IBM
NALAIYA THIRAN

PROJECT REPORT
ON
WEB PHISHING DETECTION

TEAM ID: PNT2022TMID03756

SAVEETHA ENGINEERING COLLEGE

Team Members:

Varshini M
Vaishnavi R
Karthika Devi J
Karan Kumar B

TABLE OF CONTENTS

ABSTRACT

As a result of the current constant expansion of data and the expanding availability of the internet, people are constantly considering new ways to obtain data from others, typically via the internet. Phishing is one of the most common methods of stealing information. Phishing is a type of online fraud in which users use a variety of techniques to obtain sensitive data from others, such as credit card information. Phishing has become a significant threat to many people and businesses as the internet has grown. As new techniques for combating phishing emerge, so do new methods of information theft. People are now exposed to a plethora of dangers as new phishing techniques emerge from all types of people and locations. Several approaches have been developed in recent years to either prevent or detect phishing. Although the risks associated with phishing have not been completely eliminated, they have been greatly reduced. The methods used to prevent and detect phishing have both advantages and disadvantages. Many used examples have been shown, along with their primary weaknesses and the numerous anti-phishing strategies that were used to prevent 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
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 type of social engineering in which an attacker sends a bogus (e.g., spoofed, false, or otherwise misleading) communication in order to trick a victim into giving the attacker access to sensitive information or installing dangerous software, such as ransomware, on the victim's infrastructure. Phishing attacks are becoming increasingly sophisticated, and they frequently transparently mirror the site being attacked, allowing the attacker to observe everything the victim does there and to bypass any additional security barriers.

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.

Email phishing, spear phishing, voice phishing, SMS phishing, and clone phishing are examples of phishing attacks. Our project assists us in detecting illegitimate websites by analysing the given datasets with the help of some parameters such as IP address, url length, rightclick, domain registration, having @ symbol in the url, and so on, and training the machine learning model with the given dataset. Also, create a website that detects phishing websites and deploy it in a 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

2.1 STUDY

Detecting Phishing Websites Using Machine Learning

Amani Alswailem, Bashayr Alabdullah

Phishing websites are one of the internet security problems that target human vulnerabilities rather than software vulnerabilities. It can be described as the process of attracting online users to obtain their sensitive information such as usernames and passwords. The system acts as an additional functionality to an internet browser as an extension that automatically notifies the user when it detects a phishing website. The system is based on a machine learning method, particularly supervised learning.
We have selected the Random Forest technique due to its good performance in classification.It focuses to pursue a higher performance classifier by studying the features of phishing websites and choose the better combination of them to train the classifier. As a result, they conclude their paper with an accuracy of 98.8% and a combination of 26 features.

Real Time Detection Of Phishing Websites

Abdulghani Ali Ahmed, Nurul Amirah Abdullah

Web Spoofing lures the user to interact with the fake websites rather than the real ones. The main objective of this attack is to steal the sensitive information from the users.
The attacker creates a ‘shadow’ website that looks similar to the legitimate website. This fraudulent act allows the attacker to observe and modify any information from the user. This paper proposes a detection technique of phishing websites based on checking Uniform Resources Locators (URLs) of web pages. The proposed solution is able to distinguish between the legitimate web page and fake web page by checking the Uniform Resources Locators (URLs) of suspected web pages. URLs are inspected based on particular characteristics to check the phishing web pages.The detected attacks are reported for prevention. The performance of the proposed solution is evaluated using Phistank and Yahoo directory datasets. The obtained results show that the detection mechanism is deployable and capable of detecting various types of phishing attacks maintaining a low rate of false alarms.

Detection of Phishing Websites from URL’s by using Classification Techniques on WEKA

Buket Geyik; Kübra Erensoy; Emre Kocyigit

Phishing is a type of fraud committed by intruders by using fake web pages to access people's private information such as user-id, password, credit card number and bank account numbers, etc. These scammers can also send email from many important institutions and organizations by using phishing attacks which imitate these web pages and act as if they are original. Traditional security mechanisms can not prevent these attacks because they directly target the weakest part of connection: end-users.
Machine learning technology has been used to detect and prevent this type of intrusions. The anti-phishing method has been developed by detecting the attacks made with the technologies used. In this paper, they combined the websites used by phishing attacks into a dataset, then obtained some results using 4 classification algorithms with this dataset. The experimental results showed that the proposed systems give very good accuracy levels for the detection of these attacks.

An effective detection approach for phishing websites using URL and HTML features

Qingshan Jiang, Abdur Rasool ,Hui Chen, Qiang Qu & Yang Wang

This paper proposes a new approach to solve the anti-phishing problem. The new features of this approach can be represented by URL character sequence without phishing prior knowledge, various hyperlink information, and textual content of the webpage, which are combined and fed to train the XGBoost classifier. One of the major contributions of this paper is the selection of different new features, which are capable enough to detect 0-h attacks, and these features do not depend on any third-party services. In particular, we extract character level Term Frequency-Inverse Document Frequency (TF-IDF) features from noisy parts of HTML and plaintext of the given webpage. Moreover, our proposed hyperlink features determine the relationship between the content and the URL of a webpage.

A Desktop Application to Detect Phishing Webpages through Heuristic Approach

Routhu Srinivasa Rao and Syed Taqi Ali

In this paper, we implemented a desktop application called PhishShield, which concentrates on URL and Website Content of phishing page. PhishShield takes URL as input and outputs the status of URL as phishing or legitimate website. The heuristics used to detect phishing are footer links with null value, zero links in body of html, copyright content, title content and website identity. PhishShield is able to detect zero hour phishing attacks which blacklists unable to detect and it is faster than visual based assessment techniques that are used in detecting phishing. The accuracy rate obtained for PhishShield is 96.57% and covers a wide range of phishing web sites resulting less false negative and false positive rate.

2.2 EXISITING PROBLEM

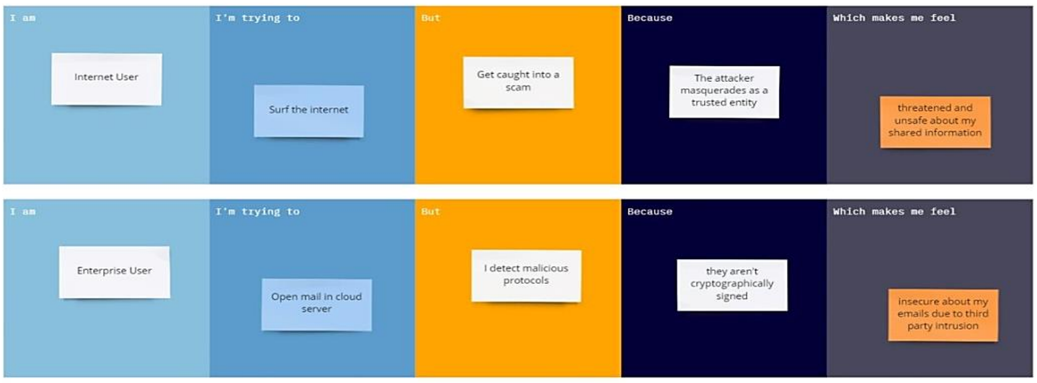
The problem with phishing is that perpetrators are constantly coming up with new and inventive ways to trick people into thinking their activities are connected to a trustworthy website or email. Phishers are becoming more skilled at creating fake websites that look exactly like the real thing. They've even begun to include logos and images in their phishing emails to increase their effectiveness. There are dangerous new sophisticated phishing techniques that use publicly available personal data to create realistic and convincing assaults that target victims directly. Social phishing and context-aware phishing are two attacks that take advantage of the massive amount of publicly available data to make their schemes more effective.

2.3 DEFINING PROBLEM STATEMENT

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 onseveral 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.

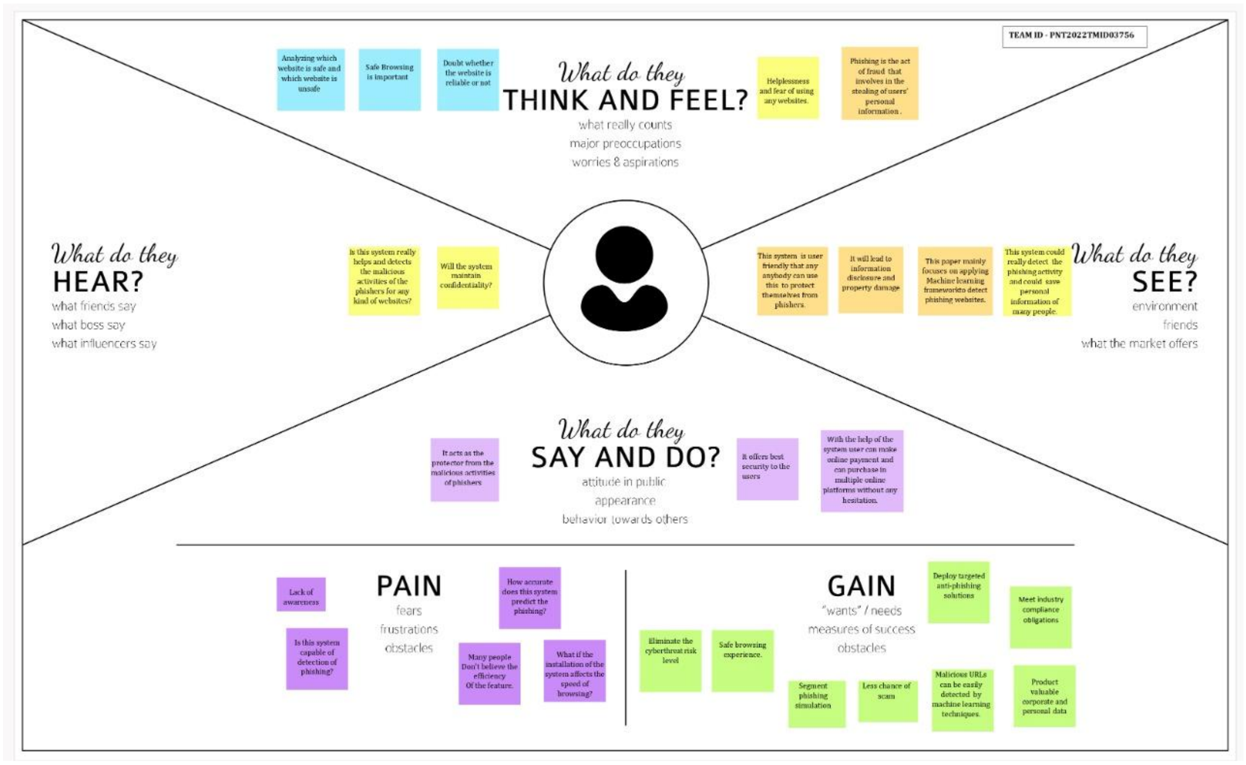
Phishing is said to be the most deceptive attack of all these. In order to achieve maximum accuracy and a clear model, our primary goal in this paper is the classification of a phishing website using various machine learning techniques.


I am
Internet User
I'm trying to
Surf the internet
But
Get caught into a scam
Because
The attacker masquerades as a trusted entity
Which makes me feel
threatened and unsafe about my shared information
I am
Enterprise User
I'm trying to
Open mail in cloud server
But
I detect malicious protocols
Because
they aren't cryptographically signed
Which makes me feel
Insecure about my emails due to third party intrusion

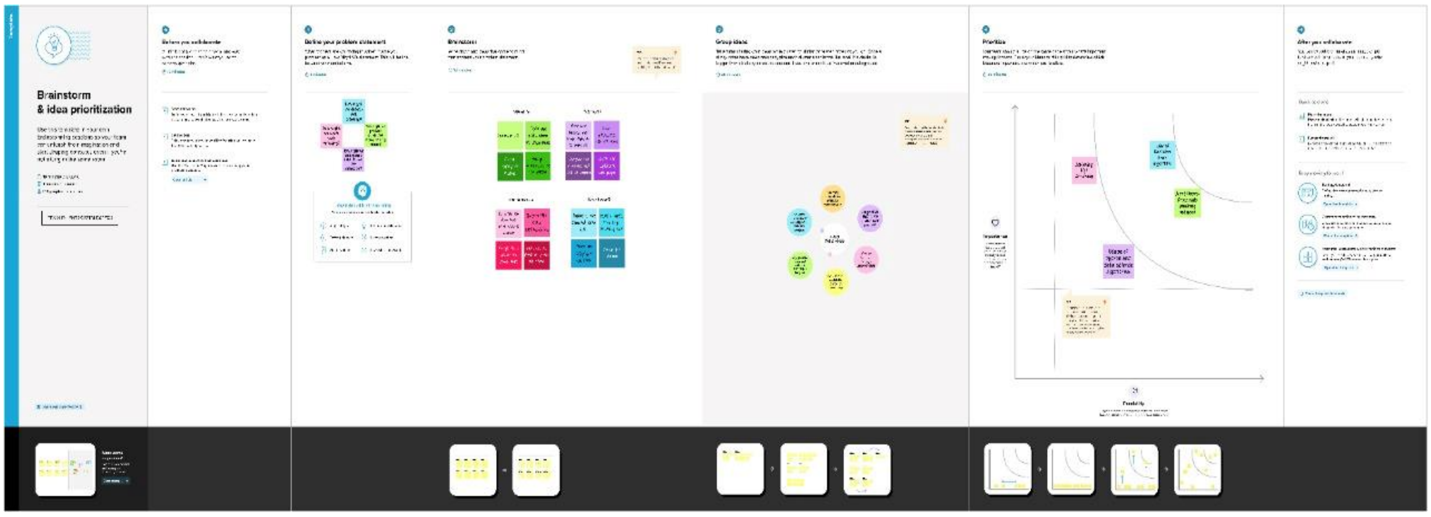
CHAPTER 3

IDEATION & PROPOSED SOLUTION

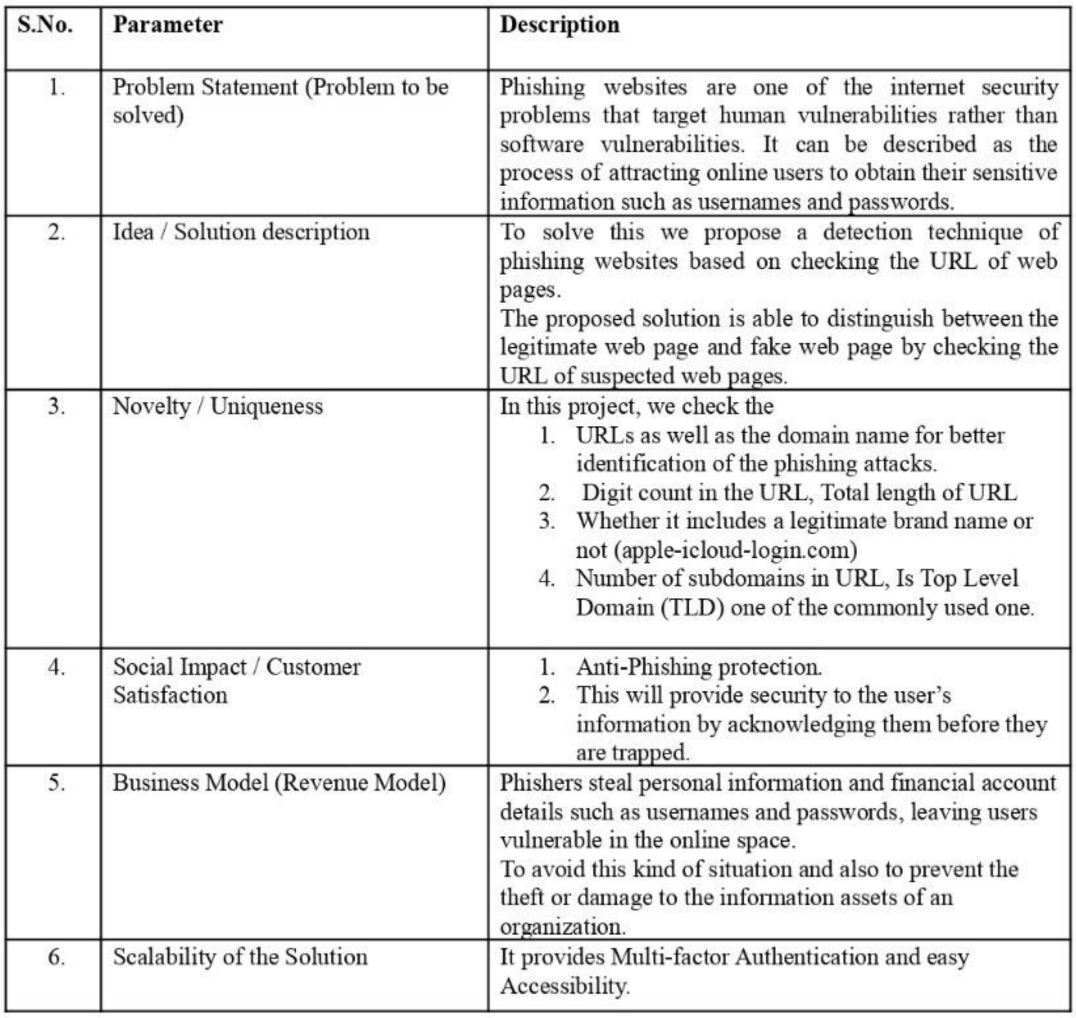
3.1 EMPATHY MAP CANVAS


TEAM ID - PNT2022TMID03756
Analyzing which website is safe and which website is unsafe
Safe Browsing is important
Doubt whether the website is reliable or not
What do they THINK AND FEEL?
what really counts
major preoccupations
worries & aspirations
Helplessness and fear of using any websites.
Phishing is the act of fraud that involves in the stealing of users' personal information .
What do they HEAR?
what friends say
what boss say
what influencers say
Is this system really helps and detects the malicious activities of the phishers for any kind of websites?
Will the system maintain confidentiality?
This system is user friendly that any anybody can use this to protect themselves from phishers.
It will lead to information disclosure and property damage
This paper mainly focuses on applying Machine learning frameworkto detect phishing websites.
This system could really detect the phishing activity and could save personal information of many people.
What do they SEE?
environment
friends
what the market offers
It acts as the protector from the malicious activities of phishers
What do they SAY AND DO?
attitude in public
appearance
behavior towards others
It offers best security to the users
With the help of the system user can make online payment and can purchase in multiple online platforms without any hesitation.
Lack of awareness
Is this system capable of detection of phishing?
PAIN
fears
frustrations
obstacles
How accurate does this system predict the phishing?
Many people Don't believe the efficiency Of the feature.
What if the installation of the system affects the speed of browsing?
Eliminate the cyberthreat risk level
Safe browsing experience.
GAIN
"wants" / needs
measures of success
obstacles
Deploy targeted anti-phishing solutions
Meet industry compliance obligations
Segment phishing simulation
Less chance of scam
Malicious URLs can be easily detected by machine learning techniques.
Product valuable corporate and personal data

3.2 BRAIN STORMING

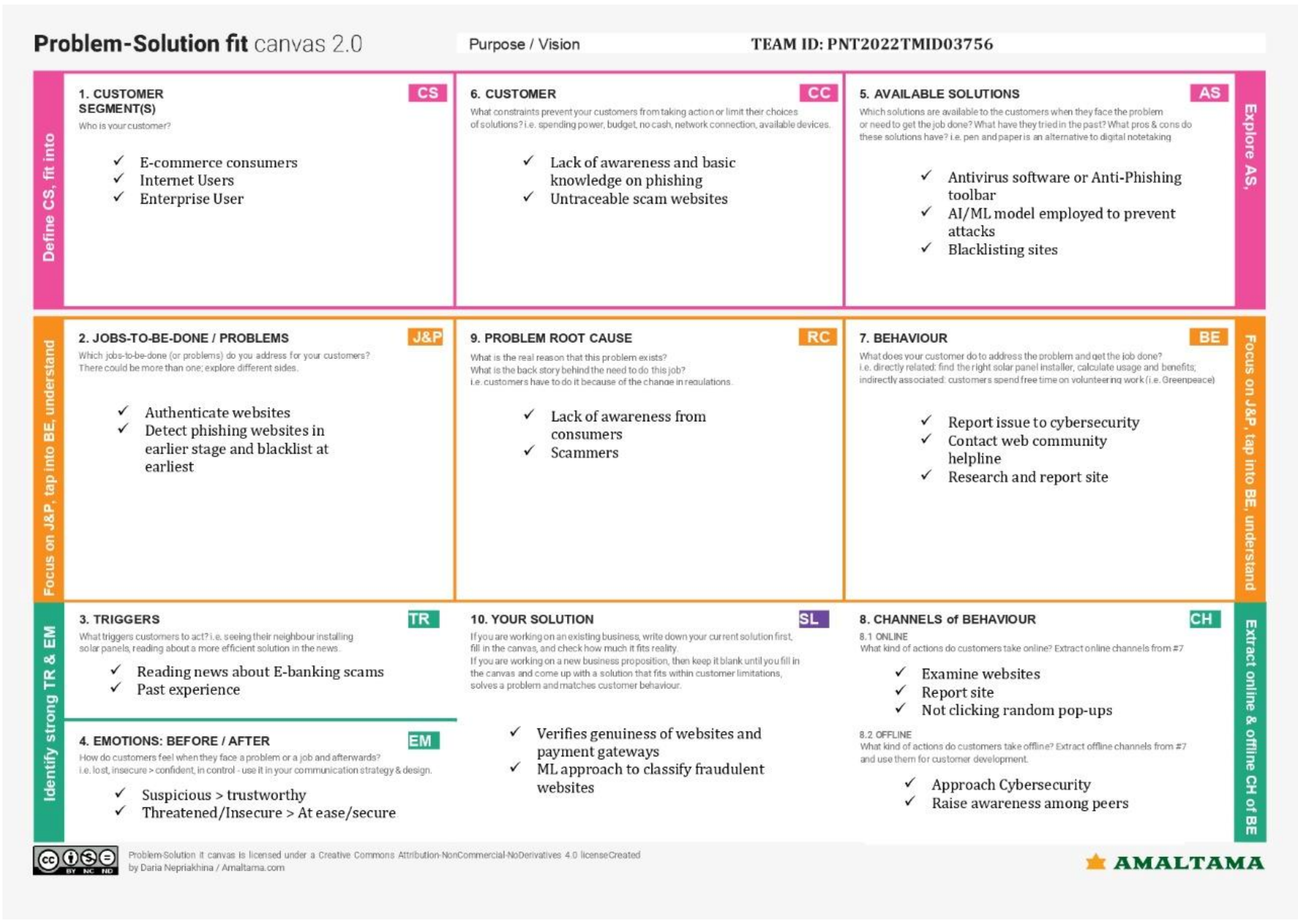


3.3 PROPOSED SOLUTION



S.No.	Parameter	Description
1.	Problem Statement (Problem to be solved)	Phishing websites are one of the internet security problems that target human vulnerabilities rather than software vulnerabilities. It can be described as the process of attracting online users to obtain their sensitive information such as usernames and passwords.
2.	Idea / Solution description	To solve this we propose a detection technique of phishing websites based on checking the URL of web pages. The proposed solution is able to distinguish between the legitimate web page and fake web page by checking the URL of suspected web pages.
3.	Novelty / Uniqueness	In this project, we check the 1. URLs as well as the domain name for better identification of the phishing attacks. 2. Digit count in the URL, Total length of URL 3. Whether it includes a legitimate brand name or not (apple-icloud-login.com) 4. Number of subdomains in URL, Is Top Level Domain (TLD) one of the commonly used one.
4.	Social Impact / Customer Satisfaction	1. Anti-Phishing protection. 2. This will provide security to the user's information by acknowledging them before they are trapped.
5.	Business Model (Revenue Model)	Phishers steal personal information and financial account details such as usernames and passwords, leaving users vulnerable in the online space. To avoid this kind of situation and also to prevent the theft or damage to the information assets of an organization.
6.	Scalability of the Solution	It provides Multi-factor Authentication and easy Accessibility.

3.4 PROBLEM SOLUTION FIT



Problem-Solution fit canvas 2.0

Purpose / Vision

TEAM ID: PNT2022TMID03756

Define CS, fit into

1. CUSTOMER SEGMENT(S) CS

Who is your customer?

- ✓ E-commerce consumers
- ✓ Internet Users
- ✓ Enterprise User

6. CUSTOMER CC

What constraints prevent your customers from taking action or limit their choices of solutions? i.e. spending power, budget, no cash, network connection, available devices.

- ✓ Lack of awareness and basic knowledge on phishing
- ✓ Untraceable scam websites

5. AVAILABLE SOLUTIONS AS

Which solutions are available to the customers when they face the problem or need to get the job done? What have they tried in the past? What pros & cons do these solutions have? i.e. pen and paper is an alternative to digital notetaking

- ✓ Antivirus software or Anti-Phishing toolbar
- ✓ AI/ML model employed to prevent attacks
- ✓ Blacklisting sites

Explore AS,

Focus on J&P, tap into BE, understand

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

Which jobs-to-be-done (or problems) do you address for your customers? There could be more than one; explore different sides.

- ✓ Authenticate websites
- ✓ Detect phishing websites in earlier stage and blacklist at earliest

9. PROBLEM ROOT CAUSE RC

What is the real reason that this problem exists?
What is the back story behind the need to do this job?
i.e. customers have to do it because of the change in regulations.

- ✓ Lack of awareness from consumers
- ✓ Scammers

7. BEHAVIOUR BE

What does your customer do to address the problem and get the job done?
i.e. directly related: find the right solar panel installer, calculate usage and benefits; indirectly associated: customers spend free time on volunteering work (i.e. Greenpeace)

- ✓ Report issue to cybersecurity
- ✓ Contact web community helpline
- ✓ Research and report site

Focus on J&P, tap into BE, understand

Identify strong TR & EM

3. TRIGGERS TR

What triggers customers to act? i.e. seeing their neighbour installing solar panels, reading about a more efficient solution in the news.

- ✓ Reading news about E-banking scams
- ✓ Past experience

4. EMOTIONS: BEFORE / AFTER EM

How do customers feel when they face a problem or a job and afterwards?
i.e. lost, insecure > confident, in control - use it in your communication strategy & design.

- ✓ Suspicious > trustworthy
- ✓ Threatened/Insecure > At ease/secure

10. YOUR SOLUTION SL

If you are working on an existing business, write down your current solution first, fill in the canvas, and check how much it fits reality.
If you are working on a new business proposition, then keep it blank until you fill in the canvas and come up with a solution that fits within customer limitations, solves a problem and matches customer behaviour.

- ✓ Verifies genuiness of websites and payment gateways
- ✓ ML approach to classify fraudulent websites

8. CHANNELS of BEHAVIOUR CH

8.1 ONLINE
What kind of actions do customers take online? Extract online channels from #7

- ✓ Examine websites
- ✓ Report site
- ✓ Not clicking random pop-ups

8.2 OFFLINE
What kind of actions do customers take offline? Extract offline channels from #7 and use them for customer development.

- ✓ Approach Cybersecurity
- ✓ Raise awareness among peers

Extract online & offline CH of BE

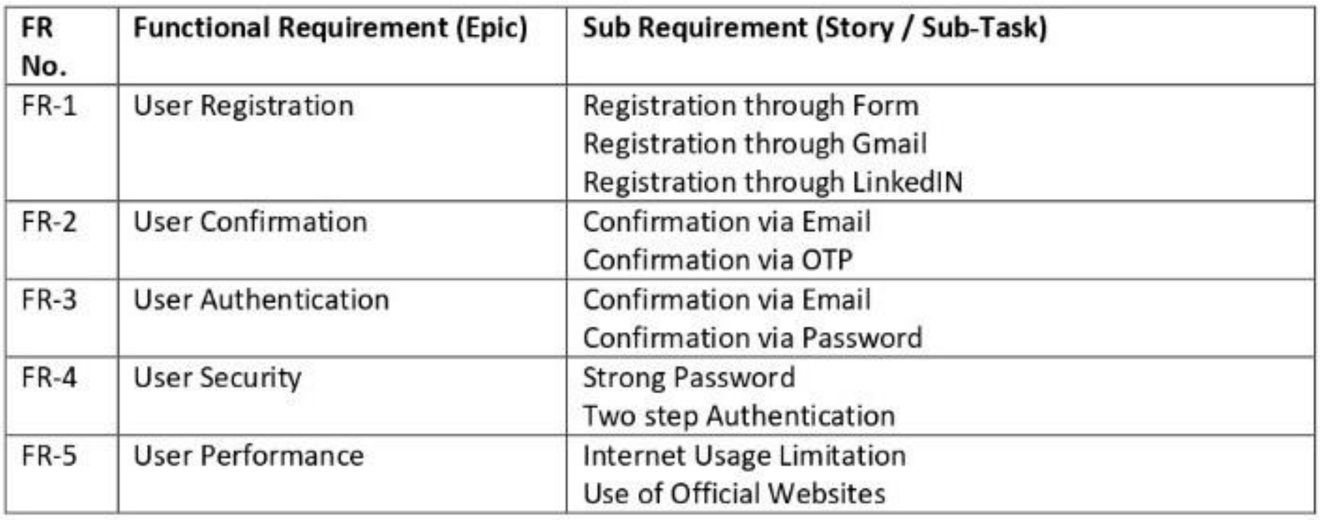
AMALTAMA

CHAPTER 4

REQUIREMENT ANALYSIS

4.1 FUNCTIONAL REQUIREMENTS:

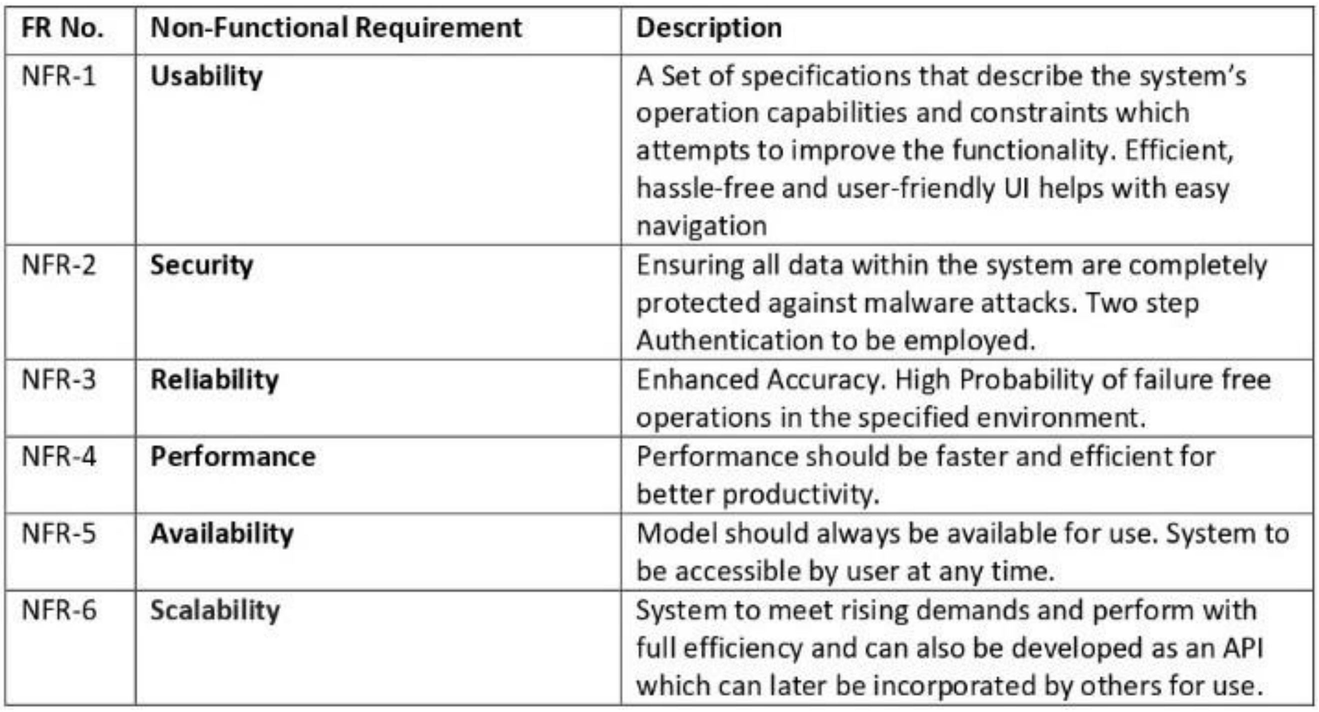
Following are the functional requirements of the proposed solution.



FR No.	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
FR-1	User Registration	Registration through Form Registration through Gmail Registration through LinkedIN
FR-2	User Confirmation	Confirmation via Email Confirmation via OTP
FR-3	User Authentication	Confirmation via Email Confirmation via Password
FR-4	User Security	Strong Password Two step Authentication
FR-5	User Performance	Internet Usage Limitation Use of Official Websites

4.2 NON-FUNCTIONAL REQUIREMENTS:

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

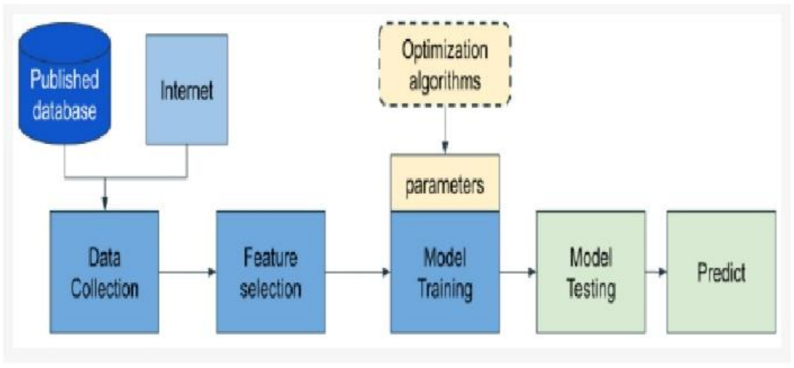


FR No.	Non-Functional Requirement	Description
NFR-1	**Usability**	A Set of specifications that describe the system's operation capabilities and constraints which attempts to improve the functionality. Efficient, hassle-free and user-friendly UI helps with easy navigation
NFR-2	**Security**	Ensuring all data within the system are completely protected against malware attacks. Two step Authentication to be employed.
NFR-3	**Reliability**	Enhanced Accuracy. High Probability of failure free operations in the specified environment.
NFR-4	**Performance**	Performance should be faster and efficient for better productivity.
NFR-5	**Availability**	Model should always be available for use. System to be accessible by user at any time.
NFR-6	**Scalability**	System to meet rising demands and perform with full efficiency and can also be developed as an API which can later be incorporated by others for use.

CHAPTER 5

PROJECT DESIGN

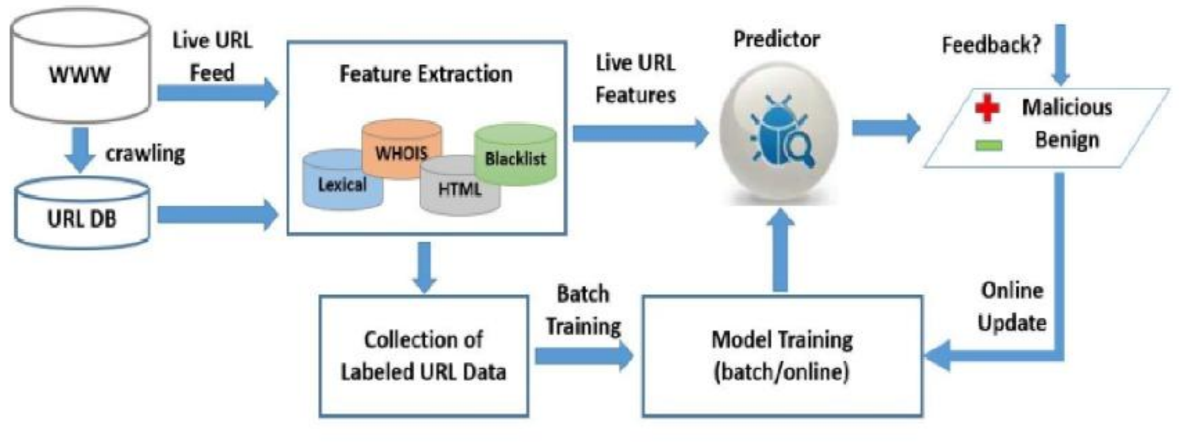
5.1 DATA FLOW DIAGRAM:



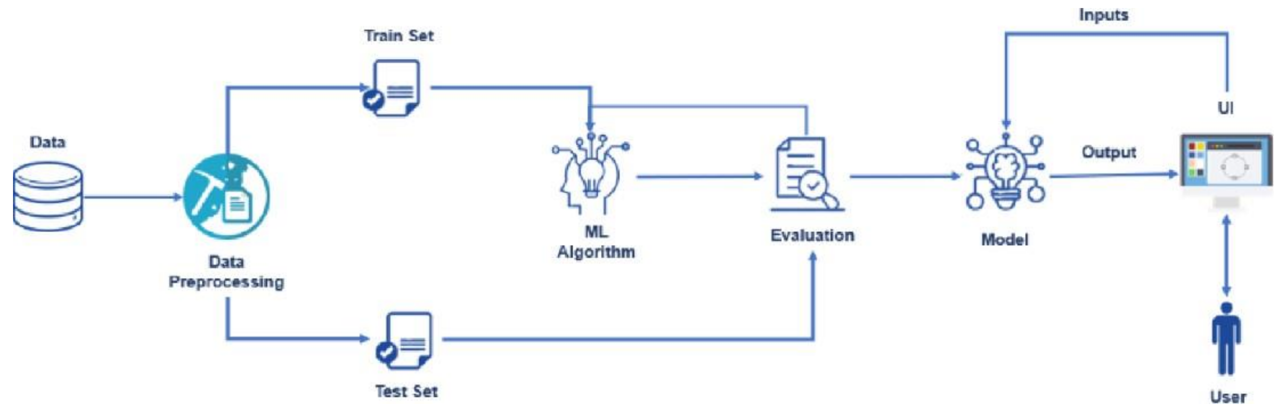
A Data Flow Diagram (DFD) is a traditional visual representation of the information flows within a system. A neat and clear DFD can depict the right amount of the system requirement graphically. It shows how data enters and leaves the system, what changes the information, and where data is stored.

5.2 SOLUTION AND TECHNICAL ARCHITECTURE:

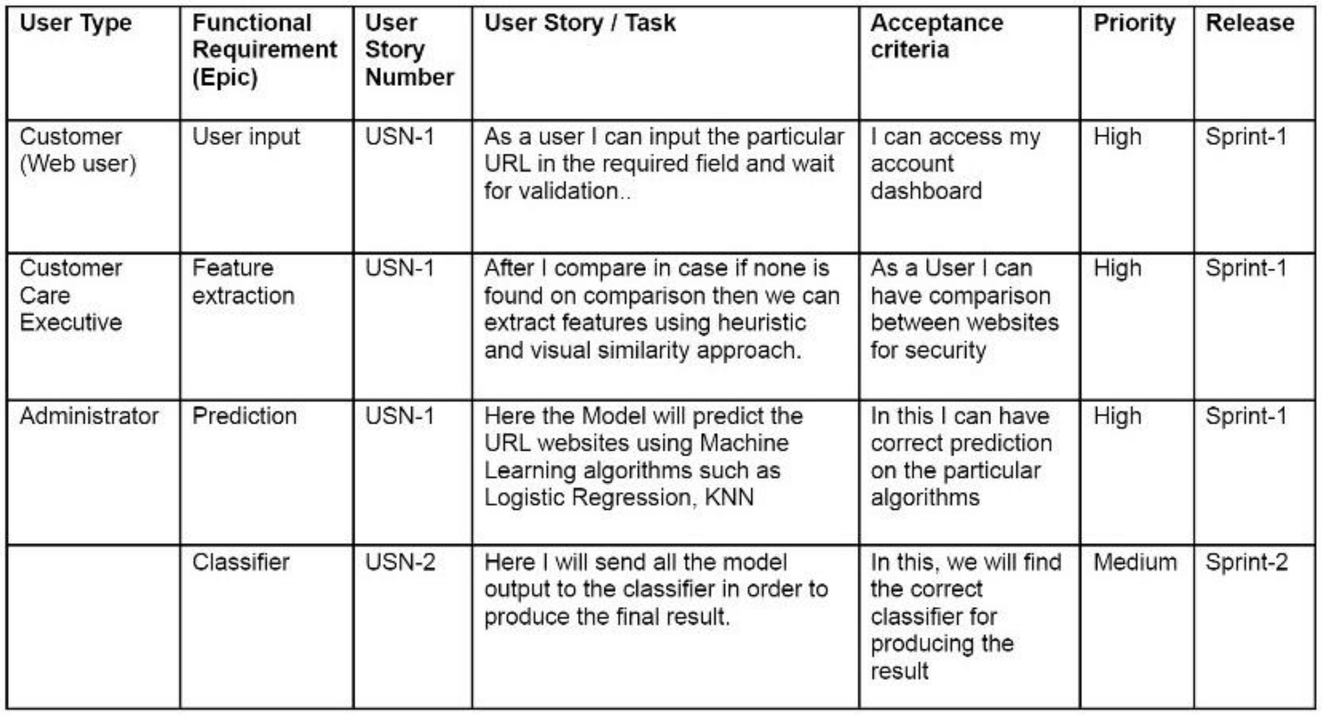
Solution Architecture:



Technical Architecture:



5.3 USER STORIES

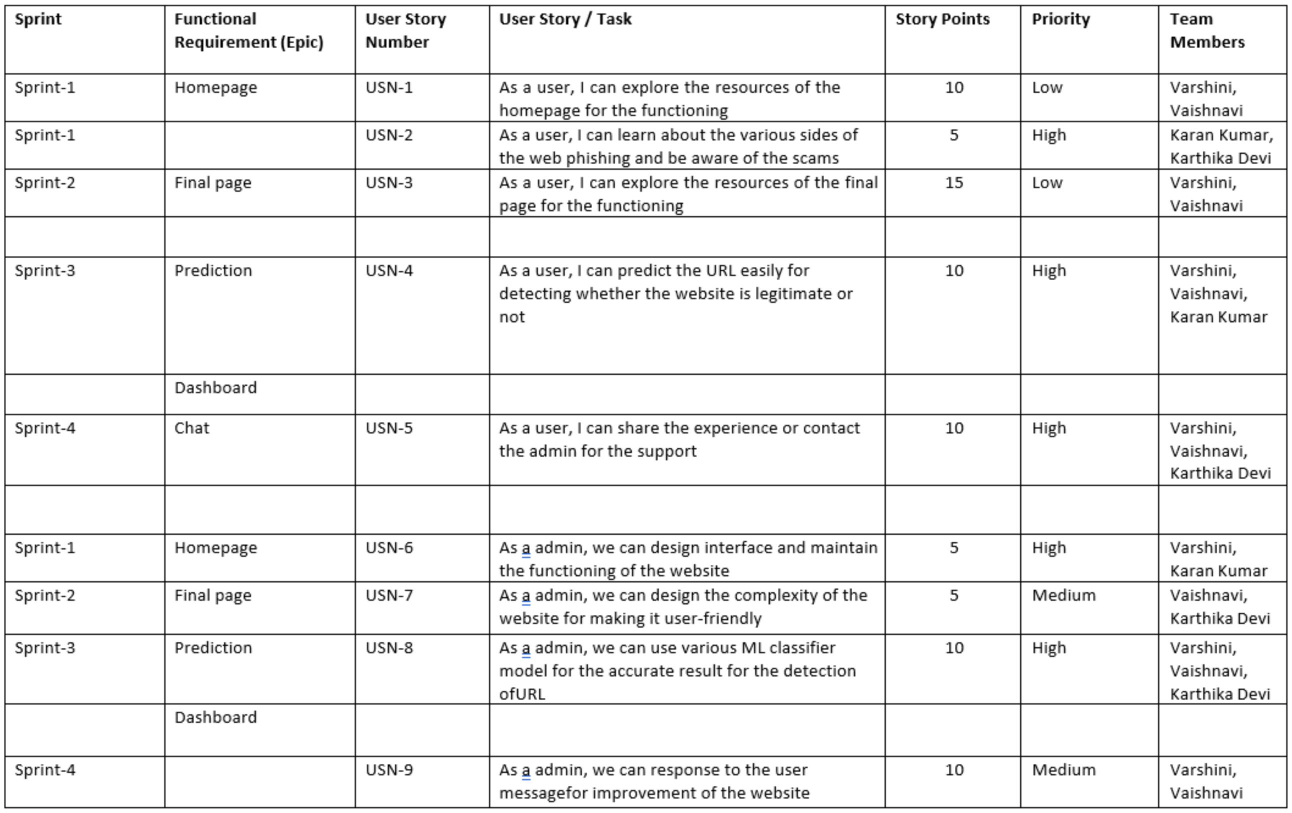


User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
Customer (Web user)	User input	USN-1	As a user I can input the particular URL in the required field and wait for validation..	I can access my account dashboard	High	Sprint-1
Customer Care Executive	Feature extraction	USN-1	After I compare in case if none is found on comparison then we can extract features 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 I can have correct prediction on the particular algorithms	High	Sprint-1
	Classifier	USN-2	Here I will send all the model output to the classifier in order to produce the final result.	In this, we will find the correct classifier for producing the result	Medium	Sprint-2

CHAPTER 6

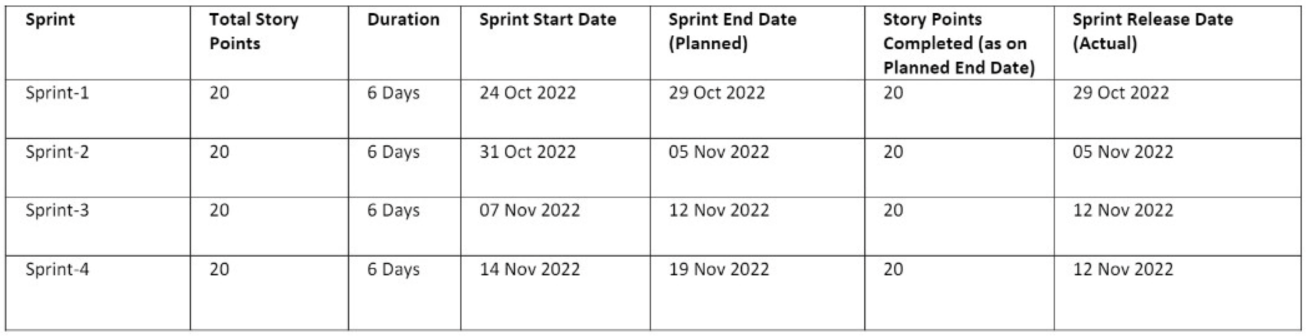
PROJECT PLANNING & SCHEDULING

6.1 SPRINT PLANNING



Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-1	Homepage	USN-1	As a user, I can explore the resources of the homepage for the functioning	10	Low	Varshini, Vaishnavi
Sprint-1		USN-2	As a user, I can learn about the various sides of the web phishing and be aware of the scams	5	High	Karan Kumar, Karthika Devi
Sprint-2	Final page	USN-3	As a user, I can explore the resources of the final page for the functioning	15	Low	Varshini, Vaishnavi
Sprint-3	Prediction	USN-4	As a user, I can predict the URL easily for detecting whether the website is legitimate or not	10	High	Varshini, Vaishnavi, Karan Kumar
	Dashboard					
Sprint-4	Chat	USN-5	As a user, I can share the experience or contact the admin for the support	10	High	Varshini, Vaishnavi, Karthika Devi
Sprint-1	Homepage	USN-6	As a admin, we can design interface and maintain the functioning of the website	5	High	Varshini, Karan Kumar
Sprint-2	Final page	USN-7	As a admin, we can design the complexity of the website for making it user-friendly	5	Medium	Vaishnavi, Karthika Devi
Sprint-3	Prediction	USN-8	As a admin, we can use various ML classifier model for the accurate result for the detection ofURL	10	High	Varshini, Vaishnavi, Karthika Devi
	Dashboard					
Sprint-4		USN-9	As a admin, we can response to the user messagefor improvement of the website	10	Medium	Varshini, Vaishnavi

6.2 SPRINT DELIVERY



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	12 Nov 2022

CHAPTER-7

CODING SECTION

7.1 Code

//app.py

```
import numpy as np
import pickle
import inputScript
from flask import Flask, render_template, request
app = Flask(_name_)

model = pickle.load(open('phishing_website.pkl','rb'))

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

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

@app.route("/y_predict", methods = ['GET', 'POST'])
def y_predict():
    geturl = request.form['url']
    check_prediction = inputScript.main(geturl)
    prediction = model.predict(check_prediction)
    print(prediction)
    output = prediction[0]
    if(output==1):
        pred ='Your are safe!! This is a Legitimate Website.'
    else:
        pred = 'You are on the wrong site. Be cautions!'
    return render_template('final.html',url_path = geturl,url = pred)



if _name_ == '_main_':
   app.run(debug = True)
```

//inputScript.py

```
import regex
from tldextract import extract
import ssl
import socket
from bs4 import BeautifulSoup
import urllib.request
import whois
import datetime


def url_having_ip(url):
#using regular function
#    symbol = regex.findall(r'(http((s)?)://)((((\d)+).)*)((\w)+)(/((\w)+))?',url)
 #   if(len(symbol)!=0):
  #      having_ip = 1 #phishing
   # else:
    #    having_ip = -1 #legitimate
    #return(having_ip)
    return 0


def url_length(url):
    length=len(url)
    if(length<54):
        return -1
    elif(54<=length<=75):
        return 0
    else:
        return 1


def url_short(url):
    #ongoing
    return 0

def having_at_symbol(url):
    symbol=regex.findall(r'@',url)
    if(len(symbol)==0):
        return -1
    else:
        return 1

def doubleSlash(url):
    #ongoing
    return 0
```

```
def prefix_suffix(url):
    subDomain, domain, suffix = extract(url)
    if(domain.count('-')):
        return 1
    else:
        return -1

def sub_domain(url):
    subDomain, domain, suffix = extract(url)
    if(subDomain.count('.')==0):
        return -1
    elif(subDomain.count('.')==1):
        return 0
    else:
        return 1

def SSLfinal_State(url):
    try:
#check wheather contains https
        if(regex.search('^https',url)):
            usehttps = 1
        else:
            usehttps = 0
#getting the certificate issuer to later compare with trusted issuer
        #getting host name
        subDomain, domain, suffix = extract(url)
        host_name = domain + "." + suffix
        context = ssl.create_default_context()
        sct = context.wrap_socket(socket.socket(), server_hostname = host_name)
        sct.connect((host_name, 443))
        certificate = sct.getpeercert()
        issuer = dict(x[0] for x in certificate['issuer'])
        certificate_Auth = str(issuer['commonName'])
        certificate_Auth = certificate_Auth.split()
        if(certificate_Auth[0] == "Network" or certificate_Auth == "Deutsche"):
            certificate_Auth = certificate_Auth[0] + " " + certificate_Auth[1]
        else:
            certificate_Auth = certificate_Auth[0]
        trusted_Auth = 
['Comodo','Symantec','GoDaddy','GlobalSign','DigiCert','StartCom','Entrust','Verizon','Trustwave','Unizeto'
,'Buypass','QuoVadis','Deutsche Telekom','Network 
Solutions','SwissSign','IdenTrust','Secom','TWCA','GeoTrust','Thawte','Doster','VeriSign']
#getting age of certificate
        startingDate = str(certificate['notBefore'])
        endingDate = str(certificate['notAfter'])
        startingYear = int(startingDate.split()[3])
        endingYear = int(endingDate.split()[3])
        Age_of_certificate = endingYear-startingYear
```

```
#checking final conditions
    if((usehttps==1) and (certificate_Auth in trusted_Auth) and (Age_of_certificate>=1) ):
        return -1 #legitimate
    elif((usehttps==1) and (certificate_Auth not in trusted_Auth)):
        return 0 #suspicious
    else:
        return 1 #phishing


except Exception as e:

    return 1

def domain_registration(url):
  try:
    w = whois.whois(url)
    updated = w.updated_date
    exp = w.expiration_date
    length = (exp[0]-updated[0]).days
    if(length<=365):
      return 1
    else:
      return -1
  except:
    return 0

def favicon(url):
  #ongoing
  return 0

def port(url):
  #ongoing
  return 0

def https_token(url):
  subDomain, domain, suffix = extract(url)
  host =subDomain +'.' + domain + '.' + suffix
  if(host.count('https')): #attacker can trick by putting https in domain part
    return 1
  else:
    return -1

def request_url(url):
  try:
    subDomain, domain, suffix = extract(url)
    websiteDomain = domain

    opener = urllib.request.urlopen(url).read()
    soup = BeautifulSoup(opener, 'lxml')
    imgs = soup.findAll('img', src=True)
    total = len(imgs)
```

```
        linked_to_same = 0
        avg =0
        for image in imgs:
            subDomain, domain, suffix = extract(image['src'])
            imageDomain = domain
            if(websiteDomain==imageDomain or imageDomain==''):
                linked_to_same = linked_to_same + 1




vids = soup.findAll('video', src=True)
        total = total + len(vids)

        for video in vids:
            subDomain, domain, suffix = extract(video['src'])
            vidDomain = domain
            if(websiteDomain==vidDomain or vidDomain==''):
                linked_to_same = linked_to_same + 1
        linked_outside = total-linked_to_same
        if(total!=0):
            avg = linked_outside/total

        if(avg<0.22):
            return -1
        elif(0.22<=avg<=0.61):
            return 0
        else:
            return 1
    except:
        return 0


def url_of_anchor(url):
    try:
        subDomain, domain, suffix = extract(url)
        websiteDomain = domain

        opener = urllib.request.urlopen(url).read()
        soup = BeautifulSoup(opener, 'lxml')
        anchors = soup.findAll('a', href=True)
        total = len(anchors)
        linked_to_same = 0
        avg = 0
        for anchor in anchors:
            subDomain, domain, suffix = extract(anchor['href'])
            anchorDomain = domain
            if(websiteDomain==anchorDomain or anchorDomain==''):
                linked_to_same = linked_to_same + 1
        linked_outside = total-linked_to_same
```

```
 if(total!=0):
            avg = linked_outside/total

        if(avg<0.31):
            return -1
        elif(0.31<=avg<=0.67):
            return 0
        else:
            return 1
    except:
        return 0


def Links_in_tags(url):
    try:
        opener = urllib.request.urlopen(url).read()
        soup = BeautifulSoup(opener, 'lxml')

        no_of_meta =0
        no_of_link =0
        no_of_script =0
        anchors=0
        avg =0
        for meta in soup.find_all('meta'):
            no_of_meta = no_of_meta+1
        for link in soup.find_all('link'):
            no_of_link = no_of_link +1
        for script in soup.find_all('script'):
            no_of_script = no_of_script+1
        for anchor in soup.find_all('a'):
            anchors = anchors+1
        total = no_of_meta + no_of_link + no_of_script+anchors
        tags = no_of_meta + no_of_link + no_of_script
        if(total!=0):
            avg = tags/total

        if(avg<0.25):
            return -1
        elif(0.25<=avg<=0.81):
            return 0
        else:
            return 1
    except:
        return 0

def sfh(url):
    #ongoing
    return 0
```

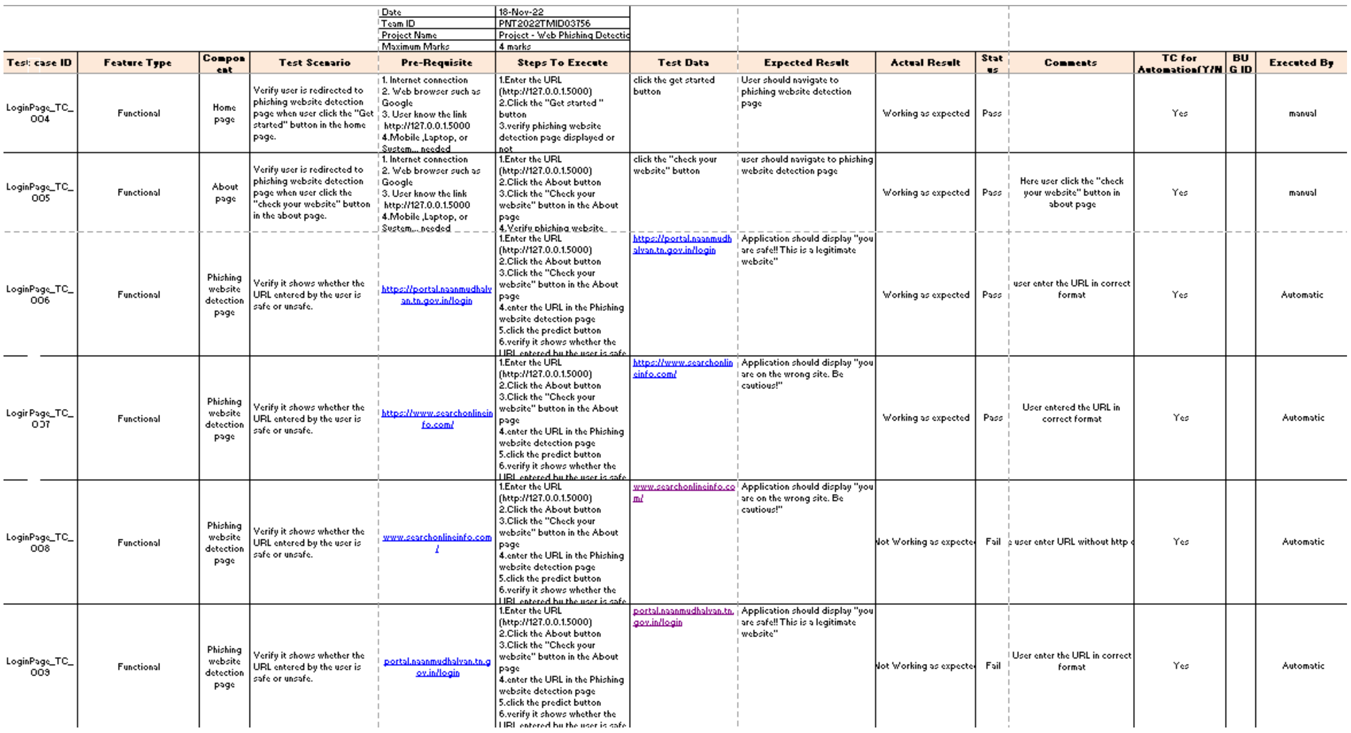
```
def email_submit(url):
    try:
        opener = urllib.request.urlopen(url).read()
        soup = BeautifulSoup(opener, 'lxml')
        if(soup.find('mailto:')):
            return 1
        else:
            return -1
    except:
        return 0

def abnormal_url(url):
    #ongoing
    return 0



def redirect(url):
    #ongoing
    return 0

def on_mouseover(url):
    #ongoing
    return 0

def rightClick(url):
    #ongoing
    return 0

def popup(url):
    #ongoing
    return 0

def iframe(url):
    #ongoing
    return 0

def age_of_domain(url):
    try:
        w = whois.whois(url)
        start_date = w.creation_date
        current_date = datetime.datetime.now()
        age =(current_date-start_date[0]).days
        if(age>=180):
            return -1
        else:
            return 1
    except Exception as e:
        print(e)
        return 0
```

```
def dns(url):
   #ongoing
   return 0

def web_traffic(url):
   #ongoing
   return 0

def page_rank(url):
   #ongoing
   return 0

def google_index(url):
   #ongoing
   return 0


def links_pointing(url):
   #ongoing
   return 0

def statistical(url):
   #ongoing
   return 0

def main(url):



   check = [[url_having_ip(url),url_length(url),url_short(url),having_at_symbol(url),
            doubleSlash(url),prefix_suffix(url),sub_domain(url),SSLfinal_State(url),
             domain_registration(url),favicon(url),port(url),https_token(url),request_url(url),
             url_of_anchor(url),Links_in_tags(url),sfh(url),email_submit(url),abnormal_url(url),
             redirect(url),on_mouseover(url),rightClick(url),popup(url),iframe(url),
             age_of_domain(url),dns(url),web_traffic(url),page_rank(url),google_index(url),
             links_pointing(url),statistical(url)]]


   print(check)
   return check
```

CHAPTER 8

TESTING

8.1 TEST CASES



Date	18-Nov-22
Team ID	PNT2022TMID03756
Project Name	Project - Web Phishing Detectic
Maximum Marks	4 marks

Test case ID	Feature Type	Component	Test Scenario	Pre-Requisite	Steps To Execute	Test Data	Expected Result	Actual Result	Status	Comments	TC for Automation(Y/N	BUG ID	Executed By
LoginPage_TC_004	Functional	Home page	Verify user is redirected to phishing website detection page when user click the "Get started" button in the home page.	1. Internet connection 2. Web browser such as Google 3. User know the link http://127.0.0.1.5000 4.Mobile ,Laptop, or System... needed	1.Enter the URL (http://127.0.0.1.5000) 2.Click the "Get started " button 3.verify phishing website detection page displayed or not	click the get started button	User should navigate to phishing website detection page	Working as expected	Pass		Yes		manual
LoginPage_TC_005	Functional	About page	Verify user is redirected to phishing website detection page when user click the "check your website" button in the about page.	1. Internet connection 2. Web browser such as Google 3. User know the link http://127.0.0.1.5000 4.Mobile ,Laptop, or System... needed	1.Enter the URL (http://127.0.0.1.5000) 2.Click the About button 3.Click the "Check your website" button in the About page 4.Verify phishing website	click the "check your website" button	user should navigate to phishing website detection page	Working as expected	Pass	Here user click the "check your website" button in about page	Yes		manual
LoginPage_TC_006	Functional	Phishing website detection page	Verify it shows whether the URL entered by the user is safe or unsafe.	https://portal.naanmudhalvan.tn.gov.in/login	1.Enter the URL (http://127.0.0.1.5000) 2.Click the About button 3.Click the "Check your website" button in the About page 4.enter the URL in the Phishing website detection page 5.click the predict button 6.verify it shows whether the URL entered by the user is safe	https://portal.naanmudhalvan.tn.gov.in/login	Application should display "you are safe!! This is a legitimate website"	Working as expected	Pass	user enter the URL in correct format	Yes		Automatic
LogirPage_TC_OO7	Functional	Phishing website detection page	Verify it shows whether the URL entered by the user is safe or unsafe.	https://www.searchonlineinfo.com/	1.Enter the URL (http://127.0.0.1.5000) 2.Click the About button 3.Click the "Check your website" button in the About page 4.enter the URL in the Phishing website detection page 5.click the predict button 6.verify it shows whether the URL entered by the user is safe	https://www.searchonlineinfo.com/	Application should display "you are on the wrong site. Be cautious!"	Working as expected	Pass	User entered the URL in correct format	Yes		Automatic
LoginPage_TC_008	Functional	Phishing website detection page	Verify it shows whether the URL entered by the user is safe or unsafe.	www.searchonlineinfo.com/	1.Enter the URL (http://127.0.0.1.5000) 2.Click the About button 3.Click the "Check your website" button in the About page 4.enter the URL in the Phishing website detection page 5.click the predict button 6.verify it shows whether the URL entered by the user is safe	www.searchonlineinfo.com/	Application should display "you are on the wrong site. Be cautious!"	Not Working as expecte	Fail	user enter URL without http	Yes		Automatic
LoginPage_TC_009	Functional	Phishing website detection page	Verify it shows whether the URL entered by the user is safe or unsafe.	portal.naanmudhalvan.tn.gov.in/login	1.Enter the URL (http://127.0.0.1.5000) 2.Click the About button 3.Click the "Check your website" button in the About page 4.enter the URL in the Phishing website detection page 5.click the predict button 6.verify it shows whether the URL entered by the user is safe	portal.naanmudhalvan.tn.gov.in/login	Application should display "you are safe!! This is a legitimate website"	Not Working as expecte	Fail	User enter the URL in correct format	Yes		Automatic

8.2 UAT EXECUTION

Purpose of Document

The purpose of this document is to briefly explain the test coverage and open issues of theWeb 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, 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	11	2	4	20	37
Not Reproduced	0	0	1	0	1
Skipped	0	0	1	1	2
Won't Fix	0	5	2	1	8
Totals	24	14	13	26	77

Test Case Analysis

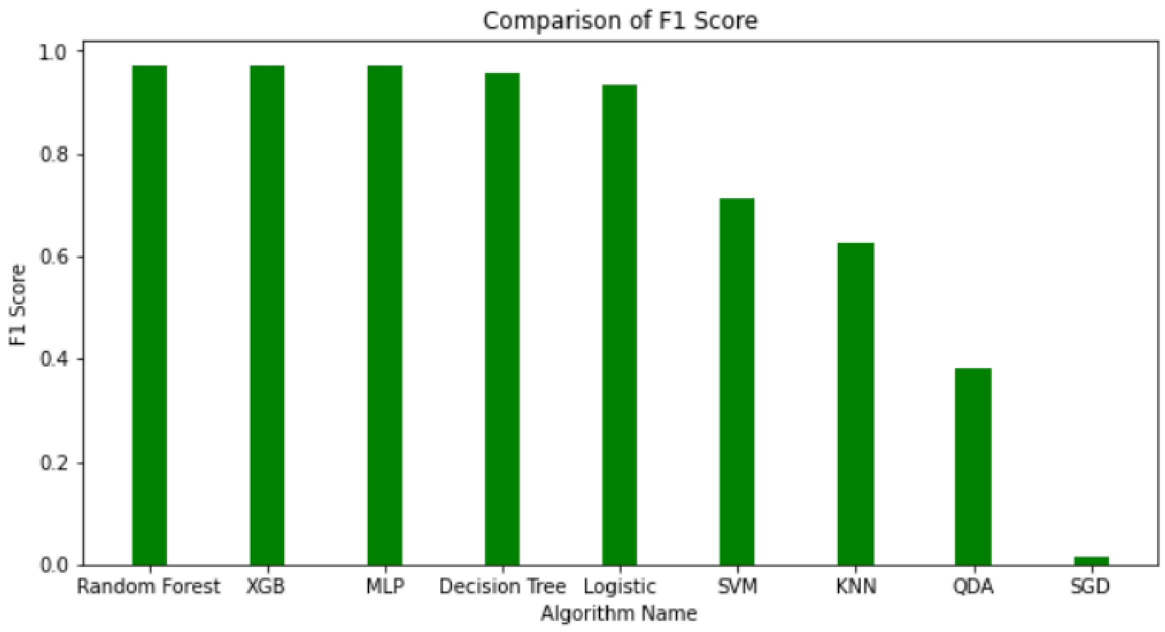
This report shows the number of test cases that have passed, failed, and untested

Section	Total Cases	Not Tested	Fail	Pass
Print Engine	7	0	0	7
Client Application	51	0	0	51
Security	2	0	0	2
Outsource Shipping	3	0	0	3
Exception Reporting	9	0	0	9
Final Report Output	4	0	0	4
Version Control	2	0	0	2

CHAPTER 9

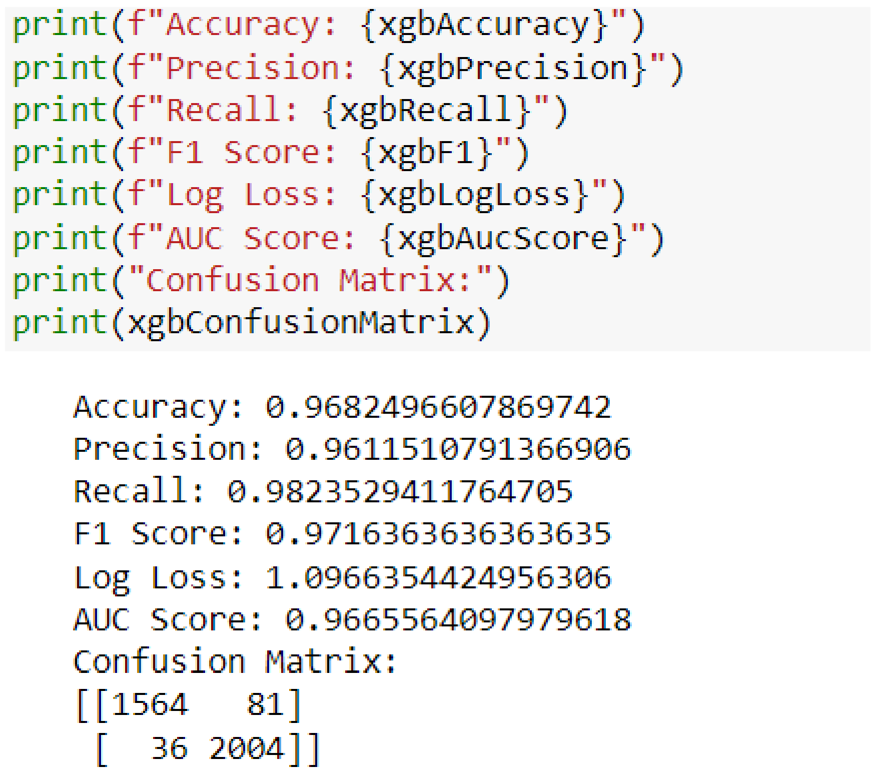
RESULTS

9.1 PERFORMANCE METRICS



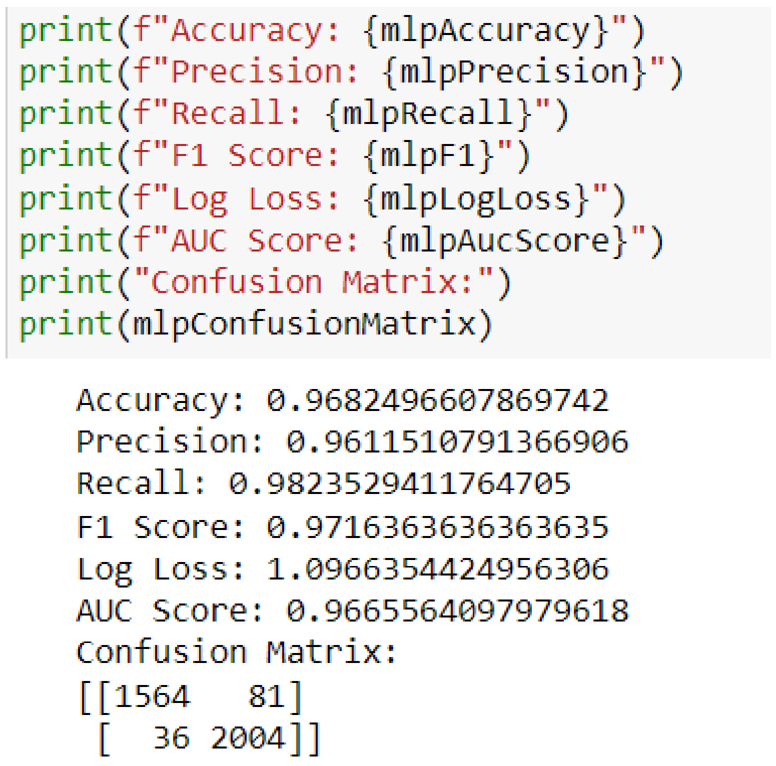
```
print(f"Accuracy: {randomForestAccuracy}")
print(f"Precision: {randomForestPrecision}")
print(f"Recall: {randomForestRecall}")
print(f"F1 Score: {randomForestF1}")
print(f"Log Loss: {randomForestLogLoss}")
print(f"AUC Score: {randomForestAucScore}")
print("Confusion Matrix:")
print(randomForestConfusionMatrix)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

```
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]]
```



```python
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]]
```



```python
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

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

Disadvantages

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

CHAPTER 11

CONCLUSION

It is impressive that a reliable anti-phishing program should be able to foresee attacks in a timely manner. We accept that in order to broaden the scope of phishing site identification, a reliable anti-phishing solution must be made promptly available. This gadget should be improved regularly via consistent retraining. Our model uses the random forest approach to address this problem. Because our model automates the organizing process and barely needs any client-defined parameters, it will assist this process if we construct a model to combat phishing and need to change it for whatever reason.

CHAPTER 12

FUTURE SCOPE

Deep learning techniques will be utilized to better accurately anticipate phishing websites and make the website more user-friendly. Additionally including this undertaking or function as an addition in search engines as a result, the project is adaptable and can always be upgraded with newer 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-26590-1660030178

PROJECT DEMO LINK:

https://drive.google.com/file/d/1Ff5_p15nKtHH8ovHkBCgH21B98y2n911/view?usp=share_link