IBM
NALAIYA THIRAN

PROJECT REPORT
ON
WEB PHISHING DETECTION

TEAM ID: PNT2022TMID03812

SRI SAI RAM ENGINEERING COLLEGE

Team Members

ABDUL HAMEEDH S

ARAVIND S

AVINASH M

RAGUL V B

SARVESH PRIYAN S

TABLE OF CONTENTS

ABSTRACT

People are always considering new ways to obtain data from others, typically over the internet, as a result of the present continual expansion of data and the expanding availability of the internet. Phishing is one of the most popular methods of information theft. Phishing is a type of online fraud where users employ a number of techniques in an effort to obtain sensitive data, such as credit card information, from others. Due to the growth of the internet, phishing has become a significant threat to many people and businesses. As new techniques are created to combat phishing, new methods of information theft also develop. Individuals are now connected with a lot of hazards since there are constantly new phishing techniques coming from all types of people and locations. Over the last years, several approaches have been created to either prevent or detect phishing. Although the hazards connected with phishing have not entirely been eradicated, they have been greatly reduced. There are a number of pros and downsides to the methods used to prevent and detect phishing. Numerous utilized examples have been shown together with their primary weaknesses and the many anti-phishing strategies that were employed to stop the phishing**.**

Keywords: Machine learning, Classification, Phishing attack, Phishing website detection, Phishing website datasets, Phishing website features.

PRE-REQUISITES

TOOLS : Jupyter Notebook, Flask, IBM Watson Studio

OPERATING SYSTEM : Windows 10

LANGUAGE : Python

LIBRARIES REQUIRED: Pandas, Numpy, Seaborn, Matplotlib

OTHER REQUIREMENTS:

beautifulsoup4==4.9.3
Flask==2.1.3
googlesearch-python==1.1.0
lxml==4.9.1
numpy==1.23.1
pandas==1.4.3
python-dateutil==2.8.2
python-whois==0.8.0
requests==2.25.1
scikit-learn==1.1.1
sklearn==0.0
soupsieve==2.3.2.post1
urllib3==1.26.10

CHAPTER 1

INTRODUCTION

Phishing is a sort of social engineering in which an attacker delivers a phoney (e.g., spoofed, false, or other misleading) communication intended to dupe a person into giving the attacker access to sensitive information or to install dangerous software, such as ransomware, on the victim's infrastructure. Phishing attacks are getting more and more complex, and they frequently transparently mirror the site that is being attacked, allowing the attacker to watch everything the victim does there and to cross any further security barriers with them.

The term "phishing" was first used in Koceilah Rekouche's 1995 cracking toolkit AOHell, while it's probable that it was used earlier in a print version of the hacker magazine 2600. The phrase refers to the employment of more sophisticated lures to "fish" for users' sensitive information. It is a fishing-related word that was influenced by phreaking.

Some of the phishing attacks are email phishing, spear phishing, voice phishing, SMS phishing, clone phishing. Our project helps us to find out illegitimate websites by using machine learning algorithms, analyzing the given datasets with help of some parameters like ip address, url length, rightclick, domain registration, having @ symbol in the url etc and train the machine learning model with the given dataset. Also to create an website which traps the phishing websites and deploy the website in an cloud environment.

1.1 PROJECT OVERVIEW

- To develop a machine learning model to detect phishing websites
- Integrating the built model using flask to create a website which predicts the phishing websites

1.2 PURPOSE

- To create awareness to the public about the method of phishing attacks
- An approach for safe browsing in the internet so the user credentials are not stolen
- Safe and secure browsing to the users

CHAPTER 2

LITERATURE SURVEY

PAPER 2.1: A Machine Learning Approach for URL Based Web Phishing Using Fuzzy Logic as Classifier

Authors:

Abstract:

Phishing is the major problem of the internet era. In this era of internet the security of our data in web is gaining an increasing importance. Phishing is one of the most harmful ways to unknowingly access the credential information like username, password or account number from the users. Users are not aware of this type of attack and later they will also become a part of the phishing attacks. It may be the losses of financial found, personal information, reputation of brand name or trust of brand. So the detection of phishing site is necessary. In this paper we design a framework of phishing detection using URL.

PAPER 2.2: Which web browser work best for detecting phishing.

Authors: Mazher N, Ashraf I & Altaf A.

Abstract:

In our experiment we develop pages which are visually similar to original web sites. We upload these pages on free hosting site. After preparing that all we asked student to go to specified URL, which was www.myfb.comze.com Participants are given following scenario “Imagine that you receive an email message that asks you to click on one of the following links”. And one restriction was to use only from one of above three web browsers. After entering address on their browsers they have to report what happened in case of each web browser.

PAPER 2.3: A Methodical Overview on Detection, Identification and Proactive Prevention of Phishing Websites

Authors: Bhagwat M. D, Dr. Patil P. H, Dr. T. S. Vishawanath

Abstract:

Detecting and finding some phishing websites in real-time for a day now is really a dynamic and nuanced topic involving several variables and requirements. Fuzzy logic strategies may be an important method in detecting and testing phishing websites due to the ambiguities involved in the detection. Instead of exact principles, Fuzzy logic provides a more intuitive way of dealing with quality variables. An approach to fuzziness resolution and an open and intelligent phishing website detection model will be proposed in the Phishing website assessment. This approach is based on smooth logic and machine learning algorithms that define various factors on the phishing website. A total of 30 characteristics or features and phishing website attributes can be used for phishing detection with high accuracy. A real-time phishing dataset is used which is downloaded from the UCI machine learning repository.

PAPER 2.4: HTMLPhish: Enabling Phishing Web Page Detection by Applying Deep Learning Techniques on HTML Analysis

Authors: Chidimma Opara , Bo Wei , Yingke Chen

Abstract:

Recently, the development and implementation of phishing attacks require little technical skills and costs. This uprising has led to an ever-growing number of phishing attacks on the World Wide Web. Consequently, proactive techniques to fight phishing attacks have become extremely necessary. In this paper, we propose HTMLPhish, a deep learning based datadriven end-to-end automatic phishing web page classification approach. Specifically, HTMLPhish receives the content of the HTML document of a web page and employs Convolutional Neural Networks (CNNs) to learn the semantic dependencies in the textual contents of the HTML. The CNNs learn appropriate feature representations from the HTML document embeddings without extensive manual feature engineering. Furthermore, our proposed approach of the concatenation of the word and character embeddings allows our model to manage new features and ensure easy extrapolation to test data. We conduct comprehensive experiments on a dataset of more than 50,000 HTML documents that provides a distribution of phishing to benign web pages obtainable in the real-world that yields over 93% Accuracy and True Positive Rate. Also, HTMLPhish is a completely language-independent and client-side strategy which can, therefore, conduct web page phishing detection regardless of the textual language.

PAPER 2.5: Detection and Prevention of Phishing Websites using Machine Learning Approach

Authors: Vaibhav Patil, Tushar Bhat, Pritesh Thakkar, Prof. S. P. Godse, Chirag Shah

Abstract:

Phishing costs Internet user's lots of dollars per year. It refers to exploiting weakness on the user side, which is vulnerable to such attacks. The phishing problem is huge and there does not exist only one solution to minimize all vulnerabilities effectively, thus multiple techniques are implemented. In this paper, we discuss three approaches for detecting phishing websites. First is by analyzing various features of URL, second is by checking legitimacy of website by knowing where the website is being hosted and who are managing it, the third approach uses visual appearance-based analysis for checking genuineness of website. We make use of Machine Learning techniques and algorithms for evaluation of these different features of URL and websites. In this paper, an overview about these approaches is presented.

2.1 EXISTING PROBLEM

The issue with phishing is that perpetrators are always coming up with fresh and original ways to deceive people into thinking their activities are connected to a reliable website or email. Phishers are becoming better at creating fake websites that seem just like the real thing. They've even started adding logos and pictures to their phishing emails to increase their effectiveness. There are hazardous new sophisticated phishing techniques that use publicly accessible personal data to create realistic and convincing assaults that directly target victims. Attacks that take advantage of the enormous quantity of publicly available data to make their schemes more effective include social phishing and context-aware phishing.

2.2 REFERENCES

- H.Chapla, R. Kotak and M. Joiser, "A Machine Learning Approach for URL Based Web Phishing Using Fuzzy Logic as Classifier," 2019 International Conference on Communication and Electronics Systems (ICCES), 2019, pp. 383-388, doi: 10.1109/ICCES45898.2019.9002145.

- N. Mazher, I. Ashraf and A. Altaf, "Which web browser work best for detecting phishing," 2013 5th International Conference on Information and Communication Technologies, 2013, pp. 1-5, doi: 10.1109/ICICT.2013.6732784.

- M. D. Bhagwat, P. H. Patil and T. S. Vishawanath, "A Methodical Overview on Detection, Identification and Proactive Prevention of Phishing Websites," 2021 Third International Conference on Intelligent Communication Technologies and Virtual Mobile Networks (ICICV),
- 2021, pp. 1505-1508, doi: 10.1109/ICICV50876.2021.9388441.

- C. Opara, B. Wei and Y. Chen, "HTMLPhish: Enabling Phishing Web Page Detection by Applying Deep Learning Techniques on HTML Analysis," 2020 International Joint Conference on Neural Networks (IJCNN), 2020, pp. 1-8, doi: 10.1109/IJCNN48605.2020.9207707.

- V. Patil, P. Thakkar, C. Shah, T. Bhat and S. P. Godse, "Detection and Prevention of Phishing Websites Using Machine Learning Approach," 2018 Fourth International Conference on Computing Communication Control and Automation (ICCUBEA), 2018, pp. 1-5, doi: 10.1109/ICCUBEA.2018.8697412.

2.3 PROBLEM STATEMENT DEFENETION

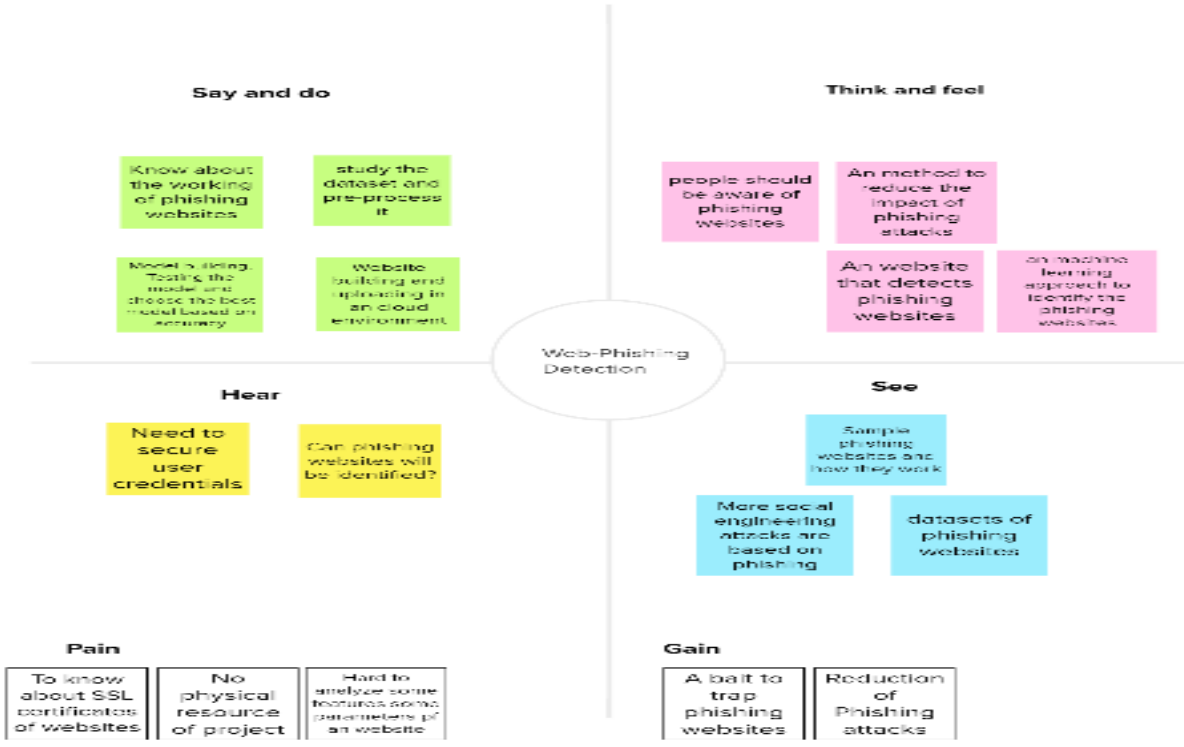
We proposed an intelligent, adaptable, and successful system that is built on applying classification algorithms in order to identify and forecast phishing websites. In order to classify the legitimacy of the phishing datasets, we used classification algorithms and approaches to extract the relevant characteristics. In the ultimate phishing detection rate, the phishing website may be identified based on several crucial elements including the URL, having@symbol, domain identity, security and encryption criteria. Our technology uses a data mining algorithm to determine whether an e-banking website is a phishing website when a user submits their information or the URL of the website.

Due to exponential growth in the number of people using the internet, the internet now controls much of the world. Due to the anonymity offered by the internet and the rapid growth of online transactions, hackers try to trick end users by using techniques like phishing, SQL injection, malware, man-in-the-middle attacks, domain name system tunnelling, ransomware, web trojans, and so on. Phishing is said to be the most deceptive attack of all these. In order to achieve maximum accuracy and a clear model, our primary goal in this paper is the classification of a phishing website using various machine learning techniques.

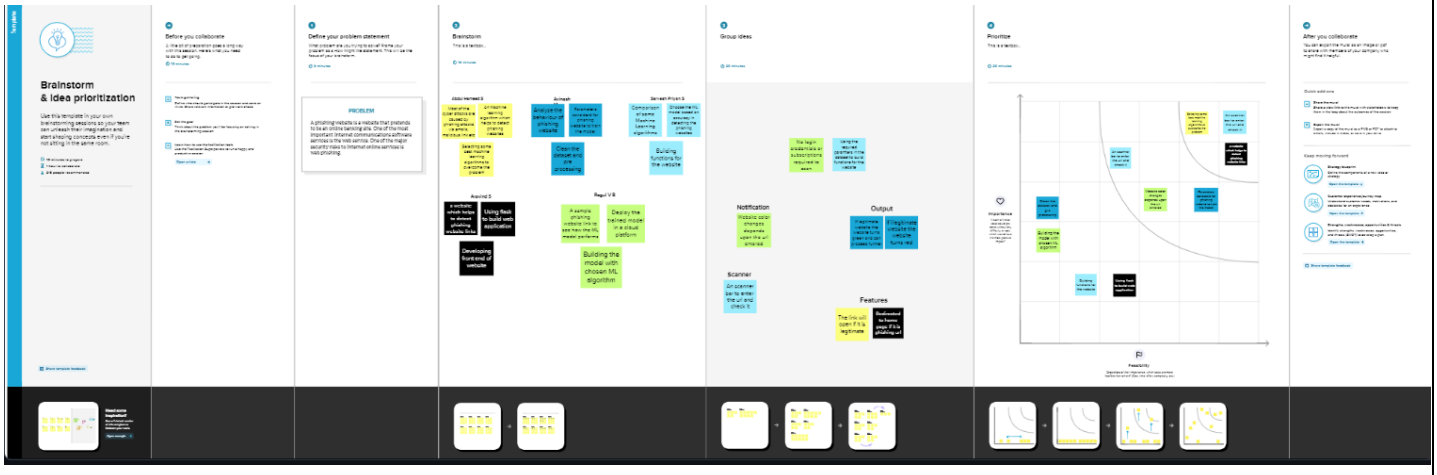
CHAPTER 3

IDEATION & PROPOSED SOLUTION

3.1 Empathy Map Canvas



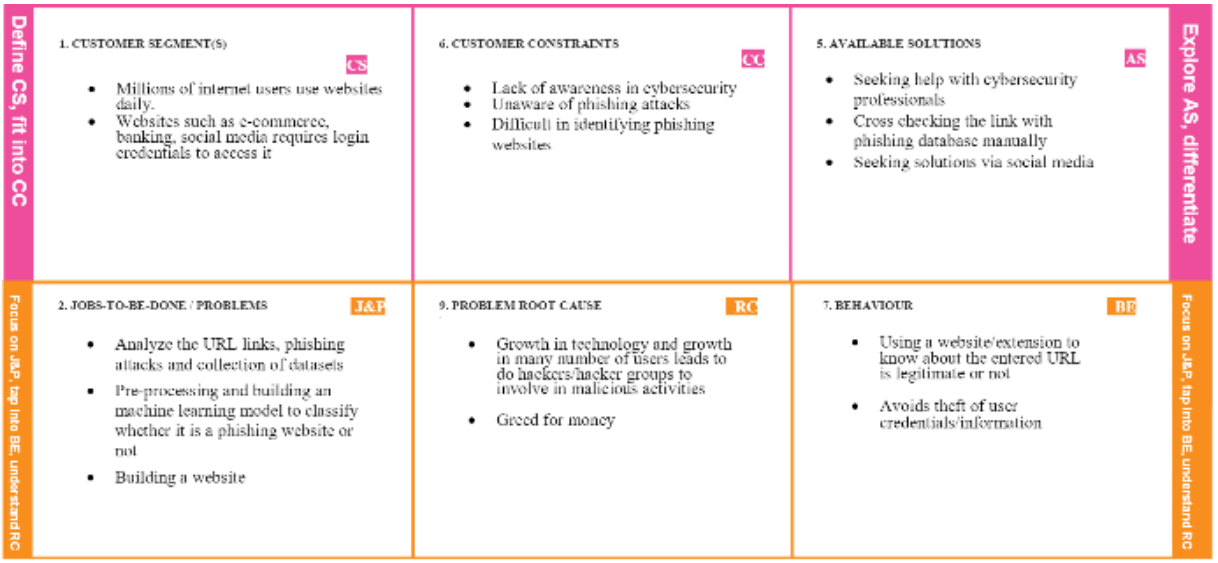
3.2 Ideation & Brainstorming

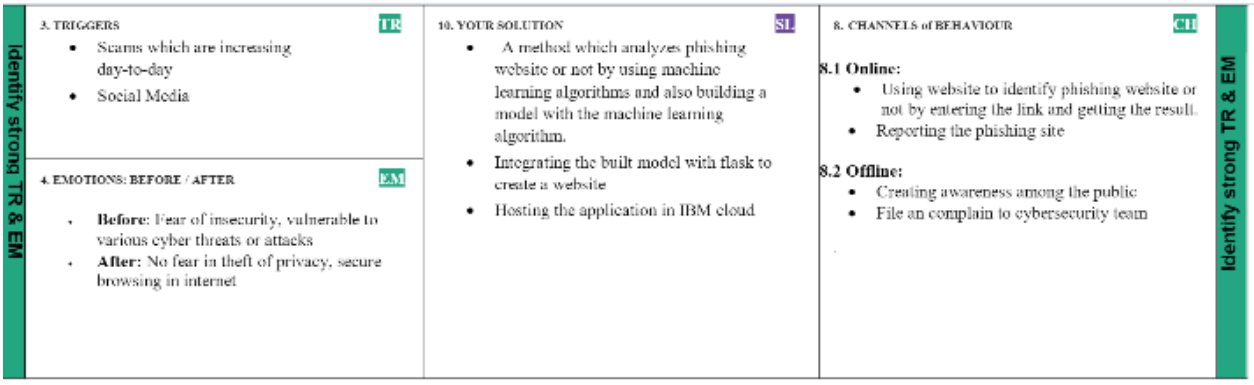


3.3 Proposed Solution

S.No.	Parameter	Description
1.	Problem Statement (Problem to be solved)	There are many users who buy goods online and pay by e-banking. There are e-banking websites that require users to provide sensitive information such as usernames, passwords and credit card details, often for malicious reasons. This type of e-banking website is known as a phishing website. Web services are one of the Internet's most important communications software services. Web phishing is one of many security threats to web services on the Internet.
2.	Idea / Solution description	The proposed solution involves using classification or regression algorithms to predict phishing websites. Also developing a web app and integrating with the built model which delivers a warning when a phishing url is entered and detected.
3.	Novelty / Uniqueness	The color of the website turns green when the url is legitimate and proceeds further to the website. The color of the website will turn red when a phishing url is entered and goes back to the home page.
4.	Social Impact / Customer Satisfaction	The customers do not require to pay for any subscriptions or sign up to use the website.
5.	Business Model (Revenue Model)	-
6.	Scalability of the Solution	By setting up this system, people can know about legitimate, phishing websites and methodology of phishing websites. It also helps to stop phishing websites through detection. Future technologies can also be integrated into this system.

3.4 Problem Solution fit:





REQUIREMENT ANALYSIS

4.1 Functional Requirements:

Following are the functional requirements of the proposed solution.

FR No.	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
FR-1	Verifying input	User inputs an URL (Uniform Resource Locator) in necessary field to check its validation.
FR-2	Website Evaluation	Model evaluates the website with some parameters defined in the function
FR-3	Extraction and Prediction	Based upon the parameters chosen, the values/ result of the URL is predicted (i.e) phishing or not
FR-4	Real Time monitoring	When a user visits a phished website, the Extension plugin should display a warning pop-up. The extension plugin will be able to recognise the newest and most sophisticated phishing websites.
FR-5	Authentication	Authentication ensures the security of enterprise information, secure processes, and websites.

4.2 Non-functional Requirements:

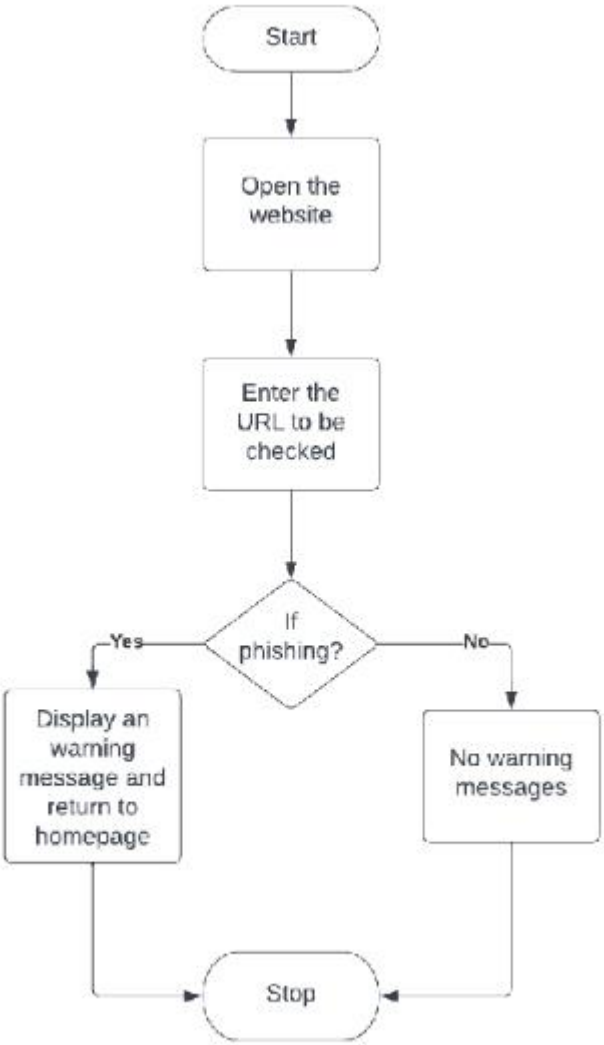
Following are the non-functional requirements of the proposed solution

FR No.	Non-Functional Requirement	Description
NFR-1	**Usability**	Users' potential needs in terms of web phishing detection, behavior, and experience may be better understood by designers if consumer product usability is analysed during the design process with user experience at its core.
NFR-2	**Security**	Any data included inside the system or its parts will be protected from virus threats and unauthorised access, according to this assurance. Identify the login process and various user roles as system behaviour or user activities if you want to restrict unauthorised access to the admin panel.
NFR-3	**Reliability**	It describes the likelihood that the system or a particular component will function properly for a specific period of time under particular circumstances.
NFR-4	**Performance**	It is concerned with determining how quickly the system reacts under various load conditions.
NFR-5	**Availability**	It indicates the possibility that a user will have access to the system at a specific time. It can be expressed as the anticipated percentage of requests that are successful, but it can also be expressed as the percentage of time the system is active over a given period of time.
NFR-6	**Scalability**	It has access to the heaviest workloads necessary to meet the performance requirements. Vertical and horizontal scaling are the two methods that allow the system to expand as workloads rise.

PROJECT DESIGN

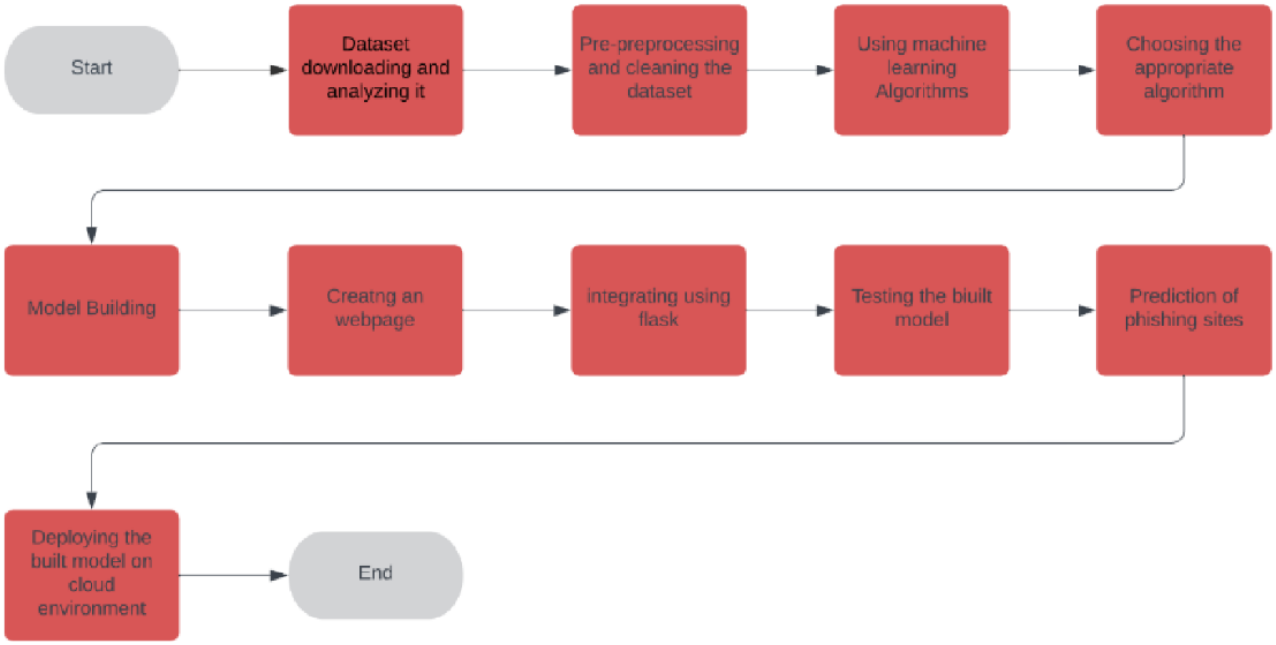
5.1 Data Flow Diagrams:

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.



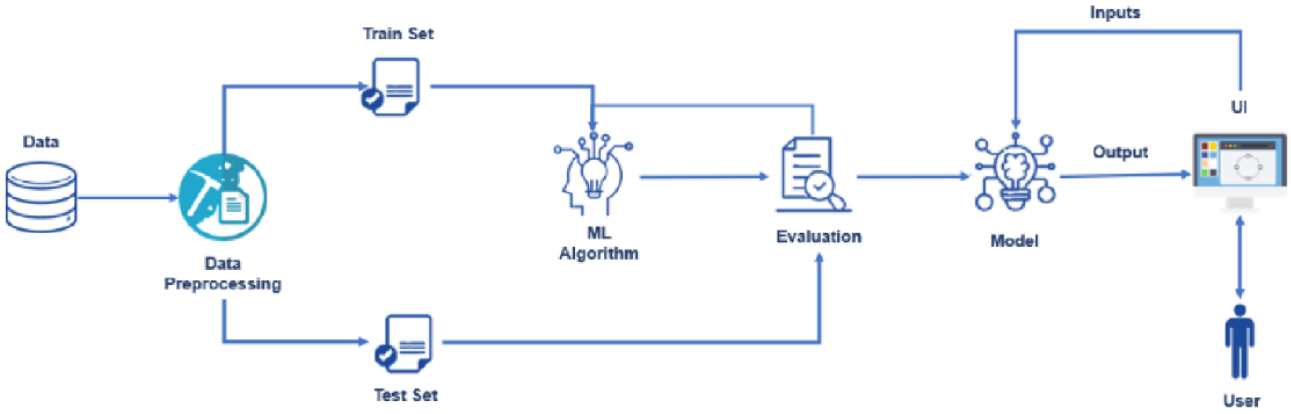
5.2 Solution and Technical Architecture

Solution Architecture



Technical Architecture:

MODEL FOR WEB PHISHING DETECTION



5.3 USER STORIES

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
Customer (Web user)	User input	USN-1	As a user, I 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-3	After comparison in case if none found on comparison then we can extract feature using Machine learning approach.	As a User i can have comparison between websites for security.	High	Sprint-2
Administrator	Prediction	USN-4	Here the Model will predict the URL websites using Machine Learning algorithms such as Logistic Regression, Random Forest, Classification algorithms	In this I can have correct prediction on the particular algorithms	High	Sprint-2
	Classifier	USN-5	Here I will send all the model output to classifier in order to produce final result.	I this I will find the correct classifier for producing the result	Medium	Sprint-3
	Announcement	USN-6	Model displays the output whether the website is legitimate or not	I will find the URL is phishing URL or not	High	Sprint-4
	Events	USN-7	This model needs the capacity of retrieving and displaying accurate result for a website	I display the output of the model	High	Sprint-4

PROJECT PLANNING & SCHEDULING

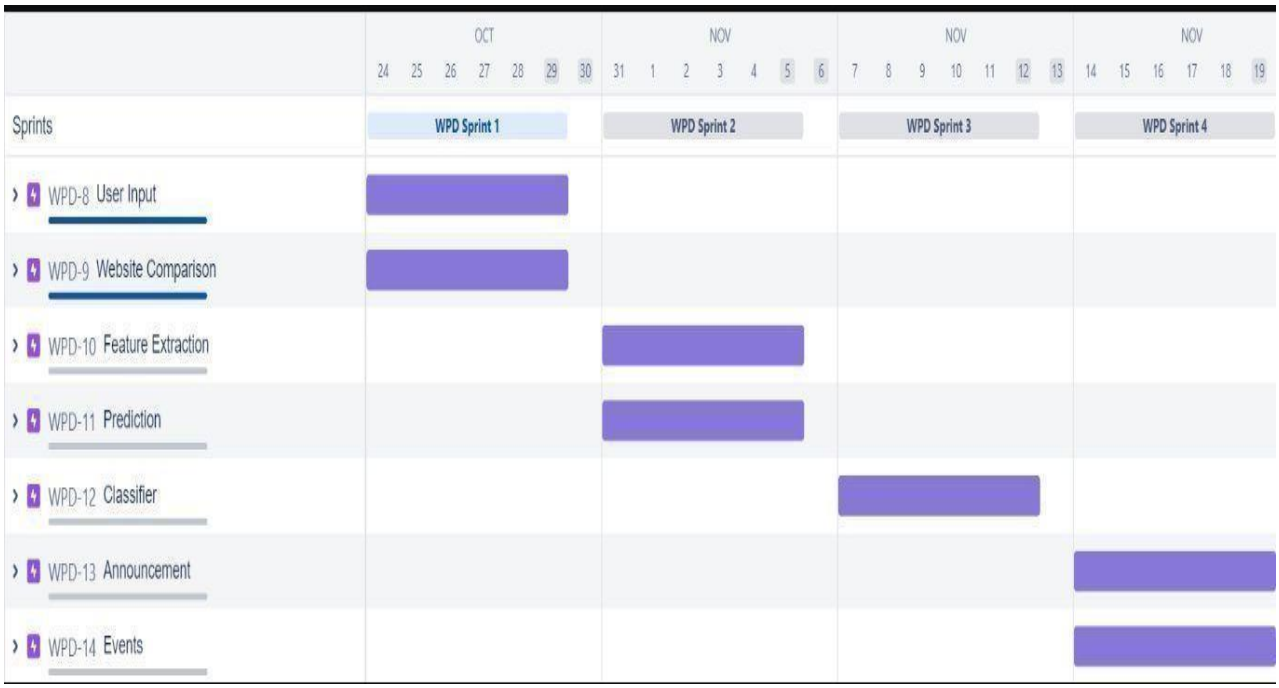
6.1 Sprint Planning & Estimation

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

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 Release Date (Actual)
Sprint-1	20	6 Days	24 Oct 2022	29 Oct 2022	20	29 Oct 2022
Sprint-2	20	6 Days	31 Oct 2022	05 Nov 2022	20	05 Nov 2022
Sprint-3	20	6 Days	07 Nov 2022	12 Nov 2022	20	12 Nov 2022
Sprint-4	20	6 Days	14 Nov 2022	19 Nov 2022	20	19 Nov 2022

6.3 Reports from JIRA



CHAPTER-7

CODING & SOLUTION

7.1 Feature 1

```
 from flask import Flask, request, render_template
import numpy as np
import FeatureExtractor
import pickle

app = Flask(__name__)

@app.route("/")
def welcome():
   return render_template("index.html");

@app.route("/about")
def about():
   return render_template("about.html");

@app.route("/product")
def product():
   return render_template("product.html")

@app.route("/predict", methods=['POST'])
def predict():

   # getting the URL from the website
   url = request.form["url"]
   # url = "https://www.linkedin.com/"

   # loading the saved model
   model = getModel()

   # getting the features from the url
   features = FeatureExtractor.getFeatures(url)

   result = model.predict(features)
   # probability = model.predict_proba(features)
   result = result.tolist()
   print(type(result))
   print(result)
   # print(probability)

   # result = [1]
   message = ""
   if(result[0] == 1):
      message = "Legitimate site"
   else:
      message = "Suspecious site"
   return render_template("product.html", message = message, url = url)
```

```
def getModel():
    file = open("./model.pkl","rb")
    model = pickle.load(file)
    file.close()
    return model

def getFeaturesFromURL(url: str) -> np.ndarray:
    features = FeatureExtractor.getFeatures(url)
    features = np.array(features).reshape(1, 27)
    return features

if __name__ == "__main__":
    app.run(debug = True)
```

7.2 Feature 2

```
import ipaddress

import re

import socket

import traceback

import urllib.request

from datetime import date

from urllib.parse import urlparse


import numpy as np

import pandas as pd

import requests

import whois

from bs4 import BeautifulSoup

from googlesearch import search


url = ""


def getDomainName() -> str:

    domainName = ""

    try:

        urlparser = urlparse(url)
```

```
        domainName = urlparser.netloc
    except:
        traceback.print_exc()
        raise Exception("Could not find the Domain Name")
    return domainName

def getSoupObject():
    try:
        responseObj = requests.get(url)
        soupObj = BeautifulSoup(responseObj.text, 'html.parser')
        return soupObj
    except:
        traceback.print_exc()
        raise Exception("Could not get the beatiful soup response object")



def getFeatures(URL: str) -> np.ndarray:
    global url
    url = URL

    features = [
        isHavingIp(),
        isLongURL(),
        isURLShorteningServiceUsed(),
        isAtSymbolPresent(),
        isRedirectedUsingSlashes(),
        isHyphenPresent(),
        subDomain(),
        isUsingHTTPS(),
        domainRegistrationLength(),
        isUsingNonStdPort(),
        isHTTPSInDomainPart(),
        requestURL(),
```

```
        URLOfAnchor(),
        linksInMetaScriptLinkTag(),
        serverFormHandler(),
        submittingInfoToEmail(),
        isAbnormalURL(),
        websiteForwarding(),
        statusBarCustomization(),
        isRightClickDisabled(),
        ageOfDomain(),
        checkDNSRecord(),
        websiteTraffic(),
        pageRank(),
        googleIndex(),
        linksPointingToPage(),
        statsReport()
    ]

    features = np.array(features).reshape(1, 27)
    return features

# 1. Using the IP Address
def isHavingIp() -> int:
    domainName = getDomainName()
    try:
        ipaddress.ip_address(domainName)
        print("1. Success")
        return -1
    except:
        traceback.print_exc()
        print("1. Fail")
        return 1

# 2. Long URL to Hide the Suspicious Part
```

```
def isLongURL() -> int:
    try:
        urlLength = len(url)
        print("2. Success")
        if urlLength < 54:
            return 1
        elif 54 <= urlLength <= 75:
            return 0
        return -1
    except:
        print("2. Fail")
        return 0


# 3. Using URL Shortening Services
def isURLShorteningServiceUsed() -> int:
    try:
        commonURLShorteners = [
            't\\.co', 'bitly\\.com', 'is\\.gd', 'prettylinkpro\\.com', 'cutt\\.us', 'rubyurl\\.com', 'tr\\.im',
            'v\\.gd', 'snipr\\.com', 'tinyurl', 'cli\\.gs', 'x\\.co', 'filoops\\.info', 'wp\\.me', 'q\\.gs', 't\\.co',
            'ow\\.ly', 'tiny\\.cc', 'migre\\.me', 'om\\.ly', 'bkite\\.com', 'twit\\.ac', 'db\\.tt', 'kl\\.am',
            'link\\.zip\\.net', 'x\\.co', 'u\\.bb', 'doiop\\.com', 'shorte\\.st', 'goo\\.gl', 'qr\\.net', 'u\\.to',
            'loopt\\.us', 'adf\\.ly', 'buzurl\\.com', 'post\\.ly', '1url\\.com', 'goo\\.gl', 'ff\\.im', 'short\\.ie',
            'to\\.ly', 'bit\\.ly', 'yfrog\\.com', 'yourls\\.org', 'vzturl\\.com', 'lnkd\\.in', 'ity\\.im', 'go2l\\.ink',
            'fic\\.kr', 'Just\\.as', 'su\\.pr', 'bit\\.ly', 'url4\\.eu', 'qr\\.ae', 'po\\.st', 'scrnch\\.me', 'tr\\.im',
            'twitthis\\.com', 'tweez\\.me', 'ping\\.fm', 'snipurl\\.com', 'j\\.mp', 'ow\\.ly', 'bit\\.do',
'short\\.to',
            'BudURL\\.com', 'twurl\\.nl', 'bc\\.vc', 'is\\.gd', 'tinyurl\\.com', 'cur\\.lv'
        ]
        commonURLShorteners = "|".join(commonURLShorteners)
        isURLShortenerPresent = re.search(commonURLShorteners, url)
        print("3. Success")
        if(isURLShortenerPresent):
            return -1
```

```
        return 1
    except:
        print("3. Fail")
        return 0

# 4. URLs having “@” Symbol
def isAtSymbolPresent() -> int:
    try:
        if ( '@' in url ):
            print("4. Success")
            return -1
        print("4. Success")
        return 1
    except:
        print("4. Fail")
        return 0

# 5. Redirecting using “//”
def isRedirectedUsingSlashes() -> int:
    try:
        lastOccurenceOfDoubleSlash = -1
        if("//" in url):
            lastOccurenceOfDoubleSlash = url.rindex("//")
        if(lastOccurenceOfDoubleSlash > 6):
            print("5. Success")
            return -1
        print("5. Success")
        return 1
    except:
        print("5. Fail")
        return 0

# 6. Adding Prefix or Suffix Separated by (-) to the Domain
```

```
def isHyphenPresent() -> int:
    try:
        if ( '-' in url ):
            print("6. Success")
            return -1
        print("6. Success")
        return 1
    except:
        print("6. Fail")
        return 0

# 7. Sub Domain and Multi Sub Domains
def subDomain() -> int:
    try:
        if ( "www." in url ):
            modifiedUrl = url.replace("www.", "")

        # Country-Code Top Level Domains
        ccTLD = pd.read_csv("./country-codes-tlds.csv")
        ccTLD = ccTLD['tld'].to_list()
        ccTLD = [code.strip() for code in ccTLD]
        for code in ccTLD:
            if code in modifiedUrl:
                modifiedUrl = modifiedUrl.replace(code, "")

        dotCount = modifiedUrl.count(".")

        if dotCount <= 1:
            print("7. Success")
            return 1
        elif dotCount == 2:
            print("7. Success")
            return 0
```

```
        print("7. Success")
        return -1
    except:
        print("7. Fail")
        return 0

# 8. HTTPS (Hyper Text Transfer Protocol with Secure Sockets Layer)
def isUsingHTTPS() -> int:
    try:
        scheme = urlparse(url).scheme
        if scheme == 'https':
            print("8. Success")
            return 1
        print("8. Success")
        return -1
    except:
        print("8. Fail")
        return 1

# 9. Domain Registration Length
def domainRegistrationLength() -> int:
    try:
        whoisResponse = whois.whois(getDomainName())

        creationDate = whoisResponse.creation_date
        expirationDate = whoisResponse.expiration_date

        try:
            if(creationDate):
                creationDate = creationDate[0]
        except:
            traceback.print_exc()
```

```
        try:
            if(expirationDate):
                expirationDate = expirationDate[0]
        except:
            traceback.print_exc()

        print(creationDate.year, creationDate.month)
        ageOfDomainInMonths = ((expirationDate.year - creationDate.year) * 12) + (expirationDate.month
- creationDate.month)

        if ageOfDomainInMonths >= 12:
            print("9. Success")
            return 1
        print("9. Success")
        return -1
    except:
        traceback.print_exc()
        print("9. Fail")
        return -1

# 10. Using Non-Standard Port
def isUsingNonStdPort() -> int:
    try:
        preferredStatusOpenPorts = [80, 443]
        preferredStatusClosePorts = [21, 22, 23, 445, 1433, 1521, 3306, 3389]

        openPortsNumber, closedPortsNumber = [], []

        domain = getDomainName()

        for portNumber in preferredStatusOpenPorts + preferredStatusClosePorts:
            sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
            socket.setdefaulttimeout(1)
```

```
            result = sock.connect_ex((domain, portNumber))
            if(result == 0):
                openPortsNumber.append(portNumber)
            else:
                closedPortsNumber.append(portNumber)
            sock.close

        for portNumber in openPortsNumber:
            if portNumber in preferredStatusClosePorts:
                print("10. Success")
                return -1
        print("10. Success")
        return 1
    except:
        print("10. Fail")
        return 0

# 11. The Existence of “HTTPS” Token in the Domain Part of the URL
def isHTTPSInDomainPart() -> int:
    try:
        domain = getDomainName()
        if( ("https" in domain) and ("http" in domain) ):
            print("11. Success")
            return -1
        print("11. Success")
        return 1
    except:
        print("11. Fail")
        return 0

# 12. Request URL
def requestURL() -> int:
    try:
```

```
soupObj = getSoupObject()
domainName = getDomainName()

i, success = 0, 0

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

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

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

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

try:
    percentage = success/float(i) * 100
    if( percentage < 22.0 ):
        print("12. Success")
```

```
            return 1
        elif( (percentage >= 22.0) and (percentage < 61.0) ):
            print("12. Success")
            return 0
        else:
            print("12. Success")
            return -1
    except:
        print("12. Success")
        return 0
except:
    traceback.print_exc()
    print("12. Fail")
    # return -1
    return 0

# 13. URL of Anchor
def URLOfAnchor() -> int:
    try:
        i, unsafe = 0, 0
        for a in getSoupObject().find_all('a', href=True):
            if (a['href'] == "#") or ("javascript" in a['href'].lower()) or ("mailto" in a['href'].lower()) or (not (url
in a['href'])) or (getDomainName() in a['href']):
                unsafe = unsafe + 1
            i += 1

        try:
            #TODO always percentage is 100. Have to check
            percentage = unsafe / float(i) * 100
            if percentage < 31.0:
                print("13. Success")
                return 1
            elif ((percentage >= 31.0) and (percentage < 67.0)):
```

```
            print("13. Success")
            return 0
        else:
            print("13. Success")
            return -1
    except:
        traceback.print_exc()
        return -1
except:
    traceback.print_exc()
    # return -1
    print("13. Fail")
    return 0

# 14. Links in <Meta>, <Script> and <Link> tags
def linksInMetaScriptLinkTag():
    try:
        soupObj = getSoupObject()
        domainName = getDomainName()

        i, success = 0, 0

        for link in soupObj.find_all('link', href=True):
            dots = [x.start(0) for x in re.finditer('\.', link['href'])]
            if url in link['href'] or domainName in link['href'] or len(dots) == 1:
                success = success + 1
            i = i+1

        for script in soupObj.find_all('script', src=True):
            dots = [x.start(0) for x in re.finditer('\.', script['src'])]
            if url in script['src'] or domainName in script['src'] or len(dots) == 1:
                success = success + 1
            i = i+1
```

```
        try:
            percentage = success / float(i) * 100
            if percentage < 17.0:
                print("14. Success")
                return 1
            elif((percentage >= 17.0) and (percentage < 81.0)):
                print("14. Success")
                return 0
            else:
                print("14. Success")
                return -1
        except:
            print("14. Success")
            return 0
    except:
        traceback.print_exc()
        # return -1
        print("14. Fail")
        return 0

# 15. Server Form Handler (SFH)
def serverFormHandler():
    try:
        soupObj = getSoupObject()
        domainName = getDomainName()

        if len(soupObj.find_all('form', action = True)) == 0:
            print("15. Success")
            return 1
        else :
            for form in soupObj.find_all('form', action = True):
                    if (form['action'] == "") or (form['action'] == "about:blank"):
```

```
                print("15. Success")
                return -1
            elif (url not in form['action']) and (domainName not in form['action']):
                print("15. Success")
                return 0
            else:
                print("15. Success")
                return 1
    except:
        traceback.print_exc()
        # return -1
        print("15. Fail")
        return 0

# 16. Submitting Information to Email
def submittingInfoToEmail():
    #TODO wrong
    try:
        #TODO pattern thappu
        if ( re.findall(r"mail\(\)|mailto:?", str(getSoupObject())) ):
            print("16. Success")
            return -1
        else:
            print("16. Success")
            return 1
    except:
        traceback.print_exc()
        # return -1
        print("16. Fail")
        return 0

# 17. Abnormal URL
def isAbnormalURL(): #wrong
```

```
    try:
        if (requests.get(url)).text == whois.whois(getDomainName()):
            print("17. Success")
            return 1
        else:
            print("17. Success")
            return -1
    except:
        print("17. Fail")
        return 0
        # return -1

# 18. Website Forwarding
def websiteForwarding():
    try:
        if len((requests.get(url)).history) <= 1:
            print("18. Success")
            return 1
        elif len((requests.get(url)).history) <= 4:
            print("18. Success")
            return 0
        else:
            print("18. Success")
            return -1
    except:
        print("18. Fail")
        return 0
        # return -1

# 19. Status Bar Customization
def statusBarCustomization():
    try:
        if re.findall("<script>.+onmouseover.+</script>", (requests.get(url)).text):
```

```
            print("19. Success")
            return 1
        else:
            print("19. Success")
            return -1
    except:
        print("19. Fail")
        return 0
        # return -1

# 20. Disabling Right Click
def isRightClickDisabled(): #wrong
    try:
        if re.findall(
            r"event.button ?== ?2",
            (requests.get(url)).text
        ):
            print("20. Success")
            return 1
        else:
            print("20. Success")
            return -1
    except:
        print("20. Fail")
        return 0
        # return -1

# 21. Age of Domain
def ageOfDomain() -> int:
    try:
        creationDate = whois.whois(getDomainName()).creation_date
        try:
            if(len(creationDate)):
```

```
            creationDate = creationDate[0]
        except:
            pass

        today = date.today()

        domainAgeInMonths = ((today.year - creationDate.year) *12) + (today.month -
creationDate.month)
        if domainAgeInMonths >= 6:
            print("21. Success")
            return 1
        print("21. Success")
        return -1
    except:
        print("21. Fail")
        return 0
        # return -1

# 22. DNS Record
def checkDNSRecord() -> int:
    try:
        whoisResponse = whois.whois(getDomainName())
        if (whoisResponse):
            print("22. Success")
            return 1
        print("22. Success")
        return -1
    except:
        print("22. Fail")
        return 0
        # return -1

# 23. Website Traffic
```

```
def websiteTraffic() -> int:
    try:
        websiteRank = BeautifulSoup(
                urllib.request.urlopen("http://data.alexa.com/data?cli=10&dat=s&url=" + url).read(),
                features = "xml"
            ).find("REACH")['RANK']
        if (int(websiteRank) < 100000):
            print("23. Success")
            return 1
        print("23. Success")
        return 0
    except :
        traceback.print_exc()
        print("23. Fail")
        return 0
        # return -1

# 24. PageRank
def pageRank() -> int:
    try:
        checkerResponse = requests.post("https://www.checkpagerank.net/index.php", {"name":
getDomainName()})
        globalRank = int(re.findall(r"Global Rank: ([0-9]+)", checkerResponse.text)[0])
        if ( 0 < globalRank < 100000 ):
            print("24. Success")
            return 1
        print("24. Success")
        return -1
    except:
        traceback.print_exc()
        print("24. Fail")
        return 0
        # return -1
```

```
# 25. Google Index
def googleIndex() -> int:
    modifiedUrl = "site:" + url
    try:
        searchResults = search(modifiedUrl, 5)
        searchResultsList = []
        for gen in searchResults:
            searchResultsList.append(gen)
        if searchResultsList:
            print("25. Success")
            return 1
        else:
            print("25. Success")
            return -1
    except:
        traceback.print_exc()
        print("25. Fail")
        return 0
        # return -1

# 26. Number of Links Pointing to Page
def linksPointingToPage() -> int: # Wrong
    try:
        noOfLinks = len(re.findall(r"<a href=", requests.get(url).text))
        if noOfLinks == 0:
            print("26. Success")
            return 1
        elif noOfLinks <= 2:
            print("26. Success")
            return 0
        else:
            print("26. Success")
```

```
            return -1
    except:
        print("26. Fail")
        return 0
        # return -1
# 27. Statistical-Reports Based Feature
def statsReport() -> int:
    try:
        isUrlMatched = 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(getDomainName())
        isIpMatched = 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|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|107\.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\.156
```

```
\.19|37\.157\.192\.102|204\.11\.56\.48|110\.34\.231\.42',
            ip_address
        )
        if isUrlMatched:
            print("27. Success")
            return -1
        elif isIpMatched:
            print("27. Success")
            return -1
        print("27. Success")
        return 1
    except:
        print("27. Fail")
        return 0
        # return 1
```

CHAPTER 8
TESTING

8.1 Test Cases

				Date	15-Nov-22							
				Team ID	PNT2022TMID03926							
				Project Name	Project-WebPhishing Detection							
				Maximum Marks	4marks							
Test case ID	Feature Type	Component	TestScenario	Pre-Requisite	Steps To Execute	TestData	Expected Result	Actual Result	Status	Comments	TC for Automation(Y/N)	BUG ID	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 pr ocessing or not.	https://phishing-shield.herokuapp.com/	Should Display the Webpage	Workingas expected	Pass		N		S Balaji
LoginPage_TC_OO 2	UI	Home Page	Verify the UIeleme nts is Responsive		1. Enter URL and click go 2. Type orcopypaste the URL 3. Check whether the button is responsive or not 4. Reload and TestSimultaneously	https://phishing-shield.herokuapp.com/	Should Wait for Response and then getsAcknowledge	Workingas expected	Pass		N		RAbisheik
LoginPage_TC_OO 3	Functional	Home page	Verify whether the link is legitimate ornot		1. Enter URL and click go 2. Type orcopypaste the URL 3. Check the website is legitimate or not 4. Observe the results	https://phishing-shield.herokuapp.com/	User should observe whether the websiteislegitimate ornot.	Workingas expected	Pass		N		TS Aswin
LoginPage_TC_OO 4	Functional	Home Page	Verify user is able to access the legitimate website ornot		1. Enter URL and click go 2. Type orcopypaste the URL 3. Check the website is legitimate or not 4. Continue if the website is legitimate or be cautious if itis notlegitimate.	https://phishing-shield.herokuapp.com/	Application shouldshow that Safe Webpage orUnsafe.	Workingas expected	Pass		N		Balajee A V
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 orbe cautiousif itis not secure	1. https://avbalajee.github.io/welcome 2.totalpad.com https://www.kince.edu salescript.info 5. https://www.google.com/ 6.delgets.com	User can able to identify the websites whether it is secure or not	Workingas expected	Pass		N		Balajee A V

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 howthey 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

3. Test Case Analysis

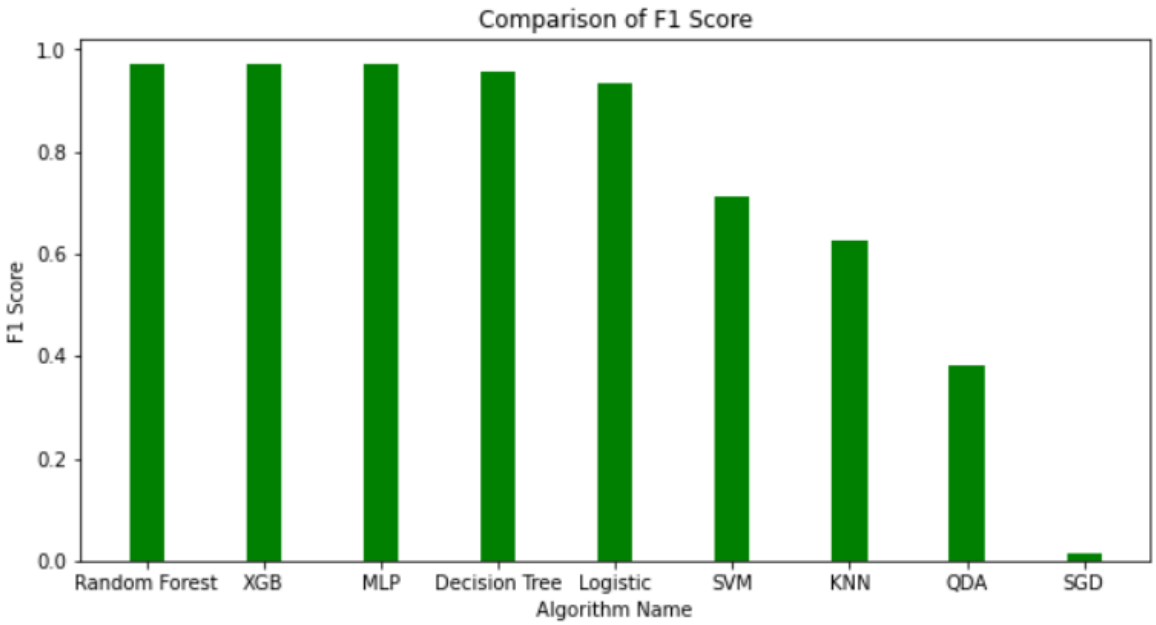
This report shows the number of test cases that have passed, failed, and untested

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




```
print(f"Accuracy: {randomForestAccuracy}")
print(f"Precision: {randomForestPrecision}")
print(f"Recall: {randomForestRecall}")
print(f"F1 Score: {randomForestF1}")
print(f"Log Loss: {randomForestLogLoss}")
print(f"AUC Score: {randomForestAucScore}")
print("Confusion Matrix:")
print(randomForestConfusionMatrix)
```

```
Accuracy: 0.9682496607869742
Precision: 0.9611510791366906
Recall: 0.9823529411764705
F1 Score: 0.9716363636363635
Log Loss: 1.0966354424956306
AUC Score: 0.9665564097979618
Confusion Matrix:
[[1564   81]
 [  36 2004]]
```

```
print(f"Accuracy: {xgbAccuracy}")
print(f"Precision: {xgbPrecision}")
print(f"Recall: {xgbRecall}")
print(f"F1 Score: {xgbF1}")
print(f"Log Loss: {xgbLogLoss}")
print(f"AUC Score: {xgbAucScore}")
print("Confusion Matrix:")
print(xgbConfusionMatrix)
```

```
Accuracy: 0.9682496607869742
Precision: 0.9611510791366906
Recall: 0.9823529411764705
F1 Score: 0.9716363636363635
Log Loss: 1.0966354424956306
AUC Score: 0.9665564097979618
Confusion Matrix:
[[1564   81]
 [  36 2004]]
```

```
print(f"Accuracy: {mlpAccuracy}")
print(f"Precision: {mlpPrecision}")
print(f"Recall: {mlpRecall}")
print(f"F1 Score: {mlpF1}")
print(f"Log Loss: {mlpLogLoss}")
print(f"AUC Score: {mlpAucScore}")
print("Confusion Matrix:")
print(mlpConfusionMatrix)
```

```
Accuracy: 0.9682496607869742
Precision: 0.9611510791366906
Recall: 0.9823529411764705
F1 Score: 0.9716363636363635
Log Loss: 1.0966354424956306
AUC Score: 0.9665564097979618
Confusion Matrix:
[[1564   81]
 [  36 2004]]
```

CHAPTER -10

Advantages of web phishing detection

1. URL verification is one of the great advantages of three-factor authentication. According to a trusted source, 79% of phishing attacks are blocked by URL verification. Open ID decreases the detection times of phishing attacks.
2. Users can identify legitimate and illegitimate websites
3. Secure browsing while using unknown website links

Disadvantages of web phishing detection

1. User cannot check all the websites they visit as it is a time-consuming process
2. This is an approach to detect phishing websites not a permanent solution/tool
3. Requires internet connection for this model to function
4. The system will be useless if the user has already entered into the malicious website

CHAPTER 11

CONCLUSION

It is remarkable that a good anti-phishing tool should be able to predict attacks in a reasonable amount of time. We acknowledge that a good anti-phishing tool must be available in a timely manner in order to expand the scope of phishing site detection. Consistent retraining should be used to continuously enhance this device. Our model resolves this issue by using randon=m forest method . As a result, if we build a model to combat phishing and need to update it for any reason, our model will support this process because it automates the organizing process and requires hardly any client-defined parameters.

CHAPTER-12

Future Scope

Deep learning methods will be used to predict the phishing websites with more accuracy and improving the website more user-friendly. Also adding this project or feature as an extension in search engines. 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-15646-1659602252

Project Demo Link:
https://drive.google.com/file/d/1DPUSep_vz4TIHKV88IrplaMALcMqAJXU/view?usp=drivesdk