WEB PHISHING DETECTION

NALAIYA THIRAN PROJECT REPORT

2022

Submitted By

Chamala Sudheshna	**310619205020**
Dasari Shimmy Roy	**310619205023**
Gavin Gladston A	**310619205028**
Gurran Balaji	**310619205032**

Team ID - PNT2022TMID35524

INDEX

Project Report Format

1. INTRODUCTION

1.1 Project Overview

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.

Common threats of web phishing:

- Web phishing aims to steal private information, such as usernames, passwords, and credit card details, by way of impersonating a legitimate entity.
- It will lead to information disclosure and property damage.
- Large organizations may get trapped in different kinds of scams.

This Guided Project mainly focuses on applying a machine-learning algorithm to detect Phishing websites.

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.

1.2 Purpose

The importance to safeguard online users from becoming victims of online fraud, divulging confidential information to an attacker among other effective uses of phishing as an attacker's tool, phishing detection tools play a vital role in ensuring a secure online experience for users.

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. These effects work together to cause loss of company value, sometimes with irreparable repercussions.

2. LITERATURE SURVEY

2.1 Existing problem

Phishing Detection techniques do suffer low detection accuracy and high false alarm especially when novel phishing approaches are introduced. Besides, the most common technique used, blacklist-based method is inefficient in responding to emanating phishing attacks since registering new domain has become easier, no comprehensive blacklist can ensure a perfect up-to-date database.

2.2 Reference

- TITLE: Phishing Web Page Detection Methods: URL and HTML Features Detection

  AUTHORS: Humam, Faris, Setiadi, Yazid.

  ABSTRACT:

Phishing is a form of online fraud where fraudulent web pages replicate real web pages and attempt to entice users into giving private and confidential information to the phisher. According to the numbers provided by APWG and Phistank, there will likely be a rise in phishing sites between 2015 and 2020. Numerous studies have been conducted to find solutions to this issue by utilising a variety of techniques. Sadly, the use of many approaches was concluded to be ineffective since design and assessment are solely concerned with achieving accuracy in detection and not with application in the actual world. A security detecting device should, however, be deployable, effective, and have good performance. In this study, the authors assessed a number of approaches and put forth rule-based software programs that are more effective in phishing detection.

- TITLE: Phishing Detection in Websites using Parse Tree Validation

  AUTHORS: C. Emilin Shyni, Anesh D Sundar, G.S.Edwin Ebby.

  ABSTRACT:

Phishing is a method of deceiving people into divulging personal information, such as usernames and passwords, credit card information, sensitive bank information, etc., through the use of spoof emails, instant messages, or phoney websites with a realistic-looking design. This study proposes a method for determining whether a webpage is real or phishing, known as parse tree validation. By capturing every hyperlink on a page using the Google API and building a parse tree out of the hyperlinks, it is a new way to identify phishing websites. Thousand phishing pages and thousand genuine pages are used to test this strategy. A false negative rate of 7.3% and a false positive rate of 5.2% was achieved.

- TITLE: WC-PAD: Web Crawling based Phishing Attack Detection

  AUTHORS: Nathezhtha, Sangeetha, Vaidehi.

  ABSTRACT:

Phishing is a crime that involves the stealing of users' private information. Phishing websites target people, businesses, hosting services for cloud storage, and official websites. Although software-based approaches to anti-phishing are preferable owing to cost and operational considerations, hardware-based alternatives are still often deployed. The existing methods for phishing detection are unable to address issues like zero-day phishing website attacks. A three-phase attack detection system called the Web Crawler based Phishing Attack Detector (WC-PAD) has been presented to address these problems and accurately detect the existence of phishing. It classifies phishing and non-phishing websites based on input factors such as web traffic, web content, and Uniform Resource Locators (URLs). With datasets gathered from actual phishing situations, the proposed WC-experimental PAD's study is carried out. According to the testing results, the suggested WCPAD provides 98.9% accuracy in phishing attack detection, including zero-day phishing attacks.

- TITLE: Phishing Detection from URLs Using Deep Learning Approach

  AUTHORS: Shweta Singh, M.P. Singh, Ramprakash Pandey

  ABSTRACT:

The Internet is now accessible everywhere. People enjoy using an e-commerce platform to buy or sell their goods all over the world. As a result, cyberattackers now gravitate toward crimes in cyberspace. Phishing is one such strategy where attackers/criminals have used the unidentified structure of the Internet with the intention of misleading people with the use of the fictitious website and emails in order to gain their credentials (like account numbers, passwords, and PINs). Due to this semantic framework, it might be difficult to tell whether a web page is real or phishing. In order to stop such attempts, a phishing detection system is created in this work using deep learning methods. Convolutional neural networks (CNNs) are used by the system to analyse URLs in order to identify phishing websites. Our proposed system's accuracy was better than the prior model in paper [19], which had a 97.98% accuracy rating, at 98.00%. As the CNN automatically extracts features from the URLs through its hidden layers, this system doesn't require any feature engineering. This is another benefit of the proposed approach above that which was previously disclosed in [19], as feature engineering takes a lot of effort.

- TITLE: A Deep Learning-Based Framework for Phishing Website Detection

  AUTHORS: LIZHEN TANG, QUSAY H. MAHMOUD

  ABSTRACT:

Phishing attackers spread phishing links through e-mail, text messages, and social media platforms. They use social engineering skills to trick users into visiting phishing websites and entering crucial personal information. In the end, the stolen personal information is used to defraud the trust of regular websites or financial institutions to obtain illegal benefits. With the development and applications of machine learning technology, many machine learning-based solutions for detecting phishing have been proposed. Some solutions are based on the features extracted by rules, and some of the features need to rely on third-party

services, which will cause instability and time-consuming issues in the prediction service. In this paper, a deep learning-based framework for phishing website detection is proposed. When a user visits a website, the framework has been implemented as a browser plug-in that can alert them if there is a phishing danger in real time. The real-time prediction service integrates a number of techniques, such as whitelist filtering, blacklist interception, and machine learning (ML) prediction, to increase accuracy, lower false alarm rates, and shorten computation times. We compared various machine learning models utilizing various datasets in the ML prediction module. The RNN-GRU model has the highest accuracy of 99.18% according to the trial findings, proving the viability of the suggested approach.

2.3 Problem Statement Definition

There are e-banking websites that requests the users to provide more sensitive information such as credit card details, password etc., for malicious reasons. These websites that mimics trustful URLs and webpages are known as phishing websites. Common causes for web phishing attacks involve:

- Users lack of security awareness
- Not performing sufficient due diligence
- Low-cost phishing and ransomware tools are easy to get hold of
- Malware is becoming more sophisticated and so on

Web phishing is considered to be a threat in various aspects of security on the internet, which might involve scams and private information disclosure.

Some of the common threats of web phishing are:

- Attempt to fraudulently solicit personal information from an individual or organization.
- Attempt to deliver malicious software by posing as a trustworthy organization or entity.
- Installing those malwares infects the data that cause a data breach or even nature's forces that takes down your company's data headquarters, disrupting access.

For this purpose, the objective of our project involves building an efficient and intelligent system to detect such websites by applying a machine-learning algorithm which implements classification algorithms and techniques to extract the phishing datasets criteria to classify their legitimacy and as a result of which whenever a user makes a transaction online and 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.

Problem Statement (PS)	I am	I'm trying to	But	Because	Which makes me feel
PS-1	Internet user	Browse the website	I identify the malicious activity	An attacker makes a malicious attack	Unsafe about the information which was shared by me in the website
PS-2	Business user	Checks the emails in the server	I identify the scam related activity	Which are not securely authenticated	Emails are unverified and involves third party activities

This project can be further extended by creating a browser extension or develop a GUI which takes the URL and analyze its nature to determine if it is a legitimate or a phishing website.

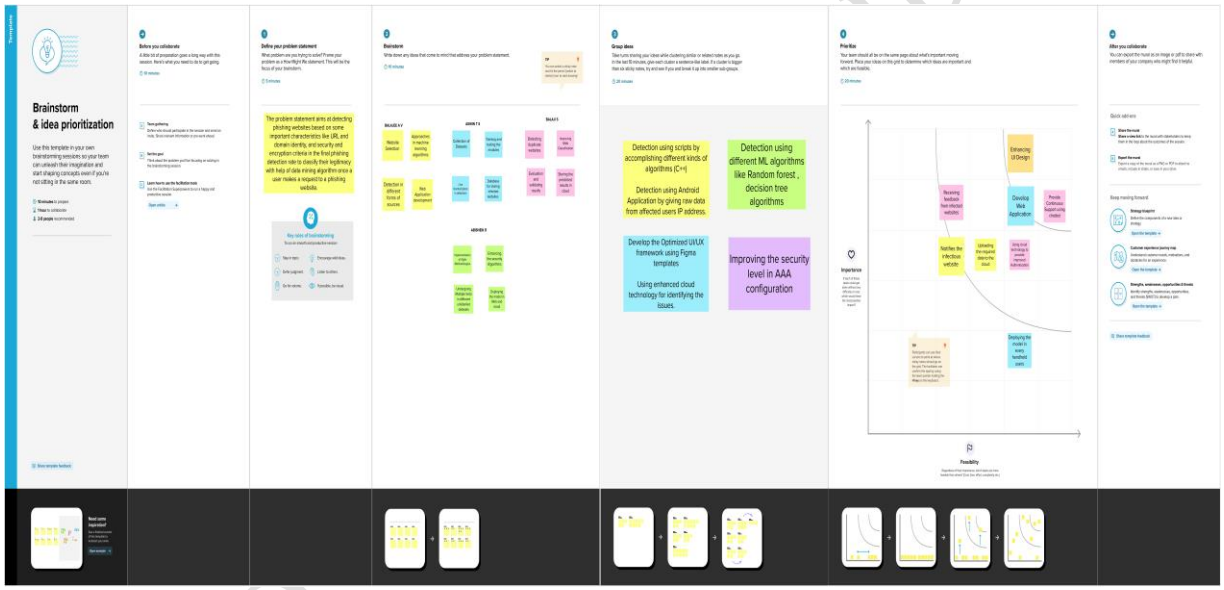
3. IDEATION & PROPOSED SOLUTION

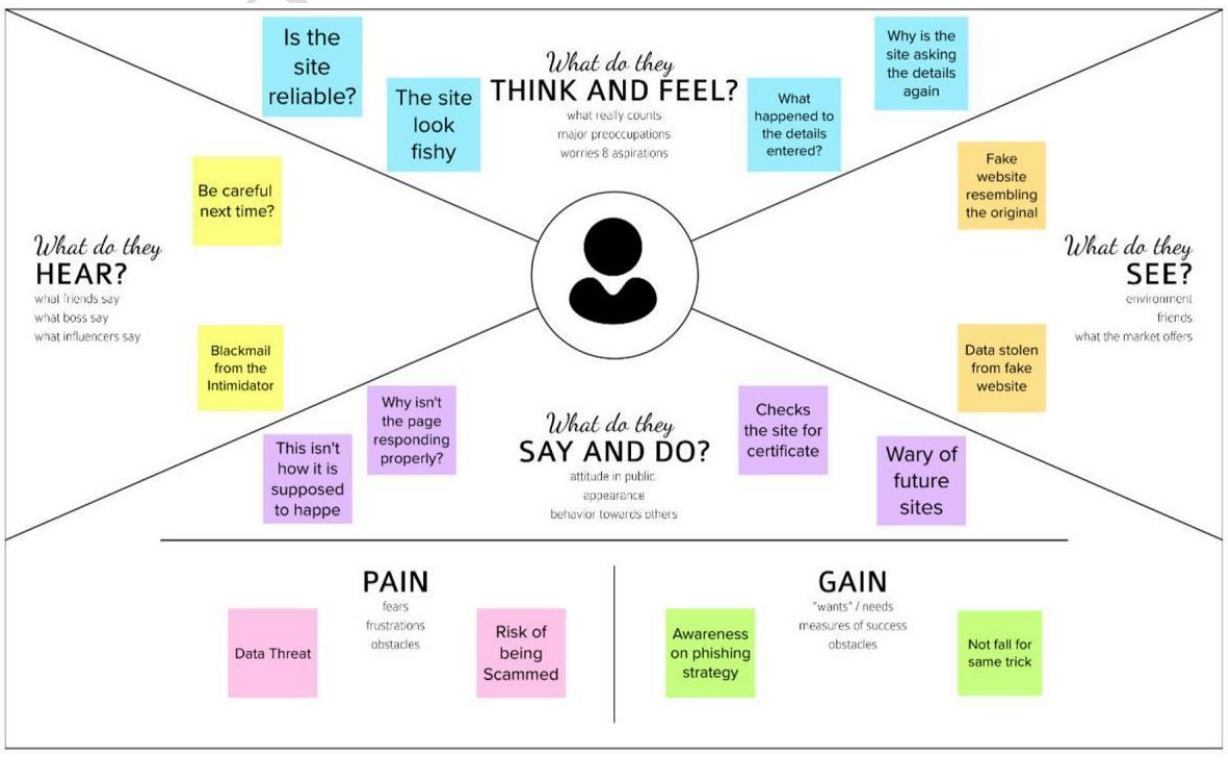
3.1 Empathy Map Canvas

An empathy map is a simple, easy-to-digest visual that captures knowledge about a user's behaviors and attitudes. It is a useful tool to helps teams better understand their users.

Creating an effective solution requires understanding the true problem and the person who is experiencing it. The exercise of creating the map helps participants consider things from the user's perspective along with his or her goals and challenges.

3.2 Ideation & Brainstorming





3.3 Proposed Solution

S. No.	Parameter	Description
1.	Problem Statement (Problem to be solved)	To reduce the people falling for web phishing scams by creating a sophisticated tool that classifies a website as malicious or safe to use.
2.	Idea / Solution description	Our solution is to build an efficient and intelligent system to detect phishing sites by applying a machine learning algorithm which implements classification algorithms and techniques to extract the phishing datasets criteria to classify their legitimacy.
3.	Novelty / Uniqueness	Uses an Ensemble model. Explores weighted features for Neural Network approaches, Extensive feature extraction strategy from the URL Simple and Easy-to-Understand UI.
4.	Social Impact / Customer Satisfaction	By using this application, the customer has the sense of safety whenever he attempts to provide sensitive information to a site.
5.	Business Model (Revenue Model)	This developed model can be used as an enterprise application by organizations which handles sensitive information and also can be sold to government agencies to prevent the loss of potential important data.
6.	Scalability of the Solution	Solution can use additional hardware resources when the number of users and activity is increased. The API can ensure that multiple requests at the same time are handled in a parallel fashion.

13.1 Problem Solution fit

Define CS, fit into CC

Focus on J&P, tap into BE, understand RC

Identify strong TR & EM

1. CUSTOMER SEGMENT(S) CS

A netizen who is willing to buy online products.

An enterprise user surfing throughthe internet for information.

6. CUSTOMER CONSTRAINTS CC

Customers have very little awareness on phishing websites.

5. AVAILABLE SOLUTIONS AS

Which solutions are available

The existing solutions are blocking such phishing sites and by triggering a message to the customer about dangerous nature of the website.

But the blocking of phishing sites is not effective as the attackers use a different/new site to steal potential data. Thus, an AI/ML model can be used to prevent customers from these kinds of sites which steals data

2. JOBS-TO-BE-DONE / PROBLEMS J&P

The phishing websites must be detected in an earlier stage.

The user can be blocked from entering such sites for the prevention of such issues.

9. PROBLEM ROOT CAUSE RC

The hackers use new ways to cheat the internet users.

Very limited research is performed on this part of the internet.

7. BEHAVIOUR BE

The option to check the legitimacy of the Websites is provided.

Users get an idea about what to do and moreimportantly what not to do.

Focus on J&P, tap into BE, understand RC

Identify strong TR & EM

3. TRIGGERS TR

A trigger message can be popped warningthe user about the site.

Phishing sites can be blocked by the ISP and can show a "site is blocked" or "phishing sitedetected" message.

4. EMOTIONS: BEFORE / AFTER EM

How do customers feel when they face a problem or a joband afterwards?

The customers feel lost and insecure to use the internet afterfacing such issues.

Unwanted panicking of the customers is felt after encounterloss of potential data to such sites.

10. YOUR SOLUTION SL

An option for the users to check the legitimacy of the websites is provided.

This increases the awareness among users and prevents misuse of data, data theft etc.,

8. CHANNELS of BEHAVIOUR CH

8.1 ONLINE

Customers tend to lose their data to phishing sites.

8.2 OFFLINE

Customers try to learn about the ways they get cheated from various resources viz., books, other people etc.,

Identify strong TR & EM

4. REQUIREMENT ANALYSIS

4.1 Functional requirement

FR NO.	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
FR-1	User Input	User inputs an URL in required field to check its validation.
FR-2	Website Comparison	Model compares the websites using Blacklist and White list approach.
FR-3	Feature extraction	After comparing, if none found on comparison then it extracts feature using heuristic and visual similarity approach.
FR-4	Prediction	Model predicts the URL using Machine Learning algorithms such as Logistic Regression, KNN
FR-5	Classifier	Model sends all output to classifier and produces final result.
FR-6	Announcement	Model then displays whether website is a legal site or a phishing site.
FR-7	Events	This model needs the capability of retrieving and displaying accurate result for a website

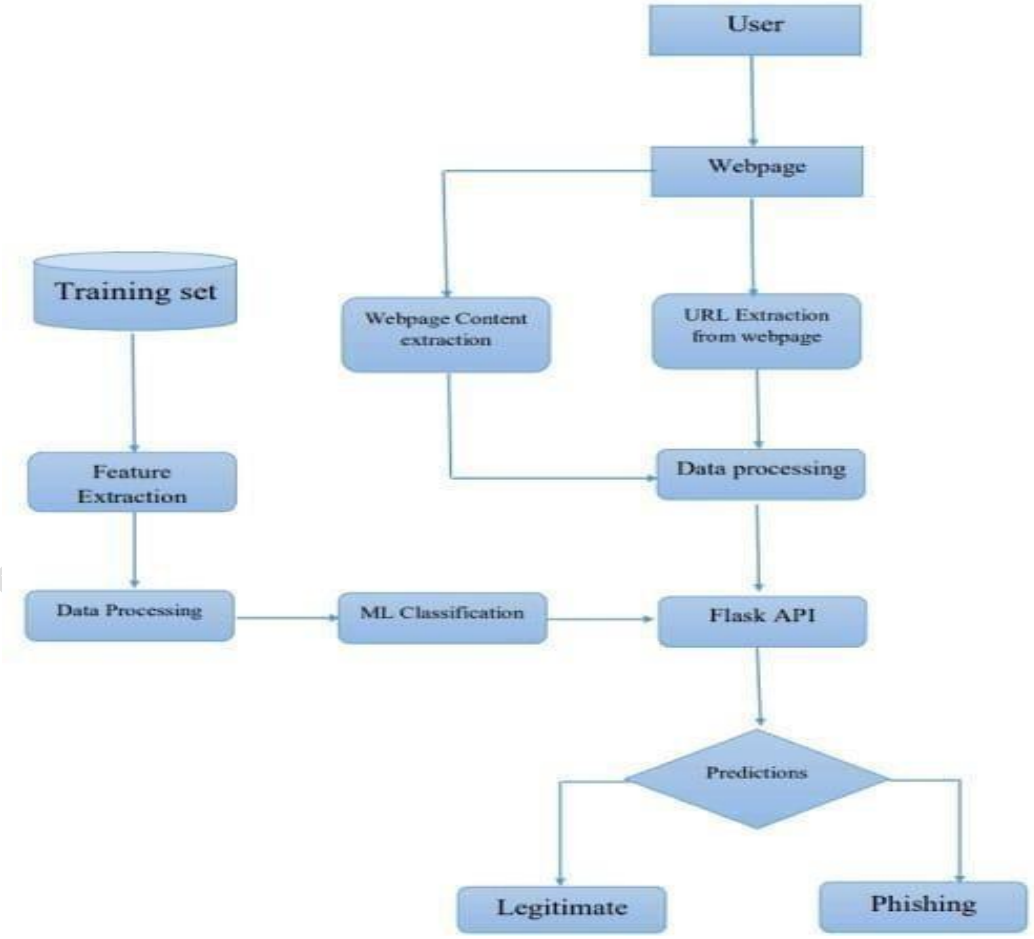
4.2 Non-Functional requirements

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

FR No.	Non-Functional Requirement	Description
NFR-1	**Usability**	Responsive UI / UX Design and users can easily configure the settings based on their preference.
NFR-2	**Security**	Implementation of Updated security algorithms and techniques.
NFR-3	**Reliability**	Reliability Factor determines the possibility of a suspected site to be Valid or Fake.
NFR-4	**Performance**	The two main characteristics of a phishing site are that it looks extremely similar to a legitimate site and that it has at least one field to enable users to input their credentials.
NFR-5	**Availability**	It occurs when an attacker, masquerading as a trusted entity, dupes a victim into opening an email, instant message, or text message.
NFR-6	**Scalability**	Scalable detection and isolation of phishing, 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.

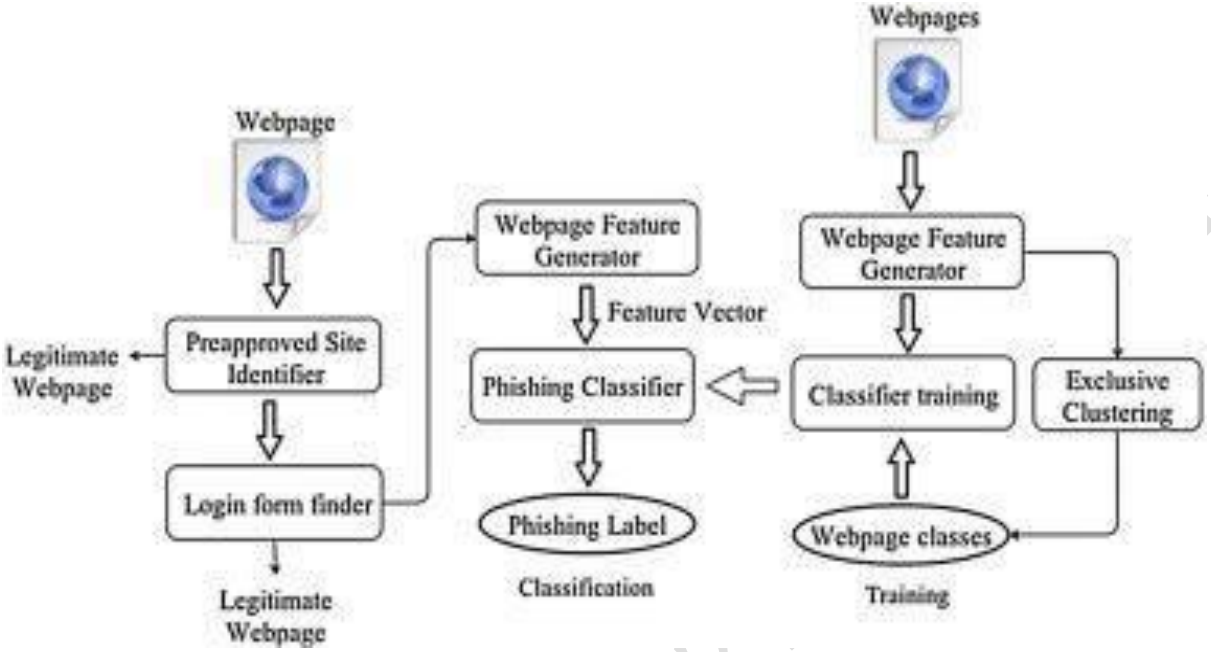
5. PROJECT DESIGN

5.1 Data Flow Diagrams

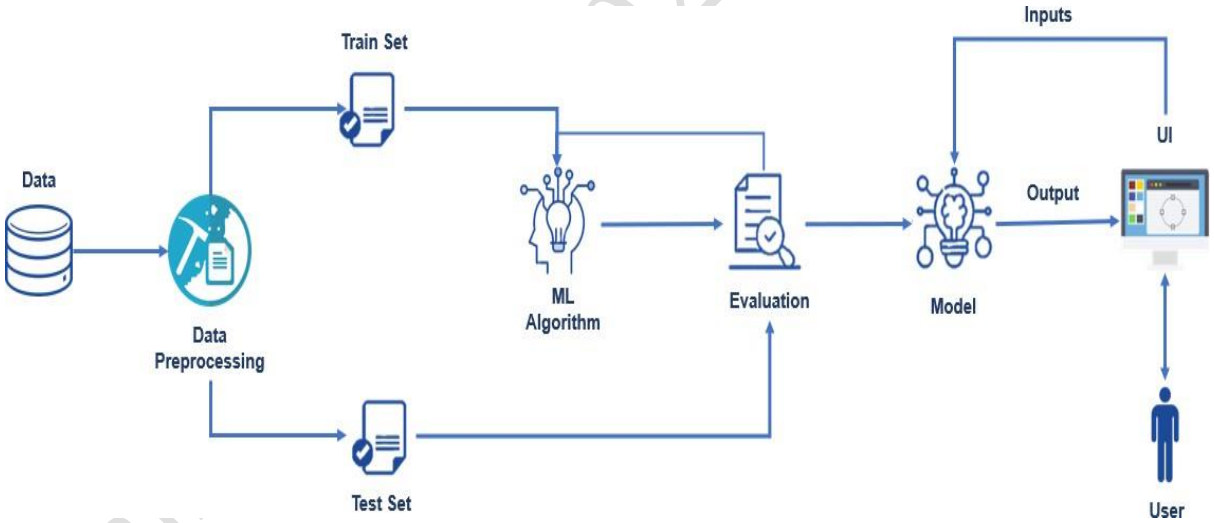


5.2 Solution & Technical Architecture

Solution Architecture:



Technical Architecture:



5.3 User Stories

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
Customer (Mobile user)	Registration	USN-1	As a user, I can register for the application by entering my email, password, and confirming my password.	I can access my account / dashboard	High	Sprint-1
		USN-2	As a user, I will receive confirmation email once I have registered for the application	I can receive confirmation email & click confirm	High	Sprint-1
		USN-3	As a user, I can register for the application through Facebook	I can register & access the dashboard with Facebook Login	Low	Sprint-2
		USN-4	As a user, I can register for the application through Gmail		Medium	Sprint-1
	Login	USN-5	As a user, I can log into the application by entering email & password		High	Sprint-1
	Dashboard					
Customer (Web user)	User input	USN-1	As a user can input the particular URL in the required field and waiting for validation.	I can go access the website without any problem	High	Sprint-1
Customer Care Executive	Feature extraction	USN-1	After it compares in case if none found on comparison then we can extract feature using heuristic and visual similarity approach.	As a User i can have comparison between websites for security.	High	Sprint-1
Administrator	Prediction	USN-1	Here the Model will predict the URL websites using Machine Learning algorithms such as Logistic Regression, KNN	In this it can have correct prediction on the particular algorithms	High	Sprint-1
	Classifier	USN-2	Here it will send all the model output to classifier in order to produce final result.	In this it will find the correct classifier for producing the result	Medium	Sprint-2

6. PROJECT PLANNING & SCHEDULING

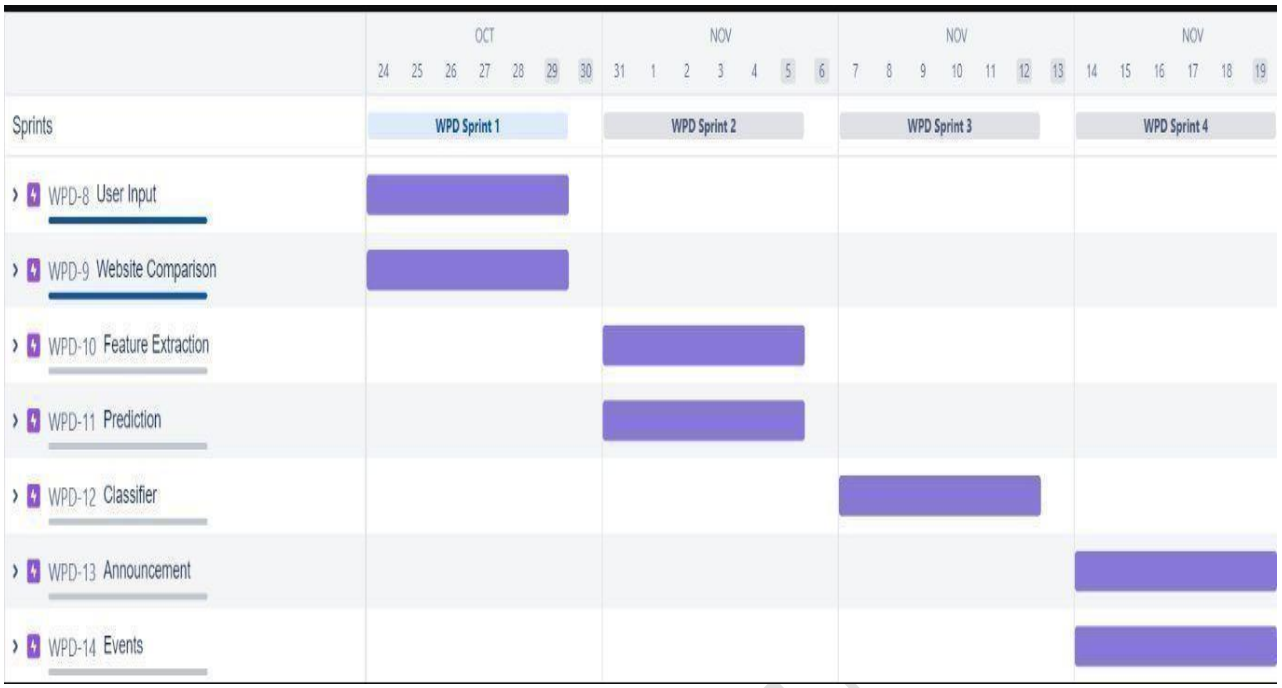
6.1 Sprint Planning & Estimation

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority
Sprint-1	User input	USN-1	User inputs an URL in the required field to check its validation.	1	Medium
Sprint-1	Website Comparison	USN-2	Model compares the websites using Blacklist and Whitelist approach.	1	High
Sprint-2	Feature Extraction	USN-3	After comparison, if none found on comparison then it extract feature using heuristic and visual similarity.	2	High
Sprint-2	Prediction	USN-4	Model predicts the URL using Machine learning algorithms such as logistic Regression, KNN, Tree Regression.	1	Medium
Sprint-3	Classifier	USN-5	Model sends all the output to the classifier and produces the final result and predict.	1	Medium
Sprint-4	Announcement	USN-6	Model then displays whether the website is legal site or a phishing site.	1	High
Sprint-4	Events	USN-7	This model needs the capability of retrieving and displaying accurate result for a website.	1	High

6.2 Sprint Delivery Schedule

Sprint	Total Story Points	Duration	Sprint Start Date	Sprint End Date (Planned)	Story Points Completed (as on planned end date)	Sprint End Date (Actual)
Sprint-1	20	6 days	24 October 2022	29 October 2022	20	12 November 2022
Sprint-2	20	6 days	31 October 2022	05 November 2022	20	14 November 2022
Sprint-3	20	6 days	07 November 2022	12 November 2022	20	16 November 2022
Sprint-4	20	6 days	14 November 2022	19 November 2022	20	19 November 2022

6.3 Reports from JIRA



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

7.1 Feature 1

```
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')
from feature import FeatureExtraction
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)
```

```
30          try:
31              self.urlparse = urlparse(url)
32              self.domain = self.urlparse.netloc
33          except:
34              pass
35          try:
36              self.whois_response = whois.whois(self.domain)
37          except:
38              pass
39          self.features.append(self.UsingIp())
40          self.features.append(self.longUrl())
41          self.features.append(self.shortUrl())
42          self.features.append(self.symbol())
43          self.features.append(self.redirecting())
44          self.features.append(self.prefixSuffix())
45          self.features.append(self.SubDomains())
46          self.features.append(self.Hppts())
47          self.features.append(self.DomainRegLen())
48          self.features.append(self.Favicon())
49          self.features.append(self.NonStdPort())
50          self.features.append(self.HTTPSDomainURL())
51          self.features.append(self.RequestURL())
52          self.features.append(self.AnchorURL())
53          self.features.append(self.LinksInScriptTags())
54          self.features.append(self.ServerFormHandler())
55          self.features.append(self.InfoEmail())
56          self.features.append(self.AbnormalURL())
57          self.features.append(self.WebsiteForwarding())
58          self.features.append(self.StatusBarCust())
```

```python
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
```

```python
# 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\.gd|tr\.im|link\.zip\.net')
    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
```

```
    # 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
```

```
    # 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:
```

```
    # 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:
```

```
    # 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'] 
                    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'])]
```

```
    # 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:
```

```
    # 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
```

```
    # 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
            if (int(rank) < 100000):
                return 1
            return 0
        except :
            return -1

    # 27. PageRank
```

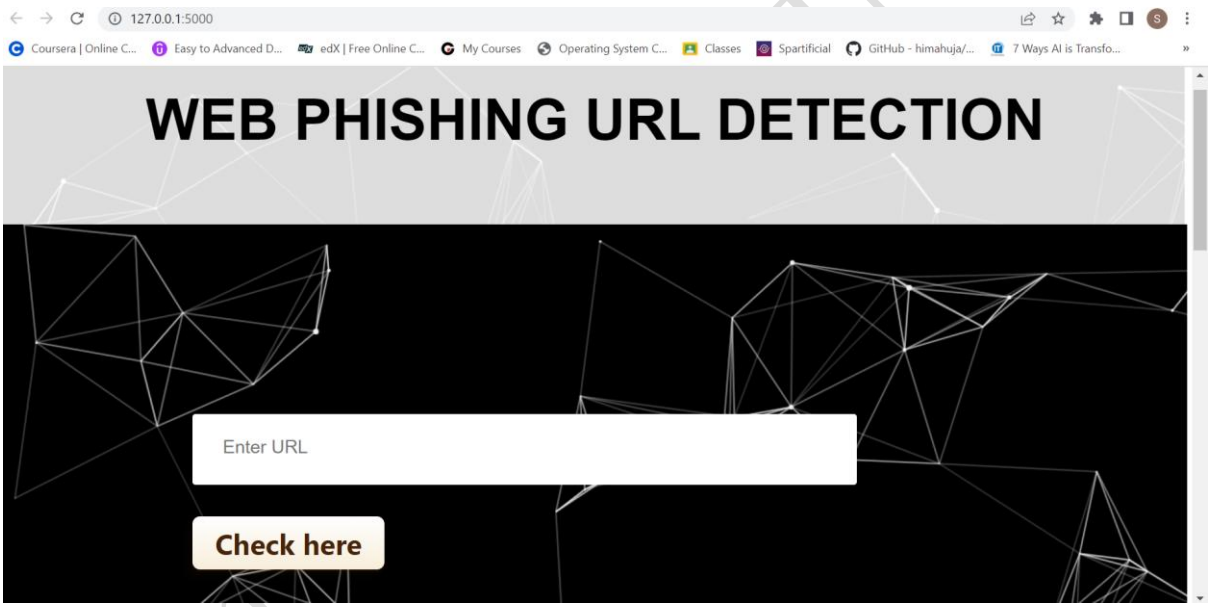
```
    # 27. PageRank
    def PageRank(self):
        try:
            prank_checker_response = requests.post("https://www.checkpagerank.net/index.php", {"name": self.domain})

            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\.211\.158|181\.174\.165\.13|
                                 '107\.151\.148\.44|107\.151\.148\.107|64\.70\.19\.203|199\.184\.144\.27|107\.151\.148\.108|107\.151\.148\.109|
                                 '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\.
                                 '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\
                                 '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
                                 '216\.38\.62\.18|104\.130\.124\.96|47\.89\.58\.141|78\.46\.211\.158|54\.86\.225\.156|54\.82\.156\.19|37\.157\.
            if url_match:
                return -1
            elif ip_match:
                return -1
            return 1
        except:
            return 1

    def getFeaturesList(self):
        return self.features
```

7.2 Feature 2

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="This website is develop for identify the safety of url.">
    <meta name="keywords" content="phishing url,phishing,cyber security,machine learning,classifier,python">

    <!-- BootStrap -->

    <link href="style.css" rel="stylesheet">
    <title>URL detection</title>

</head>

<body>
 <div class="flex-container">
    <div class="row">
        <div class="form col-md" id="form1">
            <h1>WEB PHISHING URL DETECTION</h1>


            <br>
            <div class="flex-item-left">
            <form action="/" method ="post">
                <input type="text" class="form__input" name ='url' id="url" placeholder="Enter URL" required="" />
                <label for="url" class="form__label">URL</label>
                <button class="button" role="button" >Check here</button>
            </form>
```

```html
    <div class="col-md" id="form2">


        <br>
        <h3 id="prediction"></h3>
        <button class="button2" id="button2" role="button" onclick="window.open('{{url}}')" target="_blank" >Still want to
        Continue</button>
        <button class="button1" id="button1" role="button"  onclick="window.open('{{url}}')" target="_blank">Continue</button>
    </div>
</div></div>
<div class="flex-item-right">
<br>
<h3>GitHub Team ID : PNT2022TMID35524</h3>
<h4>Team Leader: Chamala Sudheshna </h4>
<h4>Team Member 1: Dasari Shimmy Roy </h4>
<h4>Team Member 2: Gavin Gladston A </h4>
<h4>Team Member 3: Gurram Balaji</h4>

</div></div>

    <!-- JavaScript -->
    <script src="https://code.jquery.com/jquery-3.5.1.slim.min.js"
        integrity="sha384-DfXdz2htPH0lsSSs5nCTpuj/zy4C+OGpamoFVy38MVBnE+IbbVYUew+OrCXaRkfj"
        crossorigin="anonymous"></script>
    <script src="https://cdn.jsdelivr.net/npm/popper.js@1.16.0/dist/umd/popper.min.js"
        integrity="sha384-Q6E9RHvbIyZFJoft+2mJbHaEWldlvI9IOYy5n3zV9zzTtmI3UksdQRVvoxMfooAo"
        crossorigin="anonymous"></script>
    <script src="https://stackpath.bootstrapcdn.com/bootstrap/4.5.0/js/bootstrap.min.js"
```

```
    <script src="https://stackpath.bootstrapcdn.com/bootstrap/4.5.0/js/bootstrap.min.js"
        integrity="sha384-OgVRvuATP1z7JjHLkuOU7Xw704+h835Lr+6QL9UvYjZE3Ipu6Tp75j7Bh/kR0JKI"
        crossorigin="anonymous"></script>


    <script>

            let x = '{{xx}}';
            let num = x*100;
            if (0<=x && x<0.50){
                num = 100-num;
            }
            let txtx = num.toString();
            if(x<=1 && x>=0.50){
                var label = "Website is "+txtx +"% safe to use...";
                document.getElementById("prediction").innerHTML = label;
                document.getElementById("button1").style.display="block";
            }
            else if (0<=x && x<0.50){
                var label = "Website is "+txtx +"% unsafe to use..."
                document.getElementById("prediction").innerHTML = label ;
                document.getElementById("button2").style.display="block";
            }

    </script>

</body>

</html>
```



8. TESTING

8.1 Test Cases

Test case ID	Feature Type	Component	Test Scenario	Steps To Execute	Test Data	Expected Result	Actual Result	Status	TC for Automation(Y/N)	Executed By
LoginPage_TC_OO 1	Functional	Home Page	Verify user is able to see the Landing Page when user can type the URL in the box	1. Enter URL and click go 2. Type the URL 3. Verify whether it is processing or not.	http://127.0.0.1:5000/	Should Display the Webpage	Working as expected	Pass	N	Chamala Sudheshna Dasari, Shimmy Roy Gavin Gladston, Gurram Balaji
LoginPage_TC_OO 2	UI	Home Page	Verify the UI elements is Responsive	1. Enter URL and click go 2. Type or copy paste the URL 3. Check whether the button is responsive or not 4. Reload and Test Simultaneously	http://127.0.0.1:5000/	Should Wait for Response and Interacts Acknowledge	Working as expected	Pass	N	Chamala Sudheshna Dasari, Shimmy Roy Gavin Gladston, Gurram Balaji
LoginPage_TC_OO 3	Functional	Home page	Verify whether the link is legitimate or not	1. Enter URL and click go 2. Type or copy paste the URL 3. Check the website is legitimate or not 4. Observe the results	http://127.0.0.1:5000/	User should observe whether the website is legitimate or not.	Working as expected	Pass	N	Chamala Sudheshna Dasari, Shimmy Roy Gavin Gladston, Gurram Balaji
LoginPage_TC_OO 4	Functional	Home Page	Verify user is able to access the legitimate website or not	1. Enter URL and click go 2. Type or copy paste the URL 3. Check the website is legitimate or not 4. Continue if the website is legitimate or be cautious if it is not legitimate.	http://127.0.0.1:5000/	Application should show that Safe Webpage or Unsafe.	Working as expected	Pass	N	Chamala Sudheshna Dasari, Shimmy Roy Gavin Gladston, Gurram Balaji
LoginPage_TC_OO 5	Functional	Home Page	Testing the website with multiple URLs	1. Enter URL (https://phishing- shield.herokuapp.com/) and click go 2. Type or copy paste the URL to test 3. Check the website is legitimate or not 4. Continue if the website is secure or be cautious if it is not secure.	1. tvtabod.com 2. salescript.info 3.https://www.google.com/ 4. dalaats.com	User can able to identify the websites whether it is secure or not	Working as expected	Pass	N	Chamala Sudheshna Dasari, Shimmy Roy Gavin Gladston, Gurram Balaji

8.2 User Acceptance Testing

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).

Defect Analysis

This report shows the number of resolved or closed bugs at each severity level, andhow 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

Test Case Analysis

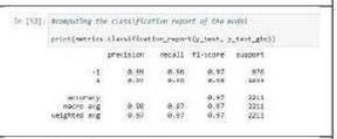
This report shows the number of test cases that have passed, failed, and untested.

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

9. RESULTS

9.1 Performance Metrics

Project team shall fill the following information in model performance testing template.

S.No.	Parameter	Values	Screenshot
1.	Metrics	Random forest Accuray Score- 96.9	
2.	Tune the Model	Hyperparameter Tuning - 96.9 Validation Method – KFOLD ∝ Cross Validation Method	

1. METRICS:
CLASSIFICATION REPORT:

In [40]:
```
#computing the classification report of the model

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

```
              precision    recall  f1-score   support

          -1       0.97      0.96      0.96       956
           1       0.97      0.98      0.97      1255

    accuracy                           0.97      2211
   macro avg       0.97      0.97      0.97      2211
weighted avg       0.97      0.97      0.97      2211
```

PERFORMANCE:

In [46]:
```
# dispalying total result
sorted_result
```

Out[46]:

ML Model	Accuracy	f1_score	Recall	Precision

Out[46]:

	ML Model	Accuracy	f1_score	Recall	Precision
0	Random Forest	0.969	0.973	0.994	0.988
1	Decision Tree	0.960	0.965	0.992	0.991
2	Support Vector Machine	0.957	0.963	0.982	0.966
3	K-Nearest Neighbors	0.953	0.959	0.990	0.989
4	Logistic Regression	0.924	0.933	0.947	0.927

TUNE THE MODEL – HYPERPARAMETER TUNING

In [51]:
```
from sklearn.ensemble import RandomForestRegressor
rf = RandomForestRegressor(random_state = 42)
from pprint import pprint
# Look at parameters used by our current forest
print('Parameters currently in use:\n')
pprint(rf.get_params())
```

```
Parameters currently in use:

{'bootstrap': True,
 'ccp_alpha': 0.0,
 'criterion': 'squared_error',
 'max_depth': None,
 'max_features': 'auto',
 'max_leaf_nodes': None,
 'max_samples': None,
 'min_impurity_decrease': 0.0,
 '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': 42,
 'verbose': 0,
 'warm_start': False}
```

VALIDATION METHODS: KFOLD & Cross Folding

In [52]:
```
rf = RandomForestClassifier(n_estimators=40)
rf.fit(X_train, y_train)
rf.score(X_test, y_test)
```

Out[52]:
```
0.966078697421981
```

In [57]:
```
from sklearn.model_selection import cross_val_score
score_rf=cross_val_score(RandomForestClassifier(n_estimators=40),X, y,cv=3)
print(score_rf)
print(np.average(score_rf))
```

```
[0.96933514 0.97313433 0.92510176]
0.9558570782451379
```

In [62]:
```
scores1 = cross_val_score(RandomForestClassifier(n_estimators=5),X, y, cv=10)
print('Avg Score for Estimators=5 and CV=10 :')
print(np.average(scores1))
```

```
Avg Score for Estimators=5 and CV=10 :
0.9660735764607693
```

In [63]:
```
scores2 = cross_val_score(RandomForestClassifier(n_estimators=20),X, y, cv=10)
print('Avg Score for Estimators=20 and CV=10 :')
print(np.average(scores2))
```

```
Avg Score for Estimators=20 and CV=10 :
0.9660735764607693
```

In [63]:
```
scores2 = cross_val_score(RandomForestClassifier(n_estimators=20),X, y, cv=10)
print('Avg Score for Estimators=20 and CV=10 :')
print(np.average(scores2))
```

```
Avg Score for Estimators=20 and CV=10 :
0.972134879268163
```

In [64]:
```
scores3 = cross_val_score(RandomForestClassifier(n_estimators=30),X, y, cv=10)
print('Avg Score for Estimators=30 and CV=10 :')
print(np.average(scores3))
```

```
Avg Score for Estimators=30 and CV=10 :
0.972859106641683
```

In [65]:
```
scores4 = cross_val_score(RandomForestClassifier(n_estimators=40),X, y, cv=10)
print('Avg Score for Estimators=40 and CV=10 :')
print(np.average(scores4))
```

```
Avg Score for Estimators=40 and CV=10 :
0.9727681179579915
```

10. ADVANTAGES & DISADVANTAGES

ADVANTAGES

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

DISADVANTAGES

- All website related data will be stored in one place.
- It is a very time-consuming process.

11. CONCLUSION

Use machine learning technologies to improve the detection process for phishing websites. With the least number of false positives, we used the logistic regression method to reach a detection accuracy of 90%. Additionally, the results demonstrate that classifiers perform better when more data is used as training data. In the future, hybrid technology that combines the blacklist approach with the random forest algorithm of machine learning technology will be utilized to more reliably detect phishing websites.

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 canbe done in an efficient manner. Thus, the project is flexible and can be enhanced at any time with more advanced features.

13. APPENDIX

Source Code:

https://github.com/IBM-EPBL/IBM-Project-260551659982405/tree/main/Final%20Deliverables

GitHub: https://github.com/IBM-EPBL/IBM-Project-26055-1659982405

Project Demo Link: https://github.com/IBM-EPBL/IBM-Project-260551659982405/tree/main/Final%20Deliverables/Demo