KINGDOM OF SAUDI ARABIA
Ministry of Education
Taibah University
College of Computer Science and Engineering
(Male Section)

المملكة العربية السعودية
وزارة التعليم
جامعة طيبة
كلية علوم و هندسة الحاسب الآلي
(قسم الطلاب)

PhishWarden

Machine Learning System for Detecting Phishing Attacks

Graduation Project (2)

By

Abdulaziz Faisal Alharbi	4100569
Osama Ali Ahmed Naji	4108881
Safwan Ibrahim Alsubhi	4100780
Mazen Abdullah Ali Qalo	3703641

A project submitted in partial fulfilment of the requirements for degree of Bachelor of science (Computer Science)

Supervised by

Dr. Mohammed M. Alsuraihi

2st Semester - Academic Year 1445/1446 (2023/2024)

Abstract

Phishing attacks have become one of the most prominent attacks faced by internet users, governments, and service-providing organizations. In a phishing attack, the attacker(s) collects the client's sensitive data (i.e., user account login details, credit/debit card numbers, etc.) by using spoofed emails or fake websites.

Our project employs machine learning technology for detection of phishing URLs by extracting and analyzing various features of legitimate and phishing URLs. and Natural Language Processing (NLP) to extract information from emails, organizing them into corpora. Using a feature-based approach, we analyze these corpora to determine whether the content exhibits characteristics indicative of phishing attempts. By leveraging NLP for semantic analysis, our system assesses language patterns, context, and structural elements within the emails. Features such as suspicious keywords, irregular sender behavior, and abnormal formatting are utilized to identify potential phishing threats. This focused approach allows for an efficient and accurate detection mechanism tailored to the nuances of email communications. Through the integration of NLP and feature-based analysis, our project aims to enhance the capability of distinguishing between legitimate and phishing emails, contributing to a more robust defense against cyber threats.

Keywords: Phishing Attack; Cybersecurity; Machine Learning; Real-time Monitoring; Threat Detection.

Acknowledgements

We begin by expressing our deepest gratitude to Allah, whose guidance and blessings paved the way for the successful completion of this project. Our hearts are filled with appreciation for the unwavering support and capacity bestowed upon us. We want to take a moment to express our deepest appreciation and gratitude to each team member, for the exceptional dedication, hard work, and unwavering commitment throughout every phase of this project. Our individual and collective efforts have been nothing short of indispensable.

Gratitude also goes to our families for their unyielding support and admiration, providing a pillar of strength throughout this journey. Special thanks are due to all team members who collaborated seamlessly, offering mutual assistance in various facets of the work.

A special acknowledgment goes to our esteemed supervisor, Dr. Mohammad M. Alsuraihi. His guidance, encouragement, and invaluable insights were instrumental at every stage of our endeavor. We are profoundly thankful for their direction and assistance, which significantly contributed to the fruition of this work. Our sincere appreciation extends to the entire faculty at Taibah University, College of Engineering and Computer Science. The knowledge and skills imparted during courses such as software engineering, software analysis, and modeling have been foundational to our project. Lastly, we extend our thanks to everyone who played a role in motivating and aiding us in the development of this project.

Table of Content

List of Figures

List of Tables

List of Abbreviations

Abbreviations	Full description
AI	Artificial Intelligence
API	Application Programming Interface
CEO	Chief Executive Officer
CIA	Central Intelligence Agency
DOM	Document Object Model
ES	Elasticsearch
FA	Feature Analysis
FN	False Negative
FP	False Positive
FR	False Rejection
HTML	Hypertext Markup Language
IBM	International Business Machines Corporation
IDF	Inverse Document Frequency
IEEE	Institute of Electrical and Electronics Engineers
IET	Institution of Engineering and Technology
IP	Intellectual Property
MA	Moving Average
ML	Machine Learning
MLDL	Machine Learning and Deep Learning
MLP	Multilayer Perceptron
MVP	Minimum Viable Product
NIST	National Institute of Standards and Technology
NLP	Natural Language Processing
PA	Precision Agriculture
PCA	Principal Component Analysis
PIN	Personal Identification Number
POS	Point of Sale
RF	Radio Frequency
SDLC	Software Development Life Cycle
SMS	Short Message Service
SVD	Singular Value Decomposition
SVM	Support Vector Machine
TF	Term Frequency
UML	Unified Modeling Language
URL	Uniform Resource Locator
USB	Universal Serial Bus
XG	Extreme Gradient
XGB	Extreme Gradient Boosting

Chapter 1. Introduction

1.1 Introduction

The detection and prevention of phishing attempts have become increasingly important in the ever-changing cybersecurity landscape. Phishing attacks, and misleading attempts to trick individuals or organizations into disclosing sensitive information, have become more sophisticated and continue to pose a danger to internet security. To effectively address this threat, the incorporation of machine learning has emerged as a potent and flexible method. Machine learning, a subset of artificial intelligence, uses data-driven techniques to identify patterns, anomalies, and behavioral clues that help differentiate authentic messages from fraudulent ones. This convergence of cybersecurity and machine learning presents a promising paradigm for proactively recognizing and defeating phishing attempts. The purpose of this project is to highlight the essential principles and benefits of employing machine learning for phishing attack detection, highlighting its potential to revolutionize cybersecurity practices.

The project is divided into two phases. This report documents phase 2 of the project, with more focus on the design, implementation, and testing phases of our proposed system which was introduced in phase 1. The technical aspects of analysis and design in phase 1 of the project was tested and the following modifications was suggested:

- Conducting in-depth research on related work and similar existing systems, with focus on design and implementation aspects.
- Highlighting the machine learning algorithms to be applied on the solution domain of our proposed system, with consideration to high precision in prediction and results.
- Collecting the datasets to be used for training and testing the system model.

In this chapter, we shed light on the problem of phishing attacks and the proposed solution for it. The chapter lists the goal and objectives of this project, the methodology, and the plan it followed. Finally, the chapter ends with a summary of the chapter sections and also gives a link to the work documented in Chapter 2.

1.2 Problem Need Solution

Phishing attacks are a common and deceptive cybersecurity threat that involves malevolent actors trying to trick people or organizations into disclosing private information like bank account information, login passwords, or personal information. These assaults usually manifest as false emails, texts, or websites that impersonate reliable organizations in an attempt

to trick victims into doing things that jeopardize their security. Phishing attacks have the potential to have serious repercussions, such as identity theft, financial losses, data breaches, and reputational harm to a person or company.

Our solution revolves around the advanced application of machine learning to tackle the ever-evolving challenge of phishing attack detection. Our goal is to proactively detect and stop fraudulent phishing attempts by utilizing machine learning techniques. We can quickly distinguish between genuine and phishing messages by analyzing trends, behaviors, and anomalies in digital interactions thanks to machine learning, a data-driven methodology. Organizations and individuals can more successfully protect against the ubiquitous threat of phishing attempts with this proactive and adaptable approach. We are at the vanguard of the fight against phishing by incorporating machine learning into our solution, providing improved security and comfort in an increasingly linked digital world. This methodology has been applied in numerous prior studies as reviewed in Chapter 2. The benefits of this approach are countless, but can be focused to:

- **Real-Time Protection:** Immediate detection and prevention of phishing threats as they occur.
- **Multi-Channel Coverage:** Protection across various communication channels, including email, messaging, and websites.
- **Cost-Efficiency:** Automation reduces the need for manual monitoring, saving resources.
- **Proactive Defense:** Identification of threats before they cause damage.

1.3 Aim and Objectives

The aim of this project is to finish the design, implementation, and testing of our proposed system to detect phishing attacks and threats on emails and websites.

To fulfill this goal, we worked on achieving the following objectives:

1. To review related work and similar existing system in terms of design, implementation, and testing aspects.
2. To complete the design phase steps with consideration to the coding environment using Python and PHP Laravel programming languages.
3. Competing the implementation phase regarding reuse and code implementation.
4. Applying a validation method for testing the system and satisfying its requirements.
5. To list the lessons learnt from working as a team in this project.

1.4 Project Methodology

We applied the following strategy for each of the previous objectives in order to achieve them: To achieve the first objective (To review earlier, related work to our project.), we searched for several kinds of related subjects and analyzed a number of systems that were already in place as well as internet pages, papers, conferences, and research journals. Additionally, when completing the writing assignment, we referenced and cited using the IEEE approach.

For the second objective, which revolves around completing the system design with future implementation capabilities, we systematically defined and understood the problem, scrutinized similar systems, and referenced scientific papers and websites offering analogous solutions. Our approach involved utilizing UML Use Cases, Class Diagrams, and Architectural Designs to model the system requirements.

To achieve the successful implementation of the third project objective involving coding, system implementation, and testing, a structured approach was employed. This entailed translating the system's architectural design into executable code while adhering to coding standards and rigorous testing practices. The comprehensive testing procedures encompassed unit tests, integration tests, and system tests to ensure both functional and non-functional aspects were thoroughly evaluated, addressing usability, reliability, and scalability. This methodology, comprising planning, coding, integration, and validation phases, facilitates continuous improvement and eventual project closure.

To achieve the fourth objective, we have developed specific test cases based on our chosen validation methods. These tests are being executed, and we're analyzing the results to identify any discrepancies. Any issues we find are being addressed, and we'll continue to retest until we're confident that all requirements are met.

Finally, To achieve the fifth objective of our project (To list the lessons learnt from working as a team on this project.), we Holding team meetings specifically to look at the project and discuss what was done correctly, in addition to team members writing down their personal thoughts, lessons learned, and challenges they faced and overcame and Highlight moments when the team performed well and dealt effectively with challenges. Identify and document these successes.

1.5 Project Timeline

Figure 1-1 below shows the project timeline and activities in detail divided by weeks for the 2nd Semester of the academic year 1445 (2024).

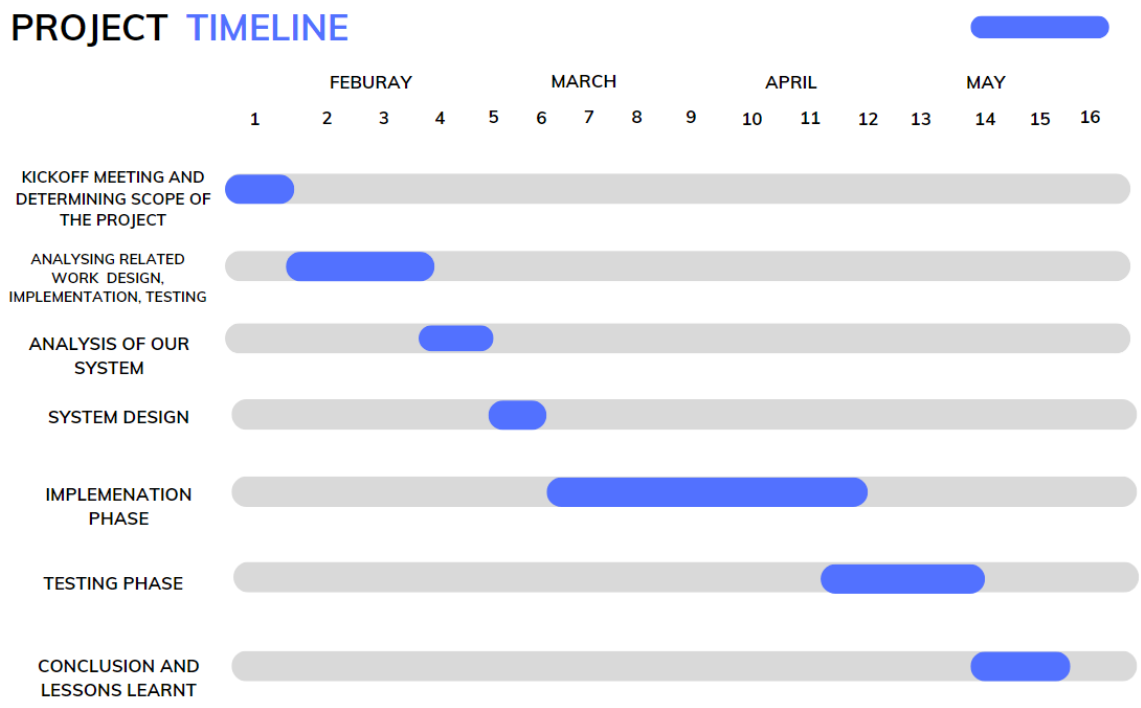


Figure 1-1 Project Timeline v1.

1.6 Document Organization

Chapter 1 In this chapter, we discussed the plan followed, the definition of the problem, the solution to the problem, the approach followed, and the objectives to be achieved.

Chapter 2 In this chapter, the stage of literary survey, reading, and review of systems similar to our project takes place, in addition to making a comparison table.

Chapter 3 In this chapter, we delve into system analysis, covering developmental methodology, requirement definition, specification, and elicitation, functional, non-functional, and domain requirements. Additionally,

Chapter 4 in this chapter, we will explore the design of our proposed system, including architectural design, object-oriented design incorporating structural static and dynamic models, data modeling, and user interface design.

Chapter 5 Show the software tools and programming languages used, including how the final project was developed and tested. Explain the process from designing to implementing the system. Discuss most important codes and analyze the results of system testing in detail.

Chapter 6 provides a conclusion about the project, A presentation of the lessons learned from the work of this project and future work.

1.7 Summary

In this chapter, we get a general idea of the problems people face from deliberate attacks on their emails, URLs, and personal contacts, and we will build a machine learning-based model to provide detection, which can contribute to reducing phishing attacks and information theft and increasing security. We presented the discussants' comments on the first phase of the project. Later in the second chapter, we will talk about related work and similar solutions, and we will talk about them in detail, with a focus on (design, implementation, testing), and everything related to them will be collected.

Chapter 2. Review of Related Work

2.1 Introduction

This chapter has two main parts. In the first part, we start by giving a general overview of the cybersecurity field. We introduce Social Engineering, which is a common topic, and discuss two significant problems: Phishing attacks and pretexting. We also cover incident response. In the second part, we look at how we can use AI (Artificial Intelligence) to solve these problems, specifically by using Machine Learning (ML). We explain the connection between ML and cybersecurity and explore some ML technologies that can be helpful in our field of study.

The second section can be split into two parts. The first part discusses projects that use AI in Machine Learning, specifically focusing on detecting phishing attacks in URLs and emails. The second part delves into SDLC (Software Development Life Cycle), followed by SDLC for ML. To conclude this chapter, we provide a comparison between these relevant projects and our own project hence we successfully did the first objective of the project.

2.2 Research Methodology

In order to obtain data, we use Google Scholar and IEEE to find a range of scientific literature sources, including articles, conference papers, research papers, and e-books. These sources can assist us in understanding related studies and their advantages, disadvantages, and differences. We make use of Google Scholars and the IEEE for citation and reference.

1. IEEE: The Institute of Electrical and Electronics Engineers is the largest technical professional association in the world. It actively participates in conferences, critical local and international discussions about today's most pertinent technological subjects, and research and authorship.

Annually, IEEE organizes more than 1,600 conferences and events globally and produces approximately one-third of all technical publications in the fields of electronics, computer science, and electrical engineering [1]. A list of pertinent software systems is provided in the next section. The IEEE official website is considered a reliable source for scientific research. The report on the website used IEEE style for all related references because IEEE style is mainly used in computer science and electronic and electrical engineering.

2-Google Scholar: “Google Scholar enables you to search specifically for scholarly literature, including peer reviewed papers, theses, books, preprints, abstracts and technical reports from all broad areas of research. Use Google Scholar to find articles from a wide variety

of academic publishers, professional societies, preprint repositories and universities, as well as scholarly articles available across the web.” [2]

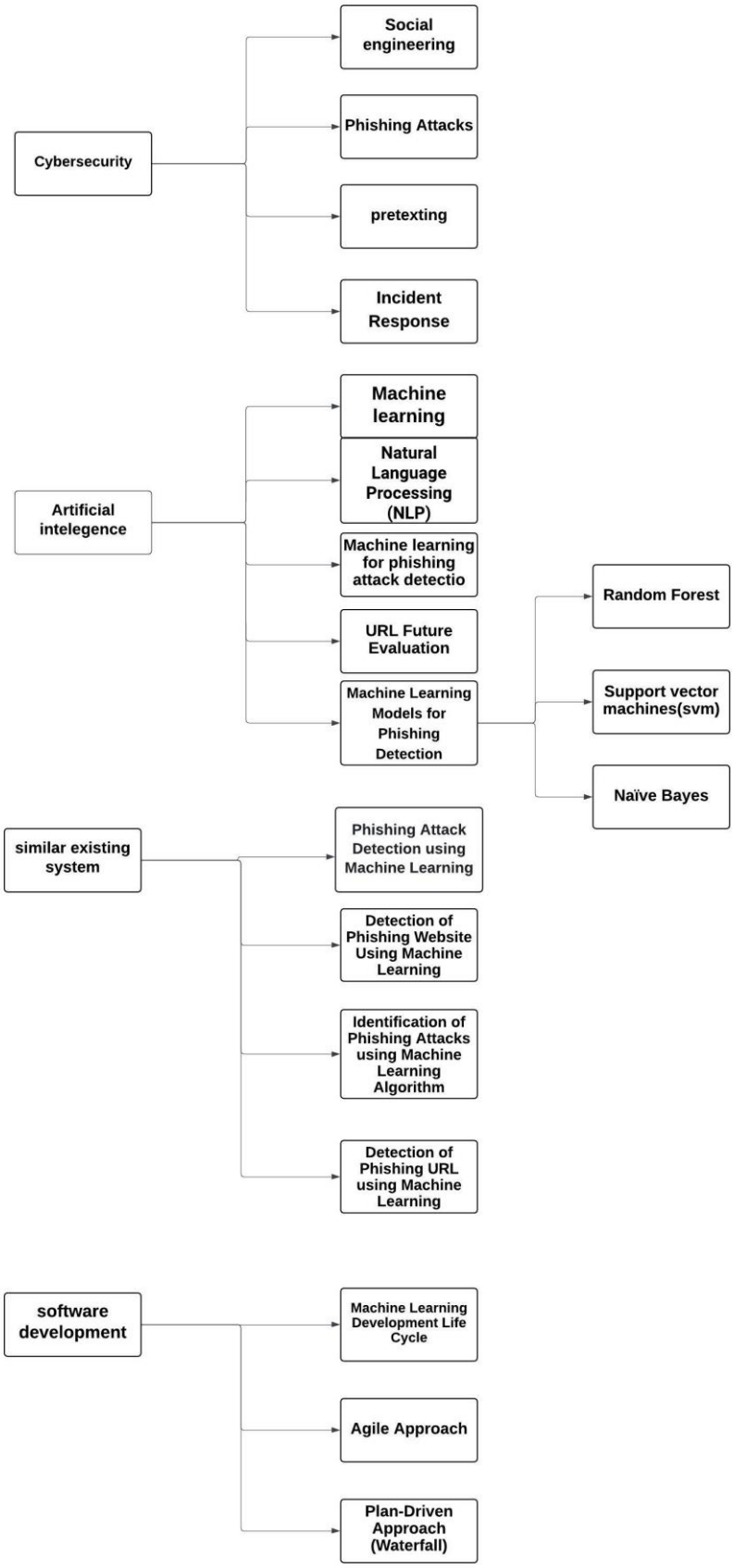


Figure 2-1 Schema of reviewed related topics.

2.3 Cybersecurity

Before we start explaining what Phishing is, we must first talk about Cybersecurity, In the digital age, cybersecurity emerges as an imperative facet of modern society, predicated upon the premise of safeguarding computing systems, networks, and sensitive information from a multitude of potential threats and vulnerabilities. This multifaceted discipline encompasses an array of technologies, methodologies, and practices, collectively orchestrated to preserve the sanctity and resilience of the virtual realm.

At its core, cybersecurity revolves around the pivotal triad of confidentiality, integrity, and availability. Confidentiality mandates that sensitive data remains accessible solely to duly authorized entities, precluding unauthorized intrusion. Integrity obligates the preservation of data accuracy and reliability by shielding against unwarranted alterations. Availability dictates that digital assets and services remain accessible and uninterrupted, impervious to downtime, ensuring seamless functionality. [3]

Figure 2-2 the 3 pillars of security CIA [3]

The cyber defense landscape utilizes a complex array of methods to meet these objectives. Authentication mechanisms validate the authenticity of users and systems, guaranteeing that entry is permitted exclusively to legitimate parties. Authorization protocols outline access rights, limiting activities in line with user roles. Risk management approaches, built on thorough

assessments of threats and vulnerabilities, work to recognize, assess, and alleviate potential dangers. [4]

2.3.1 Social Engineering

Social engineering is a tactic used by cybercriminals and malicious individuals that involves psychological manipulation to take advantage of the human aspect of security. This approach leverages people's inclination to trust and assist others, making it a significant threat in cybersecurity. Social engineers employ diverse strategies to trick, influence, and obtain confidential information from individuals who are not aware of their intentions.

Key Characteristics of Social Engineering:

Trust Exploitation: Social engineers often build trust with their targets, either by impersonating trusted entities or by leveraging personal information gleaned from sources like social media.

Manipulation Techniques: Psychological manipulation tactics, such as fear, urgency, curiosity, and authority, are commonly used to influence victims' behavior.

Target Diversity: Social engineering can target anyone, from employees within organizations to individuals in their personal lives. Attackers adapt their strategies to exploit the specific vulnerabilities of their targets.

Social engineering encompasses several common techniques:

Phishing: This involves the use of deceitful emails or messages to mislead recipients into interacting with harmful links, downloading malicious software, or disclosing confidential data**.**

Pretexting: This technique entails fabricating a false scenario or pretext to extract information from a target, often involving impersonation and manipulation.

Baiting: Baiting consists of offering enticing digital or physical "bait" (such as free software or USB drives) infected with malware to compromise unsuspecting victims.

Tailgating: This tactic involves gaining unauthorized physical access to a restricted area by either following an authorized person closely or impersonating an authorized individual.

Real-World Example: You get a pressing email that seems to be from a popular online store, stating that they have special, time-limited discounts on your preferred items. The email includes a link that guides you to a webpage where you can access these exclusive offers [5].

2.3.2 Phishing Attacks

Phishing is a cyber-attack where an attacker, often pretending to be a trusted source, tries to trick people or groups into giving away sensitive information like passwords, financial data, or personal identification details. This usually happens through deceitful emails, messages, or websites that imitate real, trustworthy sources. The aim of phishing is to make the target believe they are dealing with a reliable entity, causing them to share confidential information that can be misused, like unauthorized access or identity theft [6].

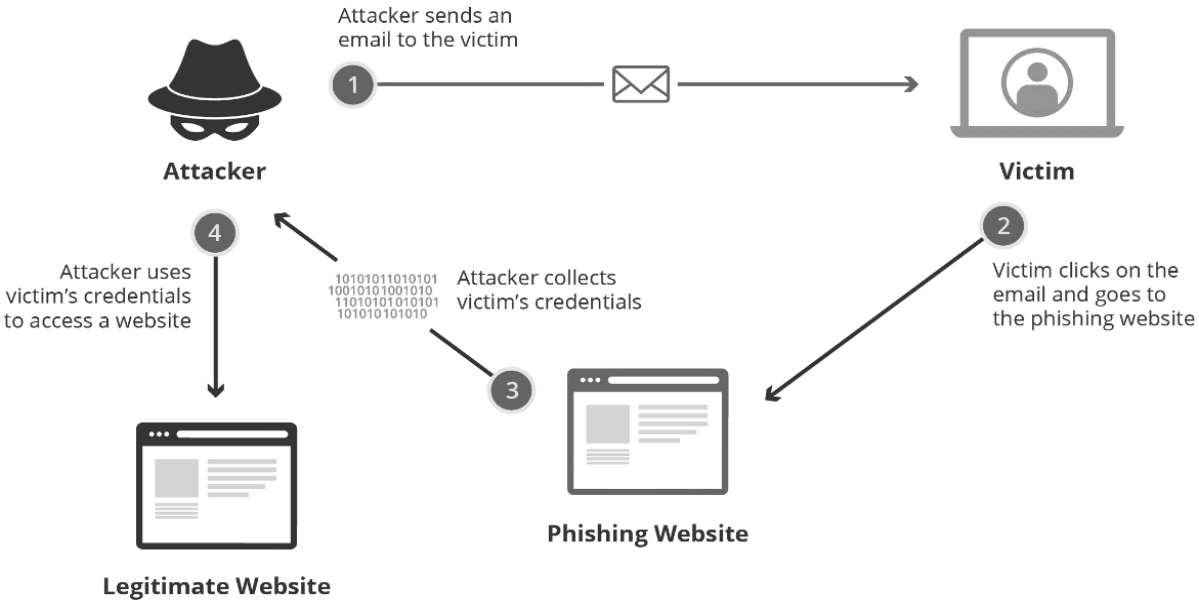


Figure 2-3 Phishing attack through email scenario [6]

Phishing tricks are when dishonest people pretend to be trustworthy to get important info. As tech improves, these tricks get fancier. It's vital to know the common types:

Email Phishing: Most common. It tricks with fake links in emails to get personal info. They might act like a big company or even a coworker.

Malware Phishing: Tricky emails hide harmful software as safe stuff like resumes. Opening them can freeze your computer.

Spear Phishing: Sneakier. Attackers research specific people to send personalized emails. Harder to catch with basic safety.

Whaling: Targets big shots like CEOs. Scammers learn a lot about them to steal valuable info.

Smishing: Mixes text messages with tricks. Scammers seem like trusted sources. People trust texts more, as they're personal and plain.

Vishing: Fake call centers try to get sensitive info over the phone. They might use tricks to convince victims to install disguised harmful apps. [7]

Key Attributes of Phishing Attacks:

1. **Impersonation**: Attackers commonly pretend to be trusted sources like banks, social media sites, or acquaintances in order to earn the trust of their targets.
2. **Misleading Content**: Phishing messages frequently feature urgent or tempting content, with the intention of provoking immediate responses from recipients.
3. **Harmful Links or Attachments**: Phishing emails include links to counterfeit websites or malicious attachments meant to compromise the security of the victims' systems.

- **Strategies to Counter Phishing:** Educating Users: Providing training to help individuals identify phishing attempts and promote safe online behavior.
- Email Filtering: Deploying email filtering and spam detection systems to pinpoint and prevent phishing emails.
- Implementing Two-Factor Authentication (2FA): Enforcing 2FA to enhance security, even in cases where login credentials are compromised. [8]

2.3.3 Pretexting

Pretexting is a form of social engineering that involves creating a fabricated scenario to manipulate individuals into revealing sensitive information or performing actions against their best interests.

Important Aspects of Pretexting:

Made-Up Stories: Pretexters create complex narratives to trigger sympathy, curiosity, or cooperation from their targets.

Assumed Identities: They frequently take on false personas, like pretending to be a colleague, client, or service provider, to build trust.

Research: Pretexters may do thorough investigations to collect personal information about their targets, enhancing the credibility of their fabricated stories.

Real-World Example: An attacker might pretext as a customer service representative from a bank, claiming to have detected fraudulent activity on the target's account. They request the target's account number and PIN to "verify their identity," ultimately aiming to steal funds from the account [5].

2.3.4 Incident Response

Incident response is a structured method for handling and reducing the impact of security incidents, breaches, or cyberattacks to limit harm, safeguard confidential data, and quickly

return to normal operations. It's a fundamental part of a strong cybersecurity plan, designed to efficiently deal with unexpected threats, in the case of our project incident response will play a big role which is alerting the user of a potential threat message.

Key Components of Incident Response:

Preparation: preparation involves setting up the system and ensuring it's ready to identify and respond to phishing threats. This includes configuring email filters, setting up monitoring tools, and defining the roles and responsibilities of team members responsible for managing the system.

Detection and Identification: It focuses on the continuous monitoring of email communications and network traffic for signs of phishing attempts. The system uses anomaly detection and intrusion detection techniques to promptly identify potential phishing incidents.

Containment: the detection system initiates containment measures. It might isolate affected user accounts, block access to malicious URLs, and prevent the spread of the attack by filtering out phishing emails.

Eradication: the phishing detection system can assist in eradicating the root cause of the attack. This often involves removing malware introduced through the phishing incident, patching vulnerabilities in email systems, and implementing security updates to prevent similar attacks in the future.

Recovery: The recovery phase is about ensuring that the affected systems and services return to normal operation. The system helps in data restoration, system reconfiguration, and enhancing security measures to prevent future phishing attempts.

Lessons Learned: After the incident is resolved, it's essential to conduct a thorough post-incident analysis to understand what happened, why it happened, and how to prevent similar incidents in the future.

Incident Response Team: It typically includes experts in cybersecurity, forensics, legal, and communication who collaborate closely. Cybersecurity experts are responsible for managing and configuring the detection system. Forensic analysts gather evidence for legal actions. Legal advisors ensure compliance with data protection laws, and communication specialists manage messaging to stakeholders [9].

2.4 Artificial intelligence

Artificial Intelligence (AI) involves creating machines that mimic human intelligence, enabling them to think, learn, and perform tasks like humans. This field encompasses a diverse set of technologies and methods that empower computers to handle tasks that usually demand human-like intelligence, such as problem-solving, language comprehension, and decision-making.

Figure 2-4 displays definitions of artificial intelligence from eight different textbooks. There are two major differences between these definitions. In general, the ones at the top deal with thinking and mental processes, while the ones at the bottom deal with behavior. While the definitions on the right measure against an idealized version of intelligence that we will refer to as rationality, the definitions on the left assess success in terms of faithfulness to human performance. If a system acts in accordance with what it knows and performs the "right thing," it is rational.

Systems that think like humans	**Systems that think rationally**
"The exciting new effort to make computers think . . . *machines with minds*, in the full and literal sense." (Haugeland, 1985)	"The study of mental faculties through the use of computational models." (Charniak and McDermott, 1985)
"[The automation of] activities that we associate with human thinking, activities such as decision-making, problem solving, learning . . ." (Bellman, 1978)	"The study of the computations that make it possible to perceive, reason, and act." (Winston, 1992)
Systems that act like humans	**Systems that act rationally**
"The art of creating machines that perform functions that require intelligence when performed by people." (Kurzweil, 1990)	"Computational Intelligence is the study of the design of intelligent agents." (Poole *et al.*, 1998)
"The study of how to make computers do things at which, at the moment, people are better." (Rich and Knight, 1991)	"AI . . . is concerned with intelligent behavior in artifacts." (Nilsson, 1998)

Figure 2-4 Some definitions of artificial intelligence, organized into four categories[10]

Throughout history, people have explored all four methods of AI. As expected, there's a clash between human-focused and logic-focused approaches. The human-centered way relies on experiments and testing ideas, while the rational approach uses math and engineering. These groups have criticized and aided each other. [10]

	Computer	Human Brain
Computational units	1 CPU, 10^8 gates	10^{11} neurons
Storage units	10^{10} bits RAM	10^{11} neurons
	10^{11} bits disk	10^{14} synapses
Cycle time	10^{-9} sec	10^{-3} sec
Bandwidth	10^{10} bits/sec	10^{14} bits/sec
Memory updates/sec	10^9	10^{14}

Figure 2-5 A crude comparison of the raw computational resources available to computers[10].

AI branches include the following main fields:

Machine Learning (ML): ML is a subfield of AI that involves the development of algorithms and models that enable computers to learn from data and make predictions or decisions without being explicitly programmed.

Natural Language Processing (NLP): NLP is concerned with the interaction between computers and human language. It encompasses tasks such as language understanding, translation, sentiment analysis, and chatbot development.

Expert Systems (ES): Expert systems are AI programs that mimic the decision-making abilities of a human expert in a specific domain. They use knowledge-based rules to make decisions and provide recommendations.

2.4.1 Machine Learning

Machine learning, a subset of artificial intelligence and computer science, centers on using data and algorithms to mimic human learning, getting better over time. It plays a crucial role in the expanding domain of data science. By employing statistical techniques, algorithms are taught to make classifications or predictions, and to reveal important findings in data mining endeavors. These discoveries then guide decision-making in applications and businesses, ideally impacting key growth metrics [11].

Machine learning models fall into three primary categories:

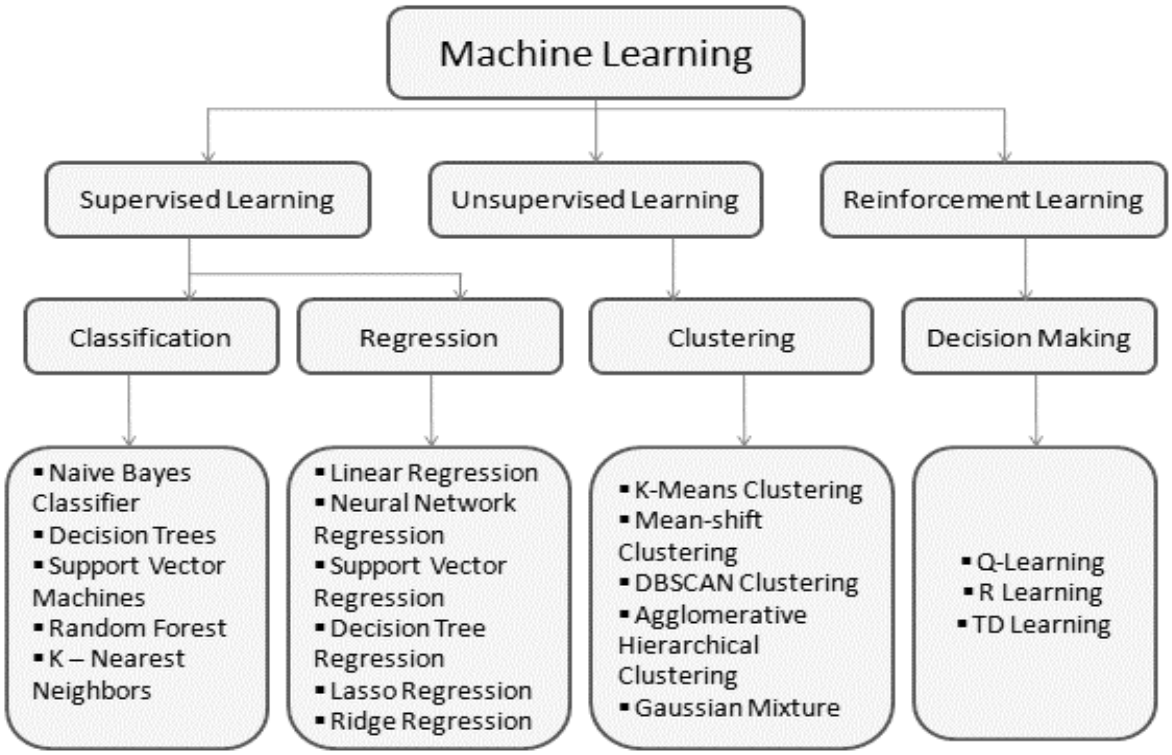


Figure 2-6 Machine Learning Models [11]

Supervised machine learning, or supervised learning, is characterized by the use of labeled datasets to train algorithms for precise outcome prediction or data classification. The model modifies its weights when input data is entered until a satisfactory fit is achieved. This happens during the cross-validation phase, which makes sure the model doesn't overfit or underfit. Detecting phishing attempts and putting spam in a different folder from your inbox are just two examples of the many real-world issues that supervised learning assists businesses in solving at scale. Neural networks, naïve bayes, logistic regression, random forest, linear regression, and support vector machines (SVM) are a few techniques used in supervised learning.

Unsupervised learning, also called unsupervised machine learning, employs machine learning algorithms to examine and group datasets that lack labels. These algorithms identify underlying patterns or groupings in the data without requiring human guidance. This method's capacity to uncover similarities and differences in information makes it valuable for tasks like exploring data, devising cross-selling strategies, segmenting customers, and recognizing images and patterns. It's also employed to streamline models by reducing the number of features, a process known as dimensionality reduction. Principal component analysis (PCA) and singular value decomposition (SVD) are two common techniques for this purpose. Other

algorithms used in unsupervised learning encompass neural networks, k-means clustering, and probabilistic clustering methods.

Semi-Supervised machine learning, or supervised learning, is characterized by the use of labeled datasets to train algorithms for precise outcome prediction or data classification. The model modifies its weights when input data is entered until a satisfactory fit is achieved. This happens during the cross-validation phase, which makes sure the model doesn't overfit or underfit. Detecting phishing attempts and putting spam in another spot from your inbox are just two examples of the many real-world issues that supervised learning assists businesses in solving at scale. Neural networks, naïve bayes, logistic regression, random forest, linear regression, and support vector machines (SVM) are a few techniques used in supervised learning. Many real-world issues that supervised learning assists businesses in solving at scale. Neural networks, naïve bayes, logistic regression, random forest, linear regression, and support vector machines (SVM) are a few techniques used in supervised learning [11].

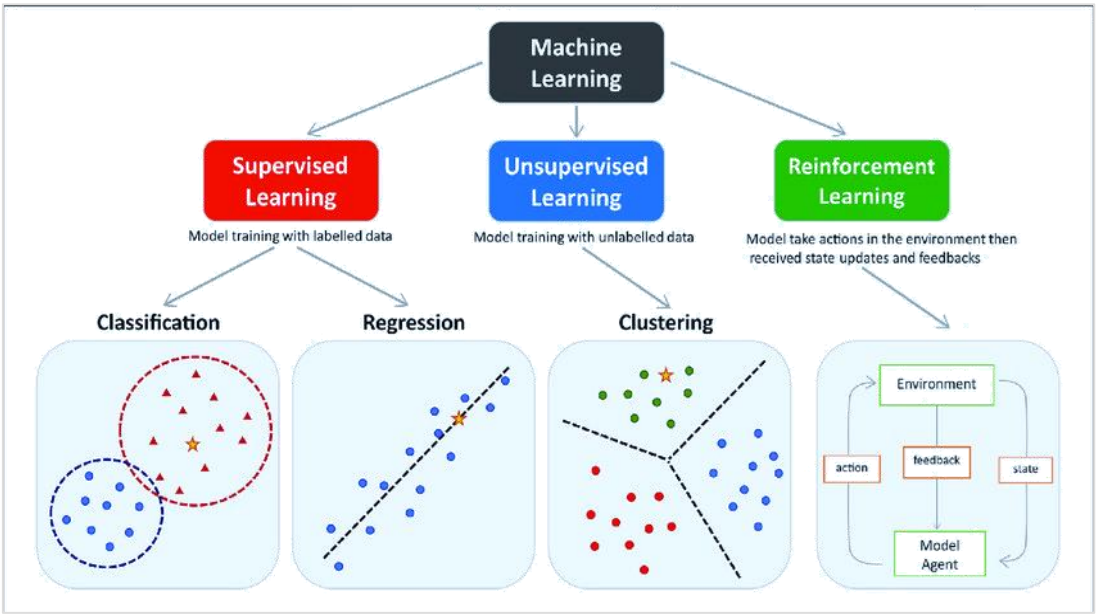


Figure 2-7 Machine Learning Models [11]

2.4.2 Natural Language Processing

NLP stands for Natural language Processing. It is defined as a field of AI that helps computer to communicate with humans. Because of NLP, it becomes possible for the computers to read, hear, edit and interpret text, speech and determine which parts are important. Basic NLP tasks include: removing stop words, punctuations, special characters, tokenization, stemming, tagging, language detection and identification of semantic relationships. It is also explained as the means of handling the natural language by automatic means using a software.

Normalizing (Text Analysis) text means converting it to a more convenient, standard form. Tokenization - Splitting a phrase, sentence, paragraph, or an entire text document into smaller units, such as individual words or terms.

Lemmatization - The task of determining that two words have the same root, despite their surface differences. – The words “sang”, “sung”, and “sings” are forms of the verb “sing”. The word sing is the common lemma of these words, and a lemmatize maps from all of these to “sing”.

Stemming - We mainly just strip suffixes from the end of the word. – The words “caring”, “careful” are stemmed to “car”, and the words “history” and “historical” are stemmed to “history”.

Sentence Segmentation - We break up a text into individual sentences, using cues like periods or exclamation points.

Part-of-Speech (POS) Tagging: Assigning grammatical categories (noun, verb, etc.) to words in a sentence.

Text Understanding:

Syntax Analysis: Parsing the grammatical structure of sentences to understand the relationships between words and phrases.

Semantics Analysis: Understanding the meaning and context of words, phrases, and sentences. This often involves disambiguating word meanings based on the surrounding context.

Feature Engineering for Natural Language Processing:

• Machine Learning algorithms learn from a pre-defined set of features from the training data to produce output for the test data.

• Machine learning algorithms cannot work on the raw text directly

• Need some feature extraction techniques to convert text into a matrix(or vector) of features.

• Some of the most popular methods of feature extraction are : – Bag of Words – TF-IDF

TF-IDF (Features Extraction Method):

TF-IDF (Term Frequency-Inverse Document Frequency) is a technique that helps identify the relevance of terms in a document within a larger context. Term Frequency (TF): Term Frequency measures how often a term appears in a document. It is calculated by counting the number of times a term occurs in a document. The idea is that words that appear more frequently

in a document are likely to be more important in representing the content of that document. TF(t, d) = (Number of times term t appears in document d) / (Total number of terms in document d)

Inverse Document Frequency (IDF): Inverse Document Frequency measures the importance of a term within the entire corpus. It quantifies how unique or rare a term is across the collection of documents. Terms that are common across many documents receive a lower IDF value, while terms that are rare receive a higher IDF value. The IDF of a term is calculated as: IDF(t, D) = log_e(Total number of documents in the corpus / Number of documents containing term t)

TF-IDF Score: The TF-IDF score for a term in a document is calculated by multiplying the term frequency (TF) and the inverse document frequency (IDF): TF-IDF(t, d, D) = TF(t, d) * IDF(t, D) The resulting TF-IDF score reflects the importance of a term within a specific document relative to its importance in the entire corpus. High TF-IDF scores indicate that a term is important within a document but relatively rare in the corpus, while low scores suggest that the term is either common or less relevant to the document [12].

Morphology: In Natural Language Processing (NLP), morphology refers to the study of the internal structure of words and the rules governing the formation of words in a language. It involves analyzing the smallest units of meaning within words, known as morphemes. Morphemes can be further classified into roots, prefixes, and suffixes, each contributing to the overall meaning of a word. Understanding morphology is crucial in NLP for tasks such as part-of-speech tagging, stemming, and lemmatization.

Syntax: In Natural Language Processing (NLP), syntax refers to the study of the structure, arrangement, and relationships of words within sentences to form grammatically correct and meaningful language expressions. Syntactic analysis is crucial for understanding the grammatical rules and hierarchical structures that govern the composition of sentences.

Semantic: In Natural Language Processing (NLP), semantics involves the study of meaning in language. It goes beyond the syntactic structure of sentences and focuses on understanding the meanings of words, phrases, and entire texts. Semantic analysis is crucial for extracting the intended meaning from language data, and it plays a central role in several NLP tasks.

Information Extraction: Information extraction (IE) in Natural Language Processing (NLP) refers to the automatic extraction of structured information from unstructured text. The goal is

to identify and capture specific types of information, such as entities, relationships, and events, from large volumes of textual data.

2.4.3 Machine Learning for Phishing Attack Detection

Machine Learning techniques are popular for detecting phishing websites and emails, and it simplifies the classification process. To train a machine learning model for a learning-based detection system, the available data must include phishing and legitimate website classes. To detect a phishing attempt, various classifiers are utilized. Previous research has shown that when robust Machine Learning algorithms are utilized, detection accuracy is high. To decrease features, several feature selection strategies are utilized. Figure 2-8 depicts how the machine learning model works. A set of input data is fed into the machine learning model to train it to predict phishing attacks or legitimate traffic [13].

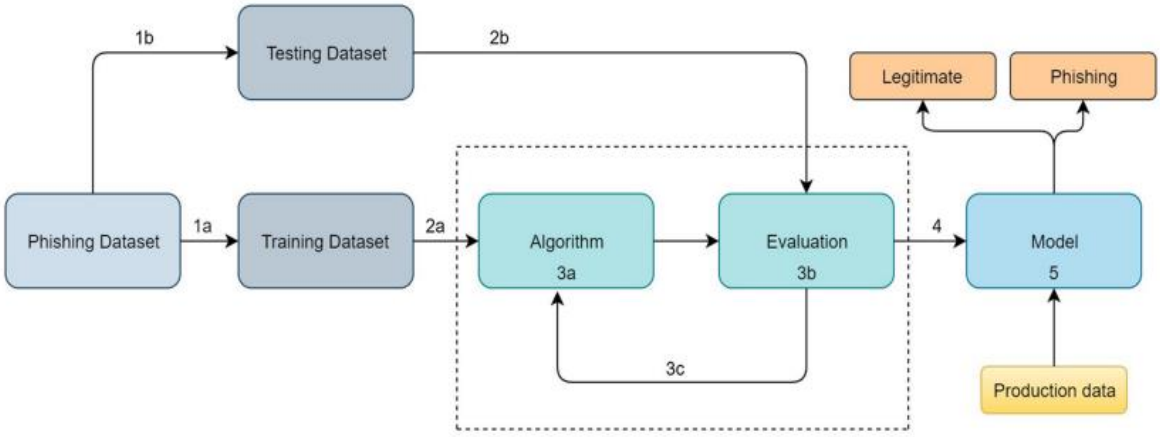


Figure 2-8 Machine learning for phishing attack detection [13].

Machine learning methods are commonly employed to uncover hidden patterns in datasets. The most commonly used algorithms are K-nearest neighbor, decision trees, random forest, and support vector machine [14].

2.4.4 URL Future Evaluation

URL Future Evaluation makes use of URL properties to assess the legitimacy of websites. A web address, often known as a URL, is made up of the following elements: path, file name, domain name, and protocol. Attackers typically attempt to mimic authentic websites by manipulating one or more characters in the web address in order to trick the victims.

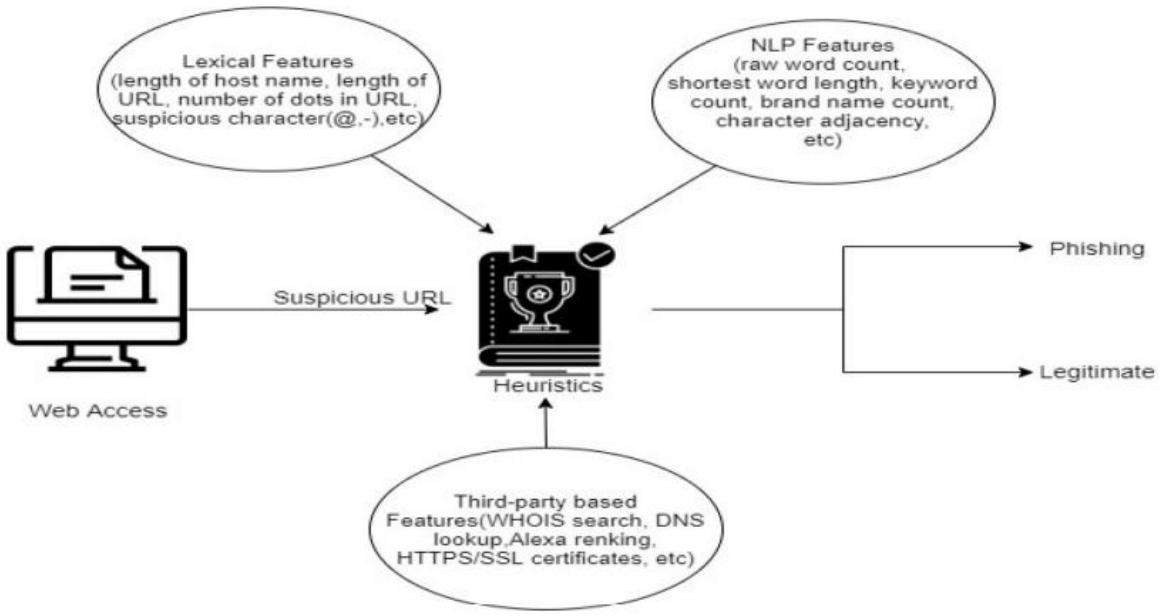


Figure 2-9 Heuristic-based web phishing detection [15]

The process of the heuristic rule-based web address evaluation approach is shown in Figure 2-9 Heuristic criteria are applied to the URL based on the conventions of the standard web address when a suspicious website appears, helping to establish the website's legitimacy. In order to identify URL-based web phishing attempts, Sahingoz et al. [15] used heuristics to extract natural language processing (NLP) properties from the URL. Based on factors like raw word count, short word length, Alexa rating, similar brand name count, etc., the heuristics are developed. Yukun Li et al. [16] used a number of heuristics on the URL to check for anomalies, including https, the number of dots in a domain name, top-level domains, suspicious symbols (e.g. @, _), and URL length information, number of dots in a domain name, sensitive vocabulary, and top level domain, Fourteen heuristics were computed by Jeeva and Rajsingh [17]: the host URL's length, the number of slashes in the URL, the number of dots in the host name of the URL, the number of terms in the host name of the URL, special characters, IP address, Unicode in the URL, transport layer security, subdomain, a specific keyword in the URL, top-level domain, number of dots in the path of the URL, the hyphen in the host name of the URL, and URL length respectively. Algorithms for associative rule mining are then given the extracted features. When a user accesses a website, Varshney et al. [18] suggested a simple phish detector that retrieves the webpage title and the domain name of the URL. A search is conducted using a title page and the retrieved URL domain name and the title page are searched using a search engine to determine the legitimacy [19].

Table 2-1 List of Feature's in Lexical Feature Group [19]

Feature	Data Type	Description
NumDots	Numeric	The number of dots In the URL.
SubdomainLevel	Numeric	Determines the number of subdomain levels.
PathLevel	Numeric	Determining the level of the path in the URL.
UrlLength	Numeric	Length of each URL used in the dataset. The length contains the number of letters or symbols used to create the URL.
NumDash	Numeric	Total number of dash in a URL.
NumDashInHostname	Numeric	The number of dashes in a hostname
AtSymbol	Boolean	Total number of '@' symbol in the URL.
TildeSymbol	Boolean	Total number of tilde '~' symbol in the URL.
NumUnderscore	Numeric	Number of underscores'_' used in the URL.
NumPercent	Numeric	Total number of percent symbol present in the URL.
NumQueryComponents	Numeric	Total number of query components.
NumAmpersand	Numeric	Total number of '&' character.
NumHash	Numeric	Total number of '#' character.
NumNumericChars	Numeric	The total number of numeric characters.
NoHttps	Boolean	Check if there is a HTTPS in the URL.
RandomString	String	Set of Characters that are random.
IPAddress	Boolean	Check if the hostname of the URL uses the IP address.
DomainsInSubDomains	Boolean	Determines if TLD or CCTLD is in the subdomain of URL.
DomainsInPaths	Boolean	Determines if the website link has used TLD or CCTLD.
HttpsInHostname	Boolean	Determines if HTTPS is disorderly in the hostname of the URL.
HostnameLength	Numeric	Length of hostname which includes all the characters and symbols.
PathLength	Numeric	Length of all paths in each URL.
QueryLength	Numeric	Length of query in the URL.
DoubleSlashInPath	Boolean	Checks if there is a double slash in the path.
NumSensitiveWords	Numeric	Checks if there are any sensitive words like secure, sign in, login, etc.
EmbeddedBrandName	Boolean	Checks if there is the name of a brand in the domain.
PctExtHyperLinks	Float	Checks the percentage of external hyperlinks in the source code.

2.4.5 Random Forests

Random Forests is an Ensemble approach for classification and regression. Random Forest classifier constructs number of decision trees during the training time and outputs a class that is the mode of the classification classes of the individual trees. Random Forest classification performs better than any other decision tree algorithms as it uses a forest of classification trees to take a decision . In Random Forests, to classify a new object from an input vector, we give the input vector each of the trees in the forest, each tree gives a classification, and we say that the tree 'votes' for that class. The forest chooses that classification which has more votes over all the trees in the forest. The training algorithm for Random forests applies the general technique called Bagging. Given a training set of N size, X=x1,x2,.....xn with responses Y=y1,y2,.....y_n, Bagging selects a random sample from the training set with replacement and try fitting trees to these samples. If there are V variables in the input vector, Random Forest uses a modified tree learning algorithm that selects a random subset of features at each candidate split in the learning process. This random selection of features sometimes referred as "feature bagging". Typically, if a dataset is having R features, $\sqrt{V}$ features are used at each split [20].

2.4.6 Support Vector Machines

Support vector machines (SVMs) are one of the most popular classifiers. The idea behind SVM is to get the closest point between two classes by using the maximum distance between classes. This technique is a supervised learning model used for linear and nonlinear classification. Nonlinear classification is performed using a kernel function to map the input to a higher-dimensional feature space. Although SVMs are very powerful and are commonly used in classification, it has some weakness. They need high calculations to train data. Also, they are sensitive to noisy data and are therefore prone to overfitting [20].

2.4.7 Naïve Bayes

One of the simplest yet powerful classifier algorithms, Naive Bayes is based on Bayes' Theorem Formula with an assumption of independence among predictors. Given a Hypothesis A and evidence B, Bayes' Theorem calculator states that the relationship between the probability of Hypothesis before getting the evidence P(A) and the probability of the hypothesis after getting the evidence P(A|B) is:

Here:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

- A, B = events
- P(A|B) = probability of A given B is true
- P(B|A) = probability of B given A is true
- P(A), P(B) = the independent probabilities of A and B

2.5 Similar Existing Systems

The related literature contains several similar systems to our proposed system. One of the recent works is the one conducted by [21], which fortifies phishing defense using cutting-edge models, trained on a diverse dataset of 737,000 URLs.

- Required libraries: scikit-learn, pandas, matplotlib.
- Execute cells in a Jupyter Notebook environment.
- The uploaded code has been executed and tested successfully within the Google Colab environment.

Binary-class classification problem Task is to classify the likelihood of a URL: Phishing , Benign. Key Tasks Undertaken**:**

Data Concatenation: Concatenated multiple Data Frames vertically into a single Data Frame

Concatenated multiple Data Frames vertically into a single Data Frame. PhishStorm-URL dataset: 96011 Data Size. ISCX-URL2016 dataset: Extracted only Phishing / Legitimate from165366 rows. Malicious URL dataset: 651,191 Data Size

Data Loading : This code reads a CSV file located at the specified path (data_path) into a DataFrame named Dataset using pandas. The parameter error_bad_lines=False tells pandas to skip lines that contain errors. Finally, it displays the contents of the Dataset.

```
data_path = r"E:\data\data_Features (2).csv"
Dataset = pd.read_csv(data_path, error_bad_lines=False)
Dataset
```

	label	malicious_probability	domain_length	ip	url_length	redirection	digits	letters	path_count	ipv	...	url_count_equal
0	1	0.72	0	0	16	0	0	13	0	0	...	0
1	0	0.97	5	0	35	0	1	29	2	0	...	0
2	0	0.98	7	0	31	0	1	25	3	0	...	0
3	1	0.72	9	0	88	0	7	63	1	0	...	4
4	1	1.00	9	0	235	0	22	199	1	0	...	3
...	...	...	...	...	...	...	...	...	...	...	...	...
737027	0	0.84	7	0	39	0	12	21	3	0	...	0
737028	0	0.91	8	0	44	0	7	29	4	0	...	0
737029	0	0.79	7	0	42	0	3	33	4	0	...	0
737030	0	0.96	4	0	45	0	0	36	2	0	...	0
737031	0	0.73	4	0	41	0	0	36	3	0	...	0

737032 rows × 26 columns

Figure 2-10 Display dataset.

Feature Engineering and Data Cleaning:

Handling Null Values and Duplicate Rows, Handling Repeated Maximum Values, Splitting Data: Train 80% , Test 20%, Oversampling with SMOTE: Used SMOTE (Synthetic Minority Over-sampling Technique) to balance the training data, especially for the minority class.

Checking for Null values

```
print("Number of NULL values:\n", Dataset.isnull().sum())
```

Figure 2-11 Checking Null value.

Checking for Duplicate Rows

```
print("Number of Duplicate Rows:", Dataset.duplicated().sum())
```

```
Number of Duplicate Rows: 160058
```

Figure 2-12 Checking for Duplicate Rows

Check outliers.

```
numeric_columns = Dataset.select_dtypes(include=['number'])

# Extract column names
numeric_column_names = numeric_columns.columns.tolist()

# Print the names of numerical columns
print("Numerical Column Names:")
print(numeric_column_names)
```

```
Numerical Column Names:
['label', 'malicious_probability', 'domain_length', 'ip', 'url_length', 'redirection', 'digits', 'letters', 'path_count',
'ipv', 'short', 'url_count_dash', 'url_count_at', 'url_count_question', 'url_count_percent', 'url_count_dot', 'url_count_eq
ual', 'url_count_http', 'url_count_https', 'url_count_www', 'entropy', 'is_encoded', 'unusual_character_count', 'sus', 'rat
io', 'tokens']
```

```
# Create separate boxplots for each numerical column
plt.figure(figsize=(20, 15))

for i, column in enumerate(numeric_column_names):
    plt.subplot(6, 6, i + 1)  # Adjust the subplot grid as needed
    sns.boxplot(x=column, data=Dataset, palette="Set2")
    plt.title(f"Boxplot of {column}")

plt.tight_layout()
plt.show()
```

Figure 2-13 Check outliers.

Features Selection: Developed a function to assess and identify features with overwhelmingly repeated maximum values.

- Evaluated the percentage of occurrences for the most frequent value in each feature.
- Removed features where the maximum value was repeated over 90% of the time.
- Applied a 90% repetition threshold to exclude less informative or near-constant features.
- Improved model efficiency and computational performance by reducing redundancy in the dataset.

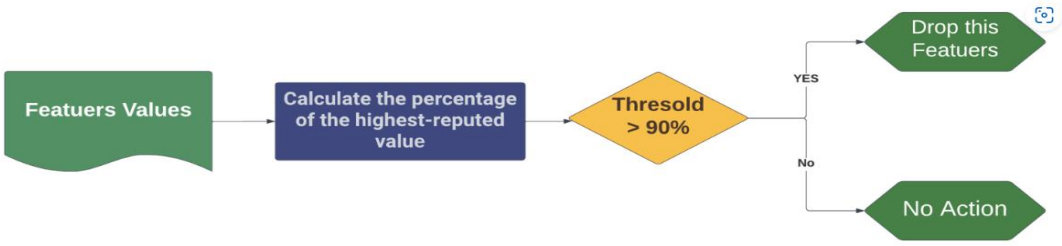


Figure 2-14 Features Selection

Modeling:

- Model Training: Trained various classification models (Logistic Regression, SVM, Decision Tree, Random Forest, XGBoost, etc.) using LazyClassifier.

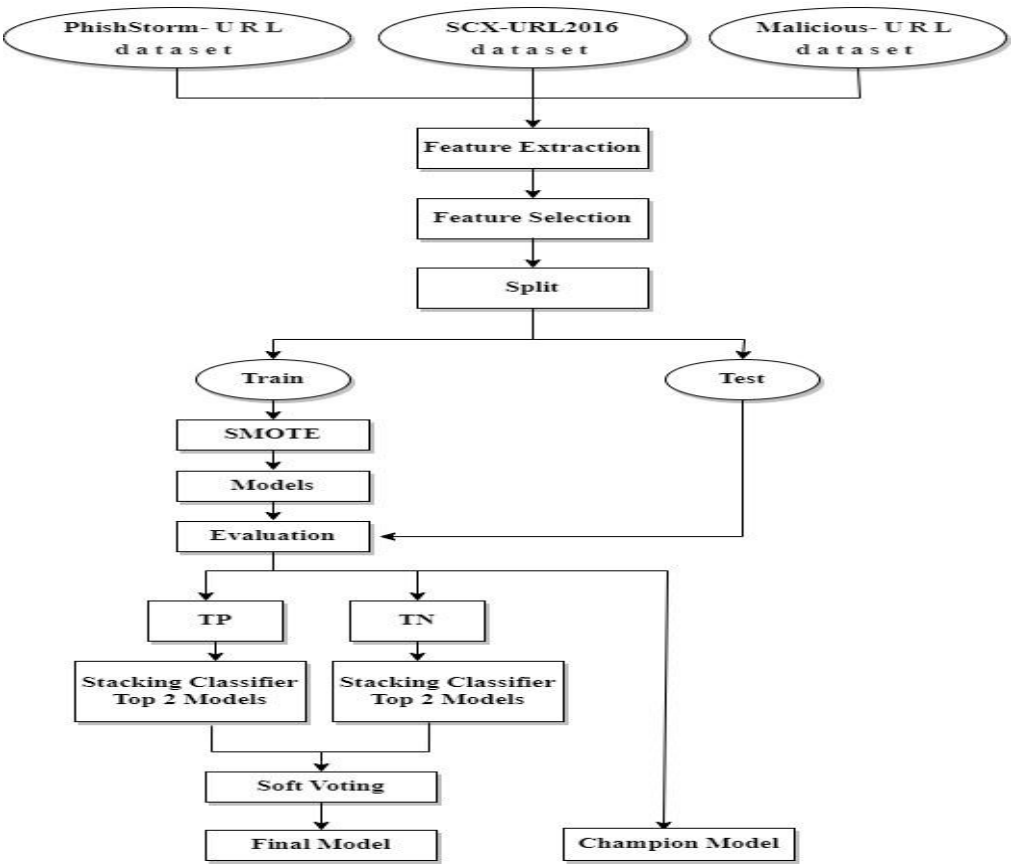


Figure 2-15 Modeling.

LazyClassifier :

LazyClassifier is a Python library that provides a simple and easy-to-use interface for fitting and evaluating multiple machine learning models with minimal code.

```
clf = LazyClassifier(predictions=True)
models, predictions = clf.fit(X_train_resampled, X_test, y_train_resampled, y_test)
```

```
models
```

Figure 2-16 this code to fit multiple models and generate predictions on test data for comparison.

Model	Accuracy	Balanced Accuracy	ROC AUC	F1 Score	Time Taken
LGBMClassifier	0.86	0.87	0.87	0.87	5.09
XGBClassifier	0.86	0.87	0.87	0.86	37.95
SVC	0.86	0.87	0.87	0.87	25362.20
AdaBoostClassifier	0.85	0.84	0.84	0.85	31.99
NuSVC	0.83	0.83	0.83	0.83	45207.52
KNeighborsClassifier	0.81	0.81	0.81	0.82	1609.17
LogisticRegression	0.82	0.80	0.80	0.82	2.41
CalibratedClassifierCV	0.82	0.80	0.80	0.82	769.51
LinearSVC	0.82	0.80	0.80	0.82	198.22
SGDClassifier	0.82	0.80	0.80	0.82	2.22
RidgeClassifier	0.82	0.79	0.79	0.82	1.02
RidgeClassifierCV	0.82	0.79	0.79	0.82	2.10
LinearDiscriminantAnalysis	0.82	0.79	0.79	0.82	1.60
GaussianNB	0.80	0.78	0.78	0.80	0.72
Perceptron	0.80	0.77	0.77	0.79	3.56
BaggingClassifier	0.77	0.76	0.76	0.78	52.59
RandomForestClassifier	0.77	0.76	0.76	0.77	208.51
ExtraTreesClassifier	0.77	0.76	0.76	0.77	87.60
QuadraticDiscriminantAnalysis	0.81	0.75	0.75	0.80	1.54

Figure 2-17 Comparisons between models.

Sort the Models Results in Descending Order Based on TP

```
# Display the results in descending order based on tp_test
sorted_results_tp = sorted(models_results.items(), key=lambda x: x[1]['TP'], reverse=True)

# Create a DataFrame from the results
df_tp = pd.DataFrame(sorted_results_tp, columns=['Model', 'Models Results in Descending Order based on TP'])

# If you want to add styling, you can use the following:
styled_table_tp = df_tp.style.background_gradient(cmap='Blues')
# display(styled_table_tp)
styled_table_tp
```

Figure 2-18 Models Results in Descending Order based on TP.

	Model	Models Results in Descending Order based on TP
0	Stochastic Gradient Descent SGD	{'TP': 51882, 'TN': 938, 'f1': 0.2712905523573978}
1	XGB Extreme X Gradient Boosting	{'TP': 45651, 'TN': 81582, 'f1': 0.8544955559867369}
2	LGBM	{'TP': 45605, 'TN': 81912, 'f1': 0.8563529939655842}
3	Catboost	{'TP': 45594, 'TN': 81441, 'f1': 0.8531108199786871}
4	Gradient Boosting	{'TP': 43836, 'TN': 82828, 'f1': 0.8486938005436107}
5	Bernoulli Naive Bayes	{'TP': 43018, 'TN': 71708, 'f1': 0.7695649307391925}
6	AdaBoost	{'TP': 42638, 'TN': 82203, 'f1': 0.8350287807100614}
7	K Nearest Neighbors	{'TP': 41349, 'TN': 76704, 'f1': 0.7887095427709013}
8	Logistic Regression	{'TP': 38567, 'TN': 82322, 'f1': 0.8027215614231051}
9	Random Forest	{'TP': 36563, 'TN': 77349, 'f1': 0.753933850804531}
10	Bagging Classifier	{'TP': 36534, 'TN': 77318, 'f1': 0.7534950737195822}
11	Extra Trees Classifier	{'TP': 36120, 'TN': 77795, 'f1': 0.7530536818207502}
12	Decision Tree Classifier	{'TP': 35121, 'TN': 77714, 'f1': 0.744100335933428}
13	Extra Tree Classifier	{'TP': 35011, 'TN': 77492, 'f1': 0.7417664472133702}
14	Gaussian Naive Bayes	{'TP': 26922, 'TN': 89277, 'f1': 0.7421466968378196}

Figure 2-19 Display the Results of Descending order.

One of the notable studies in the related literature to our proposed system is the work titled ' Detection of Phishing Website Using Machine Learning ' by Prasanth Baskaran [22].The objective of this project is to train machine learning models and deep neural network on the dataset created to predict phishing websites. Both phishing and legitimate URLs of websites are gathered to form a dataset and from them required URL and website content-based features are extracted. The performance level of each model is measured and compared.

System design: System modeling involves the process of developing an abstract model of a system, with each model presenting a different view or perspective of the system. The proposed system was modeled using the following diagrams:

i. Architecture diagram

ii. Flowcharts

The proposed system will be implemented using Python Programming language along with different machine learning models and libraries such as pandas, scikit-learn, python who-is, beautiful-Soup, NumPy, seaborn, and matplotlib. Etc.

Architectural design is concerned with understanding how a system should be organized and designing the overall structure of that system, it shows how different components of the system work together to achieve its main objectives. Figure 2-20 shows the architecture view of the proposed phishing detection system such that a user enters a URL link, and the link moves through different trained machine learning and deep neural network models and the best model with the highest accuracy is selected. Thus, the selected model is deployed as an API (Application Programming Interface) which is then integrated into a web application. Hence, a user interacts with the web application which is accessible across different display devices such as computers, tablets, and mobile devices.

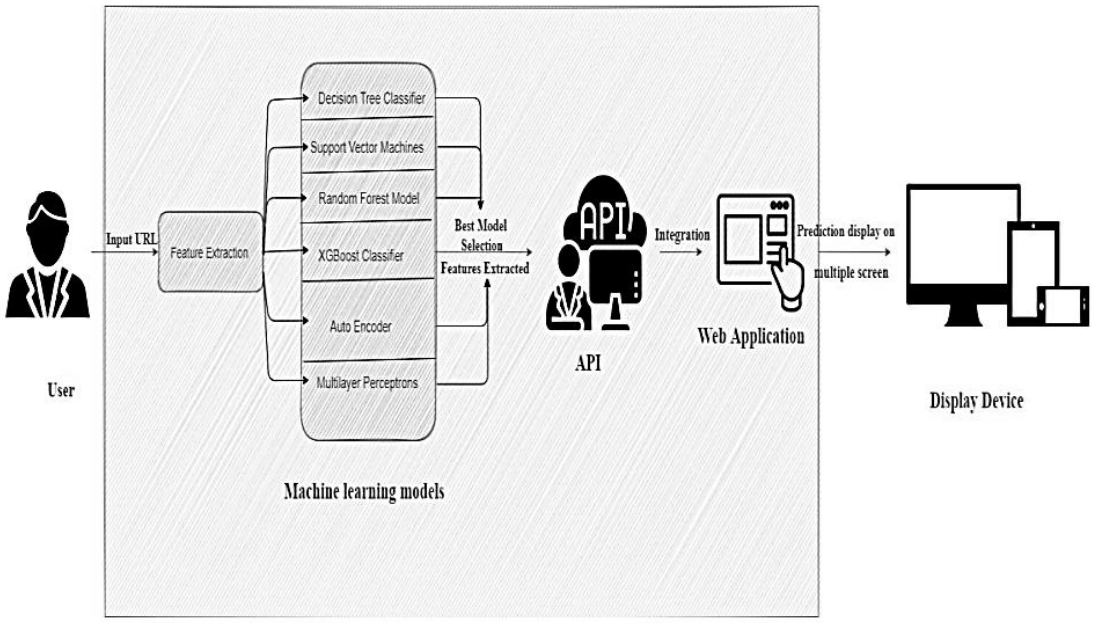


Figure 2-20 Architectural Design of the Proposed System

A **flowchart** is a diagram that depicts a process, system, or computer algorithm. It is a graphical representation of the steps that are to be performed in a system, it shows the steps in sequential order**.** Figure 2-21 shows the flow of phishing detection systems using the machine learning process.

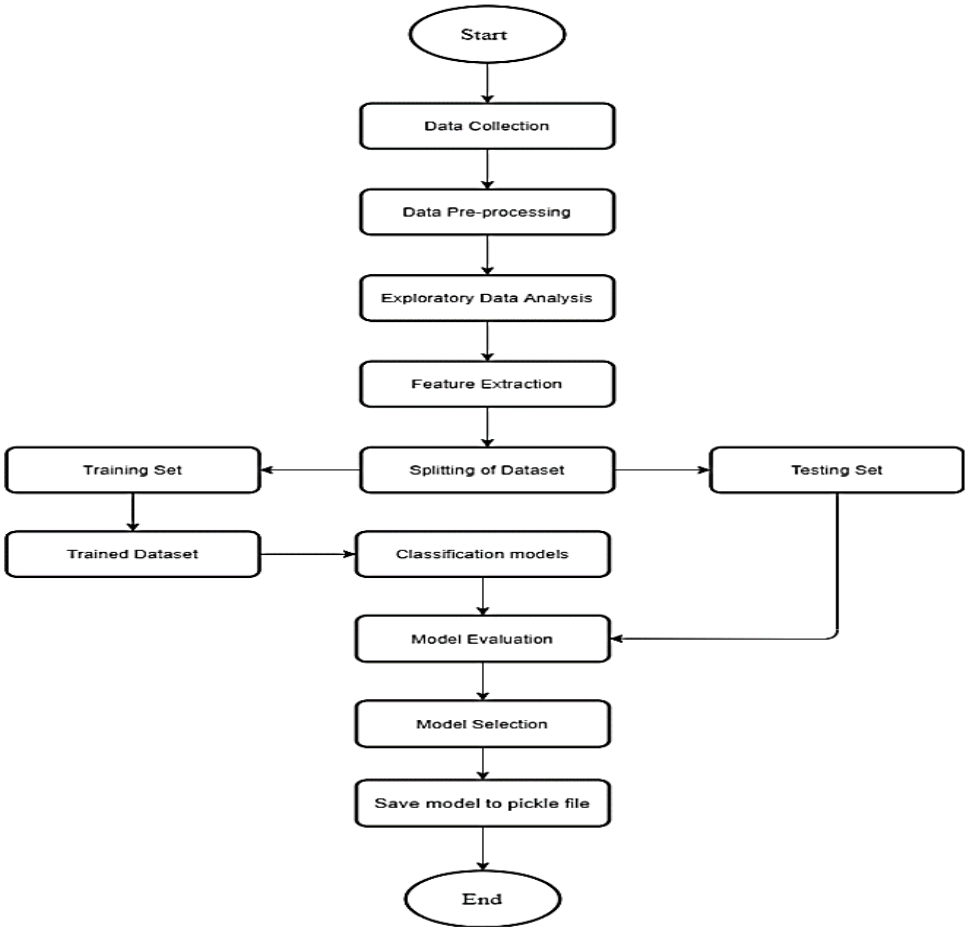


Figure 2-21 Flowchart of the proposed System.

Figure 2-22 shows the phishing detection web interface system. The user inputs a URL link, and the website validates the format of the URL and then predicts if the link is phishing or legitimate.

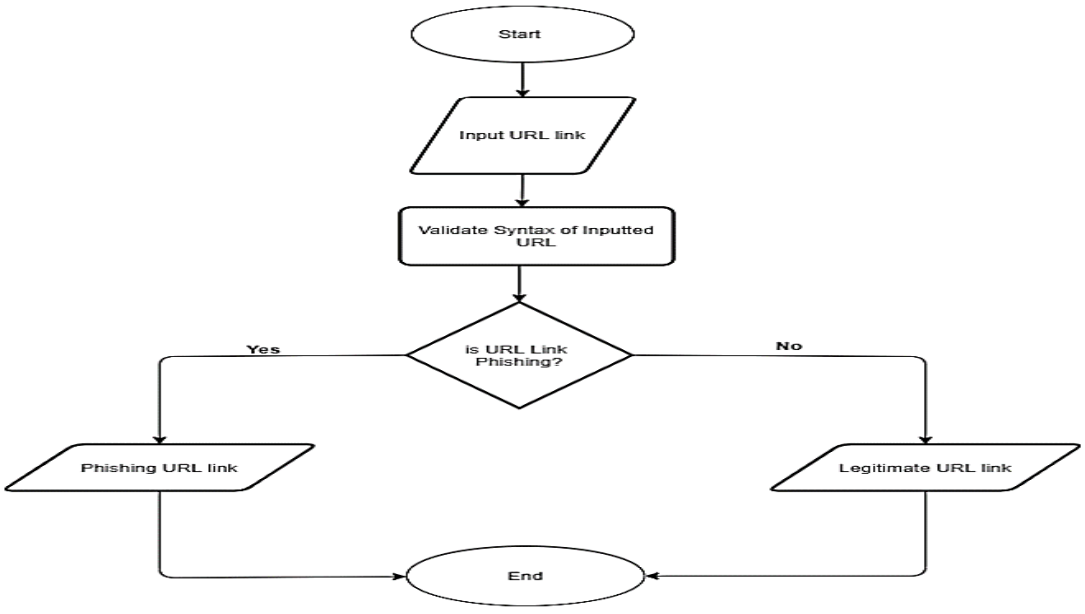
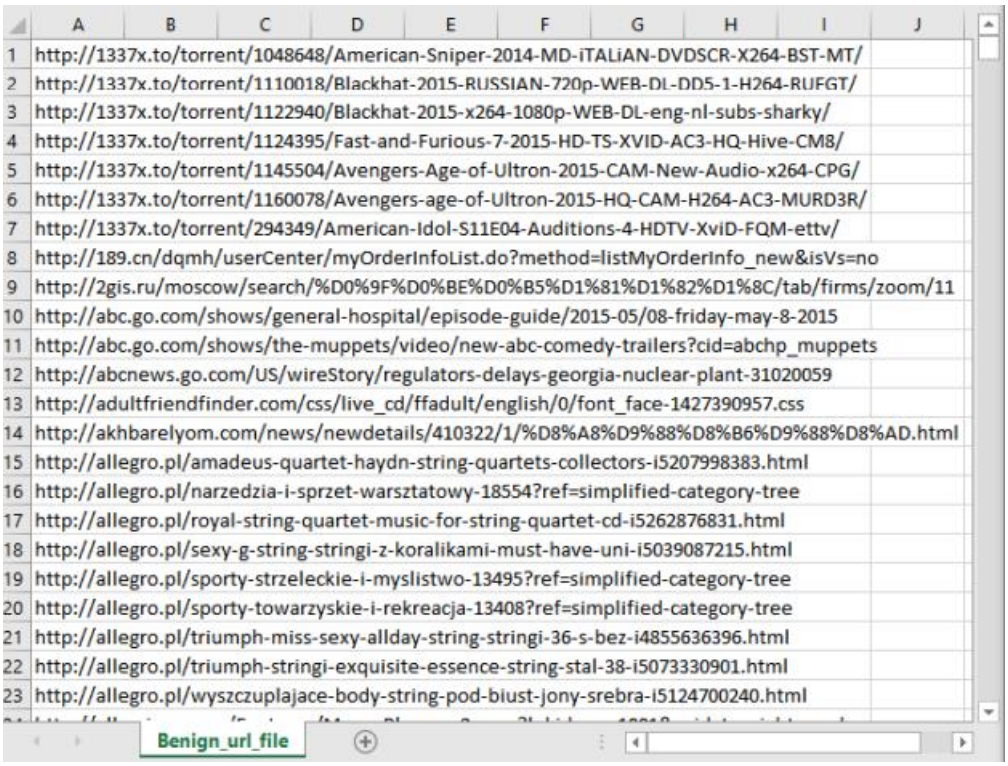


Figure 2-22 Flowchart of the web interface.

System implementation:

First data collection the dataset used for classifying the dataset into phishing and legitimate URLs was sourced from open-source websites, samples of which are shown below in figure 2-23.



	A
1	http://1337x.to/torrent/1048648/American-Sniper-2014-MD-iTALiAN-DVDSCR-X264-BST-MT/
2	http://1337x.to/torrent/1110018/Blackhat-2015-RUSSIAN-720p-WEB-DL-DD5-1-H264-RUFGT/
3	http://1337x.to/torrent/1122940/Blackhat-2015-x264-1080p-WEB-DL-eng-nl-subs-sharky/
4	http://1337x.to/torrent/1124395/Fast-and-Furious-7-2015-HD-TS-XVID-AC3-HQ-Hive-CM8/
5	http://1337x.to/torrent/1145504/Avengers-Age-of-Ultron-2015-CAM-New-Audio-x264-CPG/
6	http://1337x.to/torrent/1160078/Avengers-age-of-Ultron-2015-HQ-CAM-H264-AC3-MURD3R/
7	http://1337x.to/torrent/294349/American-Idol-S11E04-Auditions-4-HDTV-XviD-FQM-ettv/
8	http://189.cn/dqmh/userCenter/myOrderInfoList.do?method=listMyOrderInfo_new&isVs=no
9	http://2gis.ru/moscow/search/%D0%9F%D0%BE%D0%B5%D1%81%D1%82%D1%8C/tab/firms/zoom/11
10	http://abc.go.com/shows/general-hospital/episode-guide/2015-05/08-friday-may-8-2015
11	http://abc.go.com/shows/the-muppets/video/new-abc-comedy-trailers?cid=abchp_muppets
12	http://abcnews.go.com/US/wireStory/regulators-delays-georgia-nuclear-plant-31020059
13	http://adultfriendfinder.com/css/live_cd/ffadult/english/0/font_face-1427390957.css
14	http://akhbarelyom.com/news/newdetails/410322/1/%D8%A8%D9%88%D8%B6%D9%88%D8%AD.html
15	http://allegro.pl/amadeus-quartet-haydn-string-quartets-collectors-i5207998383.html
16	http://allegro.pl/narzedzia-i-sprzet-warsztatowy-18554?ref=simplified-category-tree
17	http://allegro.pl/royal-string-quartet-music-for-string-quartet-cd-i5262876831.html
18	http://allegro.pl/sexy-g-string-stringi-z-koralikami-must-have-uni-i5039087215.html
19	http://allegro.pl/sporty-strzeleckie-i-myslistwo-13495?ref=simplified-category-tree
20	http://allegro.pl/sporty-towarzyskie-i-rekreacja-13408?ref=simplified-category-tree
21	http://allegro.pl/triumph-miss-sexy-allday-string-stringi-36-s-bez-i4855636396.html
22	http://allegro.pl/triumph-stringi-exquisite-essence-string-stal-38-i5073330901.html
23	http://allegro.pl/wyszczuplajace-body-string-pod-biust-jony-srebra-i5124700240.html

Benign_url_file

Figure 2-23 Dataset of phishing URLs.

Source: The Dataset is collected from an open-source service called Phish-Tank. This dataset consists of 5,000 random phishing URLs which are collected to train the ML models.

The features extraction used on the dataset are categorized into:

i. Address bar-based features.
ii. Domain-based features
iii. Html & java-script based features.

In figure 2-24, figure 2-25, and figure 2-26 the images show the list of code feature extraction done on the dataset.

3.1. Address Bar Based Features:

Many features can be extracted that can be consided as address bar base features. Out of them, below mentioned were considered for this project.

- Domain of URL
- IP Address in URL
- "@" Symbol in URL
- Length of URL
- Depth of URL
- Redirection "//" in URL
- "http/https" in Domain name
- Using URL Shortening Services "TinyURL"
- Prefix or Suffix "-" in Domain

Each of these features are explained and the coded below:

In [12]:
```
# importing required packages for this section
from urllib.parse import urlparse,urlencode
import ipaddress
import re
```

3.1.1. Domain of the URL

Here, we are just extracting the domain present in the URL. This feature doesn't have much significance in the training. May even be dropped while training the model.

In [13]:
```
# 1.Domain of the URL (Domain)
def getDomain(url):
  domain = urlparse(url).netloc
  if re.match(r"^www.",domain):
        domain = domain.replace("www.","")
  return domain
```

In [14]:
```
# 2.Checks for IP address in URL (Have_IP)
def havingIP(url):
  try:
    ipaddress.ip_address(url)
    ip = 1
  except:
    ip = 0
  return ip
```

3.1.3. "@" Symbol in URL

Checks for the presence of '@' symbol in the URL. Using "@" symbol in the URL leads the browser to ignore everything preceding the "@" symbol and the real address often follows the "@" symbol.

If the URL has '@' symbol, the value assigned to this feature is 1 (phishing) or else 0 (legitimate).

In [15]:
```
# 3.Checks the presence of @ in URL (Have_At)
def haveAtSign(url):
  if "@" in url:
    at = 1
  else:
    at = 0
  return at
```

3.1.4. Length of URL

Computes the length of the URL. Phishers can use long URL to hide the doubtful part in the address bar. In this project, if the length of the URL is greater than or equal 54 characters then the URL classified as phishing otherwise legitimate.

If the length of URL >= 54 , the value assigned to this feature is 1 (phishing) or else 0 (legitimate).

In [16]:
```
# 4.Finding the length of URL and categorizing (URL_Length)
def getLength(url):
  if len(url) < 54:
    length = 0
  else:
    length = 1
  return length
```

Figure 2-24 Code for Address bar-based feature extraction.

3.2. Domain Based Features:

Many features can be extracted that come under this category. Out of them, below mentioned were considered for this project.

- DNS Record
- Website Traffic
- Age of Domain
- End Period of Domain

Each of these features are explained and the coded below:

In [23]:
```
!pip install python-whois
```

```
Requirement already satisfied: python-whois in c:\users\goodness\anaconda3\lib\site-packages (0.7.3)
Requirement already satisfied: future in c:\users\goodness\anaconda3\lib\site-packages (from python-whois) (0.18.2)
```

In [24]:
```
# importing required packages for this section
import re
from bs4 import BeautifulSoup
import whois
import urllib
import urllib.request
from datetime import datetime
```

In [26]:
```
# 12.Web traffic (Web_Traffic)
def web_traffic(url):
  try:
    #Filling the whitespaces in the URL if any
    url = urllib.parse.quote(url)
    rank = BeautifulSoup(urllib.request.urlopen("http://data.alexa.com/data?cli=10&dat=s&url=" + url).read(), "xml").find(
        "REACH")['RANK']
    rank = int(rank)
  except TypeError:
        return 1
  if rank <100000:
    return 1
  else:
    return 0
```

3.2.3. Age of Domain

This feature can be extracted from WHOIS database. Most phishing websites live for a short period of time. The minimum age of the legitimate domain is considered to be 12 months for this project. Age here is nothing but different between creation and expiration time.

If age of domain > 12 months, the vlaue of this feature is 1 (phishing) else 0 (legitimate).

In [27]:
```
# 13.Survival time of domain: The difference between termination time and creation time (Domain_Age)
def domainAge(domain_name):
  creation_date = domain_name.creation_date
  expiration_date = domain_name.expiration_date
  if (isinstance(creation_date,str) or isinstance(expiration_date,str)):
    try:
      creation_date = datetime.strptime(creation_date,'%Y-%m-%d')
      expiration_date = datetime.strptime(expiration_date,"%Y-%m-%d")
    except:
      return 1
  if ((expiration_date is None) or (creation_date is None)):
      return 1
  elif ((type(expiration_date) is list) or (type(creation_date) is list)):
      return 1
  else:
    ageofdomain = abs((expiration_date - creation_date).days)
    if ((ageofdomain/30) < 6):
      age = 1
    else:
      age = 0
  return age
```

Figure 2-25 Code for domain-based features extraction.

3.3. HTML and JavaScript based Features

Many features can be extracted that come under this category. Out of them, below mentioned were considered for this project.

- IFrame Redirection
- Status Bar Customization
- Disabling Right Click
- Website Forwarding

Each of these features are explained and the coded below:

In [29]:
```
# importing required packages for this section
import requests
```

3.3.1. IFrame Redirection

IFrame is an HTML tag used to display an additional webpage into one that is currently shown. Phishers can make use of the "iframe" tag and make it invisible i.e. without frame borders. In this regard, phishers make use of the "frameBorder" attribute which causes the browser to render a visual delineation.

If the iframe is empty or repsonse is not found then, the value assigned to this feature is 1 (phishing) or else 0 (legitimate).

In [30]:
```
# 15. IFrame Redirection (iFrame)
def iframe(response):
  if response == "":
      return 1
  else:
      if re.findall(r"[<iframe>|<frameBorder>]", response.text):
          return 0
      else:
          return 1
```

3.3.2. Status Bar Customization

Phishers may use JavaScript to show a fake URL in the status bar to users. To extract this feature, we must dig-out the webpage source code, particularly the "onMouseOver" event, and check if it makes any changes on the status bar

If the response is empty or onmouseover is found then, the value assigned to this feature is 1 (phishing) or else 0 (legitimate).

In [31]:
```
# 16.Checks the effect of mouse over on status bar (Mouse_Over)
def mouseOver(response):
  if response == "" :
    return 1
  else:
    if re.findall("<script>.+onmouseover.+</script>", response.text):
      return 1
    else:
      return 0
```

3.3.3. Disabling Right Click

Phishers use JavaScript to disable the right-click function, so that users cannot view and save the webpage source code. This feature is treated exactly as "Using onMouseOver to hide the Link". Nonetheless, for this feature, we will search for event "event.button==2" in the webpage source code and check if the right click is disabled.

If the response is empty or onmouseover is not found then, the value assigned to this feature is 1 (phishing) or else 0 (legitimate).

In [32]:
```
# 17.Checks the status of the right click attribute (Right_Click)
def rightClick(response):
  if response == "":
    return 1
  else:
    if re.findall(r"event.button ?== ?2", response.text):
      return 0
    else:
      return 1
```

Figure 2-26 Code for Html & java-script based features extraction.

Pre-processing data so the datasets were first cleaned to remove empty entries and fill some entries by applying data pre-processing techniques and transform the data to use in the models. Figure 2-27 shows the summary of the dataset.

```
In [8]: tuna.describe()
```

Out[8]:

	Have_IP	Have_At	URL_Length	URL_Depth	Redirection	https_Domain	TinyURL	Prefix/Suffix	DNS_Record	Web_Traffic	Do
count	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	100
mean	0.005500	0.022600	0.773400	3.072000	0.013500	0.000200	0.090300	0.093200	0.100800	0.845700	
std	0.073961	0.148632	0.418653	2.128631	0.115408	0.014141	0.286625	0.290727	0.301079	0.361254	
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	
25%	0.000000	0.000000	1.000000	2.000000	0.000000	0.000000	0.000000	0.000000	0.000000	1.000000	
50%	0.000000	0.000000	1.000000	3.000000	0.000000	0.000000	0.000000	0.000000	0.000000	1.000000	
75%	0.000000	0.000000	1.000000	4.000000	0.000000	0.000000	0.000000	0.000000	0.000000	1.000000	
max	1.000000	1.000000	1.000000	20.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	

Figure 2-27 Summary of the dataset.

System validation:

The based methodology stated that the proposed system utilizes machine learning models and deep neural networks. These models consist of Decision Tree, Support Vector Machine, XGBooster, Multilayer Perceptions, Auto Encoder Neural Network, and Random Forest. The models determine whether a website URL is phishing or legitimate. The models help give a 2-class prediction (legitimate (0) and phishing (1)). In the model development process, over six (6) machine learning models and deep neural network algorithms all together were used to detect phishing URLs using Jupyter notebook IDE with packages such as pandas, Beautiful Soup, who-is, urllib, etc.

Here are the models, their accuracy was tested using sklearn matrices with an accuracy score and their matrices are shown in figure 2-28. The XGBooster model had the highest performance score of 86.6%, the Multilayer Perceptions model had an accuracy of 86.5%, the Decision Tree model had an accuracy of 81.4%, the Random Forest model had an accuracy of 81.8%, the Support Vector Machine model had an accuracy of 80.4%, and the Auto Encoder Neural Network model had an accuracy of 16.1%.

```
#Sorting the dataframe on accuracy
results.sort_values(by=['Test Accuracy', 'Train Accuracy'], ascending=False)
```

49]:

	ML Model	Train Accuracy	Test Accuracy
3	XGBoost	0.866	0.864
2	Multilayer Perceptrons	0.865	0.864
0	Decision Tree	0.814	0.812
1	Random Forest	0.818	0.811
5	SVM	0.804	0.794
4	AutoEncoder	0.161	0.177

For the above comparision, it is clear that the XGBoost Classifier works well with this dataset.

So, saving the model for future use.

Figure 2-28 Accuracy performance of models.

One of the notable studies in the related literature to our proposed system is the work titled ' Identification of Phishing Attacks using Machine Learning Algorithm ' by [22]. The paper addresses the issue of phishing attacks, a common cybercrime method used for stealing sensitive information from both individuals and organizations. It emphasizes the evolving nature of these attacks and their diverse methods, such as emails, phone calls, and instant chats. The article's objective is to evaluate and comprehend phishing attacks through the development of a comprehensive model that considers attack stages, attackers, threats, targets, attack methods, and strategies. To accomplish this, machine learning techniques like Random Forest, XGBoost, and Logistic Regression are employed to categorize websites as either legitimate or phishing.

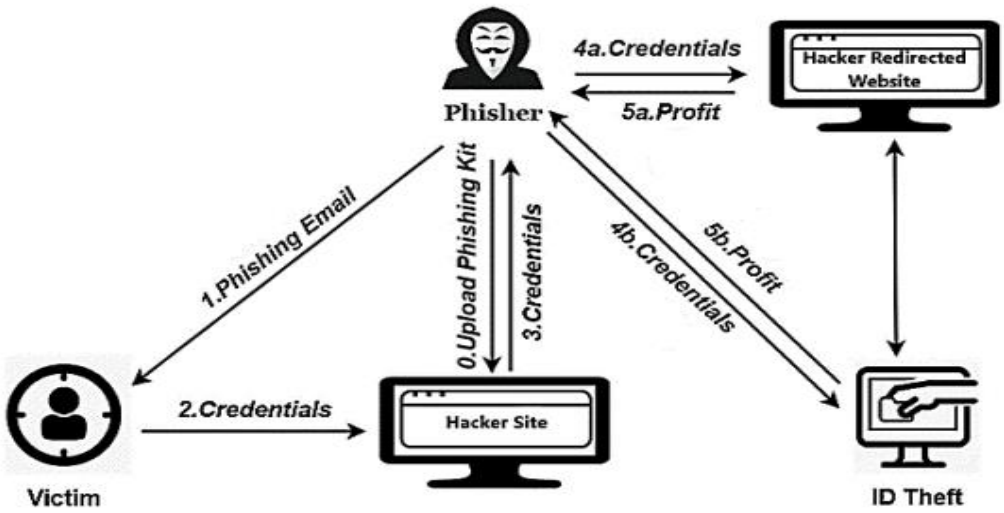


Figure 2-29 Typical phishing attack [22]

Technologies Employed:

Machine Learning: specifically Random Forest, XGBoost, and Logistic Regression, plays a pivotal role in website classification.

Data Collection: To enhance the process, a dataset of phishing URLs is collected from an open-source service (phishTank). This dataset serves as the foundation for training and assessing the machine learning models.

Machine learning models: The article utilizes supervised machine learning models, with a specific focus on phishing detection. These models encompass:

1. **Random Forest:**

Random Forest is a robust technique that helps categorize websites as genuine or potentially phishing sites. In the paper, it's used with a dataset of phishing URLs. The model learns from this dataset, which includes examples of both valid and phishing websites. Information from URLs and web pages is used to make this classification. This model excels

in handling complex datasets and offers accurate classification, making it an excellent choice for the task.

2. **XGBoost:**

XGBoost, like Random Forest, is a classification expert. It's trained using the same dataset of phishing URLs. It considers various URL and web page attributes. XGBoost assists in predicting if a URL is legitimate or suspicious. Known for its efficiency, it enhances prediction accuracy and complements the other models used in the paper.

3. **Logistic Regression:**

Logistic Regression is a basic yet effective algorithm for classifying websites. It, too, is trained with the dataset of phishing URLs, including their features. Logistic Regression calculates the likelihood of a URL being a phishing attempt. By setting a specific threshold, it sorts URLs into legitimate or phishing categories. This algorithm is simple and powerful for binary classification tasks, such as this one in the paper Top of Form [22].

One of the notable studies in the related literature to our proposed system is the work titled ' Detection and classification of phishing websites' by [23] The paper provides a comprehensive exploration of phishing as a cyber threat and outlines a methodology for classifying phishing websites using a combination of machine learning techniques and data analysis Phishing, a method combining social engineering and technical tricks, poses a threat by capturing sensitive information through deceptive means. and installing malicious software to intercept user credentials. Detecting phishing websites involves either blacklisting known malicious URLs or employing heuristic-based methods that analyze site features to distinguish phishing from legitimate sites.

Technologies Used

In the context of phishing website detection, the paper employs data mining techniques, specifically focusing on classification rules. Two main types of rules-induction techniques in data mining—associative and classification-rule techniques—are discussed, with an emphasis on the latter. The paper also utilizes machine learning techniques for classification, implemented in Python.

Dataset gathering:

- URL collection.
- Data flow.
- Host-Based Analysis.

- Lexical Feature Extraction.

Machine learning algorithms:

- Logistic regression.
- The support Vector Machine performs.
- XG Boost.
- A Multilayer Perceptron.
- Auto Encoders. [23]

One of the notable studies in the related literature to our proposed system is the work titled ' Detection of Phishing URL using Machine Learning ' by [24] The content presented covers the difficulty of detecting phishing websites, which are a major security risk. Phishing attacks can compromise sensitive information, with major ramifications for individuals, organizations, and countries. While there are advanced methods for detecting phishing websites, many of them involve manual feature engineering and struggle to detect new phishing assaults, according to the text. As a result, the project hopes to use machine learning to build a mechanism for automatically detecting phishing websites, including zero-day phishing efforts.

Technology Used is Machine Learning Libraries: Python's machine learning libraries were leveraged to build and train models for phishing website detection.

SDLC (Software Development Life Cycle): The research follows a linear-sequential model, known as the Waterfall model. The Waterfall model is a traditional approach, suitable for projects with well-defined requirements. It divides the application into smaller components that are built using frameworks and hand-written code.

Machine Learning Models:

- Decision Tree Classifier
- Random Forest Classifier
- MLPs
- XGBoost
- Autoencoder
- SVM.

System Design:

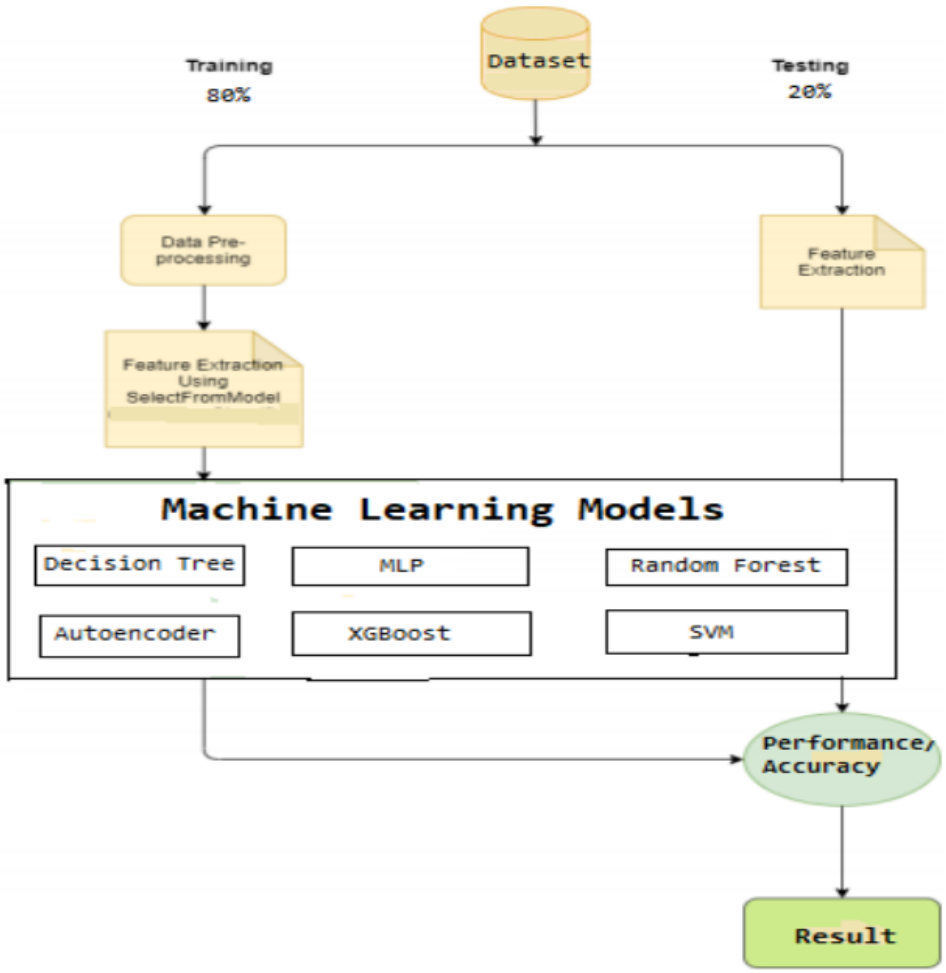


Figure 2-30 System architecture

Dataset Gathering:

The datasets have been gathered from the open-source platform Phishing tank. The collected dataset was in csv format. The dataset contains 18 columns, and they modified it using a data pre-processing technique. and then They familiarized ourselves with a few data frame approaches in order to see the features in the data. A few plots and graphs are provided for visualization and to examine how the data is spread and how features are related to one another. The Domain column has no bearing on machine learning model training. There are now 16 features and a target column. In the feature extraction file, the recovered features from the genuine and phishing URL datasets are simply concatenated, with no shuffling.

Libraries Used:

- Pandas
- Sklearn
- Numpy
- MAPTlotlib

Evaluation:

Experiment 1/ Feature Distribution

The figure below displays how the data is spread out and how different features are connected to each other. It includes several plots and graphs that illustrate these relationships.

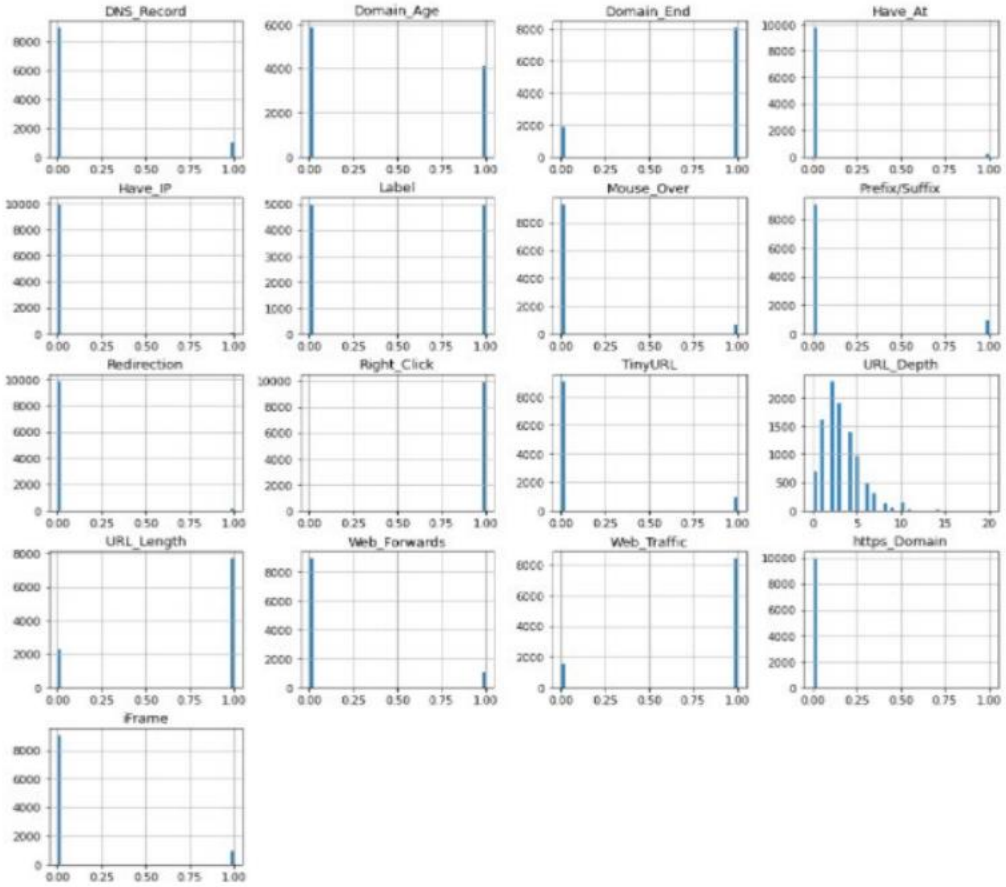


Figure 2-31 Feature Distribution

Experiment 2/ Decision Tree Classifier

The approach examines various tests to identify the most informative one for predicting the target variable. Accuracy was assessed on both training and test datasets, yielding 82.6% and 81% respectively. The Decision Tree Classifier algorithm was employed, configuring multiple parameters for model creation. The dataset was divided into X and Y train, X and Y test sets for accuracy evaluation.

```
# Decision Tree model
from sklearn.tree import DecisionTreeClassifier

# instantiate the model
tree = DecisionTreeClassifier(max_depth = 5)
# fit the model
tree.fit(X_train, y_train)
```

```
DecisionTreeClassifier(ccp_alpha=0.0, class_weight=None, criterion='gini',
                       max_depth=5, max_features=None, max_leaf_nodes=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, presort='deprecated',
                       random_state=None, splitter='best')
```

```
#predicting the target value from the model for the samples
y_test_tree = tree.predict(X_test)
y_train_tree = tree.predict(X_train)
```

Performance Evaluation:

```
#computing the accuracy of the model performance
acc_train_tree = accuracy_score(y_train,y_train_tree)
acc_test_tree = accuracy_score(y_test,y_test_tree)

print("Decision Tree: Accuracy on training Data: {:.3f}".format(acc_train_tree))
print("Decision Tree: Accuracy on test Data: {:.3f}".format(acc_test_tree))
```

```
Decision Tree: Accuracy on training Data: 0.810
Decision Tree: Accuracy on test Data: 0.826
```

Figure 2-32 Decision Tree Classifier

Experiment 3/ Random Forest Classifier

The approach to minimizing overfitting involves averaging outcomes from multiple trees, each of which overfits in different ways. Begin by determining the number of trees for a random forest model. Random forests are powerful, often effective without extensive parameter adjustments, and don't require scalable data. Accuracy for test and training sets is 83.4% and 81.4%.

```
# Random Forest model
from sklearn.ensemble import RandomForestClassifier

# instantiate the model
forest = RandomForestClassifier(max_depth=5)

# fit the model
forest.fit(X_train, y_train)
```

```
RandomForestClassifier(bootstrap=True, ccp_alpha=0.0, class_weight=None,
                       criterion='gini', max_depth=5, max_features='auto',
                       max_leaf_nodes=None, max_samples=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, n_estimators=100,
                       n_jobs=None, oob_score=False, random_state=None,
                       verbose=0, warm_start=False)
```

```
#predicting the target value from the model for the samples
y_test_forest = forest.predict(X_test)
y_train_forest = forest.predict(X_train)
```

Performance Evaluation:

```
#computing the accuracy of the model performance
acc_train_forest = accuracy_score(y_train,y_train_forest)
acc_test_forest = accuracy_score(y_test,y_test_forest)

print("Random forest: Accuracy on training Data: {:.3f}".format(acc_train_forest))
print("Random forest: Accuracy on test Data: {:.3f}".format(acc_test_forest))
```

```
Random forest: Accuracy on training Data: 0.814
Random forest: Accuracy on test Data: 0.834
```

Figure 2-33 Random Forest Classifier

Result

In thier project, XGBoost ML model showed the highest accuracy among all models, while SVM had the lowest. XGBoost performed well across various metrics, indicating its robustness and effectiveness in avoiding overfitting. Its strategies, like subsampling, regularization, and column subsampling, contribute to its superior performance compared to SVM, MLP, and Random Forests.

```
#creating dataframe
results = pd.DataFrame({ 'ML Model': ML_Model,
    'Train Accuracy': acc_train,
    'Test Accuracy': acc_test})
results
```

	ML Model	Train Accuracy	Test Accuracy
0	Decision Tree	0.810	0.826
1	Random Forest	0.814	0.834
2	Multilayer Perceptrons	0.858	0.863
3	XGBoost	0.866	0.864
4	AutoEncoder	0.819	0.818
5	SVM	0.798	0.818

Figure 2-34 Accuracy of testing and training.

2.6 Software Development

In the ever-evolving landscape of software development, choosing the right methodology can significantly impact the success of a project. When it comes to developing a project aimed at preventing phishing attacks, the choice between Agile and plan-driven (sometimes referred to as "Waterfall") and MLDL (Machine Learning Development Life Cycle) methodologies is a critical decision.

2.6.1 Machine Learning Development Life Cycle

The machine learning development lifecycle is a structured procedure employed when creating machine learning models for addressing tasks or challenges. It encompasses a series of phases, such as problem definition, data gathering, data preparation, feature crafting, model choice, model training, model assessment, model enhancement, deployment, and continuous monitoring and upkeep. This iterative process aims to progressively improve and refine machine learning models to ensure their accuracy and effectiveness over time. [25]

Why the Machine Learning Development Lifecycle is Used:

1. **Structured Approach:** The machine learning development lifecycle offers a well-structured methodology for the creation of machine learning models. It delineates a series of sequential phases, ensuring that data scientists and engineers possess a well-defined roadmap for their projects.
2. **Problem Definition:** It commences with problem definition, a pivotal step for comprehending the business objectives and aligning the machine learning solution with the organization's overarching goals.
3. **Data Quality:** Emphasis is placed on the critical nature of data collection and data preprocessing. High-quality data is underscored as indispensable for the effective training of accurate models.
4. **Feature Engineering:** Feature engineering involves the art of selecting or crafting pertinent features that can enhance the model's predictive capabilities. It guarantees that the model is trained using the most appropriate input variables.
5. **Model Selection:** Within the lifecycle, a meticulous process of selecting machine learning algorithms or model architectures takes place, tailored to the specific problem. This selection is influenced by various factors, including data type, problem type, and performance requisites.
6. **Model Training and Evaluation:** Models undergo training using a subset of the data, followed by evaluation utilizing metrics to gauge their performance. This aids in selecting the optimal model and identifying areas for enhancement.
7. **Optimization:** Model optimization involves the fine-tuning of model parameters and hyperparameters to elevate overall performance.
8. **Deployment:** Once a satisfactory model has been developed, it is ready for deployment in a production environment to make real-time predictions or support decision-making processes.
9. **Continuous Improvement:** Acknowledging the dynamic nature of data, the lifecycle acknowledges the need for machine learning models to adapt to evolving data patterns and shifting requirements. Hence, regular monitoring and maintenance are pivotal for keeping models current.
10. **Reproducibility and Accountability:** Adhering to a structured development process ensures that all developmental and decision-making steps are thoroughly documented.

This, in turn, fosters the principles of reproducibility and accountability in machine learning projects.

Proofpoint [27] outlines their structured approach for creating and training machine learning models, referred to as the Machine Learning Development Lifecycle (MLDL). This process consists of the following phases:

1. **Requirements Gathering and Analysis:** Proofpoint initiates by collecting and analyzing requirements from their clients to gain insight into their specific needs for detecting phishing attacks. This involves understanding the types of phishing attacks clients face and the performance criteria for the solution.
2. **Model Design:** Once Proofpoint has a comprehensive understanding of the requirements, they proceed to design the machine learning model. This entails selecting the most suitable machine learning algorithm, determining the input features, and defining the output labels.
3. **Data Collection and Preparation:** Subsequently, Proofpoint acquires and prepares the data necessary to train the machine learning model. This dataset includes both phishing and legitimate emails, and it undergoes a thorough cleansing and preprocessing process to ensure compatibility with the machine learning algorithm.
4. **Model Training:** Proofpoint then proceeds to train the machine learning model using the prepared dataset. This involves feeding the data to the algorithm and allowing it to identify the patterns that distinguish phishing emails from legitimate ones.
5. **Model Evaluation:** Following the model's training, Proofpoint assesses its performance using a separate test set comprising emails the model has not encountered before. This evaluation is crucial to confirm that the model effectively generalizes and does not overfit to the training data.
6. **Model Deployment:** Once the machine learning model has been thoroughly evaluated and proven to perform well, Proofpoint deploys it for practical use. This means the model is employed to classify incoming emails as either phishing or legitimate. [28]

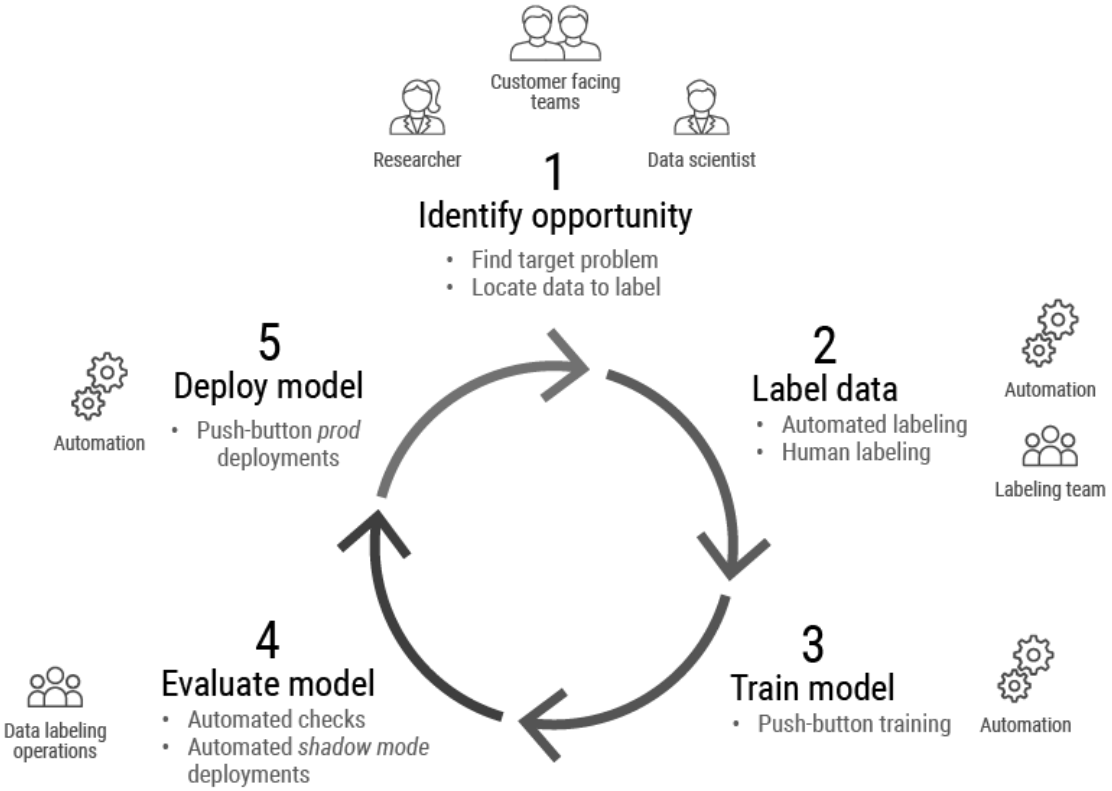


Figure 2-35 overview of MLDL life cycle at Proofpoint [28]

2.6.2 Waterfall

The Waterfall Model is a conventional software development methodology known for its linear and step-by-step project management approach. It is typically employed in large-scale software projects with clearly defined requirements that are unlikely to change significantly during development. The Waterfall Model follows a structured sequence of phases, including requirements gathering, system design, implementation, testing, deployment, and maintenance, which must be completed sequentially.

How Machine Learning Projects Use Waterfall:

The Waterfall model is a traditional, linear approach to software development, and its rigid structure may not align perfectly with the iterative and exploratory nature of machine learning projects. However, some organizations might choose to use a modified or adapted version of the Waterfall model for certain aspects of their machine learning initiatives.

Reasons for Using Waterfall in Machine Learning Projects:

1-Clearly Defined Requirements: If a machine learning project has well-defined and stable requirements upfront, the Waterfall model may be suitable. For example, if the dataset,

features, and desired outcomes are known in advance, the project may follow a structured progression through planning, modeling, coding, testing, and deployment.

2- Regulatory Compliance: In industries where, regulatory compliance is crucial and requires documentation at each phase, the Waterfall model might be preferred. This is often the case in sectors like finance or healthcare where strict guidelines must be followed.

3- Small-Scale ML Projects: Smaller machine learning projects with limited complexity and clear objectives might find the Waterfall model more manageable. Projects with a narrow scope and well-defined goals can benefit from a systematic and phased approach.

4- Stability in Requirements: If the project requirements are well-understood, stable, and unlikely to change significantly during development, the Waterfall model can be advantageous. This is particularly true when dealing with machine learning projects that have clearly defined objectives and datasets.

5-Regulatory Compliance: In industries where adherence to regulations is critical, such as healthcare or finance, the Waterfall model's documentation-centric approach can provide the necessary audit trail and transparency.

6-Predictable Timeline: If there is a need for a predictable timeline and a well-structured plan, the Waterfall model allows for a sequential and phased development approach.

7-Client Involvement at Key Stages: If client or stakeholder involvement is concentrated at specific milestones and less frequent feedback is acceptable, the Waterfall model might be suitable.

8-Resource Planning: Waterfall can be useful when resources need to be allocated and planned well in advance, and when the team is accustomed to a more structured and linear development process.

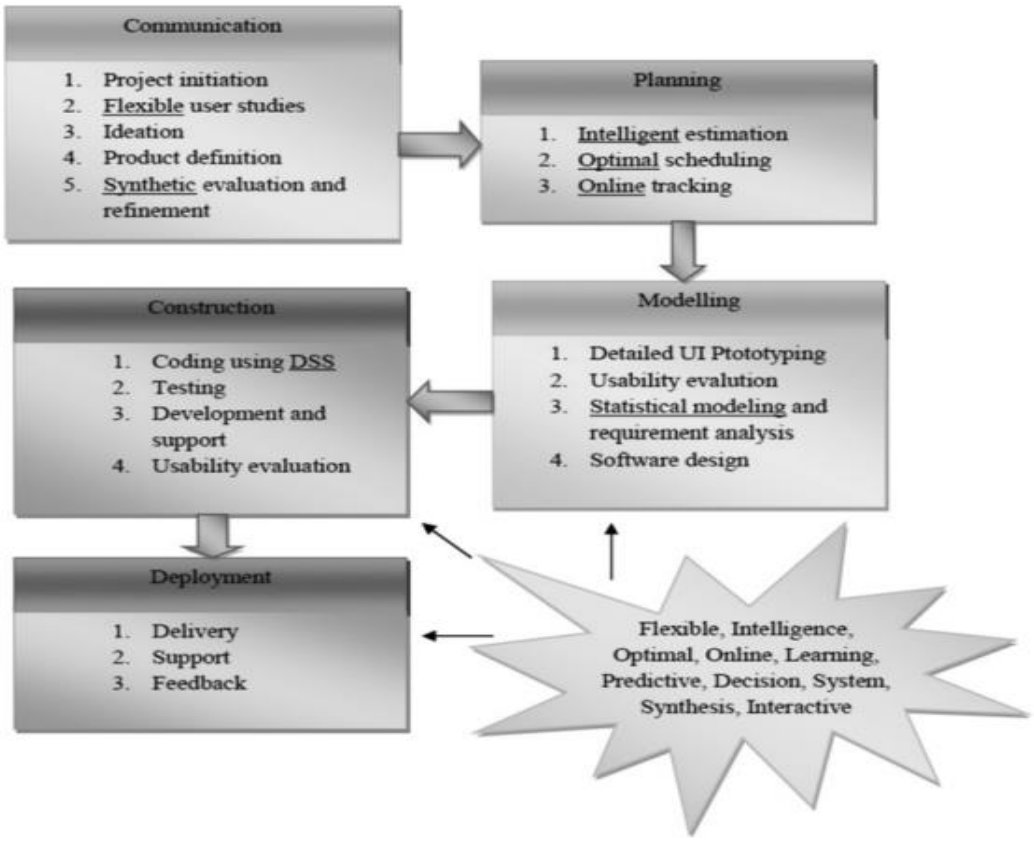


Figure 2-36 Integrating AI activities in waterfall model [24]

The Waterfall Model is used for software development projects in situations where the following conditions are met:

1. **Clear Requirements:** When project requirements are well-defined and leave little room for ambiguity or significant changes.
2. **Stability:** In situations where the project's scope, objectives, and technology stack remain stable and are unlikely to undergo major alterations during development.
3. **Regulatory Compliance:** Projects subject to strict regulatory compliance, such as those in aerospace or healthcare industries, may favor the Waterfall Model due to its rigorous documentation and testing processes.
4. **Predictability:** When stakeholders require a well-structured project plan with clearly defined milestones and deliverables, making progress tracking more straightforward.
5. **Legacy Systems:** It is suitable for projects involving the maintenance or update of existing legacy systems, where a structured approach is essential.
6. **Risk Aversion:** Organizations that prioritize risk minimization may opt for the Waterfall Model due to its thorough planning and documentation, reducing the likelihood of unexpected issues. [29]

2.6.3 Agile

Agile Overview: Agile represents a software development methodology that underscores flexibility, collaboration, and a customer-centric approach. It diverges from conventional software development models, such as the Waterfall model, by fragmenting the development process into smaller, recurrent cycles. Key aspects of Agile encompass:

1. Iterative Development: Agile divides the project into compact iterations or sprints, delivering a potentially shippable product increment at the end of each iteration.
2. Collaboration: Agile advocates for close collaboration between multifunctional teams, including developers, testers, and customers.
3. Customer Involvement: Agile promotes continuous customer feedback and adjusts the software in response to evolving requirements.
4. Adaptability: Agile permits flexibility in addressing changing customer needs, requirements, and priorities.

Enhanced Agile Model with AI Activities:

The extended Agile model integrates AI activities into both the communication and iterative phases.

Communication Phase: User Studied, Ideation, Product Definition, Usability Evaluation: These steps involve advanced natural language processing and optimization algorithms to define products and evaluation criteria.

Iterative Phase:

Release: Predictive Initialization Predicting and computing the velocity of the project using predictive models from AI to enhance the software development cycle.

Planning: Optimal Planning Formulating plans considering different constraints and objectives. It involves mathematical programming to find the optimal solution.

Relative Prediction: Integrating AI-based relative prediction to improve the planning chart and reduce the iterative phase.

Design:

Intelligent Design and Interactive User Interface: Including intelligent design and interactive interfaces to improve the design process.

Automatic Suggestion Tool: Integrating a tool to suggest improvements within the system, reducing manual costs.

Coding: Error-Tolerant Decision Support System Integrating an AI-based system to reduce human intervention and mistakes during coding.

Test: Objective and Constraint-Aware Optimal Testing Integrating optimal testing phases considering objectives like coverage, fault detection, capability, and cost.

Multi-Objective Based Regression Testing: Using an optimization algorithm for regression testing to find a test suite that satisfies various requirements.

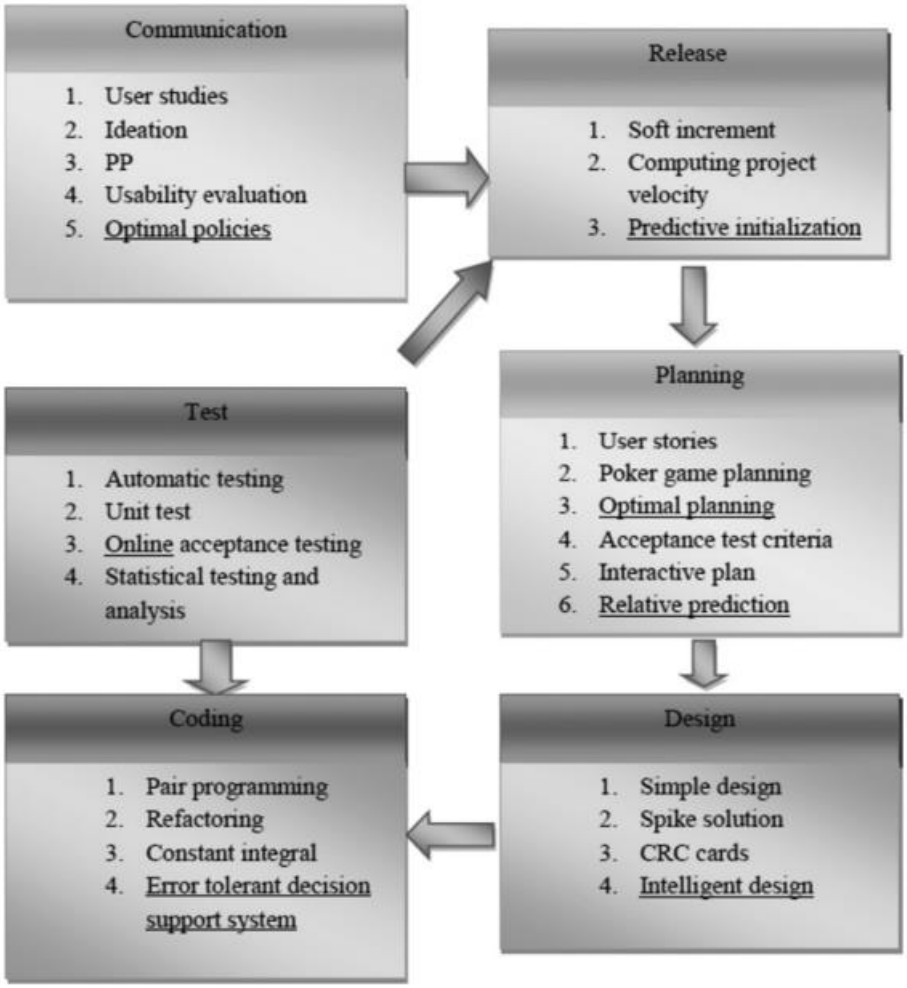


Figure 2-37 Integrating AI activities in agile model [24]

Reasons for Using Agile in Machine Learning Projects:

1. Flexibility: Agile allows for flexibility in handling evolving data, models, and project requirements.
2. Collaboration: Facilitates constant communication and collaboration between team members and stakeholders.
3. Adaptability: Enables quick adjustments based on feedback, improving the model's accuracy and performance.

4. Customer Satisfaction: Frequent releases ensure that stakeholders see tangible progress and can provide timely feedback. [29]

2.6.4 Relationship Between the Relevant Work and Our Own Work

Table 2-2 The relationship between different systems and our own system.

PROJECT	PHISHING ATTACK BY URL	PHISHING ATTACK BY EMAIL	WEBSITE	MULTI LANGUAGES	USING MACHINE LEARNING	EXTRACTION METHODS
[21]	✓	✗	✗	✗	✓	FEATURE EXTRACTION
[22]	✓	✗	✗	✗	✓	FEATURE EXTRACTION
[23]	✓	✗	✗	✗	✓	FEATURE EXTRACTION
[24]	✓	✗	✗	✗	✓	FEATURE EXTRACTION
PHISHWARDEN	✓	✓	✓	✓	✓	FEATURE EXTRACTION

2.7 Summary

In This chapter we covered 4 sections, first we did an overview of Artificial intelligence & Cybersecurity technology, including its definition, main platforms, technologies used, and general use cases of Ai technology & Security. Second, focus on related work that uses Ai & Cybersecurity in the field of social engineering, specifically in the areas of Phishing attacks. Lastly, the chapter ends with a comparison between Software development methodologies of building the project. The next chapter will provide an analysis of the system.

In this chapter, the primary objective was to comprehensively review earlier, related work to our project. By conducting an in-depth examination of existing literature and projects, we aimed to establish a strong foundation of knowledge and identify key insights relevant to our research area. This allowed us to build on the collective wisdom of previous endeavors and inform the direction of our project, hence Moving forward, Analysis Chapter will focus on analyzing the system requirements and selecting the appropriate models for the system analysis. This step is crucial for understanding the specific needs and constraints of our project, laying the groundwork for a systematic and effective analysis that will inform the subsequent stages of development.

Chapter 3. System Analysis

3.1 Introduction

This chapter is divided into three sections. The first section details the software development and analysis methods. The AI technology that is used for this project, and all the tools required to complete the project. The second section discusses requirements elicitation, which determines the proposed system's functional and non-functional requirements. The third section documents the requirements specification process and presents and discusses the system's analysis models (use cases and class diagram) in high-level description, hence in this chapter we successfully did the second objective in our project.

3.2 Methodology

Based on our previous research in Chapter 2 of AI-based development systems, we found several methods used in building a machine learning model, such as waterfall, agile development, and the Machine Learning Development Life Cycle. In addition to Integrating AI activities in agile model and Integrating AI activities in waterfall, in the next two sections we will talk about SDLC and the Analysis.

we will discuss the software development method, and which one is appropriate for this project, as well as the AI platform, setup, and tools/technologies used We have three key activities in the program analysis stage that must be specified. Before beginning the testing process, gather needs and specifications, then analyze, and test them. They must be validated. During the analysis phase of our proposed solution, we used this strategy.

Establishing requirements in this section, we will express the functional and non-functional needs in natural language that the developer and customer can comprehend. We will create tables to explain functional requirements in greater detail and we will specify how are we going to achieve the non-functional requirement throughout the requirements specification step. In System modeling to design the system, we will utilize the star UML application and create UML drawings such as class diagrams, use case diagrams, and sequence diagrams.

3.2.1 SDLC

According to Michael and Nagasunder [20] The Waterfall model is used in the research. It is a linear-sequential approach. The Waterfall model is a standard strategy that works effectively for projects with specific objectives. It subdivides the application into smaller components constructed with frameworks and hand-written code.

3.2.2 Reasons for The Use of Waterfall Model

There are many reasons Supportive to choose Waterfall model:

Clear Project Scope: Waterfall is suitable when the project requirements are well-defined and unlikely to change during the development process. It's effective for projects with a stable and fixed scope.

Predictable and Sequential Phases: Waterfall follows a linear and sequential approach with distinct phases. This structure makes it easy to plan, monitor, and control each stage of the development process.

Small to Medium-sized Projects: Waterfall is often suitable for small to medium-sized projects with straightforward requirements. It might face challenges with large and complex projects where changes are more likely.

Splitting the project into two terms: Waterfall is suitable for us because we will implement the documentation part in term one and the implementation part in the second term.

3.2.3 Analysis

The Figure 3-1 illustrates the sequential process of requirement elicitation, requirement specification, and system modeling. Requirement elicitation involves understanding the system's needs and constraints. Following this, requirement specification documents these requirements clearly. Finally, system modeling creates abstract representations of the system.

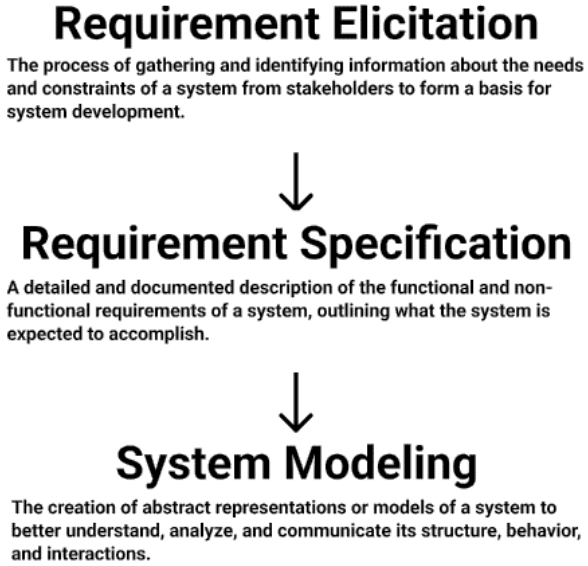


Figure 3-1: Description of the Analysis stage [30]

Before initiating the design phase, it is essential to collect requirements and specifications, followed by their analysis and validation. These three crucial steps must be specified during the

program analysis stage. In the process of developing our proposed solution, we utilized the analysis phase to examine and validate the gathered needs and specifications.

In Requirement elicitation Here we will describe the functional and non-functional requirements in natural language that are easy to understand for the developer and the customer, we can do this by using brainstorming and revision of similar existing systems as a mean to gather the functionality of the system requirements.

In the Requirement specification stage, we will design tables to describe functional and non-functional requirements in more detail, we can do this by applying Structured Natural Language requirements specification methods.

In System modeling Here we will use STARUML program to design the system, we will make UML diagrams, by doing this we can visualize the Object-oriented models.

3.3 Requirements Elicitation

The Requirements elicitation describe the system services and constraints. This section provides the functional, non-functional, and user or domain requirements. We gathered these requirements from some of the related work projects we inspected, such as Proofpoint, SpoofGuard, Detection of Phishing URL using Machine Learning. We, also, did brainstorming to figure out how the project will function [31].

3.3.1 System Requirements

SR1. To build an AI-powered website to classify and detect emails and URLs for phishing trials, with high accuracy and precision and easy to use system.

SR2. The website shall provide clear and concise reporting information about the phishing classification and analysis results.

3.3.2 Functional Requirements

For SR1 :

FR1. The system must allow all entities to submit email/website address for analysis.

FR2. The machine learning model must do Feature extraction to the Email/URL.

FR3. The machine learning model must classify the Email/URL.

FR4. The machine learning model must do Pre-processing to the Email/URL.

For SR2 :

FR5. The system must allow all entities to view the result of detection.

FR6. The system must allow all entities to make feedback.

3.3.3 Non-Functional Requirements

- **Reliability:** The system should be able to handle and prevent errors.
- **Performance:** The system should be able to optimize the performance of the machine learning module to deliver fast and efficient.
- **Accuracy:** The system should achieve a high level of accuracy in predicting phishing attacks while minimizing false positives.
- **Ease of use:** app user interface should be easy to explore, user should be familiar after second use.

3.4 Requirements Specification

This section comprises use case illustrations and descriptions, which are an important part of the Requirements Specification. A use case in the Unified Modeling Language (UML) is a scenario-based technique for identifying and characterizing the participants in an interaction. Figure 3-2 depicts a collection of use cases that describe all possible interactions with the system, followed by a description table for each that describes, Participating Actors, inputs, Output, Output destination, Input resources, enter condition, Exit condition, Flow of events, Exceptions of flow of event, Special requirements. And then how the nonfunction will be achieved Functional Requirements.

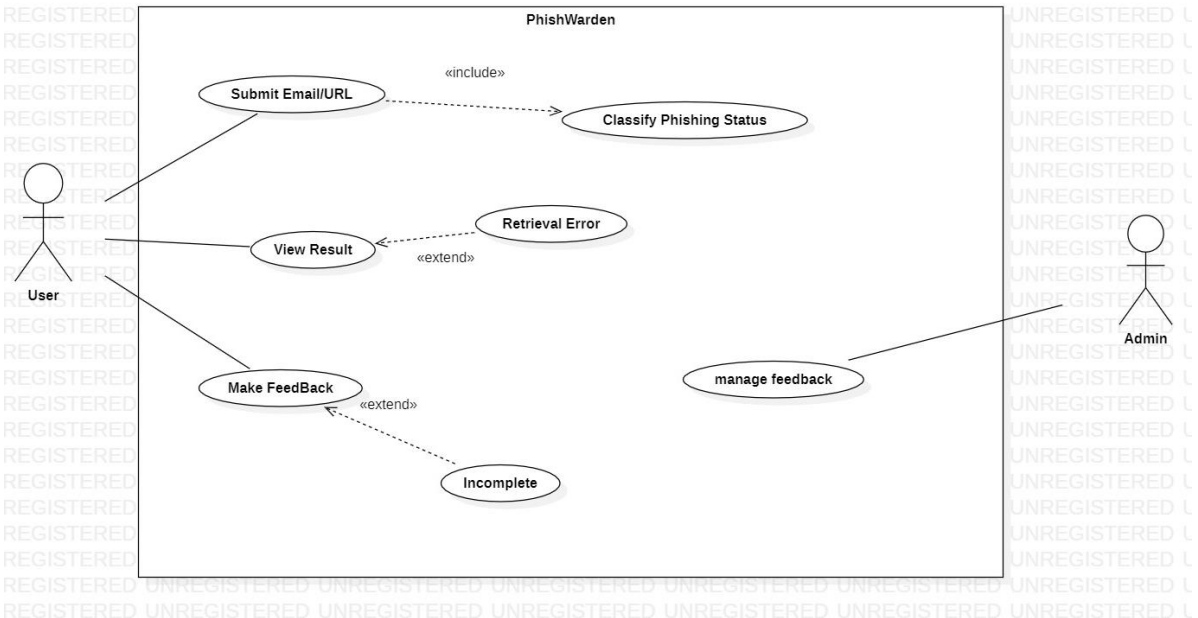


Figure 3-2 Use Case Diagram

The PhishWarden machine learning model is designed to classify emails and URLs as phishing or legitimate. The two main actors involved in the system are the user and the Admin. The User interacts with the system by submitting emails and URLs for classification.

The User submits an email or URL to the system. And The system pre-processes the email or URL by tokenizing it, removing stop words, and lemmatizing the text. Also, the system extracts relevant features from the pre-processed text. Then The system feeds the extracted features into the PhishWarden Machine Learning Model. Then The PhishWarden Machine Learning Model uses its trained knowledge to classify the email or URL as phishing or legitimate. Then The system returns the classification result to the User.

The user can view the classification result and provide feedback if necessary. The admin can manage the feedback that comes from user.

1-Submit Email/URL: The user submits an email or URL to the system to be classified as phishing or legitimate.

2-Classify Email/URL: The system classifies the submitted email or URL as phishing or legitimate.

3-View Result: The user views the result of the classification.

4-Make Feedback: The user can provide feedback on the classification result.

5-Manage feedback: the admin can manage the feedback.

Table 3-1 Description of the submit Email/URL

Requirement	FR1
Brief description	The system enables users to submit email addresses or website URLs for phishing analysis. This functionality allows users to leverage the machine learning model to classify whether the provided email/website is phishing or legitimate
Participating Actors	User
Inputs	Email address or website URL submitted by the user.
Input resources	Keyboard.
Output	Phishing or legitimate classification result for the submitted email address or website URL
Output destination	Screen, Database.
Enter condition	Users must be authenticated by logging into their accounts
Exit condition	The user receives the classification result after submitting an email or website
Flow of events	2- User logs in. 3- User enters an email address or website URL.

	4- The system utilizes the machine learning module to analyze the submitted email/website for phishing characteristics. 5- The user receives immediate feedback on whether the submission is classified as phishing or legitimate.
Exceptions of flow of event	None
Special requirements	None.

Table 3-2 Description of the feature extraction

Requirement	FR2
Brief description	This function involves extracting relevant features from pre-processed email and URL data to create a representation suitable for input into the machine learning model for classification.
Participating Actors	User, Developer
Inputs	Pre-processed email data refers to information that has undergone Stemming, tokenization, and Remove Stop words, Lemmatization ,while pre-processed URL data represents URLs that have been processed and are prepared for feature extraction.
Input resources	Pre-processed email data is the output from the pre-processing function and is ready for feature extraction.
Output	A numerical representation of the extracted features that serves as input for the machine learning model.
Output destination	The feature vector is passed as input to the machine learning model for classification
Enter condition	Pre-processed data email and URL must be available for feature extraction.
Exit condition	The feature vector is successfully created and ready to be used as input for the machine learning model.
Flow of events	1- The Machine Learning Model receives pre-processed email data and URL 2- Identify and select relevant features that are indicative of phishing or legitimate characteristics. 3- Convert the selected features into a numerical representation suitable for machine learning algorithms. 4- Assemble the numerical features into a feature vector, forming a structured input for the machine learning model. 5- The feature vector is generated and passed on as input to the machine learning model for classification.
Exceptions of flow of event	None
Special requirements	None

Table 3-3 Description of the classification for Email/URL

Requirement	FR3
Brief description	The Machine Learning Model for Email/URL Classification is designed to analyze and categorize input data, specifically email addresses and website URLs, into either phishing or legitimate categories. The model utilizes pre-trained parameters and algorithms to make predictions based on patterns learned during the training phase.
Participating Actors	User

Inputs	Numerical representation of extracted features from pre-processed email and URL data.
Input resources	Trained Model and Pre-processing Algorithms.
Output	The outcome of the classification process, indicating whether the email or URL is classified as phishing or legitimate.
Output destination	Screen
Enter condition	The model is triggered when it receives a request for the classification of an email address or website URL.
Exit condition	The model completes its analysis and generates a classification result.
Flow of events	1- The machine learning model receives raw email addresses or website URLs for classification. 2- The input data undergoes pre-processing steps to ensure it is in a suitable format for analysis. 3- Relevant features are extracted from the pre-processed data to represent important characteristics. 4- The pre-processed data is fed into the trained model. 5- The model applies its learned parameters and algorithms to make predictions. 6- The model generates a classification result, indicating whether the input is classified as phishing or legitimate.
Exceptions of flow of event	None
Special requirements	None.

Table 3-4 Description of the pre-processing function for Email/URL

Requirement	FR4
Brief description	The machine learning model incorporates a pre-processing function for emails and URLs to enhance the quality of input data before conducting phishing detection. This pre-processing ensures that the data is in a standardized and optimal format for effective analysis.
Participating Actors	User
Inputs	Raw email addresses or URLs the unprocessed input data containing information to be analyzed.
Input resources	Unprocessed email addresses or URLs submitted by users.
Output	The cleaned and transformed data ready for feature extraction and analysis.
Output destination	Input to Feature Extraction: The pre-processed data is passed as input to the feature extraction stage.
Enter condition	The machine learning model receives raw email or URL data for analysis.
Exit condition	The pre-processing stage is successfully completed, and the cleaned data is ready for feature extraction
Flow of events	1- The machine learning model receives raw email addresses or URLs as input. 2- Reduce words to their base or root form to normalize variations. 3- Eliminate common and irrelevant words that may not contribute to the analysis. 6- Break down the email or URL into smaller units (tokens) for analysis. 7- Reduce words to their base or dictionary form to ensure uniformity. 8- Extract relevant features from the pre-processed data for the phishing detection model.

	9- The pre-processed email or URL data is generated as the input to Feature extraction.
Exceptions of flow of event	If the submitted email address or URL is empty or contains no usable information, the model may skip processing or generate an appropriate response.
Special requirements	None

Table 3-5 Description of the view results of phishing detection

Requirement	FR5
Brief description	The system provides a functionality that allows users to view the results of phishing detection. Users can access the classification outcome (phishing or legitimate) for email addresses or website URLs submitted for analysis.
Participating Actors	User
Inputs	Request to view the detection results for their submitted entry.
Input resources	Database storing detection results and analysis reports. User authentication system for access control.
Output	Classification result.
Output destination	Display the classification results to the user via the web interface.
Enter condition	Users must log in with valid credentials to access the detection results
Exit condition	Detection results are displayed.
Flow of events	1-User Requests Detection Result. 2- System Retrieves Results. 3- The classification result is displayed to the user.
Exceptions of flow of event	The system detects that there was an error retrieving the detection results.
Special requirements	None.

Table 3-6 Description of the submit feedback.

Requirement	FR6
Brief description	The system incorporates a feedback mechanism that enables users to provide feedback on the accuracy of phishing detection results
Participating Actors	User
Inputs	Feedback submitted by the user regarding the accuracy of phishing detection results.
Input resources	Keyboard
Output	Confirmation message acknowledging the successful submission of feedback.
Output destination	Screen.
Enter condition	Users must be authenticated by logging into their accounts.
Exit condition	The user receives a confirmation message after successfully submitting feedback.
Flow of events	1-User navigates to the feedback section. 2-User provides feedback on the accuracy of phishing detection results. 3-System records the feedback in the database. 4-Confirmation message is displayed to the user.

Exceptions of flow of event	If the provided feedback is incomplete or invalid, the system prompts the user to provide valid input.
Special requirements	None.

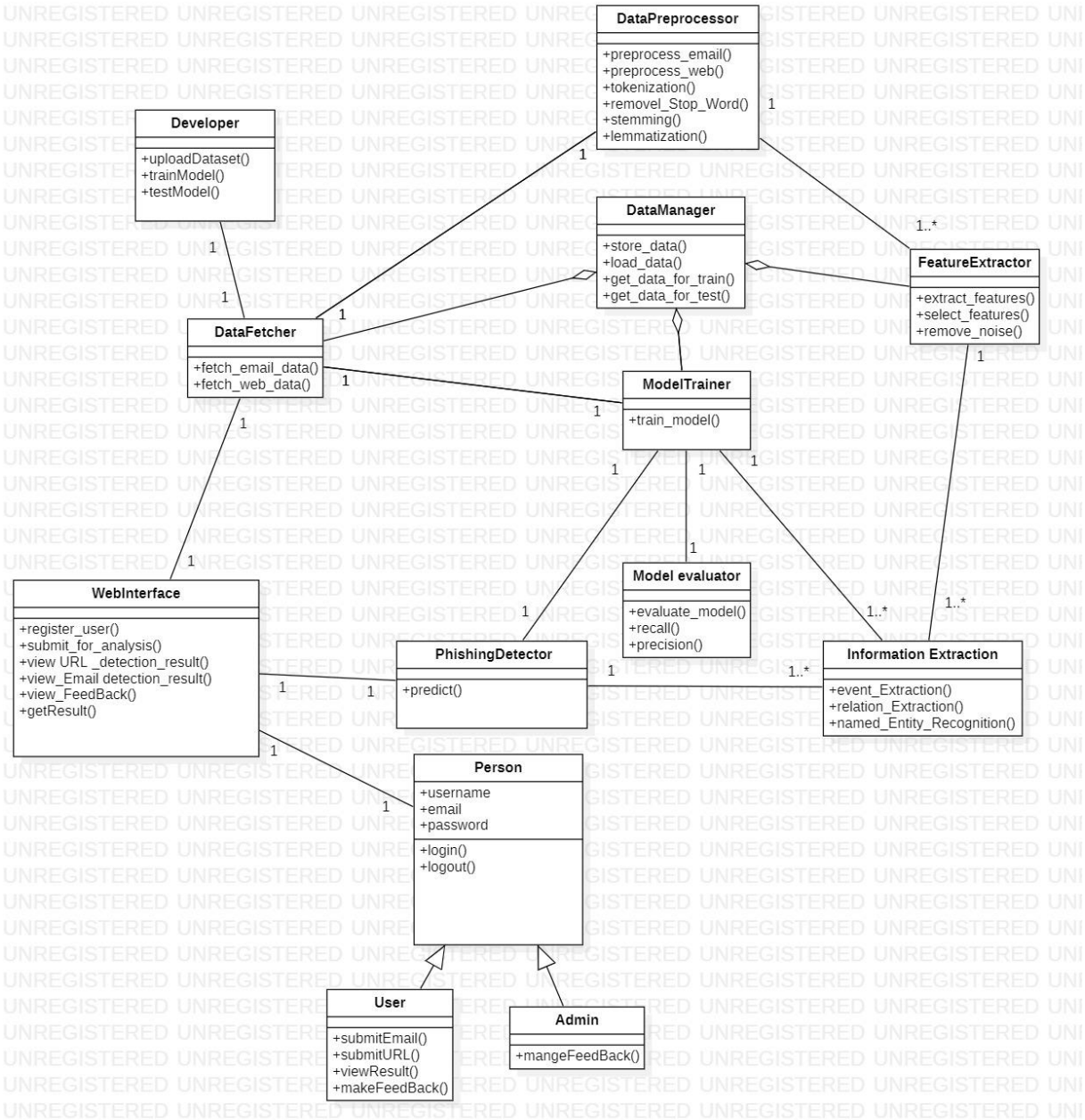


Figure 3-3 Class Diagram

As shows in the figure 3-3 The class diagram of the PhishWarde system, highlighting key classes and their interactions. Central classes, The User class represents the users of the PhishWarden system. Users can be individuals, organizations, or other entities that need to protect themselves from phishing attacks The WebInterface class provides a user interface for users to interact with the PhishWarden system. The User class and the WebInterface class are essential components of the PhishWarden phishing attack detection system. They allow users to interact with the system and benefit from its features , The Data Fetcher class is responsible for fetching data from external sources. The DataManager class is responsible for storing and

managing data, features, models, and other information. It aggregates all other classes in the system, meaning that it holds objects of all other classes. This allows the other components of the system to access the information they need without having to worry about managing it themselves. The Feature Extractor class is responsible for extracting relevant features from data. These features can be used to train machine learning models to detect phishing attacks. The Information Extraction class is responsible for extracting information from data. This information can be used to train machine learning models to detect phishing attacks. The Model Trainer class is responsible for training machine learning models on the extracted features. The trained models can then be used to classify emails and URLs as phishing or legitimate. The Model Evaluator class is responsible for evaluating the performance of trained machine learning models on a held-out test set. This helps to ensure that the trained models can generalize to new data and accurately detect phishing attacks. The WebInterface class provides a user interface for users to interact with the system. Users can use the WebInterface to submit emails and URLs for classification, view classification results, and provide feedback.

3.4.1 Non-functional Requirements

We asked multiple programmers on how we are going to successfully achieve and implement those non-functional requirements:

- To achieve performance

1- Optimized Algorithms: Develop and optimize machine learning algorithms for phishing detection to ensure fast and accurate predictions.

2- Communicating with machine learning locally

- Reliability

1- Error handling, to prevent potential errors.

- To achieve accuracy

1- Machine Learning Models: Train machine learning models with a diverse and representative dataset to enhance accuracy.

- To achieve ease of use

1- Friendly User Interface design

2- Responsive design

3.5 Summary

This chapter covers 3 sections. First are developmental methodologies inside it we described our software development process, we also used a table of Ai technology that will be used, and all the tools needed to accomplish our goal. Second is the requirement elicitation which contains functional, non-functional requirements and user/domain requirements. Third

section is about requirement specification containing the use case diagram, Chapter ends with two Implementation parts that are split into functional requirement which contains high level description and a description for every use case in the system and non-functional requirement that explains how we are going to implement it and how we add a class diagram based on system requirements and the use case.

The primary objective of this chapter was to conduct a thorough analysis of the system requirements. By gathering and documenting these requirements, we aimed to gain a comprehensive understanding of the functionalities and features the system must possess. Additionally, the chapter involved selecting the most suitable models for the system analysis, ensuring a structured and methodical approach to the development process, hence next are the System design Chapter that will focus on designing the system with future implementation capability in mind. The goal is to create a robust and scalable system architecture that not only addresses the current requirements but also accommodates potential future enhancements and updates. This forward-thinking design approach will contribute to the long-term sustainability and adaptability of the system.

Chapter 4. System Design

4.1 Introduction

This chapter documents the design phase steps. According to Summerville [32], the first step in design is to build the system architecture. Based on our readings of related existing systems in the literature, we designed the training and testing models [21] [25] and depicting the system component design which zooms in the class-design of each component. Following the component design is the data model design using Object-Oriented Data Modeling (OOD) techniques [32]. Finally, the interface design will take place to picture the system user interfaces and interactions.

4.2 Architecture Design

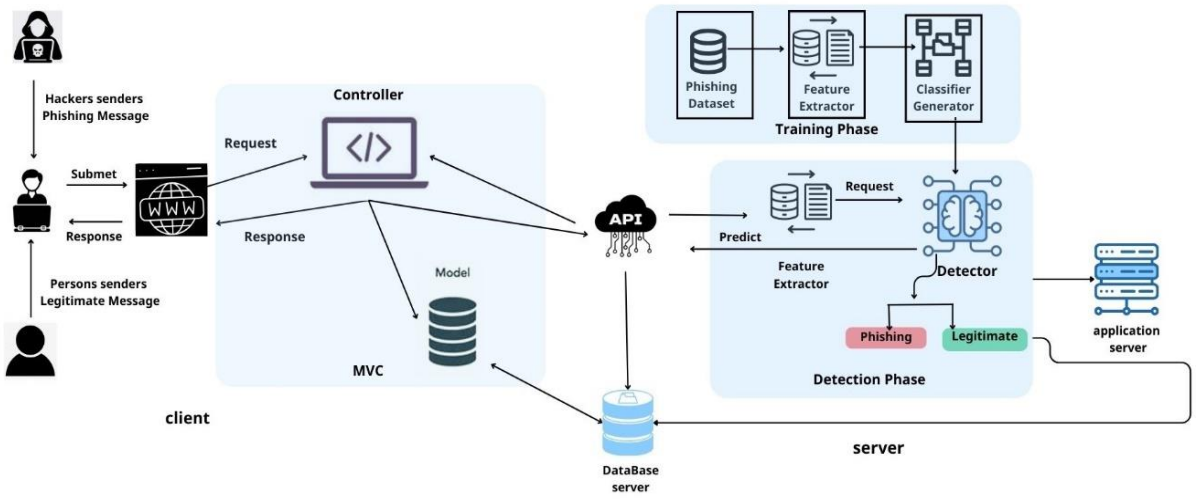


Figure 4-1 Architecture Design

The architecture diagram shows a high-level overview of a phishing attack detection system. The system uses a machine learning model to identify the features of phishing messages and predict whether a new message is a phishing message or a legitimate message. The architecture can be considered a client-server architecture. In a client-server architecture, there are two main types of nodes: clients and servers. Clients initiate requests for resources, while servers provide those resources. In the context of the phishing attack detection system, the clients would be the devices that send and receive emails, while the servers would be the machines that run the data collector, feature extractor, machine learning module, classifier, and alerting system.

The system has the following components:

Data Collector: This component collects data on phishing messages and legitimate messages from various sources, such as honeypots, spam traps, and public databases. **Feature Extractor:** This component extracts the features of a message, such as the sender's email address, the subject line, the body of the message, and the links in the message. **Machine Learning Module:** This component is trained on a dataset of phishing messages and legitimate messages to identify the features that are most likely to be found in phishing messages. **Classifier:** This component uses the output of the machine learning module to predict whether a new message is a phishing message or a legitimate message. **Alerting System:** This component alerts users to suspected phishing messages. **API:** The API is an interface that allows clients to interact with the machine learning model. **Database server:** The database server stores the dataset of labeled phishing and legitimate messages. It also stores the trained machine learning model.

The system works as follows: The data collector collects data on phishing messages and legitimate messages from various sources. The feature extractor extracts the features of each message. The machine learning module predicts whether each message is a phishing message or a legitimate message. The classifier uses the output of the machine learning module to predict whether a new message is a phishing message or a legitimate message. The alerting system alerts users to suspected phishing messages.

Our system employs the Model-View-Controller (MVC) architecture to facilitate seamless interaction between components: the web interface (View) for user interaction, the model layer (Model) for data management and business logic, and external APIs (Controller) that connect directly to machine learning (ML) models. The Controller component manages user input from the web interface, interacts with the Model for data operations, and communicates with APIs to access ML functionalities. This architecture ensures a structured approach to software development, enables efficient data processing, and leverages the power of ML models through API integration, enhancing the system's capabilities in data analysis and decision-making.

The client-server architecture is well-suited for the phishing attack detection system. It allows for the system to be distributed across multiple servers, which can improve performance and scalability. It also allows for the system to be easily updated, as new phishing techniques are discovered.

4.3 Object Oriented Design

In the Object-Oriented Design and Data Modeling section, we explore fundamental aspects of software engineering, focusing on Component Diagrams, Class Diagrams, Data Modeling, and User Interface Design. Our Component Diagrams provide architectural blueprints, illustrating system modularity and interconnections, while Class Diagrams detail object-oriented design elements. Data Modeling involves defining data structures and relationships, crucial for system functionality. Additionally, our emphasis on User Interface Design prioritizes user-centric interfaces.

In our view layer, we don't use traditional classes typical of object-oriented programming. Instead, we adopt a functional or component-based approach, structuring views with HTML elements.

4.3.1 System Component

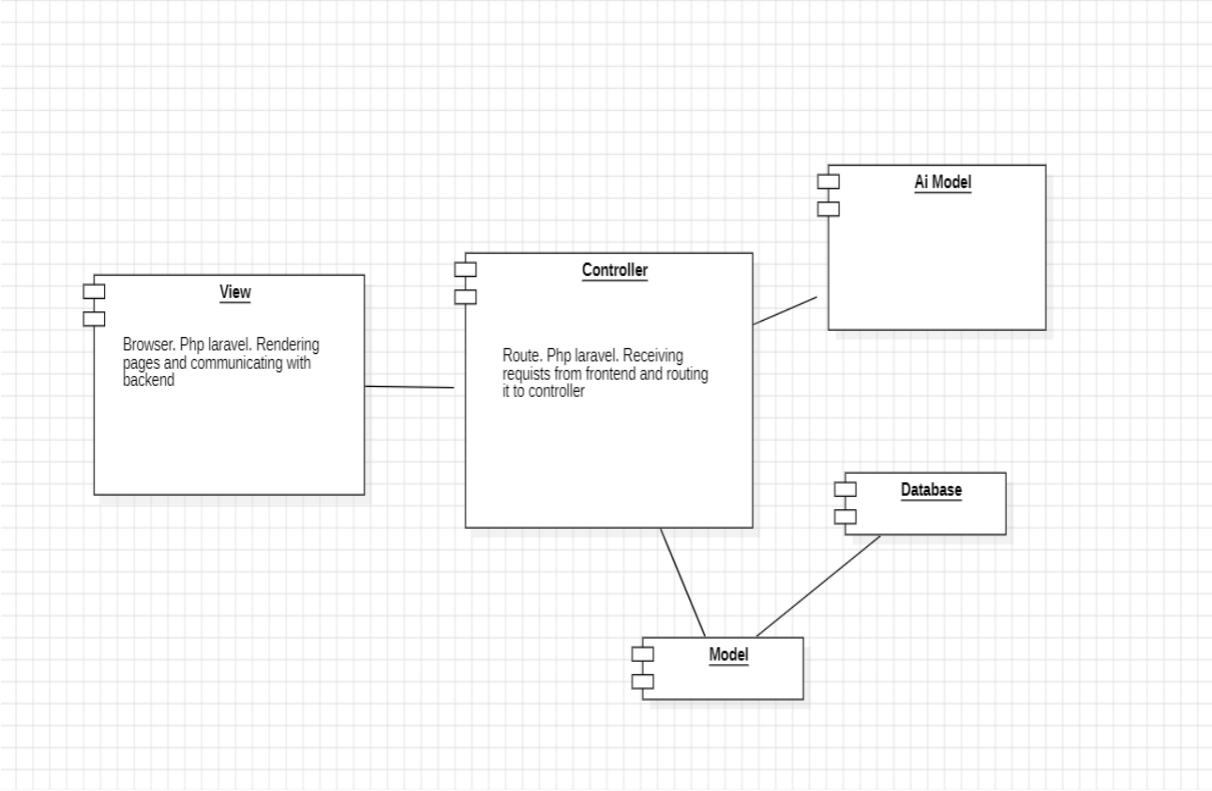


Figure 4-2 component diagram.

In this architecture shown above of our website, we employed a component-based design illustrated through a component diagram view, providing a structured representation of the system's components and their interactions. At the core of this architecture is the component controller, serving as a pivotal link between the user interface (UI) and the backend functionalities. This controller efficiently manages user inputs, orchestrates requests, and navigates data flow throughout the system.

Key components integrated into this architecture include the AI model and the database. The component controller interfaces with the AI model, harnessing its intelligent capabilities such as machine learning algorithms and natural language processing to enhance user experiences and provide the service expected. Simultaneously, the controller establishes a connection with the database, facilitating data storage, retrieval, and management essential for the website's functionality.

Furthermore, within the backend infrastructure, a structured relationship exists between the database and the model. The model, comprising data structures and access mechanisms, defines how information is organized and accessed within the application. The database, serving as the persistent storage layer, ensures data integrity and availability, reinforcing the reliability and efficiency of the system.

This component-based architecture delineates clear boundaries of responsibility, enabling modular development, easier maintenance, and scalability. Through this design approach, my website achieves a cohesive and efficient structure that optimizes performance and enhances user interactions.

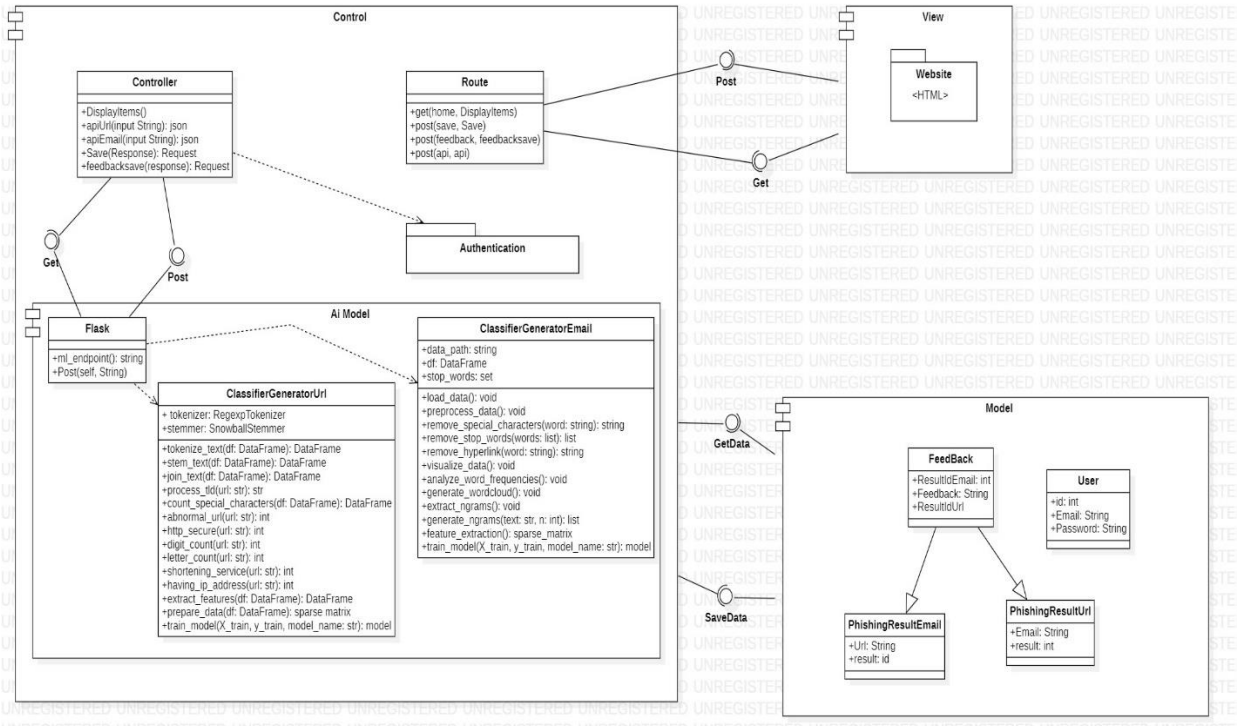


Figure 4-3 Website Class Diagram.

In the class diagram representing the architecture of my website, several key classes and their relationships define the system's structure and functionality. The View class encapsulates the visual elements of the website and collaborates with the Website class through an

association, enabling the rendering of pages which was written by HTML such as home, welcome, and app interfaces. The Route class, serving as the routing component, not only interacts with the View component for page rendering but also relies on the Controller class for request handling and backend logic execution. This relationship is denoted by a dependency between the Route class and the Controller class, ensuring seamless data flow and application control.

Moreover, the Route class incorporates authentication mechanisms, as indicated by its dependency on the authentication package, reinforcing the system's security measures. The Controller class, central to request management, maintains dependencies with both the authentication package for user validation and the Feedback and PhishingResult database models for data handling. Through methods such as displayitems() for presenting content, apiURL() and apiEmail for handling API request and receive results from the Machine learning models, save() for data storage to database, and feedbacksave() for processing user feedback on result that occurred, also it was connected via flask class which we use flask language to implement the connection, the Controller class orchestrates the system's functionalities effectively.

The Feedback class, representing user feedback data, includes attributes like result ID and feedback content, facilitating the storage and analysis of user sentiments. On the other hand, the PhishingResult class captures details related to phishing detection outcomes, featuring attributes such as URL and detection results, which are crucial for monitoring and addressing security threats.

The class diagram clarifies how different parts of the website's backend work together. It shows how we organize tasks, keep data safe, and ensure everything runs smoothly. This diagram highlights how we focus on breaking tasks into manageable parts, keeping data accurate, and making sure everything is secure, which are key principles guiding how we design and run our system.

The ClassifierGeneratorUrl class is designed to perform classification tasks on phishing websites using various natural language processing and machine learning techniques. It preprocesses website URLs and extracts features to train machine learning models for classification.

Attributes:

1. tokenizer: An instance of the RegexpTokenizer class from the NLTK library, used for tokenizing text.

2. stemmer: An instance of the SnowballStemmer class from the NLTK library, used for stemming words.

Methods:

1. **tokenize_text :** Tokenizes the text data in the DataFrame df, returning a DataFrame with tokenized text.
2. **stem_text :** Stems the tokenized text data in the DataFrame df, returning a DataFrame with stemmed text.
3. **join_text :** Joins the stemmed text in the DataFrame df to create a single text column, returning a DataFrame with the joined text.
4. **process_tld :** Extracts the top-level domain (TLD) from the input URL url, returning the TLD as a string.
5. **count_special_characters :** Counts the occurrences of special characters in the URLs in the DataFrame df, returning a DataFrame with the counts.
6. **abnormal_url :** Determines if the input URL url is abnormal, returning an integer indicator (1 for abnormal, 0 for normal).
7. **http_secure :** Determines if the input URL url uses the HTTPS protocol, returning an integer indicator (1 for secure, 0 for insecure).
8. **digit_count :** Counts the number of digits in the input URL url, returning the count as an integer.
9. **letter_count :** Counts the number of letters in the input URL url, returning the count as an integer.
10. **shortening_service :** Determines if the input URL url uses a URL shortening service, returning an integer indicator (1 for using a shortening service, 0 for not using).
11. **having_ip_address :** Determines if the input URL url contains an IP address, returning an integer indicator (1 for having an IP address, 0 for not having).
12. **extract_features :** Extracts features from the DataFrame df, incorporating the processed URL text and other extracted features, returning a DataFrame with the extracted features.
13. **prepare_data :** Prepares the DataFrame df for model training, converting it into a sparse matrix format suitable for machine learning models.
14. **train_model :** Trains a machine learning model using the features X_train and labels y_train, where model_name specifies the type of model to train. Returns the trained model.

4.4 Data Modeling

Data modeling is a fundamental aspect of database design and development, serving as a blueprint for structuring and organizing data within a system. It involves creating a conceptual representation of the data, defining its structure, relationships, constraints, and rules that govern how data is stored, accessed, and manipulated.

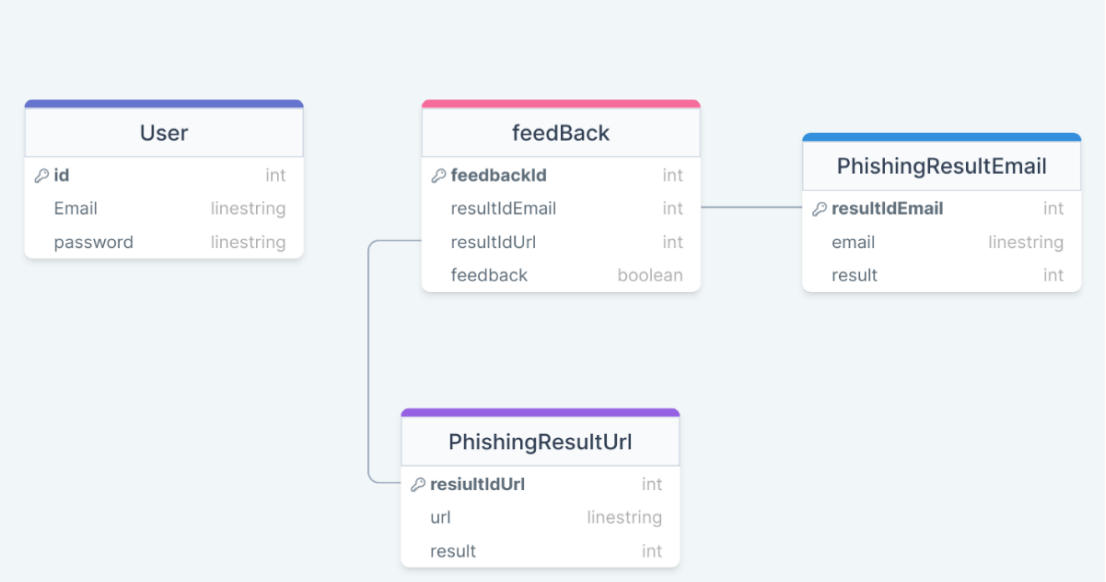


Figure 4-4 Data Modeling

User Table: This table is designed to store user information. It includes columns such as id (primary key), email, and password. The id column uniquely identifies each user, while email and password store the user's email address and encrypted password, respectively.

Feedback Table: This table is used to collect feedback related to phishing attempts. It contains columns like feedbackid (primary key), resultIdEmail, and ResultIdUrl. The feedbackid uniquely identifies each feedback entry. The resultIdEmail and ResultIdUrl columns are foreign keys that establish relationships with the PhishingResultEmail and PhishingResultUrl tables, respectively. These relationships link feedback entries to specific phishing results based on email or URL.

PhishingResultUrl Table: This table stores information about URL-based phishing attempts. It includes columns like ResultIdUrl (primary key), url, and result. The ResultIdUrl column uniquely identifies each URL-based phishing result. The url column stores the URLs involved in phishing attempts, while the result column indicates the outcome of the phishing attempt.

PhishingResultEmail Table: This table is similar to the PhishingResultUrl table but focuses on email-based phishing attempts. It includes columns such as resultIdEmail (primary

key), Email, and result. The resultIdEmail column serves as the primary key, uniquely identifying each email-based phishing result. The Email column stores the Message involved in phishing attempts, while the result column indicates the outcome of the email-based phishing attempt.

The relationships between these tables are crucial for organizing and analyzing data effectively. The Feedback table connects user feedback to specific phishing results stored in the PhishingResultUrl and PhishingResultEmail tables, allowing for comprehensive analysis of phishing attempts and their impact on users.

Data Sets:

This dataset, sourced from Kaggle in CSV format, contains 549,346 rows of URL entries paired with corresponding labels indicating whether each URL is categorized as 'bad' (potentially phishing) or 'good' (legitimate), providing valuable data for training models to classify URLs based on their potential security risks.

The second dataset, sourced from Kaggle in CSV format, contains 83,448 rows of email entries paired with corresponding labels indicating whether each URL is categorized as '0' (potentially phishing) or '1' (legitimate), providing valuable data for training models to classify emails based on their potential security risks.

The first dataset was divided into 450,000 rows, distributed between legitimate and phishing websites at a specific ratio. After conducting several experiments in model building, where the dataset was further divided into 359,329 rows for legitimate websites and 90,671 rows for phishing websites, we achieved the highest accuracy rate. We initially balanced the data between the two classifications, gradually increasing the number of rows in each iteration of model training and evaluation. We observed that deviating from the 450,000-row dataset size resulted in a decrease in accuracy.

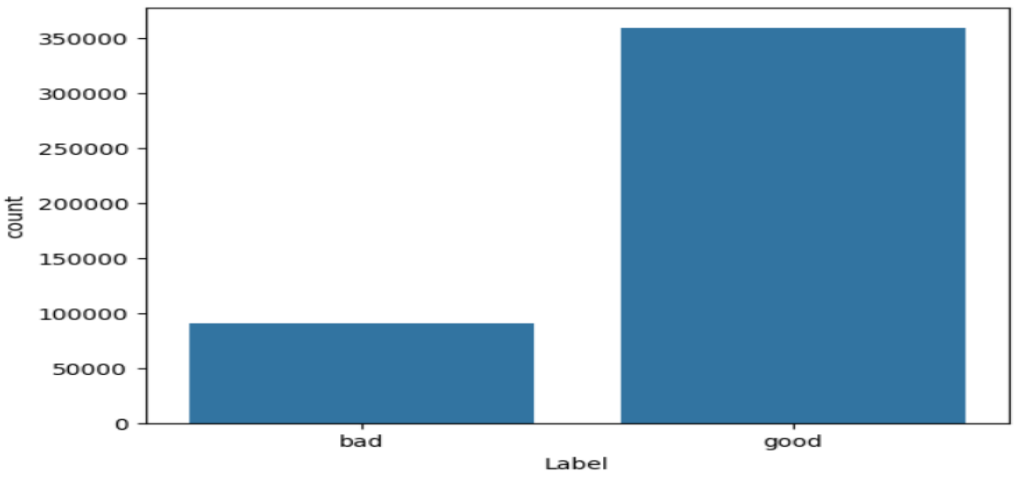


Figure 4-5 a bar chart shows the frequency of phishing and non-phishing URLs.

Graphical representation illustrating the percentage distribution of the dataset:

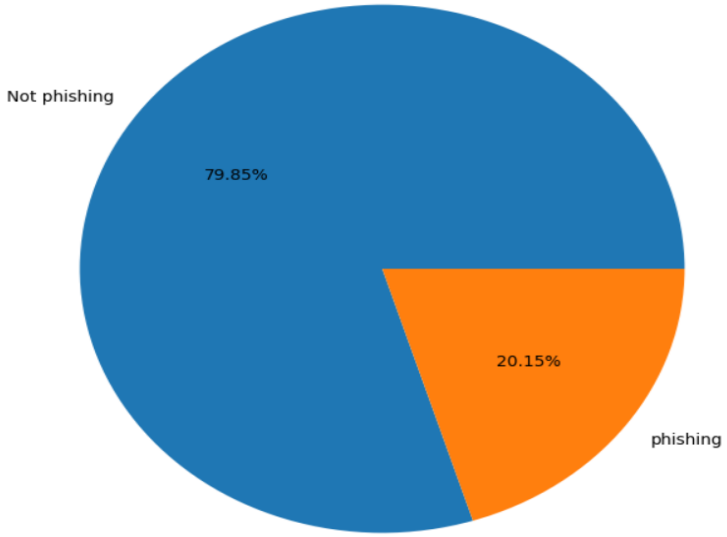


Figure 4-6 pie chart that shows the distribution of phishing and non-phishing URLs.

This is a random simple from dataset:

	URL	Label
121103	iitp.org.br/invel/wp-includes/quotationfile/do...	bad
41273	9d345009-a-62cb3a1a-s-sites.googlegroups.com/s...	bad
260108	wn.com/Bishop_of_Oxford	good
407350	nutsie.com/album/Imaginary%20Voyage/2804558	good
89153	www.crsn.com/corrosion/	good

Figure 4-7 a sample of URLs dataset.

The second dataset consisted of 83,448 rows, categorized between legitimate and phishing emails at a specific ratio. It was divided into 39,538 rows for legitimate emails and 43,910 rows for phishing emails.

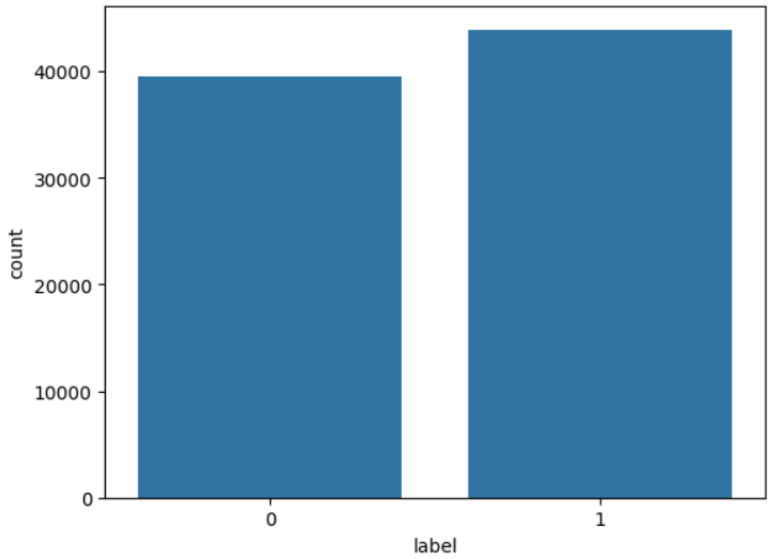


Figure 4-8 a bar chart shows the frequency of phishing and non-phishing Emils.

Graphical representation illustrating the percentage distribution of the dataset:

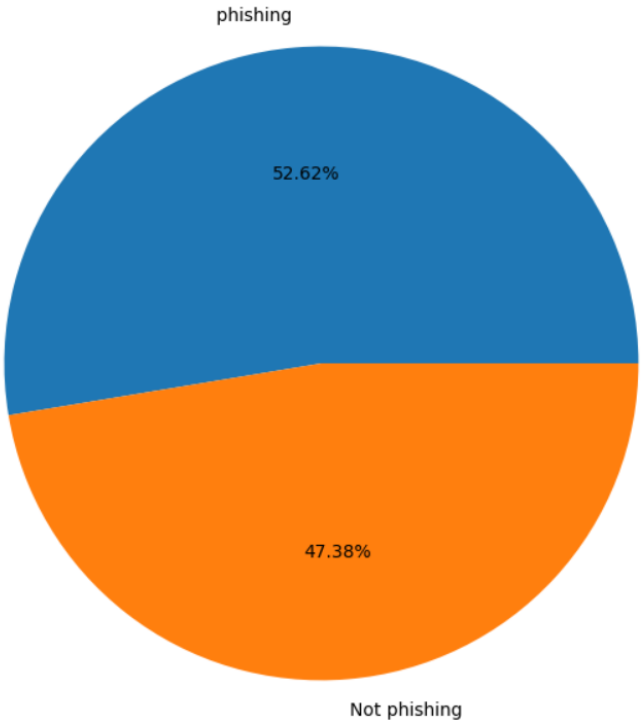


Figure 4-9 pie chart that shows the distribution of phishing and non-phishing Emails.

This is a random simple from dataset:

	label	text
28613	1	oem software throw packing case leave cd dvd u...
37840	0	attached please find the invitation for the la...
456	0	all as an addition to my earlier posting i've ...
58950	0	could all of you please give access to your ca...
14498	1	you are going to be amazed at how many benefit...

Figure 4-10 a sample of URLs dataset.

4.5 User Interface Design

User interface design focuses on what users may need to do and makes sure that it contains elements that are easy to access, understand, and use. The user interface combines concepts from visual and interaction design.

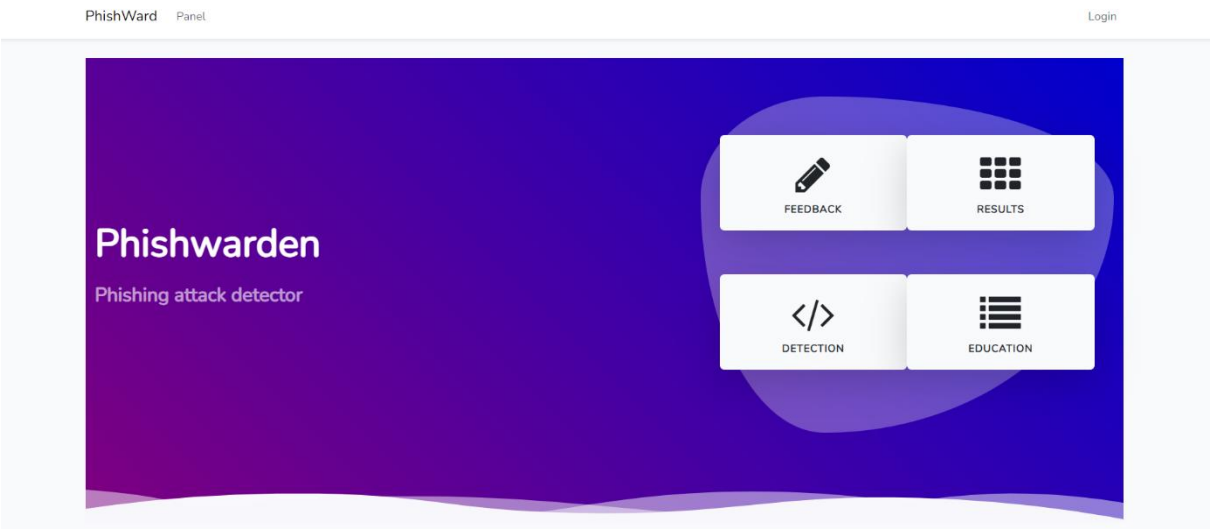


Figure 4-11 Main service page.

The main page of the Phishing Attack Detection website, as depicted in the figure, provides users with an overview of accessible sections for their benefit. The navigation bar features the project's name and two options: 'Panel' for the admin dashboard, granting access to sensitive data, and 'Login' for users to gain website privileges.

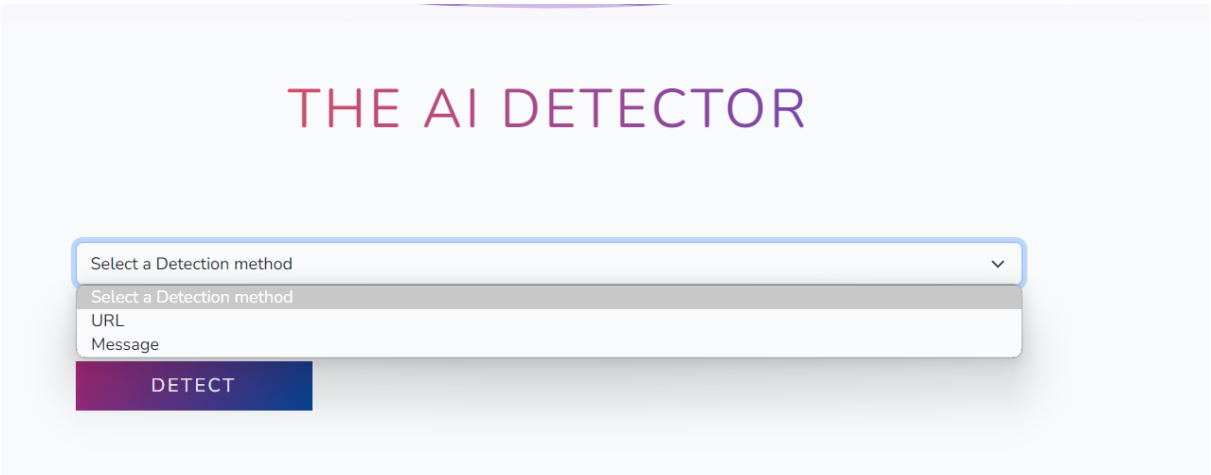


Figure 4-12 Detection section - service option.

In the AI DETECTOR section, users can interact with the main service, which is designed to facilitate their engagement. The name "AI DETECTOR" is self-explanatory, signaling to users that they have reached the desired service. This section provides users with an input option to select between 'URL' for running the URL ML model and 'Message' for running the Message ML model, enabling them to interact accordingly.

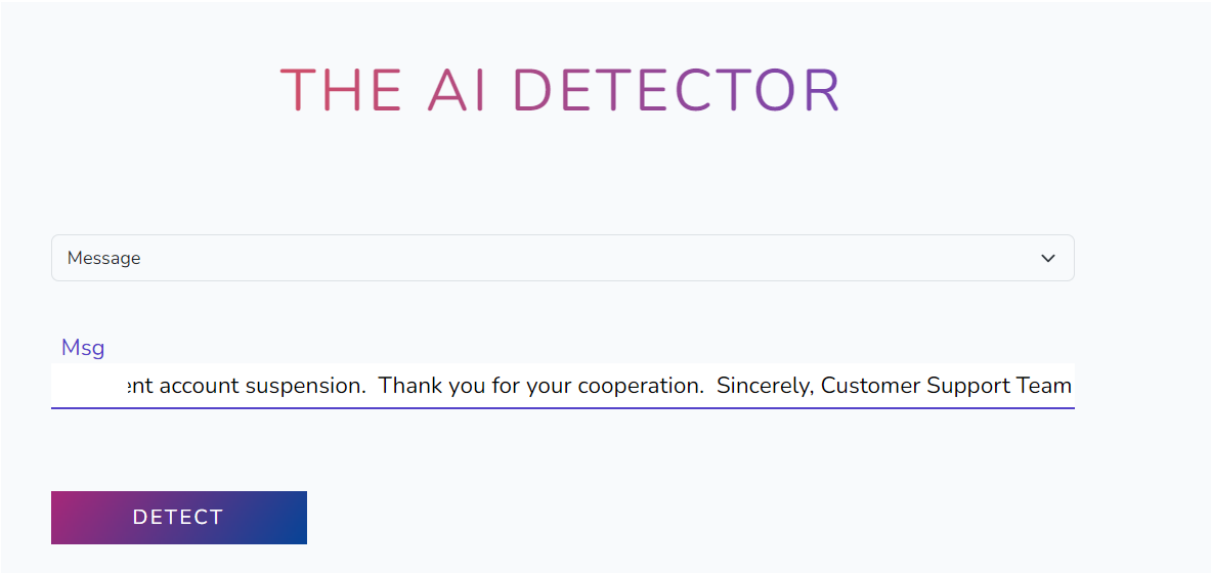


Figure 4-13 Detection section - input.

Once the user chooses the 'Message' option, they are presented with an input field to type the message they want to detect or are suspicious about. This input is then processed by the ML Model, allowing for interaction based on the user's input.

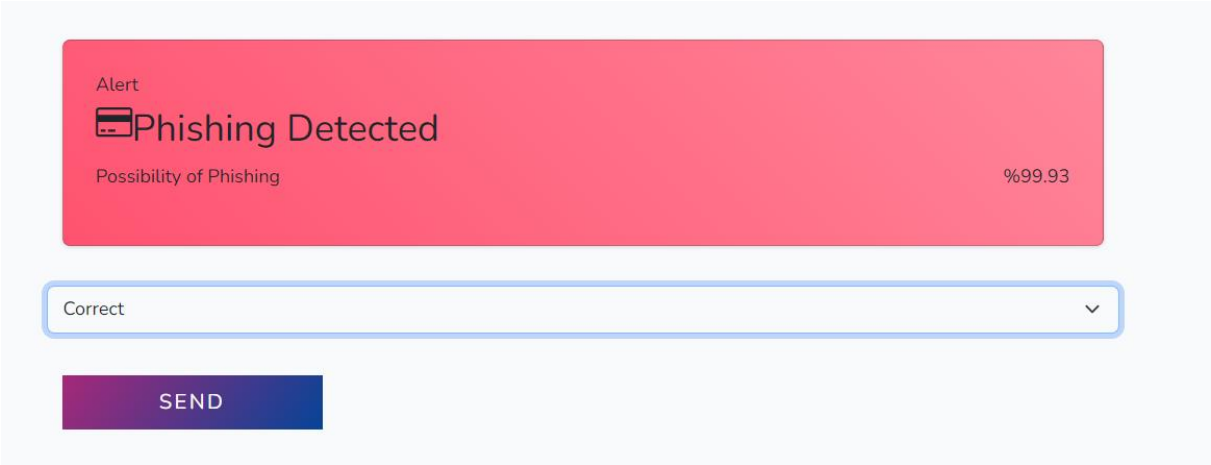


Figure 4-14 Detection section - Phishing result.

Upon receiving the result from the system, which indicates a phishing attack attempt with a probability of 99.93% according to the ML Model, the user is presented with an option to provide feedback. They can choose between 'Correct' or 'Not Correct' to help improve the ML Model for future enhancements.

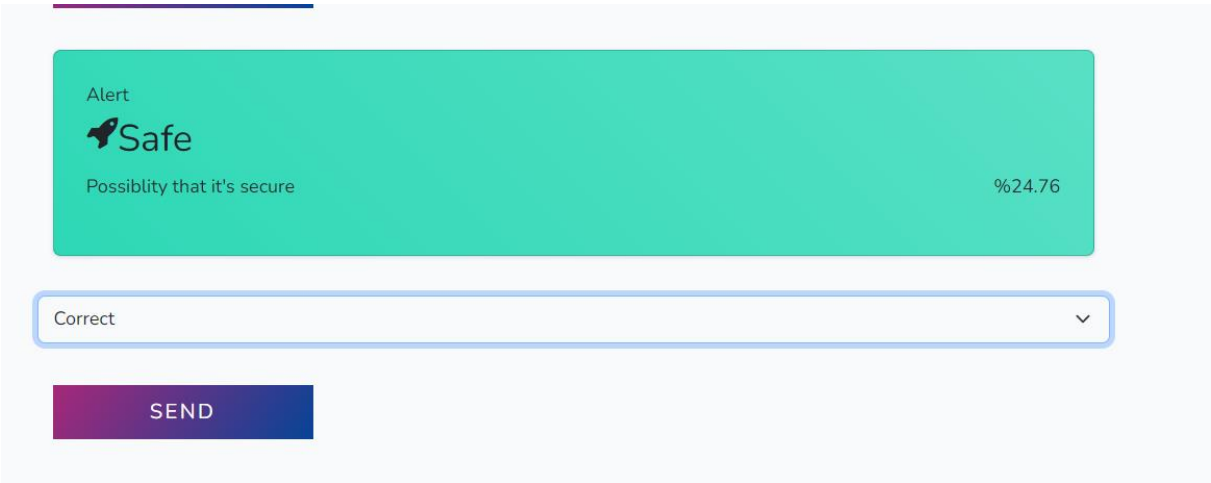


Figure 4-15 Detection section - Safe result.

The user's system incorporates an ML model to test user inputs. When the model determines that the input is 'Safe,' it indicates that it is not a phishing attempt. The probability of this being true is 24.76%, as calculated by the ML Model.

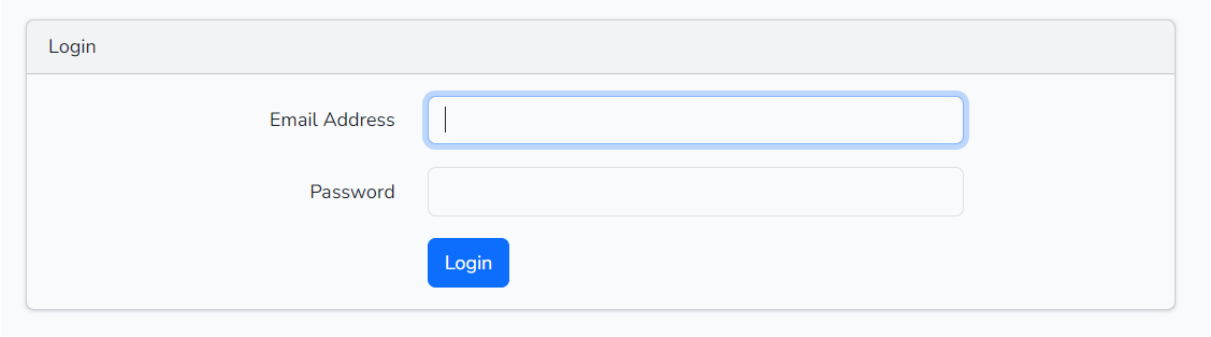


Figure 4-16 Login page.

For the login page, users can input their registered email and password to gain admin privileges.

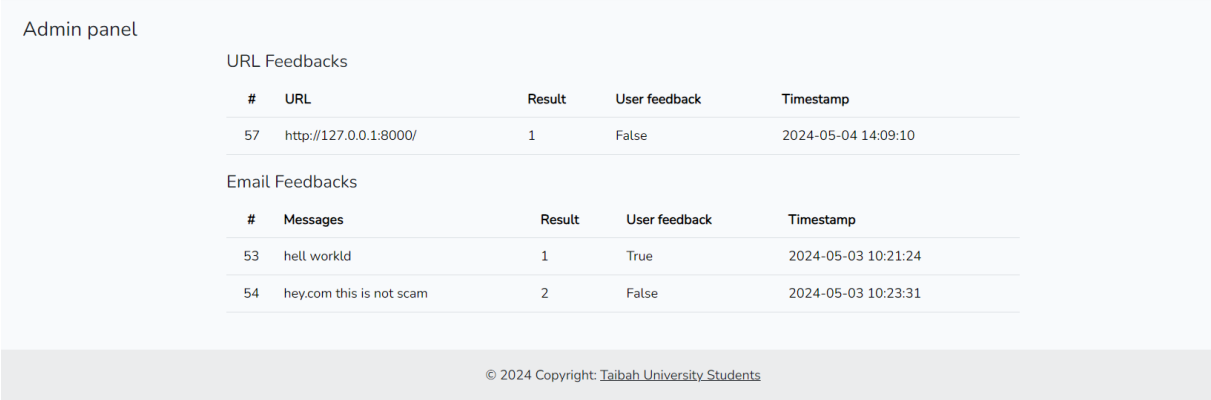


Figure 4-17 Admin panel.

In the system, there's an "Admin panel" accessible through the "Panel" navigation bar. Within this panel, there are two feedback tables: one for "URL Feedback," capturing user interactions with the website and the corresponding URL detection by the ML model, and another for "Email Feedback," reflecting user interactions and message detection by the ML model.

4.6 Summary

In this chapter, we detailed the architectural design of our system. Moreover, we developed a prototype for the website's user interface to better understand its design. Next chapter we will discuss the conclusion and lessons learnt.

The primary objective of this chapter was to design the system with future implementation capability in mind. Through careful consideration of architecture, scalability, and modularity, we aimed to create a system design that not only meets the current requirements but also allows for seamless integration of future enhancements. This forward-looking approach is essential for ensuring the longevity and adaptability of the developed system, hence next are Conclusion chapters which will involve listing the lessons learned from working as a team in this project. Reflecting on our collaborative experiences, challenges faced, and successful strategies employed, the objective is to extract valuable insights that can be applied to improve teamwork dynamics and project management in future endeavors.

Chapter 5. System Implementation

5.1 Introduction

In this chapter, we will discuss the final process of the project development and actual implementation and we will mention the tools and languages that were used in the system's implementation. Detailed explanation of how the discussion on system design relates to the actual implementation. Also, we will explain the most important codes in the system, and then we will test the system to ensure that its components are well integrated such as test cases, unit testing . We will present the results and logical conclusions based on the facts.

5.2 Tools and Languages

In the following table 5-1, we clarify some of the tools and languages that we used throughout the project to help us get with the final application.

Table 5-1 Tools and languages.

Icon	Name	Description	Use
	Python	Python is a high-level, versatile programming language known for its simplicity and readability, favored for web development, data analysis, AI, automation	Writing AI models using libraries like scikit-learn, NumPy, and Pandas.
	Google Colab	Colab is a free Google service for AI developers, offering cloud-based Python notebooks with GPU support, facilitating collaborative AI model development and training.	Writing and executing Python code
	Visual Studio	Visual Studio is a powerful integrated development environment (IDE) by Microsoft for coding, debugging, and deploying software across various platforms.	Writing and executing Python code and HTML,CSS,JavaScript
	HTML	HTML is a markup language used to create and structure web pages, defining the content and layout elements for browsers.	Writing the structure and design web pages.
	CSS	CSS is a styling language that enhances HTML, allowing developers to control the appearance and layout of web pages.	Writing the style and format HTML elements on web pages.
	Flask	Flask is a Python web framework suitable for building APIs, offering simplicity, flexibility, and scalability for developing backend services.	Use it for APIs and connect between backend and frontend
	JavaScript	JavaScript is a versatile programming language primarily used for client-side web development, enabling interactive features and dynamic content on websites.	Use it for front-end and back-end development.
	MySql	MySQL is an open-source relational database management system (RDBMS) known for its speed, reliability, and ease of use in web applications.	Create database for storing and organizing, and managing data in various applications.
	PHP laravel	Laravel is a popular PHP framework known for its elegant syntax, MVC architecture, and rich ecosystem, making web development faster and more efficient.	Use it for website creation with its powerful tools and clear syntax, ensuring robust, scalable websites that are easy to develop and manage.
	Bootstrap	Bootstrap is a popular front-end framework for developing responsive and mobile-first websites. It includes CSS- and JavaScript-based design templates for faster and easier web development.	Use it for building good-looking websites that work on all devices. Bootstrap has pre-made parts like boxes, buttons, and menus, making it easier to create a site that looks good

5.3 System implementation

In this section, we delve into the critical phase of system implementation, were theoretical design transforms into tangible reality. This stage is pivotal in actualizing the envisioned functionalities of the system.

5.3.1 Reuse

We center our attention on the fundamental concept of reuse, exploring how existing components, modules, or frameworks can be leveraged to streamline development efforts and enhance efficiency.

Authentication:

```
14
15
16      public function up(): void
17      {
18          Schema::create('users', function (Blueprint $table) {
19              $table->id();
20              $table->string('name');
21              $table->string('email')->unique();
22              $table->timestamp('email_verified_at')->nullable();
23              $table->string('password');
24              $table->rememberToken();
25              $table->timestamps();
26          });
27      }
28
```

Figure 5-1 Database users table migration.

It defines a migration file that creates a database table named 'users' with specific columns and properties. The 'id' column is set as the primary key, 'name' and 'email' columns are defined as strings, with 'email' being unique to each user. The 'email_verified_at' column is for timestamp data related to email verification, 'password' column stores user passwords securely, and 'rememberToken' is for user authentication. Lastly, 'timestamps()' automatically adds 'created_at' and 'updated_at' columns to track when each user record is created and updated. This code is part of setting up a user authentication system in a web application.

```php
43
    0 references | 0 overrides
44  protected function validator(array $data)
45  {
46      return Validator::make($data, [
47          'name' => ['required', 'string', 'max:255'],
48          'email' => ['required', 'string', 'email', 'max:255', 'unique:users'],
49          'password' => ['required', 'string', 'min:8', 'confirmed'],
50      ]);
51  }
52
    0 references | 0 overrides
53  protected function create(array $data)
54  {
55      return User::create([
56          'name' => $data['name'],
57          'email' => $data['email'],
58          'password' => Hash::make($data['password']),
59      ]);
60  }
```

Figure 5-2 Validating and creating a user instance.

The first method is a validator in PHP, commonly used in Laravel applications to validate incoming data before processing it. It takes an array of data as input and uses Laravel's Validator class to define validation rules for each field. For example, it ensures that 'name' is required, a string, and not longer than 255 characters, 'email' is required, a valid email format, unique among users, and not longer than 255 characters, and 'password' is required, a string with a minimum length of 8 characters, and matches the 'password_confirmation' field.

The second method is a data creation function also used in Laravel applications. It takes an array of data and creates a new user record in the database using the User model. It assigns values for 'name', 'email', and 'password', where 'name' and 'email' are taken directly from the input data, and 'password' is hashed using Laravel's Hash::make function for security. This method is typically called after data validation to ensure that only valid data is used to create new user records.

```php
    0 references | 0 overrides
public function definition(): array
{
    return [
        'name' => fake()->name(),
        'email' => fake()->unique()->safeEmail(),
        'email_verified_at' => now(),
        'password' => static::$password ??= Hash::make('password'),
        'remember_token' => Str::random(10),
    ];
}
```

Figure 5-3 Fake data user generator.

a method that returns an array representing a fake user's data. It uses Laravel's Faker library to generate realistic fake data. The 'name' key generates a fake name, 'email' generates a unique and safe email address, 'email_verified_at' sets the current timestamp for email verification, 'password' generates a hashed password using Laravel's Hash::make function (or a default password if already set), and 'remember_token' generates a random 10-character string for authentication purposes. This method is often used in testing environments to create mock user data for scenarios like automated testing or seeding databases with sample data.

5.3.2 Mapping Design to Implementation

In this part, we take a closer look at how we are mapping design plans into real, working parts of the system. We focus on making sure that what we've imagined matches up with what we're building. By breaking down this process, we'll see how we take ideas on paper and turn them into tangible parts of the system that work. This section stresses the importance of careful planning and getting the details right to make sure our ideas become reality.

1- User Interface

As it shown in figure 4-12 in (Chapter 4) where it demonstrates the interface of the Ai detection service, we would like to spot the lights on the implementation we did on this current section.

```
121
122         <section class="get-in-touch">
123             <h1 class="title">The Ai Detector</h1>
124 <form action="{{ route('save') }}" method="POST" class="contact-form row" id="phishingForm">
125     @csrf
126     <div class="form-field col-lg-12">
127         <select name="detect" id="detectOption" class="form-select" aria-label="Default select example">
128             <option selected disabled>Select a Detection method</option>
129             <option value="url">URL</option>
130             <option value="message">Message</option>
131         </select>
132     </div>
133     <div class="form-field col-lg-12" id="urlInput" style="display: none;">
134         <input id="url" name="url" class="input-text js-input" type="text">
135         <label class="label" for="url">URL</label>
136     </div>
137     <div class="form-field col-lg-12" id="messageInput" style="display: none;">
138         <input id="message" name="message" class="input-text js-input" type="text">
139         <label class="label" for="message">Msg</label>
140     </div>
141     <div class="form-field col-lg-12">
142         <button class="submit-btn" type="submit" value="Submit">Detect</button>
143     </div>
144 </form>
```

Figure 5-4 Implementation of Ai detector section.

Firstly, we segment the selection process into a method, which is then integrated into a form for transmission to the controller through routing classes and methods. Next, the user inputs the required data and, upon completion, triggers the submission process by pressing the submit button, which signifies the data transmission via the post method.

2 - database

In Figure 4-4 in (Chapter 4), focusing on data modeling, we'll delve into the practical implementation of this design, culminating in the creation of dedicated database tables for each component.

```
16      public function up(): void
17      {
18          Schema::create('phishing_result_urls', function (Blueprint $table)
19              $table->id();
20              $table->string('url');
21              $table->boolean('result');
22              $table->timestamps();
23          });
24      }
```

Figure 5-5 Phishing result urls Database Table.

```
16      public function up(): void
17      {
18          Schema::create('feed_backs', function (Blueprint $table) {
19              $table->id();
20              $table->int('resultIdEmail');
21              $table->int('resultIdUrl');
22              $table->boolean('feedback');
23              $table->timestamps();
24          });
25      }
```

Figure 5-6 Feedback Database Table.

```
14      public function up(): void
15      {
16
17          Schema::create('phishing_results_email', function (Blueprint $table) {
18              $table->id();
19              $table->string('url');
20              $table->boolean('result');
21              $table->timestamps();
22          });
23      }
```

Figure 5-7 Phishing Result Email Database Table.

In both figure 5-5 and figure 5-7, these tables serve a similar function, albeit for distinct purposes and with different schemas. Using schema: create, a method inherited from the DB class, we initiate the creation of a new database table. The $table variable acts as a blueprint, detailing the necessary information for the system. For instance, id() generates a unique identifier for each row, string("url") captures the user-input URL for detection, Boolean("result") signifies the ML Model's output, and timestamp() records the initiation time.

In Figure 5-6, we observe tables similar to those previously mentioned, with the addition of two columns: int('resultidEmail'), representing the foreign key linking to the 'id' in the PhishingResultEmail table, and int('resultidUrl'), indicating the foreign key linked to the 'id' in the PhishingResultUrl table.

5.3.3 Main important codes

Finally, we wrap up by highlighting the most important bits of code. These are the key snippets or segments that make the system work smoothly and reliably. By focusing on these vital components, we aim to give you a clear picture of how the system functions, making it easier for you to understand and work with it.

1. **Feature Engineering Techniques:**

   - Introduce new features derived from existing ones that might improve the model's performance.
   - Examples include creating binary indicators for specific patterns (e.g., presence of IP address in URL), and having Shortening Service, extracting the primary domain from a given URL, aggregating features (e.g., digit and letter count).

```
[ ] tokenizer = RegexpTokenizer(r'[A-Za-z]+')
```

```
tokenizer.tokenize(df.URL[0])
```

```
[ ]
    print('Getting words tokenized ...')
    t0= time.perf_counter()
    df['text_tokenized'] = df.URL.map(lambda t: tokenizer.tokenize(t))
    t1 = time.perf_counter() - t0
    print('Time taken',t1 ,'sec')
```

```
Getting words tokenized ...
Time taken 2.7034836309999832 sec
<ipython-input-15-721228981218>:3: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df['text_tokenized'] = df.URL.map(lambda t: tokenizer.tokenize(t))
```

```
[ ] stemmer = SnowballStemmer("english")
```

```
[ ] print('Getting words stemmed ...')
    t0= time.perf_counter()
    df['text_stemmed'] = df['text_tokenized'].map(lambda l: [stemmer.stem(word) for word in l])
    t1= time.perf_counter() - t0
    print('Time taken',t1 ,'sec')
```

```
Getting words stemmed ...
Time taken 62.736923658999984 sec
<ipython-input-18-ea33128d07f1>:3: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df['text_stemmed'] = df['text_tokenized'].map(lambda l: [stemmer.stem(word) for word in l])
```

```
[ ] df.sample(5)
```

	URL	Label	text_tokenized	text_stemmed
157510	changerecruitmentgroup.net/	good	[changerecruitmentgroup, net]	[changerecruitmentgroup, net]
442690	tenntrips.com/planner/listings/citypage.php?na...	good	[tenntrips, com, planner, listings, citypage, ...	[tenntrip, com, planner, list, citypag, php, n...
245770	testaae.greenwood.com/doc_print.aspx?fileID=C8...	good	[testaae, greenwood, com, doc, print, aspx, fi...	[testaa, greenwood, com, doc, print, aspx, fil...
131695	helplinesluck.tripod.com/	bad	[helplinesluck, tripod, com]	[helplinesluck, tripod, com]
247032	theofficialpageof.com/russell-hornsby/	good	[theofficialpageof, com, russell, hornsby]	[theofficialpageof, com, russel, hornsbi]

```
df['url_len'] = df['URL'].apply(lambda x: len(str(x)))
```

```
<ipython-input-25-211b0490bc4c>:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df['url_len'] = df['URL'].apply(lambda x: len(str(x)))
```

```
def process_tld(url):
    try:
        res = get_tld(url, as_object = True, fail_silently=False,fix_protocol=True)
        pri_domain= res.parsed_url.netloc
    except :
        pri_domain= None
    return pri_domain
```

```
df['domain'] = df['URL'].apply(lambda i: process_tld(i))
```

```
<ipython-input-27-10e657bfb1c5>:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df['domain'] = df['URL'].apply(lambda i: process_tld(i))
```

```
def digit_count(url):
    digits = 0
    for i in url:
        if i.isnumeric():
            digits = digits + 1
    return digits
```

```
df['digits']= df['URL'].apply(lambda i: digit_count(i))
```

```
<ipython-input-38-d4076427c6cc>:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df['digits']= df['URL'].apply(lambda i: digit_count(i))
```

```
def letter_count(url):
    letters = 0
    for i in url:
        if i.isalpha():
            letters = letters + 1
    return letters
```

```
df['letters']= df['URL'].apply(lambda i: letter_count(i))
```

```
def Shortining_Service(url):
    match = re.search('bit\.ly|goo\.gl|shorte\.st|go2l\.ink|x\.co|ow\.ly|t\.co|tinyurl|tr\.im|is\.gd|cli\.gs|'
                      'yfrog\.com|migre\.me|ff\.im|tiny\.cc|url4\.eu|twit\.ac|su\.pr|twurl\.nl|snipurl\.com|'
                      'short\.to|BudURL\.com|ping\.fm|post\.ly|Just\.as|bkite\.com|snipr\.com|fic\.kr|loopt\.us|'
                      'doiop\.com|short\.ie|kl\.am|wp\.me|rubyurl\.com|om\.ly|to\.ly|bit\.do|t\.co|lnkd\.in|'
                      'db\.tt|qr\.ae|adf\.ly|goo\.gl|bitly\.com|cur\.lv|tinyurl\.com|ow\.ly|bit\.ly|ity\.im|'
                      'q\.gs|is\.gd|po\.st|bc\.vc|twitthis\.com|u\.to|j\.mp|buzurl\.com|cutt\.us|u\.bb|yourls\.org|'
                      'x\.co|prettylinkpro\.com|scrnch\.me|filoops\.info|vzturl\.com|qr\.net|1url\.com|tweez\.me|v\.gd|'
                      'tr\.im|link\.zip\.net',
                      url)
    if match:
        return 1
    else:
        return 0
```

```
df['Shortining_Service'] = df['URL'].apply(lambda x: Shortining_Service(x))
```

Figure 5-8 Feature Engineering Techniques

1. Tokenization:

- The RegexpTokenizer from the NLTK library is used to tokenize the text. It splits the text into words using a regular expression pattern [A-Za-z]+, which matches one or more alphabetical characters.
- The tokenizer is applied to each URL in the DataFrame using a lambda function, and the tokenized words are stored in a new column named 'text_tokenized'.

2. **Stemming:**
    - The SnowballStemmer from the NLTK library is used for stemming, which reduces words to their base or root form.
    - Stemming is applied to the tokenized words in the 'text_tokenized' column using a lambda function, and the stemmed words are stored in a new column named 'text_stemmed'.
3. **URL Length Calculation:**
    - This method calculates the length of each URL and stores it in a new column named 'url_len' in the DataFrame. It uses a lambda function to apply the len() function to each URL string.
4. **Extracting Primary Domain:**
    - The process_tld function extracts the primary domain from each URL using the 'get_tld' function. It handles exceptions and returns the primary domain or None if extraction fails. The primary domain is stored in a new column named 'domain' in the DataFrame using a lambda function.
5. **Counting Digits in URL:**
    - The digit_count function counts the number of digits in each URL and stores the count in a new column named 'digits' in the DataFrame.
6. **Counting Letters in URL:**
    - The letter_count function counts the number of letters (alphabets) in each URL and stores the count in a new column named 'letters' in the DataFrame.
7. **Detecting Shortening Services:**
    - The Shortining_Service function checks if the URL is shortened using popular URL shortening services. If a match is found, it returns 1; otherwise, it returns 0. This function is applied to each URL in the DataFrame to create a new column named 'Shortining_Service'.

Overall, these feature engineering techniques transform raw text data (URLs) into a structured format that is more amenable to further analysis or modeling. By tokenizing, stemming, and rejoining the words, the data is prepared for tasks such as sentiment analysis, text classification, or other natural language processing (NLP) tasks. This preprocessing step is crucial for extracting meaningful features and improving the performance of machine learning algorithms applied to text data.

2. **Controller :**

```
1 reference | 0 overrides
24      public function apiURL($inputText)
25      {
26          $response = Http::post('http://localhost:5001/mlurl', [
27              'input_text' => $inputText
28          ]);
29
30          return $response->json();
31
32      }
33
        1 reference | 0 overrides
34      public function apiEmail($inputText)
35  {
36      $response = Http::post('http://localhost:5001/ml', [
37          'input_text' => $inputText,
38      ]);
39
40      return $response->json();
41  }
```

Figure 5-9 Two apiURL functions interact with ML Model

apiURL($inputText) and apiEmail($inputText), which interact with machine learning (ML) models for URL and email analysis, respectively. The apiURL function sends an HTTP POST request to a local server at 'http://localhost:5001/mlurl', passing the input text for URL analysis. Similarly, the apiEmail function sends an HTTP POST request to 'http://localhost:5001/ml' with the input text for email analysis. Both functions expect a response in JSON format from the ML models and return this response for further processing. These functions demonstrate the integration of PHP with external ML services, highlighting the role of APIs in facilitating communication between different components of a software system.

```php
public function Save(Request $request){
    $Emaildata=null;
    $Urldata = null;
    $result = -1;
    $Prob = -1;
    if(!empty($request->message)){
        $messag = $this->apiEmail($request->message);
        $result = $messag['det'];
        $Prob = $messag['prob'];
        $Emaildata = PhishingResult::create([
        'url'=>$request->message,
        'result'=>$messag['det']
    ]);
    $Emaildata->save();
}

if(!empty($request->url)){
    $messag = $this->apiURL($request->url);
    $result = $messag['det'];
    $Prob = $messag['prob'];
    $Urldata = PhishingResultUrl::create([
        'url'=>$request->url,
        'result'=>$messag['det']
    ]);
    $Urldata->save();
}

$urldataId = $Urldata ? $Urldata->id : null;
 $emaildataId = $Emaildata ? $Emaildata->id : null;
return redirect('/')->with(['ResultIdUrl' => $urldataId ,'ResultIdEmail' => $emaildataId, 'result' => $result, 'prob'=> $Prob]);
}
```

Figure 5-10 Save function for interacting with database.

Save(Request $request) is designed to handle incoming requests for phishing attack detection within a larger software system. It initializes variables to store email and URL data, as well as variables for the detection result and probability.

When a request containing a message is received, the function uses an API call (apiEmail($request->message)) to analyze the message using an email analysis ML model. It then stores the result, probability, and message in a database table (PhishingResult).

Similarly, if the request includes a URL, the function utilizes another API call (apiURL($request->url)) to analyze the URL using a URL analysis ML model. It stores the result, probability, and URL in a separate database table (PhishingResultUrl).

After processing the data, the function redirects the user back to the main page with the analysis results and IDs of the stored data for further reference or display. This function demonstrates the integration of PHP, Laravel framework, and ML models for automated phishing attack detection and data management in a web-based environment.

```php
    public function feedbacksave(Request $request){
        $data = FeedBack::create([
            'ResultIdUrl'=>$request->ResultIdEmail,
            'ResultIdEmail'=>$request->ResultIdUrl,
            'feedback'=>$request->feedback
        ]);
        $data->save();
        return redirect('/');
    }

    1 reference | 0 overrides
    public function DisplayItems(){
        $emailData = DB::table('phishing_results')
    ->join('feed_backs', 'phishing_results.id', '=', 'feed_backs.ResultIdUrl')
    ->select('phishing_results.*', 'feed_backs.feedback')
    ->get();

    $urlData = DB::table('phishing_result_urls')
    ->join('feed_backs', 'phishing_result_urls.id', '=', 'feed_backs.ResultIdEmail')
    ->select('phishing_result_urls.*', 'feed_backs.feedback')
    ->get();
        return view('home',['urldata'=>$urlData, 'emaildata'=>$emailData]);
    }
```

Figure 5-11 Feedback function for interacting with database, displayitem for displaying data from database.

The feedbacksave(Request $request) function is responsible for saving user feedback related to phishing attack analysis results. It creates a new record in the FeedBack database table with the feedback text provided by the user, along with the IDs of the analyzed results from the email and URL analysis.Once the data is created, it saves the record and redirects the user back to the main page.The DisplayItems() function retrieves and displays the feedback along with the corresponding analysis results. It queries the database to fetch data from the phishing_results, phishing_result_urls, and feed_backs tables, joining them based on the result IDs. The retrieved data includes the analysis results (URL or email) and the associated feedback.

Finally, it returns the fetched data to the view 'home' for display, organizing it into variables (urldata and emaildata) for easy access and rendering in the user interface. This setup allows users to view the analysis results and associated feedback in a structured manner on the home page of the application.

3. Flask :

```python
def detection(new_email_text):
    new_email_text = new_email_text.lower()
    new_email_text = remove_special_characters(new_email_text)
    new_email_text = remove_hyperlink(new_email_text)
    new_email_tokens = word_tokenize(new_email_text)
    new_email_tokens = remove_stop_words(new_email_tokens)
    new_email_cleaned = ' '.join(new_email_tokens)
    new_email_features = cv.transform([new_email_cleaned])
    nb_loaded = joblib.load("C:/Users/7rbe2/OneDrive/سطح المكتب/Main projects/Grad project/Phishward/PhishWarden/app/Python/phe1.joblib")
    new_email_features = cv.transform([new_email_cleaned])
    prediction = nb_loaded.predict(new_email_features)
    probability = nb_loaded.predict_proba(new_email_features)
    ph = "detect"
    if prediction[0] == 1:
        ph = 1
        prob = format(probability[0][1] * 100, '.2f')
    else:
        ph = 0
        prob = format(probability[0][0] * 100, '.2f')
    return ph, prob

from flask import Flask, request, jsonify
app = Flask(__name__)
@app.route('/ml', methods=['POST'])
def ml_endpoint():
    content = request.json
    input_text = content['input_text']
    result, prob =detection(input_text)
    print(result)
    return jsonify({'det': result, 'prob': prob})
__name__ == '__main__'
app.run(debug=True, port=5001)
```

Figure 5-12 Feature engineering and endpoint initiating.

In figure 5-12 shows, function detection(new_email_text) that performs phishing attack detection on email text. The function preprocesses the input text by converting it to lowercase, removing special characters and hyperlinks, tokenizing it into words, and removing stop words. It then transforms the cleaned text into features using a CountVectorizer (cv) and predicts whether the email is a phishing attempt using a pre-trained Naive Bayes model loaded from a joblib file. The function returns the prediction result (ph) and the probability (prob) of the prediction.

The Flask application (app) sets up an endpoint (/ml) that receives POST requests containing input text for phishing detection. It extracts the input text from the request, calls the detection function to get the detection result and probability, and returns the result and probability in JSON format. The code also includes a check to run the Flask app only if the script is executed directly (not imported as a module) using __name__ == '__main__'.

Overall, this code sets up a Flask API endpoint for phishing attack detection using a pre-trained machine learning model and provides a way to interact with the detection functionality via HTTP requests.

5.4 System Testing

System testing plays a critical role in ensuring the quality and reliability of software systems before their deployment. Two primary methodologies used in system testing are black box testing and white box testing. Black box testing focuses on the functionality of the software

without considering its internal structure or implementation details. Test cases are designed based on the system's specifications and expected behavior, simulating inputs, and evaluating outputs to verify that the system functions as intended. This approach is akin to examining a black box, where the tester interacts with the system externally to validate its functionality and detect any deviations from expected results.

On the other hand, white box testing delves into the internal logic and structure of the software. Test cases are designed based on an understanding of the system's code, algorithms, and data structures. This method allows testers to assess the correctness of individual components, paths, and decision points within the system. By analyzing the system's internal workings, white box testing can uncover issues related to code errors, logic flaws, and performance bottlenecks that may not be apparent through black box testing alone.

As we delve into the implementation of black box test cases for system testing, we'll explore how to derive scenarios from various sources such as use cases, functional requirements, and non-functional requirements. These elements form the backbone of our testing strategy, ensuring that we cover critical functionalities, user interactions, and system behaviors effectively. Additionally, we'll discuss the role of unit testing in validating individual code components and how requirement calibration keeps our test cases aligned with evolving system specifications. This integrated approach aims to enhance system reliability, detect defects early, and drive continuous improvement in software quality.

5.4.1 Test Cases

In the black box testing approach, we've selected test cases from the use case model to ensure that your system functions according to requirements. This type of testing focuses on the system's external behavior without considering its internal logic or structure, we have implemented it in two ways one which is functional and second is non-functional requirement.

1- Testing system for functional requirements

Table 5-2 Test FR1 for submitting Email/URL.

Input	Expected output	Actual output
https://eas.taibahu.edu.sa/TaibahReg/student_login.jsp	show the results page	show the results page
We regret to inform you that your PayPal account has been temporarily suspended due to suspicious activity detected on your account. To resolve this issue and prevent permanent suspension, we require you to verify your account information immediately. Please click on the link below to log in and verify your identity:	show the results page	show the results page

[Verify Your Account Now] Please note that failure to verify your account within 24 hours will result in permanent suspension and the loss of access to your PayPal account. Thank you for your cooperation.		

Table 5-3 Test FR2 for feature extraction for Email/URL.

Input	Expected output	Actual output
https://eas.taibahu.edu.sa/TaibahReg/student_login.jsp	Cleaned data	{'url_length': 54, 'domain':'eas.taibahu.edu.sa', 'abnormal_url': 1, 'http_secure': 1, 'digit_count': 0, 'letter_count': 44, 'shortening_service': 0, 'having_ip_address': 0, 'preprocessed_url': 'https ea taibahu edu sa taibahreg student login jsp'}

Table 5-4 Test FR3 classify Email/URL.

Input	Expected output	Actual output
https://eas.taibahu.edu.sa/TaibahReg/student_login.jsp	Phishing/legitimate	legitimate
We regret to inform you that your PayPal account has been temporarily suspended due to suspicious activity detected on your account. To resolve this issue and prevent permanent suspension, we require you to verify your account information immediately. Please click on the link below to log in and verify your identity: [Verify Your Account Now] Please note that failure to verify your account within 24 hours will result in permanent suspension and the loss of access to your PayPal account. Thank you for your cooperation.	Phishing/ legitimate	Phishing

Table 5-5 Test FR4 Preprocess.

Input	Expected output	Actual output
Mr. Osama who are you?	Cleaned data	mr osama
Hi Ahmed, I hope this message finds you well. I wanted to remind you about our scheduled meeting tomorrow at 10:00 AM	Cleaned data	hi ahmed hope message finds well wanted remind scheduled meeting tomorrow 1000

```
# Clean the email text (convert to lowercase, remove special characters, hyperlinks,
new_email_text = new_email_text.lower()
new_email_text = remove_special_characters(new_email_text)
new_email_text = remove_hyperlink(new_email_text)

# Tokenize the text
new_email_tokens = word_tokenize(new_email_text)


# Remove stop words
new_email_tokens = remove_stop_words(new_email_tokens)

# Rejoin the tokens into a single string
new_email_cleaned = ' '.join(new_email_tokens)
print(new_email_cleaned)
```

Figure 5-13 FR4 for Preprocess for Email/URL.

```
hi ahmed hope message finds well wanted remind scheduled meeting tomorrow 1000
```

Figure 5-14 The output of FR4.

Table 5-6 Test FR5 View result of detection.

Input	Expected output	Actual output
Subject: Invitation to Tech Conference Dear John Doe, We are excited to invite you to the Tech Innovations Conference, which will be held on June 15th, 2024 at Tech Plaza Convention Center. This conference is an excellent opportunity to network with industry leaders, attend informative workshops, and discover the latest trends in technology. Keynote speakers include renowned experts such as Dr. Jane Smith, CEO of TechCorp, and Mr. James Brown, CTO of InnovateX. They will share their insights on emerging technologies like artificial intelligence, blockchain, and cybersecurity. The agenda also features panel discussions on topics such as digital transformation, cloud computing, and data analytics.	Phishing/ legitimate	Phishing

Additionally, there will be interactive demo sessions showcasing cutting-edge tech products and solutions. Please RSVP by June 1st, 2024 to confirm your attendance. We look forward to welcoming you to this insightful event. Best regards, Alex Johnson Event Coordinator Tech Innovations Inc.		

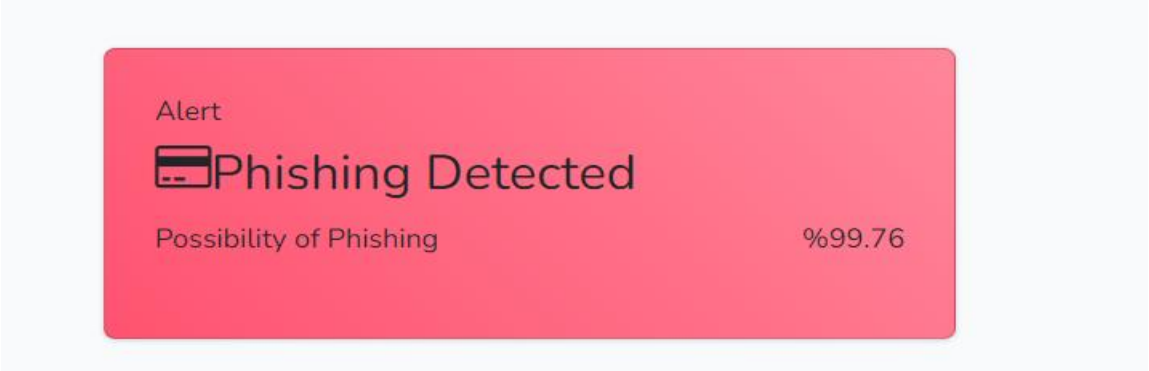


Figure 5-15 User viewing result of detection.

Table 5-7 Test FR6 Make feedback.

Input	Expected output	Actual output
Correct trigger	Submitted successfully	Submitted successfully

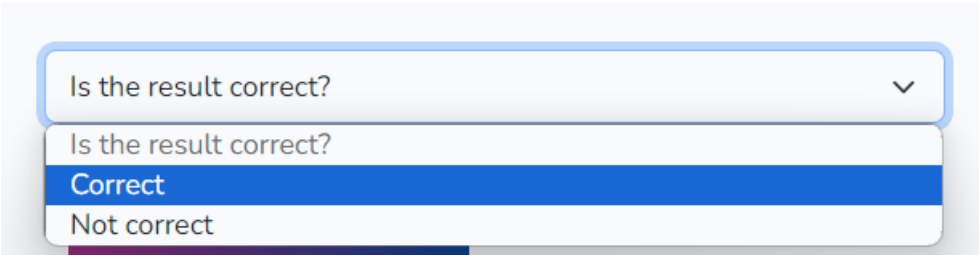


Figure 5-16 User making feedback on result.

2- Testing system for non-functional requirements

Performance:

Connecting our controller to an ML model locally enhances performance by reducing network latency, optimizing data transfer speed, and improving resource utilization. This setup also reduces dependency on external services, offers better scalability, and enhances security and privacy. Overall, local connections provide faster, more efficient, and more reliable interactions between our system and the ML model, ensuring a smoother user experience and easier management of our infrastructure.

Reliability:

Our implementation has significantly enhanced reliability through several key mechanisms. Firstly, it seamlessly integrates with external APIs for phishing detection, leveraging specialized machine learning models for accurate results. This integration not only improves detection capabilities but also ensures the system stays up-to-date with the latest security measures. Secondly, robust error handling mechanisms are in place, handling empty inputs and API responses effectively to prevent unexpected issues and maintain smooth operation. Thirdly, data validation checks prior to saving data guarantee that only valid and safe inputs are processed, reducing the risk of data corruption or security breaches. Additionally, the structured database operations and transaction safety enhance data integrity, crucial for maintaining a reliable system. Lastly, the feedback loop provided to users after processing inputs adds transparency and user trust, contributing to an overall reliable and error-free user experience.Top of Form

Accuracy: This is a general measure of how often the system correctly classifies URLs and emails. It's calculated as the number of correct classifications (true positives and true negatives) divided by the total number of classifications.

To achieve accuracy, We Train machine learning models with a diverse and representative dataset to enhance accuracy.

True Positive (TP): A phishing URL/email correctly identified as phishing.

True Negative (TN): A legitimate URL/email correctly identified as legitimate.

False Positive (FP): A legitimate URL/email incorrectly classified as phishing (raises a false alarm).

False Negative (FN): A phishing URL/email incorrectly classified as legitimate (misses a real phishing attempt).

Precision: This metric tells you how many of the positive detections were actually correct (avoiding false positives). It's calculated as TP divided by (TP + FP).

Recall (Sensitivity): Recall, also known as sensitivity or true positive rate, measures the proportion of true positive cases that are correctly identified by the model. It is calculated using the formula:

Recall = True Positives / (True Positives + False Negatives).

F1 Score: The F1 score is a measure of a model's accuracy that balances both precision and recall. It is the harmonic mean of precision and recall and is calculated using the formula:

F1= (2 × precision × recall) / (precision + recall) [10].

Here are some examples of algorithms model we get :

1- MultinomialNB

```
Training Accuracy : 0.9831431949637299
Testing Accuracy : 0.9756015722366025

CLASSIFICATION REPORT
              precision    recall  f1-score   support

         Bad       0.99      0.96      0.97     10197
        Good       0.96      0.99      0.98     10665

    accuracy                           0.98     20862
   macro avg       0.98      0.98      0.98     20862
weighted avg       0.98      0.98      0.98     20862

CONFUSION MATRIX
<Axes: >
```

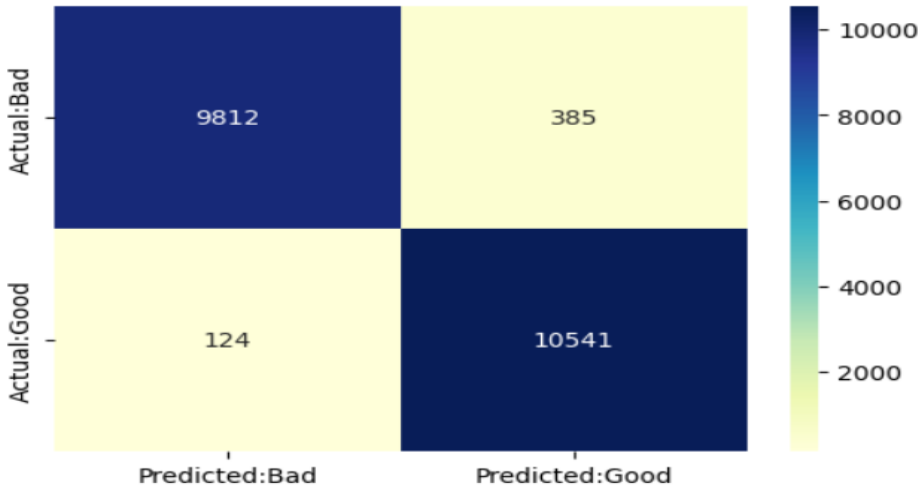


Figure 5-17 Classification report of MultinomialNB.

2 - Logistic Regression :

```
Training Accuracy : 0.9982264404179848
Testing Accuracy : 0.985571853130093

CLASSIFICATION REPORT
              precision    recall  f1-score   support

         Bad       0.98      0.99      0.98      9715
        Good       0.99      0.98      0.99     11147

    accuracy                           0.99     20862
   macro avg       0.99      0.99      0.99     20862
weighted avg       0.99      0.99      0.99     20862

CONFUSION MATRIX
<Axes: >
```

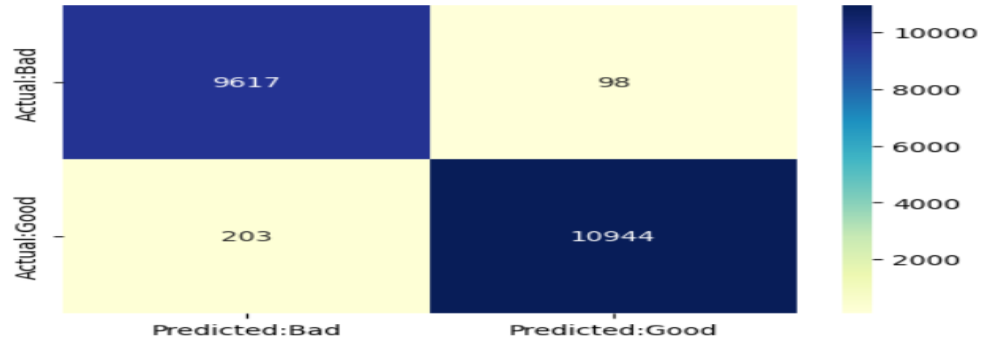


Figure 5-18 Classification report of Logistic Regression.

3- Decision Tree :

```
[ ]
Training Accuracy : 0.9958874074074074
Testing Accuracy : 0.9712266666666667

CLASSIFICATION REPORT

              precision    recall  f1-score   support

         Bad       0.91      0.95      0.93     21468
        Good       0.99      0.98      0.98     91032

    accuracy                           0.97    112500
   macro avg       0.95      0.96      0.95    112500
weighted avg       0.97      0.97      0.97    112500

CONFUSION MATRIX
<Axes: >
```

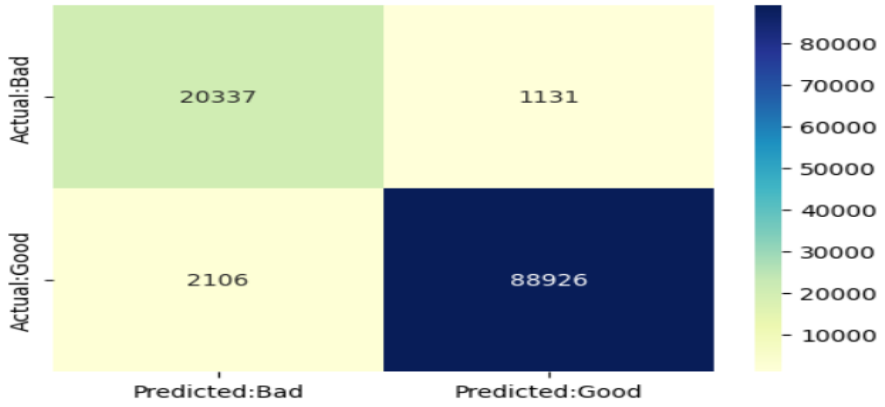


Figure 5-19 Classification report of Decision Tree.

This is tables of the Accuracy for all models :

1- Accuracy for email models :

	Model	Accuracy
0	LogisticRegression	0.985572
1	Decision Tree Classifier	0.961845
2	MultinomialNB	0.977279
3	svm_classifier	0.979388

Figure 5-20 Accuracy for email models.

2- Accuracy for URL models :

	Model	Accuracy
0	LogisticRegression	0.982996
1	Decision Tree Classifier	0.989876
2	MultinomialNB	0.983262

Figure 5-21 Accuracy for URL models .

ease of use:

To achieve this, we've prioritized friendly user interface design, which focuses on a seamless and enjoyable user experience, we have succeeded in providing an interactive environment that responds to users' needs with ease and simplicity. The principles of friendly interface design clarity, consistency, accessibility, providing feedback, and simplicity are fundamental to making our website easy to use and adaptable for all user categories. This strong focus on ease of use ensures an efficient and comfortable experience for users of our website, helping them to detect and confront online phishing attacks with confidence and ease.

5.4.2 Unit Testing

Here we will be writing for white box testing especially unit testing.

Table 5-8 Unit Testing.

Test ID	Test Function	Pass/Fail
UT1	testNewUser	PASS
UT2	testNewUserRoute	PASS
UT3	testSaveEmail	PASS
UT4	testFeedbackSave	PASS
UT5	testExistUser	PASS

Explanation for each test:

- **testNewUser**: This test has successfully passed because of the assertion for redirecting to '/home' after registering a new user with the provided similar email and password. So, we succeeded in detecting that the email was duplicated.
- **testNewUserRoute:** This test passes as it asserts that after logging in with specific credentials ('admin@gmail.com' and 'z1234567'), the user is redirected to '/home'.
- **testSaveEmail:** This test passes as it checks if saving a specific message in the database after posting to '/save' and redirects to '/' is successful.

- **testFeedbackSave:** This test passes as it checks if saving feedback in the database after posting to '/feedback' and redirects to '/' is successful.
- **testExistUser:** This test passes as it checks if the user 'admin' exists in the 'users' table, so it was able to interact with database smoothly.

5.4.3 Requirements Calibration

Table 5-9 Requirements Calibration.

Requirement	Achieved degree
Build an AI-powered website to classify and detect emails and URLs for phishing trials, with high accuracy and precision and easy to use system.	Achieved
The website shall provide clear and concise reporting information about the phishing classification and analysis results.	Achieved
The system shall be able to integrate website with Machine learning model.	Achieved
The system must allow all entities to submit email/website address for analysis.	Achieved
The machine learning model must do Feature extraction to the Email/URL.	Achieved
The machine learning model must classify the Email/URL	Achieved
The machine learning model must do Pre-processing to the Email/URL.	Achieved
The system must allow all entities to view the result of detection.	Achieved
The system must allow all entities to make feedback.	Achieved
The system should be able to optimize the performance of the machine learning module to deliver fast and efficiently.	Achieved

The system should achieve a high level of accuracy in predicting phishing attacks while minimizing false positives.	Achieved
The user interface should be easy to explore, user should be familiar after second use.	Achieved
The system should be able to handle and prevent errors	Achieved
The system is built by the SDLC methodology (waterfall)	Achieved
The system meets the requirements of use cases	Achieved
The system achieves and mapping design Moodle	Achieved
The system utilized a combination of black box and white box testing methodologies	Achieved

5.5 Summary

In this chapter, our primary emphasis was on the process of transitioning from design to implementation. We commenced by introducing the tools and programming languages employed in our project, offering comprehensive descriptions of each to facilitate understanding. Subsequently, we delved into the intricate process of mapping design concepts to practical implementation. Following this, we elucidated the results obtained from rigorous system testing, ensuring the functionality and reliability of our implementation. Concluding the chapter, we encapsulated our findings and underscored their significance in reinforcing the project's objectives and outcomes.

Chapter 6. Conclusion and Lessons Learnt

6.1 Conclusion

Phishing attacks pose a significant threat to online security, causing substantial financial losses annually. Machine learning (ML) proves to be a potent tool for highly accurate phishing detection, employing various algorithms like support vector machines (SVMs), random forests, and neural networks. These algorithms learn from diverse features such as sender email addresses, URL links, and email content, typically trained on extensive datasets of labeled phishing and legitimate emails. ML-based phishing detection systems are gaining popularity, playing a crucial role in protecting individuals and businesses from phishing attacks.

Our comprehensive project focuses on developing a machine learning solution for detecting phishing attacks in emails and URLs. In Chapter 1, we outlined the project plan, defined the problem, proposed solutions, discussed the approach, and highlighted objectives. Chapter 2 involved exploring features in similar systems, conducting a meticulous analysis, and identifying pivotal features for our ML system. In Chapter 3, we delved into system requirements analysis, actively devising precise solutions and gaining insights into the integration process.

Chapter 4 focuses on System Design, emphasizing the integration of Architecture Design and a user-friendly website into our ML System for Detecting Phishing Attacks. This integration brings several benefits, including improved system structure aiding scalability, optimized data flow, enhanced feature extraction, seamless module collaboration, increased accessibility through a user-friendly interface, adaptability to evolving demands, and an efficiently structured architecture.

Chapter 5 delves into the technical aspects of the project, focusing on Tools and Languages, System Implementation, and System Testing. In Tools and Languages, the project utilized a combination of Python and Php laravel for backend development, Flask for web framework, HTML/CSS and bootstrap for frontend, and libraries such as nltk for natural language processing. System Implementation covered the integration of machine learning models for phishing detection, user authentication using Php laravel sessions, and MySQL database integration for storing user feedback and system data. System Testing involved black box testing based on use cases, functional and non-functional requirements, and unit testing to ensure the system's functionality, security, and user-friendliness.

6.2 Lessons Learnt

- Revealing the Depths of Artificial Intelligence: A Journey into Algorithmic Understanding
- This endeavor has granted us valuable insights into the intricate landscape of artificial intelligence, with a specific emphasis on comprehending algorithms.
- Mastering the Dynamics of Phishing Attacks: A Focus on Email and URL Security
- Through this project, we've deepened our understanding of phishing attacks, specifically delving into the nuances of email and URL security.
- Precision in Problem Articulation: Ensuring Clarity for Seamless Research Progress
- By meticulously formulating problem statements, we aim to eliminate ambiguity, fostering a clear path for subsequent research phases.
- A comprehensive literature review is crucial. It not only helps you understand the existing work but also guides you in identifying the gaps in research.
- Clearly define the problem you're addressing. Ambiguity in the problem statement can lead to confusion in the later stages of your research.
- Choose your research methodology wisely. Different methodologies suit different projects. Understand the strengths and limitations of each.
- Data collection can be time-consuming and challenging. Be prepared for unexpected issues, such as the availability of datasets or ethical considerations.
- Data collection can be time-consuming and challenging. Be prepared for unexpected issues, such as the availability of datasets or ethical considerations.
- Communicating your ideas clearly is essential. Ensure your paper is well-structured, and your arguments are logically presented.
- Seek feedback from peers, mentors, or professors throughout the writing process.
- MVC helped in clearly defining the responsibilities of each component (Model, View, Controller), leading to cleaner and more maintainable code.
- Gained insights into handling errors effectively, such as invalid inputs or unexpected behaviors, to improve system robustness and reliability.
- Learned best practices for integrating machine learning models with web applications using HTTP requests and handling JSON responses.
- Learned about implementing secure authentication mechanisms, such as password hashing and session management, to protect user accounts and sensitive data.
- Gained experience in designing and integrating databases to store and manage application data efficiently.

- Explored user experience (UX) design principles to create a user-friendly and intuitive interface, enhancing user satisfaction and engagement.

6.3 Limitations

Due to the lack of knowledge in the field of Ai and Machine learning, we had to reduce the features of our project so that we can implement the project accurately. the features that We were planning to add if we had the knowledge, such as:

- Using agile methodology as a main development for our project so we can implement our project as soon as possible and see results.
- Automation for the email detection using Gmail, outlook, or any email company so we can make out product easy and efficient for the use.
- Gather Arabic dataset for supporting Arabic phishing detection.

6.4 Future Work

In our future endeavors, we envision an expansion of our website to encompass a wider range of phishing attack scenarios. This strategic enhancement will specifically target various forms of phishing, including SMS phishing, voice phishing, malvertising. To elevate user experience, we plan to integrate a user-friendly dashboard directly into the website interface, aiming to simplify navigation and provide an intuitive interaction. Furthermore, recognizing the prevalence of mobile users, we plan to introduce a dedicated application tailored for users who primarily access our services via their phones.

References

[1] T. w. l. t. p. o. f. t. a. o. technology, The world's largest technical professional, IEEE.

[2] Google, "GoogleScholar," Google, 2005. [Online]. Available: http://scholar.google.com/scholar/about.html.

[3] IBM, "IBM," [Online]. Available: https://www.ibm.com.

[4] H. J. M. Michael E. Whitman, Principles of Information Security, Boston, MA 02210: COURSE TECHNOLOGY-CENGAGE learning, 2011.

[5] W. L. Kevin D.mitnick, THE ART OF DECEPTION:Controlling the Human Element of Security, Wiley, 2002.

[6] Cloudflare, "Cloudflare," 2019. [Online]. Available: https://www.cloudflare.com.

[7] microsoft, "What is phishing?," [Online]. Available: https://www.microsoft.com/en-us/security/business/security-101/what-is-phishing#dangers-of-phishing-emails.

[8] A. Garcia, "Mitigating Phishing Attacks: Strategies for Prevention and Detection.," *Cybersecurity Journal,* vol. 15, no. 2, 2020.

[9] T. M. T. G. K. S. Paul Cichonski, Computer Security Incident Handling Guide, National Institute of Standards and Technology (NIST).

[10] P. N. Stuart Russell, Artificial Intelligence: A Modern Approach 2nd Edition, Prentice Hall, 2003.

[11] IBM, "learning?, What is machine," [Online]. Available: https://www.ibm.com/topics/machine-learning.

[12] J. H. M. Daniel Jurafsky, Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition, September 23, 2018.

[13] M. Z. ,. X. L. ,. A. R. J. ,. Z. J. ,. K. K. Abdul Basit, A comprehensive survey of AI-enabled phishing attacks detection, 2020.

[14] A. A. I. K. M. M. H. a. I. H. S. S. Hossain, ""Crime Prediction Using Spatio-Temporal Data", in Computing Science,," Singapore, 2020.

[15] S. O. B. E. D. O, "Machine learning based phishing detection from URLs," 2019.

[16] L. Y. Y. Z. C. X, "A stacking model using URL and HTML features for phishing webpage detection," 2019.

[17] J. S. R. E.B., "Intelligent phishing URL detection using association rule mining," 2016.

[18] V. G. M. M. A. P.K, "A phish detector using lightweight search features," 2016.

[19] S. M. S. H. Y. M. U. orgM. Vijayalakshmi, Web phishing detection techniques: a surveyon the state-of-the-art, taxonomy and futuredirections, 2020.

[20] [Online]. Available: http://dataaspirant.com/2017/05/22/random-forest algorithm-machine-learing/.

[21] R. Touny, "Phishing-Attack-Detection-using-Machine-Learning," 2023. [Online]. Available: https://github.com/RimTouny/Phishing-Attack-Detection-using-Machine-Learning.

[22] P. Baskaran1, Phishing Websites Spotting with Help of using Machine Learning Tools, may 2023.

[23] Dinesh, Mukesh, Navaneethan, Sabeeniam, Paramasivm and Manjunathan, Identification of Phishing Attacks using Learning Algorithm, Department of Electronics and Communication Engineering, Sona College of Technology, 2023.

[24] M. P, Detection and classification of phishing websites, 2021.

[25] M. P. Nagasunder Rao Pawar Babu Rao Pawar, Detection of Phishing URL using Machine, National College of Ireland.

[26] A. S. A. S. Roberto Basili, "Classification of Musical Genre: A Machine Learning Approach," Roma (Italy), University of Rome Tor Vergata, Department of Computer Science, Systems and Production.

[27] "About Proofpoint - Company Overview.," Proofpoint, [Online]. Available: https://www.proofpoint.com/us/company/about.

[28] "How Proofpoint Aegis Uses Machine Learning," Proofpoint, [Online]. Available: https://www.proofpoint.com/us/blog/engineering-insights/leveraging-ml-to-detect-ai-generated-phishing-emails.

[29] P. P. Rajesh H. Kulkarni, Integration of artificial intelligence activities in software development processes and measuring effectiveness of integration, IET Software, 16th September 2016.

[30] P. H. P. a. D. Y. K. Sharma, "Implication of Artificial Intelligence in Software Development Life Cycle: A state of the art review," *International Journal of Recent Research Aspects,* vol. Vol. 6, no. Issue 2, pp. 93-928, June 2019.

[31] B. B. &. A. H. Dutoit, Object-Oriented Software Engineering Using UML, Patterns, and Java, Munich,Pittsburgh, PA: Technical University of Munich Department of Computer Science,Carnegie Mellon University School of Computer Science.

[32] I. Sommerville, Software Engineering.

[33] A. A. A. a. A. O. Adewumi, Classification of Phishing Email Using Random Forest Machine, 2014.

[34] R. L. T. D. B. C. M. Neil Chou, Client-side defense against web-based identity theft, Computer Science Department, Stanford University.

[35] R. S. Pressman, Software Engineering: A Practitioner's Approach, McGraw-Hill.