WEB PHISHING DETECTION

NALAIYA THIRAN PROJECT BASED LEARNING ON PROFESSIONAL READINES FOR INNOVATION ,EMPLOYMENT AND ENTERPRENEURSHIP

PROJECT REPORT

Team ID = PNT2022TMTD36897

NANDHINI V (211219204010)
POOVARASI S (211219205013)
PAVITHRA M (211219205012)
PREETHI E (211219104008)

MADHA INSTITUTE OF ENGINEERING AND TECHNOLOGY

CHENNAI 600 128

Table of contents

WEB PHISHING DETECTION

1. INTRODUCTION:

A. PROJECT OVERVIEW:

In recent times, Phishing becomes an important area of concern for security researchers because it is not difficult to develop the phishing website, which looks so close to legitimate website. Experts can identify phishing websites but not all the users can identify the phishing website and such users become the victim of phishing attack. Main aim of the attacker is to steal banks account details and personal information. In United States businesses, there is a loss of US$2billion per year because their clients become victim to phishing. As per Index Report released in 2020, it was estimated that the annual worldwide impact of phishing could be as high as $1.6 million. Phishing attacks are becoming successful because lack of user awareness. Since phishing attack exploits the weaknesses found in users, it is very difficult to reduce them but it is very important to enhance phishing detection techniques. In recent times, Phishing becomes an important area of concern for security researchers because it is not difficult to develop the phishing website, which looks so close to legitimate website. Experts can identify phishing websites but not all the users can identify the phishing website and such users become the victim of phishing attack. Main aim of the attacker is to steal banks account details and personal information. In United States businesses, there is a loss of US$2billion per year because their clients become victim to phishing. As per Index Report released in 2020, it was estimated that the annual worldwide impact of phishing could be as high as $1.6 million. Phishing attacks are becoming successful because lack of user awareness. Since phishing attack exploits the weaknesses found in users, it is very difficult to reduce them but it is very important to enhance phishing detection techniques.

ABSTRACT:

This chapter discusses the various studies spanning across various researches carried out on phishing detection and related works. The chapter is organized as follows: first and foremost, a quick dive-in to the meaning of phishing in details to enlighten the reader on why phishing is an important area of research is given; second, different existing anti-phishing approaches are examined in terms of accuracy and limitations; third, a brief acknowledgment of existing techniques and how these techniques serve as a baseline to our research is presented. Furthermore, their advantages as well as the setbacks experienced in the implementation of these techniques are discussed. Fourth, we discuss the close technicalities of our work as implemented by other researchers in the same domain. This also attributed to the basic knowledge behind the choice of algorithms and approaches used. In addition, the main data preprocessing method used is also introduced in this chapter. In concluding this chapter, a tabulated summary of the most relevant and recent work that served as enlightenment to our cause on the study is also included.

B.PURPOSE:

It is extension of listing technique. In this technique, features of the websites are extracted such as URL's, content and they are used for comparison among different sites. If they match then those new websites are considered as phishing sites. These are better than listing techniques and their results gives more accuracy but their response time is low A different approach for detection of zero-hour phishing attack is discussed.

2. LITERATURE SURVEY :

Literature review Introduction: In recent times, Phishing becomes an important area of concern for security researchers because it is not difficult to develop the phishing website, which looks so close to legitimate website. Experts can identify phishing websites but not all the users can identify the phishing website and such users become the victim of phishing attack. Main aim of the attacker is to steal banks account details and personal information. In United States businesses, there is a loss of US$2billion per year because their clients become victim to phishing. As per Index Report released in 2020, it was estimated that the annual worldwide impact of phishing could be as high as $1.6 million. Phishing attacks are becoming successful because lack of user awareness. Since phishing attack exploits the weaknesses found in users, it is very difficult to reduce them but it is very important to enhance phishing detection techniques. In recent times, Phishing becomes an important area of concern for security researchers because it is not difficult to develop the phishing website, which looks so close to legitimate website. Experts can identify phishing websites but not all the users can identify the phishing website and such users become the victim of phishing attack. Main aim of the attacker is to steal banks account details and personal information. In United States businesses, there is a loss of US$2billion per year because their clients become victim to phishing. As per Index Report released in 2020, it was estimated that the annual worldwide impact of phishing could be as high as $1.6 million. Phishing attacks are becoming successful because lack of user awareness. Since phishing attack exploits the weaknesses found.

in users, it is very difficult to reduce them but it is very important to enhance phishing detection techniques. Literature review: phishing detection and protection scheme (1): Developing with the anti-phishing methods, phishers use various phishing methods and more complex and hard-to-detect approaches. The most

straightforward way for a phisher to swindle people is to make the phishing web page similar to their target. However, many distinctive and features can distinguish the original legitimate website from the clone phishing website like the spelling error, image alteration, long URL address and abnormal DNS records. The full list is revealed in Table 3 which is used later in our analysis and classification study. If an attacker clones a legitimate website as a whole or designed to look similar as they usually do in most attacks in recent times,our approach is that similar looking phishing web page con-tent is not left for the users to check for the indicator or the authenticity attentively, but can detect by automated methods. Our approach is based on website phishing detection using the features of the site, content and their appearance. These properties are stored in a local database (Excel table) as a knowledge model and first compared with the newly loaded site at the time of loading against the dangerous web page offline. After the comparison was unable to detect the similarity, then the critical approach to compare the legitimate and fake using the features of the website with machine learning for an intelligent decision. The critical contribution of our approach includes Result:The output is determined by the classifier, in the phishing detection stage which predicts if the web page is suspicious, legitimate or phishing. The knowledge model and plug-in development will be developed at a later stage.

System detection related work (2): Nowadays most people uses internet for various purposes such as online shopping like purchasing or selling products, chat with friends, sending mail. Internet users now spend more time on social networking sites Information can spread very fast and easily within the social media networks. Social media systems depend onusers for content contribution and sharing. Facebook had over 1.3 billion active users as of June 2014. there are over 1.3 billion (the number is keep growing) pages from various categories, such as company, product/service, musician/band, local business, politician, government, actor/director, artist, athlete, author, book, health, beauty, movie, cars, clothing,

community. Fans not only can see information submitted by the page, but also can post comments, photos and videos to the page. Result: Domain anomaly features are used to identify possible malicious domains based on lexical and reputation factors, whereas social anomaly features represent anomalous user behaviors in social communications.

Learning to Detect Phishing Emails (3): An alternative for detecting these attacks is a relevant process of reliability of machine on a trait intended for the reflection of the besieged deception of user by means of electronic communication. This approach can be used in the detection of phishing websites, or the text messages sent through emails that are used for trapping the victims. Approximately, 800 phishing mails and 7,000 non-phishing mails are traced till date and are detected accurately over 95% of them along with the categorization on the basis of 0.09% of the genuine emails. Result: We can just wrap up with the methods for identifying the deception, along with the progressing nature of attacks.

Phishing websites machine learning (4): Phishing URL is a widely used and common technique for cybersecurity attacks. Phishing is a cybercrime that tries to trick the targeted users into exposing their private and sensitive information to the attacker. The motive of the attacker is to gain access to personal information such as usernames, login credentials, passwords, financial account details, social networking data, and personal addresses. These private credentials are then often used for malicious activities such as identity theft, notoriety, financial gain, reputation damage, and many more illegal activities. This paper aims to provide a comprehensive and comparative study of various existing free service systems and research-based systems used for phishing website detection. The systems in this survey range from different detection techniques and tools used by many researchers. The approach included in these researched papers ranges from Blacklist and Heuristic features to visual and content-based features. The studies

presented here use advanced machine learning and deep learning algorithms to achieve better precision and higher accuracy while categorizing websites as phishing or benign. This article would provide a better understanding of the current trends and existing systems in the phishing detection domain. Result: Phishing URL detection plays a pivotal role for many cybersecurity software and applications. In this paper, we researched and reviewed works based on the advanced machine learning techniques and approaches that promise a fresh approach in this domain.

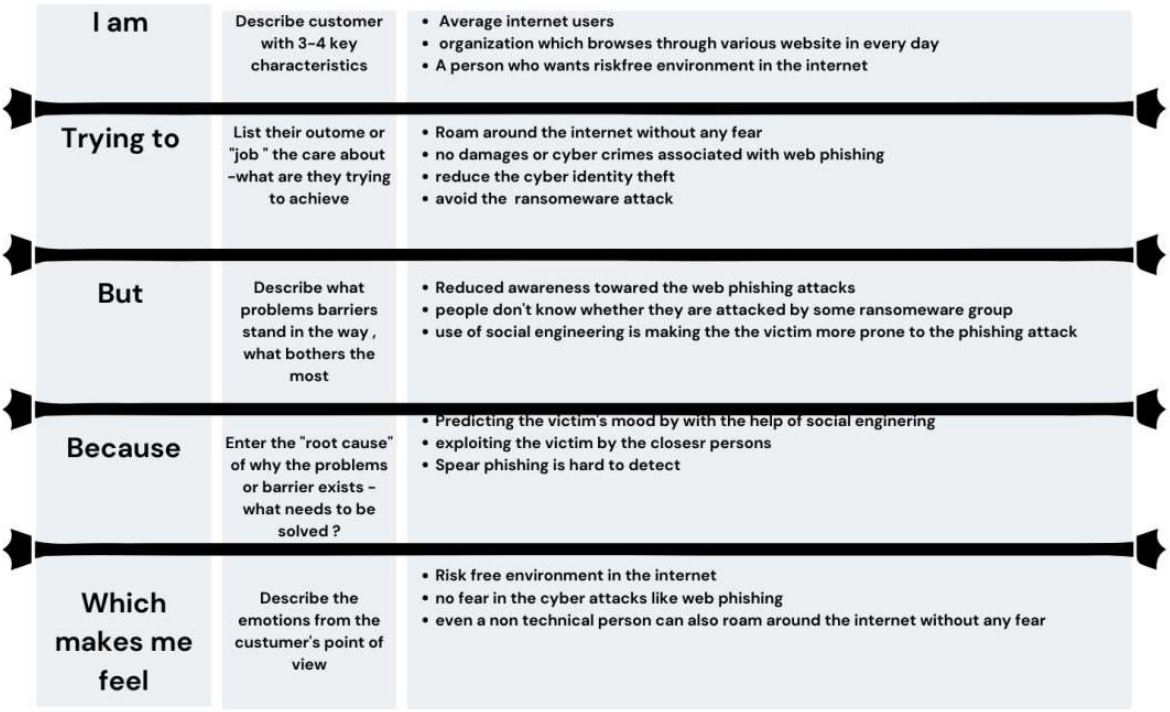
A.EXISTING PROBLEM:

To detect phishing sites, the existing system employs Classifiers, Fusion Algorithms, and Bayesian Models.learning based methods extract features from the third party, search engine, etc. Therefore, they are complicated, slow in nature, and not fit for the real-time environment. To solve this problem, this paper presents a machine learning based novel anti-phishing approach that extracts the features from client side only. Below architecture diagram as shown in Fig. 1. represents mainly flow of training phase to Detection phase. First data need to be pre-processed and feature extraction using different feature sets and later we need to train this dataset with the corresponding algorithms and the output is displayed. Result: In future we can use a combination of any other two or more classifier to get maximum accuracy. We can also explore various phishing techniques that uses Lexical features.Text and visual material can be classified by the classifiers. Text classifiers are used to categorise text material, whereas Image classifiers are used to categorise image content. The threshold value is calculated using a Bayesian model. The Fusion Algorithm uses the results of both classifiers to determine whether or not the site is phishing. Correct classification ratio, F-score, Matthews' correlation coefficient, False negative ratio, and False alarm ratio are used to evaluate the performance of different classifiers.

B.REFERENCES:

TITLE	PROBLEM IDENTIFIED	METHODOLOGY	STRENGTH	WEAKNESS
Phish Shield: A desktop application to detect phishing webpages through heuristic approach (Rao & ali,2015)	To detect URL and website content of phishing pages	Heuristic approach	Ability to detect zero hour phishing attacks & increased speed in detecting speed in detecting phishing attack	High computational cost,inability to immediately update the whitelist & blacklist
Mitigating cyber identity fraud advanced multi anti phishing technique(Yusuf et al,2013)	To tackle loopholes in electronic payment system security challenges in online banking transaction	Semantic content analysis,Earth mover Distance(EMD) & biometric authentication with finger print	Detecting of phishing webpages & preventing unauthorized online banking transfer & withdrawal	It achieved 80% true negative
Efficient prediction of phishing website using supervised learning algorithms,Santhana Laksmi.v & vijaya MS,2011	Phishers are using new techniques to break all antiphishing mechanisms	Supervised learning algm,ie,.multi layer perceptron ,decision tree induction & machine learning techniques to model the prediction task & naive Bayes classification to explore result	It can predict whether a given website is legitimate or phishing website	Time taken to build the model & predictions accuracy is high in the case of decision tree induction
Anti phishing based on automated individual whitelist (AIWL)YE	Blacklist is not completely effective in detecting phishing URL because of	Naive bayesian classifier	It keeps a whitelist of users all familiar login user interface(LUIs)of website .it guides	It requires gathering the website IP & this is time consulting as IP needs to be

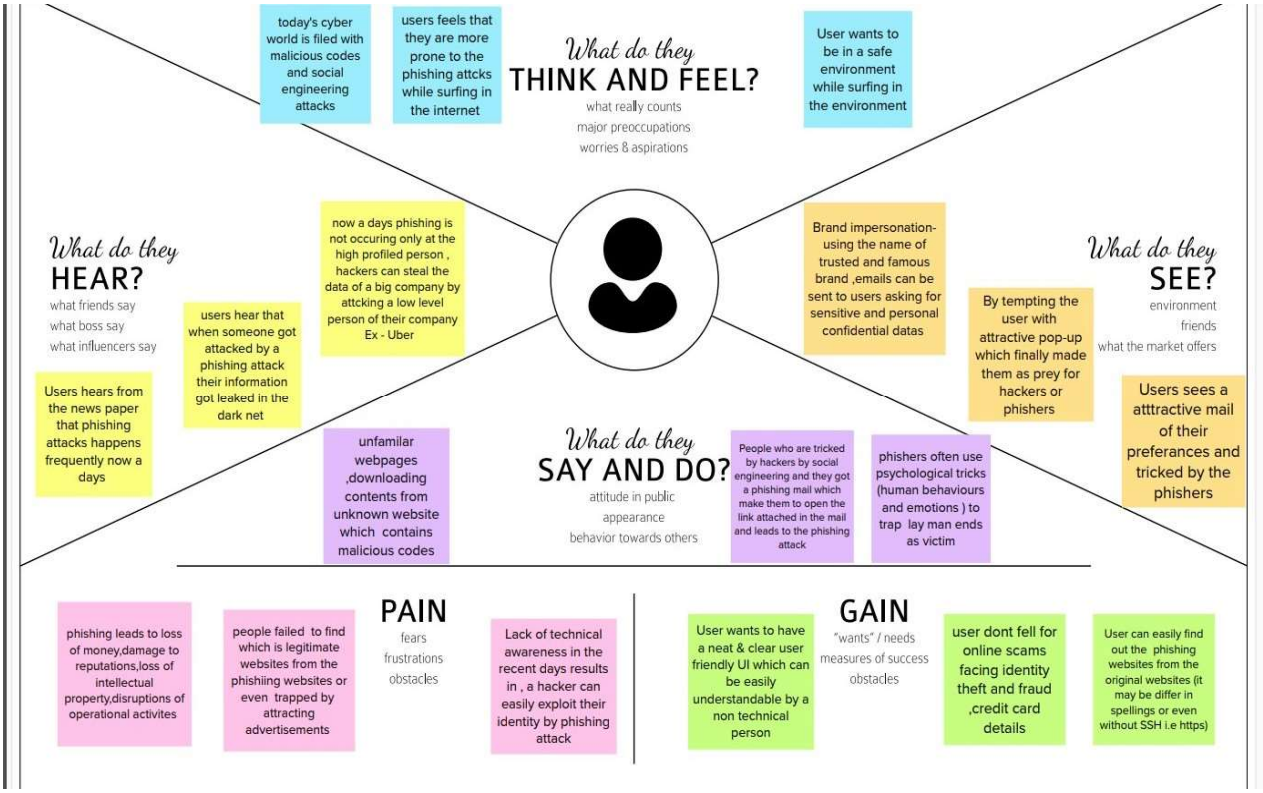
C.PROBLEM STATEMENT DEFINITION:



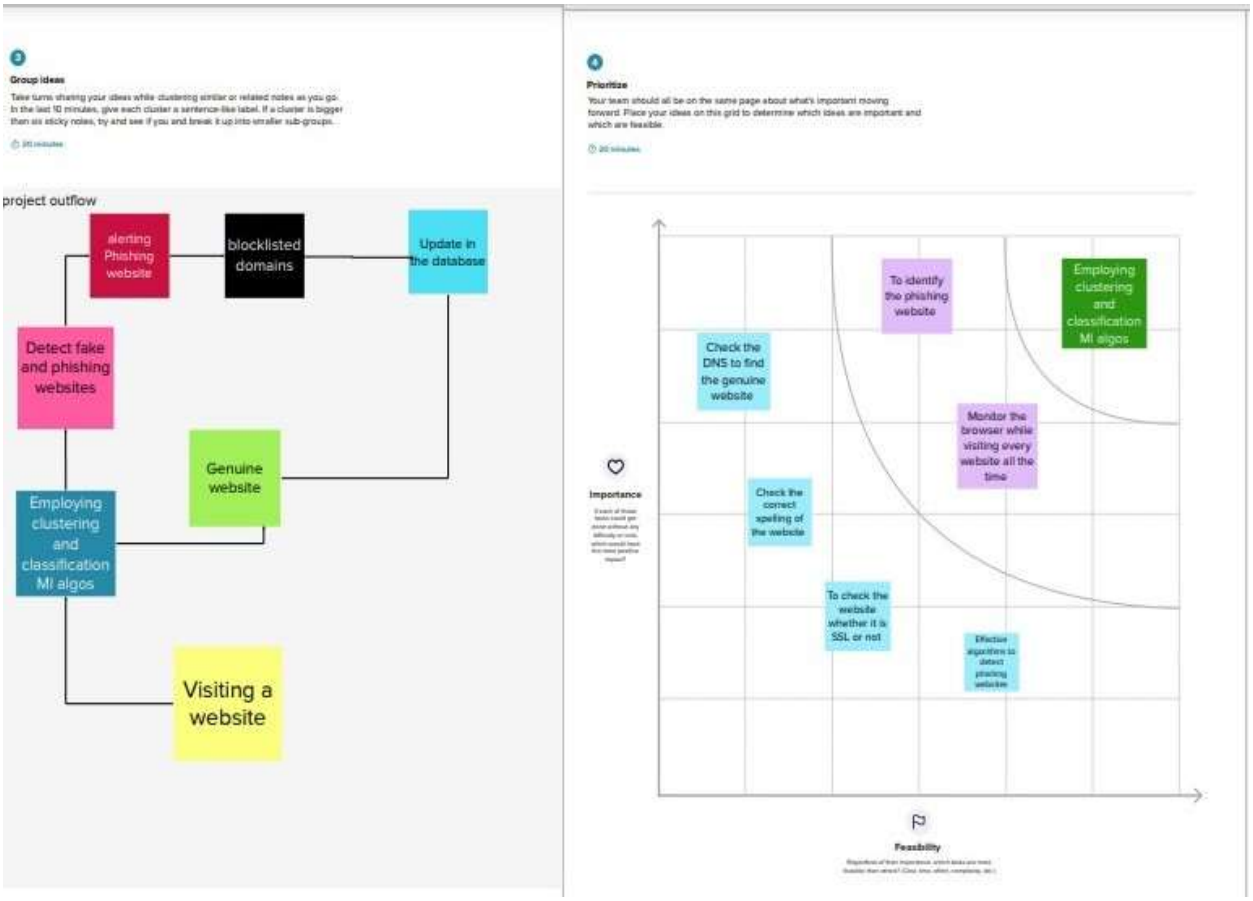
Problem Statement (PS)	I am (Customer)	I'm trying to	But	Because	Which makes me feel
PS-1	An average internet user	Browse for a website in the internet	Can't find the correct website	There are several fake websites among them	Feels like taking a risk in visiting on a website
PS-2	An organizational user	Uses browser for the organization everyday	He can't tell every mail he receives ia from a legimidate source	Risk of being attacked by a phishing attack and results in some ransomeware	Fear of became a victime for a phishing attack and results in ransomeware

3. IDEATION & PROPOSED SOLUTION

A. EMPATHY MAP CANVAS



B.IDEATION & BRAINSTORMING



C. PROPOSED SOLUTION:

S.No	**Parameter**	**Description**
1	Problem statement	• Web phishing is one of the major problems which handles with sensitive information. • Malicious link will often steals users credentials without their consent which must be solved • The main objective is to identify phishing e-payment website and safeguard user information from phishing to protect users privacy
2	Idea / Solution description	• Our proposed model is capable of detecting phishing websites by use of Machine learning & classification algo's. • As we came to know while doing literature survey ,efficient models like decision

		tree,random forest could also be tried out. ● use of pre-defined blacklisted website dataset
3	Novelty / Uniqueness	● As we are providing a service(SAAS) which doesn't need any kind of computational resources. ● The specialized feature is that we provide users to enable our project as a Chrome extension with user-friendly UI/UX which gives them a higher level of confidence while doing transactions or web surfing . ● Our model is designed in such a way which gives alerts while entering into phishing websites.
		● Our model is designed in such a way which gives alerts while entering into phishing websites.
4	Social impact	● Our project will have a definite impact on society by making users free from data theft . ● secure users from proxies and scams ● Using our product people can feel safer and secure from the cyber-attack like web phishing
5	Business Model	● We are providing the product as Software as a service model, so that users can easily use the

		• Even though our product is not designed to generate revenue directly , it helps indirectly for e-commerce & banking institutions by providing trustiness over the products which further increase the intakes of services .
6	Scalability of the solution	• Apart from E-banking and e-commerce sector the idea proposed can be developed into platform independent model also. • Machine Learning models and effective feature engineering techniques helps identify phishing websites and come up with key features that are common in most phishing websites.

D.PROBLEM SOLUTION FIT:

Project Title: Web phishing detection

Project Design Phase-I - Solution Fit Template

Team ID: PNT2022TMID42214

Define CS, fit into CC

1. CUSTOMER SEGMENT(S) CS

Who is your customer?
i.e. working parents of 0-5 y.o. Kids

- C-suite executives
- Internet based financial services business
- Online payment service users

6. CUSTOMER CONSTRAINTS 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.

- Prevent access to third party websites
- multi step verification
- prevent entry to unwanted websites
- Frequent change of passcodes

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 note taking

- Web security gateway
- Secure web gateway
- Spam filter
- use of VPN
- Check for site seals
- Firewalls and proxy
- Antispyware software

Explore AS, differentiate

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

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.

- Prevent personal data getting stolen
- Ensure user safety
- Intimating the suspicious activity or log in attempts

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?

- Large user base
- Leniency in the adaption of security measure
- Lack of awareness where layman pretends every websites looking legitimate

7. BEHAVIOUR BE

What does your customer do to address the problem and get the job done?

- Using instant firewalls
- Back up files
- Scan System for malware
- Change credentials
- Set up a fraud alert

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

3. TRIGGERS TR

What triggers customers to act?

- Coupons and gift voucher's
- Attractive advertisement and pop –ups
- Assuming everything is legitimate website

4. EMOTIONS: BEFORE / AFTER EM

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

Before the job is done:
Threatened, scared, anxious, stressed, lost

After the job is done :
Satisfied , relieved ,relaxed ,happy

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.

- Pop –up alert for fake websites
- Check websites authenticity
- Whitelist filtering
- Blacklist interception

8. CHANNELS of BEHAVIOUR CH

8.1 ONLINE

What kind of actions do customers take online? Extract online channels from

- Don't use insecure public channels while doing transactions
- Back up files
- Scan system for malware
- Set up a fraud alert

8.2 OFFLINE

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

- Change credentials
- Make Complaint in respective offices

2. REQUIREMENT ANALYSIS:

4.A.FUNCTIONAL REQUIREMENTS:

FR No.	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
FR-1	User Registration	Registration through Gmail ,Apple Registration through Facebook , Instagram
FR-2	User Confirmation	Confirmation via Email Confirmation via OTP
FR-3	Website identification	model identifies the legitimacy website based on trained data sets
FR-4	Classifier	By use of machine learning algorithms such as KNN,regression ,classification predicts the expected result
FR-5	Results	Model predicts the webpage and pop-out to the user before entering any confidential information

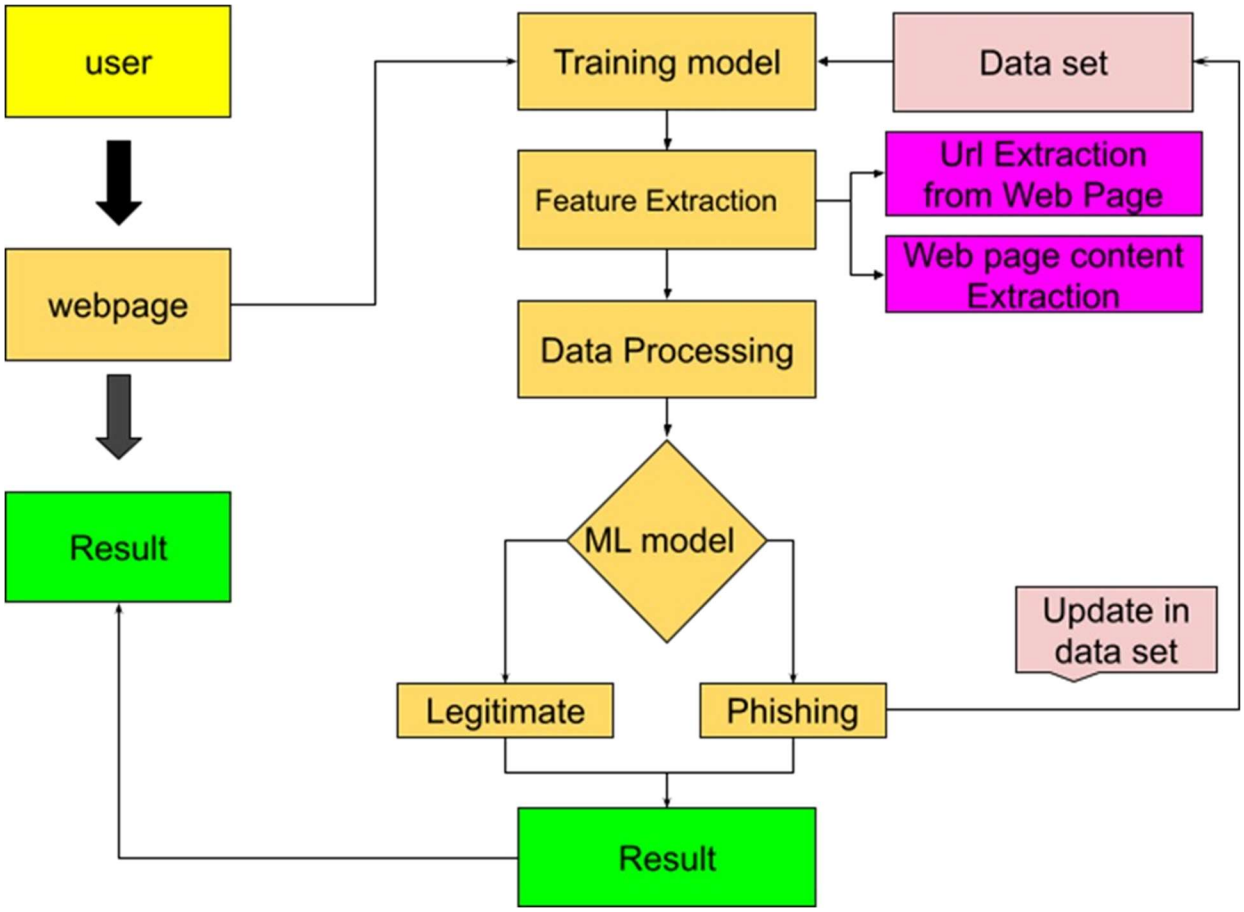
B.NON-FUNCTIONAL REQUIREMENTS:

FR NO.	Non-Functional Requirement	Description
NFR-1	Usability	with special feature web extension,it increase the effectiveness,efficiency,hassle-free navigation,overall satisfaction of the user throughout the system
NFR-2	Security	Multi factor authentication,authorization,authetication of result
NFR-3	Reliability	Probability of failure-free operations in a specified environment for a specified time
NFR-4	Performance	The performance should be faster and user friendly for the effective performance
NFR-5	Availability	The user should get availability to access the resourcemust be valid and reliable
NFR-6	Scalability	System must be highly scalable in order to handling more users without any disturbance to service

3.PROJECT DESIGN

A.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.



B.Solution & Technical Architecture

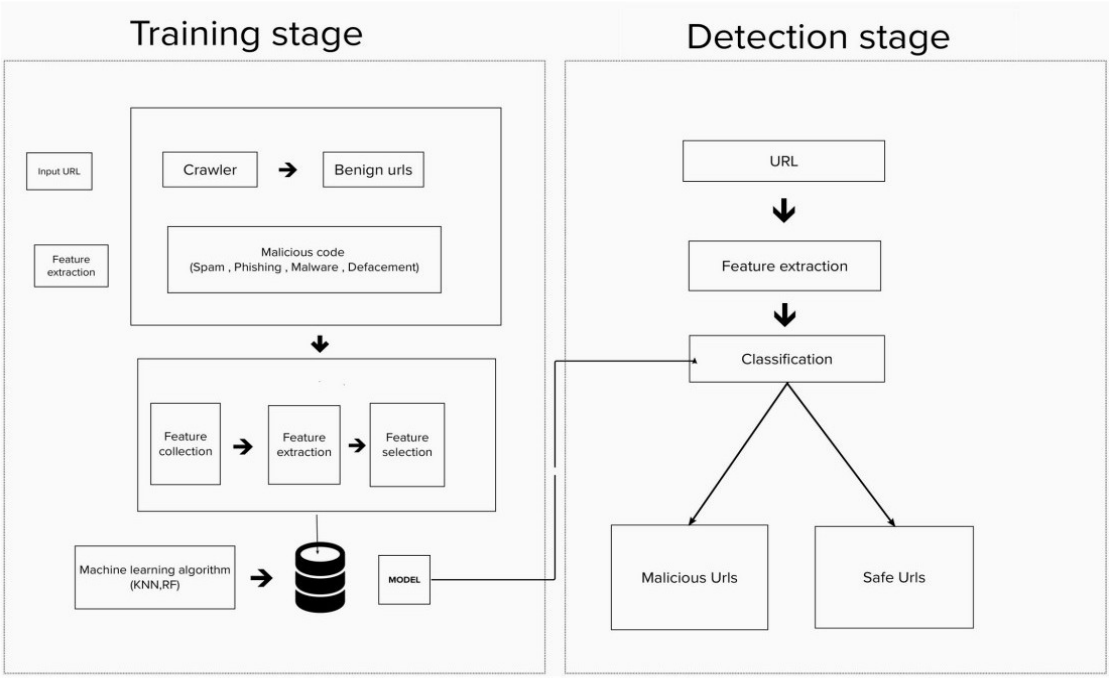


Table-1 : Components & Technologies:

S.No	Component	Description	Technology
1.	User Interface	How user interacts with application e.g. Web UI, Mobile App, Chatbot etc.	HTML, CSS, JavaScript / Angular Js / etc.
2.	Application Logic-1	Logic for a process in the application	Python
3.	Application Logic-2	Logic for a process in the application	IBM Watson STT service
4.	Application Logic-3	Logic for a process in the application	IBM Watson Assistant
5.	Database	Data Type, Configurations etc.	MySQL,
6.	Cloud Database	Database Service on Cloud	IBM DB2, IBM Cloudant etc.
7.	File Storage	File storage requirements	IBM Block Storage
8.	External API-1	Purpose of External API used in the application	Fast API
9.	External API-2	Purpose of External API used in the application	nil
10.	Machine Learning Model	Purpose of Machine Learning Model	Object Recognition Model, DBN model , Random forest classifier
11.	Infrastructure (Server / Cloud)	Application Deployment on a machine to monitor and detection of web phishing	Local, Cloud Foundry, Kubernetes, IBM watson

Table-2: Application Characteristics:

S.No	Characteristics	Description	Technology
1.	Open-Source Frameworks	open-source frameworks used in the project	Gophish , python flask
2. ,	Security Implementations	security / access controls implemented, use of firewalls etc.	RSA , SHA-256, Encryptions, proxy firewalls , OWASP etc.
3.	Scalable Architecture	Cloud infrastructure which can be used to provide services for more number of custumers at any time	IBM Watson cloud

C.User Stories

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

6. PROJECT PLANNING & SCHEDULING

A. SPRINT PLANNING&ESTIMINATION:

IDEATION PHASE

TITLE	DESCRIPTION	DATE
Literature survey	Literature survey on the selected project & gathering information by referring the, technical papers, research publications etc	**Aug 29, 2022 - Sep 3, 2022**
Empathy Map	Prepare Empathy Map Canvas to capture the user Pains & Gains, Prepare list of problem statements	**Sep 5, 2022 - Sep 10, 2022**
Problem Statement	Prepare the problem statement document	**Sep 5, 2022 - Sep 10, 2022**
Brainstorming Idea Generation Prioritization	List them by organizing the brainstorming session and prioritize the top 3 ideas based on the feasibility & importance.	**Sep 12, 2022 - Sep 17, 2022**

PROJECT DESIGN PHASE-I

TITLE	DESCRIPTION	DATE
Problem Solution Fit	Prepare problem - solution fit document.	**Sep 26, 2022 - Oct 1, 2022**
Proposed Solution	Prepare the proposed solution document, which includes the novelty, feasibility of idea, business model, social impact, scalability of solution, etc.	**Sep 19, 2022 - Sep 24, 2022**
Solution Architecture	Prepare a solution architecture document.	**Sep 19, 2022 - Sep 24, 2022**

PROJECT DESIGN PHASE- II

TITLE	DESCRIPTION	DATE
Functional Requirements	Prepare the functional requirement document.	**Oct 10, 2022 - Oct 15, 2022**
Customer Journey Map	Prepare the customer journey maps to understand the user interactions & experiences with the application (entry to exit).	**Oct 3, 2022 - Oct 8, 2022**
Data Flow Diagrams and User Stories	Draw the data flow diagrams and submit for review.	**Oct 10, 2022 - Oct 15, 2022**

Technology Stack	Prepare the technology architecture diagram	**Oct 10, 2022 - Oct 15, 2022**

PROJECT PLANNING PHASE

TITLE	DESCRIPTION	DATE
Project Planning	Prepare the planning for this project	**Oct 17, 2022 - Oct 22, 2022**
Milestone and Activity List	Prepare the milestones & activity list of the project	**Oct 17, 2022 - Oct 22, 2022**

PROJECT DEVELOPMENT

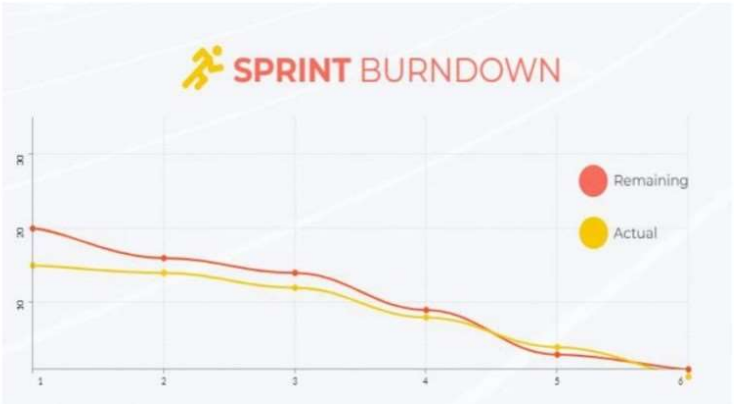
TITLE	DESCRIPTION	DATE
Project Development Delivery of Sprint-1, 2, 3 & 4	Develop & submit the developed code by testing it	**IN PROGRESS..**

Velocity:
Imagine we have a 10-day sprint duration, and the velocity of the team is 20 (points per sprint). Let's calculate the team's average velocity (AV) per iteration unit (story points per day)

$$AV = \frac{sprint\ duration}{velocity} = \frac{20}{10} = 2$$

Burndown Chart:

A burn down chart is a graphical representation of work left to do versus time. It is often used in agile software development methodologies such as Scrum. However, burn down charts can be applied to any project containing measurable progress over time.



7. CODING & SOLUTIONING

FEATURE 1:

The most critical component of defense against phishing is accurate detection of phishing websites in a timely manner. Successful recognition and blacklisting of phishing URLs would result in end users receiving a warning while being deceived to visit the phishing site. Once a striking warning such as the one presented is displayed, it is highly likely that users would decline the login/data-input requests or malicious payload-downloading popups in phishing sites.

FEATURE 2:

Hence, in this we propose an effective detection system that crawls websites and automatically discovers malicious pages. We intend our system to be used by a blacklist provider who can automatically compile and maintain an up-to-date blacklist of malicious URLs. Our system is equipped with a plentiful set of features that reflect various types of essential characteristics of the webpage content or behavior, which are impossible or difficult to be camouflaged by the miscreants. This system can proactively crawl and evaluate a given URL, labeling it to be phishing/malicious or legitimate, based on a trained classifier. Further, crawling is done from distributed vantage points, which allows the system to collect novel features and achieve higher accuracy and quicker recognition speed. By avoiding manual analysis, the blacklist can achieve better coverage and timeliness.

Source code

App.py

```
#importing required libraries

from flask import Flask, request, render_template
import numpy as np
import pandas as pd
from sklearn import metrics
import warnings
import pickle
```

```
warnings.filterwarnings('ignore')
from feature import FeatureExtraction

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


app = Flask(_name_)

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

        url = request.form["url"]
        obj = FeatureExtraction(url)
        x = np.array(obj.getFeaturesList()).reshape(1,30)

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

if__name__== "_main_":
    app.run(debug=True,port=2002)
```

Feature.py

```
import ipaddress
import re
```

```
import urllib.request
from bs4 import BeautifulSoup
import socket
import requests
from googlesearch import search
import whois
from datetime import date, datetime
import time
from dateutil.parser import parse as date_parse
from urllib.parse import urlparse

class FeatureExtraction:
    features = []
    def __init_(self,url):
        self.features = []
        self.url = url
        self.domain = ""
        self.whois_response = ""
        self.urlparse = ""
        self.response = ""
        self.soup = ""

        try:
            self.response = requests.get(url)
            self.soup = BeautifulSoup(response.text, 'html.parser')
        except:
            pass

        try:
            self.urlparse = urlparse(url)
            self.domain = self.urlparse.netloc
        except:
            pass

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

```
except:
    pass



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

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

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

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


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

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

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

```
        return -1
    return 1

# 4.Symbol@
def symbol(self):
    if  re.findall("@",self.url):
        return -1
    return 1
# 5.Redirecting//
def redirecting(self):
    if self.url.rfind('//')>6:
        return -1
    return 1
# 6.prefixSuffix
def prefixSuffix(self):
    try:
        match = re.findall('\-', self.domain)
        if match:
            return -1
        return 1
    except:
        return -1
# 7.SubDomains
def  SubDomains(self):
    dot_count = len(re.findall("\.", self.url))
    if dot_count == 1:
        return 1
    elif dot_count == 2:
        return 0
    return -1

# 8.HTTPS
def Hppts(self):
    try:
        https = self.urlparse.scheme
        if 'https' in https:
```

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

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

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

# 10. Favicon
def Favicon(self):
    try:
        for head in self.soup.find_all('head'):
            for head.link in self.soup.find_all('link', href=True):
                dots = [x.start(0) for x in re.finditer('\.', head.link['href'])]
```

```
                    if self.url in head.link['href'] or len(dots) == 1 or domain in
head.link['href']:
                return 1
        return -1
    except:
        return -1

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

# 12. HTTPSDomainURL
def HTTPSDomainURL(self):
    try:
        if 'https' in self.domain:
            return -1
        return 1
    except:
        return -1
# 13. RequestURL
def RequestURL(self):
    try:
        for img in self.soup.find_all('img', src=True):
            dots = [x.start(0) for x in re.finditer('\.', img['src'])]
            if self.url in img['src'] or self.domain in img['src'] or len(dots) == 1:
                success = success + 1
            i = i+1

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

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

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

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

        try:
            percentage = success/float(i) * 100
            if percentage < 22.0:
                return 1
            elif((percentage >= 22.0) and (percentage < 61.0)):
                return 0
            else:
                return -1
        except:
            return 0
    except:
        return -1
# 14. AnchorURL
def AnchorURL(self):
    try:
        i,unsafe = 0,0
        for a in self.soup.find_all('a', href=True):
                if "#" in a['href'] or "javascript" in a['href'].lower() or "mailto" in
a['href'].lower() or not (url in a['href'] or self.domain in a['href']):
            unsafe = unsafe + 1
```

```
            i = i + 1

        try:
            percentage = unsafe / float(i) * 100
            if percentage < 31.0:
                return 1
            elif ((percentage >= 31.0) and (percentage < 67.0)):
                return 0
            else:
                return -1
        except:
            return -1

    except:
        return -1

# 15. LinksInScriptTags
def LinksInScriptTags(self):
    try:
        i,success = 0,0
        for link in self.soup.find_all('link', href=True):
            dots = [x.start(0) for x in re.finditer('\.', link['href'])]
            if self.url in link['href'] or self.domain in link['href'] or len(dots) == 1:
                success = success + 1
            i = i+1

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

        try:
            percentage = success / float(i) * 100
            if percentage < 17.0:
                return 1
```

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

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

# 17. InfoEmail
def InfoEmail(self):
    try:
        if re.findall(r"[mail\(\)|mailto:?]", self.soap):
            return -1
        else:
            return 1
    except:
        return -1

# 18. AbnormalURL
```

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

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

# 20. StatusBarCust
def StatusBarCust(self):
    try:
        if re.findall("<script>.+onmouseover.+</script>", self.response.text):
            return 1
        else:
            return -1
    except:
        return -1

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

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

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

# 23. IframeRedirection
def IframeRedirection(self):
    try:
        if re.findall(r"[<iframe>|<frameBorder>]", self.response.text):
            return 1
        else:
            return -1
    except:
        return -1

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

        today  = date.today()
```

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

    # 25. DNSRecording
    def DNSRecording(self):
        try:
            creation_date = self.whois_response.creation_date
            try:
                if(len(creation_date)):
                    creation_date = creation_date[0]
            except:
                pass

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

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

```
            return -1

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

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

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

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

```
            return -1
        except:
            return -1

    # 30. StatsReport
    def StatsReport(self):
        try:
            url_match = re.search(
          'at\.ua|usa\.cc|baltazarpresentes\.com\.br|pe\.hu|esy\.es|hol\.es|sweddy\.com|
myjino\.ru|96\.lt|ow\.ly', url)
            ip_address = socket.gethostbyname(self.domain)
                            ip_match = re.search('146\.112\.61\.108|213\.174\.157\.151|
121\.50\.168\.88|192\.185\.217\.116|78\.46\.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 url_match:
                return -1
            elif ip_match:
                return -1
            return 1
        except:
            return 1
```

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

Integration.py

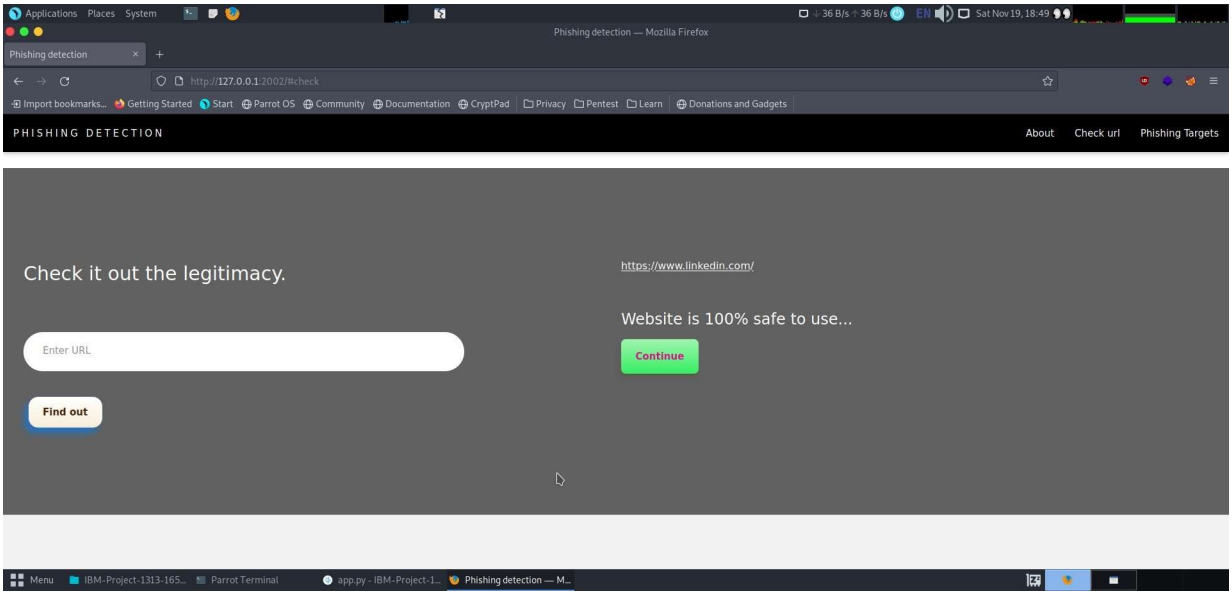
```
        #importing required libraries

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

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

        # NOTE: you must manually set API_KEY below using information retrieved
        from your IBM Cloud account.
        API_KEY = "H_eQnWI4923bdKM63m1wH9G07eo-uRpUYs9b7FqNPCVF"
        token_response = requests.post('https://iam.cloud.ibm.com/identity/token',
        data={"apikey":
         API_KEY, "grant_type": 'urn:ibm:params:oauth:grant-type:apikey'})
        mltoken = token_response.json()["access_token"]

        header = {'Content-Type': 'application/json', 'Authorization': 'Bearer ' +
        mltoken}


        app = Flask(__name__)

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

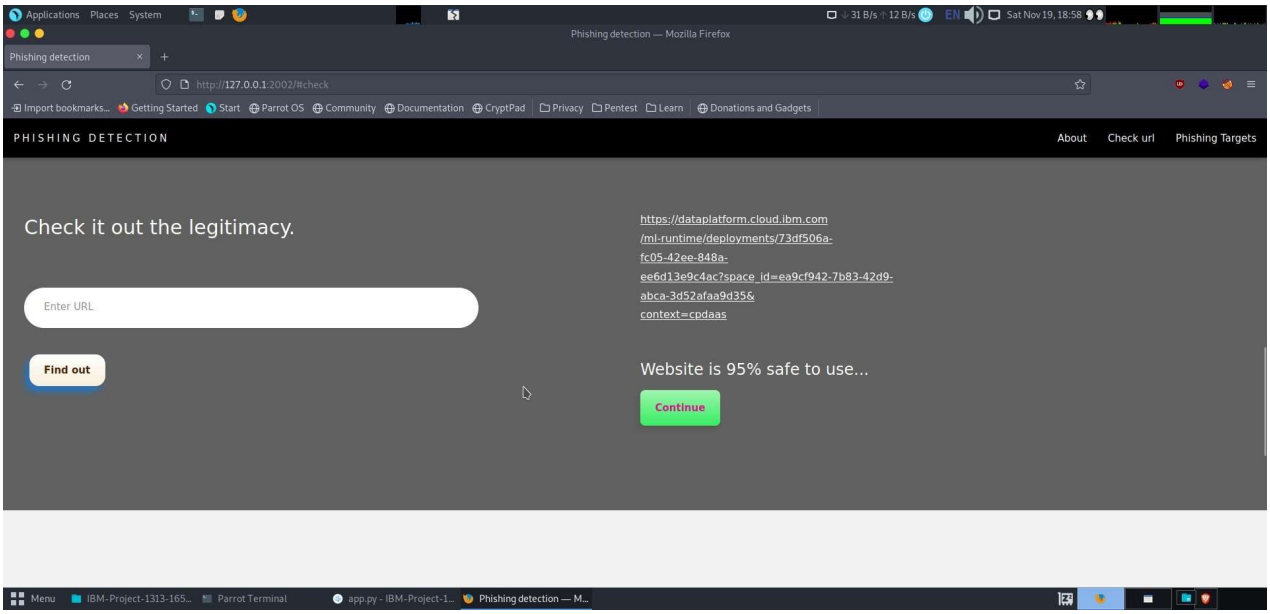
```
    if request.method == "POST":

        url = request.form["url"]
        obj = FeatureExtraction(url)
        x = np.array(obj.getFeaturesList()).reshape(1,30)

        y_pred =gbc.predict(x)[0]
        #1 is safe
        #-1 is unsafe
        y_pro_phishing = gbc.predict_proba(x)[0,0]
        y_pro_non_phishing = gbc.predict_proba(x)[0,1]
        # if(y_pred ==1 ):
        pred = "It is {0:.2f} % safe to go ".format(y_pro_phishing*100)
                                payload_scoring = {"input_data": [{"field":
[["UsingIP","LongURL","ShortURL","Symbol@","Redirecting//","PrefixSuffix-","
SubDomains","HTTPS","DomainRegLen","Favicon","NonStdPort","HTTPSDoma
inURL","RequestURL","AnchorURL","LinksInScriptTags","ServerFormHandler","
InfoEmail","AbnormalURL","WebsiteForwarding","StatusBarCust","DisableRigh
tClick","UsingPopupWindow","IframeRedirection","AgeofDomain","DNSRecord
ing","WebsiteTraffic","PageRank","GoogleIndex","LinksPointingToPage","Stats
Report"
]], "values": [[1,1,1,1,1,-1,-1,-1,-1,1,1,1,1,-1,-1,1,1,1,0,1,1,1,1,-1,-1,-1,-
1,1,0,1]]}]}
                                                        response_scoring =
requests.post('https://us-south.ml.cloud.ibm.com/ml/v4/deployments/
73df506a-fc05-42ee-848a-ee6d13e9c4ac/predictions?version=2022-11-17',
json=payload_scoring,
        headers={'Authorization': 'Bearer ' + mltoken})
        print("Scoring response")
        predictions=response_scoring.json()
#print(predictions)
        pred=print(predictions['predictions'][0]['values'][0][0])
                                    return render_template('index.html',xx
=round(y_pro_non_phishing,2),url=url )
    return render_template("index.html", xx =-1)
```

```
if__name__== "_main_":
    app.run(debug=True,port=2020)
```

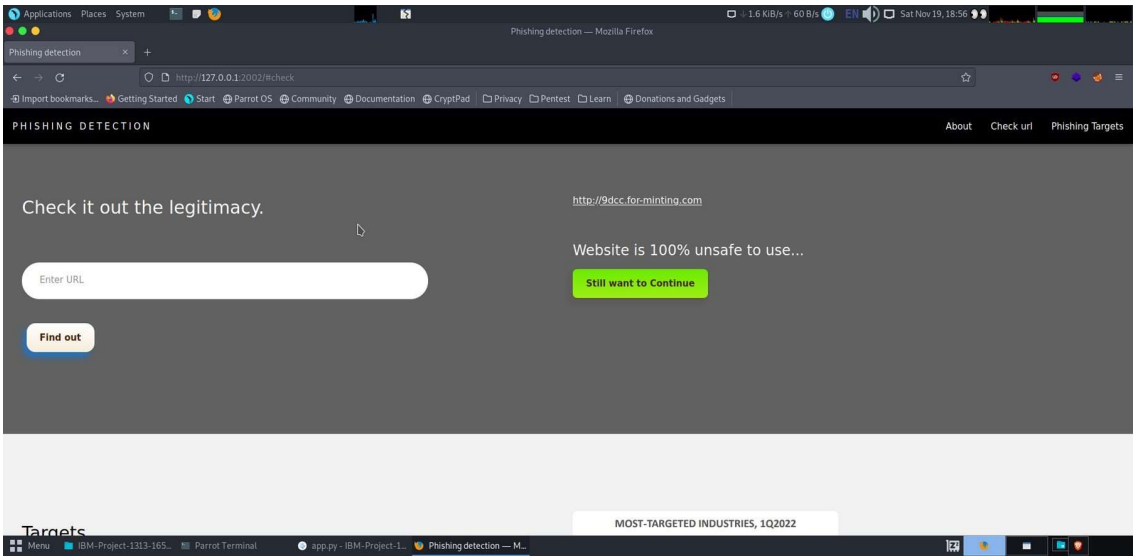
8. TESTING

A. Test case 1:url detection for legitimate website

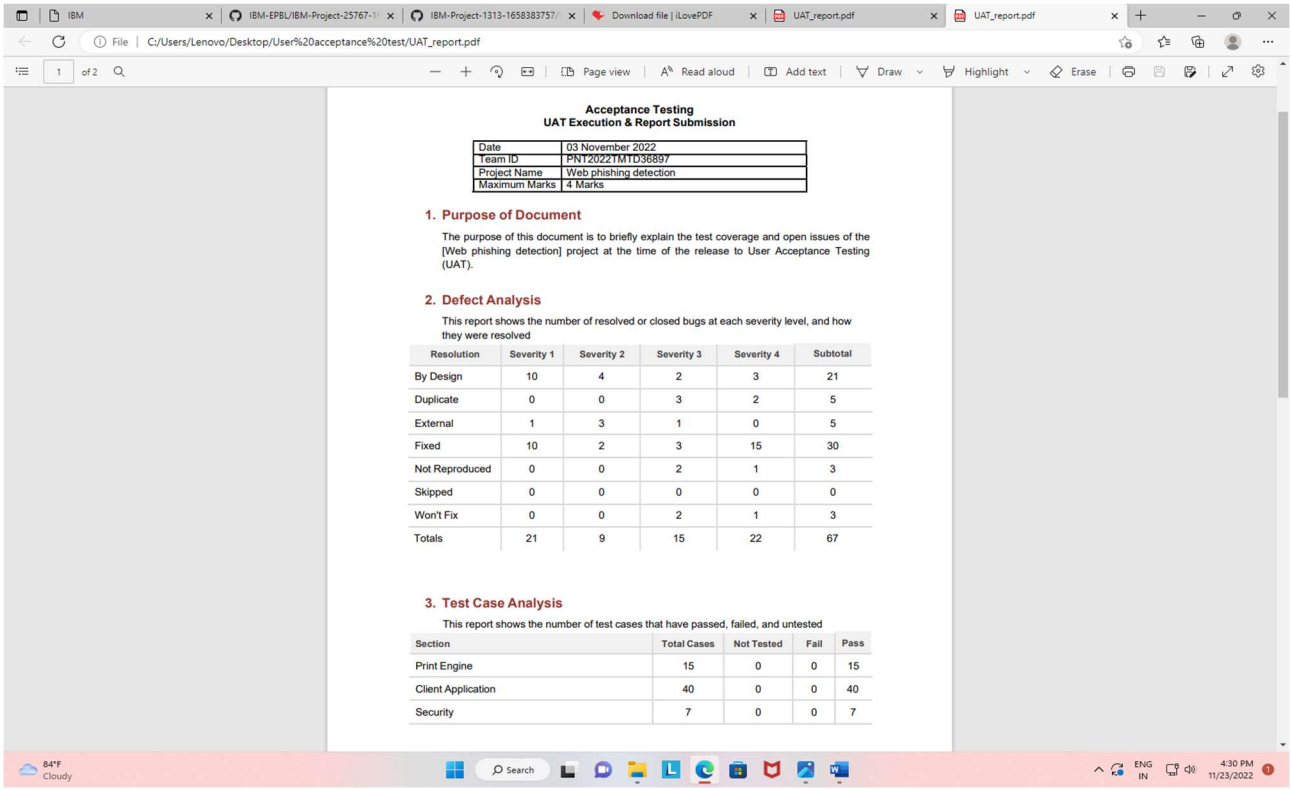




Test case 2: detecting phishing website



B. User Acceptance Testing:



Acceptance Testing
UAT Execution & Report Submission

Date	03 November 2022
Team ID	PNT2022TMTD36897
Project Name	Web phishing detection
Maximum Marks	4 Marks

1. Purpose of Document

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

2. Defect Analysis

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

Resolution	Severity 1	Severity 2	Severity 3	Severity 4	Subtotal
By Design	10	4	2	3	21
Duplicate	0	0	3	2	5
External	1	3	1	0	5
Fixed	10	2	3	15	30
Not Reproduced	0	0	2	1	3
Skipped	0	0	0	0	0
Won't Fix	0	0	2	1	3
Totals	21	9	15	22	67

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	15	0	0	15
Client Application	40	0	0	40
Security	7	0	0	7

3. Test Case Analysis

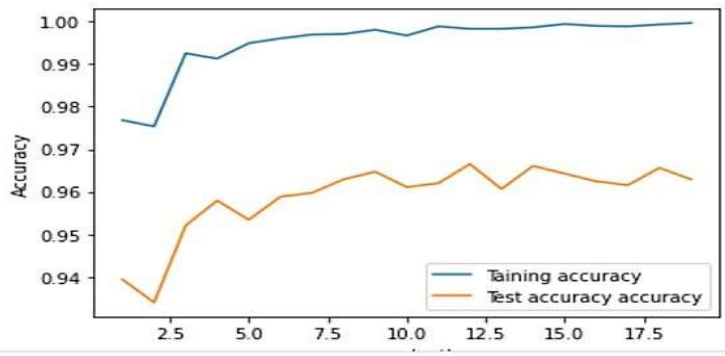
This report shows the number of test cases that have passed, failed, and untested

Section	Total Cases	Not Tested	Fail	Pass
Print Engine	15	0	0	15
Client Application	40	0	0	40
Security	7	0	0	7
Outsource Shipping	3	0	0	3
Exception Reporting	12	0	0	12
Final Report Output	12	0	0	12
Version Control	5	0	0	5

9. RESULTS

A.PERFORMANCE:

```
training_accuracy=[]
test_accuracy=[]
depth=range(1,20)
for n in depth:
  rf_test=RandomForestClassifier(n_estimators=n)
  rf_test.fit(x_train,y_train)
  training_accuracy.append(rf_test.score(x_train,y_train))
  test_accuracy.append(rf_test.score(x_test,y_test))
plt.figure(figsize=None)
plt.plot(depth,training_accuracy,label="Taining accuracy")
plt.plot(depth,test_accuracy,label="Test accuracy accuracy")
plt.ylabel("Accuracy")
plt.xlabel("max_depth")
plt.legend();
```



✓ 1s completed at 1:37 PM

B.METRICS:

Classification report

[ ]

	ML Model	Accuracy	f1_score	Recall	Precision
0	Logistic Regression	91.814	92.567	94.496	94.496
1	Random Forest	96.653	96.942	100.000	100.000
2	XgbClassifier	94.754	95.207	96.714	96.714
3	Decision tree	95.206	95.605	100.000	100.000

[ ]
```
sorted_result=result.sort_values(by=['Accuracy', 'f1_score'],ascending=False).reset_index(drop=True)
sorted_result
```

	ML Model	Accuracy	f1_score	Recall	Precision
0	Random Forest	96.653	96.942	100.000	100.000
1	Decision tree	95.206	95.605	100.000	100.000
2	XgbClassifier	94.754	95.207	96.714	96.714
3	Logistic Regression	91.814	92.567	94.496	94.496

10. ADVANTAGES :

Bayesian content filtering can be trained on a per user basis.It avoids the false positives.

DISADVANTAGES:

Scammers use “Bayesian poisoning” technique to circumvent Bayesian content filtering.phishers sometimes bypass the filter’s database by transforming the words. For example, they may replace “Viagra” with “Viaagra.”

11. CONCLUSION:

Phishing URL detection plays a pivotal role for many cybersecurity software and applications. In this , we reviewed works based on the advanced machine learning techniques and approaches that promise a fresh approach in this domain. This includes summary of the reviewed works after a systematic and comprehensive study on Phishing Website Detection systems. We believe that the presented survey would help researchers and developers with the insight of the progress achieved in the past years. Despite the tremendous progress in the field of cybersecurity, phishing website detection still pose a challenging problem with the ever evolving technology and techniques.

12. FUTURE SCOPE:

The success of these techniques highly depends on the quality and combination of relevant characteristic features of web pages which may include network traffic information, content characteristics, lexical features of URLs, and even domain name system (DNS) information. However, extracting these attributes may be costly and sometimes require downloading complete web pages [9] or looking up various DNS servers and ISPs to get enrichment data like geo-location, registration records, and network information , which face the problem of network latency, making them impractical for real-time systems . Due to the difficulties of using non-lexical features to detect malicious URLs in real-time despite their high accuracy values, previous works have explored the use of URL lexical features only and it has

been proven that URL features alone can produce an accurate means of detecting malicious webpage in realtime systems.

13. APPENDIX

Source Code:https://github.com/IBM-EPBL/IBM-Project-25767-1659972882.git