2023

A Web application to detect phishing links using machine learning approach.

Kimani, Benson Maina
School of Computing and Engineering Sciences
Strathmore University

A Web Application to Detect Phishing Links Using Machine Learning Approach

By

Benson Maina Kimani

147493

Master of Science in Information Technology

2023

A Web Application to Detect Phishing Links Using Machine Learning Approach

By

Benson Maina Kimani

147493

Submitted in Partial Fulfilment of the Requirements for the Degree of Master of Science in Information Technology at Strathmore University

School of Engineering and Computing Sciences (SCES)

Strathmore University

Nairobi, Kenya.

July, 2023

Declaration and Approval

Declaration

I declare that this work has not been previously submitted and approved for the award of a degree by this or any other University. To the best of my knowledge and belief, the thesis contains no material previously published or written by another person except where due reference is made in the thesis itself.

Student's Name: Benson Maina Kimani

Sign: _____ Date: 05/06/2023

Approval

The thesis of Benson Maina Kimani was reviewed and approved for examination by the following:

Dr Nelson Odunga Ochieng,

School of Computing & Engineering Sciences,

Strathmore University

Dr. Julius Butime,

Dean, School of Computing & Engineering Sciences,

Strathmore University

Dr. Bernard Shibwabo,

Director of Graduate Studies,

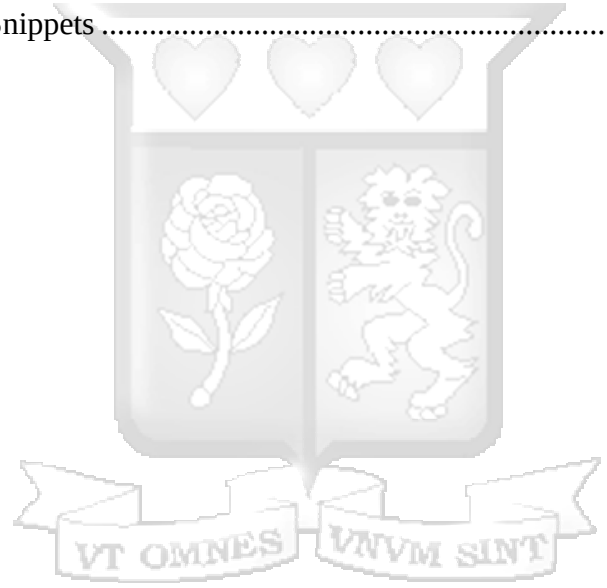
Strathmore University

Abstract

Covid-19 fast tracked the transition of ways of working and learning into remote working models and with it the rise of Cyber Crime. Phishing is one of the most effective cybercrimes where attackers deceive users into falling for baits that enable them to steal personal information including identification or social security number, password, credit card details, debit card details and usernames which they then use to commit crimes mostly associated with financial losses and or identity thefts. These attacks are frequently carried out through malicious emails, texts, and phone calls. Even in the presence of robust antivirus software and phishing detection tools, phishing has continued to be rampant and widespread as adoption of IT across the world goes up. Several methods have been implemented to deal with phishing attacks including, blacklisting phishing Uniform Resource Locators (URLs), heuristic rule based and machine learning models. The blacklist anti-phishing approach is limited in its ability to detect new phishing URLs due to its reliance on a pre-compiled list of phishing URLs, whereas most Machine Learning (ML) methods for detecting phishing websites have been reported with very decent detection accuracy but significant false alarm rates. Attackers are continuously re-inventing methods of attacks which as well makes heuristic rule-based methods vulnerable to failed detection. This study has provided further information on phishing attacks to create awareness, designed and implemented a web-based phishing links detection tool using ML deep learning model which achieved an accuracy of 94.27 complemented by heuristic rules and blacklist capabilities. Further improvement areas including continuous models training and features re-engineering for improved prediction accuracy, and equipping Internet of Things (IoT) gadgets with developed Application Programming Interface (API's) capability to determine what endpoints to or not to interact with.

Keywords: Remote working, Phishing, Identity thefts, Machine Learning, Detection tool.

Table of Contents



Definition of Terms

Artificial Intelligence	Machine modelling of human cognitive processes, notably by computer systems, includes learning, reasoning, and self-correction. Popular AI applications include expert systems, natural language processing, speech recognition, and machine vision (Burns, Nicole & Linda, 2022).
Business Email Compromise	Business email compromise (BEC) is a sort of phishing assault classified as a spear phishing attack, with the goal of duping employees into performing destructive acts and, in most cases, delivering money to the attacker ("Check Point," 2022).
Content Features	Textual contents of a website. (Aljofey et al., 2022)
Dependent Variables	A variable which changes according to changes in the independent variable (Bhandari, 2022).
Host-Based Features	Features derived from the host-name properties of a URL. (Ikwu, 2021)
Independent Variables	In an experimental study, an independent variable is the variable that is manipulated or altered to investigate the effects of its change. It is referred to be "independent" because it is unaffected by any other factors in the study. (Bhandari, 2022).
Integrated Development Environment	A program for developing applications that integrates common developer tools such as a source code editor,

	local build automation, and a debugger into a single graphical user interface (GUI) ("Redhat," 2019).
Lexical Features	Statistical characteristics taken from the literal URL string (Ikwu, 2021).
Machine Learning	A branch of artificial intelligence (AI) that focuses on developing programs that develop or enhance performance because of the information they consume (“Oracle,” 2023).
Malicious URL	Any link made for purpose of promoting scams, attacks, and frauds. When clicked on, malicious URLs can download ransomware, lead to phishing, or spear phishing emails, or cause other forms of cybercrime. (Samson, 2022)
Phisher	Individuals and or bots that perform phishing attacks (Shankar, Shetty & Badari, 2019).
Phishing	Phishing is a kind of cybercrime in which an attacker leverages email or other modes of contact to impersonate a trustworthy company or person in order to trick the user into disclosing personal information to the attacker (Gillis, 2020).
Phishing Link	Links created with the intent of inciting victims to click them to launch malware and or harvest their credentials. (Savickaitė, 2022)

Sprint	A brief window of time during which a scrum team must work to finish a specific quantity of work (Rehkopf, 2022).
Test Dataset	Dataset used to determine whether the machine learning model has fit or learned (Brownlee, 2020).
Train Dataset	Dataset used to fit or train the machine learning model (Brownlee, 2022).

List of Figures

List of Tables

Code Listings

List of Abbreviations

AI	-	Artificial Intelligence
AOL	-	American web portal and online service provider based in New York City
API	-	Application Programming Interface
AWS	-	Amazon Web Services
BEC	-	Business Email Compromise
CEO	-	Chief Executive Officer
CUDA	-	Compute Unified Device Architecture
DNN	-	Deep Neural Network
ESET	-	Slovak internet security company
FACC	-	Aerospace and defence company
HTML	-	Hypertext Transfer Markup Language
IDE	-	Integrated Development Environment
IJCSEA	-	International Journal of Computer Science, Engineering and Applications
IJRASET	-	International Journal for Research in Applied Science & Engineering Technology
IOT	-	Internet of Things
ISO	-	International Organisation for Standardisation
ML	-	Machine Learning
NIHR	-	National Institute for Health and Care Research
NLP	-	Natural Language Processing
SDLC	-	Systems Development Life Cycle
SRHD	-	Spokane Regional Health District
SVM	-	Support Vector Machines

URL	-	Universal Resource Locator
XP	-	Extreme Programming

Acknowledgements

I am very grateful to almighty God for the grace he has bestowed upon me during the time I have spent researching and composing this thesis.

I would like to thank Prof Ismail Ateya Lukandu for his knowledge and advice in Research Methodology, IT Thesis, Presentation, and Software Modelling, which helped me to undertake this work.

Further, I wish to thank my supervisor Dr Ochieng Nelson Odunga for offering his time to support me through this study, giving me constructive feedback and guiding me throughout the entire thesis.

Dedication

I dedicate this research project to Strathmore University fraternity, my friends, family, and classmates whose help has been essential in creating a climate that is conducive to learning, innovation, and development.

Chapter 1: Introduction

1.1 Background

Phishing first surfaced in the mid 1990s, but it took nearly a decade for the general public to become aware of these fraudulent practices. False emails and websites are used as bait in phishing attempts to encourage victims to freely divulge sensitive information. These schemes are commonly referred to as "phishing." For the same reason, the letter "ph" is used in place of the letter "f" in the term's spelling. Some of the first hackers were referred to as phreaks. The phrase "phreaking" refers to the investigation, testing, and research of telecommunications networks. Hackers and freaks have a long history together. Phishing scams were linked to these subterranean networks by the letter "ph." (“Phishing,” 2022).

Phishing has evolved over time, according to (Sjouwerman, 2022), Phishing assaults are as old as the internet, but the phrase was first used in 1996. Scammers sent an official-looking email to unwitting American web portal and online service provider (AOL) users, asking account information for a fictitious but plausible reason. A few people fell for the trap, and their accounts were utilized for evil purposes by hackers.

Fast forward to today, as latest as per (Rosenthal, 2022), Phishing is a severe problem that is becoming more prevalent by the year. Workers receive 14 phishing emails on average annually, according to Tessian research conducted in 2021. Certain industries were especially hard hit, with retail workers receiving an average of 49. Between the months of May and August of 2021, email-based attacks increased by 7.3%, with phishing tactics accounting for the bulk, according to research by the Slovak internet security business (ESET).

Security experts have recommended numerous anti-phishing techniques. However, and until today, there is no single solution capable of mitigating all the vulnerabilities (Vijayalakshmi et al., 2020). Some of the popular anti-phishing techniques include: List based where white and black lists of legitimate and illegitimate web URLs respectively are maintained in databases, heuristic oriented detection methods where extracted URL patterns are compared with known phishing URL patterns, visual similarity solutions where webpages are compared with screenshots of legitimate saved webpages, artificial intelligence (AI) and machine learning (ML) based solutions that are trained with known datasets and tested with different machine learning classifiers such as Naive Bayes, Random forest classifier, Support vector machine, Decision Tree, Logistic regression and so on (Kalaharshaa & Mehtre, 2021).

ML and AI based solutions tend to have higher accuracy among them Naive Bayes based solutions with an accuracy of 97.18. However, they are computationally expensive when data involved is huge. The issue with list-based solutions is their inability to have zero-hour detection capabilities which makes them quite ineffective especially when outdated lists are used, Visual Similarity solutions are expensive on memory and computation resources and only tend to be effective where phishing involves malicious webpages (Kalaharshaa & Mehtre, 2021). It follows that hybrid systems which are combinations of any of the techniques inherits the combined solution's limitations.

Review of some of the most notable phishing attacks reveal that some phishing attacks are designed to deploy malware to the target while others are designed to instruct the target to perform some activity without necessarily clicking any links. Some of the costliest successful phishing attacks on Google, Facebook and FACC could only have been identified through an anti-phishing technology which could detect Business Email Compromise (BEC) attacks via analysis of an email's body text ("Checkpoint", 2022).

1.2 Problem Statement

In large organizations, phishing attacks now cost about $15 million yearly, or more than $1,500 per employee (Proofpoint, 2021). As per Rosenthal, there are several approaches used to detect phishing baits, yet successful attacks are growing more widespread every year (2021). The different Machine Learning (ML) models proposed to detect phishing have been recorded to have attained accuracies as high as 97% (Kalaharshaa & Mehtre, 2021) and ranges of 89% for heuristic-based techniques (Vijayalakshmi et al., 2020) but are still not effectively accessible to by internet user, further, available anti-spam software used in scanning and classifying emails are pricey ("Capterra," 2023) to individual users and are mostly limited to emails while malicious URLs are increasingly being shared through social media platforms (Camu, 2022). A web-based phishing links detection tool equipped with hybrid ML and heuristic detection techniques for more accurate links identification capabilities can help users to validate legitimacy of these links.

1.3 General Objective

To develop a web application for detecting phishing links using machine learning approach.

1.4 Research Objectives

i. Investigate the challenges associated with phishing.

ii. Analyse existing methods of detecting phishing links.

iii. Develop a web application for detecting phishing links using ML approach.

iv. Test the application.

1.5 Research Questions

i. What are the challenges associated with phishing?

ii. What are current methods of detecting phishing links?

iii. How can the ML based phishing links detection web application be developed?

iv. How can the application be tested?

1.6 Justification

Some of the costly phishing attacks categorised as business email compromise (BEC) scam cost Crelan Bank approximately $78.5 million. 5 other major attacks including google and Facebook have been mentioned in the reports amounting to millions of dollars loss. Other events that justify this study include information compromise whose monetary value has not been allocated ("Checkpoint," 2022).

Digital technologies are used by more than 90% of small and medium-sized firms for communication, more than 80% for internal management and logistics, and more than 40% for sales states ("Advanced Business Portal," 2022). This information provides insights into the number of organisations faced with phishing threats which further points to the overwhelming need for this study.

1.7 Scope

This study is highly connected to email, text messages and phone calls as those are the usual methods that attackers use to bait their target, be it individuals and or organisations. More specifically and as per (Rosenthal, 2022), 96% of phishing assaults are sent by email, an additional 3% are committed through rogue websites, while only 1% use phones. This study focuses on email delivered attacks.

1.8 Limitations

1.8.1 Financial

Phishing is a global problem that affects the entire internet system. To perform comprehensive tests, there is a need to include implementation in different languages and different cultures whose financial capability is too vast for an individual graduate student. While the research results must be proper, they will only be done to a level whose available finance can handle.

1.8.2 Data Limitations

The way of testing how effective the tool developed will be limited to the knowledge of involved experts and not exactly the real hackers out there.

In addition, the environment used will be a simulation of scenarios rather than an exact work environment like employees in government offices or banks.

1.8.3 Time Limitations

While the study results should not be compromised by the time limitations that are there because of the many concurrent activities being carried out by the researcher, the study will be limited to what can be efficiently and effectively achieved within the available time.

Chapter 2: Literature Review

2.1 Introduction

There are different types of phishing attacks including but not limited to; deceptive, spear, whaling, and pharming. What these attacks have in common is the phishing process phases within which security professionals can leverage to implement automated, process and or training-based phishing bait detection techniques. In this chapter, the study examines the mentioned nature of phishing, how it has been tackled over time and what can be improved.

2.2 Theoretical Literature

2.2.1 Types of Phishing

According to (Shankar, Shetty & Badari, 2019), phishing attacks can be grouped into 4 types according to the way attack happens, they include:

i. Deceptive Phishing

Deceptive phishing is the most common type of phishing attack. It comprises sending an email to the victim that looks to be authentic while impersonating a credible website. A malicious URL or link that instructs the recipient to click on the URL may be included in the email that was received. The phishing website gathers all provided credentials and other sensitive information about the target and sends it to the attacker after the user clicks and completes the specified tasks.

ii. Spear Phishing

Spear phishing and deceptive phishing are essentially identical. The only distinction is the intended goal. Spear phishing, as opposed to deceptive phishing, targets a specific individual. The attacker selects a single target and persuades them to reveal crucial information. Scammers' emails are usually personalized for the receiver. The email would include some personal information about the target, such as name, employer, position held, and so on. The most popular venues for spear phishing are social media networks like LinkedIn, where they can easily learn about a person's occupation.

iii. Whaling

An executive with a high-ranking position, such as the CEO, is the target of this kind of phishing assault. The attacker would invest a significant amount of time characterizing the target before the attack. The attacker, like other types of phishers, sends the victim an email and convinces them into providing information to the attacker. Since those in leadership positions have access to the most sensitive information within the organization.

iv. Pharming

Pharming, in contrast to the other methods, does not need to target specific people. Without having to target specific persons specifically, the attack might cause widespread harm. Pharming can be done in two different methods:

a. The first method involves sending a piece of code to the target that modifies the local host files. The host files would convert the URLs into number strings, allowing the machine to access websites. Despite typing the correct URL, the target is sent to the fraudulent website.

b. The second method of pharming is to use a mechanism known as DNS Poisoning. This approach does not harm the machine's local host files, but it does harm the domain name system table. As a result, the target is unintentionally directed to dangerous

websites. The target believes they are viewing reputable websites, but because of DNS poisoning, they are actually visiting a malicious website.

2.2.2 The Phishing Process

According to (Alkhalil et al., 2021), While phone calls, instant chat, or physical letters can be used to initiate phishing assaults, most phishing attacks are initiated with an email, which can be sent in haphazard manner to any user or tailored to a specific group or individual.

According to their study, all phishing scams process include three primary phases as shown on Figure 2.1, sensitive valuables are requested from the target by the phisher, who then uses them for malevolent reasons after receiving the valuables from the target. The sub-processes of these phases can also be categorized in accordance with phishing trends.

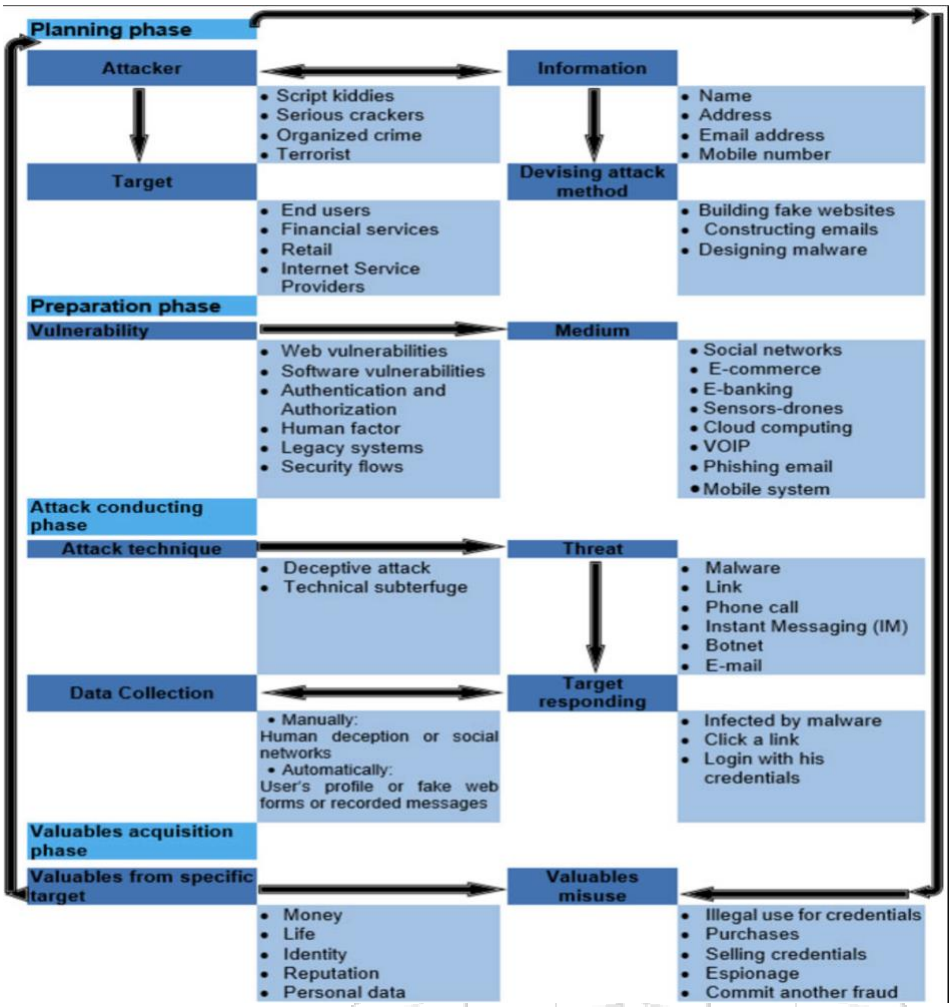


Figure 2.1: Phishing Process

(Alkhalil et al., 2021)

2.2.3 Notable Phishing Attacks

Some notable successful phishing attacks include:

i. Spokane Regional Health District's phishing attempt revealed the medical information of Washington residents.

A successful phishing attack on a local public health institution revealed the sensitive medical information of over 1,200 Washington residents. According to the Spokane Regional Health District (SRHD), 1,060 people's first and last names, initials, dates of birth, and other medical information may have been exposed. Among the health-related data exposed were diagnostic tests, drug and medication reasons, doctor references, client remarks, and pregnancy due dates (Bannister, 2022).

ii. FACC.

FACC, an Austrian manufacturer of aerospace components lost substantial funds to a BEC scam. In 2016, the organization disclosed that a phisher posing as the company's Chief Executive Officer (CEO) asked an accounting department employee to wire $61 million to an attacker’s bank account. (“Checkpoint,” 2022).

iii. Facebook and Google.

Facebook and Google were defrauded out of $100 million between 2013 and 2015. The phisher exploited the fact that both firms used Quanta, a Taiwanese service provider. The attacker sent a series of fictitious bills to a company that looked like Quanta, and both Facebook and Google paid them. (“Checkpoint,” 2022).

iv. African banking sector targeted by malware campaign.

According to Haworth, a cybercrime campaign targeting the African financial industry employed phishing emails and HTML smuggling tactics to spread malware. When the emails were opened, they contained HTML pages that directed the user to download an International Organization for Standardization (ISO) file containing a malicious Visual Basic script. HTML smuggling allows attackers to circumvent email gateway security by smuggling malicious files (2022).

2.2.4 Techniques to Detect Phishing Attacks

According to Tozzi, attackers build over a million phishing websites each month, and 95% of all attacks against enterprise networks begin with phishing. This then begs the question; how can businesses respond to this? There are two ways to respond; one is through employee training or phishing and cyber security awareness and another through automated systems. None of them is quite so effective without the other (2021).

Below include widely used automated anti-phishing protection techniques (Tozzi, 2021).

i. Filter Messages Based on Multiple Attributes.

Most security and IT professionals are aware that incoming email should be automatically filtered for dangerous material. When running scans, however, many teams and security technologies) commit the error of focusing on a single aspect of communications, generally the message's content. Although searching for phishing indicators such as misspelled words or questionable domain names is one technique to detect phishing emails, it is far from the only method. Evaluating each message based on several variables – its content, the domain from which it originated, whether it has an attachment, what type of attachment it is, etc. allows for a more informed determination of whether it is phishing. This multidimensional analysis is especially crucial for automatically detecting phishing efforts, considering that cybercriminals have become considerably more adept at creating convincing phishing

content. The days when screening email for terms like "Nigerian prince" was sufficient to identify phishers are long gone.

ii. Detonate Attachments in Sandboxes.

This entails "detonating" attachments or downloading and opening any URLs contained inside phishing content within a sandbox environment. By placing harmful content in a secure, isolated location and observing its behaviour, it is possible to identify anomalies or attack signatures that show the content is malicious. During the sandboxed explosion, the original content must remain quarantined and inaccessible to end users. The content can either be delivered to users securely or be permanently blocked, pending the outcomes of the sandbox.

iii. Block Sender Names and Domains Automatically.

If a phishing attempt is discovered, administrators can minimize its impact by blocking the sender's name and domain as early as possible using automation tools. This reduces the quantity of emails and other messages that phishers can send to users. Additionally, it hinders their capacity to interact with those whom they successfully dupe into responding. And, by blocking not only malicious sender names but also entire domains, you make it far more difficult for phishers to use several accounts to continue their attack.

iv. Automatically Scan Affected Endpoints.

Upon detecting a phishing email, one should also instantly and automatically scan any endpoints related with it, such as the PC or mobile device of the affected person. Immediate scanning will increase the likelihood of discovering and isolating any malware that the phishers were able to deploy.

v. Reset Affected User Credentials.

In addition to analysing affected endpoints, you should use automated tools to reset the login credentials of users who may have fallen victim to a phishing attempt. By logging them out of all open sessions and requiring a password reset, you mitigate an attacker's ability to exploit phished-compromised accounts.

2.2.5 Automation as the Future of Anti-Phishing.

Phishers will continue to improve at what they do. To remain competitive, firms must become more effective in their answers. This necessitates the adoption of automated anti-phishing systems that enable teams to not only detect phishing attempts quickly and correctly, but also mitigate the potential impact of a successful phishing breach on the IT infrastructure.

Automated web phishing detection as shown in Figure 2.2 include web address-based evaluation, webpage content similarity evaluation and hybrid approaches. These methods are implemented through machine learning algorithms (Vijayalakshmi et al., 2020).

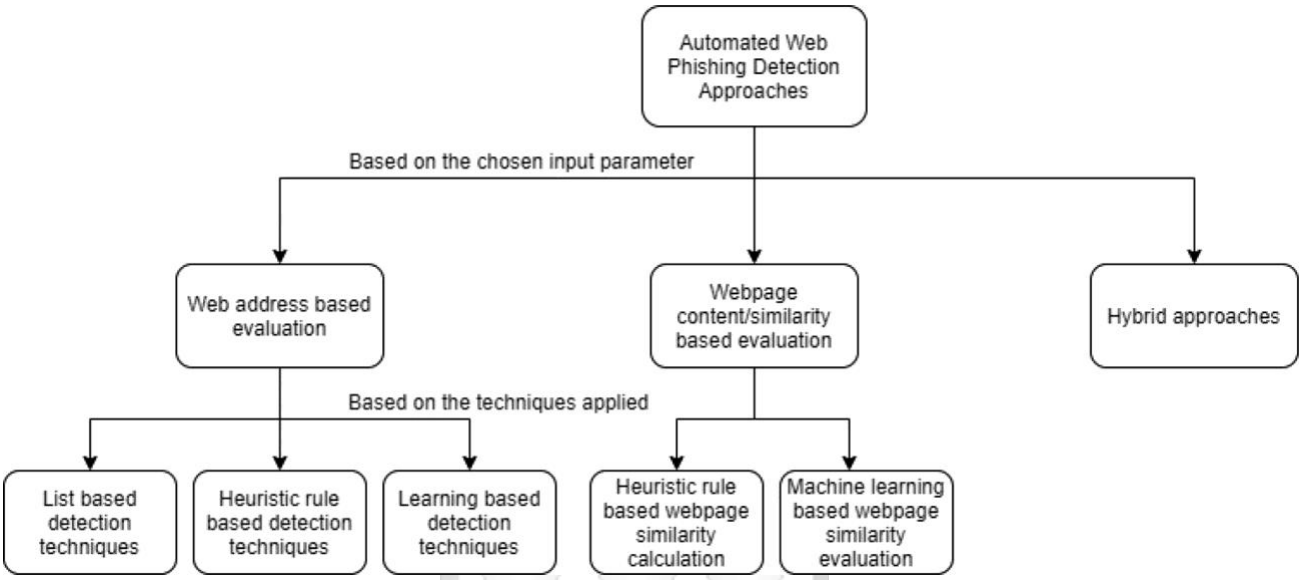


Figure 2.2: Automated Web Phishing Detection Approaches

(Vijayalakshmi et al., 2020)

Besides checking email source domains against maintained blacklists and whitelists, known phishing links lists, spelling and grammatical errors, Gmail uses TensorFlow framework to train ML based spam filters (Campbell, 2022). In addition, Gmail employs several AI-driven filters to determine which messages are marked as spam. These filters look at a variety of signals, such as IP address characteristics, domains/subdomains, bulk email sender authentication status, and user input. User feedback is also important in this filtering process, such as when a user flags an email as spam or sends a signal, because the filters learn from user actions. (Kumaran, 2022).

2.3 Empirical Literature

2.3.1 Performance Metrics for Phishing Detection Techniques.

To assess a phishing detection algorithm's performance, metrics such as zero-hour attack detection, language independence, Independence of third-party services, dataset used, and classification outcomes such as accuracy, true-positive rate (TPR), and false-negative rate (FNR) are employed (Vijayalakshmi et al., 2020).

2.3.2 Phishing Techniques Performance Evaluation

Table 2.1: Phishing Techniques Performance Evaluation

Anti-phishing approach	**Techniques used.**	**Third-party Independence**	**Dataset**	**Classification outcomes**		**Limitations**
				Accuracy	**TPR**	
Using URLs for deep learning. (Al-Ahmadi & Al-Alyan, 2020)	CNN based ML.	No	PhishTank, DomCop	97.78	95.9	Non independent phishing data.
Using URLs and webpage links. (Vijayalakshmi et al., 2020).	ML, Heuristic	Yes	2892 phishing URLs from PhishTank and 3305 genuine from Alexa's	93.98	98.3	Computationally expensive.
Auto updated whitelist. (Vijayalakshmi et al., 2020)	Lists, Heuristic, and search methods	No	PhishTank, Alexa and StuffGate	89.38	86.02	Low detection accuracy.

2.4 Models and Frameworks

Machine learning frameworks will be employed in this investigation. A machine learning framework is a platform that allows software engineers, data scientists, and machine learning engineers to build machine learning models without having to understand the mathematics and statistics that drive machine learning algorithms (“ProjectPro,” 2022).

Popular frameworks include:

2.4.1 PyTorch

PyTorch is framework for open-source machine learning that expedites the transition from research prototyping to production deployment ("PyTorch," 2022). It was built by Facebook on top of the Torch library ("ProjectPro," 2022) as a lightweight, open-source ML and DL framework. PyTorch leverages standard debuggers such as the interactive Python source code debugger and PyCharm. It was built with Python, C++, and CUDA. It is popular among newcomers in data science and machine learning due to its exceptional simplicity of handling complex tasks. It is widely used in domains such as computer vision, research, and NLP.

Pros

i. It's beginner friendly.
ii. It supports multiple GPUs.
iii. Integration with Python for data science operations is also seamless.

Cons

i. There are no data visualization and monitoring tools available.

2.4.2 Amazon Machine Learning

A cloud-based platform that enables developers of diverse skill levels to construct machine learning (ML) algorithms using an assortment of visualisation tools and wizards. It makes it easy for developers to connect to data stored in RDS or Amazon S3 and retrieve that data for developing predictive ML models. It is commonly used for image classification, binary classification, corporate predictions, and stock share forecasting, among other applications ("ProjectPro," 2022).

Pros

i. It enables machine learning projects to leverage cloud infrastructure.
ii. Superior performance.
iii. Cost-effective.

Cons.

i. Limited in integrating multiple programming language support.

2.4.3 TensorFlow

TensorFlow is an open-source, end-to-end platform for developing Machine Learning applications that allows developers to create machine learning applications by leveraging a number of tools, libraries, and community resources Johnson (2022).

TensorFlow's pros and cons ("ProjectPro," 2022).

Pros

i. Supported by Google and therefore effective.
ii. Offers seamless functionality and supports rapid updates.
iii. Permits the execution of graph subparts, enabling machines to extract discrete data.

Cons

i. Is slow compared to other frameworks.
ii. Is not supported in windows systems.

2.4.4 Spark MLLib

Spark MLlib is Apache Spark's ML component with multilingual support and the ability to scale computation massively. The issue, however, is that not all machine learning algorithms can be efficiently parallelized Dayananda (2019).

Pros and cons of Using Spark MLLib ("ProjectPro," 2022).

Pros

i. It is simple to utilize.
ii. Fast and multilingual.
iii. It has powerful analytical capabilities.

Cons

i. It does not accommodate any automatic optimization procedure.
ii. It is not appropriate for a multi-user scenario.

2.5 Architectures and Designs

2.5.1 Phishing URL Detection

This entails using heuristic methods to detect whether URLs incorporated in email messages are related to phishing. Different URL properties are extracted and submitted to a rule base, which produces a score and categorization indicating whether the URL is phishing, suspicious, or genuine (Jaynish, 2014).

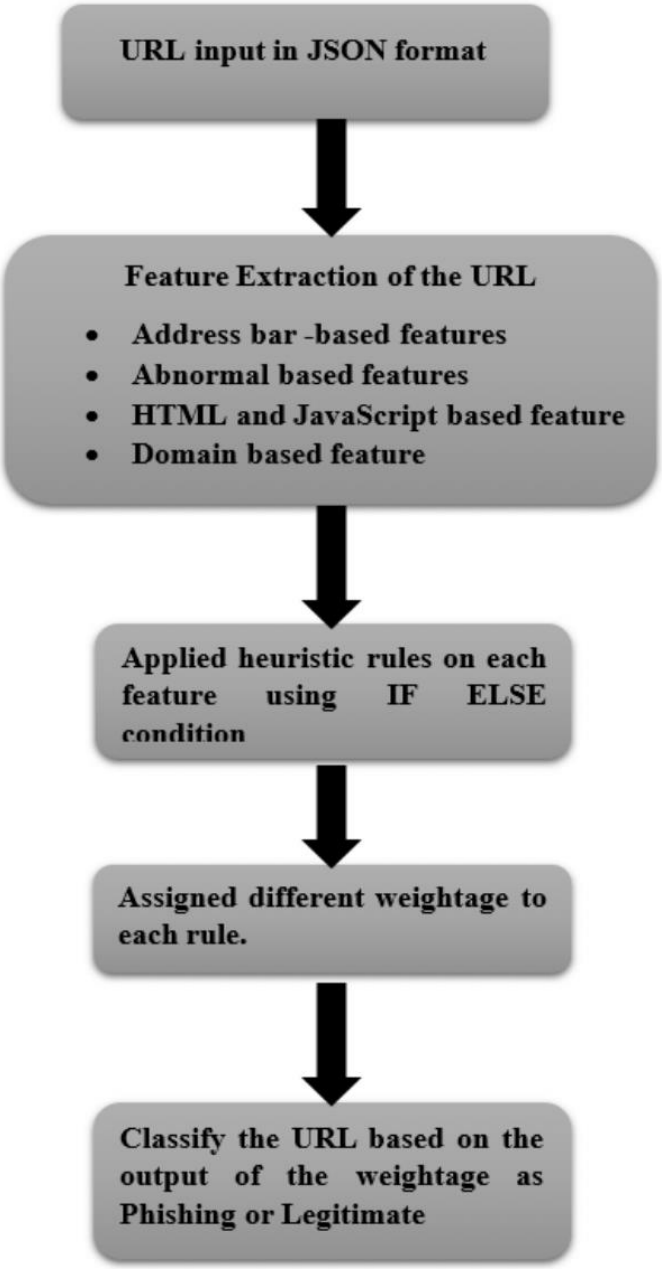


Figure 2.3: System Architecture of Heuristic Based Phishing Detection System

(Jaynish, 2014)

2.5.2 Predictive Model for Phishing Detection

Figure 2.4 demonstrates the architectural method, which includes the phases of feature extraction, machine learning classifier creation and training, evaluation, and phishing prediction. The feature extraction phase takes in mixed data and extracts features from the URL, web document properties, and web behaviour attributes. (Orunsolu, Sodiya & Akinwale, 2022).

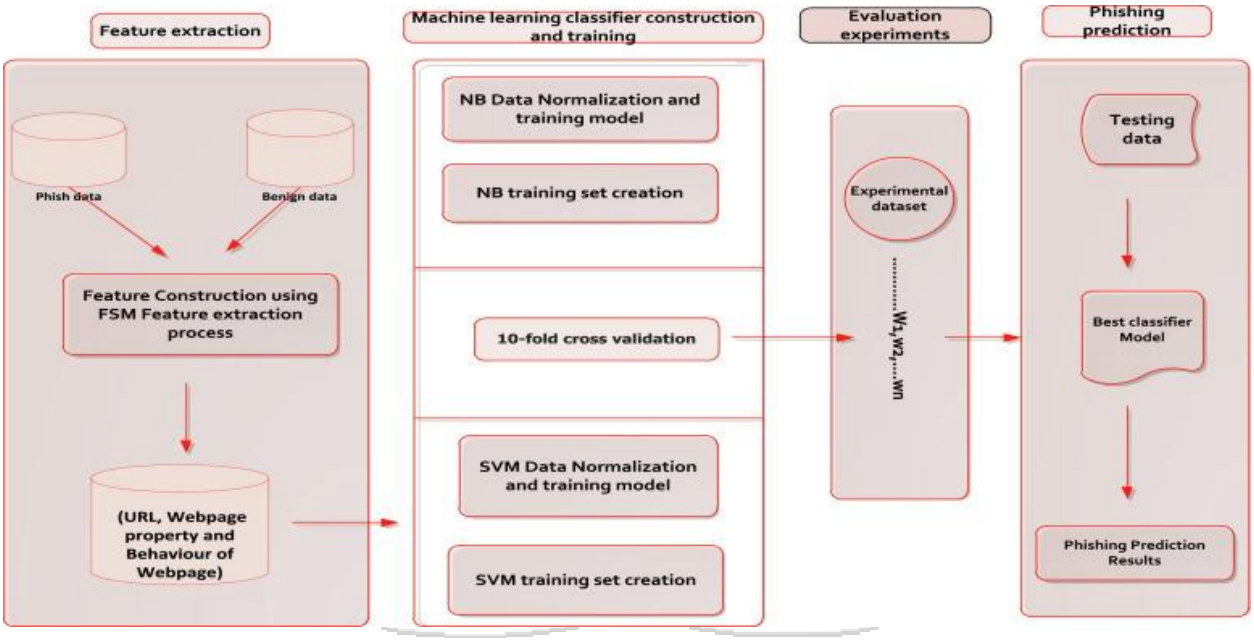


Figure 2.4: Predictive Model for Phishing Detection

(Orunsolu, Sodiya & Akinwale, 2022)

2.6 Algorithms

There many algorithms used in detecting phishing baits, this study focuses on email text extraction and analysis algorithms. The Naive Bayes family of algorithms, support vector machines (SVM), and deep learning are all prevalent text categorization algorithms. (“MonkeyLearn,” 2022).

2.6.1 Naïve Bayes Classifier

The Nave Bayes classifier is a probabilistic machine learning technique that has been used to detect phishing emails with an accuracy of 96.03% for balanced datasets and 97.21% for imbalanced datasets (Abdelaziz et al., 2020).

The Bayes' Theorem calculates the likelihood of an event occurring given the likelihood of a previous event. Equation 1 represents the mathematical version of Bayes' theorem, where A and B are events and P(B) is not equal to 0. This is to calculate the probability of event A given the truth of event B. Event B is also known as proof. P(A) takes precedence over A. (the prior probability, i.e., Probability of event before evidence is seen). The proof is an unnamed instance's attribute value (here, it is event B). P(A|B) denotes the a posteriori probability of B, which is the likelihood of an event occurring after evidence has been observed.

$$P(A|B) = \{P(B|A)\ P(A)\}/\{P(B)\}$$ (Kumar, N., 2022). Equation 1

2.6.2 Support Vector Machines (SVM)

SVM draws a line or "hyperplane" that divides a space into two subspaces, as shown in Figure 2.5. One subspace contains vectors (tags) from a specific group, while another subspace has vectors from

a different group. Afterwards, new examples are mapped onto one of the two groups of the hyperplane dependent on which side they fall on. The optimal hyperplane has the greatest separation between each tag (“MonkeyLearn,” 2022).

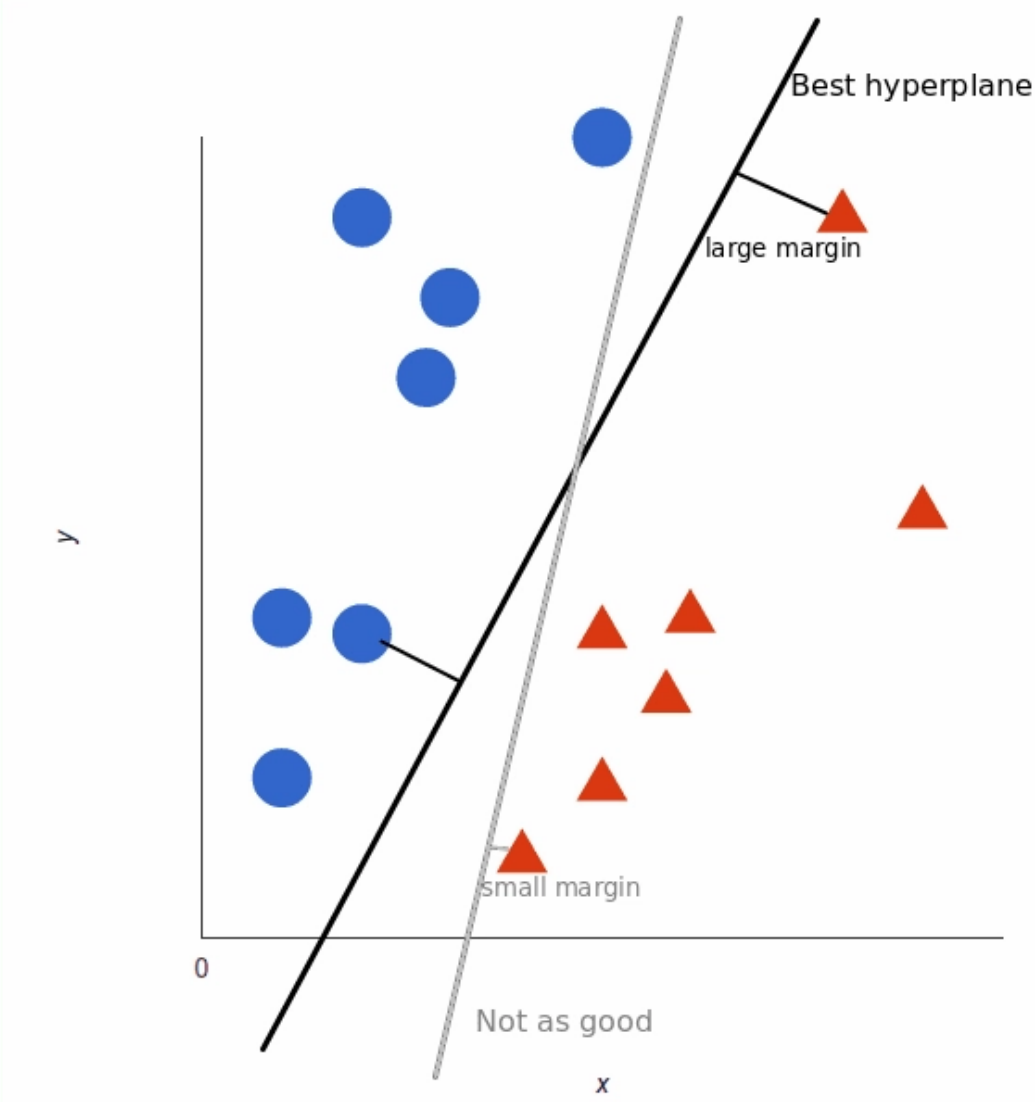


Figure 2.5: Two Dimensional SVM

(“MonkeyLearn,” 2022)

2.6.3 Deep Learning

Deep learning is a set of neural network-based approaches and methodologies inspired by the way the human brain processes. Because of their extraordinarily high accuracy with lower-level engineering and processing, deep learning architectures provide considerable benefits for text categorization. Convolutional Neural Networks (CNN) and Recurrent Neural Networks (RNN) are the two fundamental deep learning architectures for text classification ("MonkeyLearn," 2022). As training data increase, Deep learning does better than traditional ML algorithms as shown on Figure 2.6.

Figure 2.6: Deep Learning Vs Traditional Algorithms

(“MonkeyLearn,” 2022)

2.7 Systems and Applications

Table 2.1 details some popular phishing detection applications such as Snort, Spoof Guard and Catch Fish.

Table 2.2A: Phishing Detection Applications.

Application	**Type**	**Description**	**Capabilities**	**Limitations**
Snort (Al-Saaidah, 2017).	Network level	Heuristic tool good at detecting network level attacks.	Good level attacks identification.	Rules demand manual tweaks. Does not consider the content.
Spoof Guard (Al-Saaidah, 2017).	Client-Side Tool	Add-on to a web browser.	Warns the user if a link leads to a phishing website.	Warnings are usually disregarded by users. Not every email client is web-based.

Table 2.3B: Phishing Detection Applications.

Application	Type	Description	Capabilities	Limitations
Catch Phish (Al-Saaidah, 2017).	Client-side Tool	Identifies fake websites using rendered images.	Browser independent. Good outcomes on small data sets.	High processing time. Reliant on-screen resolution.
PhishShield (Rao & Ali, 2015).	Client-side Tool	Detects fake website based on URL.	Detecting zero-hour attacks.	Not able to detect using visual similarity.
Google Safe Browsing (Rao & Ali, 2015).	Client-side Tool	Detects fake website based on URL.	Is language independent.	Not able to detect zero-hour attacks.

2.8 Hypothesis

Analysis of notable phishing attacks depict that some phishing attacks are designed to deploy malware to the target while others are basically to instruct the target to perform some activity. Some of the above attacks on Google, Facebook and FACC discussed could only have been identified through an anti-phishing technology which could detect BEC attacks via analysis of an email's body text. This

means there is a need to design, test and implement a tool that can be exposed to users to help in detecting phishing baits that are embedded within text.

2.9 Proposed Solution

Major successful attacks involving google, Facebook and banks which were mostly instruction to make payments rather than visit malicious web links (“Checkpoint”, 2022) reveal that in addition to using methods that detect authenticity of URLs and web pages content a phishing detection tool that can analyse an email’s body and determine if it is authentic is key.

2.10 Conceptual Framework

Founded through the literature reviewed and gaps identified, this study proposed a web application solution based on heuristic rules and an ML deep learning algorithm as depicted on Figure 2.8 to classify URLs as either benign or phishing link. Legitimate URLs training data were obtained from Kaggle (Kumar, K., 2018) while phishing URLs were obtained from PhishTank (“PhishTank,” 2023).

The solution contains three major components, features extraction, saved trained deep learning model sated for inferencing and an API served through web interface. The application receives a link and email address as inputs from the user, the email address is used as the user identifier while the link is the object to be classified as either legitimate or phishing URL. The solution is further illustrated through the Process Flow as per Figure 2.7 which shows the five main processes involved: Data collection source, URL labelling, features extraction, model training and testing and inferencing using the saved model state.

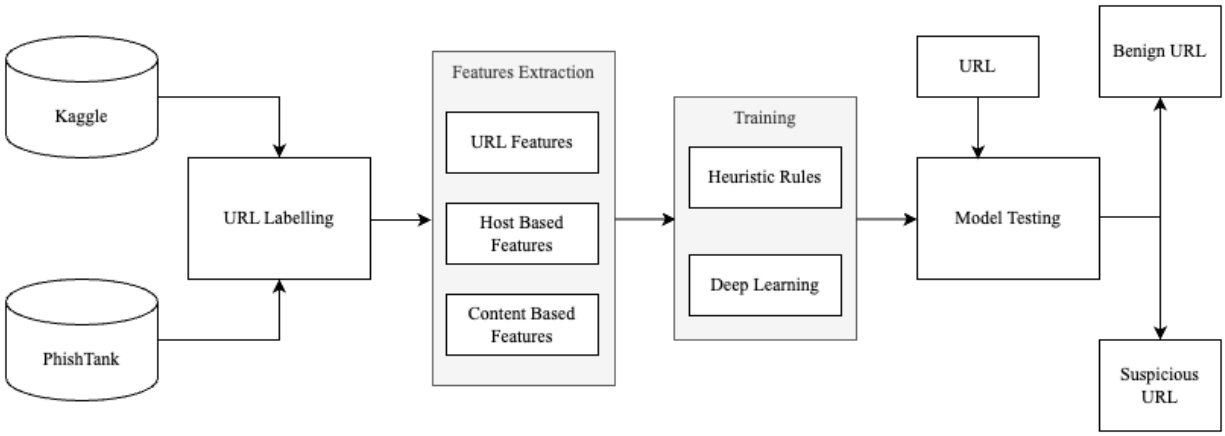


Figure 2.7: Process Flow

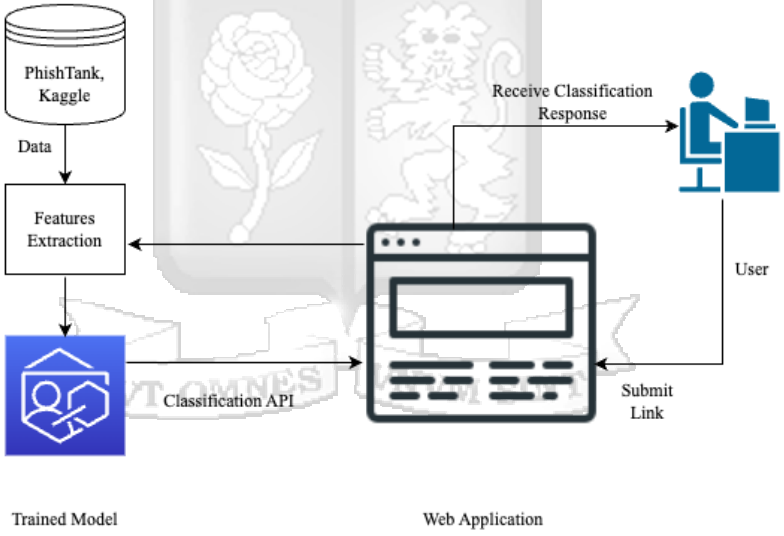


Figure 2.8: Conceptual Framework

Chapter 3: Research Methodology

3.1 Introduction

Research is one of the ways to find answers to questions (Kumar, R., 2011, p.p.26-36). The process applied to conduct the research should:

i. Be conducted in accordance with a set of philosophies.
ii. Employ procedures, methods, and strategies that have been proven to be valid and reliable.
iii. Intended to be impartial and neutral.

The existing techniques and applications of detecting phishing links were analysed through literature review and their inefficiencies described on Table 2.1 and Table 2.2 respectively. A high-level conceptual framework of the web-based phishing links detection tool was designed as depicted on Figure 2.1 which forms the basis for the design, implementation, and testing of the tool within the design and development methodologies outlined in this chapter.

3.2 Research Design and Philosophy

Research philosophies refer to methodologies such as qualitative and quantitative research. Quantitative research is concerned with numbers and statistics, whereas qualitative research is concerned with words and meanings (Streefkerk, 2019). Validity shows that the proper procedures were utilized to discover answers to a query. Reliability refers to the quality of a measurement procedure that provides repeatability and accuracy. Unbiased and objective indicates that each step was completed in an unbiased way and conclusions were derived to the best of the researcher's ability

and without the introduction of vested interest. Bias is an intentional attempt to conceal or highlight something (Gounda, 2012).

There are two types of research: applied and pure. Specific difficulties are addressed in applied research by employing well-established theories and concepts. Applied research encompasses most of the experimental research, case studies, and inter-disciplinary research. Social problem study, for example, has an immediate application. Applied research is concerned with research that has immediate potential applications, such as research on boosting machine efficiency, increasing the gain factor of material production, pollution management, disease vaccination preparation, and so on (Gounda, 2012). Pure research is academic in nature, and it is carried out to gain knowledge about phenomena that may or may not have immediate applications, as well as to build new approaches and procedures that comprise the corpus of research methodology (Kumar, R., 2011, p.p.26-36).

This study focused on a practical problem namely phishing and led to the design and implementation of application for detecting phish links, it is therefore applied research.

Methodologies for pure and applied research might be quantitative, qualitative, or both. Quantitative research is concerned with the quantification of quantities or amounts. A process expressed or described in terms of one or more quantities. Qualitative research is concerned with qualitative phenomena related to quality. Non-numerical, descriptive, logical, and linguistic. Its goal is to convey meaning and emotion while also depicting the situation (Gounda, 2012). In this study, the research is qualitative in a way that links in emails need to be analysed and categorized as either phishing or legitimate links.

3.3 Data Collection

Accurate data collection is necessary for making informed business choices, ensuring quality assurance, and maintaining research integrity (“Simplilearn,” 2022). Surveys, transactional tracking,

interviews and focus groups, observation, online tracking, and various forms of social and media monitoring are all popular data collection approaches.

In this research, independent variables data was retrieved from ("PhishTank," 2023) through their developer Application Programming Interface (API) service for Phishing URLs and Kaggle (Kumar, K., 2018) for legitimate URLs while dependent variables were determined through data analysis.

3.3.1 Data Sampling

There are different methods of sampling in research which fall within two major types: Probability and Non-Probability sampling methods McCombes (2022):

i. In quantitative research, probability sampling approaches where every member of the population has a chance of being selected are typically employed. Methods such as simple random sampling, systematic sampling, stratified sampling, and cluster sampling fall under this category.
ii. Methods of non-probability sampling when people are chosen based on non-random criteria and not everyone has a chance to be involved. This category contains techniques such as convenience sampling, purposeful sampling, snowball sampling, and quota sampling.

The data set used was from within the last six years to achieve relevancy, the set included 43,103 phishing links and 345,738 benign links, due to limitations in response time and number of requests that could be made to whois and requests packages to extract features, 1,027 phish and 1,100 benign links were selected randomly for features extraction and a balanced random set of 1,000 each used for training and test data. The extracted data set was split into two, test dataset at 80% and 20% respectively. Train: 80% and test: 20%, train: 67% and test: 33%, and train: 50% and test: 50% are all prevalent split percentages (Brownlee, 2020).

3.4 Data Analysis

There are numerous data analysis techniques, but they all fit into one of two categories: qualitative analysis and quantitative analysis (Kelley, 2022).

This study dealt with qualitative data and therefore used qualitative data analysis methods which include:

i. Content analysis method for evaluating patterns within a piece of content, for example, words and phrases i.e., has HTTPs and has HTTP in this study, and grouping them into categories identified by specified codes (Warren, 2020)
ii. Thematic analysis where data is grouped into categories based on determinant patterns within them (Warren, 2020). In this study, URL Length was grouped into short medium and long.

3.5 Research Quality and Reliability

Research quality is determined by its validity and reliability. Reliability is characterized by consistent similar test results over time while validity is depicted by accurate results presentation, sound cause and effect relationship, good research design, and applicability and transferability of research results (Price, 2019).

In this study, the results extracted from the solution developed were reproducible and consistent in line with the research quality philosophy.

3.6 System Development Methodology

To design and develop a phishing link detection tool, several tools and processes were used including but not limited to programming languages, system development methodology, data acquisition, data verification, testing, and reporting. For the system development methodologies, there are 2 major popular methods namely agile, and waterfall as analysed in the analysis of methods.

3.6.1 Analysis of Methods

i. Waterfall

The waterfall process for software development necessitates the creation of project specifications in advance. It is divided into distinct phases or steps. Before proceeding, it is critical that both developers and customers have a thorough understanding of the project's needs and scope at the early phase. Determine the project's requirements and scope before designing, implementing, testing, deploying, and ultimately maintaining it. This is a short order because this strategy lacks flexibility; the customer and developer's initial choices must be followed. If changes or errors are required during the final stages, the Waterfall method frequently necessitates a full restart. Typically, one phase must be finished before the next can begin, which can help with task organization and completion. Furthermore, because the full project scope is known ahead of time, software progress can be simply measured (Appelbaum, 2019).

ii. Agile

Agile software development, also known as Agile, is a development technique that anticipates the need for flexibility and applies pragmatism to the end product's delivery (Brush, 2019). Figure 3.1 is the graphical representation of how a progresses from planning Iterations involving team collaboration and the management of delivered product and the feedback from stakeholders. The feedback becomes

part of the next iteration for actual implementation. (“Synopsys Editorial Team,” 2017). In this study, agile software development methodology will be used as SLDL as it emphasises flexibility and fast delivery as opposed to Waterfall methodology.

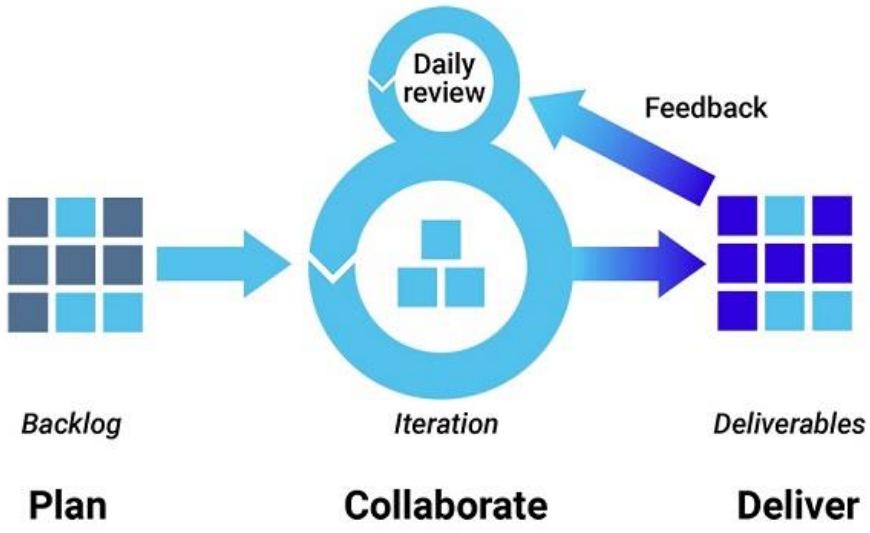


Figure 3.1 Agile development methodology

(“Synopsys Editorial Team,” 2017)

There are different types of agile methodologies (Sarangam, 2022). The most popular ones include:

a. Kanban

Kanban is a Japanese term that translates to "visual board or signboard" and is related with the phrase "just in time!" The Kanban Board, which is separated into columns to depict the software development process flow, represents Kanban project management.

b. Scrum

Scrum separates software development phases into stages or cycles known as "sprints." The development time for each sprint is minimized and committed, allowing for the management of only one sprint at a time. Scrum, like other agile techniques, emphasizes ongoing deliverables, allowing designers to change their priorities such that any incomplete or delayed sprints receive extra attention. The scrum team is made up of project leaders such as a scrum master and a product owner who maintain open lines of communication on daily scrum operations.

c. Crystal

Crystal is a collection of small agile techniques distinguished by characteristics like as system criticality, team size, and project priority. Depending on the project's criticality, any of the crystal series can be used: Crystal crimson, crystal orange, and crystal clear are just a few examples.

d. Extreme Programming (XP)

Among the agile software development approaches, XP was utilized since it promotes frequent product releases in short development cycles, which automatically increases the system's productivity and creates a checkpoint where any needs can be reviewed quickly Hamilton (2022). XP involves 6 Phases including: Planning, Analysis, design, Execution, Wrapping and Closing (Hamilton, 2022).

3.6.2 System Development

The development environment is a workspace including a collection of programming procedures and tools used to create the source code for an application or software product (Zola, 2022). This study involved development of a web application which required a development environment that was optimised for the web. Key factors considered were Integrated Development Environment (IDE), programming languages to be used, hosting and web server application.

i. Programming language

Java and Python are among the most popular programming languages for developing AI and machine learning applications, in this study python was used because it currently has the largest online support communities and has proven helpful in data science, artificial intelligence, and machine learning projects (Aguilar, 2022).

ii. IDE

An IDE that stands out for programming in python is PyCharm which executes edits and debugs Python code without any external requirements (Simpao, 2022).

iii. Framework

Major python frameworks for building web applications are Django and Flask, both have a whole lot of community support and documentation. Flask was used because of its better ability to accommodate dynamic requirements changes compared to Django (Singh, 2022).

iv. Hosting

The application was developed in containerized services using Docker, these containerized services are cloud ready and can be hosted in a cloud service provider like Amazon Web Services (AWS) which provides easy-to-use virtual private servers. ("Amazon Web Services," 2021)

v. Solution Development

The solution was developed using Agile XP methodology.

a. Planning.

In the planning stage, system requirements were gathered and reviewed, and the development environment configured.

b. Analysis.

In the analysis phase, the requirements were broken down into scenarios and or implementable use cases.

c. Design.

This step involves system analysis and design, breaking down the use cases identified in analysis into components that were be implemented in a sequential manner.

d. Execution.

The execution involved system implementation and or coding according to use case diagrams, data flow diagrams and interaction diagrams created in the design phase.

e. Wrapping.

This step involved small releases, regression testing, demos and reviews and development of new stories based on relevant system improvements.

f. Closure.

This stage included compiling results on both the performance and accuracy of the tools against test dataset.

3.7 System Implementation and Testing

This section describes the steps involves in implementing the proposed solution.

i. System development

The machine learning component was implemented using PyTorch in tandem with python-based packages for data manipulation and analysis including NumPy and Pandas, these packages were deployed withing a python virtual environment which were later composed within a docker container for quicker and seamless deployment. The API for inferencing was developed on top of flask python-based web development framework and supported using MySQL database for requests and users'

management. The API was then exposed to a web interface using HTML and Cascading Style Sheets (CSS) web scripting languages.

a. Features Extraction

In features extraction, the features listed in the Appendix E are extracted for each url and appended in a csv file for both the phishing and benign links.

b. Features engineering.

In this step, the features are analysed to determine which ones to include and or exclude in the machine learning and validation. The methods used to determine features relevance include features correlation maps, violin plotting and best fitting method.

In addition, some features were considered as they have been found to be revelated in phishing links classification by other researchers.

c. Model Training and Validation

To facilitate the implementation of the DNN machine learning algorithm, the data samples were labelled as phishing and benign targets. The training sets and testing sets were separated for training and evaluating the model, respectively. Eighty percent of the data used for training, while the remaining twenty percent was used to evaluate the model.

d. Saving the Model State

The trained model state in saved in preparation for deployment to enable real time URLs classification capability. This is achieved using PyTorch's built in model saving function. The saved model state is then deployed and exposed to the web interface using an inferencing API.

ii. System Testing

System testing included the validation of inferencing API which receives user input and responds with classification instance using randomly selected benign and phish links.

3.8 Utilization and Dissemination of Research Results

Effective dissemination of research results involves getting the findings of the research to the organizations and or people who can make use of them ("NIHR," 2019).

In addition to submission of the research findings to Strathmore Graduate School, this study will include identification of relevant conferences and publishing service providers to sign up to and facilitate the efforts towards its utilization and application.

3.9 Ethical Considerations/ Issues

This study involved use of data and information from other sources which was presented exactly the way it was collected without violating privacy and ownership. Other researchers' work was cited as so.

Chapter 4: System Analysis and Design

4.1 Introduction

System analysis and design help in critical assessment of proposed system components, interfaces, interactions, and architectures of a system aimed at meeting the end-user's requirements (Prateek, 2021). It enables the development team to have an ariel view of the system structure as well as provision of a diagrammatic representation of the same to the stakeholders by use of world-wide accepted UML notation. The system requirements for the ML based phishing links detection web application were broken down into general system architecture, implementable use cases, data flow diagrams, class diagrams and the underlying database schema.

4.2 Requirements Analysis

This research's objective was to develop a web application that will enable users validate the legitimacy of a link based on the URL features within it. The requirements defined in this chapter were implemented to achieve the laid-out objectives.

4.2.1 Functional Requirements

i. The application should accept a link inform of a text input from user via client or browser.
ii. The application should accept an email address from user via client or browser.
iii. The application should extract URLs features from the link submitted by the user.
iv. The application should classify the link as either benign and or a phish based on the saved trained DNN ML model state.
v. The application should return this information to the user's client or browser.

4.2.2 Non-functional Requirements

i. Usability

The web application should be accessible accessed through the web browser where the user will be required to either search the domain name and or just enter the application's URL which will then provide a webpage where they can enter or copy paste their text from social media inbox and or email, then submit through a submit button which will then send data to the server and provide response within a few seconds.

ii. Reliability

The ML algorithms should be consistent in classifying the links according to the achieved percentage of accuracy during training and testing. Provided with the same inputs, the application should consistently provide accurate results.

iii. Availability

The web application components should be hosted within containerized services.

iv. Scalability

Considering how widespread phishing is, there will be a lot of requests being submitted which in turn demands the application to be performant and scalable. Scalability here being the capability of system to handle high and increasing traffic efficiently and effectively up to a certain level. e.g., the application will be tested and optimized for ability to handle 20 requests per second.

v. Security

To secure the application's requests and feedback from interception and modification, the connection between the client and the server will be equipped with a Secure Socket Layer (SSL) certificate to ensure encryption of data in transit.

4.3 System Design and Architecture

System architecture refers to abstract, conceptualization-oriented, global, and focused to achieve the mission and life cycle concepts of the system. Its purpose is to define a comprehensive solution based on principles, concepts, and properties logically related to and consistent with each other (Faisandier, Garry & Rick, 2022). The Figure 4.1 illustrates the general system architecture of the phish link detection web application, the researcher used data from ("Phishing", 2022) and (Kumar, K., 2018) for training and testing the ML model based on the DNN deep learning algorithm in tandem with heuristic methods and lists. The trained model was then integrated with the web application through an API where users will be submitting links for classification as either benign or suspicious.

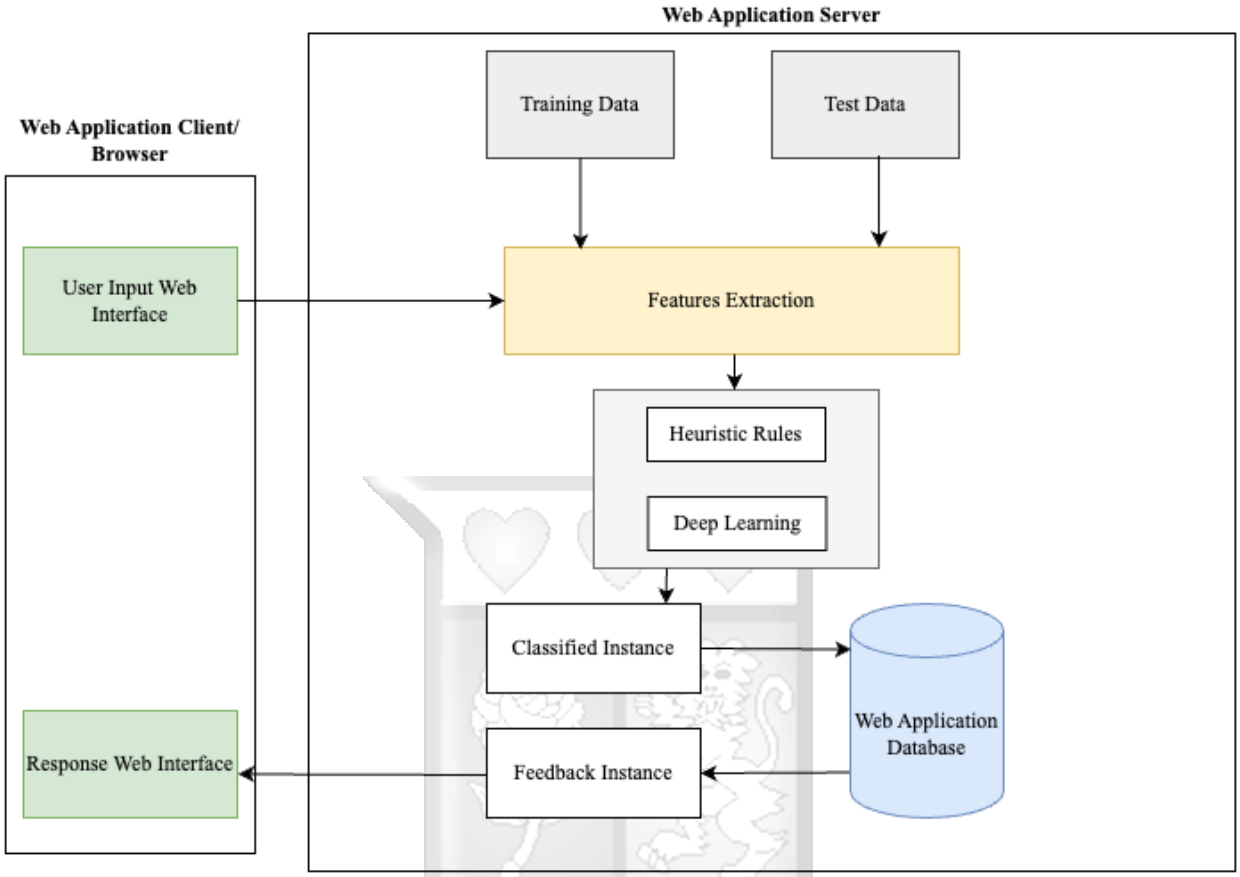


Figure 4.1: System Architecture

4.4 Use Case Diagram

The UML use case diagram represents possible interactions between an actor and or actors and the system. The phishing link detection application will include 4 main actors as show on Figure 4.2:

i. The system engineer who will be developing, training, and maintaining the model.
ii. The system administrators who will be maintaining users.

iii. The client and or user who will be using the system to check legitimacy of a suspected email.

iv. The URL feature extractor service that will be processing the link and extracting relevant elements to be used in classifying the link.

Table 4.1A: Classify Link Use Case

Use Case: Classify Link
Primary Actor: User
Preconditions: Client has submitted a link classification request. Client is allocated an identifier.
Post Conditions: Classified item line is saved, and response of classification sent back to the client. The classified line item includes attributes as defined; id, link, classification, and date_submitted.

Table 4.2B: Classify Link Use Case

Main Success Scenario	
Actor Responsibility	**System Responsibility**
1. User accesses the application via the browser.	
2. User submits a link.	
	3. System receives the link.
	4. System classifies the link.
	5. System responds with the classification of the link as either benign or suspicious.
6. User receives the classification response.	
7. User exits the browser.	

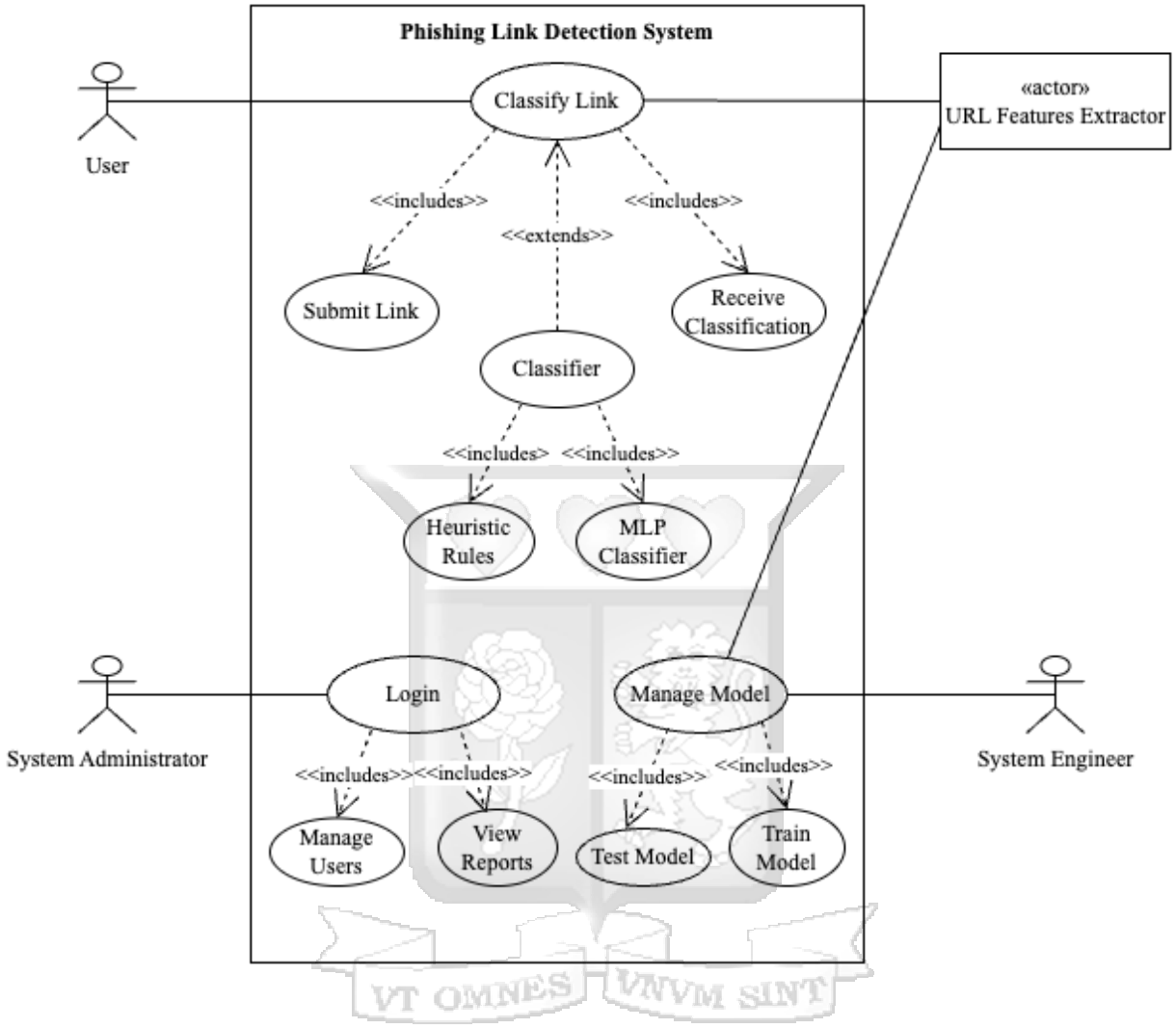


Figure 4.2: Use Case Diagram

4.5 Data Flow Diagram

A data-flow diagram (DFD) is a visual representation of how data moves through a system or a process. The DFD additionally gives details about each entity's inputs and outputs as well as the

process itself. In this study, the information is submitted by a client, processed by the system, stored and classification status relayed back to the client.

4.5.1 Context Diagram

Interactions of an internal software with external entities are depicted in a context diagram. A context diagram is primarily employed to assist organizations in comprehending the extent of a system to help in determining the best way to build a new system and its specifications or how to enhance an existing system. Figure 4.3 shows interaction between the client, engineers, and system administrators with the web-based phishing detection tool.

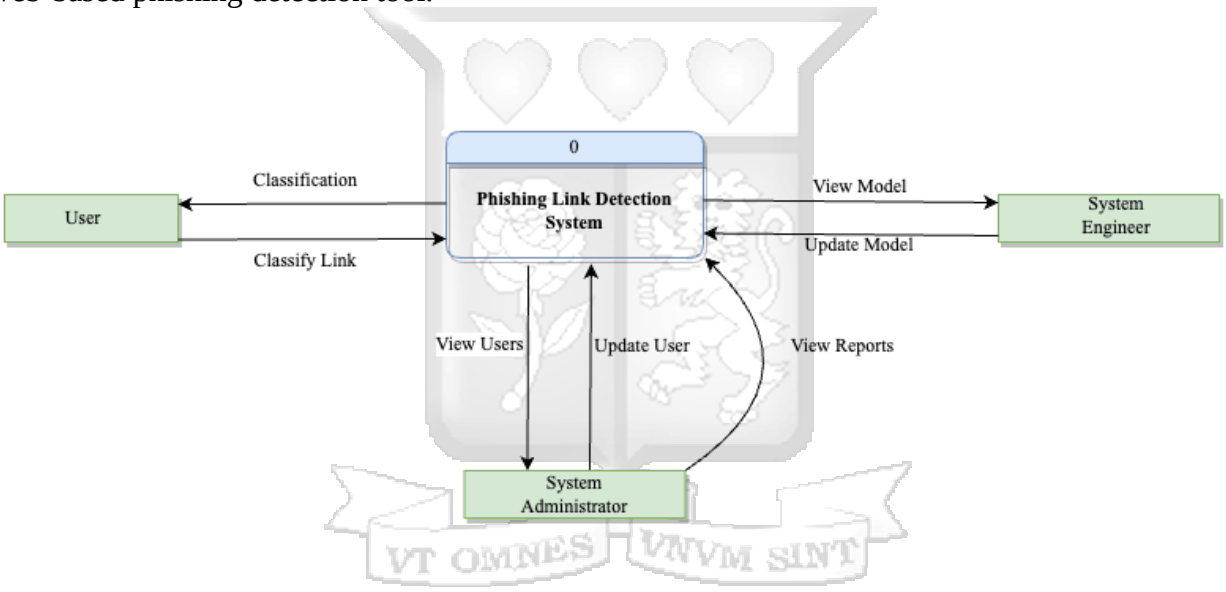


Figure 4.3: Context Diagram

4.5.2 Level 0 Diagram

Level 0 diagram takes things a step further by breaking down the information flow to complement the context diagram, which further clarifies the data flow process. Figure 4.4 shows the processes, datastores and information flow across them.

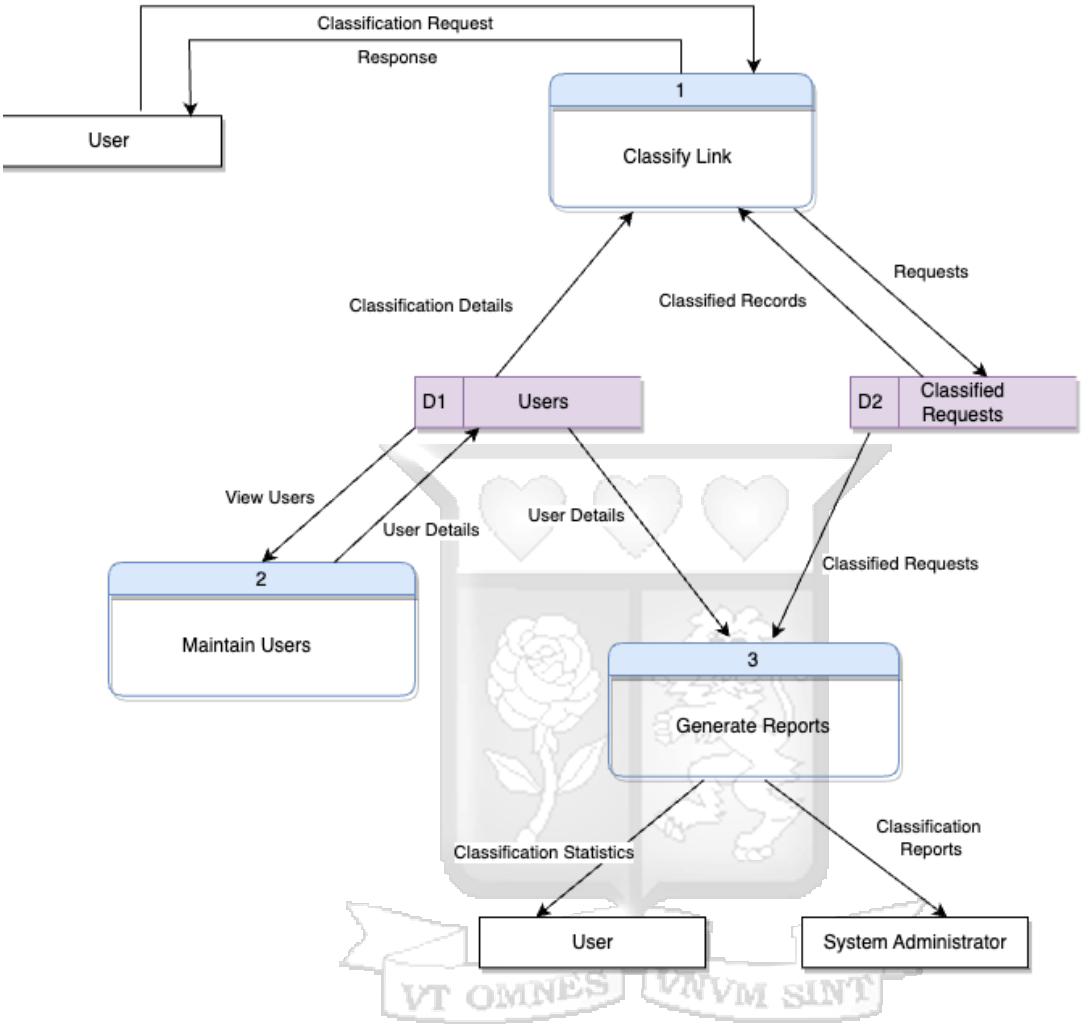


Figure 4.4: Level 0 Data Flow Diagram

4.5.3 Sequence Diagram

A sequence diagram is a diagram created using the Unified Modelling Language (UML) that shows the flow of messages during an interaction between objects. The illustrations on Figure 4.5 break down the sequence of actions to demonstrate how the phishing link detection tool accomplishes its goal.

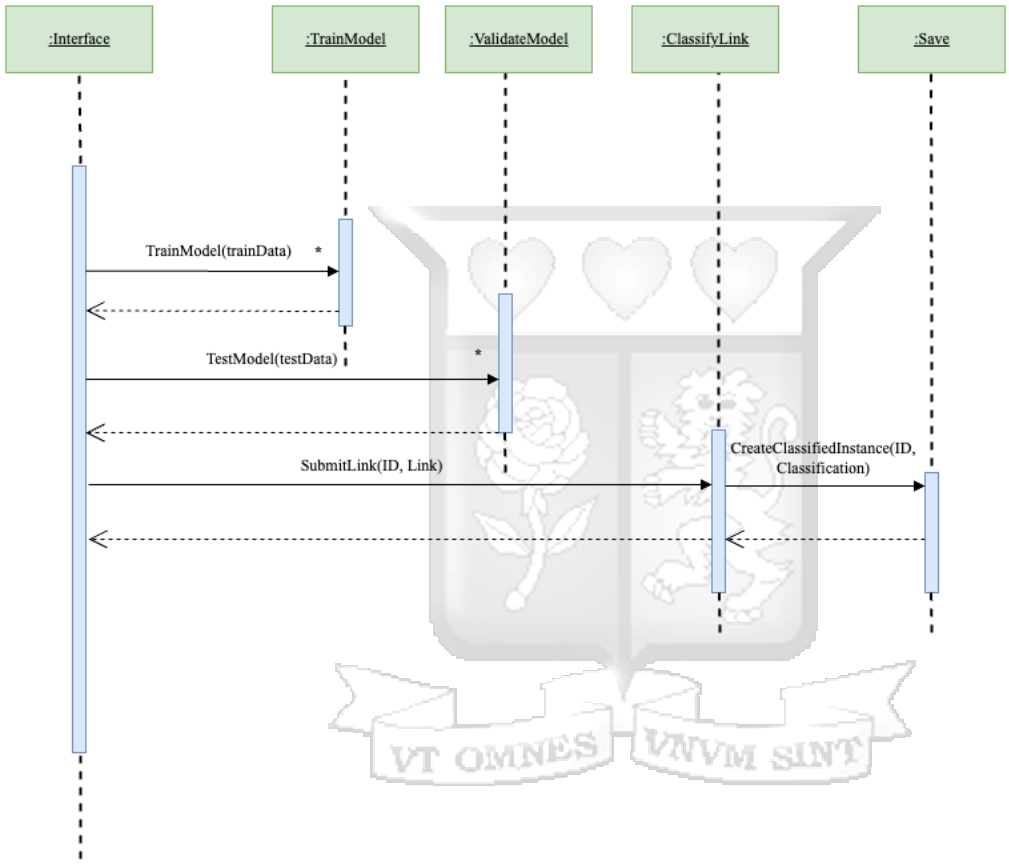


Figure 4.5: Sequence Diagram

4.5.4 Collaboration Diagram

A collaboration diagram illustrates the connections and interactions between software elements. These diagrams can be used to define each object's function as well as the dynamic behaviour of a specific use case. Figure 4.6 depicts the interaction that enable the submission, processing, and classification of link as either suspicious or benign.

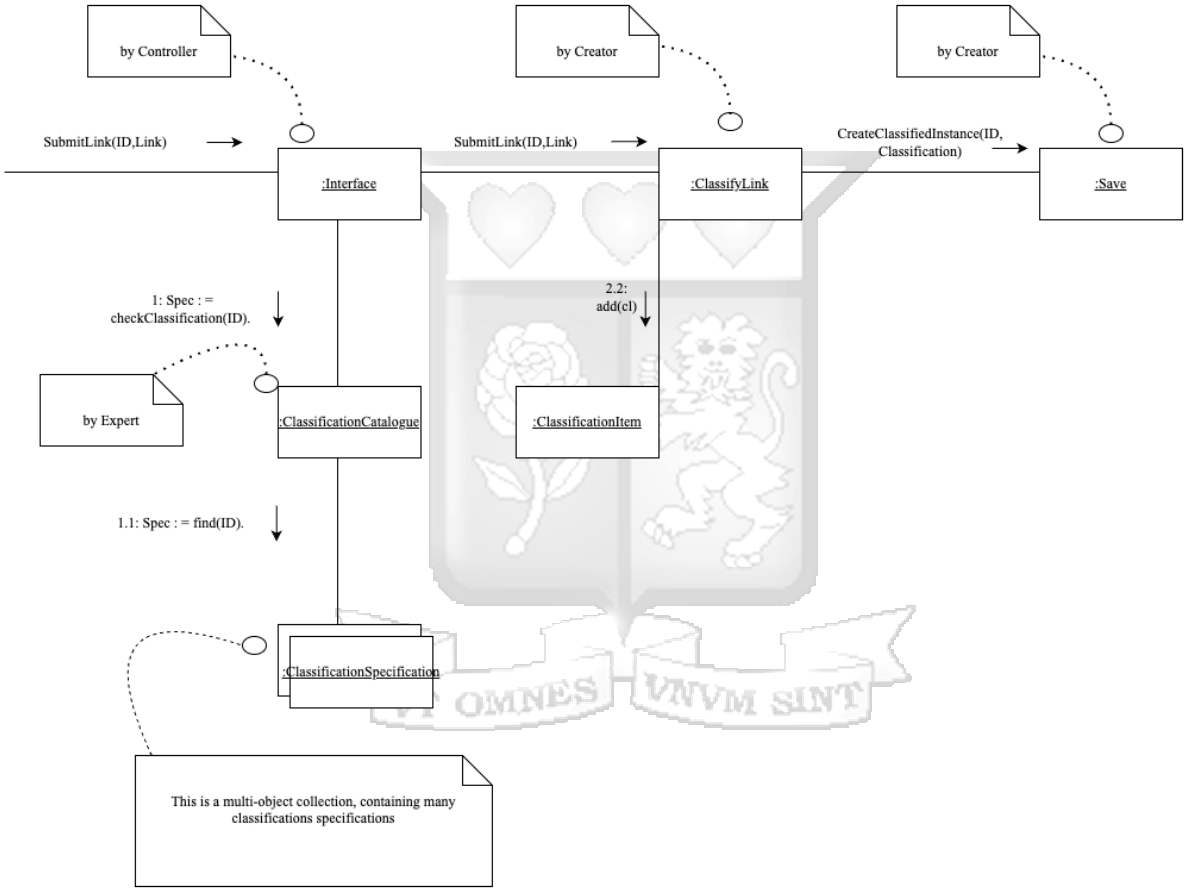


Figure 4.6: Collaboration Diagram

4.6 Class Diagram

Class diagrams are used to represent the system's components, show how they are related, and explain the functions and services that each offer. Figure 4.7 shows the components that make up the phishing link detection tool and how they offer services across the system.

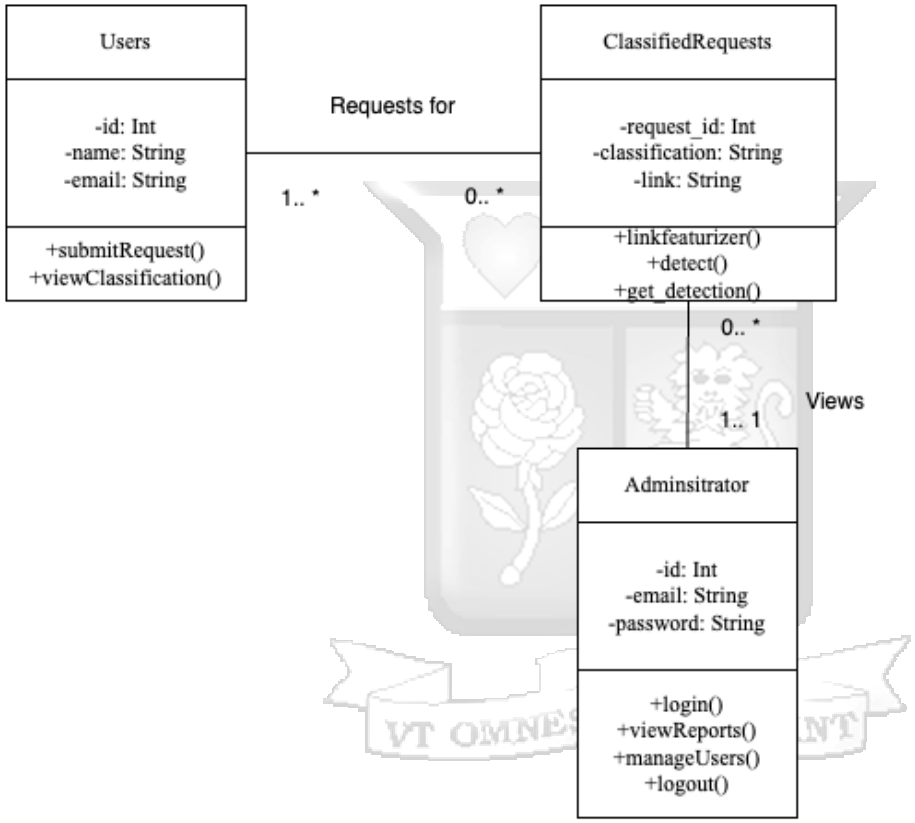


Figure 4.7: Class Diagram

4.7 Database Schema

A database schema is a blueprint representing how the database of an application is implemented, it highlights the main entities and the relationships between and or among them. Figure 4.8 shows the database schema of the phishing link detection tool including raw requests, classified requests, users, and the relationships between them.

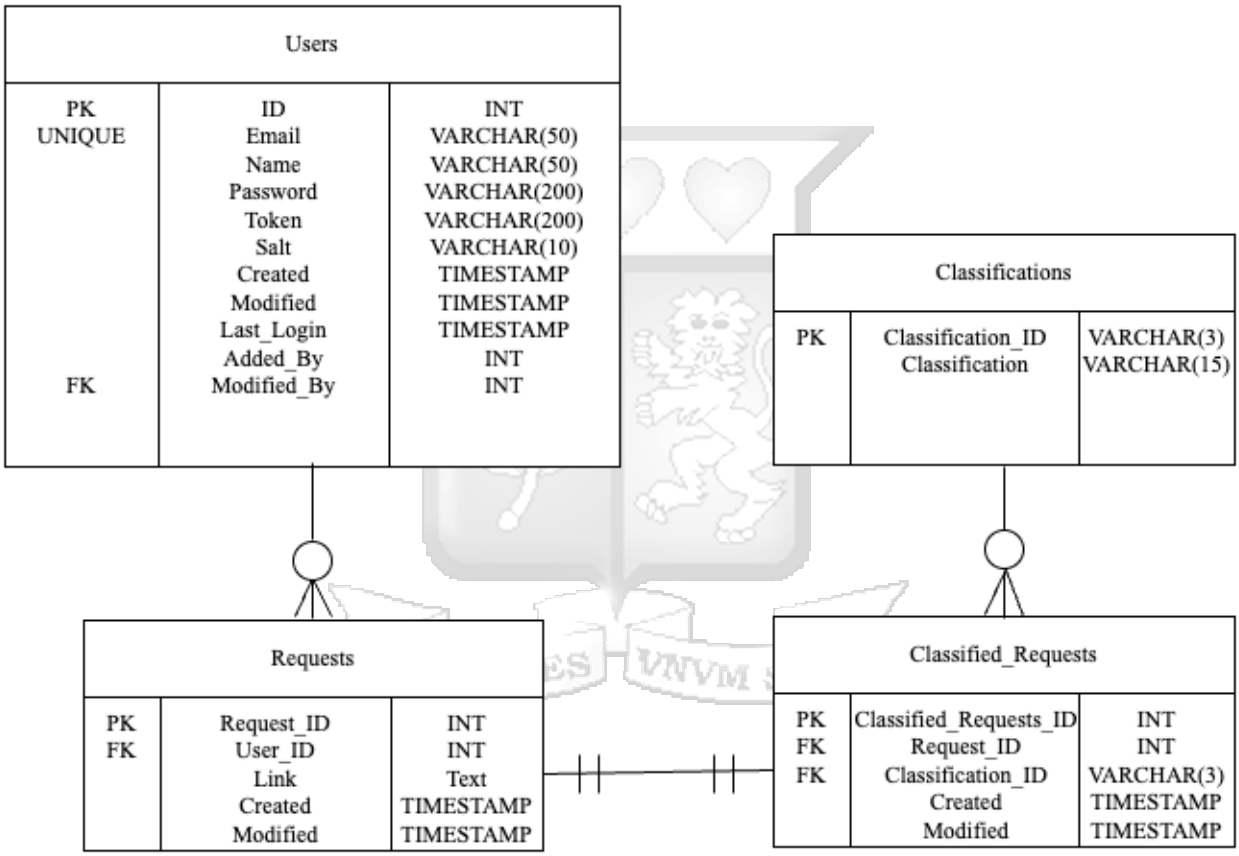


Figure 4.8: Database Schema

4.8 Wireframes

A wireframe is a schematic or blueprint used to facilitate communication between software and or website development stakeholders regarding the organizational layout of the same. Figure 4.9 shows the landing page user interface, Figure 4.10 shows the sign-up page, Figure 4.11 the login page, Figure 4.12 the requests report page and figure 4.13 the manage user's page.

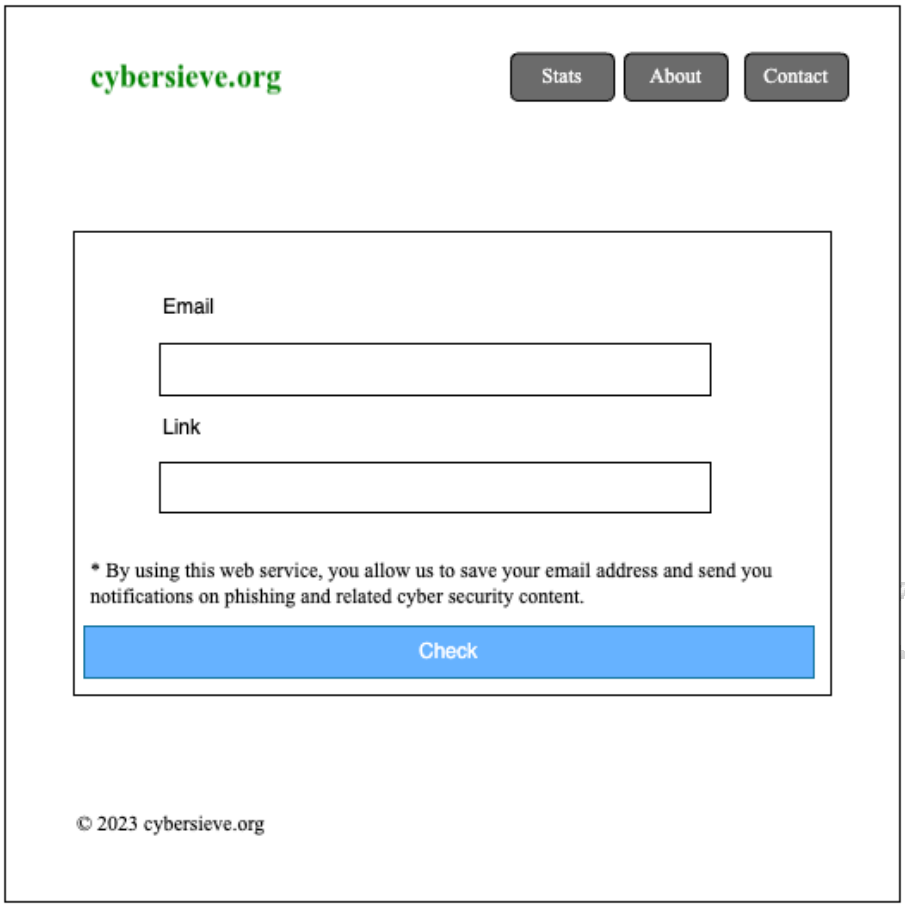


Figure 4.9: Landing Page

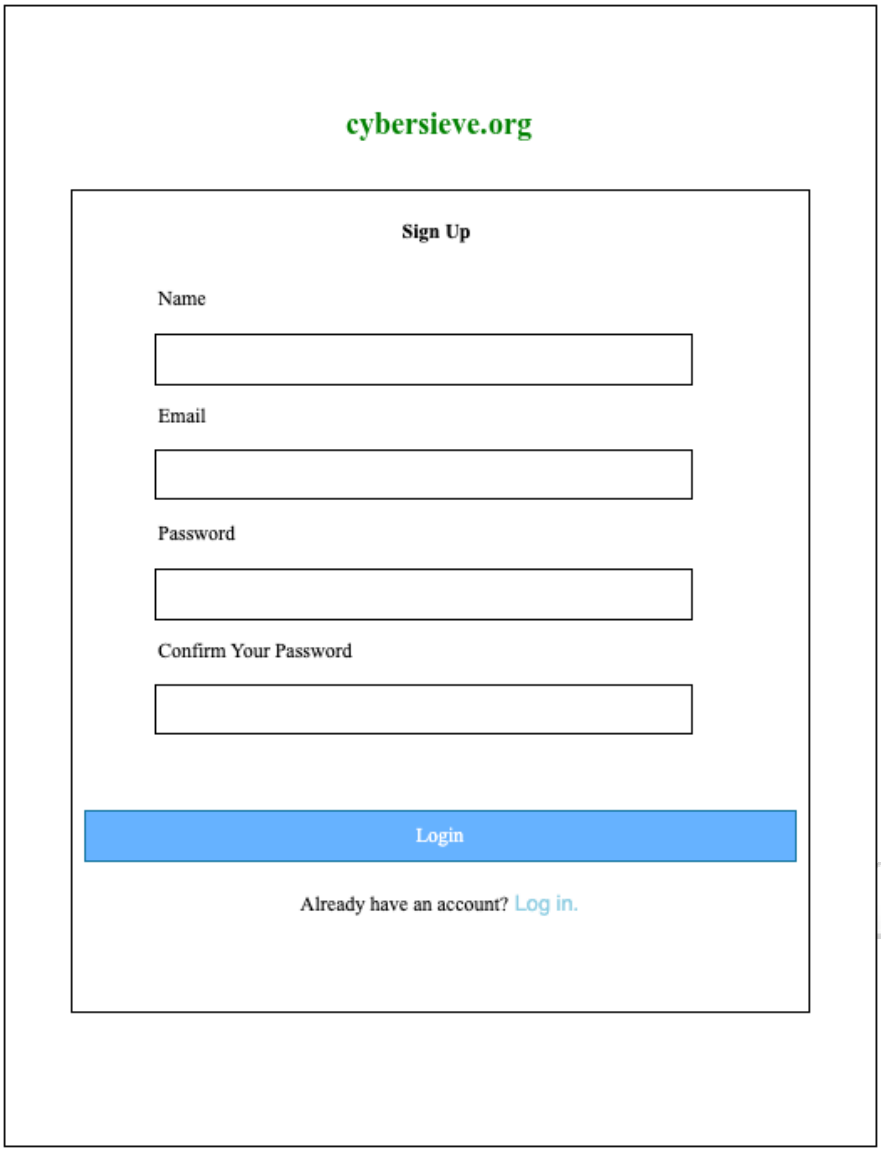


Figure 4.10: Sign Up

Figure 4.11: Log In

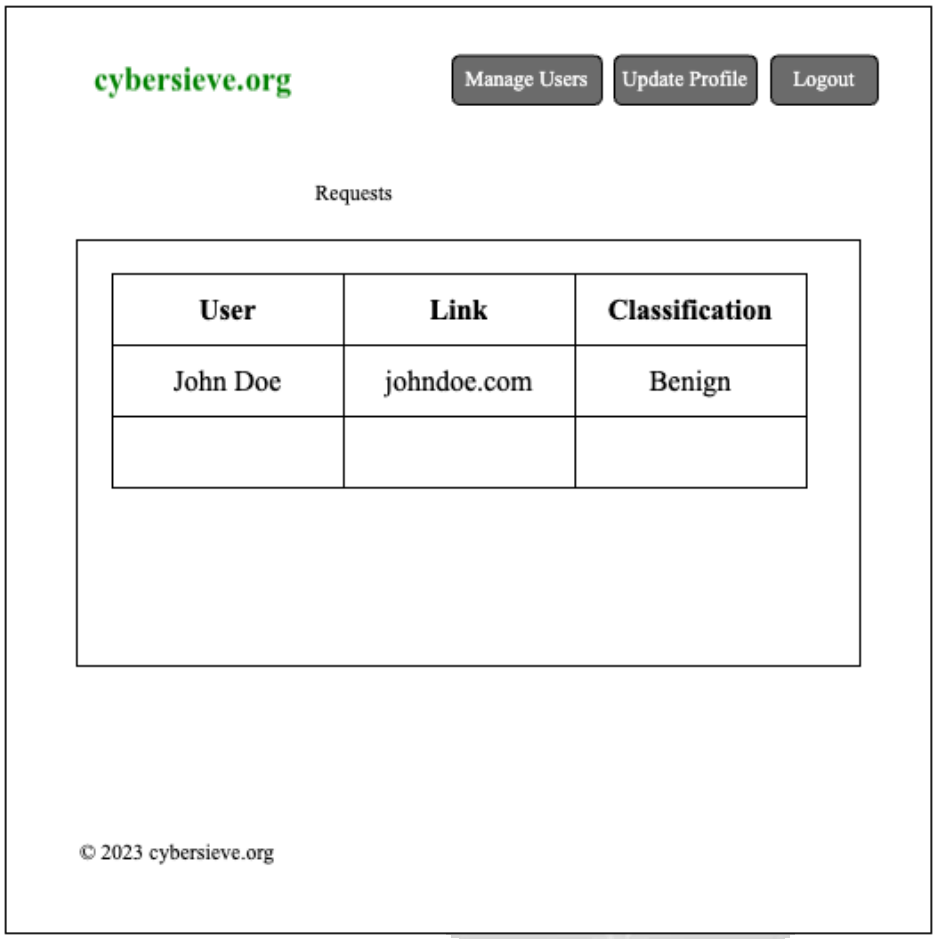


Figure 4.12: Requests Report

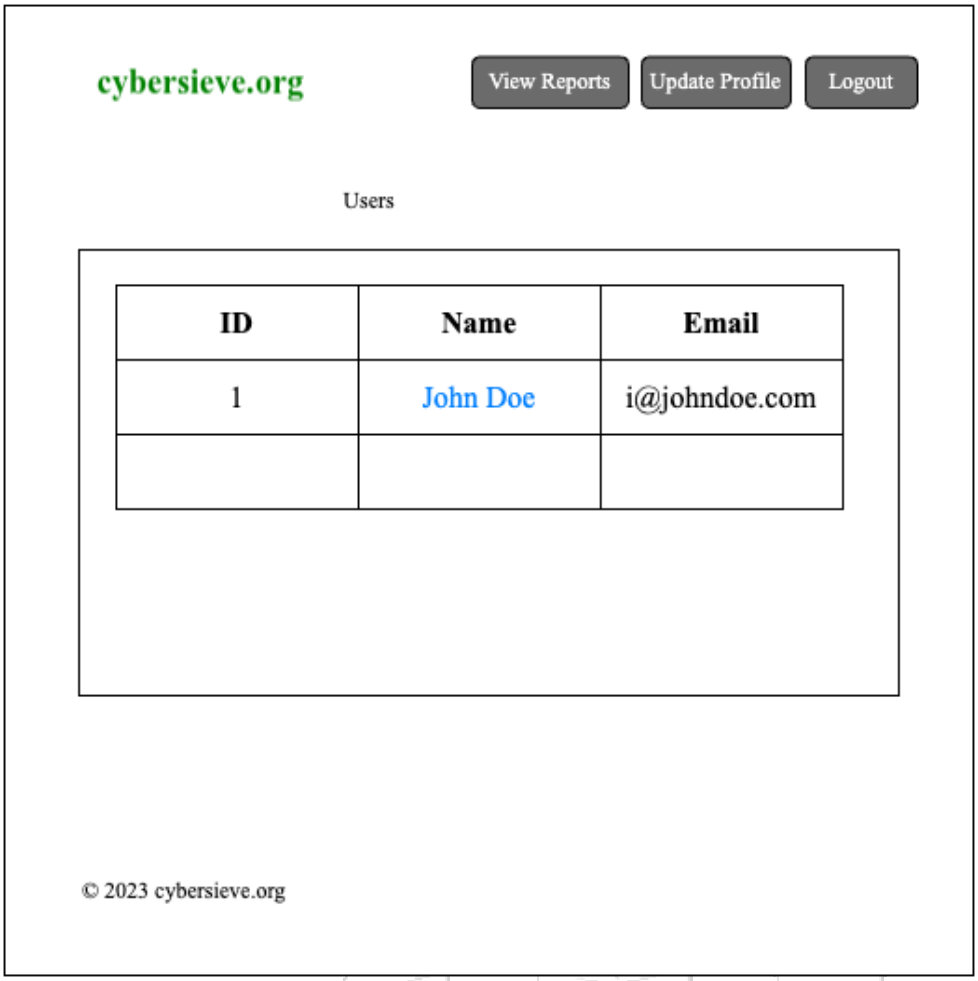


Figure 4.13: Manage Users

Chapter 5: Implementation and Testing

5.1 Introduction

This chapter outlines the processes and activities involved in the development and testing of the phishing link detection application. The initial step included identifying use cases, artifacts, and inputs or features that would help deliver optimal results. The features include lexical, host, and content attributes of links capable of differentiating malicious and benign links. The next step included creating functions for extracting these features and converting them into a format whose deep learning algorithm can consume them. The third step included defining parameters for the deep learning algorithm as well as the reporting and model state-saving functions. Finally, the saved model state was exposed to the client and or a web browser using an API for real-time links classification. The study borrows a lot from previous works on malicious link detection with citations within the paper. The data used for training and testing the model was gathered from PhishTank and Kaggle. This data was taken through features extraction, normalization, labelling, and categorization and then sprit into training and test data sets before being fed to the DNN model.

5.2 Hardware and Software Environments

Containerized microservices were used for application development and testing. Docker was used for containerisation, with three services hosted: a database, a backend, and a frontend web application. The database was built with MySQL 8.1, the front end and backend with flask which is a micro web python-based framework, the model with PyTorch machine learning framework, and data normalisation with pandas and Numpy which are both python-based packages. The use of containers in this instance was strategic to allow deployment in any cloud service's virtual machines. However, in this case, these containers were hosted locally on a MacOS running on Apple's M1 series.

Table 5.1: Hardware and Software Configurations

Software		**Hardware**	
Name	**Version**	**Item**	**Specifications**
Docker	20.10.2	CPU	Apple M1
Python	3.9.6		
Torch	1.13.1	RAM	8GB
Pandas	1.5.3		
Flask	2.2.3		
Numpy	1.24.2		
whois	0.9.27		
MySQL	8.1		

5.3 Application Components

The phishing link detection web application is made of four main components, the feature extraction, the DNN model, the database, and the API for exposing capability for inferencing using the saved DNN model state.

5.3.1 Feature Extractor

The whois in tandem with math and requests python-based packages were used to create functions highlighted in Figure 8.1 used to extract features listed in Table 8.3. Files of phish links and benign links obtained from PhishTank and Kaggle were fed into feature extracting library which in turn generated an output file containing the extracted features.

5.3.2 Model

The DNN model was implemented using PyTorch machine learning framework, Listing 5.1 shows the block of code used to implement the module within a class dnn which inherits nn or neural network class containing method __init__ with two layers containing hidden layers within them and an output binary layer and a __forward__ method which receives inputs and or features in the right format then iterating them through the hidden layers as well as churning out the output per instance which is fed forward as input into the next layer. The value t contains the feed forward output and the probability value of the target tensor format.

```
class dnn(nn.Module):

    def __init__(self):
        super(dnn, self).__init__()
        self.layer_1 = nn.Linear(features, hiddenLayer1Size)
        self.layer_2 = nn.Linear(hiddenLayer1Size, hiddenLayer2Size)
        self.layer_out = nn.Linear(hiddenLayer2Size, 1)


        self.relu = nn.ReLU()
        self.sigmoid =  nn.Sigmoid()
        self.dropout = nn.Dropout(p)
        self.batchnorm1 = nn.BatchNorm1d(hiddenLayer1Size)
        self.batchnorm2 = nn.BatchNorm1d(hiddenLayer2Size)


    def forward(self, inputs):
        t = self.relu(self.layer_1(inputs))
        t = self.batchnorm1(t)
        t = self.relu(self.layer_2(t))
        t = self.batchnorm2(t)
        t = self.dropout(t)
        t = self.sigmoid(self.layer_out(t))

        return t
```

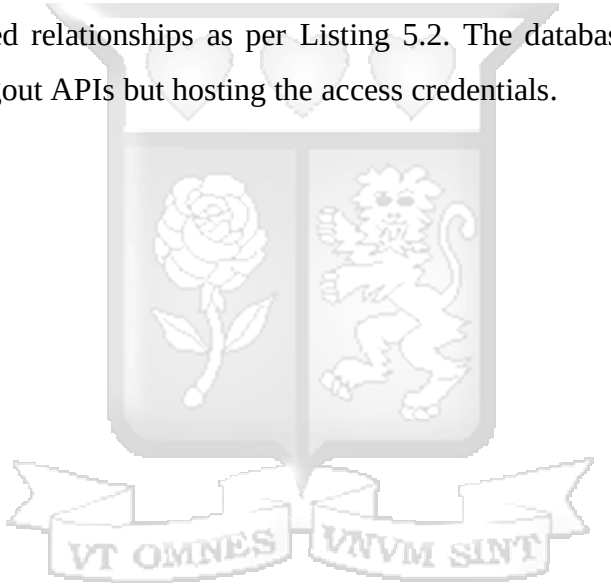
Listing 5.1: DNN Model

5.3.3 API

The API was developed on top of flask framework and exposed to a web page containing a form to capture an email address and the link to be classified. Upon form submission, the API saves the request and sends the link to the features extractor for features extraction and formatting for submission to the saved classifier state. After classification, the API receives the feedback and relays classification response to the user.

5.3.4 Database

The MySQL based database was initialized to create the user, requests, classifications and classified requests tables and related relationships as per Listing 5.2. The database saves users, request and supports the login and logout APIs but hosting the access credentials.



```sql
CREATE TABLE IF NOT EXISTS users (
    id int PRIMARY KEY AUTO_INCREMENT,
    email VARCHAR(50) UNIQUE,
    name VARCHAR(50),
    password VARCHAR(200),
    token VARCHAR(200),
    salt VARCHAR(12),
    created TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP,
    modified TIMESTAMP,
    last_login TIMESTAMP,
    added_by int,
    modified_by int,
    FOREIGN KEY(modified_by) REFERENCES users(id)
);

CREATE TABLE IF NOT EXISTS requests (
    request_id VARCHAR(20) PRIMARY KEY,
    user_id int NOT NULL,
    link text,
    created TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP,
    modified TIMESTAMP,
    -- FOREIGN KEY (user_id) REFERENCES users(user_id),
    FOREIGN KEY(user_id) REFERENCES users(id) ON DELETE CASCADE
);


CREATE TABLE IF NOT EXISTS classifications (
    classification_id VARCHAR(3) PRIMARY KEY,
    classification VARCHAR(15)
);

CREATE TABLE IF NOT EXISTS classified_requests (
    classified_requests_id int PRIMARY KEY AUTO_INCREMENT,
    request_id VARCHAR(20) NOT NULL,
    classification_id VARCHAR(3) NOT NULL,
    created TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP,
    modified TIMESTAMP,
    -- FOREIGN KEY (request_id) REFERENCES requests(request_id),
    -- FOREIGN KEY (classification_id) REFERENCES classifications(classification_id)
    FOREIGN KEY (request_id) REFERENCES requests(request_id) ON DELETE CASCADE,
    FOREIGN KEY (classification_id) REFERENCES classifications(classification_id) ON DELETE CASCADE

);

INSERT INTO users (email, name, password, token) VALUES("bensonspage@gmail.com", "Benson Kimani",
    "sha256$MnGmaYTZNwGCMn9L$9c408a0886b824ed3b7e873e4e95e5149d42c2b5fd7020035f6ed5536a630b89", "TKN03578");
INSERT INTO classifications (classification_id, classification) VALUES("SSP", "SUSPICIOUS");
INSERT INTO classifications (classification_id, classification) VALUES("LGT", "LEGITIMATE");

CREATE USER IF NOT EXISTS 'x' IDENTIFIED BY 'x';
GRANT ALL PRIVILEGES ON webappfdpl.* TO 'x'@'%';
FLUSH PRIVILEGES;
```

Listing 5.2: Database

5.4 System Implementation

The user input is received through a web form containing the user's email address and the link for classification, once the user submits the link, the system checks whether that link has been classified before and if so, creates a new instance for the current user and responds with the classification status to the user in real time. If the link has not been classified before, the system extracts the features of the link, formats it and submits to the model for inferencing. The model responds with status which is then saved and an instance of the same relayed back to the use. The API is for receiving user requests was hosted as the frontend, the administrative features to manage users and requests as backend and the database service to persist requests and access, this was achieved through containerization in docker as shown in Figure 5.1.

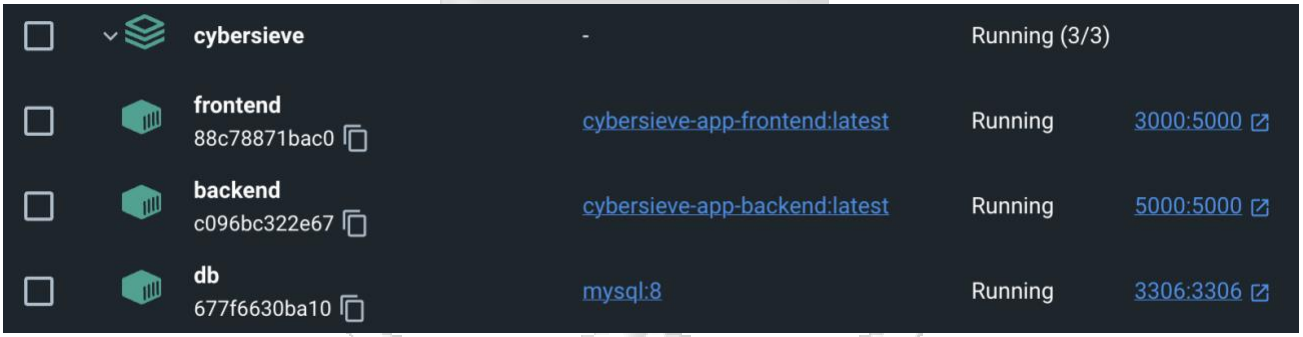


Figure 5.1: System Implementation

5.4.1 Inferencing Interface

The inferencing interface contains a trained saved model state which is exposed to the web using an API, following the DNN model training, the model state is saved as per Listing 5.4.

```
184  # Saving Model
185
186  # Specify a path
187  PATH = "state_dict_model.pth"
188  # Save
189  torch.save(model.state_dict(), PATH)
```

Listing 5.3: Save Model State

5.4.2 Testing

The main objective of the study was to develop a web application for detecting phishing links using machine learning approach. This has been accomplished through deep learning involving training of an DNN model using previous data phishing and legitimate links. Table below summarizes the tests done.

Table 5.2: Test Cases

Test Case	**Priority**	**Result**
Does the application capture the link as submitted?	High	Pass
Does the application classify link correctly?	High	Pass. (The model's accuracy has put this at 89 percent of the times)
Does the application save classification instance?	High	Pass
Does the application provide classification feedback in real time?	High	Pass
Does the application use previous classification results to provide quick feedback?	High	Pass

Chapter 6: Discussions

6.1 Introduction

The research sought to create a web application that detects phishing links using machine learning technique. Previous research has shown that deep learning increases accuracy as more high-quality data is used ("MonkeyLearn," 2022). This study attempts to capitalize on this to provide a more accurate solution. Through an API, the application receives a link from a user, classifies it, and saves an instance of the classification.

This chapter highlights how the features are extracted, analysed, normalised, and prepared for training the deep learning model as well as using the model for inferencing through an API.

6.2 Features Engineering

The quality of training data, and thus the quality of features fed into the machine learning model, determines the success of any learning endeavour. In feature engineering, features extracted were taken through analysis and normalization techniques to determine which ones better informs on whether a link is malicious or not, or whether it is benign or.

6.2.1 Feature Extraction

Host, lexical and content-based features were extracted from phishing and benign links as listed on Appendix E using requests, whois and requests python-based packages as shown on Listing 6.1.

```
l = ['tt-data/phish_links_tiny.csv', 'tt-data/benign_links_tiny.csv',
'tt-data/benign_links.csv', 'tt-data/phish_links.csv']

emp = linkfeaturizer.linkfeaturizer("").run().keys()

A = pd.DataFrame(columns = emp)
t=[]
for j in l:
    print(j)
    d=pd.read_csv(j,header=None).to_numpy().flatten()
    for i in tqdm(d):
        try:
                temp=linkfeaturizer.linkfeaturizer(i).run()
                temp["File"]=j.split(".")[0]
                t.append(temp)
        except RuntimeError:
            pass
A=A.append(t)
os.chdir('tt-data/')
A.to_csv("benign_phish_links_features.csv")
```

Listing 6.1: Benign and Phish Links Features Extraction

6.2.2 Feature Analysis

Features extracted were analysed as follows.

i. Feature Normalization

All True and or False values were converted to 1 and 0, the dependent variables were converted to 0 for phish and 1 for benign.

The values with huge range differences e.g., body length, URL age, and script length were converted into low, medium, and high categories grouped as 1 2 and 3 while zero values were left at zero. The grouping into categories was determined by use of percentiles, listing 6.2 show logic used to group

URL length into 25th, 50th and 75th percentile ranges whose values were determined by use of an excel formula as shown in Equation 2.

PERCENTILE = PERCENTILE.EXC([data], [percentile]) Equation 2

```
62      def urlLength(self):
63          total_urllength = len(self.url)
64
65          if (total_urllength <= 0):
66              urllength = 0.0
67          elif (total_urllength <= 35):
68              urllength = 0.25 # Short URL
69          elif (total_urllength <= 46):
70              urllength = 0.5 # Medium Length URL
71          elif (total_urllength <= 63):
72              urllength = 0.75 # Long URL
73          else:
74              urllength = 1.0 # Very Long URL
75
76          return urllength
```

Listing 6.2: URL Length Normalization

The values which were the same for both the phish and benign links were removed from the training dataset, i.e., if URL Has Port in String.

ii. Correlation Analysis

The correlation map on Figure 6.1 was generated across the features to show how well they related to the dependent variable and or target which is negative for phish and positive for benign link.

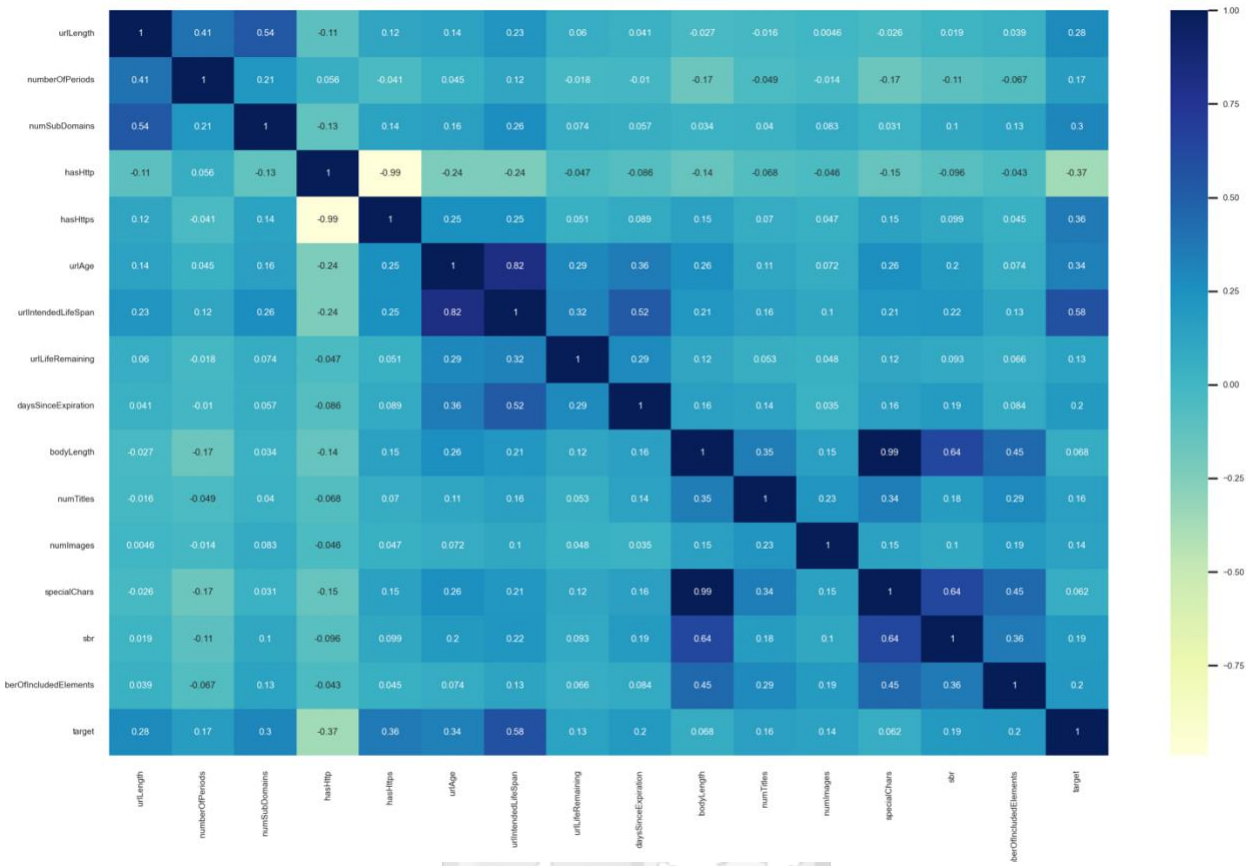


Figure 6.1: Features Correlation Map

iii. Violin plot for HTTP and HTTPS

Violin plot for HTTP and HTTPS protocol as depicted in Figure 6.4 shows that all benign links in the dataset use HTTPS while nearly half of phishing links use unsecured HTTP protocol.

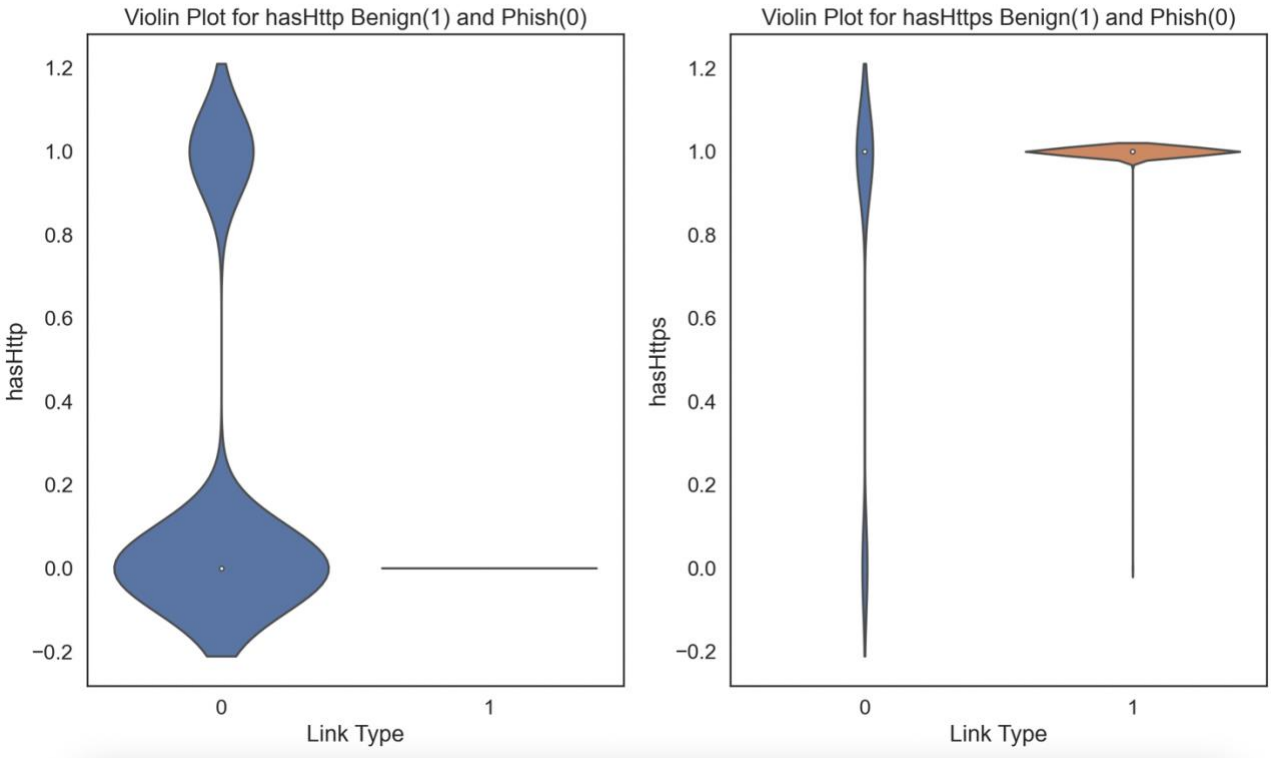


Figure 6.2: Violin plot for HTTP and HTTPS

6.2.3 Feature Selection

While the features with high correlation were considered in selection, univariate feature selection was applied in the selection of best 15 as per Figure 6.3. In addition, multiple sub domains, loading resources from other sources, URL Length, domain age, existence of HTTPS were considered key features as they have been found to be significant in detecting phishing links (Kakarla, 2022). Among the 15 best initially selected, the features combination that yielded the highest accuracy were, URL length, number of sub domains, has HTTP, has HTTPS, URL age, URL intended lifespan, body length and number of included elements.

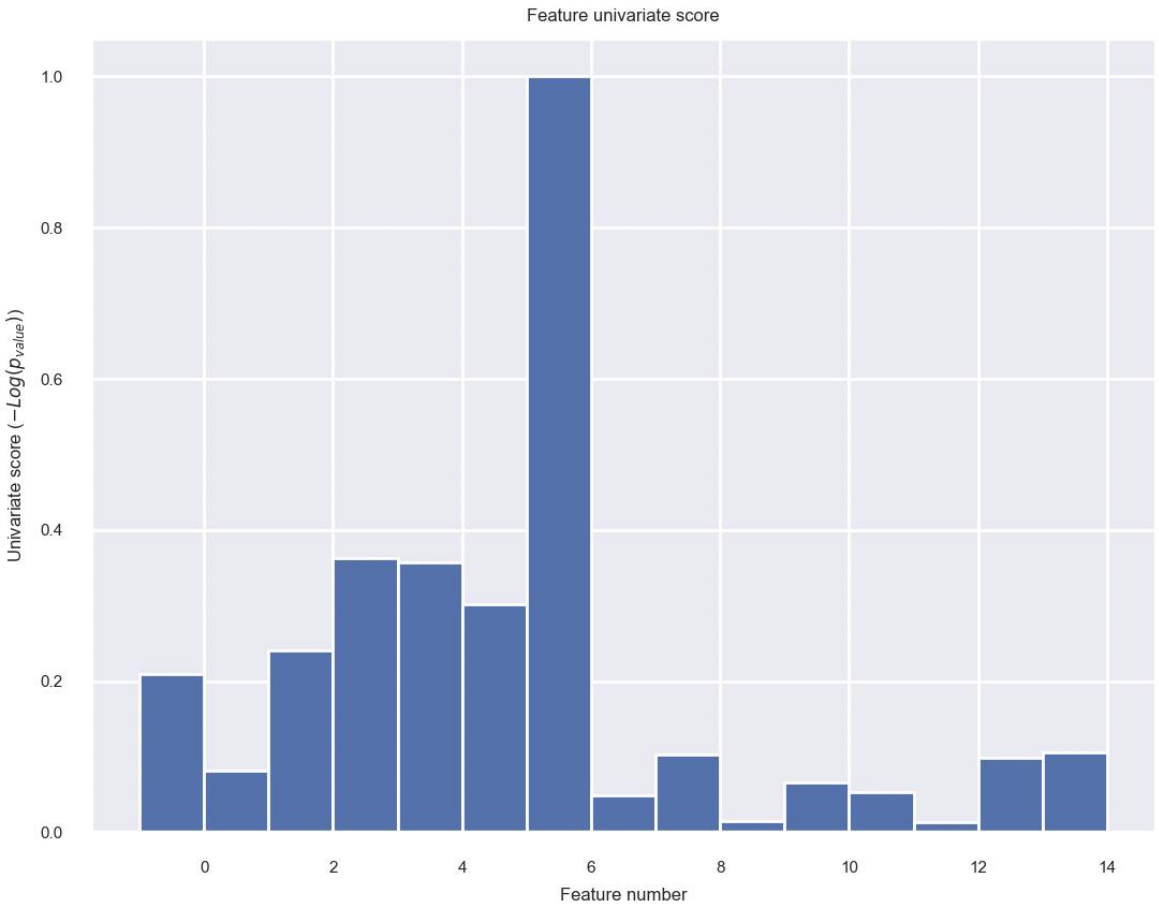


Figure 6.3: Feature Univariate Score

6.3 Application Validation

6.3.1 Model

With dataset containing 2127 samples at a distribution of 51.76 and 48.24 percent for benign and phish links respectively, the model achieved accuracy of 89.098% as show by performance of the classification model on Figure 6.4.

```
Accuracy :  89.098 %

Classification Report :
              precision    recall  f1-score   support

           0       0.88      0.90      0.89       265
           1       0.90      0.88      0.89       267

    accuracy                           0.89       532
   macro avg       0.89      0.89      0.89       532
weighted avg       0.89      0.89      0.89       532
```

Figure 6.4: Confusion Matrix

The model's learning rate stabilized at a loss rate 0.1 percent towards the fortieth Epoch as shown on Figure 6.5.

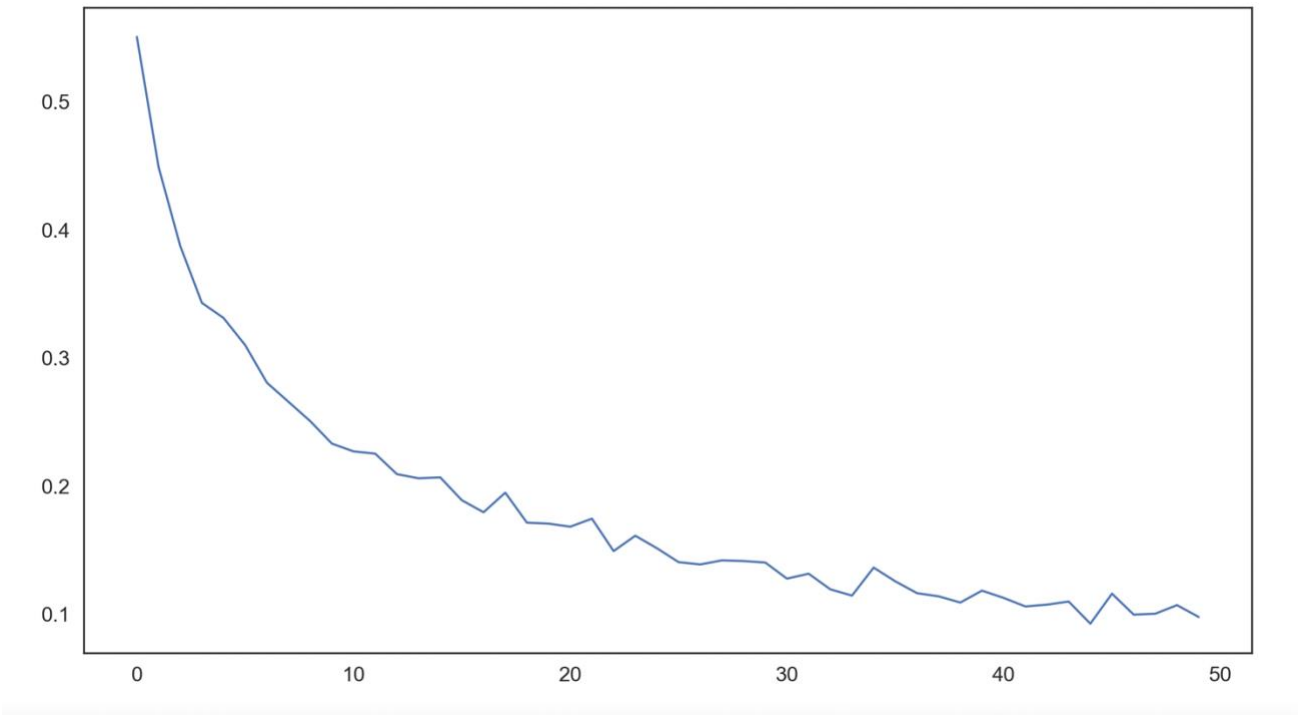


Figure 6.5: Learning Rate

Despite the high accuracy achieved, a lower accuracy state of the model was used in the deployment of the web application to production. This was since data used in the high accurate implementations included features extracted through use of third-party services which turned out to be not applicable in the real time web application implementation since some benign links with diverse top-level domain were not found in the third-party services thus leading to incorrect inferences. These features included URL age, intended lifespan, and related registration information.

Table 6.1 shows the summary of model's, data and accuracies achieved as well as the concept of better accuracies as data size increase. Further, this implementation has found optimal considering the higher accuracy of 94.27 in relation to related URLs and heurist rules-based model by (Vijayalakshmi et al., 2020) which achieved an accuracy of 93.98 percent.

Table 6.1: Model Accuracy

Data Size	**Accuracy**	**Limitation**
11430, 5715 phishing and 5715 legitimate links.	94.27	Use of third-party service leading to incorrect inference for non-listed domains.
2127 total links, 1101 benign and, 1026 phishing links with 32 features.	89.10	Use of third-party service leading to incorrect inference for non-listed domains.
2127 total links, 1101 benign and, 1026 phishing links with 18 features.	85.68	Low accuracy.

The other research considered with an accuracy as high as 97 percent (Al-Ahmadi & Al-Alyan, 2020) could as well not be implemented for this web-based classification service as the data used was not independent and thus the sources could not be relied upon during generation of features in real time.

Computation of the accuracy was calculated using Equation 3 where TP is the number of true positives, TN the number of true negatives, FP the number of false positives, and FN the number of false negatives. In this study, positives stood for benign links while negatives referred to phishing links.

$$\text{Accuracy} = TP + TN / (TP + FP + TN + FN). \text{ Equation 3}$$

Considering false positive and false negatives were important in the study, the model was further evaluated using F1-Score as shown on Figure 6.4 which is a measure of test accuracy as shown on Equation 4 where precision equals to TP divided by Predicted Positive and recall equals to TP divided by Actual Positive ("Skikit Learn," 2023).

$$F1 = 2 * (\text{precision} * \text{recall}) / (\text{precision} + \text{recall}). \text{ Equation 4}$$

6.3.2 API

The API has been validated to receive users input containing the email address and the link to be classified as well as returning classification feedback as per Figure 6.6. If a link is classified as suspicious, the user is advised to follow up with the relevant information security within their organization and or relevant cyber security support.

Further, the application developed is interoperable, it can be used independently to validate links received from all kinds of messaging platforms as well as be integrated within other messaging platforms to validate status of links before they are opened.

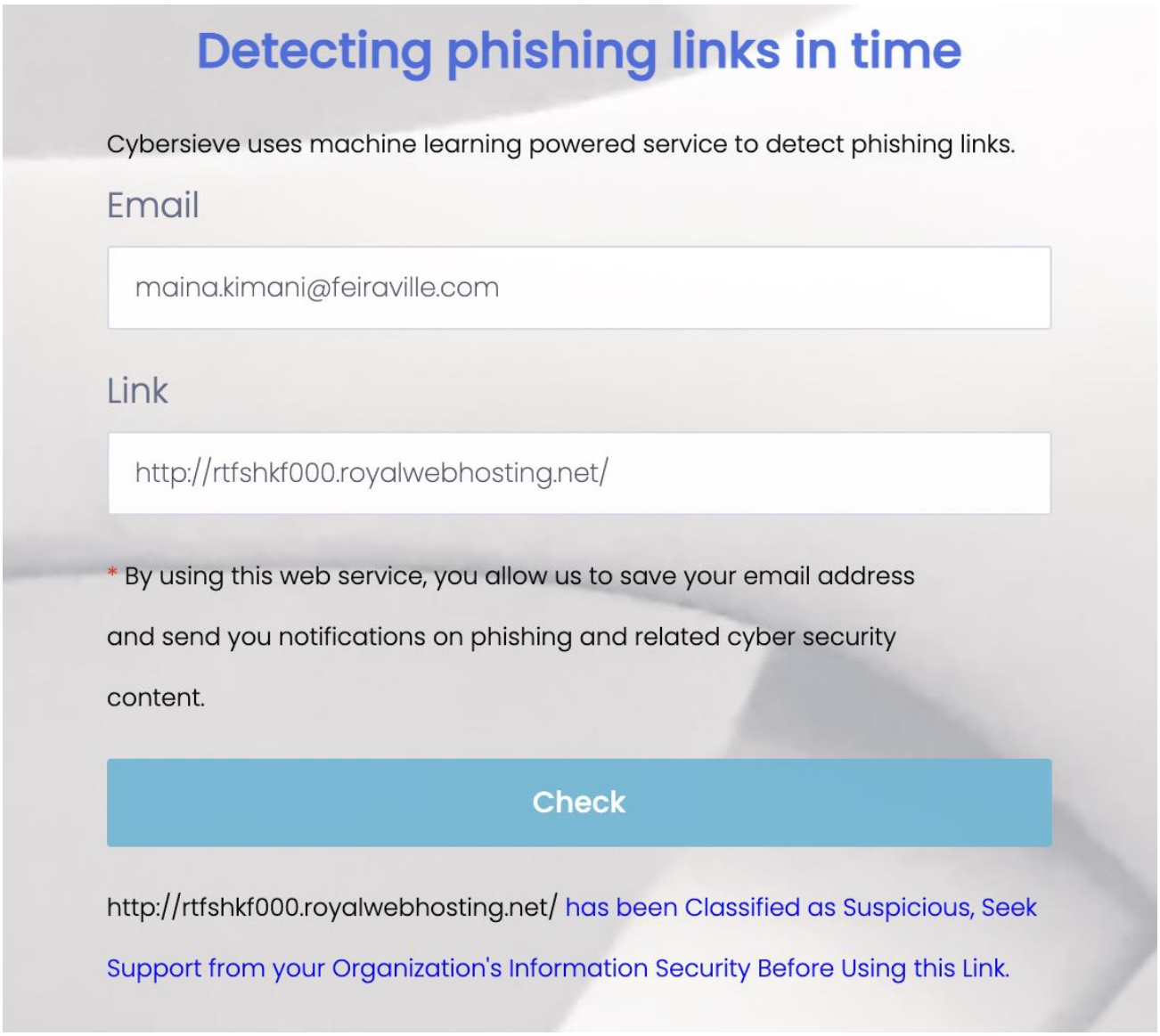


Figure 6.6: Link Classification API UI

6.4 Realtime Classification

The application has been developed and deployed inside a docker containers that can be hosted on the cloud on top of Linux and or Windows servers. The containers include three services, the database the frontend and the backend to support real-time link classification.

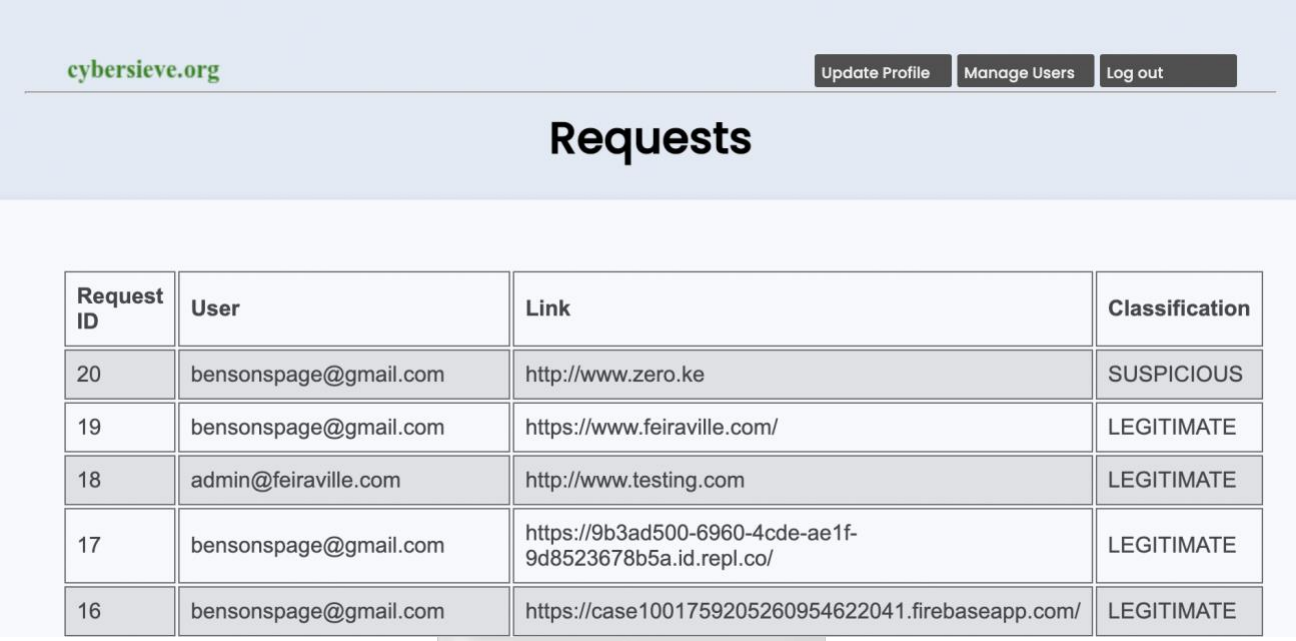
Figure 6.6 shows a real-time classification on the web interface.

6.5 Validity of the Application

The application has been validate using phishing links retrieved from ("Phistank," 2023) and benign links from Kaggle's (Kumar, K., 2018) whose features were extracted in real-time using whois and requests python-based packages.

This implementation achieved an accuracy of 94.37 percent, 0.29 percent higher than the one achieved by (Vijayalakshmi et al., 2020) using ML and heuristic rules capabilities. This implementation made use of optimal feature engineering and efficient optimization. In addition, the implementation includes use of built in heuristic rules in relation to use of IP address, Port Numbers and Secure links which are highly associated with phishing links as well as links blacklists retrieved from Phish Tank to improve on the effectiveness and efficiency of the tool.

Figure 6.7 shows classification report on the administrator's backend interface containing user email, link, and the link's classification.



Request ID	User	Link	Classification
20	bensonspage@gmail.com	http://www.zero.ke	SUSPICIOUS
19	bensonspage@gmail.com	https://www.feiraville.com/	LEGITIMATE
18	admin@feiraville.com	http://www.testing.com	LEGITIMATE
17	bensonspage@gmail.com	https://9b3ad500-6960-4cde-ae1f-9d8523678b5a.id.repl.co/	LEGITIMATE
16	bensonspage@gmail.com	https://case100175920526095462204 1.firebaseapp.com/	LEGITIMATE

Figure 6.7: Classifications Report

Chapter 7: Conclusions and Recommendations

7.1 Conclusions

The study's main goal was to create a web application for detecting phishing links using a deep machine learning approach and heuristic rules. The capacity of the systems to detect and avoid phishing baits is what makes the difference between being phished and not being phished. Hence, companies and people must constantly evaluate the technology platforms and or end points with which they interact with. This research concentrated on detecting phishing links based on previously observed features of such links versus benign ones.

The researcher examined existing literature on phishing links detection, the nature of phishing, the application of machine learning tools on phishing links detection to address the challenges and nature of phishing objectives then implemented a deep learning algorithm which was then trained and validated using different combinations of features identified using data analysis methods which achieve an optimal accuracy of 94.37% in tandem with heuristic rules and links list to address the development and testing objectives.

The researcher observed that optimal feature selection as determined by how well the feature relates to determination of the dependent variable, the amount and quality of data used played a key role in improving the accuracy of the model. The advantage of the deep model learning model used is the ability to learn and improve accuracy as more training data is used.

Further, the researcher observed that a model may perform very well during training and validation against a dataset that has complete set of the defined features. However, if the third-party services used during real-time classification does not have all the features of a specific link, then the classification

will be far more incorrect than the expected rate. This was observed during testing in real-time when whois was presented with links containing top level domains i.e. ac.ke.

7.2 Challenges

The study objectives were met as planned; however, some challenges were encountered during the research. It should be mentioned that the challenges had no significant impact on the study's outcome. First, while there were many phishing links, most of them had already been taken offline, making it impossible to extract material and host features from them. Another issue was the restriction on the number and speed of whois requests that could be made in each amount of time, which limited the number of data points that were used.

Finally, some companies have engagement polies with domain registrars not to reveal some of their registration information while some features were hard to access and time-consuming requiring high computations resources.

7.3 Recommendations

The solution is recommended for API integration with platforms that have capability to share and receive links so they can validate those links before allowing the senders to send. The model's accuracy stands at 89% meaning 11 out of a hundred classifications can be incorrect. The study proposed the following recommendations.

i. Continuous training of model since features dynamics change as well as new features emerge posing an impact on the model's prediction capability.

ii. Since deep learning algorithms get better at prediction as training data increase, this model can be improved by increasing both data points and relevant features as well as removing the less significant features.

iii. Other than .com and .net top level domains, some of the other top level-based domain's information for legitimate URLs tend not to be available for retrieval in whois hence there is need for implementation of a single service for hosting domains registration information to enable consistency in features retrieval in support of phishing links detection efforts and the fight to cybercrime.

7.4 Suggested Future Works

As internet evolve, companies automate tasks, and IOT devices becomes consumers of links, there is need to equip these solutions with phishing links detection to determine for themselves whether to access, receive and or send information to those end points. Further, this implementation can be improved by equipping it with capability to extract URLs from QR codes to classify them as well as including another ML algorithm to compliment the fact that the solution does not meet an accuracy of a hundred percent.

References

Abdelaziz, O., Sahana D., Rania, H., & Lydia, R. (2020). *A novel phishing email detection algorithm based on multinomial naive bayes classifier and natural language processing*. Columbus State University. https://doi.org/10.5220/0010412600690073

Advanced Business Portal. (2022). How many businesses use technology?. Advanced Business Portal.

Aguilar, M. (2022, March). The 5 most in-demand machine learning languages in 2022. Ironhack.

Al-Ahmadi, S. & Al-Alyan, A. (2020). *Robust URL Phishing Detection Based on Deep Learning*. Department of Computer Science, King Saud University. https://faculty.ksu.edu.sa/sites/default/files/robust_url_phishing_detection_based_on_deep_learning.pdf

Al-Saaidah, S. A. (2017). *Detecting phishing emails using machine learning techniques*. Middle East University.

Alanezi, M. (2021). Phishing detection methods: a review. University of Mosul.

Aljofey, A., Jiang, Q., Rasool, A., Chen, H., Wenyin Liu, Qu, Q. & Wang Y. (2022). *An effective detection approach for phishing websites using URL and HTML features*. Sci Rep 12, 8842. https://doi.org/10.1038/s41598-022-10841-5.

Alkhalil, Z., Chaminda, H., Liqaa, N. & Imtiaz, K. (2021). *Phishing attacks: a recent comprehensive study and a new anatomy*. Frontiers In Computer Science.

Amazon Web Services. (2021). How to serve a flask app with amazon lightsail containers. Amazon Web Services

Appelbaum, B. (2019). Top 6 software development methodologies. Plainview

Bannister, A. (2022). Washington residents' medical data exposed by phishing attack on Spokane regional health district. The Daily Swig.

Bhandari, P. (2022). Independent vs. Dependent variables | definition & examples. Scribbr.

Brownlee, J. (2020). Train-test split for evaluating machine learning algorithms. Machine Learning Mastery.

Brush, K. & Valerie, S. (2019). Agile software development. Tech Target.

Burns E. Nicole L & Linda T. (2022, February). What is artificial intelligence (AI)?. Tech Target.

Campbell, D. (2022). Gmail Spam Filter: Everything You Need to Know. Right Inbox

Camu, M. (2022). 9 examples of social media phishing schemes and how to avoid it. Red Points.

Capterra. (2022). Anti-spam Software Pricing Guide and Cost Comparison. Capterra.

Check Point. (2022). Business Email Compromise (BEC). Check Point.

Checkpoint. (2022). The 5 most expensive phishing scams of all time. Checkpoint.

David, J. (2021). How much does phishing really cost the enterprise?. Cybersecurity Dive.

Dayananda, S. (2019). Spark Mllib – machine learning library of Apache Spark. Edureka.

Desjardins, J. (2018). The rising speed of technological adoption. Visual Capitalist.

Faisandier, A. Garry R., & Rick, A. (2022). System Architecture. SEBok.

Gillis, A. S. (2020). Phishing. Tech Target.

Goundar, S. (2012). *Chapter 3 - research methodology and research method*. Research Gate

Gupta, S. (2022). Decision tree. Geeks For Geeks.

Hamilton, T. (2022). Agile methodology: what is agile model in software testing?. Guru99.

Hannousse, A. & Yahiouche, S. (2021). Web page phishing detection. Mendeley Data. V3

Haworth, J. (2022). African banking sector targeted by malware-based phishing campaign. The Daily Swig.

Ikwu, R., W. (2021). Extracting feature vectors from URL strings for malicious URL Detection.

Jaynish, P., (2014). *Design and implementation of heuristic based phishing detection techniques*. University Of Victoria

Johnson, D. (2022). What is Tensorflow? How it works? Introduction & architecture. Guru99

Kakarla, S. (2022). Phishing URL detection with python and ML. Active State.

Kalaharshaa, P., & Mehtre, B. M. (2021). *Detecting phishing sites - an overview*. University of Hyderabad.

Kelley, K. (2022). What is data analysis? Methods, process and types explained. Simplilearn.

Kumar, K. (2018). Malicious And Benign URLs. Kaggle.

Kumar, N. (2022). Naive bayes classifiers. Geeksforgeeks.

Kumar, R. (2011). *Research methodology*. SAGE Publications Ltd.

Kumaran, N. (2022). Understanding Gmail's spam filters. Google Workspace.

Mccombes, S. (2022). Sampling methods | types, techniques & examples. Scribbr.

Mehta, A. (2022). A beginner's guide to classification and regression trees. Digital Vidya.

Monkeylearn. (2022). Text classification: what it is and why it matters. Monkeylearn.

NIHR. (2019). How to disseminate your research. NIHR.

Oracle. (2023). What is machine learning?. Oracle.

Orunsolu, A. A., Sodiya, A. S., & Akinwale, A. T. (2022). *A predictive model for phishing detection*. King Saud University - Computer and Information Sciences. 34(2)

Phishing. (2022). History of phishing. Knowb4.

PhishTank. (2023). Developer information. PhishTank.

Prateek, S. (2021). Getting started with system design. Geeks for Geeks.

Price D. (2019). Understanding science: how to evaluate the research quality of scientific studies?. CQ Net

Projectpro. (2022). 15 popular machine learning frameworks for model training. Projectpro

Proofpoint. (2022). The ponemon 2021 cost of phishing study. Proofpoint.

Pytorch. (2022). From research to production. Pytorch.

Rao, R. S., & Ali, S. T. (2015). *Phishshield: a desktop application to detect phishing webpages through heuristic approach*. National Institute of Technology, India.

Redhat. (2019). What is an IDE?. Redhat.

Rehkopf, M. (2022). Scrum sprints. Atlassian.

Rosenthal, M. (2022). Must-know phishing statistics: updated 2022. Tessian.

Samson. R. (2022). What Are Malicious URLs And Links? How To Identify and Fight Them. Threat Insight

Sarangam, A. (2022). 5 important types of agile methodology (2022). Code Stringers.

Savickaitė, M. (2022). What is URL phishing, and how to avoid it?. Surfshark.

Shankar, A., Ramesh, S. & Badari, N. K. (2019). *A review on phishing attacks. Research India publications*. International Journal of Applied Engineering Research. 4(9), Pp. 2171-2175.

Simpao, C. M. (2022). 10 best python IDE & code editors [updated guide]. Hackr

Simplilearn. (2022). What is data collection: methods, types, tools, and techniques. Simplilearn.

Singh, V. (2022). Flask vs django in 2022: which framework to choose?. Hackr

Sjouwerman, S. (2020). The evolution of phishing and five tips to avoid being caught. Forbes.

Skikit Learn. (2023). sklearn.metrics.f1_score. Skikit Learn.

Streefkerk, R. (2019). Qualitative vs. Quantitative research | differences, examples & methods. Scribbr

Synopsys Editorial Team. (2017). Top 4 software development methodologies. Synopsys

Tozzi, C. (2022). 5 automated anti-phishing protection techniques. Torq.

Vadapalli, P. (2022). Naive bayes classifier: pros & cons, applications & types explained. Upgrad.

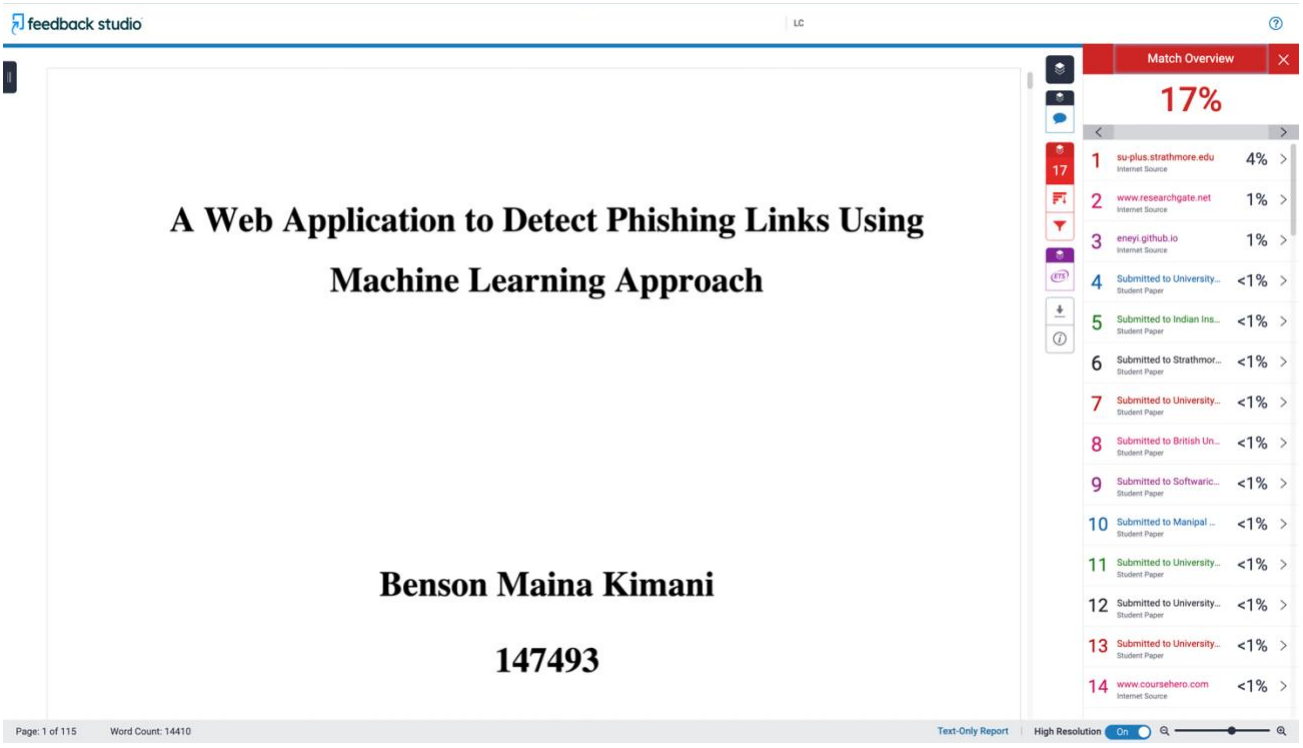
Vijayalakshmi, M., Mercy, S., Ming, H. Y., & Raja, M. U. (2020). *Web phishing detection techniques: a survey on the state-of-the-art, taxonomy and future directions*. The Institute of Engineering and Technology. https://doi.org/10.1049/iet-net.2020.0078.

Warren, K. & Eunice, R. (2020). Qualitative data analysis methods 101. GradCouch

Zola, A. (2022). Development environment. Tech Target.

Appendices

8.1. Appendix A: Plagiarism Report



8.2. Appendix B: Ethical Approval Letter

Strathmore UNIVERSITY

15th February 2023

Mr Kimani Benson Maina,
maina.kimani@strathmore.edu

Dear Mr Kimani,

RE: A Web Application to Detect Phishing Links Using Machine Learning Approach

This is to inform you that SU-ISERC has reviewed and **approved** your above **SU- master's** research proposal. Your application reference number is **SU-ISERC1585/23.** The approval period is from **15th February 2023 to 14th February 2024.**

This approval is subject to compliance with the following requirements:

i. Only approved documents including (informed consents, study instruments, and MTA) will be used
ii. All changes including (amendments, deviations, and violations) are submitted for review and approval by SU-ISERC.
iii. Death and life-threatening problems and serious adverse events or unexpected adverse events whether related or unrelated to the study must be reported to SU-ISERC within 48 hours of notification
iv. Any changes, anticipated or otherwise, that may increase the risks or affect the safety or welfare of study participants and others or affect the integrity of the research must be reported to SU-ISERC within 48 hours
v. Clearance for the export of biological specimens must be obtained from relevant institutions.
vi. Submission of a request for renewal of approval at least 60 days prior to the expiry of the approval period. Attach a comprehensive progress report to support the renewal.
vii. Submission of an executive summary report within 90 days of completion of the study to SU-ISERC.

Before commencing your study, you will be expected to obtain a research license from National Commission for Science, Technology, and Innovation (NACOSTI) https://research-portal.nacosti.go.ke/ and obtain other clearances needed.

Yours sincerely,

for: **Dr Ben Ngoye,**
Secretary; SU-ISERC

Cc: Mr Ambrose Rachier,
Chairperson; SU-ISERC

STRATHMORE UNIVERSITY INSTITUTIONAL SCIENTIFIC AND ETHICAL REVIEW COMMITTEE
(SU-ISERC)
15-Feb-2023
Email:ethicsreview@strathmore.edu
P.O BOX 59857-00200
NAIROBI-KENYA

Ole Sangale Rd, Madaraka Estate. PO Box 59857-00200, Nairobi, Kenya. Tel +254 (0)703 034000
Email admissions@strathmore.edu www.strathmore.edu

8.3. Appendix C: Use Cases

Table 8.1A: Manage Users Use Case

Use Case: Manage Users
Primary Actor: Administrator
Preconditions: Administrator has valid login credentials. Administrator logs into the system.
Post Conditions: User changes are saved. The changes in user include email, name, and modified date.

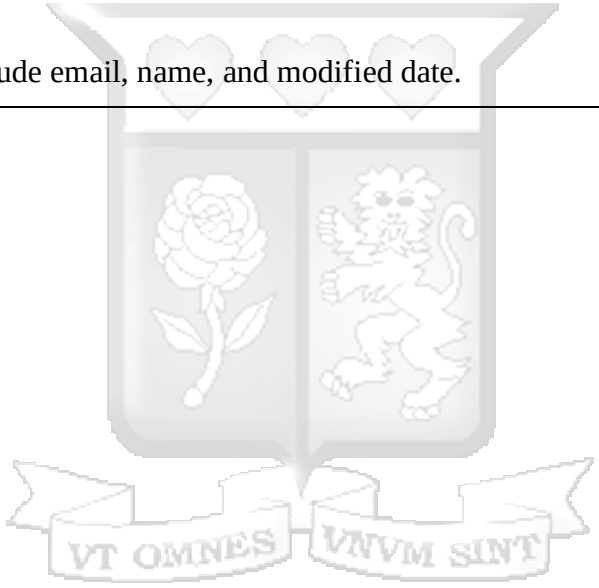


Table 8.1B: Manage Users Use Case

Main Success Scenario	
Actor Responsibility	**System Responsibility**
1. Administrator accesses the application via the browser.	
2. Administrator navigates to manage user's page.	
3. Administrator clicks on user who they want to update.	
	4. System opens the users' details in editable form.
5. Administrator makes changes and clicks update button.	
	5. System saves the changes made.
6. Administrator exits the browser.	

Table 8.2A: Manage Model Use Case

Use Case: Manage Model
Primary Actor: System Engineer
Preconditions: System has access to the machine model host. System engineer can access the machine learning model interface.
Post Conditions: Machine learning model is trained. The model state is saved.

Table 8.2B: Manage Model Use Case

Main Success Scenario	
Actor Responsibility	**System Responsibility**
1. System engineer accesses the machine learning model host.	
2. System engineer defines the model training parameters.	
3. System engineer feeds the model with test and training data.	
	4. The model learns and validates using the data provided.
	5. The model displays validation reports.
6. System engineer saves the model state for inference.	
7. System engineer exits the model interface.	

8.4. Appendix D: Consent Form

By using this website to validate link legitimacy, you allow us to save your email address and send you notifications on phishing and related cyber security content.

Email Address

Accept

8.5. Appendix E: Features

Table 8.3A: URL Features

Feature	**Description**
Lexical Features	
Entropy	The randomness factor or uncertainty in URLs.
has_ip	Whether the URL contains and IP address.
numDigits	Number of Digits in the URL.
urlLength	Total Number of characters in the URL.
numParameters	Number of values making up a query if the URL is a query.
numFragments	Number of optional components in the URL that lead to another primary part of the resource.

Table 8.3B: URL Features

Feature	**Description**
isEncoded	If URL is encoded using a "%" and a two-character hex value corresponding to their UTF-8 character.
numEncodedChar	Number of encoded characters.
numberOfPeriods	Number of "." In the URL.
urlHasPortInString	Whether URL contains a port number.
Host Features	
numSubDomains	Number of Subdomains.
hasHttp	Whether ULR has http.
hasHttps	Whether URL has https.
urlAge	Days since the URL was registered.
urlIntendedLifeSpan	Number of days from expiration to registration.
urlLifeRemaining	Number of days left to expiration.
urlIsLive	Whether the URL is online.
Content Features	
bodyLength	Total Number of characters in the URL's HTML page.

Table 8.3C: URL Features

Feature	**Description**
numTitles	Total number of titles from H1 to H6 in the URL's HTML page.
numImages	Total number of images.
numLinks	Total number of links embedded in the URL's HTML page.
scriptLength	Total number of characters in embedded scripts.
specialChars	Total number of characters in the URL's HTML page.
scriptToSpecialCharsRatio	The ratio of embedded script to special charters in the HTML page.
scriptTobodyRatio	The ratio of the total length of embedded scripts to total number of characters in the HTML page.
bodyToSpecialCharRatio	The ratio of body length to total special characters in the URL's HTML page.
numberOfWhitespace	Number of whitespaces in the HTML page.
numberOfIncludedElements	Number of included external elements.
numberOfDoubleDocuments	Number of HTML tags e.g., <body>, <html> <head> that are repeated.
numberOfIframes	Total number of iframe tags on the HTML page.

8.6. Appendix F: Code Snippets

Figure 8.1 shows the API for receiving users email address and link for classification.

```
58  @app.route('/', methods = ['GET', 'POST'])
59  def index():
60
61      form = Request()
62
63      if form.validate_on_submit():
64
65          email_address=form.email_address.data
66          link=form.link.data
```

Figure 8.1: Classification Request API

Figure 8.2 shows the class and initial method for features extraction.

```
class linkfeaturizer(object):
    def __init__(self, url):
        self.url = url
        self.domain = url.split('//')[-1].split('/')[0]
        self.today = datetime.now().replace(tzinfo=None)
        self.html = self.__get_html()
        self.now = datetime.now()
        self.today = datetime.now()

        try:
            self.whois = whois.query(self.domain).__dict__
        except:
            self.whois = None

        try:
            self.response = get(self.url)
            self.pq = PyQuery(self.response.text)
        except:
            self.response = None
            self.pq = None

    ## Lexical URL Features
    def entropy(self):
        string = self.url.strip()
        prob = [float(string.count(c)) / len(string) for c in dict.fromkeys(list(string))]
        entropy = sum([(p * math.log(p) / math.log(2.0)) for p in prob])
        return entropy

    def ip(self):
```

Figure 8.2: Feature Extraction