

WEB PHISHING DETECTION

**NALAIYA THIRAN PROJECT BASED ON LEARNING
PROFESSIONAL READLINESS FOR INNOVATION,
EMPLOYMENT AND ENTREPRENEURSHIP**

PROJECT REPORT

TEAM ID: PNT2022TMID15184

TEAM MEMBERS

- 1.A.V.S. SREE HARIKA-111519104007
- 2.CH. BINDU MADHAVI-111519104015
- 3.C. SAI SADVIKA REDDY-111519104018
- 4.D. VIJAYA SAI-111519104021

**BACHELOUR OF ENGINEERING IN
COMPUTERSCIENCE AND
ENGINEERING**

**RMD ENGINEERING
COLLEG
TIRUVALLUR – 601 206**

TABLE OF CONTENTS

1. INTRODUCTION

1.1 Project Overview

1.2 Purpose

2. LITERATURE SURVEY

2.1 Existing problem

2.2 References

2.3 Problem Statement Definition

3. IDEATION & PROPOSED SOLUTION

3.1 Empathy Map Canvas

3.2 Ideation & Brainstorming

3.3 Proposed Solution

3.4 Problem Solution fit

4. REQUIREMENT ANALYSIS

4.1 Functional requirement

4.2 Non-Functional requirements

5. PROJECT DESIGN

5.1 Data Flow Diagrams

5.2 Solution & Technical Architecture

5.3 User Stories

6. PROJECT PLANNING & SCHEDULING

6.1 Sprint Planning & Estimation

6.2 Sprint Delivery Schedule

6.3 Reports from JIRA

7. CODING & SOLUTIONING (Explain the features added in the project along with code)

7.1 Feature 1

7.2 Feature 2

7.3 Database Schema (if Applicable)

8. TESTING

8.1 Test Cases

8.2 User Acceptance Testing

9. RESULTS

9.1 Performance Metrics

10. ADVANTAGES & DISADVANTAGES

11. CONCLUSION

12. FUTURE SCOPE

13. APPENDIX

Source Code

GitHub & Project Demo Link

ABSTRACT

Phishing URL is a widely used and common technique for cybersecurity attacks. Phishing is a cybercrime that tries to trick the targeted users into exposing their private and sensitive information to the attacker. The motive of the attacker is to gain access to personal information such as usernames, login credentials, passwords, financial account details, social networking data, and personal addresses. These private credentials are then often used for malicious activities such as identity theft, notoriety, financial gain, reputation damage, and many more illegal activities. This paper aims to provide a comprehensive and comparative study of various existing free service systems and research base systems used for phishing website detection. The systems in this survey range from different detection techniques and tools used by many researchers

Keywords: Deep learning, Machine learning, Phishing website attack, Phishing website detection, Anti-phishing website, Legitimate website, Phishing website datasets, Phishing website features.

PRE-REQUISITES

TOOLS: JUPITER NOTEBOOK

OPERATING SYSTEM: WINDOWS 11

LANGUAGE: PYTHON

INSTALLING LIBRARIES

In this first step, we have to import the most common libraries used in python for machine learning such as

- Pandas
- Numpy
- Seaborn
- Matplotlib

IMPORTING DATA

In this project, we have used the URL preprocessed data.

CHAPTER 1

INTRODUCTION

The advancement of internet is resulting in attracting more and more users into this huge Internet Sea. There are a lot of perks of using internet, one can buy stuff online, way of learning and gaining knowledge has improved, etc. On the contrary, possible threats comes hand in hand. One of them is Phishing Attack. Phishing is an attack where a legitimate user is deceived to disclose sensitive information and assets with economic value. Loss of such sensitive information might cause potential economic or reputational harm an organization. Phishing basically uses social engineering techniques to trick users such as creating fake websites which clones with same attributes and design of the existing legitimate one. In a classic phishing attack a phisher send a link enclosed in a message to the user. The link redirects the user to the cloned malicious page which looks similar to the original webpage but is not and is intended to steal user's sensitive data. Such phishing attacks have proven to cause a lot of financial loss to various organizations. Thus, phishing attacks can be prevented by exterminating such harmful Website URLs are categorized into the following three classes: a. Benign: These are Safe websites that provide normal services to people. b. Malware: These websites which are created by attackers look like normal websites can make use of sensitive contents of people. c. Spam: These websites flood the user's system with advertisements, fake surveys, etc. S

1.1PROJECT OVERVIEW

- To develop a novel approach to detect malicious URL and alert users.
- To apply ML techniques in the proposed approach in order to analyze the real time URLs and produce effective results.
- To implement the concept of RNN, which is a familiar ML technique that has the capability to handle huge amount of data.

1.2PURPOSE

- To develop an unsupervised deep learning method to generate insight from aURL.
- The study can be extended in order to generate an outcome for a largernetwork and protect the privacy of an individual

CHAPTER 2

Web Phishing Detection: A Literature Survey ABSTRACT:

ABSTRACT:

Phishing URL is a widely used and common technique for cybersecurity attacks. Phishing is a cybercrime that tries to trick the targeted users into exposing their private and sensitive information to the attacker. The motive of the attacker is to gain access to personal information such as usernames, login credentials, passwords, financial account details, social networking data, and personal addresses. These private credentials are then often used for malicious activities such as identity theft, notoriety, financial gain, reputation damage, and many more illegal activities. This paper aims to provide a comprehensive and comparative study of various existing free service systems and research base systems used for phishing website detection. The systems in this survey range from different detection techniques and tools used by many researchers.

INTRODUCTION:

The advancement of internet is resulting in attracting more and more users into this huge Internet Sea. There are a lot of perks of using internet, one can buy stuff online, way of learning and gaining knowledge has improved, etc. On the contrary, possible threats comes hand in hand. One of them is Phishing Attack. Phishing is an attack where a legitimate user is deceived to disclose sensitive information and assets with economic value. Loss of such sensitive information might cause potential economic or reputational harm an organization. Phishing basically uses social engineering techniques to trick users such as creating fake websites which clones with same attributes and design of the existing legitimate one. In a classic phishing attack a phisher send a link enclosed in a message to the user. The link redirects the user to the cloned malicious page which looks similar to the original webpage but is not and is intended to steal user's sensitive data. Such phishing attacks have proven to cause a lot of financial loss to various organizations. Thus, phishing attacks can be prevented by exterminating such harmful Website URLs are categorized into the following three classes: a. Benign: These are Safe websites that provide normal services to people. b. Malware: These websites which are created by attackers look like normal websites can make use of sensitive contents of people. c. Spam: These websites flood the user's system with advertisements, fake surveys, etc.

SYSTEM ARCHITECTURE APPROACHES:

1.Detection Approaches: •User training approaches — end-users can be educated to better understand the nature of phishing attacks, which ultimately leads them into correctly identifying phishing and non-phishing messages. •Software classification approaches — these mitigation approaches aim at classifying phishing and legitimate messages on behalf of the user in an attempt to bridge the gap that is left due to the

human error or ignorance. 2. Offensive Defensive Approaches: Offensive defensive solutions aim to render phishing campaigns useless for the attackers by disrupting the phishing campaigns. This is often achieved by flooding phishing web-sites with fake credentials so that the attacker would have a difficult time to find the real credentials. 3. Correction Approaches: Once a phishing campaign is detected, the correction process can begin. In the case of phishing attacks, correction is the act of taking the phishing resources down. This is often achieved by reporting attacks to Service Providers. Phishing campaigns often rely on resources, such as: • Websites — could be a shared web host owned by the phisher, a legitimate website with phishing content uploaded to it, or a number of infected end-user work-stations in a botnet. • E-mail messages — could be sent from a variety of sources, such

as: free Email Service Provider opens Simple Mail Transfer Protocol (SMTP) relays or infected end-user machines that are part of a botnet. Prevention Approaches: The “prevention” of phishing attacks can be confusing, as it can mean different things depending on its context: • Prevention of users from falling victim — in this case phishing detection techniques will also be considered prevention techniques. However, this is not the context we refer to when “prevention” is mentioned in this survey. • Prevention of attackers from starting phishing campaigns— in this case, law suits and penalties against attackers by Law Enforcement Agencies (LEAs) are considered as prevention techniques

2.1 EXISTING PROBLEM

Due to how simple it is to create a fake website that closely resembles a legitimate website, phishing has recently become a top concern for security researchers. Experts can spot fake websites, but not all users can, and those users end up falling for phishing scams. The attacker's primary goal is to steal bank account credentials. Businesses in the US lose \$2 billion annually as a result of their customers falling for phishing scams. The annual global impact of phishing was estimated to be as high as \$5 billion in the third Microsoft Computing Safer Index Report, which was published in February 2014. Because users are unaware of phishing attacks, they are becoming more successful. Since phishing attacks take advantage of user vulnerabilities, it is highly challenging to counteract them, but it is crucial to improve phishing detection methods. The common technique, commonly referred to as the "blacklist" method, for detecting phishing websites involves adding Internet Protocol (IP) blacklisted URLs to the antivirus database. Attackers utilize clever methods to deceive people by changing the URL to seem authentic through obfuscation and many other straightforward tactics, such as fast-flux, in which proxies are automatically constructed to host the website, algorithmic production of new URLs, etc. This method's primary flaw is that it cannot identify phishing attacks that occur at zero hour.

Zero-hour phishing attacks can be detected using heuristic-based detection, which includes characteristics that have been observed to exist in phishing attacks in reality. However, the presence of these characteristics is not always guaranteed in such attacks, and the false positive rate for detection

2.2 REFERENCES

- [1] Routhu Srinivasa Rao¹ , Alwyn Roshan Pais :Detection of phishing websites using an efficient feature-based machine learning framework :In Springer 2018. Volume 3, Issue 7, September-october-2018 | [http:// ijsrcseit.com](http://ijsrcseit.com) Purvi Pujara et al. Int J S Res CSE & IT. 2018 September-October-2018; 3(7) : 395-399 399
- [2] Chunlin Liu, Bo Lang : Finding effective classifier for malicious URL detection : InACM,2018
- [3] Sudhanshu Gautam, Kritika Rani and Bansidhar Joshi : Detecting Phishing Websites Using Rule-Based Classification Algorithm: A Comparison : In Springer,2018.
- [4] M. Amaad Ul Haq Tahir, Sohail Asghar, Ayesha Zafar, Saira Gillani : A Hybrid Model to Detect Phishing-Sites using Supervised Learning Algorithms :In International Conference on Computational Science and Computational Intelligence IEEE ,2016.
- [5] Hossein Shirazi, Kyle Haefner, Indrakshi Ray: Fresh-Phish: A Framework for Auto-Detection of Phishing Websites: In (International Conference on Information Reuse and Integration (IRI)) IEEE,2017.
- [6] Ankit Kumar Jain, B. B. Gupta : Towards detection of phishing websites on client-side using machine learning based approach :In Springer Science+Business Media, LLC, part of Springer Nature 2017
- [7] Bhagyashree E. Sananse, Tanuja K. Sarode : Phishing URL Detection: A Machine Learning and Web Mining-based Approach : In International Journal of ComputerApplications,2015
- [8] Mustafa AYDIN, Nazife BAYKAL : Feature Extraction and Classification Phishing Websites Based on URL : IEEE,2015
- [9] Priyanka Singh, Yogendra P.S. Maravi, Sanjeev Sharma : Phishing Websites Detection through Supervised Learning Networks : In IEEE,2015
- [10] Pradeepthi. K V and Kannan. A: Performance Study of Classification Techniques for Phishing URL Detection: In 2014 Sixth International Conference on Advanced Computing(ICoAC) IEEE,2014
- [11] Luong Anh Tuan Nguyen[†], Ba Lam To[†] ,Huu Khuong Nguyen[†] and Minh Hoang Nguyen : Detecting Phishing Web sites: A Heuristic URL-Based Approach: In The 2013 International Conference on Advanced Technologies for Communications (ATC'13)
- [12] Ahmad Abunadi, Anazida Zainal ,Oluwatobi Akanb: Feature Extraction Process: A Phishing Detection Approach :In IEEE,2013.
- [13] Rami M. Mohammad, Fadi Thabtah, Lee McCluskey: An Assessment of Features

Related to Phishing Websites using an Automated Technique: In The 7th International Conference for Internet Technology and Secured Transactions, IEEE, 2012

Mohammad et al. [13] proposed model that automatically extracts important features for phishing website detection without requiring any human intervention. Author has concluded in this paper that the process of extracting feature by their tool is much faster and reliable than any manual extraction

2.2 PROBLEM STATEMENT DEFENETION

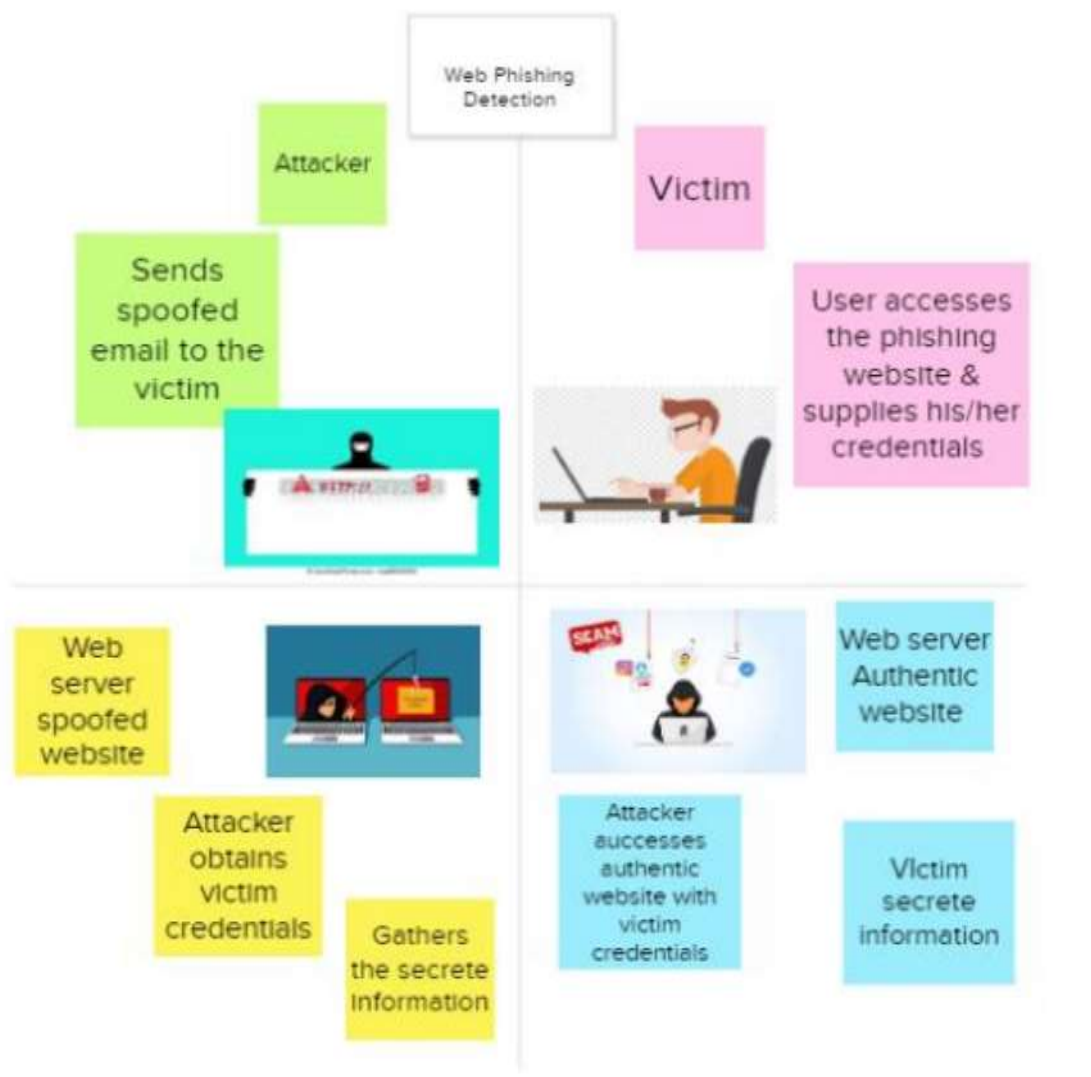
In order to detect and predict e-banking phishing websites, we proposed an intelligent, flexible and effective system that is based on using classification algorithms. We implemented classification algorithms and techniques to extract the phishing datasets criteria to classify their legitimacy. The e-banking phishing website can be detected based on some important characteristics like URL and domain identity, and security and encryption criteria in the final phishing detection rate. Once a user makes a transaction online when he makes payment through an e-banking website our system will use a data mining algorithm to detect whether the e-banking website is a phishing website or not.

Internet has dominated the world by dragging half of the world's population exponentially into the cyber world. With the booming of internet transactions, cybercrimes rapidly increased and with anonymity presented by the internet, Hackers attempt to trap the end-users through various forms such as phishing, SQL injection, malware, man-in-the-middle, domain name system tunnelling, ransomware, web trojan, and so on. Among all these attacks, phishing reports to be the most deceiving attack. Our main aim of this paper is classification of a phishing website with the aid of various machine learning techniques to achieve maximum accuracy and concise model.

CHAPTER 3

IDEATION & PROPOSED SOLUTION

3.1 Empathy Map Canvas



3.2 Ideation & Brainstorming

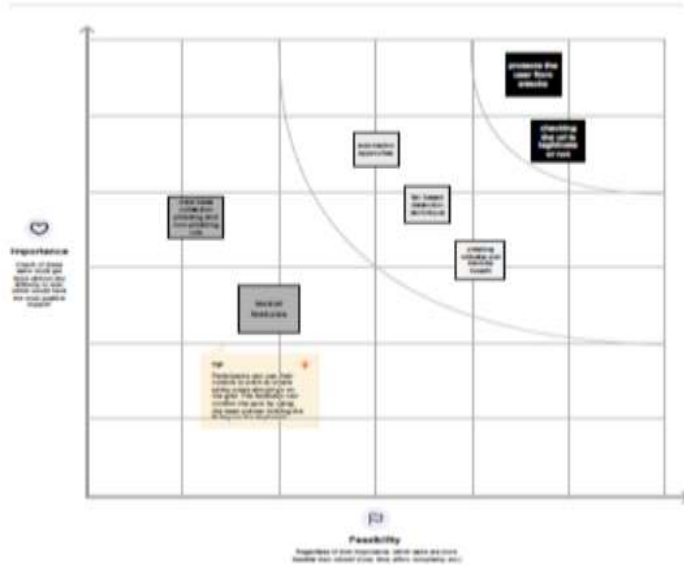
Team Gathering, Collaboration and Select the Problem Statement

1

Priority

Your team should all be on the same page about what's important moving forward. Please plot ideas on this grid to determine which ideas are important and which are feasible.

15 minutes

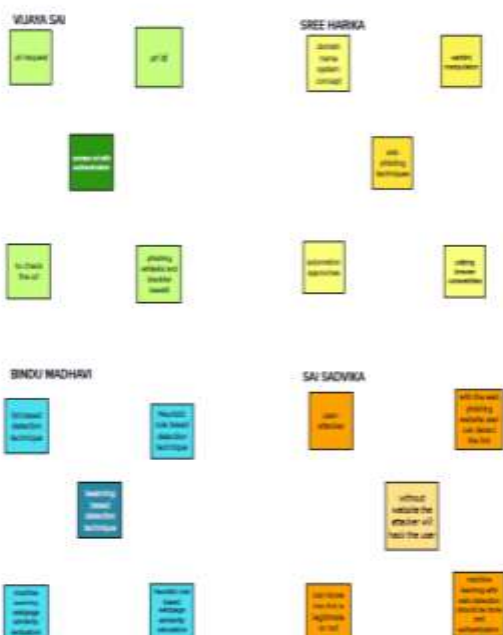


2

Brainstorm

Write down any ideas that come to mind that address your problem statement.

15 minutes



3

Group ideas

Take turns sharing your ideas while clustering similar or related notes as you go. Once all group notes have been grouped, give each cluster a sentence-like title. If a cluster is bigger than 10 ideas, try and see if you can break it up into smaller sub-groups.

15 minutes

Tip

When clustering ideas, give each cluster a sentence-like title. If a cluster is bigger than 10 ideas, try and see if you can break it up into smaller sub-groups.



3.3 Proposed Solution

Project team shall fill the following information in proposed solution template.

S.No.	Parameter	Description
1.	Problem Statement (Problem to be solved)	Web Phishing Detection There are a number of users who purchase products online and make payments through e-banking. There are e-banking websites that ask users to provide sensitive data such as username, password & credit card details, etc., often for malicious reasons. This type of e-banking website is known as a phishing website. Web service is one of the key communications software services for the Internet. Web phishing is one of many security threats to web services on the Internet.
2.	Idea / Solution description	List-based detection. List-based phishing detection methods use either whitelist or blacklist-based technique. A blacklist contains a list of suspicious domains, URLs, and IP addresses, which are used to validate if a URL is fraudulent
3.	Novelty / Uniqueness	Phish Tank's definition holds true in a number of scenarios which, roughly, cover the majority of phishing attacks (although no accurate studies have been made to reliably quantify this). However, the definition limits phishing attacks to stealing personal information, which is not always the case.
4.	Social Impact / Customer Satisfaction	Phishing has a list of negative effects on a business, including loss of money, loss of intellectual property, damage to reputation, and disruption of operational activities.
5.	Business Model (Revenue Model)	Our approach extracts and analyses different features of suspected webpages for effective identification of large-scale phishing offenses. The main contribution of this paper is the combined uses of these feature set. For improving the detection accuracy of phishing webpages, we have proposed eight new features. Our proposed features determine the relationship between the URL of the webpage and the webpage content.

6.	Scalability of the Solution	The solutions are very scalable. Anti-phishing protection refers to the security measures that individuals and organizations can take to prevent a phishing attack or to mitigate the impact of a successful attack. Certain anti-phishing protection may block email containing phishing attacks from entering a company's email system at all.
----	-----------------------------	--

3.4 Problem Solution fit

Project Title: WEB PHISHING DETECTION

ID: PNT2022TMD15184

Project Design Phase-I - Solution Fit Template

Define CS, fit into CC	1. CUSTOMER SEGMENT(S) The Participants are mostly between the ages of 18 to 25 all young people. Participants may also vary from different age groups.	4. CUSTOMER CONSTRAINTS Tips to Prevent Phishing Attacks 1.Know what a phishing scam looks like 2.Don't click on that link. 3.Get free anti-phishing add-ons. 4.Don't give your information on unsecured site. 5.Rotate passwords regularly. 6.Don't ignore those updates. 7.Install firewalls. 8.Don't be tempted by those pop-ups	5. AVAILABLE SOLUTIONS We can find solution for phishing attacks by: phishing detection websites Use anti-phishing protection and anti-spam software to protect yourself when malicious messages slip through to your computer. Anti-malware is included to prevent other types of threats. Similar to anti-spam software, anti-malware software is programmed by security researchers to spot even the stealthiest malware.	Explore AS, differentia
	2. JOBS-TO-BE-DONE / PROBLEMS Malicious links will lead to a website that often steals login credentials or financial information like credit card numbers. Attachments from phishing emails can contain malware that once opened can leave the door open to the attacker to perform malicious behavior from the user's computer.	9. PROBLEM ROOT CAUSE A phishing campaign tries to get the victim to do one of two things: Hand over sensitive information. These messages aim to trick the user into revealing important data often a username and password that the attacker can use to breach a system or account.	7. BEHAVIOUR Phishing is described as a fraudulent activity that is done to steal confidential user information such as credit card numbers, login credentials, and passwords. It is usually done by using email or other forms of electronic communication by pretending to be from a reliable business entity.	
Focus on JSP, fit into BE, understand PC	3. TRIGGERS Customers get trigger when phishing attacks are the practice of fraud communications that appear to come from a reputable source. It is usually done through email. The goal is to steal sensitive data like credit card and login information, or to install malware on the victim's machine.	10. YOUR SOLUTION Web phishing is the attempt to acquire sensitive information such as usernames, passwords, and credit card details, often for malicious reason, by masquerading as a trustworthy website on the Internet. Researchers present some solutions to detect web phishing as follows. So we need to provide a phishing detection website which can help customers to check the malicious websites and be aware of the phishing attacks. These days phishing attacks can be done in various ways like Email, social media, text accounts, and URLs.	8. CHANNELS of BEHAVIOUR ONLINE: Report it. Forward phishing emails to reportphishing@apwg.org (an address used by the Anti-Phishing Working Group, which includes ISPs, security vendors, person financial institutions, and law enforcement agencies). Let the company or that was impersonated know about the phishing scheme. OFFLINE: To overcome phishing attack we can go to near by public service centers and report a file on the attack.	Focus on JSP, fit into BE, understand PC
	4. EMOTIONS: BEFORE / AFTER From every phishing incident that has ever taken place in history, our common effect is financial loss. First is the direct loss from transferred funds by employees who were fooled by the hackers. Second is the fines for non-compliance imposed by regulatory bodies like HIPAA, PCI, and FERPA, among others.			

REQUIREMENT ANALYSIS

4.1 Functional Requirements

Following are the functional requirements of the proposed solution.

FR NO.	Functional Requirement (Epic)	Sub Requirements (Story/ Sub-Task)
FR-1	User Registration	Registration through Gmail
FR-2	User Confirmation	Confirmation via OTP and login in to the account Confirmation via email
FR-3	User login	User accesses the website by supplies login credentials
FR-4	User permission	User must give the permission access to search the engine so the intelligent system can detect the phishing website
FR-5	Using machine learning techniques	We proposed a model based on machine learning techniques to detect phishing web pages

4.2Non-functional Requirements:

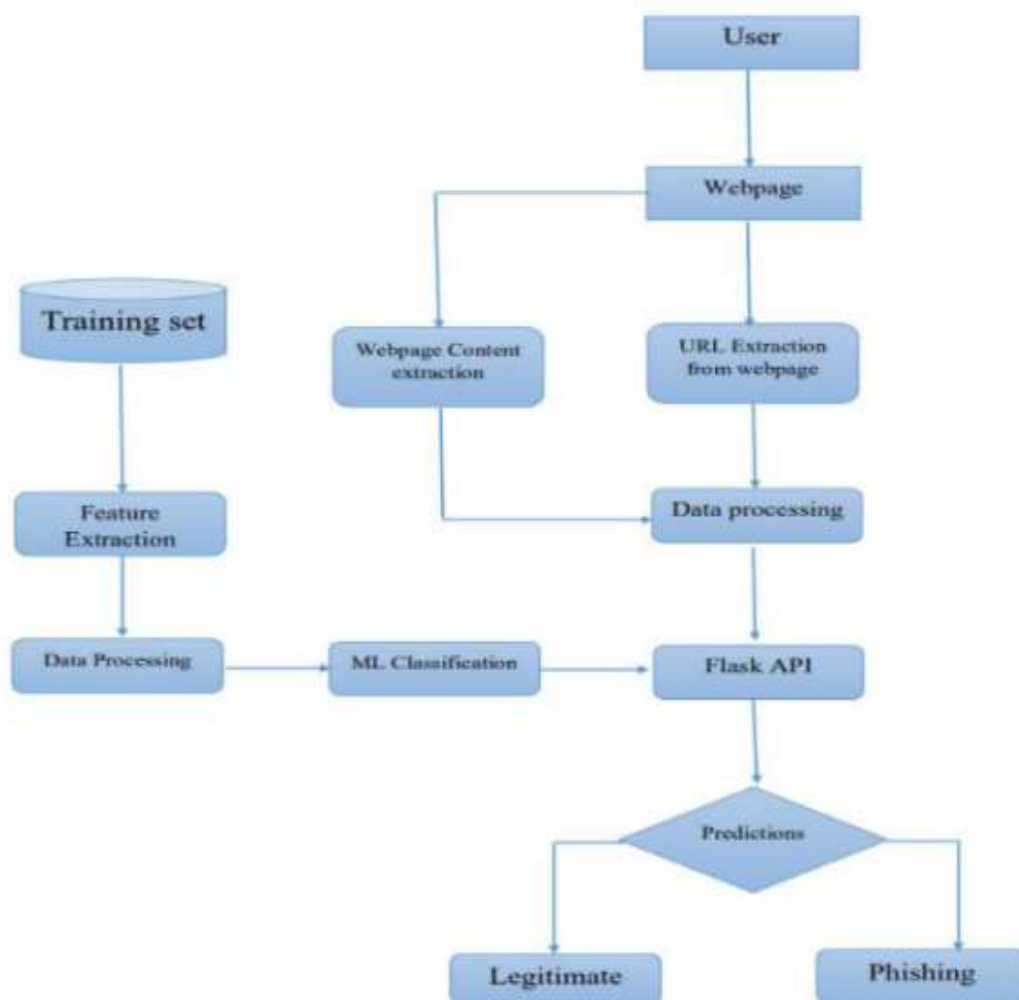
Following are the non-functional requirements of the proposed solution.

FR No.	Non-Functional Requirement	Description
NFR-1	Usability	To determine how well users are able to recognise and identify phishing web pages with anti-phishing tools ,It is a user friendly.
NFR-2	Security	It is very secure and no one cannot hack our detection website so they can trust the detection website and will be saved from financial loss
NFR-3	Reliability	phishing detection method to protect Internet users from the phishing attacks The conducted experiments show reliable and promising results.
NFR-4	Performance	The performance of web phishing detection is high and it is very efficient as it is vert easy to understand and has a high security and scalable
NFR-5	Availability	It will be available at any system like mobile phone, laptop, and desktop
NFR-6	Scalability	The main ideas are to move the protection from end users towards the network provider and to employ the novel bad neighbourhood concept, in order to detect and isolate both phishing e-mail senders and phishing web servers.

PROJECT DESIGN

5.1 Data Flow Diagrams:

A Data Flow Diagram (DFD) is a traditional visual representation of the information flows within a system. A neat and clear DFD can depict the right amount of the system requirement graphically. It shows how data enters and leaves the system, what changes the information, and where data is stored.



5.2 Solution and Technical Architecture

Solution Architecture

Example - Solution Architecture Diagram:

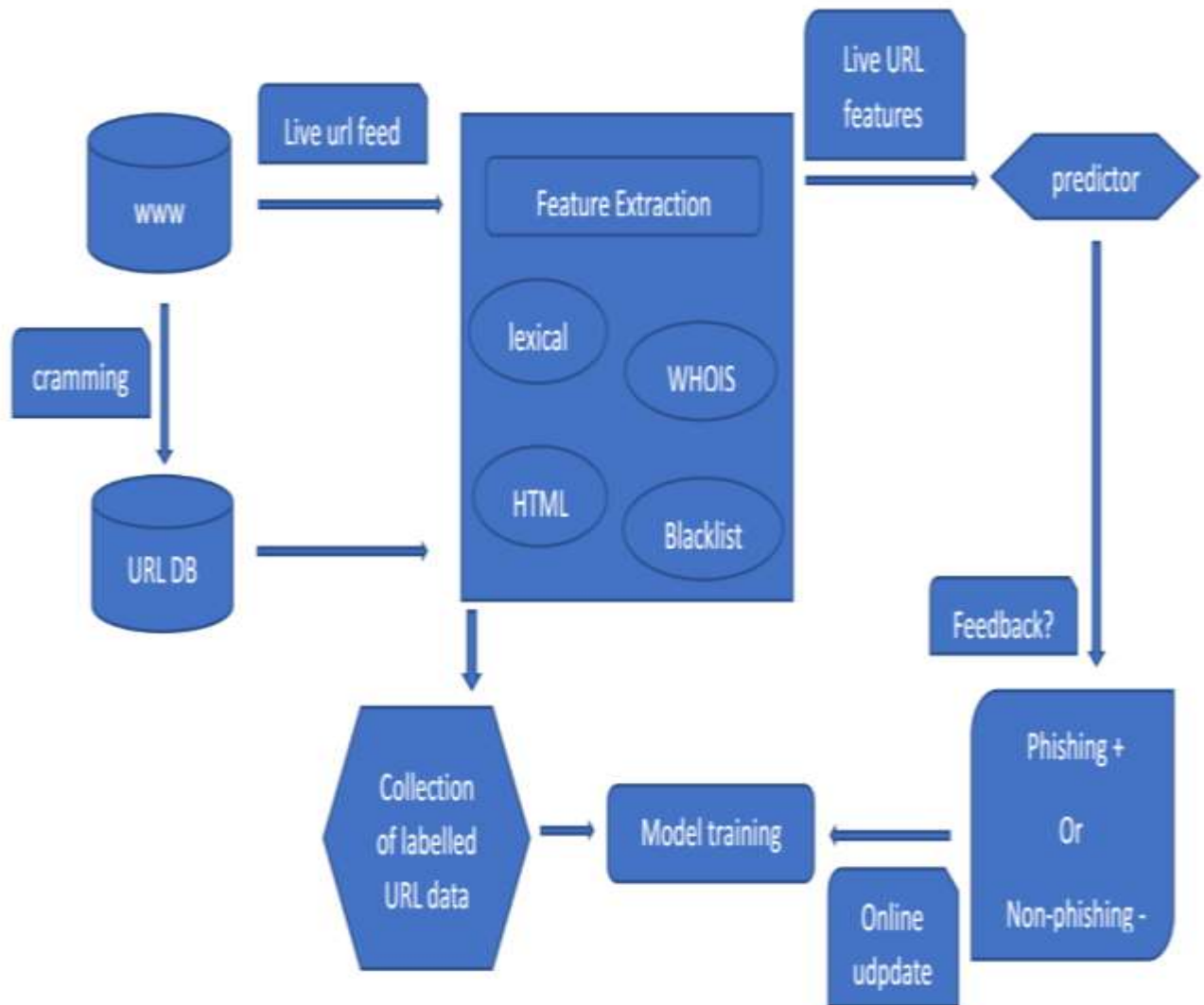
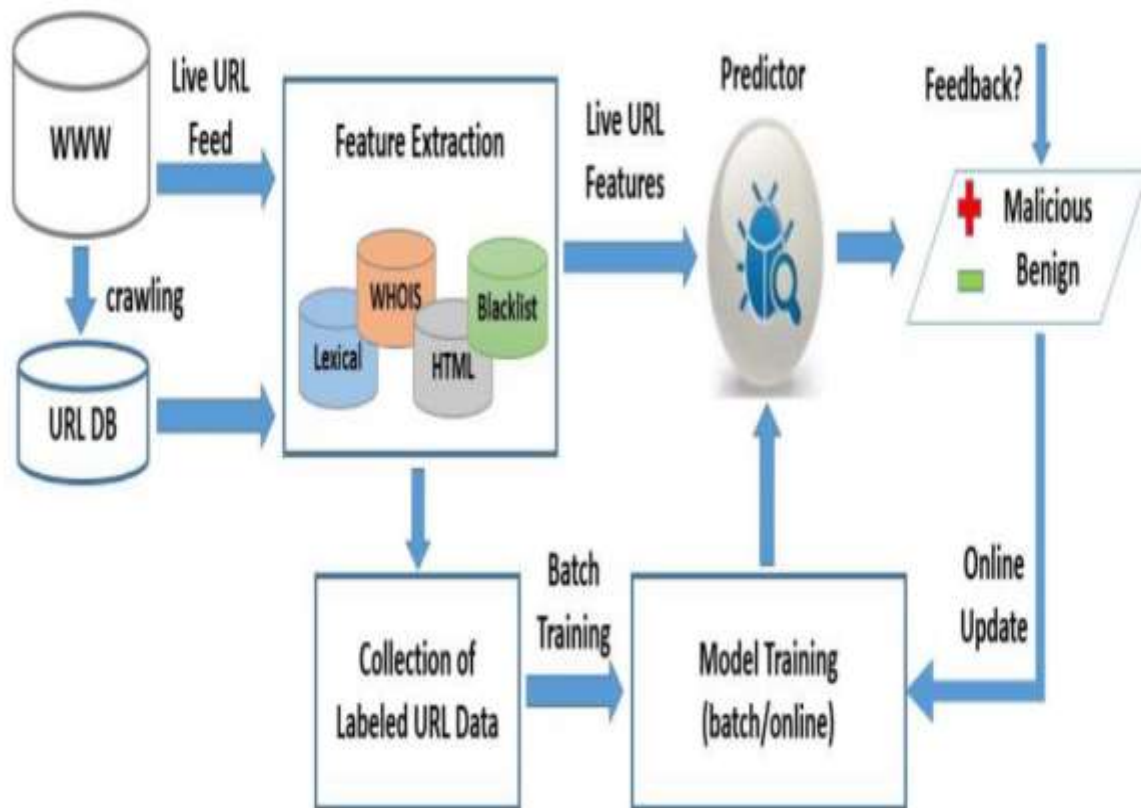


Figure 1: Architecture and data flow of the web phishing detection

Technical Architecture:
MODEL FOR WEB PHISHING DETECTION

Technical Architecture:

A general processing framework for Malicious URL Detection



5.3 USER STORIES

User Stories

Use the below template to list all the user stories for the product.

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority
Customer (Mobile user)	Download the intelligent system	USN-1	As a user, I can download the intelligent system and detect phishing websites. The system starts working immediately once you start the computer.	I can download from internet.	High
	Register	USN-2	As a user, I will register for the system using my mail and receive confirmation email once I have registered for the application	I can receive confirmation email & click confirm	High
	Login	USN-3	As a user, I can login to the application and enter my credentials and use the application.	I can login and give my credentials.	Medium
	Provide Access	USN-4	The user should provide access to google and the user's search engine so that the intelligent system can detect phishing websites.	I have to provide access for search engines.	High
	Use	USN-5	The user can use the intelligent system to detect phishing website	Finally I can use the intelligent system.	Medium
Customer (Web user)	The functional requirements are same as mobile user.	Same as mobile user	Same as mobile user	Same as mobile user	High

PROJECT PLANNING & SCHEDULING

6.1 Sprint Planning & Estimation

Use the below template to create product backlog and sprint schedule

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-1	User input	USN-1	The user enters an URL in the required field to check its validity.	5	Medium	Harika.A.V.S
Sprint-1	Website comparison	USN-2	A blacklist and whitelist approach is used to compare the websites.	10	High	Bindu.C
Sprint-1	storage	USN-3	Database storage using IBM cloud for blacklisted websites.	15	High	Sadvika.C
Sprint-2	Feature extraction	USN-4	If none of the features are found on comparison, then heuristic and visual similarity will be used to estimate their features.	10	High	Vijayasai.D
Sprint-2	Prediction	USN-5	Logistic regression algorithms are used to predict URLs.	10	Medium	Harika.A.V.S
Sprint-2	Accuracy test	USN-6	A decision is made regarding the best accurate model to be used and the next steps are processed.	15	High	Sadvika.c
Sprint-3	classifier	USN-7	Classifier receives all model outputs and produces final results.	5	Medium	Vijayasai.D
Sprint-3	Hosting	USN-8	Using IBM cloud services to set up applications and host them	10	Medium	Bindu.Ch

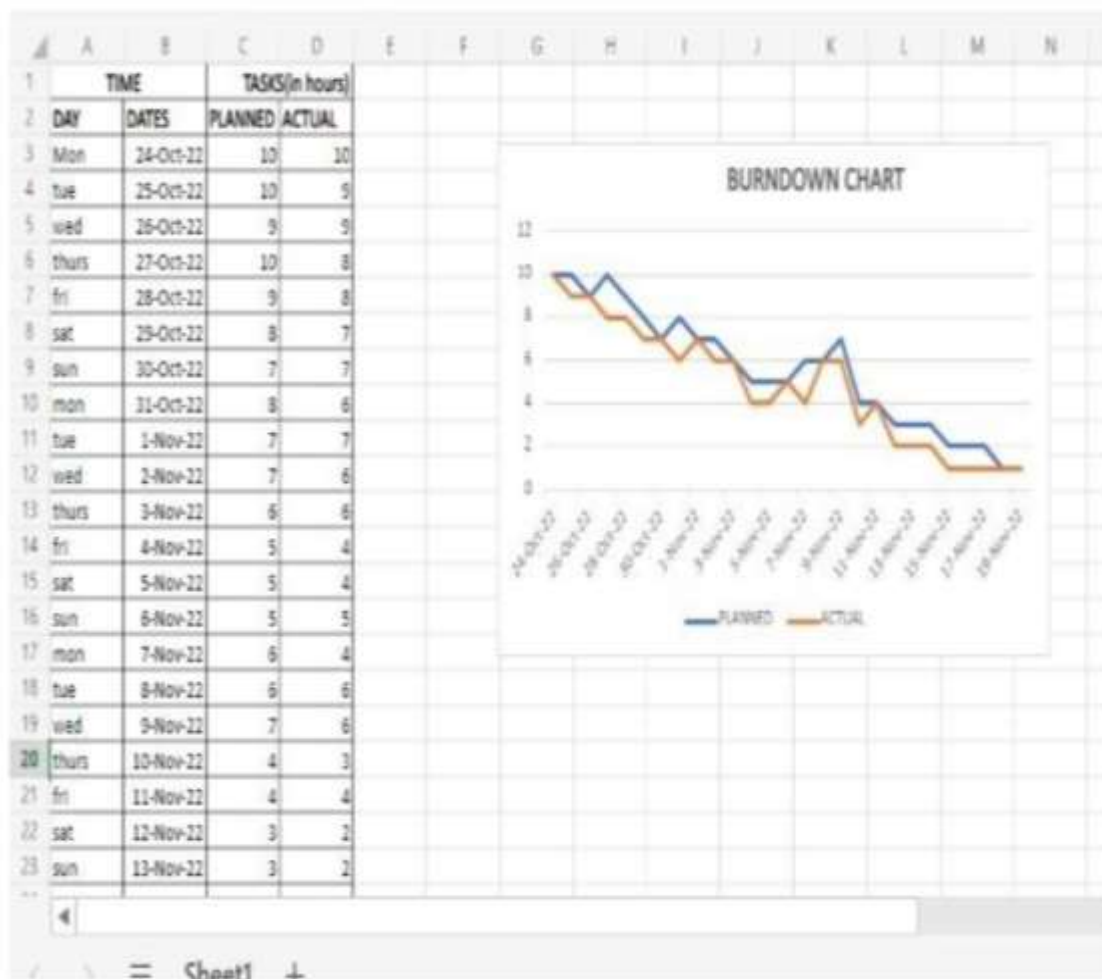
Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-4	Announcement	USN-9	When a website is scanned, the model shows whether it is a legitimate site or a phishing one.	15	High	Sadvika.c
Sprint-4	Events	USN-10	In order to use this model correctly, a website needs to be able to retrieve and display accurate results.	10	High	Harika.A.V.S

6.1 Sprint Delivery Schedule

Project Tracker, Velocity & Burndown Chart: (4 Marks)

Sprint	Total Story Points	Duration	Sprint Start Date	Sprint End Date (Planned)	Story Points Completed (as on Planned End Date)	Sprint Release Date (Actual)
Sprint-1	20	7 Days	24 Oct 2022	30 Oct 2022	19	31 Oct 2022
Sprint-2	20	5 Days	31 Oct 2022	04 Nov 2022	18	05 Nov 2022
Sprint-3	20	7 Days	05 Nov 2022	11 Nov 2022	20	11 Nov 2022
Sprint-4	20	8 Days	12 Nov 2022	19 Nov 2022	17	20 Nov 2022

6.2 Reports from JIRA



CHAPTER-7

CODING & SOLUTION

7.1 Feature 1

```
#app.py

# importing required libraries

from feature import FeatureExtraction
from flask import Flask, request, render_template
import numpy as np
import pandas as pd
from sklearn import metrics
import warnings
import pickle
warnings.filterwarnings('ignore')

file = open("model.pkl", "rb")
gbc = pickle.load(file)
file.close()

app = Flask(__name__)

@app.route("/", methods=["GET", "POST"])
def index():
    if request.method == "POST":

        url = request.form["url"]
```

```

obj = FeatureExtraction(url)
x = np.array(obj.getFeaturesList()).reshape(1, 30)

y_pred = gbc.predict(x)[0]
#1 is safe
#-1 is unsafe
y_pro_phishing = gbc.predict_proba(x)[0, 0]
y_pro_non_phishing = gbc.predict_proba(x)[0, 1]
# if(y_pred == 1):
pred = "It is {0:.2f} % safe to go ".format(y_pro_phishing*100)
return render_template('index.html', xx=round(y_pro_non_phishing, 2), url=url)
return render_template("index.html", xx=-1)

if __name__ == "__main__":
    app.run(debug=True, port=2002)

```

7.2 Feature 2

```

#feature.py
import ipaddress
import re
import urllib.request
from bs4 import BeautifulSoup
import socket
import requests
from googlesearch import search
import whois
from datetime import date, datetime
import time
from dateutil.parser import parse as date_parse

```

```
from urllib.parse import urlparse
```

```
class FeatureExtraction:
```

```
    features = []
```

```
    def __init__(self, url):
```

```
        self.features = []
```

```
        self.url = url
```

```
        self.domain = ""
```

```
        self.whois_response = ""
```

```
        self.urlparse = ""
```

```
        self.response = ""
```

```
        self.soup = ""
```

```
    try:
```

```
        self.response = requests.get(url)
```

```
        self.soup = BeautifulSoup(response.text, 'html.parser')
```

```
    except:
```

```
        pass
```

```
    try:
```

```
        self.urlparse = urlparse(url)
```

```
        self.domain = self.urlparse.netloc
```

```
    except:
```

```
        pass
```

```
    try:
```

```
        self.whois_response = whois.whois(self.domain)
```

```
    except:
```

```
        pass
```

```
self.features.append(self.UsingIp())
self.features.append(self.longUrl())
self.features.append(self.shortUrl())
self.features.append(self.symbol())
self.features.append(self.redirecting())
self.features.append(self.prefixSuffix())
self.features.append(self.SubDomains())
self.features.append(self.Hppts())
self.features.append(self.DomainRegLen())
self.features.append(self.Favicon())
```

```
self.features.append(self.NonStdPort())
self.features.append(self.HTTPSDomainURL())
self.features.append(self.RequestURL())
self.features.append(self.AnchorURL())
self.features.append(self.LinksInScriptTags())
self.features.append(self.ServerFormHandler())
self.features.append(self.InfoEmail())
self.features.append(self.AbnormalURL())
self.features.append(self.WebsiteForwarding())
self.features.append(self.StatusBarCust())
```

```
self.features.append(self.DisableRightClick())
self.features.append(self.UsingPopupWindow())
self.features.append(self.IframeRedirection())
self.features.append(self.AgeofDomain())
self.features.append(self.DNSRecording())
self.features.append(self.WebsiteTraffic())
self.features.append(self.PageRank())
self.features.append(self.GoogleIndex())
```

```
self.features.append(self.LinksPointingToPage())  
self.features.append(self.StatsReport())
```

1.UsingIp

```
def UsingIp(self):  
    try:  
        ipaddress.ip_address(self.url)  
        return -1  
    except:  
        return 1
```

2.longUrl

```
def longUrl(self):  
    if len(self.url) < 54:  
        return 1  
    if len(self.url) >= 54 and len(self.url) <= 75:  
        return 0  
    return -1
```

3.shortUrl

```
def shortUrl(self):  
    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\.g  
d|tr\.im|link\.zip\.net', self.url)
```

```
if match:  
    return -1  
  
return 1
```

```
# 4.Symbol@
```

```
def symbol(self):  
    if re.findall("@", self.url):  
        return -1  
  
    return 1
```

```
# 5.Redirecting//
```

```
def redirecting(self):  
    if self.url.rfind('/') > 6:  
        return -1  
  
    return 1
```

```
# 6.prefixSuffix
```

```
def prefixSuffix(self):  
    try:  
        match = re.findall('\-', self.domain)  
  
        if match:  
            return -1  
  
        return 1  
  
    except:  
        return -1
```

```
# 7.SubDomains
```

```
def SubDomains(self):  
    dot_count = len(re.findall("\.", self.url))
```

```
if dot_count == 1:
    return 1
elif dot_count == 2:
    return 0
return -1
```

8.HTTPS

```
def Hppts(self):
    try:
        https = self.urlparse.scheme
        if 'https' in https:
            return 1
        return -1
    except:
        return 1
```

9.DomainRegLen

```
def DomainRegLen(self):
    try:
        expiration_date = self.whois_response.expiration_date
        creation_date = self.whois_response.creation_date
        try:
            if(len(expiration_date)):
                expiration_date = expiration_date[0]
        except:
            pass
        try:
            if(len(creation_date)):
                creation_date = creation_date[0]
        except:
            pass
```

```
age = (expiration_date.year-creation_date.year)*12 + \  
      (expiration_date.month-creation_date.month)  
if age >= 12:  
    return 1  
return -1  
except:  
    return -1
```

10. Favicon

```
def Favicon(self):  
    try:  
        for head in self.soup.find_all('head'):  
            for head.link in self.soup.find_all('link', href=True):  
                dots = [x.start(0)  
                        for x in re.finditer('\.', head.link['href'])]  
                if self.url in head.link['href'] or len(dots) == 1 or domain in head.link['href']:  
                    return 1  
    return -1  
except:  
    return -1
```

11. NonStdPort

```
def NonStdPort(self):  
    try:  
        port = self.domain.split(":")  
        if len(port) > 1:  
            return -1  
    return 1  
except:  
    return -1
```

12. HTTPSDomainURL

```
def HTTPSDomainURL(self):
```

```
    try:
```

```
        if 'https' in self.domain:
```

```
            return -1
```

```
        return 1
```

```
    except:
```

```
        return -1
```

13. RequestURL

```
def RequestURL(self):
```

```
    try:
```

```
        for img in self.soup.find_all('img', src=True):
```

```
            dots = [x.start(0) for x in re.finditer('\.', img['src'])]
```

```
            if self.url in img['src'] or self.domain in img['src'] or len(dots) == 1:
```

```
                success = success + 1
```

```
            i = i+1
```

```
        for audio in self.soup.find_all('audio', src=True):
```

```
            dots = [x.start(0) for x in re.finditer('\.', audio['src'])]
```

```
            if self.url in audio['src'] or self.domain in audio['src'] or len(dots) == 1:
```

```
                success = success + 1
```

```
            i = i+1
```

```
        for embed in self.soup.find_all('embed', src=True):
```

```
            dots = [x.start(0) for x in re.finditer('\.', embed['src'])]
```

```
            if self.url in embed['src'] or self.domain in embed['src'] or len(dots) == 1:
```

```
                success = success + 1
```

```
            i = i+1
```

```

for iframe in self.soup.find_all('iframe', src=True):
    dots = [x.start(0) for x in re.finditer('\.', iframe['src'])]
    if self.url in iframe['src'] or self.domain in iframe['src'] or len(dots) == 1:
        success = success + 1
    i = i+1

try:
    percentage = success/float(i) * 100
    if percentage < 22.0:
        return 1
    elif((percentage >= 22.0) and (percentage < 61.0)):
        return 0
    else:
        return -1
except:
    return 0
except:
    return -1

```

14. AnchorURL

```

def AnchorURL(self):
    try:
        i, unsafe = 0, 0
        for a in self.soup.find_all('a', href=True):
            if "#" in a['href'] or "javascript" in a['href'].lower() or "mailto" in a['href'].lower() or not (url
in a['href'] or self.domain in a['href']):
                unsafe = unsafe + 1
            i = i + 1

    try:
        percentage = unsafe / float(i) * 100

```

```

        if percentage < 31.0:
            return 1

        elif ((percentage >= 31.0) and (percentage < 67.0)):
            return 0

        else:
            return -1

    except:
        return -1

except:
    return -1

# 15. LinksInScriptTags
def LinksInScriptTags(self):
    try:
        i, success = 0, 0

        for link in self.soup.find_all('link', href=True):
            dots = [x.start(0) for x in re.finditer('\.', link['href'])]

            if self.url in link['href'] or self.domain in link['href'] or len(dots) == 1:
                success = success + 1

            i = i+1

        for script in self.soup.find_all('script', src=True):
            dots = [x.start(0) for x in re.finditer('\.', script['src'])]

            if self.url in script['src'] or self.domain in script['src'] or len(dots) == 1:
                success = success + 1

            i = i+1

    try:
        percentage = success / float(i) * 100

```

```
        if percentage < 17.0:
            return 1
        elif((percentage >= 17.0) and (percentage < 81.0)):
            return 0
        else:
            return -1
    except:
        return 0
except:
    return -1
```

16. ServerFormHandler

```
def ServerFormHandler(self):
```

```
    try:
        if len(self.soup.find_all('form', action=True)) == 0:
            return 1
        else:
            for form in self.soup.find_all('form', action=True):
                if form['action'] == "" or form['action'] == "about:blank":
                    return -1
                elif self.url not in form['action'] and self.domain not in form['action']:
                    return 0
            else:
                return 1
    except:
        return -1
```

17. InfoEmail

```
def InfoEmail(self):
```

```
    try:
        if re.findall(r"[mail\\(\\)|mailto:?}", self.soap):
```

```
        return -1
    else:
        return 1
except:
    return -1
```

18. AbnormalURL

```
def AbnormalURL(self):
    try:
        if self.response.text == self.whois_response:
            return 1
        else:
            return -1
    except:
        return -1
```

19. WebsiteForwarding

```
def WebsiteForwarding(self):
    try:
        if len(self.response.history) <= 1:
            return 1
        elif len(self.response.history) <= 4:
            return 0
        else:
            return -1
    except:
        return -1
```

20. StatusBarCust

```
def StatusBarCust(self):
    try:
```



```
        if re.findall("<script>.+onmouseover.+</script>", self.response.text):
            return 1
        else:
            return -1
    except:
        return -1
```

21. DisableRightClick

```
def DisableRightClick(self):
    try:
        if re.findall(r"event.button ?== ?2", self.response.text):
            return 1
        else:
            return -1
    except:
        return -1
```

22. UsingPopupWindow

```
def UsingPopupWindow(self):
    try:
        if re.findall(r"alert\(", self.response.text):
            return 1
        else:
            return -1
    except:
        return -1
```

23. IframeRedirection

```
def IframeRedirection(self):
    try:
        if re.findall(r"[<iframe>|<frameBorder>]", self.response.text):
```

```
        return 1
    else:
        return -1
except:
    return -1
```

24. AgeofDomain

```
def AgeofDomain(self):
    try:
        creation_date = self.whois_response.creation_date
        try:
            if(len(creation_date)):
                creation_date = creation_date[0]
        except:
            pass

        today = date.today()
        age = (today.year-creation_date.year) * \
            12+(today.month-creation_date.month)
        if age >= 6:
            return 1
        return -1
    except:
        return -1
```

25. DNSRecording

```
def DNSRecording(self):
    try:
        creation_date = self.whois_response.creation_date
        try:
            if(len(creation_date)):
```

```

        creation_date = creation_date[0]
    except:
        pass

    today = date.today()
    age = (today.year-creation_date.year) * \
        12+(today.month-creation_date.month)
    if age >= 6:
        return 1
    return -1
except:
    return -1

# 26. WebsiteTraffic
def WebsiteTraffic(self):
    try:
        rank = BeautifulSoup(urllib.request.urlopen(
            "http://data.alexa.com/data?cli=10&dat=s&url=" + url).read(),
            "xml").find("REACH")['RANK']
        if (int(rank) < 100000):
            return 1
        return 0
    except:
        return -1

# 27. PageRank
def PageRank(self):
    try:
        prank_checker_response = requests.post(
            "https://www.checkpagerank.net/index.php", {"name": self.domain})

```

```

        global_rank = int(re.findall(
            r"Global Rank: ([0-9]+)", rank_checker_response.text)[0])
    if global_rank > 0 and global_rank < 100000:
        return 1
    return -1
except:
    return -1

# 28. GoogleIndex

def GoogleIndex(self):
    try:
        site = search(self.url, 5)
        if site:
            return 1
        else:
            return -1
    except:
        return 1

# 29. LinksPointingToPage

def LinksPointingToPage(self):
    try:
        number_of_links = len(re.findall(r"<a href=", self.response.text))
        if number_of_links == 0:
            return 1
        elif number_of_links <= 2:
            return 0
        else:
            return -1
    except:
        return -1

```

30. StatsReport

def StatsReport(self):

try:

url_match = re.search(

'at\.\ua|usa\.\cc|baltazarpresentes\.\com|.br|pe\.\hu|esy\.\es|hol\.\es|sweddy\.\com|myjino\.\ru|96\.\lt
|ow\.\ly', url)

ip_address = socket.gethostbyname(self.domain)

ip_match =

re.search('146\.\112\.\61\.\108|213\.\174\.\157\.\151|121\.\50\.\168\.\88|192\.\185\.\217\.\116|78\.\46\.\21
1\.\158|181\.\174\.\165\.\13|46\.\242\.\145\.\103|121\.\50\.\168\.\40|83\.\125\.\22\.\219|46\.\242\.\145\.\98
|'

'107\.\151\.\148\.\44|107\.\151\.\148\.\107|64\.\70\.\19\.\203|199\.\184\.\144\.\27|107\.\151\.\148\.\108|10
7\.\151\.\148\.\109|119\.\28\.\52\.\61|54\.\83\.\43\.\69|52\.\69\.\166\.\231|216\.\58\.\192\.\225|'

'118\.\184\.\25\.\86|67\.\208\.\74\.\71|23\.\253\.\126\.\58|104\.\239\.\157\.\210|175\.\126\.\123\.\219|141\
.8\.\224\.\221|10\.\10\.\10\.\10|43\.\229\.\108\.\32|103\.\232\.\215\.\140|69\.\172\.\201\.\153|'

'216\.\218\.\185\.\162|54\.\225\.\104\.\146|103\.\243\.\24\.\98|199\.\59\.\243\.\120|31\.\170\.\160\.\61|213
\.\19\.\128\.\77|62\.\113\.\226\.\131|208\.\100\.\26\.\234|195\.\16\.\127\.\102|195\.\16\.\127\.\157|'

'34\.\196\.\13\.\28|103\.\224\.\212\.\222|172\.\217\.\4\.\225|54\.\72\.\9\.\51|192\.\64\.\147\.\141|198\.\200\
.56\.\183|23\.\253\.\164\.\103|52\.\48\.\191\.\26|52\.\214\.\197\.\72|87\.\98\.\255\.\18|209\.\99\.\17\.\27|'

'216\.\38\.\62\.\18|104\.\130\.\124\.\96|47\.\89\.\58\.\141|78\.\46\.\211\.\158|54\.\86\.\225\.\156|54\.\82\.\1
56\.\19|37\.\157\.\192\.\102|204\.\11\.\56\.\48|110\.\34\.\231\.\42', ip_address)

if url_match:

return -1

elif ip_match:

return -1

return 1

except:

return 1

def getFeaturesList(self):

return self.features

CHAPTER 8

TESTING

8.1 Test Cases

[illegible]

8.2 User Acceptance Testing

UAT Execution & Report Submission

1. Purpose of Document

The purpose of this document is to briefly explain the test coverage and open issues of the [Web Phishing Detection] project at the time of the release to User Acceptance Testing (UAT).

2. Defect Analysis

This report shows the number of resolved or closed bugs at each severity level, and how they were resolved

Resolution	Severity 1	Severity 2	Severity 3	Severity 4	Subtotal
By Design	10	4	2	3	20
Duplicate	1	0	3	0	4
External	2	3	0	1	6
Fixed	10	2	4	20	36
Not Reproduced	0	0	1	0	1
Skipped	0	0	0	0	0
Won't Fix	0	0	2	1	3
Totals	23	9	12	25	60

1. Test Case Analysis



This report shows the number of test cases that have passed, failed, and untested

Section	Total Cases	Not Tested	Fail	Pass
Print Engine	10	0	0	10
Client Application	50	0	0	50
Security	5	0	0	4
Outsource Shipping	3	0	0	3
Exception Reporting	10	0	0	9
Final Report Output	10	0	0	10
Version Control	4	0	0	4

CHAPTER 9

RESULTS

9.1 Performance Metrics

S.No.	Parameter	Values	Screenshot
1.	Metrics	Classification Model: Gradient Boosting Classification Accuracy Score- 97.4%	
2.	Tune the Model	Hyperparameter Tuning - 97% Validation Method – KFOLD & Cross Validation Method	

1. METRICS:

CLASSIFICATION REPORT:

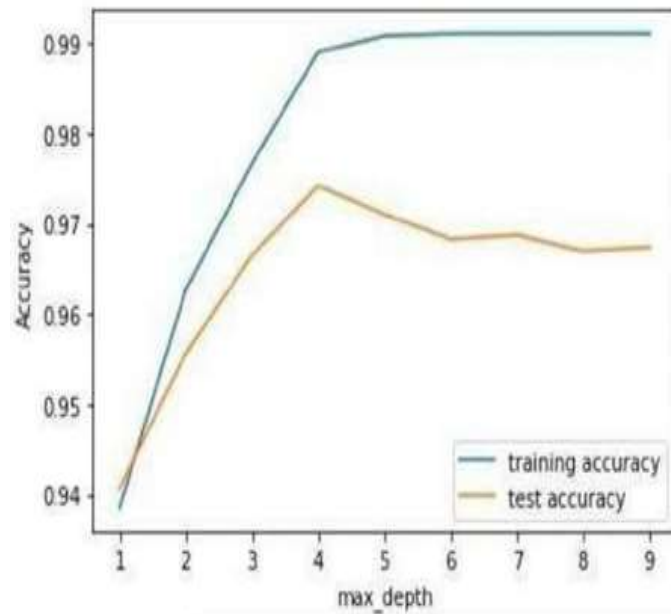
In [52]: *#computing the classification report of the model*

```
print(metrics.classification_report(y_test, y_test_gbc))
```

	precision	recall	f1-score	support
-1	0.99	0.96	0.97	976
1	0.97	0.99	0.98	1235
accuracy			0.97	2211
macro avg	0.98	0.97	0.97	2211
weighted avg	0.97	0.97	0.97	2211

PERFORMANCE:

PERFORMANCE :



Out[83]:

	ML Model	Accuracy	f1_score	Recall	Precision
0	Gradient Boosting Classifier	0.974	0.977	0.994	0.986
1	CatBoost Classifier	0.972	0.975	0.994	0.989
2	Random Forest	0.969	0.972	0.992	0.991
3	Support Vector Machine	0.964	0.968	0.980	0.965
4	Decision Tree	0.958	0.962	0.991	0.993
5	K-Nearest Neighbors	0.956	0.961	0.991	0.989
6	Logistic Regression	0.934	0.941	0.943	0.927
7	Naive Bayes Classifier	0.605	0.454	0.292	0.997
8	XGBoost Classifier	0.548	0.548	0.993	0.984
9	Multi-layer Perceptron	0.543	0.543	0.989	0.983

2. TUNE THE MODEL – HYPERPARAMETER TUNING

```
In [58]: #HYPERPARAMETER TUNING
grid.fit(X_train, y_train)
```

```
Out[58]: GridSearchCV
GridSearchCV(cv=5,
             estimator=GradientBoostingClassifier(learning_rate=0.7,
                                                    max_depth=4),
             param_grid={'max_features': array([1, 2, 3, 4, 5]),
                         'n_estimators': array([ 10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 110, 120, 130,
140, 150, 160, 170, 180, 190, 200])})
```

```
estimator: GradientBoostingClassifier
GradientBoostingClassifier(learning_rate=0.7, max_depth=4)
GradientBoostingClassifier(learning_rate=0.7, max_depth=4)
```

```
In [59]: print("The best parameters are %s with a score of %.2f"
              % (grid.best_params_, grid.best_score_))
```

The best parameters are {'max_features': 5, 'n_estimators': 200} with a score of 0.97

VALIDATION METHODS: KFOLD & Cross Folding

Wilcoxon signed-rank test

```
In [78]: #KFOLD and Cross Validation Model

from scipy.stats import wilcoxon
from sklearn.datasets import load_iris
from sklearn.ensemble import GradientBoostingClassifier
from xgboost import XGBClassifier
from sklearn.model_selection import cross_val_score, KFold

# Load the dataset
X = load_iris().data
y = load_iris().target

# Prepare models and select your CV method
model1 = GradientBoostingClassifier(n_estimators=100)
model2 = XGBClassifier(n_estimators=100)
kf = KFold(n_splits=20, random_state=None)
# Extract results for each model on the same folds
results_model1 = cross_val_score(model1, X, y, cv=kf)
results_model2 = cross_val_score(model2, X, y, cv=kf)
stat, p = wilcoxon(results_model1, results_model2, zero_method='split');
stat
```

Out[78]: 95.0

5x2CV combined F test

```
In [89]: from mlxtend.evaluate import combined_ftest_5x2cv
from sklearn.tree import DecisionTreeClassifier, ExtraTreeClassifier
from sklearn.ensemble import GradientBoostingClassifier
from mlxtend.data import iris_data

# Prepare data and clfs
X, y = iris_data()
clf1 = GradientBoostingClassifier()
clf2 = DecisionTreeClassifier()

# Calculate p-value
f, p = combined_ftest_5x2cv(estimator1=clf1,
                             estimator2=clf2,
                             X=X, y=y,
                             random_seed=1)

print('f-value:', f)
print('p-value:', p)

f-value: 1.727272727272733
p-value: 0.2840135734291782
```

CHAPTER -10

Advantages of web phishing detection

1. It Takes a Load off the Security Team
2. It Offers a Solution, not a Tool
3. If internet connection fails this system will work
4. Improve on Inefficiencies of SEG and Phishing Awareness Training
5. This system can be used by many e-commerce websites in order to have good customer relationships.
6. Separate You from Your Competitors.

Disadvantages of web phishing detection

1. All website related data will be stored in one place.
2. It is a very time-consuming process.
3. loss of money, loss of intellectual property, damage to reputation, and disruption of operational activities.

CHAPTER 11

CONCLUSION

It is outstanding that a decent enemy of phishing apparatus ought to anticipate the phishing assaults in a decent timescale. We accept that the accessibility of a decent enemy of phishing device at a decent time scale is additionally imperative to build the extent of anticipating phishing sites. This apparatus ought to be improved continually through consistent retraining. As a matter of fact, the accessibility of crisp and cutting-edge preparing dataset which may gained utilizing our very own device [30, 32] will help us to retrain our model consistently and handle any adjustments in the highlights, which are influential in deciding the site class. Albeit neural system demonstrates its capacity to take a wide assortment of classification issues, the procedure of finding the ideal structure is very difficult, and much of the time, this structure is controlled by experimentation. Our model takes care of this issue via computerizing the way toward organizing a neural system conspire; hence, on the off chance that we construct an enemy of phishing model and for any reasons we have to refresh it, at that point our model will encourage this procedure, that is, since our model will mechanize the organizing procedure and will request scarcely any client defined parameters. Phishing URL detection plays a pivotal role for many cybersecurity software and applications. In this paper, we researched and reviewed works based on the advanced machine learning techniques and approaches that promise a fresh approach in this domain. This article includes summary of the reviewed works after a systematic and comprehensive study on Phishing Website Detection systems. We believe that the presented survey would help researchers and developers with the insight of the progress achieved in the past years. Despite the tremendous progress in the field of cybersecurity, phishing website detection still pose a challenging problem with every evolving technology and techniques.

CHAPTER-12

Future Scope

There is a scope for future development of this project. We will implement this using advanced deep learning method to improve the accuracy and precision. Enhancements can be done in an efficient manner. Thus, the project is flexible and can be enhanced at any time with more advanced features.

CHAPTER-13

Appendix:

1. Application Building
2. Collection of Dataset
3. Data Pre-processing
4. Integration of Flask App with IBM Cloud
5. Model Building
6. Performance Testing
7. Training the model on IBM
8. User Acceptance Testing
9. Ideation Phase
10. Preparation Phase
11. Project Planning
12. Performance Testing
13. User Acceptance Testing

Project Link: [https://github.com/ IBM-EPBL/IBM-Project-32654-1660211183](https://github.com/IBM-EPBL/IBM-Project-32654-1660211183)