# WEB PHISHING DETECTION
# APPLIED DATA SCIENCE

**A PROJECT REPORT**

***Submitted by***

**ESTHER J**

**GAYATHRI PRIYADHARSHINI K**

**MERLIN C**

**SAKTHI ESWARI K**

**UMA S**

***in partial fulfillment for the award of the degree of***

**BACHELOR OF ENGINEERING**

***in***

**ELECTRONICS AND COMMUNICATION ENGINEERING,**

**UNIVERSITY VOC COLLEGE OF ENGINEERING,**

**THOOTHUKUDI**

**ANNA UNIVERSITY:: CHENNAI 600025**

**NOV 2022**

# ANNA UNIVERSITY : CHENNAI 600 025

## BONAFIDE CERTIFICATE

Certified that this project report **"WEB PHISHING DETECTION"** is the bonafide work of "**ESTHER J (953219106009), GAYATHRI PRIYADHARSHINI K (953219106010), MERLIN C (953219106019), SAKTHI ESWARI K (953219106028), UMA S (953219106041)"** who carried out the project work under my supervision.

**SIGNATURE**

Dr .K.ESSAKI MUTHU

**HEAD OF THE DEPARTMENT**

Electronics and Communication Engineering

University VOC College Of Engineering, Thoothukudi.

**SIGNATURE**

DR.A.MOOKAMBIKA

**MENTOR**

Assistant Professor

Electronics and Communication Engineering

University VOC College Of Engineering, Thoothukudi.

# CONTENTS

# 1. Introduction:

Phishing is a form of fraud in which the attacker tries to learn sensitive information such as login credentials or account information by sending as a reputable entity or person in email or other communication channels. Phishing attacks can paralyze a business. Staff might be unable to continue their work. Data and assets might be stolen or damaged. Customers might be unable to access online services. The reason security defenders struggle to detect phishing domains is because of the unique part of the website domain.

## 1.1 Project Overview:

**Category:** Applied Data Science

**Team ID** : PNT2022TMID50791

**Skills Required:**

IBM Cloud, IBM Watson Studio, Data Science, Machine learning, HTML,CSS, Javascript, IBM Cloud Object Storage, Python- Flask

**Project Description:**

Phishing is a form of fraudulent attack where the attacker tries to gain sensitive information by posing as a reputable source. In a typical phishing attack, a victimopens a compromised link that poses as a credible website. The victim is then asked to enter their credentials, but since it is a "fake" website, the sensitive information is routed to the hacker and the victim gets "hacked."

Phishing is popular since it is a low effort, high reward attack. Most modern web browsers, antivirus software and email clients are pretty good at detecting phishing websites at the source, helping to prevent attacks. To understand how they work, this project shows you how to build your own phishing URL detector using Pythonand Applied data science:

1.Identify the criteria that can recognize fake URLs

2.Build a decision tree that can iterate through the criteria 3.Train our model to recognize fake vs real URLs 4,Evaluate our model to see how it performs

5.Check for false positives/negatives

**Social Impact:**

• It will help to minimize the frauds while using software solutions(EX: Web applications, etc).

**Business Model/Impact:**

This application can be used by many E-commerce enterprises in order to make the whole transaction process.

## 1.2Purpose:

There are a number of users who purchase products online and make payments through e-banking. There are e-banking websites that ask users to provide sensitive data such as username, password & credit card details, etc., often for malicious reasons. This type of e-banking website is known as a phishing website.

Web service is one of the key communications software services for the Internet. Web phishing is one of many security threats to web services on the Internet.

Common threats of web phishing:

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

## 2.LITERATURE SURVEY:

### 1. WEB ADDRESS BASED EVALUATION

### 1.1. LIST BASED DETECTION TECHNIQUES

A database of URL called list is maintained. It generally holds URLs, internet protocol (IP) addresses, and keywords. Some researchers maintain a whitelist, which is a collection of legitimate URLs. Most of the researchers suggest maintaining a blacklist, which is acollection of malicious URLs. List-based detection method acts as a filtering mechanism to sweep away suspicious web pages before entering into the detection process

| SI. NO | TITLE OF PAPER | YEAR OF PUBLICATION | AUTHOR NAME | DESCRIPTION |
|---|---|---|---|---|
| 1 | Anti-phishing based on automated individual white-list | 2008 | Cao Y. Han W. Le Y. | He proposed an automated individualwhitelist (AIWL)-basedapproach that maintains alocal list of user's familiarlogin user interface (LUI)of websites to alert theuser whenever he tries to access an unfamiliarwebsite with LUI. AIWL uses a naïve Bayesianclassifier to maintain the list by adding the unknown website.However, This approach cannot stand up against the local machine trojan horse and viruses. |
| 2 | A novel approach to protect against phishing attacks at client side using auto-updated white-list | 2016 | Jain A.K. Gupta B.B | It combined the whitelist approach with heuristicsand ML to propose the auto-updated whitelist. Blacklists and whitelists are used as a filteringmodule in many web phishing detection approaches to reduce the processing time wasted on pre-processing, feature extraction, and so on. |

## 1.1. HEURISTICRULE BASED DETECTION TECHNIQUE

Heuristic rule-based techniques can identify the zero-day attacks.Therefore, it has a high-detection rate than list-based phishing detection schemes. The performance and accuracy of the technique wholly depend on the heuristics applied

| SI. NO | TITLE OF PAPER | YEAR OF PUBLICATION | AUTHOR NAME | DESCRIPTION |
|---|---|---|---|---|
| 1 | Machine learning based phishing detection from URLs | 2019 | Sahingoz O.K. Buber E. Demir O. et al | Applies heuristics to extract natural language processing (NLP) features from the URL to detect the URL-based web phishing attacks. The heuristics are derived based on parameters such as raw word count, short word length, Alexa ranking, similar brand |

| | | | | |
|---|---|---|---|---|
| | | | | name count, etc. |
| 2 | 'A stacking model using URL and HTML features for phishing webpage detection | 2019 | Li Y. Yang Z. Chen X. et al | Applies some heuristics on the URL to verify abnormalities such as suspicious symbols (e.g. @, _), https, URL length information, number of dots in a domain name, sensitive vocabulary, and top-level domain. |
| 3. | Intelligent phishing URL detection using association rule mining | 2016 | Jeeva S.C. Rajsingh E.B. | Computes 14heuristics: length of the host URL, number of slashes, dots in the host name, number of terms in the host name, special characters, IP address, unicode in URL, transport layer security, subdomain, certain keyword, top-level domain, number of dots in the path of the URL, hyphen in the host name and URL length. The extracted features are then fed into associative rule miningalgorithms. |

| | | | | |
|---|---|---|---|---|
| 4 | A phish detector using light weight search features | 2016 | Varshney G.Misra M..Atrey P.K | Proposed a lightweight phish detector, which extracts the domain name of the URL and title of the webpagewhenever a user accessing a website. The extracted URL domain name and the title page aresearched using a search engine to determine the legitimacy |

## 1.2. LEARNING BASED DETECTION TECHNIQUE

Learning algorithms such as ML and deep learning are used to detect the attacks based on the features extracted from the URL. In learning-based web phishing detection, the statistical features and NLP features of the URLs are extracted and fed into ML algorithms such as support vector machine (SVM), decision tree, naïve Bayes algorithm, random forest etc. The classifier creates a model based on the inference extracted from the Training samples. The suspicious URL is evaluated based on the model built by the classifier.

| SI. NO | TITLE OF PAPER | YEAR OF PUBLICATION | AUTHOR NAME | DESCRIPTION |
|---|---|---|---|---|
| 1 | Machine learning based phishing detection from URLs | 2019 | Sahingoz O.K. Buber E. Demir O. et al. | Practices seven different ML algorithms such as naive Bayes, random forest ,k-nearest Neighbour(KNN),Adaboost,kstar, ,sequential minimal optimization, and decision tree on the extractedfeatures from the URLand analysed the best performance amongthem. |

| | | | | |
|---|---|---|---|---|
| 2 | 'A stacking model using URL and HTML features for phishing webpage detection | 2019 | Li Y. Yang Z. Chen X. et al.: | Proposed a deep learningapproach to extract thefeatures naturally from the URLs and to detectthe web phishing attack. Convolutional neural network(CNN) is used to extract the correlation features and long short term memory (LSTM) network is used to learn sequential dependency. |
| 3 | Phishing website detection based on multidimensional features driven by deep learning | 2019 | Yang P. Zhao G. Zeng P: '. | Proposed aweb phishing detection approach usinga neural network. In thiswork, feature validity value(FVV) is introduced to examine the effect ofoptimal features. By using the FVV index, the optimal feature selection algorithm is designed to choose the optimalfeatures and is used to mitigate the over fitting problem of neural networks. |

ML algorithms can detect zero-day attacks and have a shorter detection time. However this technique is feature sensitive and the performance varies based on the characteristics of the ML algorithm applied

## 2. WEBPAGE CONTENT\ SIMILARITY BASED EVALUATION

### 2.1. HEURISTIC RULE BASED WEBPAGE SIMILARITYEVALUATION

In heuristic-based webpage similarity calculation, keywords and features are extracted from the suspicious webpage and verified against the targeted webpage using search methods to enable a secured environment against phishing scams.

| SI. NO | TITLE OF PAPER | YEAR OF PUBLICATION | AUTHOR NAME | DESCRIPTION |
|---|---|---|---|---|
| 1 | PhishWHO: phishing webpage detection via identity keywords extraction and target domain name finder | 2016 | Tan C.L. Chiew K.L. Wong K. | Proposes a phishing webpage detection approach four modules-identity keywords extraction, search engine lookup, target domain name finder, and three-tier identity matching.The target domain name and actual domain name are passed as inputs to the three-tier identity matching system to analyse the status of the query webpage. |
| 2 | Phishing-alarm: robust and efficient phishing detection via page component similarity | 2017 | Mao J. Tian W. Li P. et al.: | Proposed a phishing alarm by extracting the CSS features from the underlying architecture of the web page. Page similarity calculations are applied to the extracted features to classify the web pages |

| | | | | |
|---|---|---|---|---|
| 3 | Off-the-hook: an efficient and usable client-side phishing prevention application | 2017 | Marchal S. Armano G. Gröndahl T. et al. | Designed a client-side phishing detection tool that offers better privacy, real-time protection, effective warnings, and resilience to dynamic phish. This approach uses a phish detector and target identifier mechanisms to detect the Phishing webpages. |

## 2.2. ML-BASED WEBPAGE SIMILARITY EVALUATION

In this technique, HTML, extensible mark-up language (XML), JavaScript(JS), and CSS featuresare extracted from the source code of the webpage and are fed into ML algorithms for furtherclassification.

| SI. NO | TITLE OF PAPER | YEAR OF PUBLICATION | AUTHOR NAME | DESCRIPTION |
|---|---|---|---|---|
| 1 | Cantina+ a feature-rich machine learning framework for detecting phishing web sites | 2011 | Xiang G. Hong J. Rose C.P. et al.: ‘ | Proposed a content-basedapproach to detect webphishing by extracting URLfeatures, HTML-based features, and web-based features.The proposed approach is evaluated with twomethods that are randomised evaluationand time-basedevaluation using the Bayesian network. |

| | | | | |
|---|---|---|---|---|
| 2 | Detecting phishing websites via aggregation analysis of page layouts | 2018 | Mao J. Bian J. Tian W.et al.: | Proposed a learning-based layout similarity detection using ML algorithms. SVM and decision trees are used to classify the similarity of the webpages. |
| 3 | A new hybrid ensemble feature selection framework for machine learning-based phishing detection system | 2019 | Chiew K.L. Tan C.L. Wong K. et al. | Proposed a new feature selection framework for ML-based phishing detectionsystem. A novel cumulative distribution function gradient algorithm is designed as an automaticfeature cut-off rank identifier to produce the compact set of primary features and then dataperturbation, and function perturbation techniques are applied on these primary features to derive the hybrid ensemble features. |
| 4 | A machine learning based approach for phishing detection using hyperlinks information | 2019 | Jain A.K. Gupta B.B. | Proposed a novel web phishing detection approach by extracting hyperlinks of the web pages. The proposedapproach has extracted 12 specific hyperlink feature. The extracted features are then fed into ML algorithms such as naïve Bayes, random forest, SVM, Adaboost, neural network, C4.5, and logisticregression. The performance of all the ML algorithms was measured and reported. |

## 3.HYBRID APPROACHES

Hybrid web phishing detection techniques were proposed by combining theexisting web phishing detection schemes.

| SI. NO | TITLE OF PAPER | YEAR OF PUBLICATION | AUTHOR NAME | DESCRIPTION |
|---|---|---|---|---|
| 1 | A comprehensive and efficacious architecture for detecting Phishing webpages | 2014 | Gowtham R. Krishnamurthi I.: | Proposed a web phishingdetection approach usinga preapproved site identifier, login form finder, and ML algorithms. The websiteswhich are resulted as suspicious from the modules are furtherprocessed by the SVMML algorithm. |
| 2 | A stacking model using URL and HTML features for phishing webpage detection | 2019 | Li Y. Yang Z. Chen X. et al. | Combined URL features, HTML source code features, and HTML string embedding to detect theweb phishing scam. A stacking model of gradient boost decision tree, Xtreme Gradient Boost(XGBOOST), and LightGBM is used to improve the performance of the system. |
| 3 | Phishing website detection based on multidimensional features driven by deep learning | 2019 | Yang P. Zhao G. Zeng P | Presented a hybrid approach to attain multi-dimensional features to increase the detection rate and to reduce the detection time. URL evaluation, web page similarity approach, and contentbased approach are combined in that work. Both ML (i.e XGBOOST) and deep learning (i.e.CNN-LSTM) algorithms are applied to classify the attack. |

| | | | | |
|---|---|---|---|---|
| 4 | Two level filtering mechanism to detect phishing sites using lightweight visual similarity approach | 2019 | Rao R.S. Pais A.R. | Proposed a two level filtering mechanism to detect the webphishing attack. At thefirst level, a lightweight visual similarity-based blacklist is applied todetect near-duplicatephishing sites. At the secondlevel, heuristicfiltering is performed onthe bypassed phishing sites from the blacklists. |
| 5 | An approach for phishing validation and detection | 2017 | Li J.H. Wang S.D.: | Proposed a PhishBox approach forphis h validation and detection. This approach has a two- stage model. In the first stage, theensemble model is designed to evaluate the phish data, and active learning is applied to reduce the cost of manual labelling. In the second stage, the validated phishing data is used to train the detection model. |

## 2.1 Existing problem :

Cyber criminals use phishing emails because it's easy, cheap and effective. Email addresses are easy to obtain, and emails are virtually free to send. With little effort and cost, **attackers can quickly gain access to valuable data**.

## 2.2.References :


[1] Zou Futai, Gang Yuxiang, Pei Bei, Pan Li, Li Linsen Web Phishing DetectionBased on Graph Mining.

[2] Nick Williams, Shujun Li Simulating human detection of phishing websites: An investigation into the applicability of ACT-R cognitive behaviour architecturemodel.

[3] XIN MEI CHOO, KANG LENG CHIEW,DAYANG HANANI ABANG IBRAHIM,NADIANATRA MUSA, SAN NAH SZE, WEI KING TIONG Feature-based Phishing Detection Technique.

[4] Giovanni Armano, Samuel Marchal and N. Asokan RealTime Client-SidePhishing Prevention Add-on.

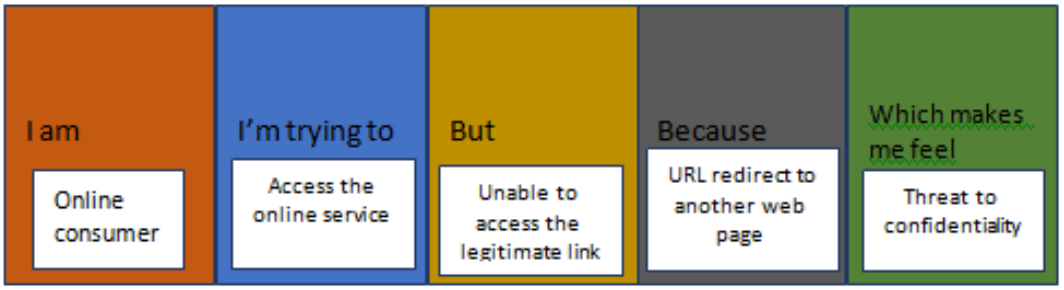
[5] Trupti A. Kumbhare and Prof. Santosh V. Chobe An Overview of AssociationRule Mining Algorithms.

[6] S.Neelamegam, Dr.E.Ramaraj Classification algorithm in Data mining: AnOverview

[7] Varsharani Ramdas Hawanna, V. Y. Kulkarni and R. A. Rane A NovelAlgorithm to Detect Phishing URL.

## 2.2 Problem Statement Definition :

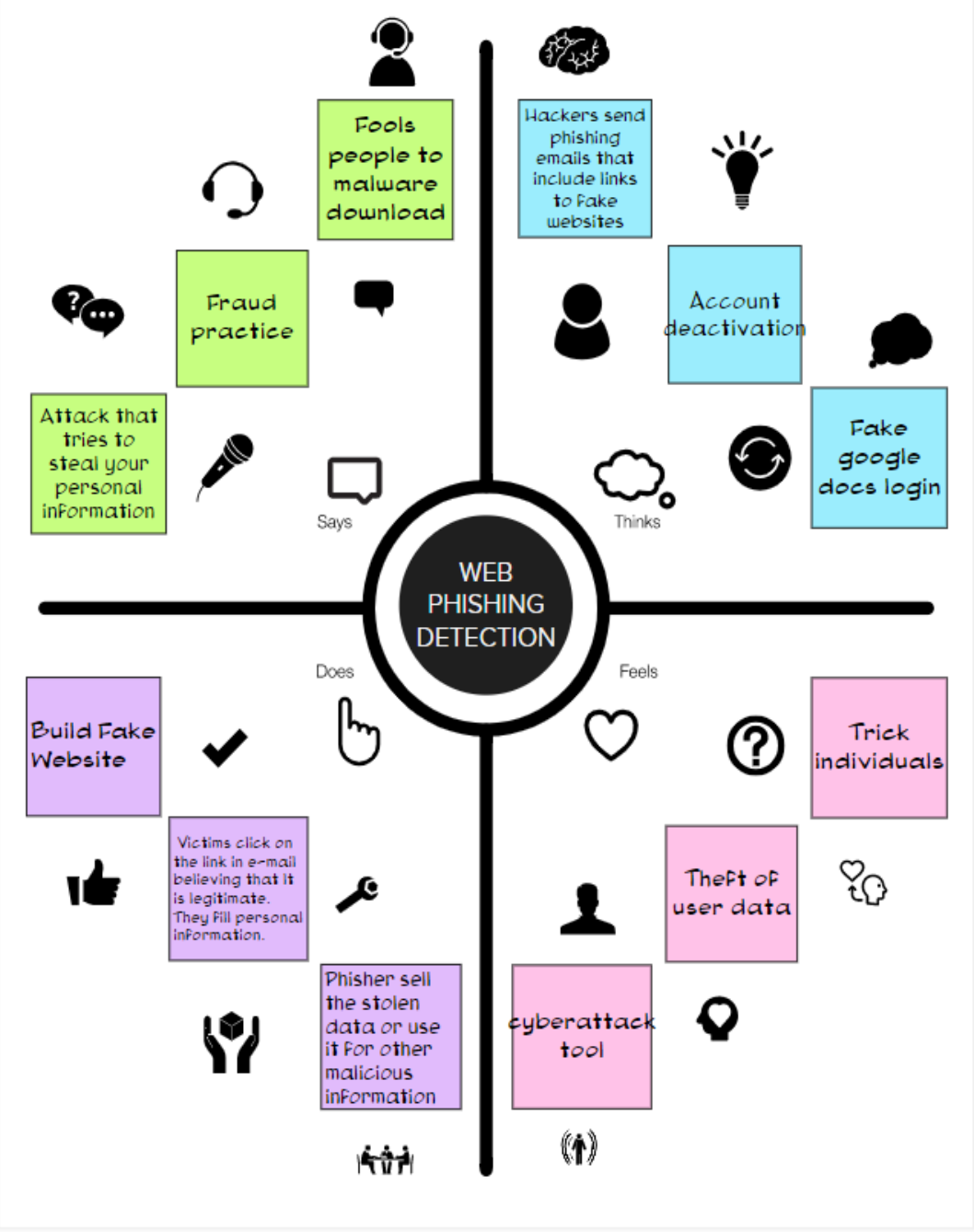
Malicious links will lead to a website that often steals login credentials or financial information like credit card numbers. Attachments from phishing emails can contain malware that once opened can leave the door open to the attacker to perform malicious behavior from the user's computer.



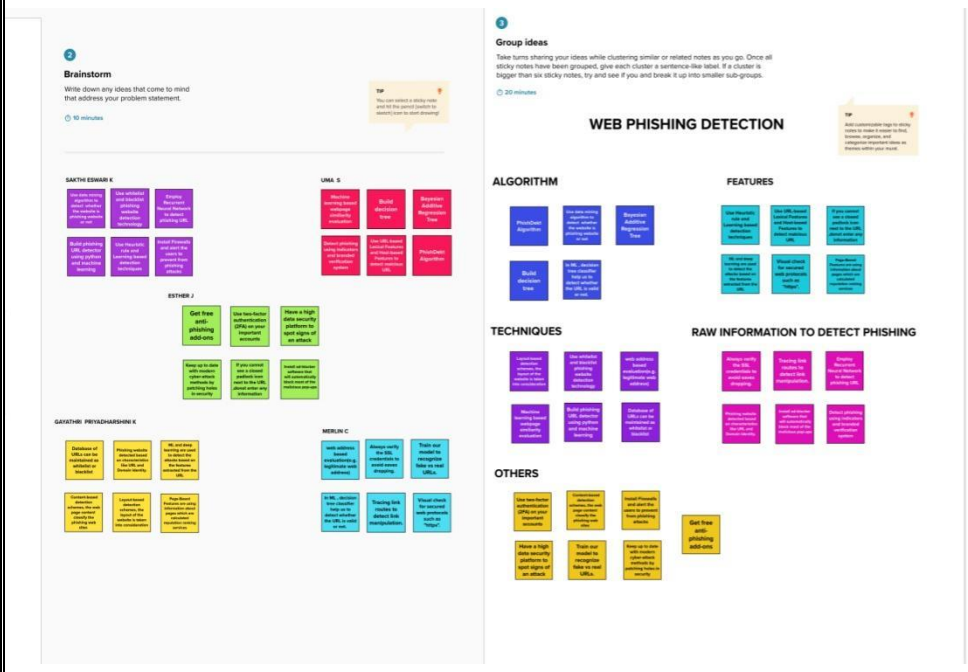

| Problem Statement (PS) | I am (Customer) | I'm trying to | But | Because | Which makes me feel |
|---|---|---|---|---|---|
| PS-1 | Online consumer | Access the online service | Unable to access the legitimate link | URL redirect to another webpage | Threat to confidentiality |
| PS-2 | Student | Apply for PAN card | Unable to access the legitimate link | URL redirect to another webpage | Insecure |

## 3. IDEATION & PROPOSED SOLUTION:

### 3.1 Empathy Map Canvas:



## 3.2 Ideation & Brainstorming:



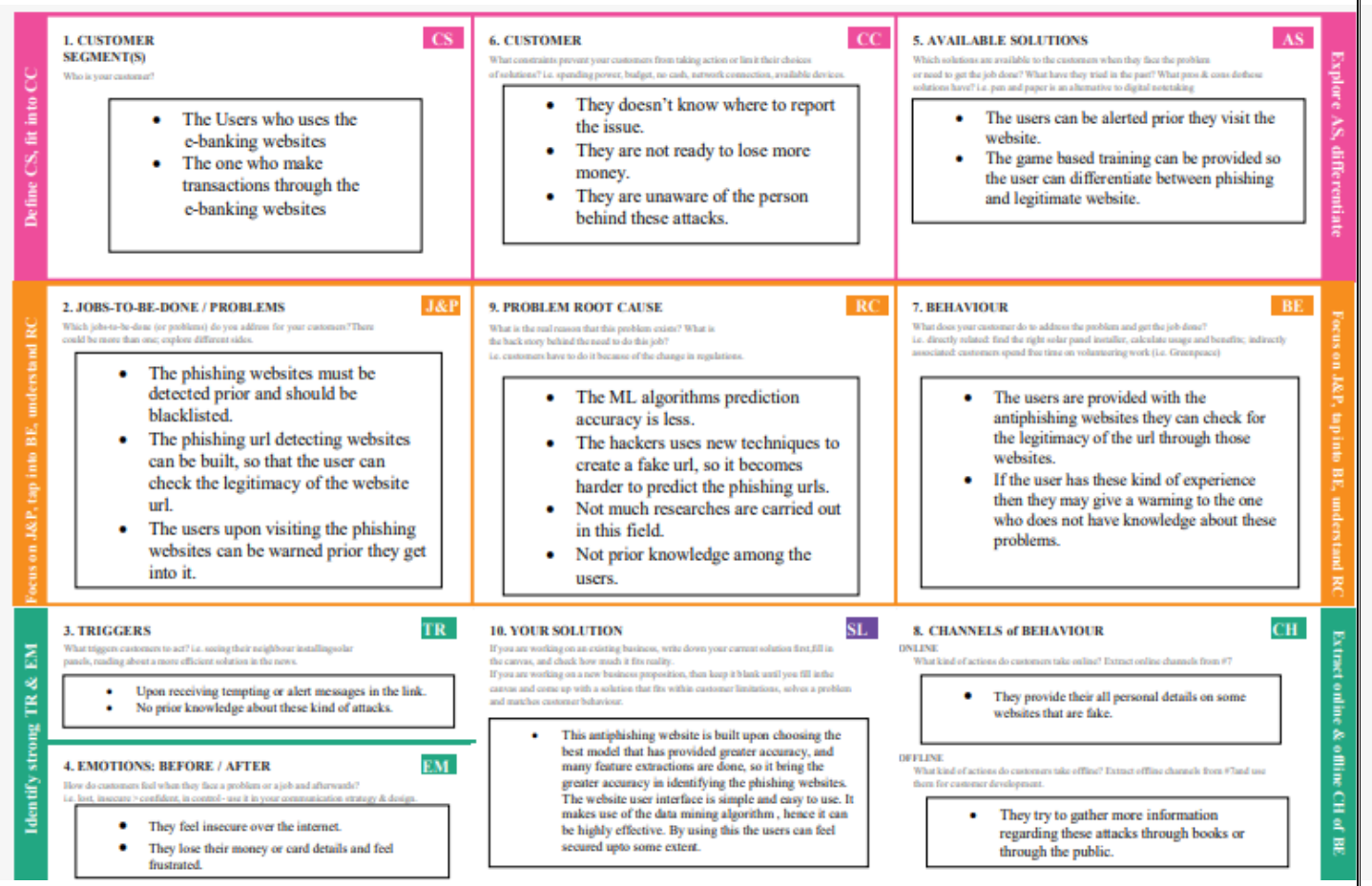

## 3.3 Proposed Solution

| S.No. | Parameter | Description |
|---|---|---|
| 1. | Problem Statement (Problem to be solved) | Attacker tries to steal your personal information and fools people to download malwares. Hackers build fake websites and send phishing emails that include links to those fake websites. They trick individuals for the theft of user data. Victims click on the link believing that it is legitimate and fill their personal information. The phisher steals the information and sells the stolen data or use it for other malicious information. |
| 2. | Idea / Solution description | Database of URLs can be maintained as whitelist or blacklist. Use data mining algorithm to detect whether the website is phishing website or not. In ML , decision tree classifer help us to detect whether the URL is valid or not. Use two-factor authentication(2FA) on your important accounts. |

| | | |
|---|---|---|
| 3. | Novelty / Uniqueness | A novel approach to protect against phishing attacks at client side using auto-updated white-list.<br>It combined the whitelist approach with heuristics and ML to propose the auto-updated whitelist. Blacklists and whitelists are used as a filtering module in many web<br>phishing detection approaches to reduce the processing time wasted on pre-processing, feature<br>extraction, and so on. |
| 4. | Social Impact / Customer Satisfaction | This system can be used by many E-commerce or other websites in order to have good customer relationship.<br>User can make online payment securely.<br>Data mining algorithm used in this system provides better performance as compared to other traditional classifications algorithms.<br>With the help of this system user can also purchase products online without any hesitation. |
| 5. | Business Model (Revenue Model) | The 2020 Cyber Security Breaches Survey identified phishing attacks as the<br>most disruptive form of cyberattack for UK businesses. For 67% of businesses, the single most disruptive attack in the last 12 months was a phishing attack.<br>Phishing attacks can paralyse a business. Staff might be unable to continue their work. Data and assets might be stolen or damaged. Customers might be unable to access online services.<br>Most businesses are able to restore operations within 24 hours. But in cases with a material outcome – including a loss of money or data – 41% of businesses take<br>a day or more to recover. |

| | | |
|---|---|---|
| 6. | Scalability of the Solution | Whitelists can reduce false positives, improve performance, and reduce vulnerability to malware. However, whitelisting can be labor-intensive and time-consuming.Data mining is used in making better decisions, having a competitive advantage, and finding major problems. The Decision Tree algorithm is inadequate for applying regression and predicting continuous values. |

## 3.1 Problem Solution fit



# 4. REQUIREMENT ANALYSIS

## 4.1 Functional requirement:

| FR No. | Functional Requirement (Epic) | Sub Requirement (Story / Sub-Task) |
|---|---|---|
| FR-1 | **Features Extraction** | Lexical Features.<br>Hyperlink Features.<br>URL Features.<br>Textual content Features. |
| FR-2 | **Data Base Collection** | Phishing URL's.<br>Non-Phishing URL. |
| FR-3 | **Machine Learning Classifier Training** | Identify the Criteria.<br>Build a decision tree.<br>Train our model.<br>Evaluate our model.<br>Check for false positives/negatives. |
| FR-4 | **Features Set Classification** | Address Bar based Features.<br>Abnormal Based Features.<br>Domain Based Features.<br>HTML & JavaScript Based Features. |
| FR-5 | **Algorithm** | Data Mining Algorithm.<br>PhishDekt Algorithm. |
| FR-6 | **Techniques** | Whitelist & Blacklist Techniques.<br>Layout Based Detection Schemes. |

## 4.2 Non-Functional requirements:

| FR No. | Non-Functional Requirement | Description |
|---|---|---|
| NFR-1 | **Usability** | The internet users can assist antiphishing tools and technology which provide essential information, such as warning of spoofed pages. |
| NFR-2 | **Security** | The list-based detection will alert the users before entering into the phishing websites. |
| NFR-3 | **Reliability** | Provide warning message to the users when it fails to detect the blacklisted URL are encountered with minor changes |
| NFR-4 | **Performance** | The phishing websites can be detected with 97.95% Accuracy |
| NFR-5 | **Availability** | Users can utilize the ML algorithm to detect attacks based on features extracted from URL |
| NFR-6 | **Scalability** | ML based models is able to detect 0-day attacks which is scalable and accurate |

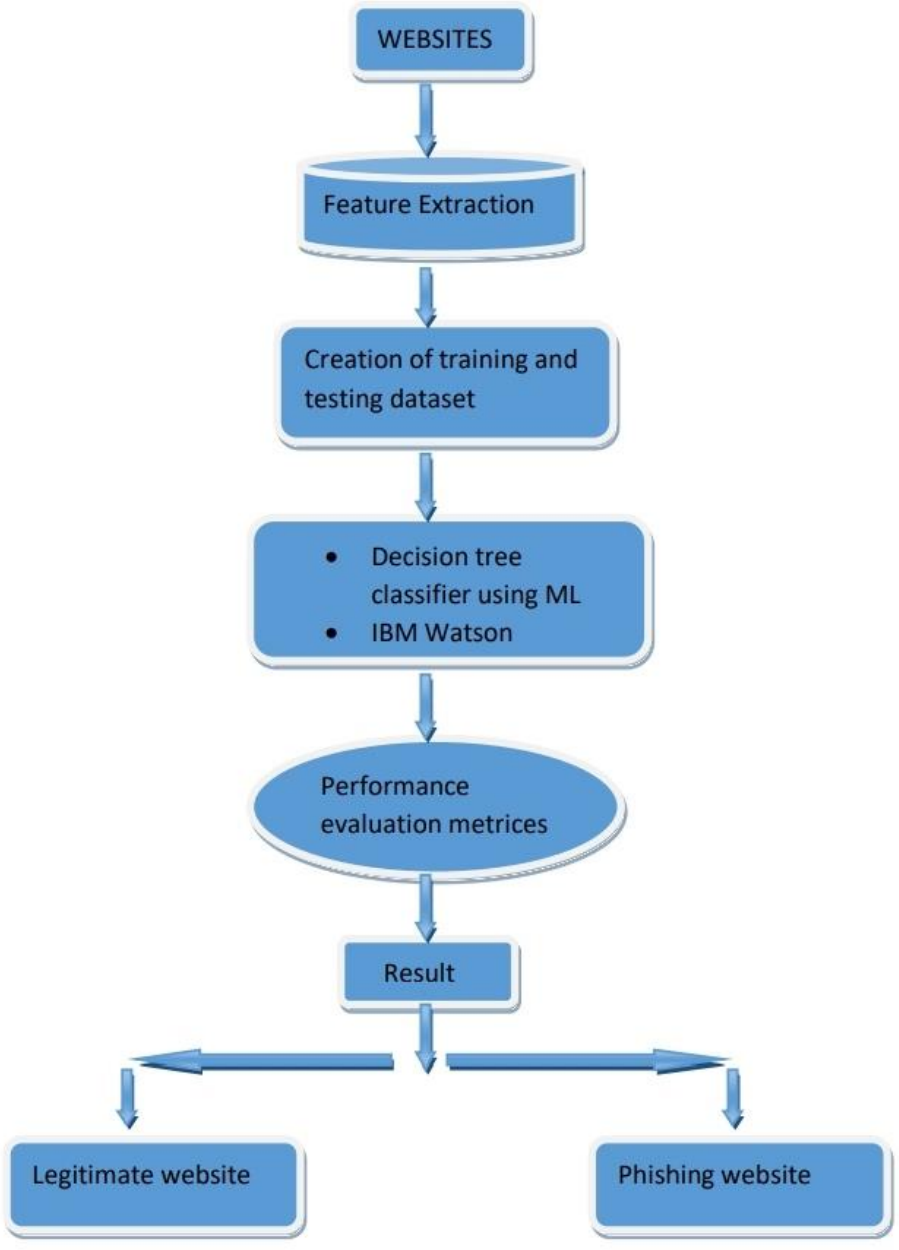
## 5. PROJECT DESIGN

Processes are something that are often overlooked in our industry, but are absolutely essential for a number of reasons.

They help you create a repeatable template for a winning formula.

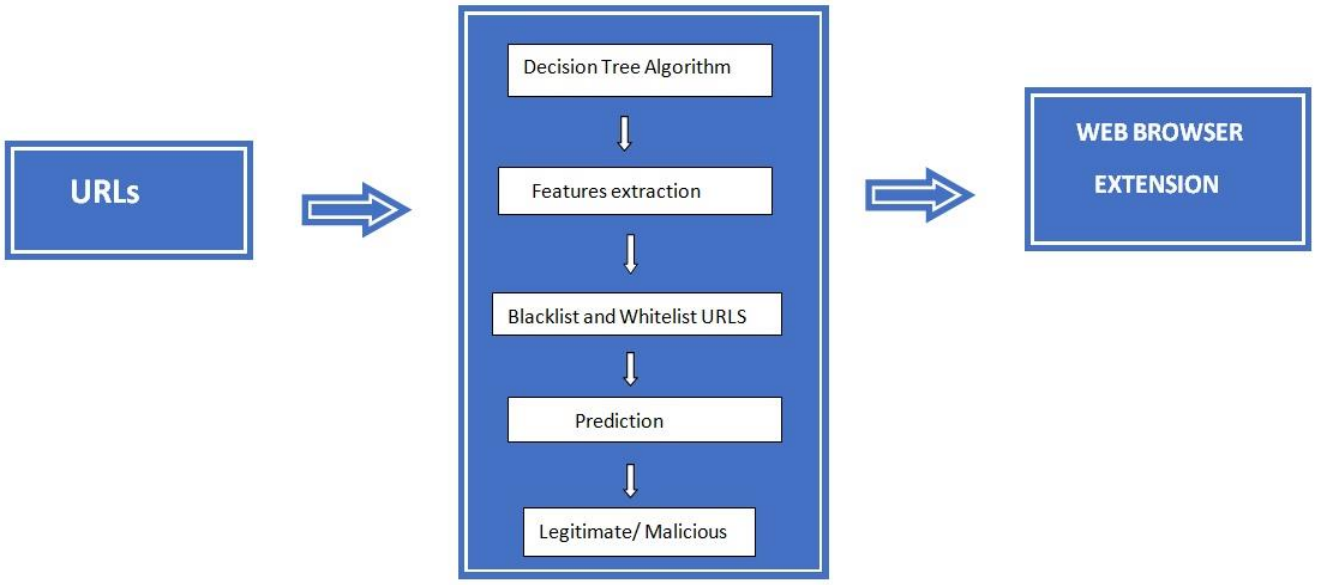
They help your team understand how to move through a project in the correct way.

### 5.1Data Flow Diagrams:

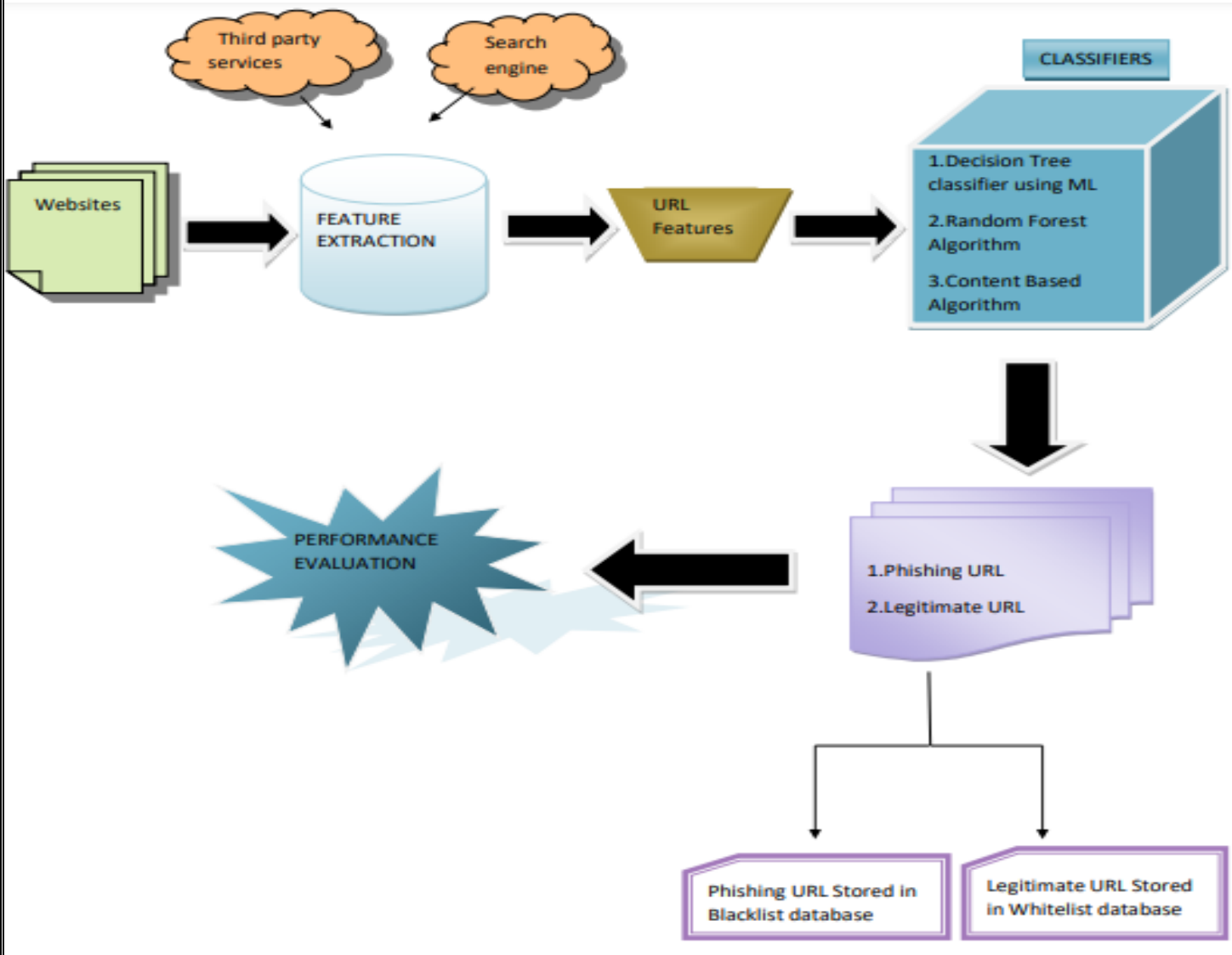


## 5.2Solution & Technical Architecture:

**TECHNICAL ARCHITECTURE:**




**SOLUTION ARCHITECTURE:**



## Table-1 : Components & Technologies:

| S.No | Component | Description | Technology |
|---|---|---|---|
| 1. | User Interface | How user interacts with application e.g.Web UI, Mobile App, Chat bot etc. | Web extension, JavaScript . |
| 2. | Application Logic-1 | Logic for a process in the application | Python/ Java |
| 3. | Application Logic-2 | Logic for a process in the application | IBM cloud , Flask server |
| 4. | Database | Data Type, Configurations etc. | Hierarchical database, networkdatabase systems |
| 5. | Cloud Database | Database Service on Cloud | IBM Watson |
| 6. | File Storage | File storage requirements | IBM Cloud Storage or Other Storage Service or Local File system |
| 7. | Machine Learning Model | Purpose of Machine Learning Model | Decision Tree classifier, Regressionmodel, etc |
| 8. | Infrastructure (Server / Cloud) | Application Deployment on Local System / CloudLocal Server Configuration: Cloud Server Configuration : | Local, Cloud Foundry, Kubernetes |

## Table-2: Application Characteristics:

| S.No | Characteristics | Description | Technology |
|---|---|---|---|
| 1. | Open-Source Frameworks | List the open-source frameworks used | Sniperphish, Gophish |
| 2. | Security Implementations | List all the security / access controls implemented,use of firewalls etc. | Two factor authentication, Firewall |
| 3. | Scalable Architecture | Justify the scalability of architecture (3 – tier,Micro-services) | Response time, Throughput |
| 4. | Availability | Justify the availability of application (e.g. use ofload balancers, distributed servers etc.) | Auto scaling based on user demand |
| 5. | Performance | Design consideration for the performance of theapplication (number of requests per sec, use of Cache, use of CDN’s) etc. | Blacklist, Whitelist, ML techniques |

## 5.3User 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 | I can register & access the dashboard with Gmail Login | Medium | Sprint-1 |
| | Login | USN-5 | As a user, I can log into the application by entering email & password | I can access my dashboard | High | Sprint-1 |
| | Dashboard | USN-6 | As a user, I can access the dashboard to get information | I can access my application | High | Sprint-1 |
| Customer (Web user) | Registration | USN-7 | As a web user, I can register my details in official websites and I will create strong passwords | I can access my dashboard/account safely | High | Sprint-1 |
| | Login & Dashboard | | As a web user, I can login into application by using my user id and password | I can access the resources | High | Sprint-1 |
| Customer Care Executive | Login | CCE-1 | As a CCE I can login to website using user id and password and I can interact with the user | I can access the website | High | Sprint-1 |
| | Dashboard | CCE-2 | As a CCE I can login to dashboard using user id and password and I can interact with the user and I can explain the app usage and rectify their issues. | I can access the resources | High | Sprint-1 |
| Administrator | Login & Dashboard | A-1 | As an administrator, I can access the dashboard and direct activities. | I will maintain the database safely | High | Sprint-1 |

## 6. PROJECT PLANNING & SCHEDULING

The definition of a sprint is a dedicated period of time in which a set amount of work will be completed on a project. It's part of the agile methodology, and an Agile project will be broken down into a number of sprints, each sprint taking the project closer to completion**.**

### 6.1 Sprint Planning & Estimation:

| Sprint | Functional Requirement (Epic) | User Story Number | User Story / Task | Story Points | Priority | Team Members |
|---|---|---|---|---|---|---|
| Sprint-1 | Numpy, pandas, matplotlib, seaborn, Scikit learn. | USN-1 | Collection of dataset and pre-processing the dataset. | 20 | High | Esther, Gayathri Priyadharshini, Merlin, Sakthi Eswari, Uma. |
| Sprint-2 | Scikit learn. | USN-2 | Building Machine learning model | 20 | High | Esther, Gayathri Priyadharshini, Merlin, Sakthi Eswari, Uma. |
| Sprint-3 | Flask app, visual studio code-html,css, Anaconda prompt. | USN-3 | Building an Application to integrate the model. | 20 | High | Esther, Gayathri Priyadharshini, Merlin, Sakthi Eswari, Uma. |
| Sprint-4 | IBM cloud,IBM watson. | USN-4 | Train the model on IBM. | 20 | High | Esther,Gayathri Priyadharshini,Merlin,Sakthi Eswari,Uma. |

### 6.2.Sprint Delivery Schedule:

### Project Tracker, Velocity & Burndown Chart:

| 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 | 07Nov 2022 |
| Sprint-2 | 20 | 6 Days | 31 Oct 2022 | 05 Nov 2022 | 20 | 10 Nov 2022 |
| Sprint-3 | 20 | 6 Days | 07 Nov 2022 | 12 Nov 2022 | 20 | 15 Nov 2022 |
| Sprint-4 | 20 | 6 Days | 14 Nov 2022 | 19 Nov 2022 | 20 | 19 Nov 2022 |

**Velocity:**

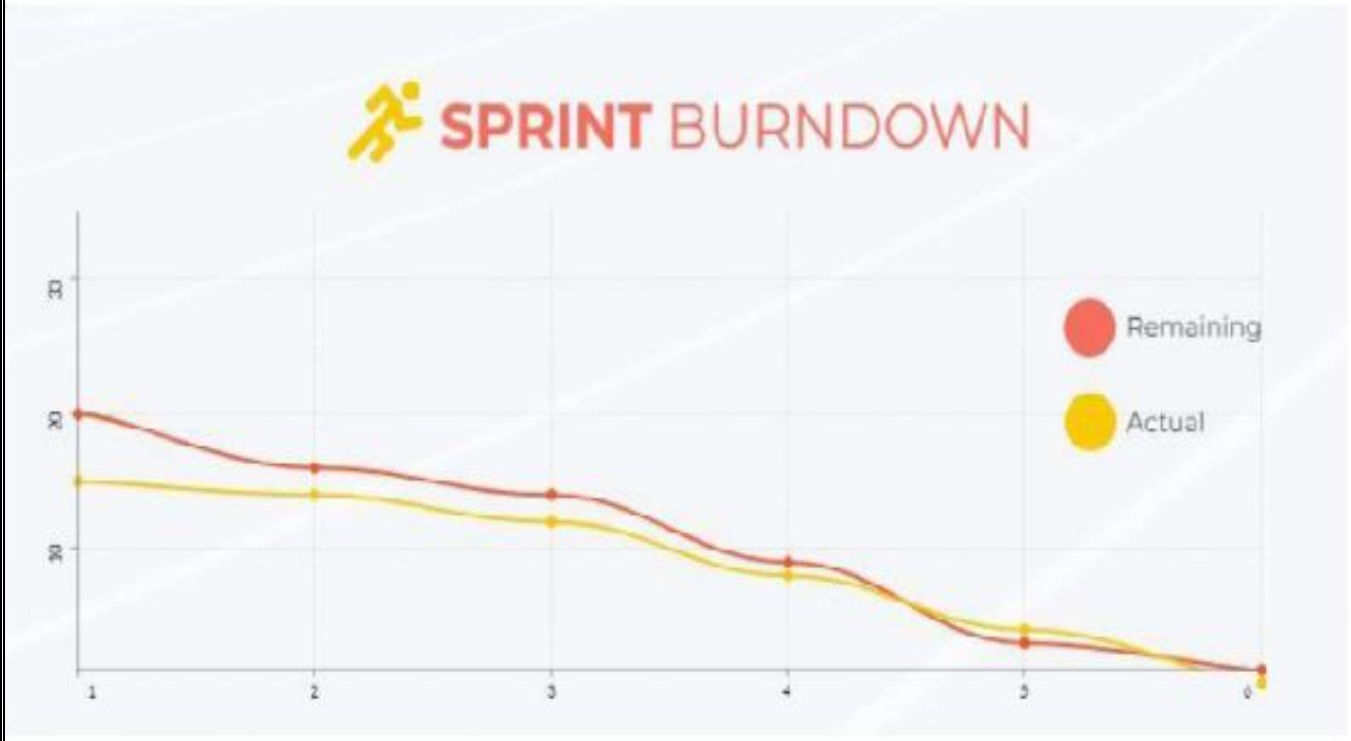
Imagine we have a 10-day sprint duration, and the velocity of the team is 20 (points per sprint). Let's calculate the team's average velocity (AV) per iteration unit (story points per day)

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

## 6.3.Reports from JIRA:

**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:

## 7.1 FEATURE 1

**app.py**

```
import numpy as np

import pandas
from flask import Flask, request, jsonify, render_template
import pickle
import inputScript

app = Flask(__name__,template_folder='templates')
model = pickle.load(open('Phishing_Website.pkl','rb'))

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



ans = ""
bns = ""
@app.route('/y_predict', methods=['POST','GET'])
def y_predict():
    url = request.form['url']
    checkprediction = inputScript.main(url)
    prediction = model.predict(checkprediction)
    print(prediction)
    output=prediction[0]
    if(output==1):
        pred="You are safe!!  This is a legitimate Website."
        return render_template('index.html',bns=pred)
    elif(output==-1):
        pred="You are on the wrong site. Be cautious!"
        return render_template('index.html',ans=pred)
    else:
        pred="You are on the wrong site. Be cautious!"
        return render_template('index.html',ans=pred)



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

    data = request.get_json(force=True)
    prediction = model.y_predict([np.array(list(data.values()))])

    output=prediction[0]
    return jsonify(output)
```

```
if __name__ == '__main__':
    app.run()
```

## 7.2 FEATURE 2
## inputScript.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
from dateutil.parser import parse as date_parse
from urllib.parse import urlparse
import favicon

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

import datetime
import requests

import re



"""
Check if URL contains any IP address. Returns -1 if contains else returns 1
"""
def having_IPhaving_IP_Address(url):
```

```
     match=regex.search(
   '(([01]?\\d\\d?|2[0-4]\\d|25[0-5])\\.([01]?\\d\\d?|2[0-4]\\d|25[0-5])\\.([01]?\\d\\d?|2[0-
4]\\d|25[0-5])\\.([01]?\\d\\d?|2[0-4]\\d|25[0-5])\\/)|'  #IPv4
                    '((0x[0-9a-fA-F]{1,2})\\.(0x[0-9a-fA-F]{1,2})\\.(0x[0-9a-fA-F]{1,2})\\.(0x[0-
9a-fA-F]{1,2})\\/)'  #IPv4 in hexadecimal
                    '(?:[a-fA-F0-9]{1,4}:){7}[a-fA-F0-9]{1,4}',url)
    #Ipv6
     if match:
        #print match.group()
        return -1
     else:
        #print 'No matching pattern found'
        return 1
"""
Check for the URL length. Return 1 (Legitimate) if the URL length is less than 54 characters
Return 0 if the length is between 54 and 75
Else return -1
"""
def URLURL_Length (url):
    length=len(url)
    if(length<=75):
         if(length<54):
            return 1
         else:
            return 0
    else:
        return -1

"""
Check with the shortened URLs.
Return -1 if any shortened URLs used.
Else return 1
"""
def Shortining_Service (url):

match=regex.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',url)
        if match:
            return -1
        else:
            return 1

    #Checking for @ symbol. Returns 1 if no @ symbol found. Else returns 0.
    def having_At_Symbol(url):
        symbol=regex.findall(r'@',url)
        if(len(symbol)==0):
            return 1
        else:
            return -1

    #Checking for Double Slash redirections. Returns -1 if // found. Else returns 1
    def double_slash_redirecting(url):
        for i in range(8,len(url)):
            if(url[i]=='/'):

                if(url[i-1]=='/'):
                    return -1
        return 1

    #Checking for - in Domain. Returns -1 if '-' is found else returns 1.
    def Prefix_Suffix(url):
        subDomain, domain, suffix = extract(url)
        if(domain.count('-')):
            return -1
        else:
            return  1

    """
    Check the Subdomain. Return 1 if the subDomain contains less than 1 '.'
    Return 0 if the subDomain contains less than 2 '.'
    Return -1 if the subDomain contains more than 2 '.'
    """
    def having_Sub_Domain(url):
        subDomain, domain, suffix = extract(url)
        if(subDomain.count('.')<=2):
            if(subDomain.count('.')<=1):
                return 1
            else:
                return 0
```

```
    else:
        return -1

#Checking the SSL. Returns 1 if it returns the respomse code and -1 if exceptions are thrown.
def SSLfinal_State(url):
    try:
        response = requests.get(url)
        return 1
    except Exception as e:
        return -1

#domains expires on ≤ 1 year returns -1, otherwise returns 1

def Domain_registeration_length(url):
    try:
        domain = whois.whois(url)
        exp=domain.expiration_date[0]
        up=domain.updated_date[0]
        domainlen=(exp-up).days
        if(domainlen<=365):
            return -1
        else:
            return 1
    except:
        return -1

#Checking the Favicon. Returns 1 if the domain of the favicon image and the URL domain match else
returns -1.
def Favicon(url):
    subDomain, domain, suffix = extract(url)
    b=domain
    try:
        icons = favicon.get(url)
        icon = icons[0]
        subDomain, domain, suffix =extract(icon.url)
        a=domain
        if(a==b):
            return 1
        else:
            return -1
    except:
        return -1

#Checking the Port of the URL. Returns 1 if the port is available else returns -1.
def port(url):
    try:
```

```
        a_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        location=(url[7:],80)
        result_of_check = a_socket.connect_ex(location)
        if result_of_check == 0:
            return 1
        else:
            return -1
        a_socket.close
    except:
        return -1

# HTTPS token in part of domain of URL returns -1, otherwise returns 1
def HTTPS_token(url):
    match=re.search('https://|http://',url)
    if (match and match.start(0)==0):
        url=url[match.end(0):]
    match=re.search('http|https',url)
    if match:
        return -1
    else:
        return 1


#% of request URL<22% returns 1, otherwise returns -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
        else:
            return -1
    except:
        return -1

#:% of URL of anchor<31% returns 1, % of URL of anchor ≥ 31% and ≤ 67% returns 0, otherwise
returns -1
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

"""
```

```
% of links in <meta>, <script>and<link>tags < 25% returns 1, % of links in <meta>,
<script> and <link> tags ≥ 25% and ≤ 81% returns 0, otherwise returns -1
"""

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

#Server Form Handling
#SFH is "about: blank" or empty → phishing, SFH refers to a different domain → suspicious,
otherwise → legitimate
def SFH(url):
    #ongoing
    return -1

#:using "mail()" or "mailto:" returning -1, otherwise returns 1
def Submitting_to_email(url):
    try:
```

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

#Host name is not in URL returns -1, otherwise returns 1
def Abnormal_URL(url):
    subDomain, domain, suffix = extract(url)
    try:
        domain = whois.whois(url)
        hostname=domain.domain_name[0].lower()
        match=re.search(hostname,url)
        if match:
            return 1
        else:
            return -1
    except:
        return -1

#number of redirect page ≤ 1 returns 1, otherwise returns 0
def Redirect(url):
    try:
        request = requests.get(url)
        a=request.history
        if(len(a)<=1):
            return 1
        else:
            return 0

    except:
        return 0


#onMouseOver changes status bar returns -1, otherwise returns 1
def on_mouseover(url):
    try:
        opener = urllib.request.urlopen(url).read()
        soup = BeautifulSoup(opener, 'lxml')

        no_of_script =0
        for meta in soup.find_all(onmouseover=True):
            no_of_script = no_of_script+1
```

```
        if(no_of_script==0):
            return 1
        else:
            return -1
    except:
        return -1

#right click disabled returns -1, otherwise returns 1
def RightClick(url):
    try:
        opener = urllib.request.urlopen(url).read()
        soup = BeautifulSoup(opener, 'lxml')
        if(soup.find_all('script',mousedown=True)):
            return -1
        else:
            return 1
    except:
        return -1

#popup window contains text field → phishing, otherwise → legitimate
def popUpWidnow(url):
    #ongoing
    return 1

#using iframe returns -1, otherwise returns 1
def Iframe(url):
    try:
        opener = urllib.request.urlopen(url).read()
        soup = BeautifulSoup(opener, 'lxml')
        nmeta=0
        for meta in soup.findAll('iframe',src=True):
            nmeta= nmeta+1
        if(nmeta!=0):
            return -1
        else:
            return 1
    except:
        return -1



#:age of domain ≥ 6 months returns 1, otherwise returns -1
def age_of_domain(url):
    try:
        w = whois.whois(url).creation_date[0].year
        if(w<=2018):
            return 1
```

```
        else:
            return -1
    except Exception as e:
        return -1

#no DNS record for domain returns -1, otherwise returns 1
def DNSRecord(url):

    subDomain, domain, suffix = extract(url)
    try:
        dns = 0
        domain_name = whois.whois(url)
    except:
        dns = 1

    if(dns == 1):
        return -1
    else:
        return 1

#website rank < 100.000 returns 1, website rank > 100.000 returns 0, otherwise returns -1
def web_traffic(url):
    try:
        rank =
BeautifulSoup(urllib.request.urlopen("http://data.alexa.com/data?cli=10&dat=s&url=" +
url).read(), "lxml").find("REACH")['RANK']
    except TypeError:
        return -1
    rank= int(rank)
    if (rank<100000):
        return 1
    else:
        return 0

#:PageRank < 0,2 → phishing, otherwise → legitimate
def Page_Rank(url):
    #ongoing
    return 1

#webpage indexed by Google returns 1, otherwise returns -1
def Google_Index(url):
    try:
        subDomain, domain, suffix = extract(url)
        a=domain + '.' + suffix
        query = url
        for j in search(query, tld="co.in", num=5, stop=5, pause=2):
```

```
            subDomain, domain, suffix = extract(j)
            b=domain + '.' + suffix
        if(a==b):
            return 1
        else:
            return -1
    except:
        return -1


#:number of links pointing to webpage = 0 returns 1, number of links pointing to webpage> 0
#and ≤ 2 returns 0, otherwise returns -1

def Links_pointing_to_page (url):
    try:
        opener = urllib.request.urlopen(url).read()
        soup = BeautifulSoup(opener, 'lxml')
        count = 0
        for link in soup.find_all('a'):
            count += 1
        if(count>=2):
            return 1
        else:
            return 0
    except:
        return -1

#:host in top 10 phishing IPs or domains returns -1, otherwise returns 1
def Statistical_report (url):
    hostname = url
    h = [(x.start(0), x.end(0)) for x in
regex.finditer('https://|http://|www.|https://www.|http://www.', hostname)]
    z = int(len(h))
    if z != 0:
        y = h[0][1]
        hostname = hostname[y:]
        h = [(x.start(0), x.end(0)) for x in regex.finditer('/', hostname)]
        z = int(len(h))
        if z != 0:
            hostname = hostname[:h[0][0]]

url_match=regex.search('at\.ua|usa\.cc|baltazarpresentes\.com\.br|pe\.hu|esy\.es|hol\.es|sweddy\.
com|myjino\.ru|96\.lt|ow\.ly',url)
    try:
        ip_address = socket.gethostbyname(hostname)
```

```
ip_match=regex.search('146\.112\.61\.108|213\.174\.157\.151|121\.50\.168\.88|192\.185\.217\.116|7
8\.46\.211\.158|181\.174\.165\.13|46\.242\.145\.103|121\.50\.168\.40|83\.125\.22\.219|46\.242\.14
5\.98|107\.151\.148\.44|107\.151\.148\.107|64\.70\.19\.203|199\.184\.144\.27|107\.151\.148\.108|1
07\.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\.1
0\.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\.1
00\.26\.234|195\.16\.127\.102|195\.16\.127\.157|34\.196\.13\.28|103\.224\.212\.222|172\.217\.4\.2
25|54\.72\.9\.51|192\.64\.147\.141|198\.200\.56\.183|23\.253\.164\.103|52\.48\.191\.26|52\.214\.1
97\.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)
    except:
        return -1

    if url_match:
        return -1
    else:
        return 1

#returning scrapped data to calling function in app.py
def main(url):



    check = [[having_IPhaving_IP_Address
(url),URLURL_Length(url),Shortining_Service(url),having_At_Symbol(url),

double_slash_redirecting(url),Prefix_Suffix(url),having_Sub_Domain(url),SSLfinal_State(url),

Domain_registeration_length(url),Favicon(url),port(url),HTTPS_token(url),Request_URL(url),

URL_of_Anchor(url),Links_in_tags(url),SFH(url),Submitting_to_email(url),Abnormal_URL(url),
              Redirect(url),on_mouseover(url),RightClick(url),popUpWidnow(url),Iframe(url),

age_of_domain(url),DNSRecord(url),web_traffic(url),Page_Rank(url),Google_Index(url),
              Links_pointing_to_page(url),Statistical_report(url)]]



    print(check)
    return check
```

# index.html

```
<!DOCTYPE html>

<html lang="en">
  <head>
    <title>Phishing URL detection</title>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1" />
    <link rel="stylesheet" href="style.css" />
  </head>
  <body>
    <div class="navbar">
      <ul class="logo_wrapper">
        <li><h1 class="hook">HOOK PHISH</h1></li>
        <li><a href="#web" class="right">Check Website</a></li>
        <li><a href="#side" class="right">About Us</a></li>
      </ul>
    </div>

    <div class="header">
      <h1>PHISHING WEBSITE DETECTION</h1>
      <p>a measure for detecting malicious websites</p>
    </div>

    <div class="row">
      <div class="side" id="side">
        <div class="wrapper">
          <div class="About">
            <div class="About_left">
              <h2>About</h2>
              <p>
                Phishing is a form of fraud in which the attacker tries to learn
                sensitive information such as login credentials or account
                information by sending as a reputable entity or person in email
                or other communication channels.Phishing attacks can paralyze a
                business. Staff might be unable to continue their work. Data and
                assets might be stolen or damaged. Customers might be unable to
                access online services. The reason security defenders struggle
                to detect phishing domains is because of the unique part of the
                website domain.
              </p>
```

```
            </div>
          </div>
        </div>
      </div>

    <div class="web" id="web">
      <div class="wrapper">
        <h2>Check Website</h2>
        <p>
          Understanding if the website is a valid one or not is important and
          plays a vital role in securing the data. To know if the URL is a valid
          one or your information is at risk. Check your website
        </p>
        <form name="form" action="/y_predict" method="post" class="body">
        <input
          type="text"
          id="url"
          name="url"
          placeholder="Enter a URL"
          size="50"
        />
          <br /><br />
        <button type="submit" class="url_button">Submit</button>
        </form>
        <h3 style="text-align: center; color: red; font-size: 20px">{{ans}}</h3>
        <h3 style="text-align: center; color: green; font-size: 20px">{{bns}}</h3>
      </div>
    </div>

    <div class="footer">
      <h2>PROTECT YOURSELF FROM PHISHING ATTACKS</h2>
      <p>Copyright © 2022 University VOC College of Engineering(Tuticorin). All Rights
Reserved.</p>
    </div>
  </body>
<style>
  html {
  scroll-behavior: smooth;
}

* {
  box-sizing: border-box;
}

/* Style the body */
```

```
body {
  font-family: Arial, Helvetica, sans-serif;
  margin: 0;
}

/* Header/logo Title */
.header {
  padding: 80px;
  text-align: center;
  background: #2a035e;
  color: rgb(250, 229, 229);
}

/* Increase the font size of the heading */
.header h1 {
  font-size: 40px;
}

/* Style the top navigation bar */
.navbar {
  overflow: hidden;
  background-color: rgb(190, 1, 1);
}

/* Style the navigation bar links */
.navbar a {
  float: left;
  display: block;
  color: white;
  text-align: center;
  padding: 14px 20px;
  text-decoration: none;
  font-size: xx-large;
}

/* Right-aligned link */
.navbar a.right {
  float: right;
  font-size: 19px;
}
.hook {
  margin: 0 auto;
  text-align: center;
  float: none !important;
  padding: 41px 157px 0px 86px !important;
  color: white;
```

```
  font-size: 40px;
}
#url {
  width: 50%;
  padding: 12px 20px;
  margin: 8px 0;
  box-sizing: border-box;
  border: none;
  background-color: #cbcddf;
  color: rgb(21, 1, 1);
}
.url_button {
  background-color: #2a035e;
  border: none;
  color: rgb(242, 229, 229);
  padding: 16px 32px;
  text-decoration: none;
  margin: 4px 2px;
  cursor: pointer;
}

/* Change color on hover */
.navbar a:hover {
  background-color: #ddd;
  color: black;
}

/* Column container */
.row {
  display: -ms-flexbox; /* IE10 */
  display: flex;
  -ms-flex-wrap: wrap; /* IE10 */
  flex-wrap: wrap;
}

/* Create two unequal columns that sits next to each other */
/* Sidebar/left column */
.side {
  -ms-flex: 30%; /* IE10 */
  flex: 30%;
  background-color: #f1f1f1;
  padding: 20px;
}
.side h2, .web h2 {
  text-align: center;
  font-weight: var(--h-font-weight);
```

```
  color: var(--heading-color);
  line-height: 1.15em;
  font-family: var(--h-family-body);
  font-size: 3rem;
}

.About p, .web p{
  line-height: 35px;
}

.About_left {
  text-align: center;
}



.phishing-img {
  width: 500px;
}
.web {
  text-align: center;
  padding: 10px 0px 50px 0px;
  }

/* Footer */
.footer {
  padding: 20px;
  text-align: center;
  background: rgba(209, 5, 5, 0.884);
  color: rgb(250, 229, 229);
}

.logo_wrapper {
  max-width: 1283px;
  margin: 0 auto;
}

.logo_wrapper li {
  list-style: none;
}

.wrapper {
  max-width: 1000px;
  margin: 0 auto;
}

/* Responsive layout - when the screen is less than 700px wide, make the two columns stack on
```

```
top of each other instead of next to each other */
@media screen and (max-width: 700px) {
  .row {
    flex-direction: column;
  }
}

/* Responsive layout - when the screen is less than 400px wide, make the navigation links stack on top of each other instead of next to each other */
@media screen and (max-width: 400px) {
  .navbar a {
    float: none;
    width: 100%;
  }
}
</style>
</html>
```

# 8.TESTING

## 8.1 Test Cases

| 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| LoginPage_TC_001 | Functional | Home Page | Verify user is able to see the landing Page when user can type the URL in the box | | 1.Enter URL and click go<br>2.Type the URL<br>3.Verify whether it is processing or not | https://phishingshield.herokuapp.com/ | Should Display the Webpage | Working as expected | Pass | | N | |
| LoginPage_TC_002 | UI | Home Page | Verify the UI elements in Responsive | | 1.Enter URL and click go<br>2. Type or copy paste the URL<br>3. Check whether the button is responsive or not<br>4. Reload and Test Simultaneously | https://phishingshield.herokuapp.com/ | Should Wait for Response and then gets Acknowledge | Working as expected | Pass | | N | |
| LoginPage_TC_003 | Functional | Home page | Verify whether the link is legitimate or not | | 1.Enter URL and click go<br>2. Type or copy paste the URL<br>3. Check the website is legitimate or not<br>4. Observe the results | https://phishingshield.herokuapp.com/ | User should observe whether the website is legitimate or not. | Working as expected | Pass | | N | |
| LoginPage_TC_004 | Functional | Home page | Verify user is able to access the legitimate website or not | | 1.Enter URL and click go<br>2. Type or copy paste the URL<br>3. Check the website is legitimate or not<br>4. Continue if the website is legitimate or be cautious if it is not legitimate. | https://phishingshield.herokuapp.com/ | Application should show that Safe Webpage or Unsafe. | Working as expected | Pass | | N | |
| LoginPage_TC_005 | Functional | Home page | Testing the website with multiple URLs | | 1.Enter URL ( https://phishingshield.herokuapp.com/) and click go<br>2. Type or copy paste the URL to test<br>3. Check the website is legitimate or not<br>4. Continue if the website is secure or be cautious if it is not secure | 1. google.com<br>2. onprice.com | User can able to identify the websites whether it is secure or not | Working as expected | Pass | | N | |

# 8.2User Acceptance Testing

## UAT Execution & Report Submission

### 1. Purpose of Document

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

### 2. Defect Analysis

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

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

### 3. Test Case Analysis

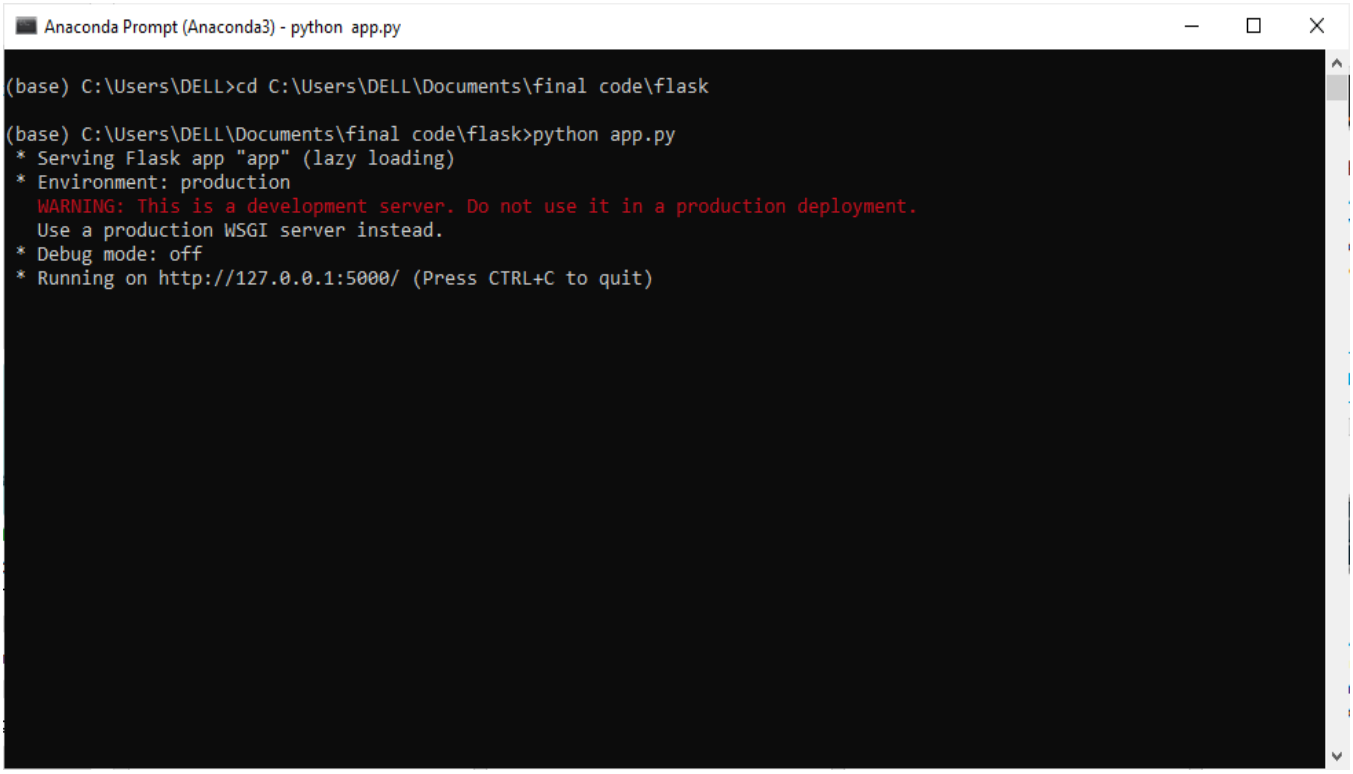
This report shows the number of test cases that have passed, failed, and untested

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

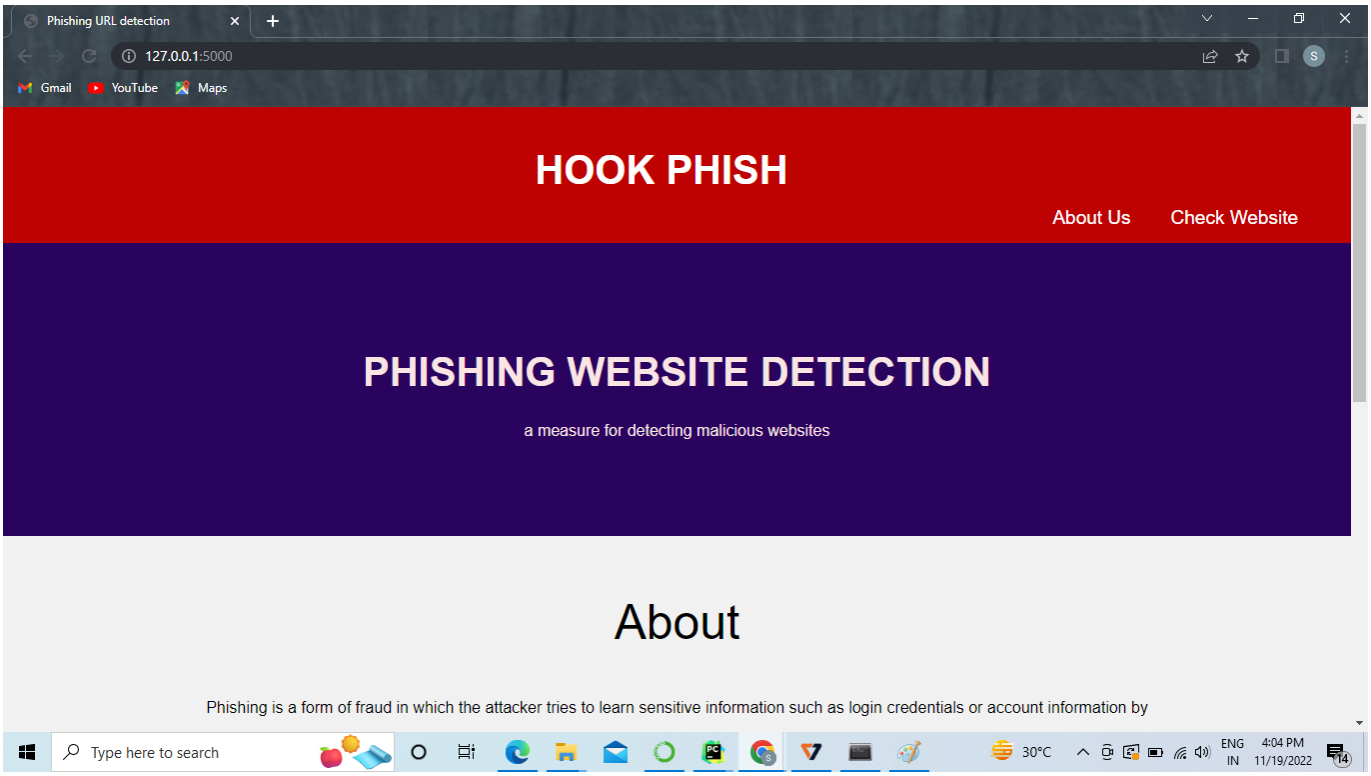
# 9. RESULTS

## 9.1Performance Metrics

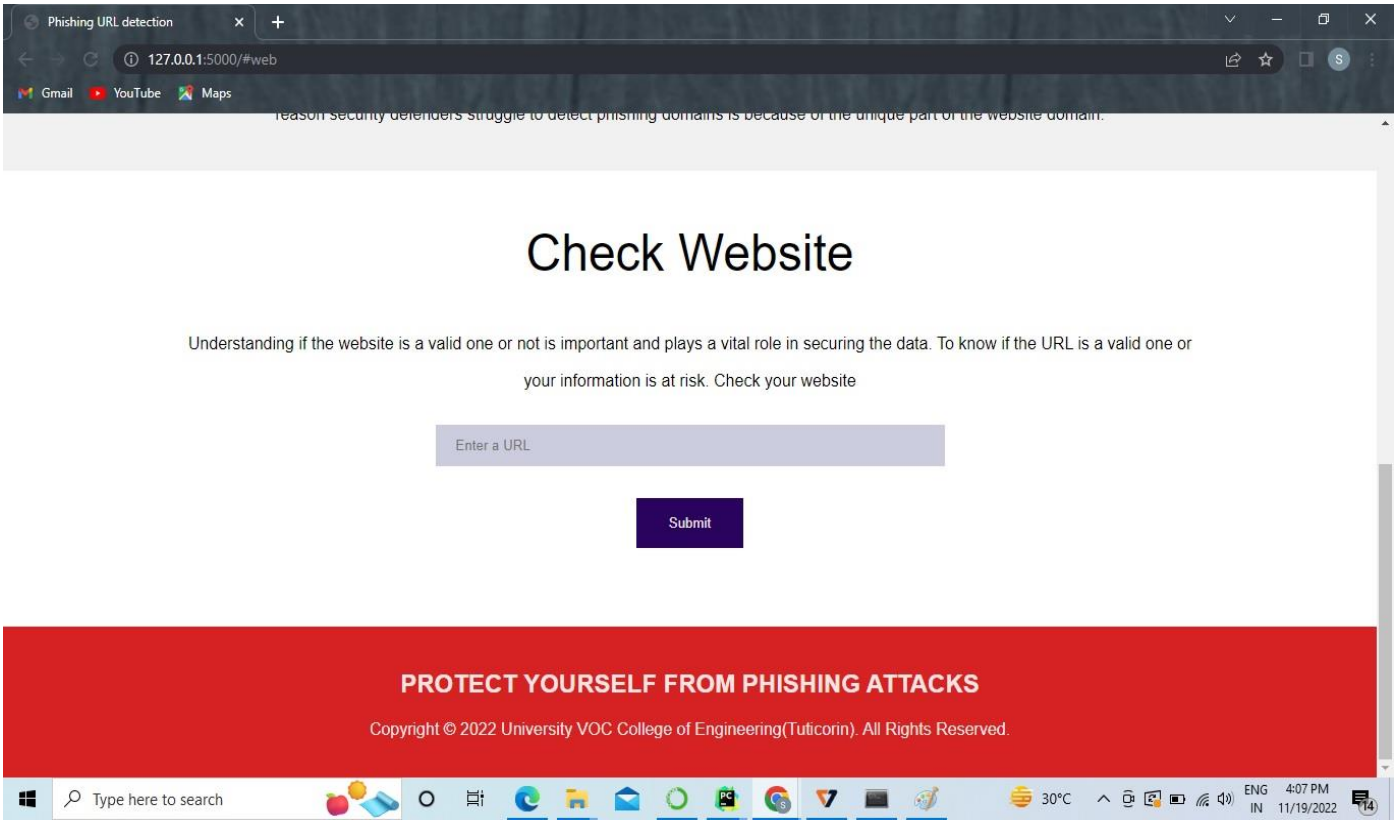
## ANACONDA PROMPT



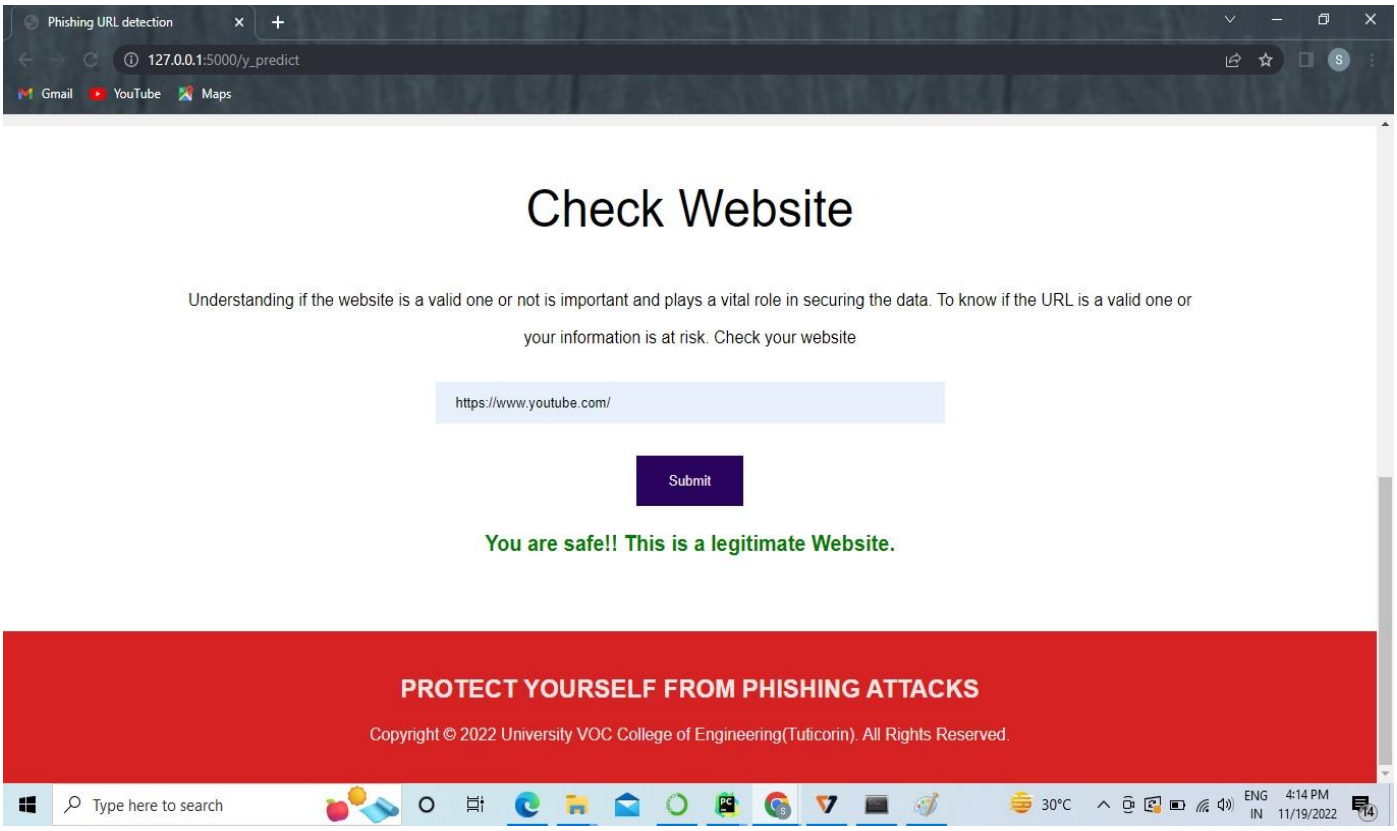
## HOMEPAGE



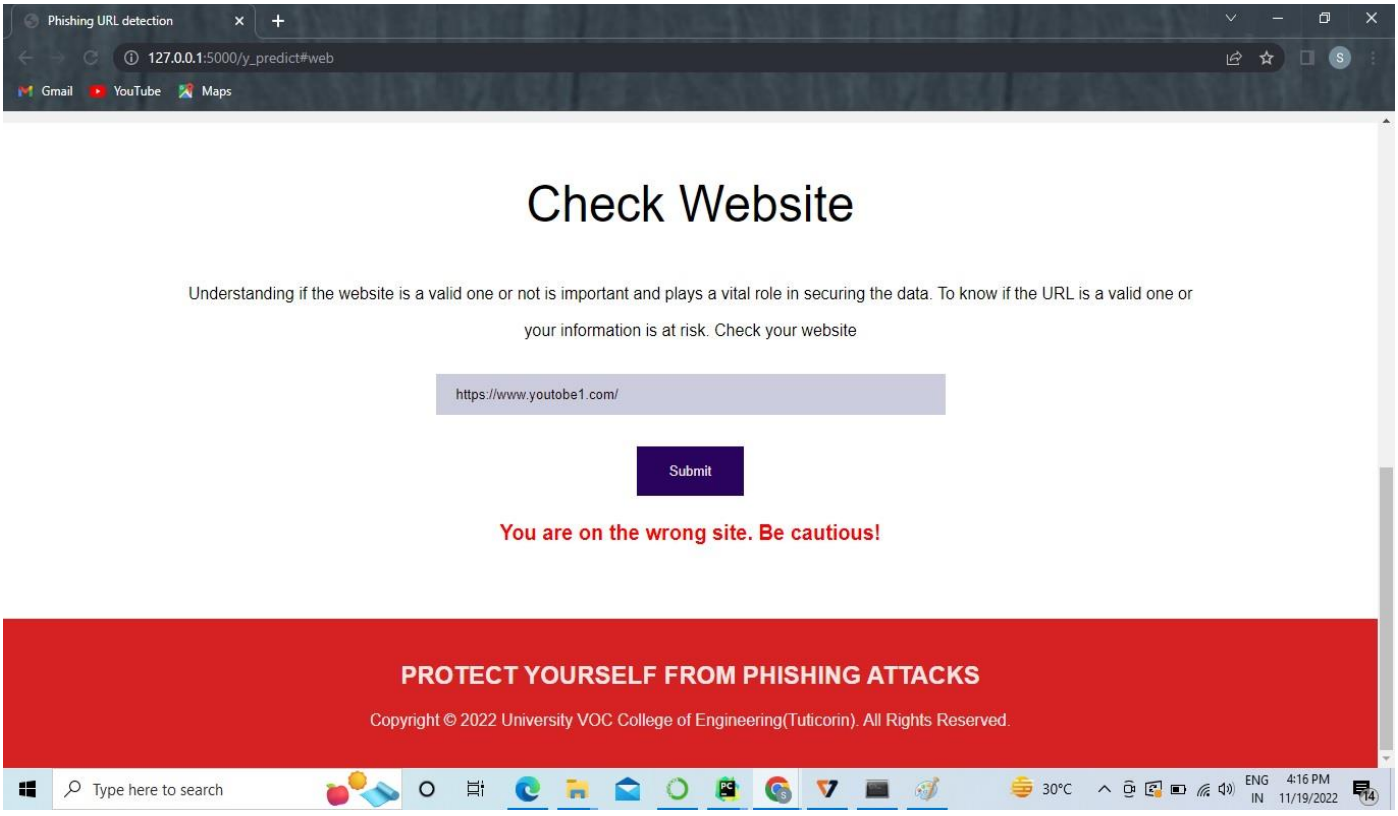

## CHECK WEBSITE



## LEGITIMATE LINK




## MALICIOUS WEBSITE



## 10.ADVANTAGES & DISADVANTAGES

### Advantages:

#### Blacklists:

- Requiring low resources on host machine
- Effective when minimal FP rates are required.

#### Heuristics and visual similarity:

- Mitigate zero hour attacks.

#### Machine Learning:

- Mitigate zero hour attacks.
- Construct own classification models.

### Disadvantages:

- Mitigation of zero-hour phishing attacks.
- Can result in excessive queries with heavily loaded servers
- Higher FP rate than blacklists
- High computational cost.
- Time consuming.
- Costly.
- Huge number of rules

## 11.CONCLUSION

Education awareness is the most significant strategy to protect users from phishing attacks. Internet users should be aware of all security recommendations made by professionals. Every user should also be taught not to mindlessly follow links to websites where sensitive information must be entered. Before visiting a website, make sure to check the URL. In the future, the system could be upgraded to automatically detect the web page and the application's compatibility with the web

browser. Additional work can be done to distinguish fraudulent web pages from authentic web pages by adding certain additional characteristics.

## 12.FUTURE SCOPE

Phishing is a considerable problem differs from the other security threats such as intrusions and Malware which are based on the technical security holes of the network systems. The weakness point of any network system is its Users. Phishing attacks are targeting these users depending on the trikes of social engineering.

Despite there are several ways to carry out these attacks, unfortunately the currentphishing detection techniques cover some attack vectors like email and fake websites. Therefore, building a specific limited scope detection system will not provide complete protection from the wide phishing attack vectors

## 13.APPENDIX

**Github link:**

https://github.com/IBM-EPBL/IBM-Project-15796-1659604710/blob/main/Final%20Deliverables/Demo%20video/PNT2022TMID50791_Web%20phishing%20detection.mp4

**Project demo link:**

https://drive.google.com/file/d/1iJRM4S-rC-WNd7KWCs7vTvGNiKBOCx4b/view?usp=drivesdk

**References:**


- https://towardsdatascience.com/phishing-domain-detection-with-ml-5be9c99293e5
- https://ietresearch.onlinelibrary.wiley.com/doi/full/10.1049/iet-net.2020.0078