IBM

NALAIYA THIRAN

ON

WEB PHISING DETECTION

(Team ID : PNT2022TMID06282)

PROJECT REPORT

TeamLeader	**:Praveen H**
TeamMember 1	**:Navaneetha krishnan**
TeamMember 2	**:Naveenkumar M**
TeamMember 3	**:Rohith K A**

INDEX

CHAPTER - 1
INTRODUCTION

1.1 Project Overview

Phishing is one of the most severe cyber-attacks where researchers are interested to find a solution. In phishing, attackers lure end-users and steal their personal information. To minimize the damage caused by phishing must be detected as early as possible. There are various phishing attacks like spear phishing, whaling, vishing, smishing, pharming and so on. There are various phishing detection techniques based on whitelist, black-list, content-based, URL-based, visualsimilarity and machine-learning. In this paper, we discuss various kinds of phishing attacks, attack vectors and detection techniques for detecting the phishing sites. Performance comparison of 18 different models along with nine different sources of datasets are given. Challenges in phishing detection techniques are also given.

In recent days cyber-attacks are increasing at an unprecedented rate.Phishing is one among those cyberattacks. In phishing, attackers lure the end-users by making them click the hyper-links which make them lose their personally identifiable information, banking and credit card details, and passwords. In this attack the attackers disguise themselves as trusted entities such as service providers, employees of the organization or technical-support team from the organization so that end-users never doubt them. It is mainly done through emails asking to update the system, or saying that account has been suspended, or asking to claim the prize and so on [59]. The main goal of phishing is to make end-users share their sensitive information. Now-a-days information regarding anything is available online and that information is stored in websites. Websites help the end-users by providing them information about their respective products,services or helping the end-users if they face any problem by chatbots, message forums and so on. Websites also store the personal information of the end-users. As websites help the end-users in gaining information they can be used as bait for trapping the end-users to obtain confidential information from them.

1.2 Purpose

Nowadays phishing becomes a main area of concern for security researchers because it is not difficult to create the fake website which looks so close to legitimate website. experts can identify fake websites but not all the users can identify the fake website and such users become the victim of phishing attack. main aim of the attacker is to steal banks account credentials. phishing attacks are becoming successful because lack of user awareness. since phishing attack exploits the weaknesses found in users, it is very difficult to mitigate them but it is very important to enhance phishing detection techniques. phishing may be a style of broad extortion that happens once a pernicious web site act sort of a real one memory that the last word objective to accumulate unstable info, as an example, passwords, account focal points, or mastercard numbers. all the same, the means that there square measure some of contrary to phishing programming and techniques for recognizing potential phishing tries in messages and characteristic phishing substance on locales, phishes think about new and crossbreed procedures to bypass the open programming and frameworks. phishing may be a fraud framework that uses a mixture of social designing what is additional, advancement to sensitive and personal data, as an example, passwords associate degree open- end credit unpretentious elements by presumptuous the highlights of a reliable individual or business in electronic correspondence. phishing makes use of parody messages that square measure created to seem substantial and instructed to start out from true blue sources like money connected institutions, online business goals, etc, to draw in customers to go to phony destinations through joins gave within the phishingwebsites.

CHAPTER - 2
LITERATURE SURVEY

2.1 Existing problem

Machine learning classifiers with wrapper features were proposed in this study. Their results were compared with the benchmark models. Machine learning with wrapper-based features outperformed the other feature selection methods. Some limitations were noticed after the evaluation of the research that was conducted in. One of these limitations is that it can't detect the embedded objects, including iframes, Flash, and HTML files to provide detection for multiple heuristics-basedapproaches.

Nguyen et al. presented a novel methodology for detection of phishing website based on a ML classifier as well as a wrapper features selection technique. The authors had achieved the detection by using selected supervised ML techniques. The key feature was selected by using the ML-based wrapper features technique that demonstrated a high performance for detection of phishing websites. The experimental results from this study presented better performance of the ML techniques because wrapper features selection was embedded with the proposed approach. Moreover, the ML technique and the wrapper-based features selection offered researchers an opportunity to extend their research to improve phishing websites' classification and detection. As compared to a single ML technique, the combined method worked better to achieve the targeted goals of detecting phishingwebsites.

Applications of ML techniques to identify phishing attacks were reported in the form of positive rate and negative rate. In this research, the authors had identified the most suitable ML algorithm for anti-phishing attacks. They had proposed a phishing classification method that captures attributes that are useful to overcome the shortcomings of phishing detection techniques. In this research, the authors had applied the use of numeric representation. Metadata of URLs were used for the determination of a website that either legitimate one or not. The authors had used ML algorithms: Random Forest, KNN, D-Tree, Linear-SVC classifier, SVM classifier, and wrapper-based (W-B) features selection. Random Forest and SVM models outperformed the rest of themodels.

2.2 References

1. Phishing Activity Trends Report: 4rd Quarter 2020. Anti-Phishing Work.Group. Retrieved April 2021, 30,2020.
2. FBI. 2019 Internet Crime Report Released-FBI. Available online: https://www.fbi.gov/news/stories/2019-internet-crime-report-released-021120. (accessed on 11 February2020).
3. Mohammad, R.M.; Thabtah, F.; McCluskey, L. Tutorial and critical analysisof phishing websites methods. Comput. Sci. Rev. 2015, 17, 1–24. [Google Scholar] [CrossRef][GreenVersion]
4. Almomani, A.; Wan, T.C.; Altaher, A.; Manasrah, A.; ALmomani, E.; Anbar, M.; ALomari, E.; Ramadass, S. Evolving fuzzy neural network for phishingemails detection. J. Comput. Sci. 2012, 8, 1099. [GoogleScholar]
5. Prakash, P.; Kumar, M.; Kompella, R.R.; Gupta, M. Phishnet: Predictive blacklisting to detect phishing attacks. In Proceedings of the 2010 Proceedings IEEE INFOCOM, San Diego, CA, USA, 14–19 March 2010; pp.1–5. [GoogleScholar]
6. Zhang, J.; Porras, P.A.; Ullrich, J. Highly Predictive Blacklisting. InProceedings of the USENIX Security Symposium, San Jose, CA, USA, 28 July–1 August 2008; pp. 107–122. [GoogleScholar]
7. Cao, Y.; Han, W.; Le, Y. Anti-phishing based on automated individual white-list. In Proceedings of the 4th ACM Workshop on Digital Identity Management, Alexandria, VA, USA, 31 October 2008; pp. 51–60. [GoogleScholar]
8. Srinivasa Rao, R.; Pais, A.R. Detecting phishing websites using automation of humanbehavior.InProceedingsofthe3rdACMWorkshoponCyber-Physical SystemSecurity,AbuDhabi,UnitedArabEmirates,2–4April2017;pp.33–42. [GoogleScholar]
9. Rao, R.S.; Ali, S.T. Phishshield: A desktop application to detect phishing webpages through heuristic approach. Procedia Comput. Sci. 2015,54, 147–156. [Google Scholar] [CrossRef][GreenVersion]
10. Joshi, Y.; Saklikar, S.; Das, D.; Saha, S. PhishGuard: A browser plug-in for protection from phishing. In Proceedings of the 2008 2nd International Conference on Internet Multimedia Services Architecture andApplications, Las Vegas, NV, USA, 14–17 July 2008; pp. 1–6. [GoogleScholar]

2.3 Problem Statement Definition

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.

Web phishing aims to steal private information, such as usernames, passwords, and credit card details, by way of impersonating a legitimate entity.

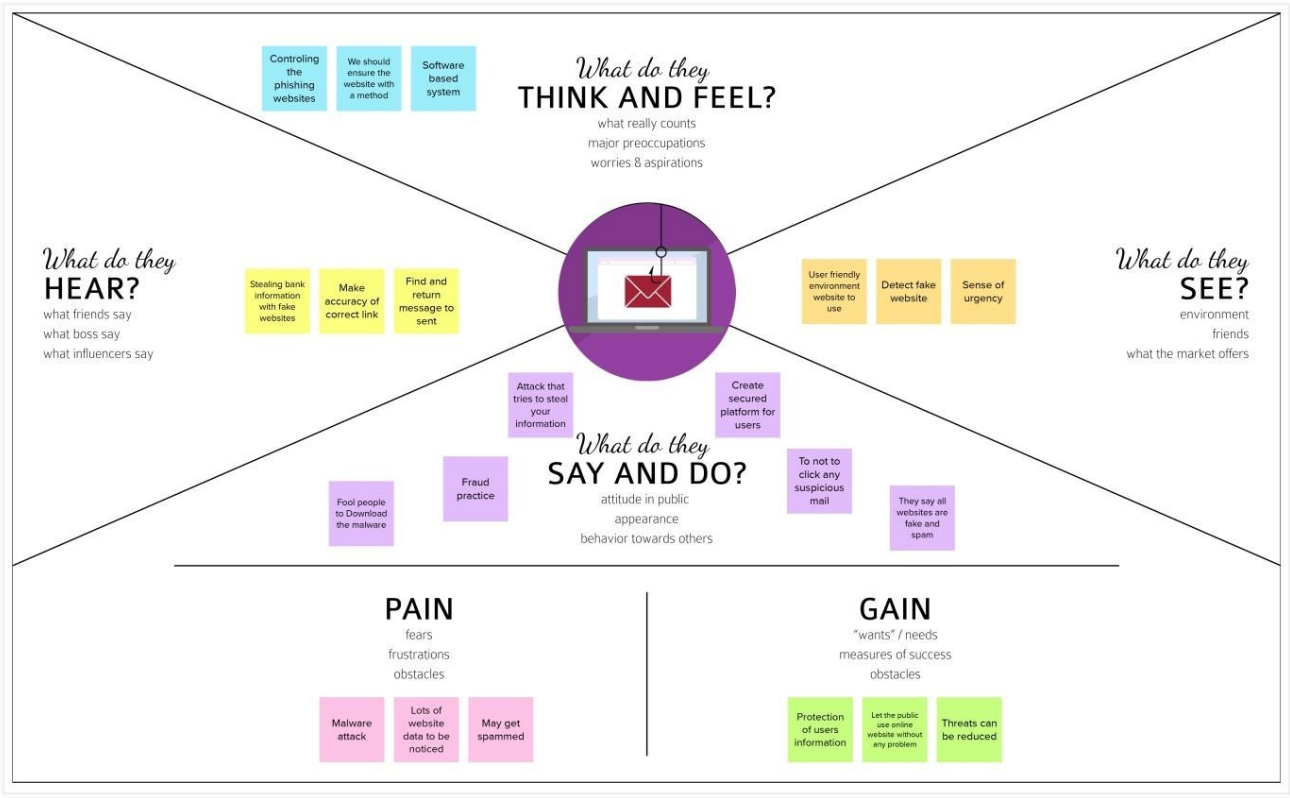
- It will lead to information disclosure and propertydamage.
- Large organizations may get trapped in different kinds ofscams.

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

CHAPTER - 3
Ideation & Proposed Solution

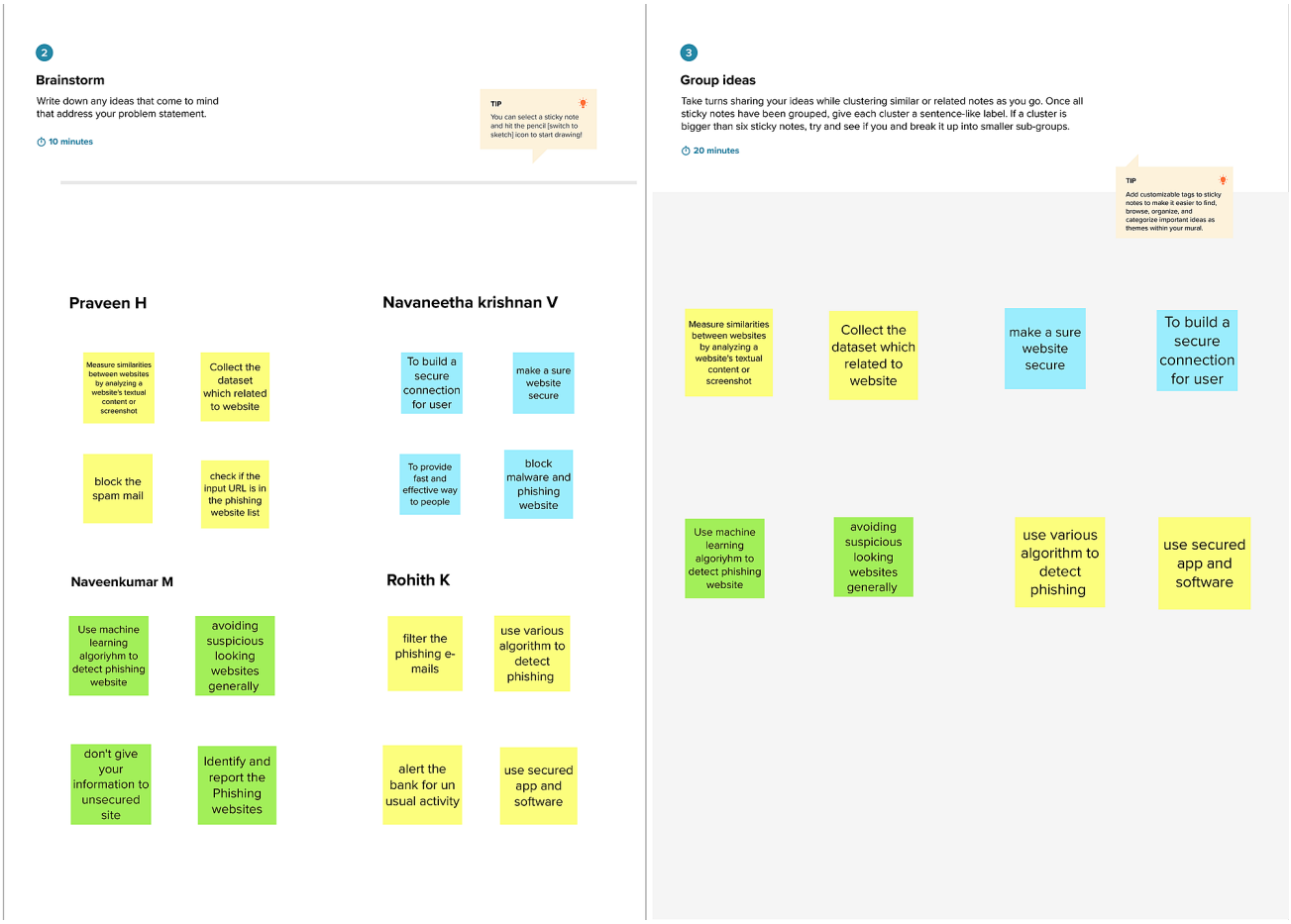
3.1 Empathy Map Canvas

WEB PHISHING DETECTION



1.

3.2 Ideation & Brainstorming



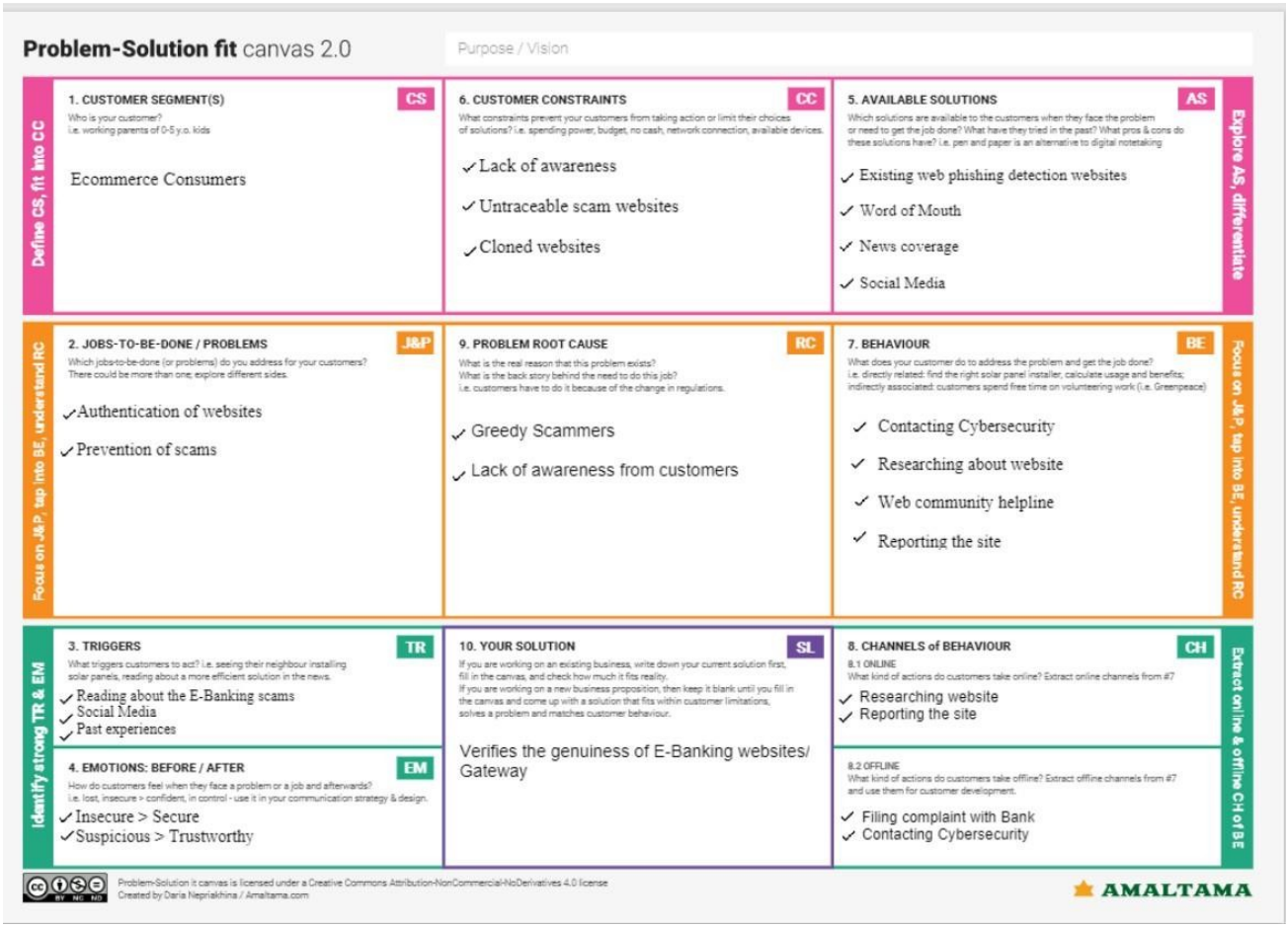
1.

3.3 Proposed Solution

S.No.	Parameter	Description
1.	Problem Statement (Problem to be solved)	Phishing sites are harmful websites that mimic trustworthy websites or web pages in an effort to steal users' personal information, including their user's name, password, and credit card number. Because phishing is mostly a semantics-based assault that focuses on human vulnerabilities rather than network or software flaws, identifying these phishing websites can be difficult.
2.	Idea / Solution description	A deep learning-based framework by implementing it as a browser plug-in capable of determining whether there is a phishing risk in real-time when the user visits a web page and gives a warning message. The real-time prediction includes whitelist filtering, blacklist interception, and machine learning (ML) prediction.
3.	Novelty / Uniqueness	Feel protected by using the website as the business-related credentials will be safe. Parents can be relaxed when kids explore educational website as the fraudulent website will be detected by our website
4.	Social Impact / Customer Satisfaction	The customer will come to know whether their details are safe/ not and the customer will be restricted from entering into the phishing websites.
5.	Business Model (Revenue Model)	Visitors engage with their ads, by generating impressions, engagements or clicks.
6.	Scalability of the Solution	Cost-effective and time-saving for global users residing at global locations.

1.

3.4 Problem Solution Fit



1.

CHAPTER - 4
Requirement Analysis

4.1 Functional Requirement

FR No.	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
FR-1	User Registration	Register by entering details such as name, email, Password.
FR-2	User Login	Login using the registered email id and password.
FR-3	Website Comparison	Blacklist filtering and Whitelist filtering techniques are used to compare the website URL.
FR-4	Feature Selection	Based on the length of an URL, number of dots in URL and check for the correct spelling and grammar.
FR-5	Feature Vectorization	Training and Testing dataset should be developed.
FR-6	Classifier	Model sends all output 10 classifier and produces final result.
FR-7	Results	Model then displays whether website is a legal site or a phishing site.

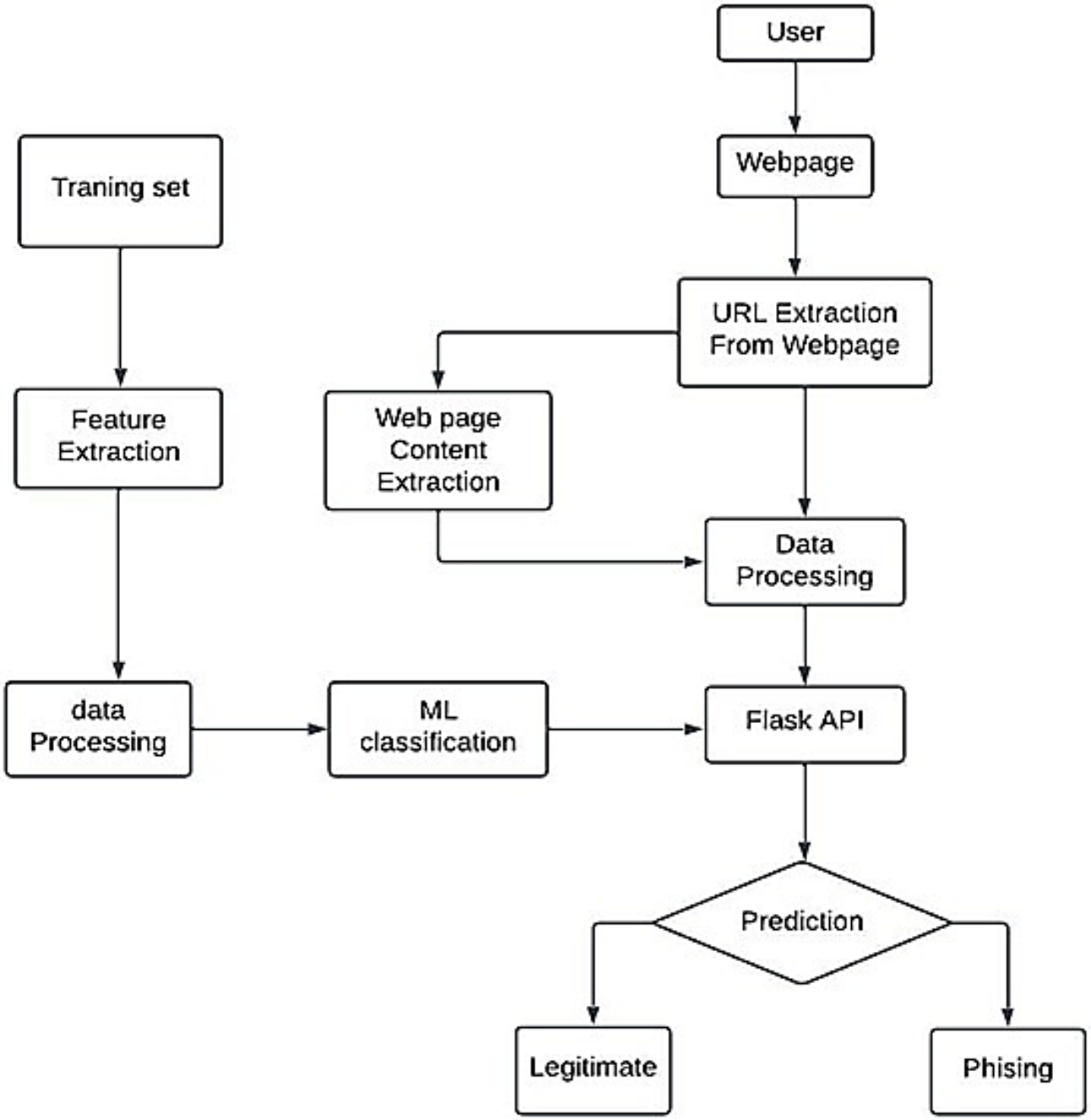
4.2 Non-functional Requirements

FR No.	Non-Functional Requirement	Description
NFR-1	**Usability**	User can access to several website easily using web phishing detection without losing any data.
NFR-2	**Security**	Alert message must be sent to the users to enable secure browsing.
NFR-3	**Reliability**	The web phishing websites must detect accurately and the result must be reliable.
NFR-4	**Performance**	The performance should be faster and user friendly for the effective performance.
NFR-5	**Availability**	The system will be accessible to the user at any point in time through a web browser.
NFR-6	**Scalability**	It must be able to handle an increase in users and loads without disrupting the end users.

1.

CHAPTER - 5
Project Design

5.1 Data Flow Diagrams

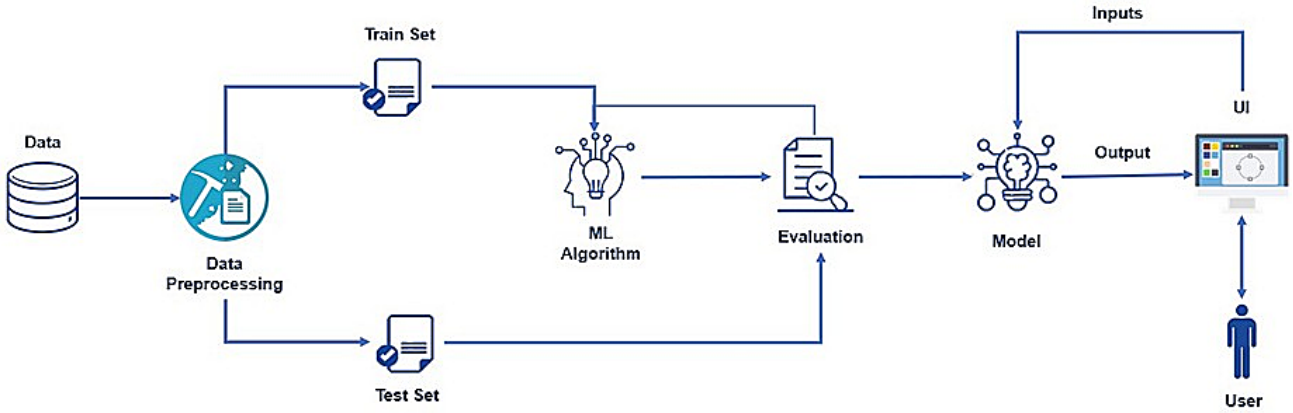


1.

5.2 Solution & Technical Architecture

Solution architecture is a complex process – with many sub-processes –that bridges the gap between business problems and technology solutions. Its goals are to:

1. Find the best tech solution to solve existing businessproblems.
2. Describethestructure,characteristics,behaviour,andotherasp ectsofthe software to projectstakeholders.
3. Define features, development phases, and solutionrequirements.
4. Provide specifications according to which the solution is defined,managed, anddelivered.



1.

5.3 User Stories

User Type	Function al Requirem ent(Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Relea se
Customer (Web user)	Registrati on	USN-1	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
	Login	USN-2	As a user, I can log into the application by entering email & password.	I can access the website	High	Sprint-1
	Website	USN-3	As a user, I enter a website to check whether the URL is safe to enter or not.	Website should be user Friendly	High	Sprint-1
	Notificati on	USN-4	If the Link is Malicious, Notification has to be sent to me.	I can receive a Notification	Medium	Sprint-2
	Dashboard	USN-5	As a user, I can see the Result	I can view that it is a Safe site or not	High	Sprint-2
Customer Care Executi ve	Help	USN-6	As a user, I can share my Queries in the Help Textbox	I can send my Queries through it	Medium	Sprint-3
Administrat or	Contact	USN-7	As a administrator, I can Answer the User Queries	I sent the Solution through User provided Email	Low	Sprint-3
		USN-8	As a Administrator, I can Improve the Accuracy	I can update the Website	High	Sprint-4

1.

CHAPTER - 6

Project Planning & Scheduling

6.1 Sprint Planning & Estimation

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-1	Registration	USN-1	a user, I can register for the application by Entering my email, password, and confirming My password.	5	High	Praveen H
Sprint-1	Login	USN-2	As a user, I can log into the application by entering email & password.	10	High	Rohith K A
Sprint-1	Website	USN-3	As a user, I enter a website to check whether the URL is safe to enter or not.	15	High	Praveen, Rohith
Sprint-2	Notification	USN-4	If the Link is Malicious, Notification has to be sent to me.	5	Medium	Navaneethan, Naveenkumar
Sprint-2	Dashboard	USN-5	As a user, I can see the Result	15	High	Rohith
Sprint-3	Help	USN-6	As a user, I can share my Queries in the Help Textbox	10	Medium	Navaneethan
Sprint-3	Contact	USN-7	As a administrator, I can Answer the User Queries	5	Low	Praveen H
Sprint-4		USN-8	As a Administrator, I can Improve the Accuracy	10	High	Navaneethan, Naveenkumar

Velocity:

Imagine we have a 6-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 = (Sprint Duration /Velocity)

= 20 /10

AV = 2

1.

6.2 Sprint Delivery Schedule

Sprint	Total Story Points	Duration	Sprint Start Date	Sprint End Date (Planned)	Story Points Completed (as on Planned End Date)
Sprint-1	20	6 Days	24 Oct 2022	29 Oct 2022	20
Sprint-2	20	6 Days	31 Oct 2022	05 Nov 2022	20
Sprint-3	20	6 Days	07 Nov 2022	12 Nov 2022	20
Sprint-4	20	6 Days	14 Nov 2022	19 Nov 2022	20

1.

CHAPTER - 7

Coding & Solutioning

App.py

```
import numpy as np
from flask import Flask, request, jsonify, render_template , redirect
import pickle
import inputScript

#   importing the inputScript file used to analyze the URL


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

#   Redirects to Webpage
@app.route('/')
def predict():
    return render_template("index.html")


#  fetches given URL and passes to inputScript
@app.route('/predict', methods=["POST"])
def y_predict():
    url = request.form['url']
    checkpredition =inputScript.main(url)
    print(checkpredition)
    prediction = model.predict(checkpredition)

    print(prediction)
    result = prediction[0]
    print(result)

    if (prediction == 1):
        pred = "you are safe!! This is a Legimate Website "
    elif (prediction == 0):
        pred = "phishing detected! You are on a worng site. Be cautious!"

    return render_template("index.html", pred_text='{}'.format(pred), url=url)
```

```
#   Takes input parameters from URL by inputScript and returns the predictions
@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(host='0.0.0.0', debug=True)
```

Index.html

```
<!DOCTYPE html>
<html lang="en">
    <head>
        <title> Web Phishing Detection</title>
        <meta charset="utf-8">
        <meta name="viewport" content="width=device-width, initial-scale=1">
        <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/css/bootstrap.min.css">
        <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-
awesome/4.7.0/css/font-awesome.min.css">
        <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.6.0/jquery.min.js"></script>
        <script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/js/bootstrap.min.js"></script>
        <style>
            body{
                margin: 0;
                padding: 0;
                font-family:Arial, Helvetica, sans-serif
            }.center {
                text-align: center;
            }
            nav{
                position:relative;
                top: 0;
                left: 0;
                width: 100%;
                height: 70px;
                padding: 10px 100px;
                box-sizing:border-box;
                background:#161616;
           }
            nav .logo{
```

1.

```
        padding: 15px;
        height: 30px;
        float: left;
        font-size: 25px;
        font-weight: bold;
        color: #fff;
    }
    nav ul {
        list-style:none;
        float: right;
        margin: 0;
        padding: 0;
        display: flex;
        font-size: 25px;
    }
    nav ul li a{
        float: right;
        display: block;
        color: #f2f2f2;
        text-align: center;
        padding: 15px;
        text-decoration: none;
        font-size: 22px;

    }
    nav ul li a:hover{
        background: rgb(200, 212, 200);
        border-radius: 6px;
        color: rgb(70, 27, 13);
    }
    nav ul li a.active{
        background: #e2472f;
        border-radius: 6px;

    }
    .end {
        overflow: hidden;
        background-color: rgb(63, 63, 63);
        position: fixed;
        bottom: 0;
        height: 55px;
        width: 100%;
    }
    .continer {
```

1.

```
            align-self:auto;
          }
          .button1{
  appearance: button;
  background-color: transparent;
  background-image: linear-gradient(to bottom, rgb(160, 245, 174), #37ee65);
  border: 0 solid #e5e7eb;
  border-radius: .5rem;
  box-sizing: border-box;
  color: #482307;
  column-gap: 1rem;
  cursor: pointer;
  display: flex;
  font-family: ui-sans-serif,system-ui,-apple-system,system-ui,"Segoe UI",Roboto,"Helvetica
Neue",Arial,"Noto Sans",sans-serif,"Apple Color Emoji","Segoe UI Emoji","Segoe UI Symbol","Noto
Color Emoji";
  font-size: 100%;
  font-weight: 700;
  line-height: 24px;
  margin: 0;
  outline: 2px solid transparent;
  padding: 1rem 1.5rem;
  text-align: center;
  text-transform: none;
  transition: all .1s cubic-bezier(.4, 0, .2, 1);
  user-select: none;
  -webkit-user-select: none;
  touch-action: manipulation;
  box-shadow: -6px 8px 10px rgba(81,41,10,0.1),0px 2px 2px rgba(81,41,10,0.2);
  display: none;
}
.button2{
  appearance: button;
  background-color: transparent;
  background-image: linear-gradient(to bottom, rgb(252, 162, 162), #ee3737);
  border: 0 solid #e5e7eb;
  border-radius: .5rem;
  box-sizing: border-box;
  color: #482307;
  column-gap: 1rem;
  cursor: pointer;
  display: flex;
  font-family: ui-sans-serif,system-ui,-apple-system,system-ui,"Segoe UI",Roboto,"Helvetica
Neue",Arial,"Noto Sans",sans-serif,"Apple Color Emoji","Segoe UI Emoji","Segoe UI Symbol","Noto
```

1.

```
Color Emoji";
  font-size: 100%;
  font-weight: 700;
  line-height: 24px;
  margin: 0;
  outline: 2px solid transparent;
  padding: 1rem 1.5rem;
  text-align: center;
  text-transform: none;
  transition: all .1s cubic-bezier(.4, 0, .2, 1);
  user-select: none;
  -webkit-user-select: none;
  touch-action: manipulation;
  box-shadow: -6px 8px 10px rgba(81,41,10,0.1),0px 2px 2px rgba(81,41,10,0.2);
  display: none;
}

  </style>
    </head>
    <body style="background-image: linear-gradient(to right,#c6ffdd, #fbd786, #f7797d);">
        <div class="center">
            <h2 style="font-family:'Franklin Gothic Medium', 'Arial Narrow', Arial, sans-serif;color: rgb(39,
41, 40);">Web Phishing Detection</h2><br>
<form action="/predict" method="post">
    <label for="url">Enter The URL:</label>
    <input type="text" placeholder="Enter the Suspicious url link" name="url">
    <br><br>
    <button type="submit" >submit</button>
  </form>
    <br>
    <a href="{{url}}">{{url}}</a>
  <br>
  <h4 >{{ pred_text }}</h4>
</div>
</body>
</html>
```

Input Script.py

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

1.

```
import whois
import datetime
import re
import requests
def url_having_ip(url):
    if len(url) < 54:
            return 1
    if len(url) >= 54 and len(url) <= 75:
            return 0
    return -1
def url_length(url):
    length=len(url)
    if(length<54):
        return -1
    elif(54<=length<=75):
        return 0
    else:
        return 1
def url_short(url):
    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',
                    url)
    if match:
        return -1
    return 1
def having_at_symbol(url):
    symbol=regex.findall(r'@',url)
    if(len(symbol)==0):
```

1.

```
        return -1
    else:
        return 1

def doubleSlash(url):
    if url.rfind('//') > 6:
        return -1
    return 1

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

def sub_domain(url):
    subDomain, domain, suffix = extract(url)
    if(subDomain.count('.')==0):
        return -1
    elif(subDomain.count('.')==1):
        return 0
    else:
        return 1
def SSLfinal_State(url):
    try:
#check wheather contains https
        if(regex.search('^https',url)):
            usehttps = 1
        else:
            usehttps = 0
#getting the certificate issuer to later compare with trusted issuer
        #getting host name
        subDomain, domain, suffix = extract(url)
        host_name = domain + "." + suffix
        context = ssl.create_default_context()
        sct = context.wrap_socket(socket.socket(), server_hostname = host_name)
        sct.connect((host_name, 443))
        certificate = sct.getpeercert()
        issuer = dict(x[0] for x in certificate['issuer'])
```

1.

```
        certificate_Auth = str(issuer['commonName'])
        certificate_Auth = certificate_Auth.split()
        if(certificate_Auth[0] == "Network" or certificate_Auth == "Deutsche"):
            certificate_Auth = certificate_Auth[0] + " " + certificate_Auth[1]
        else:
            certificate_Auth = certificate_Auth[0]
        trusted_Auth =
['Comodo','Symantec','GoDaddy','GlobalSign','DigiCert','StartCom','Entrust','Verizon','Trustwave','
Unizeto','Buypass','QuoVadis','Deutsche Telekom','Network
Solutions','SwissSign','IdenTrust','Secom','TWCA','GeoTrust','Thawte','Doster','VeriSign']
#getting age of certificate
        startingDate = str(certificate['notBefore'])
        endingDate = str(certificate['notAfter'])
        startingYear = int(startingDate.split()[3])
        endingYear = int(endingDate.split()[3])
        Age_of_certificate = endingYear-startingYear
#checking final conditions
        if((usehttps==1) and (certificate_Auth in trusted_Auth) and (Age_of_certificate>=1) ):
            return -1 #legitimate
        elif((usehttps==1) and (certificate_Auth not in trusted_Auth)):
            return 0 #suspicious
        else:
            return 1 #phishing
    except Exception as e:
        return 1
def domain_registration(url):
    try:
        w = whois.whois(url)
        updated = w.updated_date
        exp = w.expiration_date
        length = (exp[0]-updated[0]).days
        if(length<=365):
            return 1
        else:
            return -1
    except:
        return 0

def favicon(url):
    return 0
```

1.

```
def port(url):
    return 0

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

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

        linked_to_same = 0
        avg =0
        for image in imgs:
            subDomain, domain, suffix = extract(image['src'])
            imageDomain = domain
            if(websiteDomain==imageDomain or imageDomain==''):
                linked_to_same = linked_to_same + 1
        vids = soup.findAll('video', src=True)
        total = total + len(vids)

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

        if(avg<0.22):
```

1.

```
            return -1
        elif(0.22<=avg<=0.61):
            return 0
        else:
            return 1
    except:
        return 0
def url_of_anchor(url):
    try:
        subDomain, domain, suffix = extract(url)
        websiteDomain = domain

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

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

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

1.

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

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

def sfh(url):
    return 0

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

1.

```
        return 0

def abnormal_url(url):
    #ongoing
    return 0

def redirect(url):
    #ongoing
    return 0

def on_mouseover(url):
    #ongoing
    return 0

def rightClick(url):
    #ongoing
    return 0

def popup(url):
    #ongoing
    return 0

def iframe(url):
    #ongoing
    return 0

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

1.

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

def web_traffic(url):
    #ongoing
    return 0

def page_rank(url):
    #ongoing
    return 0

def google_index(url):
    #ongoing
    return 0


def links_pointing(url):
    return 0

def statistical(url):
    #ongoing
    return 0

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


    # print(check)
    return check
```

1.

CHAPTER - 8
Testing

8.1 Test Cases

TEST CASE ID	TESTCASE/ACTION TO BE PERFORMED	EXPECTED RESULT	ACTUAL RESULT	PASS/ FAIL
1	Clicking the “Scan Now” Button	Display Scan Page	Display Scan Page	Pass
2	Entering thr URL to verified	URL	URL	Pass
3	Clicking the “Predict” Button	Your are safe!! This is a Legitimate Website	Your are safe!! This is a Legitimate Website	Pass
4	Clicking the “Predict” Button	Your are Not safe!! This is a NotLegitimate Website	Your are Not safe!! This is a NotLegitimate Website	Pass
5	Clicking the “Home” Button	Returns to Home Page	Returns to Home Page	Pass

8.2 User Acceptance Testing

Purpose of Document

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

Defect Analysis

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

1.

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

1.

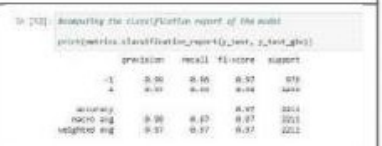
CHAPTER - 9

RESULT

9.1 Performance Metrics

Model Performance Testing:

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

S.No.	Parameter	Values	Screenshot
1.	Metrics	**Classification Model:** **Gradient Boosting Classification** Accuray Score- 97.4%	
2.	Tune the Model	Hyperparameter Tuning - 97% Validation Method – KFOLD & Cross Validation Method	

1. METRICS:
CLASSIFICATION REPORT:

```
In [52]: #computing the classification report of the model

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

```
              precision    recall  f1-score   support

          -1       0.99      0.96      0.97       976
           1       0.97      0.99      0.98      1235

    accuracy                           0.97      2211
   macro avg       0.98      0.97      0.97      2211
weighted avg       0.97      0.97      0.97      2211
```

CHAPTER - 10

Advantages & Disadvantages

Detection Technique	Advantages	Disadvantages
Blacklists	-Requiring lowresources on host machine -Effective when minimal FP rates are required.	-Mitigation of zero-hour phishing attacks. -Can result inexcessive queries with heavily loaded servers.
Heuristics and visual similarity	-Mitigate zerohour attacks.	-Higher FP rate than blacklists. -High computatio- nal cost.
Machine Learning	-Mitigate zerohour attacks. -Constuct own classification models.	-Time consuming. -Costly. -Huge number of rules.

CHAPTER - 11

Conclusion

The most important way to protect the user from phishing attack is the education awareness. Internet users must be aware of all security tips which are given by experts. Every user should also be trained not to blindly follow the links to websites where they have to enter their sensitive information. It is essential to check the URL before entering the website. In Future System can upgrade to automatic Detect the web page and the compatibility of the Application with the web browser. Additional work also can be done by adding some other characteristics to distinguishing the fake web pages from the legitimate web pages. PhishChecker application also can be upgraded into the web phone application in detecting phishing on the mobile platform. There are many features that can be improved in the work, for various other issues. The heuristics can be further developed to detect phishing attacks in the presence of embedded objects like flash. Identity extraction is an important operation and it was improved with the Optical CharacterRecognition (OCR) system to extract the text and images. More effective inferring rules for identifying a given suspicious web page, and strategies for discovering if it is a phishing target, should be designed in order to further improve the overall performance of this system. Moreover, it is an open challenge to develop a robust malware detection method, retaining accuracy for future phishing emails. In addition, the dynamic and static features complement each other, and therefore both are considered important in achieving highaccuracy.

CHAPTER - 12

Future Scope

In future if we get structured dataset of phishing we can perform phishing detection much more faster than any other technique.In future we can use a combination of any other two or more classifier to get maximum accuracy. We also plan to explore various phishing techniques that uses Lexical features, Network based features,Content based features, Webpage based features and HTML and JavaScript features of web pages which can improve the performance of the system. In particular, we extract features from URLs and pass it through the various classifiers.

It is found that phishing attacks is very crucial and it is important for us to get a mechanism to detect it. As very important and personal information of the user can be leaked through phishing websites, it becomes more critical to take care of this issue. This problem can be easily solved by using any of the machine learning algorithm with the classifier. We already have classifiers which gives good prediction rate of the phishing beside, but after our survey that it will be better to use a hybrid approach for the prediction and further improve the accuracy prediction rate of phishing websites. We have seen that existing system gives less accuracy so we proposed a new phishing method that employs URL based features and also we generated classifiers through several machinelearning.

CHAPTER - 13

Appendix

1. Application Building

2. Collection of Dataset

3. Data Pre-processing

4. Integration of Flask App with IBM Cloud

5. ModelBuilding 6. Performance Testing

7. Training the model on IBM 8. User Acceptance Testing

9. Ideation Phase 10. Preparation Phase

11. Project Planning

12. Performance Testing

13. User Acceptance Testing

SOURCE CODE

App.py

```
import numpy as np
from flask import Flask, request, jsonify, render_template , redirect
import pickle
import inputScript
#   importing the inputScript file used to analyze the URL
#   load model
app = Flask(__name__)
model = pickle.load(open('Phishing_Website.pkl','rb'))
#   Redirects to Webpage
@app.route('/')
def predict():
    return render_template("index.html")
#  fetches given URL and passes to inputScript
@app.route('/predict', methods=["POST"])
def y_predict():
    url = request.form['url']
    checkpredition =inputScript.main(url)
```

```
    print(checkpredition)
    prediction = model.predict(checkpredition)
    print(prediction)
    result = prediction[0]
    print(result)
    if (prediction == 1):
        pred = "you are safe!! This is a Legimate Website "
    elif (prediction == 0):
        pred = "phishing detected! You are on a worng site. Be cautious!"
    return render_template("index.html", pred_text='{}'.format(pred), url=url)
#   Takes input parameters from URL by inputScript and returns the predictions
@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(host='0.0.0.0', debug=True)
```

Index.html

```
<!DOCTYPE html>
<html lang="en">
    <head>
        <title> Web Phishing Detection</title>
        <meta charset="utf-8">
        <meta name="viewport" content="width=device-width, initial-scale=1">
        <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/css/bootstrap.min.css">
        <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-
awesome/4.7.0/css/font-awesome.min.css">
        <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.6.0/jquery.min.js"></script>
        <script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/js/bootstrap.min.js"></script>
        <style>
            body{
                margin: 0;
                padding: 0;
                font-family:Arial, Helvetica, sans-serif
            }.center {
                text-align: center;
            }
            nav{
                position:relative;
```

1.

```
        top: 0;
        left: 0;
        width: 100%;
        height: 70px;
        padding: 10px 100px;
        box-sizing:border-box;
        background:#161616;
    }
     nav .logo{
        padding: 15px;
        height: 30px;
        float: left;
        font-size: 25px;
        font-weight: bold;
        color: #fff;
     }
     nav ul {
        list-style:none;
        float: right;
        margin: 0;
        padding: 0;
        display: flex;
        font-size: 25px;
     }
     nav ul li a{
        float: right;
        display: block;
        color: #f2f2f2;
        text-align: center;
        padding: 15px;
        text-decoration: none;
        font-size: 22px;

     }
     nav ul li a:hover{
        background: rgb(200, 212, 200);
        border-radius: 6px;
        color: rgb(70, 27, 13);
     }
     nav ul li a.active{
        background: #e2472f;
        border-radius: 6px;

     }
```

1.

```
        .end {
            overflow: hidden;
            background-color: rgb(63, 63, 63);
            position: fixed;
            bottom: 0;
            height: 55px;
            width: 100%;
        }
        .continer {
            align-self:auto;
        }
        .button1{
  appearance: button;
  background-color: transparent;
  background-image: linear-gradient(to bottom, rgb(160, 245, 174), #37ee65);
  border: 0 solid #e5e7eb;
  border-radius: .5rem;
  box-sizing: border-box;
  color: #482307;
  column-gap: 1rem;
  cursor: pointer;
  display: flex;
  font-family: ui-sans-serif,system-ui,-apple-system,system-ui,"Segoe UI",Roboto,"Helvetica
Neue",Arial,"Noto Sans",sans-serif,"Apple Color Emoji","Segoe UI Emoji","Segoe UI Symbol","Noto
Color Emoji";
  font-size: 100%;
  font-weight: 700;
  line-height: 24px;
  margin: 0;
  outline: 2px solid transparent;
  padding: 1rem 1.5rem;
  text-align: center;
  text-transform: none;
  transition: all .1s cubic-bezier(.4, 0, .2, 1);
  user-select: none;
  -webkit-user-select: none;
  touch-action: manipulation;
  box-shadow: -6px 8px 10px rgba(81,41,10,0.1),0px 2px 2px rgba(81,41,10,0.2);
  display: none;
}
.button2{
  appearance: button;
  background-color: transparent;
  background-image: linear-gradient(to bottom, rgb(252, 162, 162), #ee3737);
```

1.

```
  border: 0 solid #e5e7eb;
  border-radius: .5rem;
  box-sizing: border-box;
  color: #482307;
  column-gap: 1rem;
  cursor: pointer;
  display: flex;
  font-family: ui-sans-serif,system-ui,-apple-system,system-ui,"Segoe UI",Roboto,"Helvetica
Neue",Arial,"Noto Sans",sans-serif,"Apple Color Emoji","Segoe UI Emoji","Segoe UI Symbol","Noto
Color Emoji";
  font-size: 100%;
  font-weight: 700;
  line-height: 24px;
  margin: 0;
  outline: 2px solid transparent;
  padding: 1rem 1.5rem;
  text-align: center;
  text-transform: none;
  transition: all .1s cubic-bezier(.4, 0, .2, 1);
  user-select: none;
  -webkit-user-select: none;
  touch-action: manipulation;
  box-shadow: -6px 8px 10px rgba(81,41,10,0.1),0px 2px 2px rgba(81,41,10,0.2);
  display: none;
}
  </style>
    </head>
    <body style="background-image: linear-gradient(to right,#c6ffdd, #fbd786, #f7797d);">
       <div class="center">
          <h2 style="font-family:'Franklin Gothic Medium', 'Arial Narrow', Arial, sans-serif;color: rgb(39,
41, 40);">Web Phishing Detection</h2><br>
<form action="/predict" method="post">
    <label for="url">Enter The URL:</label>
    <input type="text" placeholder="Enter the Suspicious url link" name="url">
    <br><br>
    <button type="submit" >submit</button>
  </form>
    <br>
    <a href="{{url}}">{{url}}</a>
  <br>
  <h4 >{{ pred_text }}</h4>
</div>
</body>
</html>
```

1.

Input Script.py

```
import regex
from tldextract import extract
import ssl
import socket
from bs4 import BeautifulSoup
import urllib.request
import whois
import datetime
import re
import requests
def url_having_ip(url):
    if len(url) < 54:
            return 1
    if len(url) >= 54 and len(url) <= 75:
            return 0
    return -1
def url_length(url):
    length=len(url)
    if(length<54):
        return -1
    elif(54<=length<=75):
        return 0
    else:
        return 1
def url_short(url):
    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',
```

1.

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

def doubleSlash(url):
    if url.rfind('//') > 6:
        return -1
    return 1

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

def sub_domain(url):
    subDomain, domain, suffix = extract(url)
    if(subDomain.count('.')==0):
        return -1
    elif(subDomain.count('.')==1):
        return 0
    else:
        return 1
def SSLfinal_State(url):
    try:
#check wheather contains https
        if(regex.search('^https',url)):
            usehttps = 1
        else:
            usehttps = 0
#getting the certificate issuer to later compare with trusted issuer
        #getting host name
```

1.

```
        subDomain, domain, suffix = extract(url)
        host_name = domain + "." + suffix
        context = ssl.create_default_context()
        sct = context.wrap_socket(socket.socket(), server_hostname = host_name)
        sct.connect((host_name, 443))
        certificate = sct.getpeercert()
        issuer = dict(x[0] for x in certificate['issuer'])
        certificate_Auth = str(issuer['commonName'])
        certificate_Auth = certificate_Auth.split()
        if(certificate_Auth[0] == "Network" or certificate_Auth == "Deutsche"):
            certificate_Auth = certificate_Auth[0] + " " + certificate_Auth[1]
        else:
            certificate_Auth = certificate_Auth[0]
        trusted_Auth =
['Comodo','Symantec','GoDaddy','GlobalSign','DigiCert','StartCom','Entrust','Verizon','Trustwave','
Unizeto','Buypass','QuoVadis','Deutsche Telekom','Network
Solutions','SwissSign','IdenTrust','Secom','TWCA','GeoTrust','Thawte','Doster','VeriSign']
#getting age of certificate
        startingDate = str(certificate['notBefore'])
        endingDate = str(certificate['notAfter'])
        startingYear = int(startingDate.split()[3])
        endingYear = int(endingDate.split()[3])
        Age_of_certificate = endingYear-startingYear
#checking final conditions
        if((usehttps==1) and (certificate_Auth in trusted_Auth) and (Age_of_certificate>=1) ):
            return -1 #legitimate
        elif((usehttps==1) and (certificate_Auth not in trusted_Auth)):
            return 0 #suspicious
        else:
            return 1 #phishing
    except Exception as e:
        return 1
def domain_registration(url):
    try:
        w = whois.whois(url)
        updated = w.updated_date
        exp = w.expiration_date
        length = (exp[0]-updated[0]).days
        if(length<=365):
            return 1
```

1.

```
        else:
            return -1
    except:
        return 0

def favicon(url):
    return 0
def port(url):
    return 0

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

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

        linked_to_same = 0
        avg =0
        for image in imgs:
            subDomain, domain, suffix = extract(image['src'])
            imageDomain = domain
            if(websiteDomain==imageDomain or imageDomain==''):
                linked_to_same = linked_to_same + 1
        vids = soup.findAll('video', src=True)
        total = total + len(vids)

        for video in vids:
            subDomain, domain, suffix = extract(video['src'])
            vidDomain = domain
```

1.

```
            if(websiteDomain==vidDomain or vidDomain==''):
                linked_to_same = linked_to_same + 1
        linked_outside = total-linked_to_same
        if(total!=0):
            avg = linked_outside/total

        if(avg<0.22):
            return -1
        elif(0.22<=avg<=0.61):
            return 0
        else:
            return 1
    except:
        return 0
def url_of_anchor(url):
    try:
        subDomain, domain, suffix = extract(url)
        websiteDomain = domain

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

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

1.

```
    except:
        return 0

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

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

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

def email_submit(url):
    try:
        opener = urllib.request.urlopen(url).read()
```

1.

```
        soup = BeautifulSoup(opener, 'lxml')
        if(soup.find('mailto:')):
            return 1
        else:
            return -1
    except:
        return 0
def abnormal_url(url):
    #ongoing
    return 0
def redirect(url):
    #ongoing
    return 0
def on_mouseover(url):
    #ongoing
    return 0
def rightClick(url):
    #ongoing
    return 0
def popup(url):
    #ongoing
    return 0
def iframe(url):
    #ongoing
    return 0
def age_of_domain(url):
    try:
        w = whois.whois(url)
        start_date = w.creation_date
        current_date = datetime.datetime.now()
        age =(current_date-start_date[0]).days
        if(age>=180):
            return -1
        else:
            return 1
    except Exception as e:
        print(e)
        return 0
def dns(url):
    #ongoing
```

1.

```
    return 0
def web_traffic(url):
    #ongoing
    return 0
def page_rank(url):
    #ongoing
    return 0
def google_index(url):
    #ongoing
    return 0
def links_pointing(url):
    return 0
def statistical(url):
    #ongoing
    return 0
def main(url):
    check = [[url_having_ip(url),url_length(url),url_short(url),having_at_symbol(url),
            doubleSlash(url),prefix_suffix(url),sub_domain(url),SSLfinal_State(url),
             domain_registration(url),favicon(url),port(url),https_token(url),request_url(url),
             url_of_anchor(url),Links_in_tags(url),sfh(url),email_submit(url),abnormal_url(url),
             redirect(url),on_mouseover(url),rightClick(url),popup(url),iframe(url),
             age_of_domain(url),dns(url),web_traffic(url),page_rank(url),google_index(url),
             links_pointing(url),statistical(url)]]


    # print(check)

    return check
```

Github & Project Demo Link

GitHub link : https://github.com/IBM-EPBL/IBM-Project-545-1658306433

Project Demo Link : https://github.com/Praveen10052001/IBM-Project-545-1658306433/tree/main/Final%20Deliverables/Demo

1.