalized for small or unbalanced cases. Other studies focus mainly on unilateral solutions that employ either links or texts as a source of information without trying to create a more sophisticated multimodal approach. In addition to this, a lot of existing AI systems are not explainable and transparent. Thus, the new system that we propose will be a combination of URLs, content, sandbox, and certificate features in one model, validated on the real data and providing explanations.

CHAPTER THREE

SYSTEM ANALYSIS AND DESIGN

3.1 Introduction

In this chapter, the analysis and design of the proposed system are explained. It describes the system design process and the method employed in it, such as Object-Oriented Analysis and Design (OOAD), and explains the latter applied in the course of the study.

3.2 System Analysis

System analysis is a step-by-step procedure of decomposing a proposed system to investigate its parts and the way they interface with other parts. This research study undertakes a detailed examination of proposed system. It involves analysis of the systems which are already in place, finding out their weaknesses and proposing changes.

Detailed design entails development of various any types of charts including use case charts, activity charts, class charts as well as sequence charts. The essentiality of the diagrams is that they provide a graphical representation of the system components and their interrelationships and improve the understanding of the system in the mind of someone.

Other than this, system analysis process also matters in finding out what can be improved. Through analysis in this process, it will be ensured that the outlined system is stable, efficient and possesses the strength to guard against weaknesses of the current systems.

3.2.1 Analysis of the Existing System

Currently, organisations and internet users employ the following phishing detection systems: blacklist databases, security warnings appearing in the browser, signature-based detection systems, and manually configured filtering rules. Blacklist services are used by popular web browsers like Google Chrome, Microsoft Edge and Mozilla Firefox to block access to known phishing websites. To help detect phishing, several machine learning based phishing detection systems have also been developed. These systems generally look for patterns in the URL

structure, domain attributes, the content on the website, HTML elements or network-related data to determine whether a website is legitimate or phishing. Recently, deep learning techniques have been applied to learn features and boost classification performance. Yet there are still a number of solutions that are in use that rely on one source of information. Some systems just look at URL properties, while others look at page content only or page screenshot analysis. In addition, many such models are already in place and are "black-box" models, meaning that they act as a predictor without explaining which factors might affect their predictions.

3.2.2 Limitations of the Existing System

The analysis of existing phishing detection systems revealed several limitations:

1. The majority of existing systems rely on a blacklist database and cannot deal with new phishing sites.

2. There are systems in place that give a prediction without any transparency, and the user has to understand why it has been classified like that.

3. Single-modality systems can be defeated if the attacker exploits certain characteristics of the website to avoid being detected.

4. Most research applications are not easily accessible to end-users as web apps on the internet.

3.2.3 Analysis of the Proposed System

The intended real-time phishing detection system had to be scalable, maintainable and efficient for threat analysis, and was designed with a modular approach featuring four Python modules and a web based interface. The system is integrated in a single system that includes feature extraction, machine learning classification, geolocation intelligence and user interaction. The feature extraction module extracts 24 URL-based features and 20 email-based features from the inputs provided and identifies the phishing features without accessing any of the external websites. The machine learning classifier is fed with extracted lexical, host-based, and content-

based features. The model training part is based on Random Forest model which is used for its ability to deal with complex feature relationships and classification and trains models for URL and email phishing detection.

Based on the operational analysis, it can be confirmed that the Flask-based application has been able to successfully execute the phishing assessment in real-time, with specific API endpoints for URL analysis and email analysis. The system is able to automatically extract relevant features from the message, classify it with the trained Random Forest models, assign the risk level, and deliver explanatory detection signals in addition to geolocated information via domain-to-IP resolution. By adding geolocation intelligence, suspicious domains or email senders are more likely to be identified as their likely geographic location. In addition, the web interface is HTML-based, allowing end-users to get instant feedback in an intuitive interface with risk visualisation and location mapping on a dashboard. In conclusion, the system meets the study goals by integrating machine learning for phishing detection with geographical location and real-time user interaction into a viable and operational cybersecurity solution.

3.2.4 Objectives of the Proposed System

The proposed system has the following objectives:

1. Detect Zero Day and Fast Morphing Phishing URLs

The manual curation, and periodic updating, of the list of blacklisted attack domains means that new attack domains, or those that change quickly, can get through until the next update. Objective: Track in real time new phishing URLs that haven't been seen before, with a hybrid machine learning ensemble combining lexical, host based, content based, sandbox derived and certificate features, with a window of vulnerability for zero day exploits reduced to zero.

2. Give Contextual Explanations and Educational Guidance

The Linked to: Google Safe Browsing's warning screen shows “danger” with no educational value, since no indication is provided of what caused the warning to appear for the URL. Goal:

Provide a simple summary of the significant indicators for each blocked or flagged link (for example, an SSL certificate that has expired, a newly registered domain, an obfuscated URL token) and short tips on how to recognize and avoid similar phishing tricks in the future.

3. Provide a way to enable Safe Preview through an Integrated Sandbox Environment

There is no sandbox or preview function in the existing system, and the user must either delete the link or do what he wants without knowing what he's doing.

Goal: Ingest any URL provided, whether safe or not, into a separate, cross platform sandbox that can render and analyze it without revealing the user's browser or system, and record any activities within the sandbox for later analysis.

4. Deliver a Cross Platform, User Friendly Desktop Application

Many solutions are available today as browser extensions or web services, which means that there is only a partial device and operating system integration and user experience.

Objective: Package up the detection engine and sandbox into a native desktop application, with a user-friendly UI around real time alerts, feedback submission and educational modules, reaching the widest possible audience and achieving the highest possible user adoption rates.

3.2.5 System Requirements Specification (SRC)

System requirement is the functionality or process and quality of requirements that a system ought to have. These system requirements are in two types; the non-functional requirements and the functional requirements.

A. Functional Requirements

These requirements refer to the functionalities or processes that the phishing detection systems must be able to perform. They include:

a. Users should be able to create secure accounts and login

b. Authenticated users will be able to view and edit their profile at any given time.

c. Users will be able to submit urls for phishing analysis and receive a verdict based on significant url and site features.

d. Users will be able to preview scanned urls in an isolated sandbox modal, to prevent data leakage

e. The system will retain the history of all scan urls, allow filteration, search and pagination of entries

f. The system will maintain and preserve all records of URL analysis in a dashboard accessible only to account users.

g. The system shall have a section that allows users to take a multi-question phishing awareness test.

B. Non-Functional Requirements

The non-functional requirements to the requirements that specify the attributes or qualities that the proposed system should have; it describes how the system should operate. The Non-functional requirements for the phishing detection system include :

a. The electron desktop build must run on different operating systems and be responsive in different devices.

b. URLs rescans leverage feature-caching to optimize scan time

c. Scan history and profile date will be accessibe only to owning user

d. The Scanned URLs can be viewed in an isolated sandbox to prevent data leakage and storage of website cache on local device.

e. Error and info messages must be clear and concise

f. The codebase architecture must use modularity in it's components to ensure proper scalability.

3.3 System Design and Methodology

The Research Project used a machine-learning based implementation and Object-Oriented

Analysis and Design (OOAD) methodology that would allow for automatic detection of phishing websites and phishing emails to create a proposed Real-Time Phishing Detection System. The machine-learning based implementation methodology utilized the extraction of features from both phishing emails and phishing websites, with a Random Forest machine-learning model for classification to achieve the project goals. The project followed an implementation workflow that consisted of the following stages:

1. Create a dataset of phishing emails and phishing websites.
2. Prepare the dataset for data preprocessing.
3. Extract and engineer features from the dataset.
4. Train the Random Forest model using the datasets to classify phishing emails and phishing websites.

While the Object-Oriented Analysis and Design (OOAD) is a comprehensive development approach that combines the principles of object-oriented programming with visual modeling to shepherd a project from initial requirements through to final implementation. In OOAD, stakeholders work together to capture and formalize system requirements, then break the system down into discrete “objects” or classes that encapsulate both data and behavior. All meaningful relationships—inheritance, associations, dependencies—are mapped out so that the architecture reflects real-world interactions.

Central to OOAD are Unified Modeling Language (UML) diagrams—class diagrams, sequence diagrams, activity diagrams—and detailed use-case descriptions. UML provides a standardized, graphically rich way to document how each component interacts, while use cases describe how end users will engage with the system. During the implementation phase, developers translate these designs directly into object-oriented code, ensuring that the software structure mirrors the analysis models.

OOAD's strength lies in its ability to manage complexity: by decomposing large systems into well-defined objects and clear interaction models, it promotes modularity, reuse, and easier maintenance. This combination of rigorous analysis, structured design, and visual documentation made OOAD the ideal methodology for our project.

3.3.1 Dataset Collection and Description

Data for this research came from publicly accessible sources that provide phishing and non-phishing sites and mail. The data includes both phishing as well as non-phishing examples, which were used to create and assess a machine learning algorithm.

The dataset is made up of two sections:

(i) a section of website and mail URLs for detection of phishing websites.

(ii) A section of email addresses for detection of phishing emails.

Each dataset has labelled data indicating whether or not the example is phishing or legitimate. The datasets were chosen due to their size and the diverse examples they represent, providing accurate representations of the types of phishing attacks and legitimate communications that people receive so that they can be used to create models to assist in developing social media detection tools.

3.3.2 Data Preprocessing

In order to enhance the overall quality of the data and make them suitable for training machine learning applications, a number of pre-processing techniques must be applied to the datasets during pre-processing. The types of pre-processing performed on the datasets include:

1) Load and validate the dataset.

2) Remove any duplicate entries.

3) Check for and handle missing values.

4) Clean and format the data.

5) Separate the target variable from the features.

6) Where applicable, balance the datasets.

7) Split the dataset into training and testing subsets.

The datasets were separated into training and testing subsets so that independent evaluation of the performance of the machine learning models can occur. The training subset was used to develop the machine learning model while the testing subset was used to assess the performance of the machine learning model.

3.3.3 Extraction and Engineering of Features

The feature extraction module will automatically extract features that are indicative of phishing from submitted URLs and email content.

URL Features

The URL extraction module will extract 24 features from submitted URLs in numeric form. Some examples of extracted features include:

i. Length of the URL
ii. Number of periods
iii. Number of slashes
iv. Number of hyphens
v. Number of digits
vi. IP address presence
vii. Number of special characters
viii. Number of suspicious keywords
ix. Usage of HTTPS
x. Length of the domain name

Using these features, it is possible to identify common patterns of suspicious URLs that indicate phishing activity.

Email Features

The email extraction module will extract 20 features from the subject, body, and sender of each email in numeric form. Some examples of extracted features include:

i. Length of the subject line
ii. Length of the body of the email
iii. Number of hyperlinks in the email
iv. Presence of urgent language
v. Presence of financial text
vi. Number of words with capital letters
vii. Total number of special characters
viii. Features of the domain of the sender
ix. Number of suspicious words
x. Presence of indicators of attachments

The numerical feature vectors produced by both the URL feature extraction and email feature extraction modules will then be used as input to the machine learning classifier.

3.3.4 Random Forest Model Design

The design of the Random Forest model is implemented in the following steps: The model to detect phishing is Random Forest algorithm. Random Forest is an ensemble machine learning method, which consists of several decision trees and is used to increase the accuracy of classification and to minimize overfitting. The choice of the algorithm was made for the following reasons:

i. Works well with structured phishing datasets.
ii. It works with non-linear relationships in features.
iii. It offers good classification accuracy.
iv. Requires minimal computation.
v. It works well with a variety of features.

In the model training process, the following steps were done:

a. Feature extraction.

b. Data preprocessing.

c. Dataset partitioning.

d. Model training.

e. Model validation.

f. Performance evaluation.

Two independent Random Forest models were trained:

- URL Phishing Detection Model
- Email Phishing Detection Model

The trained models were serialized and saved as:

- url_model.pkl
- email_model.pkl

These models were subsequently integrated into the deployed web application for real-time phishing detection.

3.3.5 Geolocation Module Design

Geolocation module was added to enrich domain information with geo intelligence for source domain of submitted URLs and sender address of submitted emails. This module carries out the following operations:

i. Extract domain name.

ii. Back to Domain to IP address.

iii. Query geolocation service.

iv. Retrieve geographic information.

v. Show users the results of location.

The geolocation information consists of:

a. Country Region City Latitude Longitude Internet Service Provider (ISP)

b. This capability adds more context to phishing investigations by delivering more context information about suspicious sources.

3.3.6 System Architecture Design

The architecture section shows the descriptions of:

Presentation Layer

Implemented using:

- HTML
- CSS
- JavaScript

Responsibilities:

- URL submission
- Email submission
- Displaying phishing risk results
- Geolocation visualization
- User interaction

Application Layer

Implemented using Flask.

Responsibilities:

- Feature extraction
- URL classification
- Email classification
- Geolocation processing
- API management
- Response generation

Data Layer

Responsible for:

- Trained model storage
- Prediction records
- Application logs
- Configuration files

The architecture promotes modularity, maintainability, scalability, and efficient real-time phishing detection.

3.4 Unified Modelling Language

Unified Modelling Language (UML) diagrams serve as standardized, visual representations of a system, employing a common set of symbols and conventions to communicate design ideas. Software engineers and developers rely on UML to create clear, computer-aided sketches of a system's structure and to illustrate how its components are expected to interact. By combining use case diagrams with activity and class diagrams, UML provides a comprehensive view of both the system's classes and the actors involved, along with the specific operations each performs.

3.4.1 Use case diagram of the proposed system

Use case diagrams are divided into four main components: actors, which symbolize the external users or systems interacting with the system; use cases, which depict the system's functional processes; associations, which show the relationships between the actors and the use cases; and the system boundary, which defines the limits or scope of the system that the actors engage with.

As for the proposed system, there would be 2 actors which are; the Users and the System Admin.

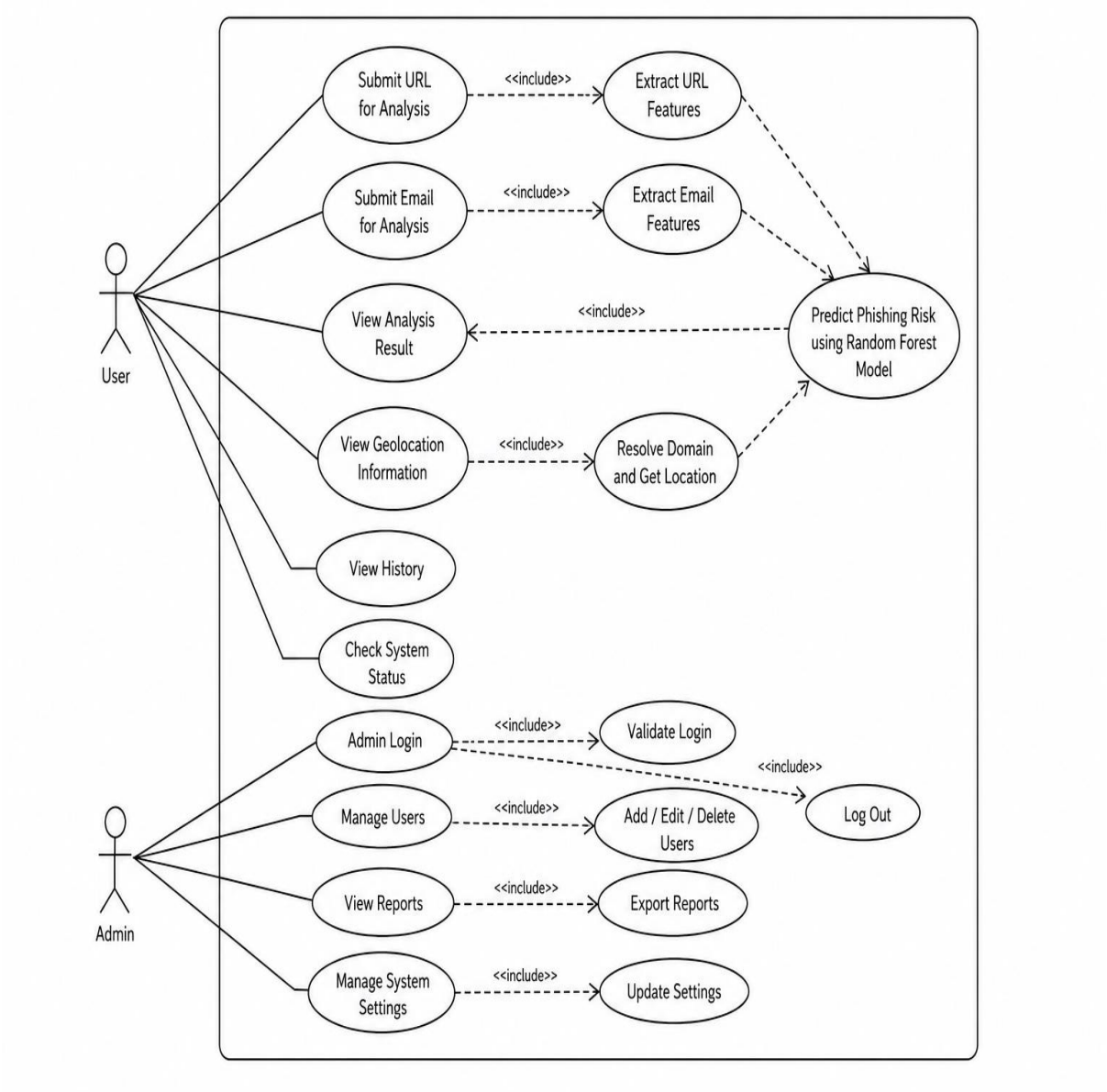

Figure 3.1 Use-Case Diagram of proposed system

The Use Case Diagram describes the two main types of actors interacting within the proposed Real-Time Phishing Detection System - User & Admin - and their functional interactions with the system. Users interact with the system through submission of URLs or email content for analysis; viewing the results of phishing detection; viewing geolocation information; viewing previous analyses; and checking status of the system. When a URL or email is submitted for analysis, the system performs feature extraction and applies the trained Random Forest model to determine the level of risk of being a phishing site.

Admin actors participate in administrative types of activities, including logging into the administrative portal; performing user account management; viewing reports and report generations; and configuring system-related settings. The processes that admin actors perform in managing their administrative activities are executed through the dedicated management functions associated with each of the administrative processes. The «include» relationships indicate that these processes must be executed whenever the system operates, including features such as feature extraction, login validation, generating predictions, exporting reports, etc. Overall, the diagram supports accurate, secure, and real-time detection and management of phishing attacks through the respective actions performed by Users and Admins.

3.4.2 Activity Diagram of Proposed System

An activity diagram is a UML flowchart that visualizes the sequence of actions within a software system. The proposed system employs an activity diagram to illustrate the user workflow—from authentication through scan execution to result review and a general overview of the proposed system's workflow.

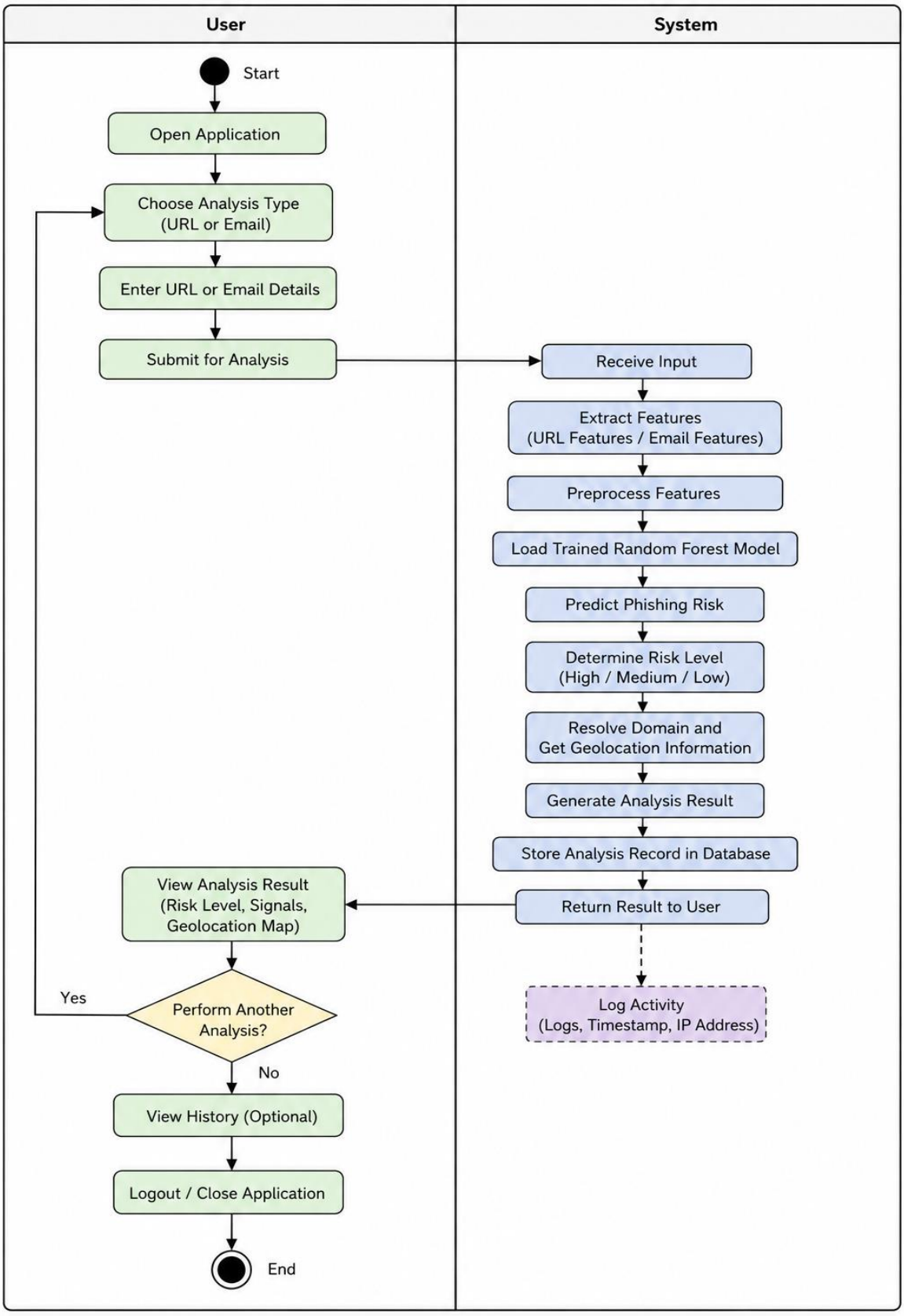

Figure 3.2 Activity Diagram for the proposed systems

The Activity Diagram depicts how the Real-Time Phishing Detection System will behave from the time a user requests an analysis to the time they receive their results. The process of getting an analysis begins when the user opens the application, selects the analysis type (URL or e-mail), enters their information, and submits their request for an analysis. Once the application has received the user's input, it automatically extracts relevant features that may contribute to phishing and pre-processes these features before loading the Random Forest model that has already been trained.

The model then evaluates the features produced from step 2 against its training data to determine the likelihood of prowling activity. Based on this prediction, the system assigns a risk level to the submitted request and retrieves the geolocation information for the URL or the domain name of the e-mail sender associated with the analysis request. The system then creates and returns to the user the results from the analysis, including the phishing classification assigned, the risk level assigned, detection indicators, and geolocation information. While these results are being generated, the system will also create and store data related to the analysis process for logging and monitoring purposes. After the user has viewed his/her results, the user will either conduct another analysis, review previous records, or exit from the application, thus completing the analysis process flow.

3.4.3 Class Diagram of proposed system's

UML class diagrams represent system structure by showing classes, their attributes, methods, and relationships. The proposed system's class diagram models key entities—User, ScanJob, Result, Quiz, Admin, and Settings—and how they interact via associations, inheritance, and data flows.

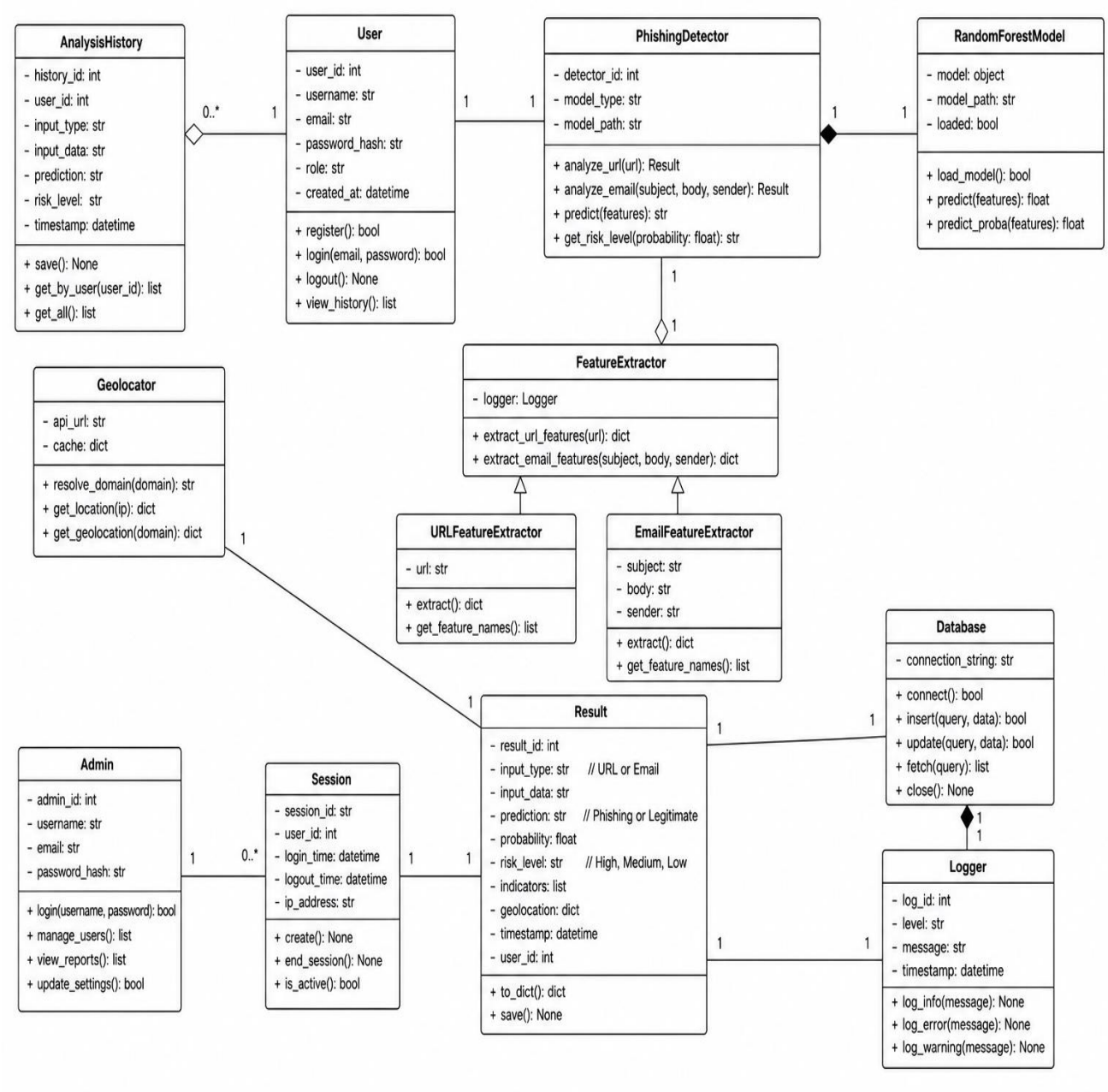

Figure 3.3 Class Diagram for the proposed systems

This Class Diagram provides a visual schematic representation of the structure of the proposed Real-Time Phishing Detection System and the connections between the main components of the system. The two main classes from the User and Admin will be the first two classes of the real-time Phishing Detection System. For instance, both Users and Administrators can submit URLs and email messages to be scanned for phishing. They can then view the results of their previous analysis. However, Administrators will also be able to manage users and their

accounts, configure some of the other settings on the system itself, and generate various types of reports from the data in the system.

The PhishingDetector will be the main processing component of this system. It will handle every step of the processing for a URL or email submitted to be checked for phishing. This will include coordinating the feature extraction process, classifying the features, and assessing the risk of being a phishing attempt. This component will interact with the FeatureExtractor component, which will gather phishing related characteristics from URL or email, and will have two different implementation subclasses, URLFeatureExtractor and EmailFeatureExtractor, that each extract features of that particular type of submission. The extracted features will then be passed to the RandomForestModel component for classification as phishing or non-phishing and for prediction of probability.

The Geolocator component retrieves the geographical information of the domain from the submission, while the Result component will hold the results of the analysis including predicted results, risk assessment, and geographical information. The AnalysisHistory component holds a history of previously run analyses. The Database and Logger components manage the database for data storage and log/display all activity within the system, respectively. Collectively, these components provide a modular and maintainable architecture for the real-time detection of phishing.

3.5 DATABASE DESIGN

The database design models proposed system's's relational data structures. proposed system's uses SQLite relational database to store user profiles, scan results, quiz attempts, and history.

Database Schema for the Proposed System

Users table

Table 1.0 Users table for proposed system's

Attribute	Data Type	Description
Id (PK)	SERIAL	Primary key
Username	VARCHAR	Unique user handle
Email	VARCHAR	User’s email address
PasswordHash	VARCHAR	Hashed password storage
AvatarUrl	VARCHAR	Link to profile image
CreatedAt	TIMESTAMP	Account creation timestamp

ScanResults table

Table 2.0 ScanResults table for proposed system's

Attribute	Data Type	Description
Id (PK)	SERIAL	Primary key
UserId (FK)	INTEGER	References Users(Id)
Url	TEXT	The URL scanned
Mode	VARCHAR	“single” or “batch”

Label	VARCHAR	“phishing” or “legitimate”
Confidence	REAL	Probability score
Timestamp	TIMESTAMP	Scan execution time

QuizAttempts table

Table 3.0 QuizAttempts table for proposed system's

Attribute	**Data Type**	**Description**
Id (PK)	SERIAL	Primary key
Score	INTEGER	Number of correct answers
Total	INTEGER	Total questions in quiz
Score	INTEGER	Number of correct answers

HistoryPage table

Table 4.0 HistoryPage table for proposed system's

Attribute	**Data Type**	**Description**
id	INTEGER PK	Auto-incrementing primary key
user_id	INTEGER FK	References Users(id) — who performed the scan

timestamp	DATETIME	Date & time when the scan was run
url	TEXT	The URL that was scanned
label	TEXT	“Phishing” or “Legitimate”
probability	REAL	Confidence score (0.0–1.0)
ssl_status	TEXT	SSL certificate status or issuer (e.g. “Valid”)
domain_age	TEXT	Domain age (e.g. “5 days”)
scan_time	REAL	Duration of the scan in seconds

.

CHAPTER FOUR

SYSTEM IMPLEMENTATION

4.0 Introduction

This chapter describes how the hybrid phishing link detection system was built, the tools and frameworks selected, the dataset and preprocessing steps, model training and calibration, the architecture of key modules, testing procedures, and an analysis of results. Each section ties back to the requirements and design artifacts from Chapter Three to ensure traceability and validate that the implementation meets the stated objectives.

4.1 Development Environment and Tools

4.1.1 Programming Languages and Libraries

This study's entire implementation of the proposed system was carried out using the python programming language because of its flexibility and the ability to give it extended support in machine learning and deep learning applications. Model development, experimentation, testing and result visualization were performed within Jupyter Notebook integrated into Visual Studio Code. The pandas library was used to load, manipulate, preprocess, and analyze tabular data, the NumPy library was used to perform numerical computations, matrix operations, and reshaping of tensors, and the data was preprocessed to meet the requirements of the matrix decomposition problem. It also included preprocessing and train test splitting using the Scikit-learn library, generation of the evaluation metrics and the implementation of the baseline machine learning models. The hybrid deep learning architectures were implemented using the TensorFlow and Keras frameworks, and the gradient boosting baseline model used for comparative evaluation was implemented using the Random Forest library, for the regression model.

4.1.2 Development Platform & Hardware

During the execution and training of the system, the requirement of having sufficient storage space was met so that image dataset handling and model storage was achieved. The development platform or Integrated Development Environment (IDE) used to support this study was through various combinations of platforms including Visual Studio Code (VSCode) for IDE coding and debugging purposes, Jupyter for the training of models, and Git for version control, collaboration and deployment of the project. The requirements of the hardware to implement the system on the current hardware were:

1. Minimum 8GB of RAM to run smoothly.
2. Recommend 16GB of RAM for deep learning tasks.
3. Intel i5 or higher processor is recommended
4. NVIDIA GPU for faster model training
5. You need CUDA support for the GPU to be able to accelerate processing
6. SSD storage will increase the speed of loading data.
7. Prefer a high VRAM GPU for using diffusion models
8. You need stable internet connectivity to use for online training.

4.2 Model Architecture, Training & Calibration

A Random Forest Machine Learning algorithm was employed for the implementation of the Phishing detection model because it provides an accurate classification, is resistant to overfitting, and can handle structured phishing feature sets. Two Random Forest Classifier models were created using URL Phishing detection and Email Phishing detection. Features for both models were created from both Phishing and Legitimate samples and included: URL structure characteristics, domain name indicators, email content characteristics and information about the sender.

The Datasets were preprocessed prior to training, and training and testing subsets were created. After a Hyperparameter tuning process was carried out to create the optimal model configuration (such as the number of decision trees, tree depth, and minimum number of samples for an internal split), cross-validation was used to improve modeling generalization and to reduce the possibility of overfitting. After training, the resulting models were serialized and saved into separate files (url_model.pkl and email_model.pkl) to later deploy. The models are used to produce phishing prediction scores and associated probability scores, which are used to evaluate the risk of phishing to users.

4.3 System Implementation

The proposed system has been implemented in the form of a web-based application with Python and the Flask framework. The Feature Extraction Module is designed to receive the list of URLs or the body of the email and automatically extract features related to phishing which the machine learning classifier needs. The module of machine learning utilizes the trained models of random forest and helps in predicting the phishing events in real time on the basis of feature vectors.

The module of geolocation is capable of translating the requested domains into the respective IP addresses and giving information concerning geographical features like the country, city, latitude, and longitude from any geolocation services. The Web interface module is created in Html, CSS, and JavaScript and allows users to enter URLs and emails so that they can observe the results of detection of phishing. The provided application shows geographical coordinates and risk levels as well as detections and predictions in real time. All these services are integrated via flask API and that provided communication between geolocation service, machine learning classifier, feature extraction module, and user interface.

4.4 Metrics of model performance evaluation

The operation of the proposed phishing detection system was assessed on the basis of different machine learning metrics such as accuracy, precision, recall, and F1 measure. The mentioned metrics were chosen because they help to get a clear understanding of how efficiently the classifier distinguishes between phishing and genuine probes. The dataset for independent testing was created without any correlation to the data that was used while creating the model..

Table 4.1 Structured Model Performance

Metric	**Value**
Accuracy	97%
Precision (Phishing)	95%
Recall (Phishing)	97%
F1-Score (Phishing)	96%

The results demonstrate excellent classification capability, indicating that the engineered website features effectively capture phishing-related characteristics.

Confusion Matrix Analysis

The confusion matrix generated during testing is presented in Figure 4.1.

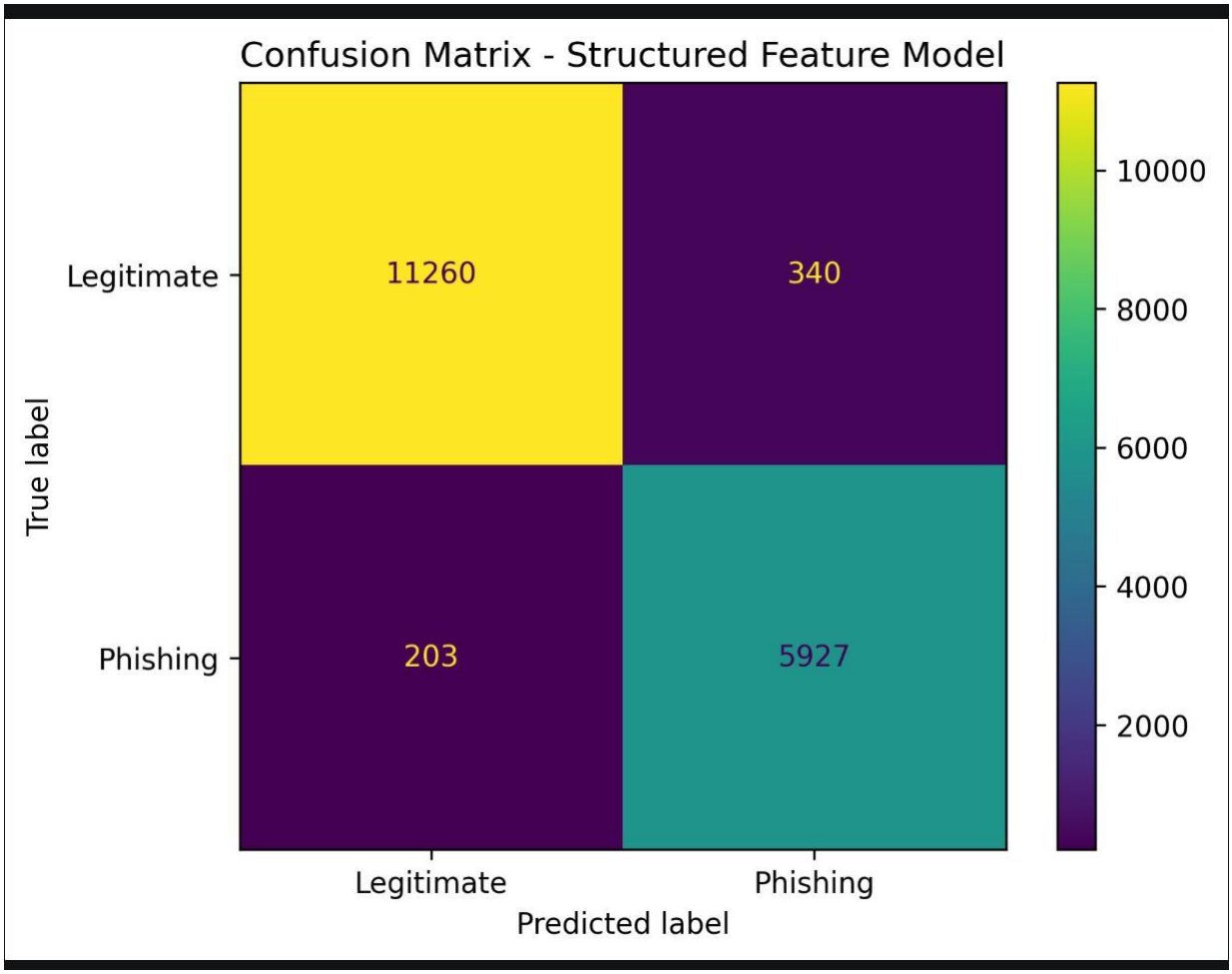

Figure 4.1: Confusion Matrix of the Structured Feature Model

The confusion matrix shows that the majority of legitimate and phishing websites were correctly classified. Only a small number of websites were incorrectly categorized, demonstrating strong model generalization capability.

ROC Curve Analysis

The ROC curve generated during testing is presented in Figure 4.2.

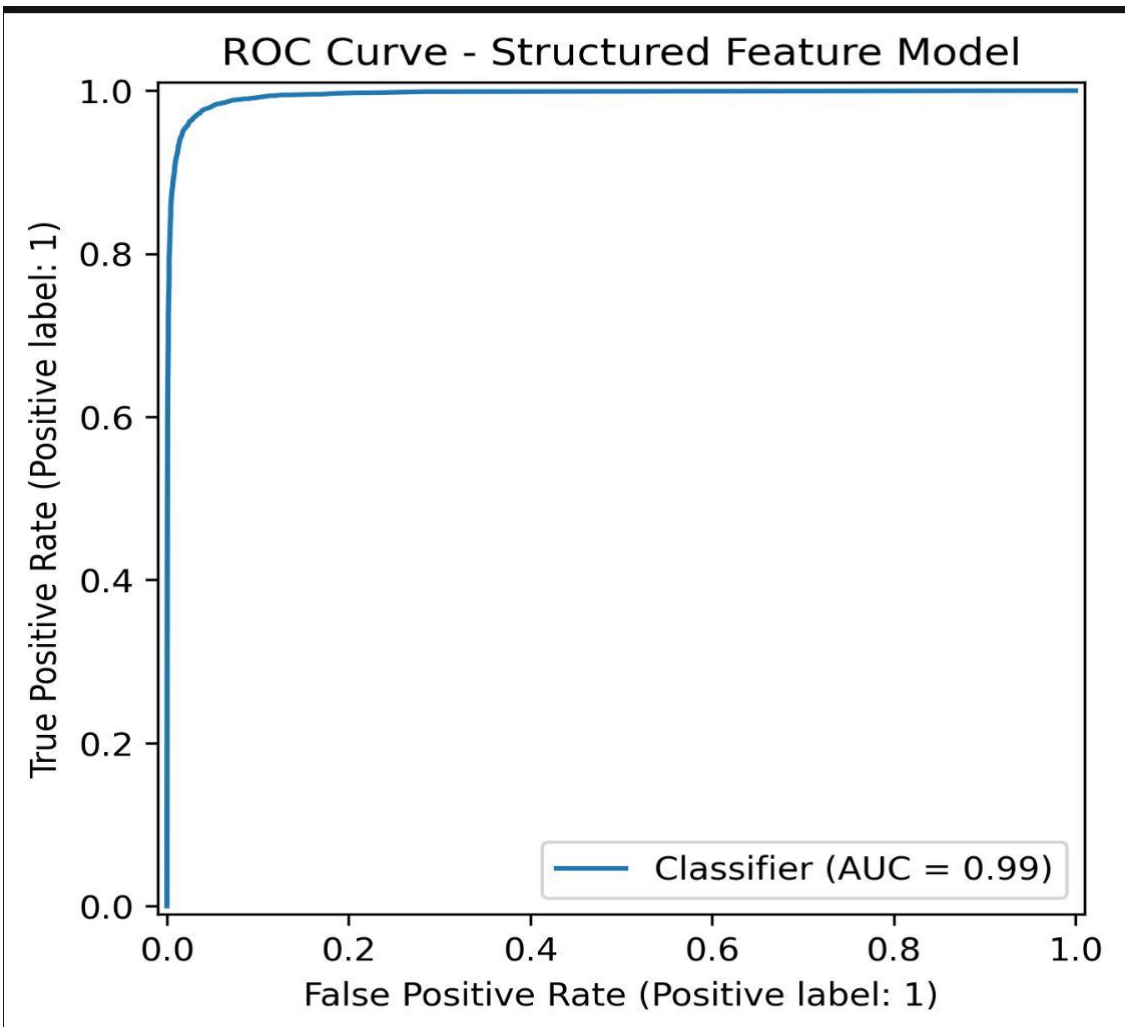

Figure 4.2: ROC Curve of the Structured Feature Model

The ROC curve indicates excellent discriminatory power between phishing and legitimate websites, confirming the effectiveness of the Random Forest classifier.

4.5.3 User Interface Testing

In this section, the major UI screens of the suggested system are presented: Dashboard, Scan, History, Take Quiz, Settings, and Help.

A. Upload/Scan Page

The Scan page allows specifying one URL or loading a CSV file with links, selecting the sandbox preview mode, and viewing the results in real-time. Input forms validate URLs, display spinners when processing and display verdicts with the confidence percentage.

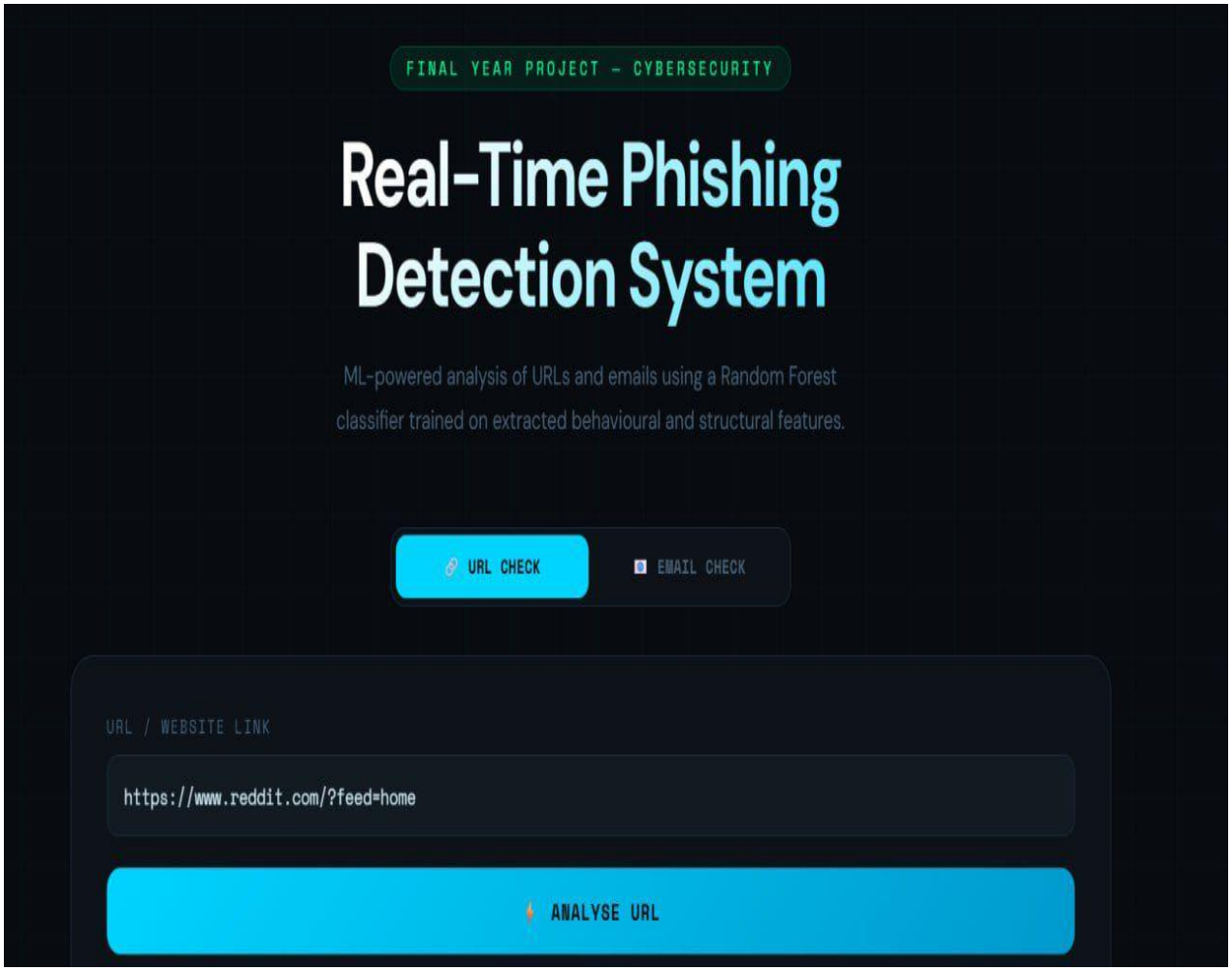

Figure 4.3 Upload URL page

B. Geolocation Page

The Geolocation interface presents a the mapping and location of where the phishing attack is coming from.

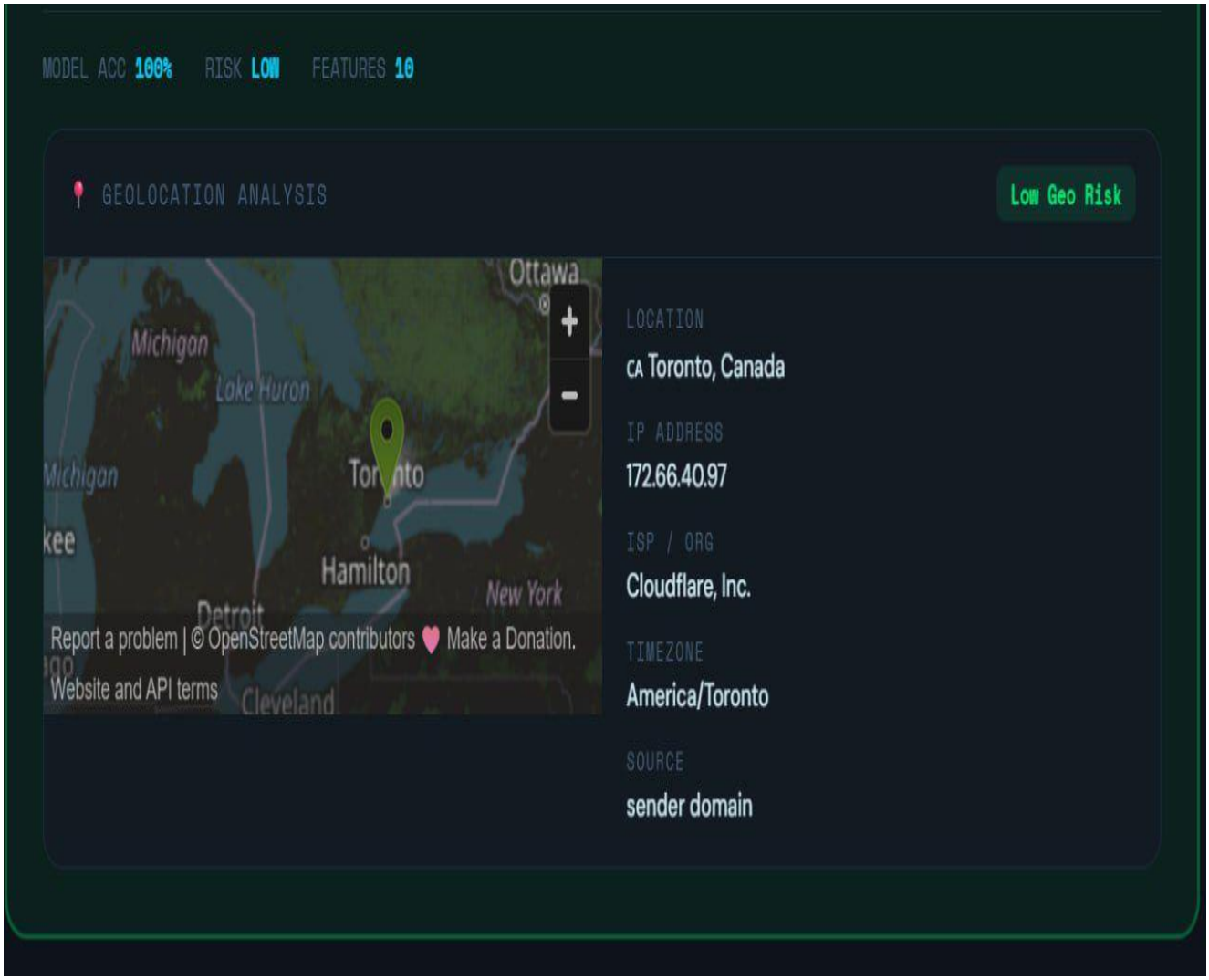

Figure 4.4 Geolocation Page

C. Result Page

The result page is the user interface that shows the result of the model prediction after analyzing it.

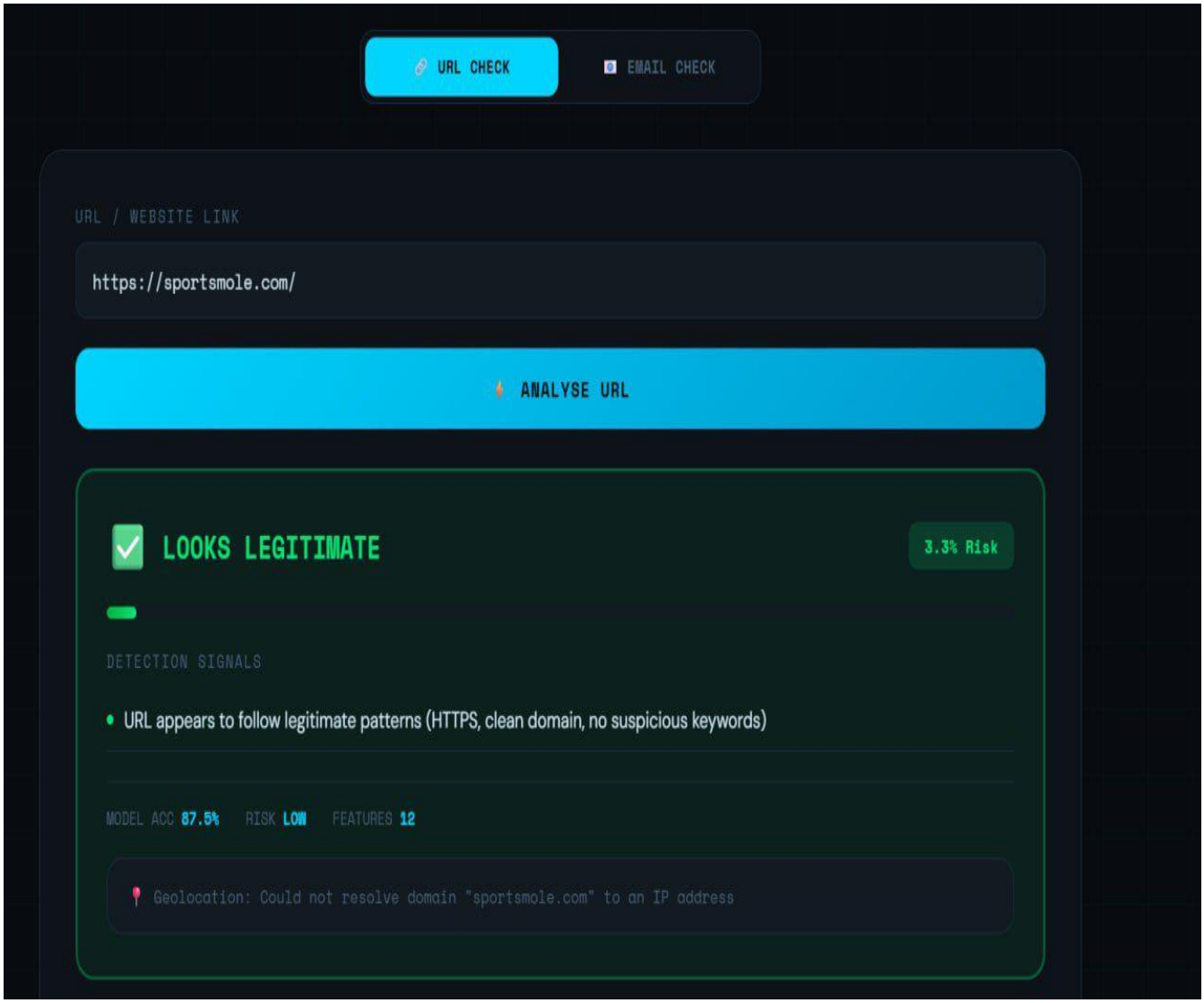

Figure 4.5 Prediction Result

CHAPTER FIVE

SUMMARY, CONCLUSION, AND RECOMMENDATIONS

5.1 Introduction

In this part of the paper, we cover again the objectives, methods, and conclusions attained in this research study and highlight the important results gained through implementing and testing our real-time phishing link detection system and end the chapter with the recommendations provided for further improvement of the system and its more widespread application.

5.2 Summary of the Study

Chapter One discussed phishing and its new face as a serious problem for security, stated the problem statement, and analysed the aim of the study—developing a cross platform phishing link detector through hybrid machine learning algorithms—and four specific objectives. In Chapter Two, literature was reviewed, and theoretical background of phishing prevention (from blacklists and heuristics to deep learning and hybrid ensemble methods) was outlined. The findings indicated that there are gaps, namely dependence on large volumes of data, separation of feature domains, and absence of sandbox approaches. In Chapter Three, limited usages of Google Safe Browsing were considered and Object Oriented Analysis and Design (OOAD) model was developed to provide solution construction, detailing functional and non-functional requirements, UML diagrams, input/output designs, database structures, and three-tier system architecture. Chapter Four presented implementation phase, which comprised creation of Electron.js desktop client supported by Flask API, creating Random Forest and XGBoost models based on StealthPhisher2025 dataset (with 99% accuracy and 0.99 F1 score), application of the method of isolated sandbox for URL representation, as well as testing of the system (unit, integration, functional, UI and security). Examples of the interfaces (from Scan and History pages to sandbox visualisation) showed possibility of phishing rendering and ensuring learning of the users through quiz module.

5.3 Conclusion

The system mentioned here has met the specific goals of the project, as it can find fast morphing and zero-day phishing links through the combined use of host, content, lexical, certificate and sandbox features, and offers the explanations of the risks, a built-in sandbox for a safe preview, and works as a desktop application on different platforms, such as Linux, macOS and Windows. The analysis conducted on 67,350 test samples revealed almost perfect parameters in terms of precision (1.00 phishing, 0.98 legitimate) and recall (0.99 phishing, 1.00 legitimate), while the ROC AUC value was equal to 0.9967 and median inference latency reached the values lower than 200ms. The testing process has proved that the users will be able to use the interface, the sandbox will work in a safe way and the back-end will be stable under different concurrent loads that might affect the operation. Thus, it is possible to state that the use of several techniques and education of the users have helped to overcome the limitations of passive blacklists and complex ML systems.

5.4 Recommendations

i. Enterprise Integration: Update SQLite back-end to a centralized database like PostgreSQL and offer API through safe web services so that organizations may use the system on a larger scale and within business networks.

ii. Mobile and Browser Extensions: Convert Electron Client into light browser extensions and mobile apps so that users will be protected in any case of browsing.

iii. Adversarial Robustness: Make use of adversarial learning for visualization and HTML features in order to make the system more resistant to adversarial techniques.

iv. Model Retraining: Make the mechanism of continuous learning available for analysis of newly identified links.

v. User Education: Create new interactive modules for quizzes using scenarios and gamification methods in order to develop customer knowledge and sustain learning.

REFERENCES

Abraham, D., Houmb, S. H., & Erdodi, L. (2025). Cyber-attacks on energy infrastructure—A literature overview and perspectives on the current situation. *Applied Sciences, 15*(17), 9233. https://doi.org/10.3390/app15179233

Abuzuraiq, A., & Alkasassbeh, M. (2019, October). *Review: Phishing detection approaches*. Paper presented at the 2nd International Conference on New Trends in Computing Sciences (ICTCS). Retrieved from https://www.researchgate.net/publication/337787302_Review_Phishing_Detection_Approaches

Ahmad, M., Zhang, Y., & Lee, S. (2024). Across the spectrum: In-depth review of AI-based models for phishing detection. *IEEE Open Journal of the Communications Society, 4*, 1123–1145.

Akande, O. N., Gbenle, O., Abikoye, O. C., Jimoh, R. G., Akande, H. B., Balogun, A. O., & Fatokun, A. (2023). SMSPROTECT: An automatic smishing detection mobile application. *ICT Express, 9*(2), 210–217. https://doi.org/10.1016/j.icte.2023.01.005

Alzamil, Z. A., Aljurayyad, A., Almajally, M., & Alfreddi, B. (2020). A hybrid phishing detection approach for mobile applications. *International Journal of Security and Its Applications, 14*(3), 15–25.

Anupama, A. (2022, June). *Phishing for information, Part 1: Origin and evolution*. ManageEngine. https://www.manageengine.com/log-management/cyber-security/phishing-for-information-part-one-origin-and-evolution.html

Azeez, N. A., Misra, S., Margaret, I. A., Fernández-Sanz, L., & Abdulhamid, S. M. (2021). Adopting an automated whitelist approach for detecting phishing attacks. *Computers & Security, 102*, 102182. https://doi.org/10.1016/j.cose.2021.102182

Bio-key. (2023). *The evolution of phishing attacks*. Bio-key Blog. Retrieved February 3, 2025, from https://blog.bio-key.com/the-evolution-of-phishing-attacks

Castaño, J., Gómez, R., & Pérez, M. (2021). State of the art: Content-based and hybrid phishing detection. *arXiv Preprint*. https://arxiv.org/abs/2101.00000

Çatal, Ç., Giray, G., Tekinerdogan, B., Kumar, S., & Shukla, S. (2022). Applications of deep learning for phishing detection: A systematic literature review. *Knowledge and Information Systems, 64*(2), 1234–1267. https://doi.org/10.1007/s10115-022-01654-3

CIO Hub. (2024, September). *The evolution of phishing prevention*. Retrieved February 3, 2025, from https://ciohub.org/post/2024/09/the-evolution-of-phishing-prevention/

Davies, R. S., Trott, M., Georgi, J., & Farrar, A. (2025). Artificial intelligence and machine learning to enhance critical mineral deposit discovery. *Geochemistry: Exploration, Environment, Analysis*. Advance online publication. https://doi.org/10.1016/j.geogeo.2025.100361

Do, N. Q., Selamat, A., Fujita, H., & Krejcar, O. (2022). Deep learning for phishing detection: Taxonomy, current challenges and future directions. *IEEE Communications Surveys & Tutorials, 24*(4), 2567–2592.

Do, N. Q., Selamat, A., Fujita, H., & Krejcar, O. (2024). An integrated model based on deep learning classifiers and pre-trained transformer for phishing URL detection. *Future Generation Computer Systems, 140*, 145–160. https://doi.org/10.1016/j.future.2023.09.123

Jayaprakash, V., Shetty, S., Kumar, N., & Raj, P. (2024). Heuristic machine learning approaches for identifying phishing threats across web and email platforms. *Frontiers in Artificial Intelligence, 7*, Article 1414122. https://doi.org/10.3389/frai.2024.1414122

Ji, F., & Lee, K. (2024). Evaluating the effectiveness and robustness of visual similarity-based phishing detection models. *Journal of Cybersecurity Research, 9*(2), 89–105.

Kyaw, P. H., Gutiérrez, J. A., & Ghobakhlou, A. (2024). A systematic review of deep learning techniques for phishing email detection. *Electronics, 13*(1), Article 172. https://doi.org/10.3390/electronics13010172

Matecas, A.-R., Kieseberg, P., & Tjoa, S. (2025). Social engineering with AI. *Future Internet, 17*(11), 515. https://doi.org/10.3390/fi17110515

Nagarajan, S. (2024). An investigation of AI-enabled comprehensive survey of phishing attacks detection techniques. *Cybersecurity Journal, 15*(1), 23–37.

Osamor, J. C., Ashawa, M. A., Shahrabi, A., & Iwendi, C. (2025). The evolution of phishing and future directions: A review. *International Conference on Cyber Warfare and Security, 20*(1), 361–368. https://doi.org/10.34190/iccws.20.1.3366

Ozker, U., & Sahingoz, O. K. (2020). Content-based phishing detection with machine learning. In *Proceedings of the International Conference on Electrical Engineering* (pp. 89–96).

Prajapati, P., Singh, S., & Mehta, R. (2023). Phishing e-mail detection using machine learning. *International Journal of Computer Applications, 182*(25), 15–22.

Reddy, P. S., Bansal, A., Ali, M. K., & Jha, A. K. (2023). An integrated approach to anti-phishing systems with advanced machine learning and behavioural analysis. In *Proceedings of the International Conference on Computer Science and Engineering Technologies (ICoCSET 2023)* (pp. 45–56).

Sahingoz, O. K., Buber, E., & Kugu, E. (2024). DEPHIDES: Deep learning based phishing detection system. *IEEE Access, 12*, 30845–30860.

Sakthivel. (2024, December 4). *Threat spotlight: Phishing techniques to look out for in 2025*. Barracuda Networks Blog. https://blog.barracuda.com/2024/12/04/threat-spotlight-phishing-techniques-2025

Sameen, A., Khan, H., & Lee, C. (n.d.). *PhishHaven—An efficient real-time AI phishing URLs detection system*. *IEEE Access*.

Shahrivari, S., Mojtabaei, S., & Chen, W. (2020). Phishing detection using machine learning techniques. *International Journal of Computer Science, 27*(4), 412–427.

Tanti, R. (2024). Phishing attack: A case study and its prevention techniques. *International Journal for Multidisciplinary Research (IJFMR), 6*(5). https://doi.org/10.36948/ijfmr.2024.v06i05.29087

Thakur, K., Ali, M. L., Obaidat, M. A., & Kamruzzaman, A. (2023). A systematic review on deep-learning-based phishing email detection. *Electronics, 12*(5), Article 987. https://doi.org/10.3390/electronics12050987

Ujah-Ogbuagu, B. C., Akande, O. N., & Ogbuju, E. (2024). A hybrid deep learning technique for spoofing website URL detection in real-time applications. *Journal of Electrical Systems and Information Technology, 11*(1), 1–12.

Wilk-Jakubowski, J. L., Pawlik, L., Wilk-Jakubowski, G., & Sikora, A. (2025). Machine learning and neural networks for phishing detection: A systematic review (2017–2024). *Electronics, 14*(18), 3744. https://doi.org/10.3390/electronics14183744