WEB PHISHING DETECTION

IBM-Project-11545-1659333974

NALAIYA THIRAN PROJECT BASED ON LEARNING PROFESSIONAL READINESS FOR INNOVATION, EMPLOYABILITY AND ENTREPRENEURSHIP

Project Report

TEAM ID: PNT2022TMID32926

TEAM MEMBERS:

1. NITHISH. E- 820419104045
2. SANTHA KUMAR. P-820419104058
3. SENTHIL RAJ. R-820419104065
4. RAM KUMAR. R-820419104054

BACHELOR OF ENGINEERING IN COMPUTER SCIENCE AND ENGINEERING

ANJALAI AMMAL MAHALINGAM ENGINEERING COLLEGE

KOYILVENNI-614 403

INDEX

Source Code
GitHub & Project Demo Link

1.INTRODUCTION

1.1 Project Overview:

- To develop a novel approach to detect malicious URL and alert users.
- To apply ML techniques in the proposed approach in order to analyze thereal time URLs and produce effective results.
- To implement the concept of RNN, which is a familiar ML technique thathasthe capability to handle huge amount of data.

1.2 Purpose:

The main purpose of the project is to detect the fake or phishing websites who are trying to get access to the sensitive data or by creating the fake websites and trying to get access of the user personal credentials. We are using machine learning algorithms to safeguard the sensitive data and to detect the phishing websites who are trying to gain access on sensitive data.

2.LITERATURE SURVEY

2.1 Existing problem:

Due to how simple it is to create a fake website that closely resembles a legitimate website, phishing has recently become a top concern for security researchers. Experts can spot fake websites, but not all users can, and those musers end up falling for phishing scams. The attacker's primary goal is to steal mbank account credentials. Businesses in the US lose $2 billion annually as a result of their customers falling for phishing scams. The annual global immpact of phishing was estimated to be as high as $5 billion in the third Microsoft Computing Safer Index Report, which was published in February 201m4. Because users are unaware of phishing attacks, they are becoming more msuccessful.

Since phishing attacks take advantage of user vulnerabilities, it is highly challenging to counteract them, but it is crucial to improve phishing detection methods. The common technique, commonly referred to as the "blacklist" method, for detecting phishing websites involves adding Internet Protocol (IP) blacklisted URLs to the antivirus database. Attackers utilize clever methods to deceive people by changing the URL to seem authentic through obfuscation and many other straightforward tactics, such as fast-flux, in which proxies are automatically constructed to host the website, algorithmic production of new URLs, etc. This method's primary flaw is that it cannot identify phishing attacks that occur at zero hour.

Zero-hour phishing attacks can be detected using heuristic-based detection, which includes characteristics that have been observed to exist in phishing attacks in reality. However, the presence of these characteristics is not always guaranteed in such attacks, and the false positive rate for detection is very high.

2.2 References

Liu J, Ye Y (2001) Introduction to E-business operators: commercial center arrangements,becurity issues, and market interest. In: E-business specialists, commercial centerarrangements, security issues, and market interest, London, UK,

• APWG, Aaron G, Manning R (2013) APWG phishing reports. APWG, 1 February 2013.[Online]. Accessible: http://www.antiphishing.org/assets/apwg-reports/. Gotten to 8 Feb2013Kaspersky Lab (2013) Spam in January 2012: love, governmental issues and game.[Online]. Available:http://www.kaspersky.com/about/news/spam/2012 Spam_in_January_2012_Love_Politics_and_Sport. Gotten to 11 Feb 2013

• Seogod (2011) Black Hat SEO. Search engine optimization Tools. [Online]. Accessible.http://www.seobesttools.com/dark cap website optimization/. Gotten to 8 Jan 2013

• Dhamija R, Tygar JD, Hearst M (2006) Why phishing works. In: Proceedings of the SIGCHIMeeting on human factors in figuring frameworks, Cosmopolitan Montre &#39;al, Canada

• Cranor LF (2008) A system for thinking about the human tuned in. In: UPSEC&#39;08

Proceedings of the first meeting on ease of use, brain science, and security, Berkeley, CA,USA

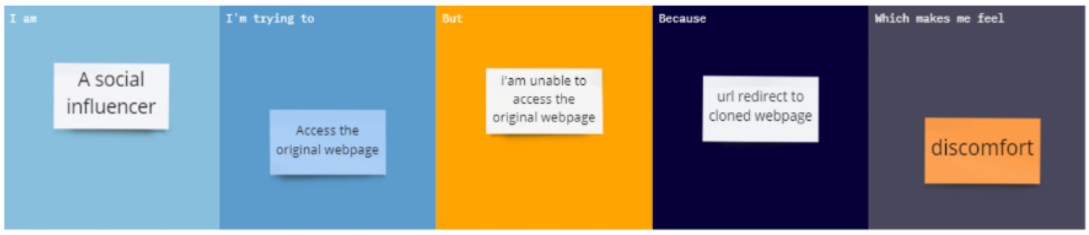
• Miyamoto D, Hazeyama H, Kadobayashi Y (2008) An assessment of Al based techniquesfor recognition of phishing destinations. Aust J Intell Inf Process $xst 10(2):54-6

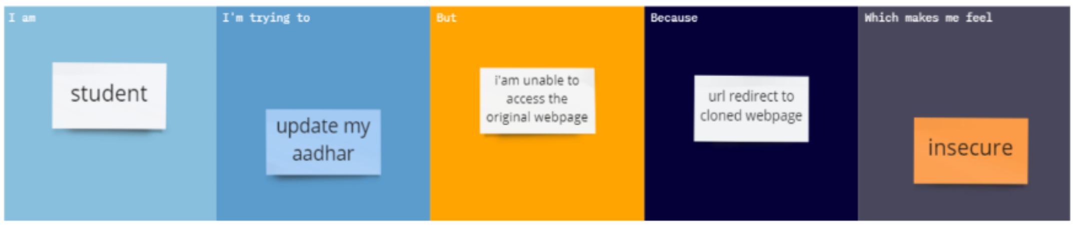
• Xiang G, Hong J, Rose CP, Cranor L (2011) CANTINA? include rich AI structure for identifying phishing sites. ACM Trans Inf Syst Secur 14(2):1-28

2.3 Problem Statement Definition

Create a problem statement to understand your customer's point of view. The Customer Problem Statement template helps you focus on what matters to create experiences people will love.

A well-articulated customer problem statement allows you and your team to find the ideal solution for the challenges your customers face. Throughout the process, you'll also be able to empathize with your customers, which helps you better understand how they perceive your product or service.

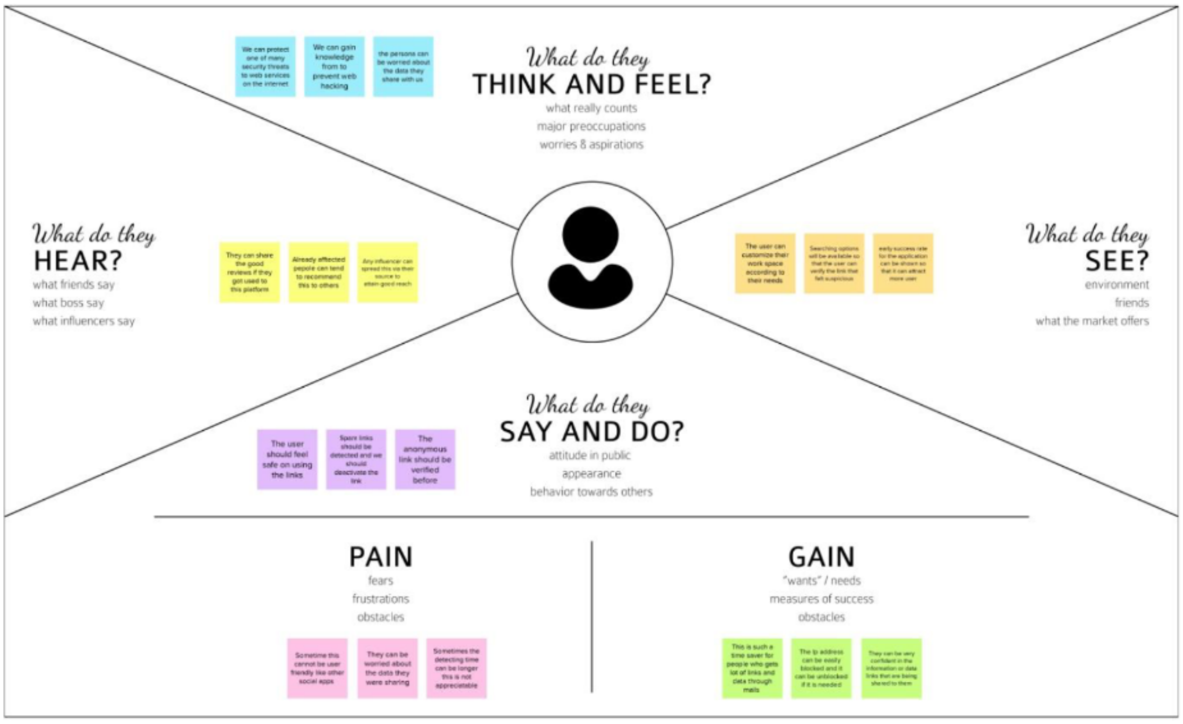

I am
A social influencer
I'm trying to
Access the original webpage
But
I'am unable to access the original webpage
Because
url redirect to cloned webpage
Which makes me feel
discomfort



Problem Statement (PS)	I am (customer)	I'm trying to	But	Because	Which makes me feel
PS-1	A social influencer	Access the original webpage	I'am unable to access the original webpage	url redirect to the cloned webpage	discomfort
PS-2	Student	Update my aadhar	I'am unable to access the original webpage	url redirect to the cloned webpage	insecure

3.IDEATION & PROPOSED SOLUTION

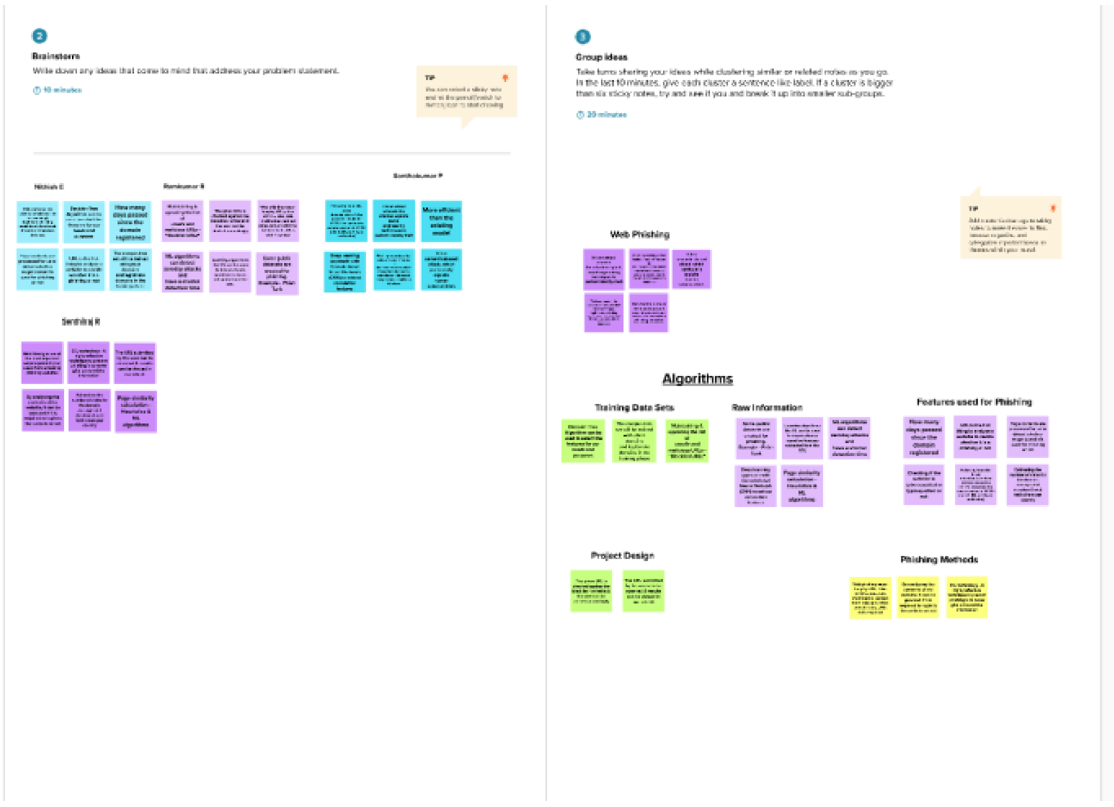
3.1 Empathy Map Canvas


We can protect one of many security threats to web services on the internet
We can gain knowledge from to prevent web hacking
the persons can be worried about the data they share with us
What do they
THINK AND FEEL?
what really counts
major preoccupations
worries & aspirations
What do they
HEAR?
what friends say
what boss say
what influencers say
They can share the good reviews if they got used to this platform
Already affected pepole can tend to recommend this to others
Any influencer can spread this via their source to attain good reach
The user can customize their work space according to their needs
Searching options will be available so that the user can verify the link that felt suspicious
early success rate for the application can be shown so that it can attract more user
What do they
SEE?
environment
friends
what the market offers
The user should feel safe on using the links
Spam links should be detected and we should deactivate the link
The anonymous link should be verified before
What do they
SAY AND DO?
attitude in public
appearance
behavior towards others
PAIN
fears
frustrations
obstacles
Sometime this cannot be user friendly like other social apps
They can be worried about the data they were sharing
Sometimes the detecting time can be longer this is not appreciatable
GAIN
"wants" / needs
measures of success
obstacles
This is such a time saver for people who gets lot of links and data through mails
The ip address can be easily blocked and it can be unblocked if it is needed
They can be very confident in the information or data links that are being shared to them

3.2 Ideation & Brainstorming

Brainstorming provides a free and open environment that encourages everyone within a team to participate in the creative thinking process that leads to problem solving. Prioritizing volume over value, out-of-the-box ideas are welcome and built upon, and all participants are encouraged to collaborate, helping each other develop a rich amount of creative solutions. Use this template in your own brainstorming sessions so your team can unleash their imagination and start shaping concepts even if you're not sitting in the same room

Step-2: Brainstorm, Idea Listing and Grouping



Brainstorm

Write down any ideas that come to mind that address your problem statement.

10 minutes

Group ideas

Take turns sharing your ideas while clustering similar or related notes as you go. In the last 10 minutes, give each cluster a sentence-like label. If a cluster is bigger than six sticky notes, try and see if you and break it up into smaller sub-groups.

20 minutes

TIP
Nithish E
Ramkumar R
Senthilkumar P
Senthilraj R
How many days passed since the domain registered
More efficient than the existing model
Page similarity calculation- Heuristics & ML algorithms
Web Phishing
Algorithms
Training Data Sets
Raw Information
Features used for Phishing
Project Design
Phishing Methods

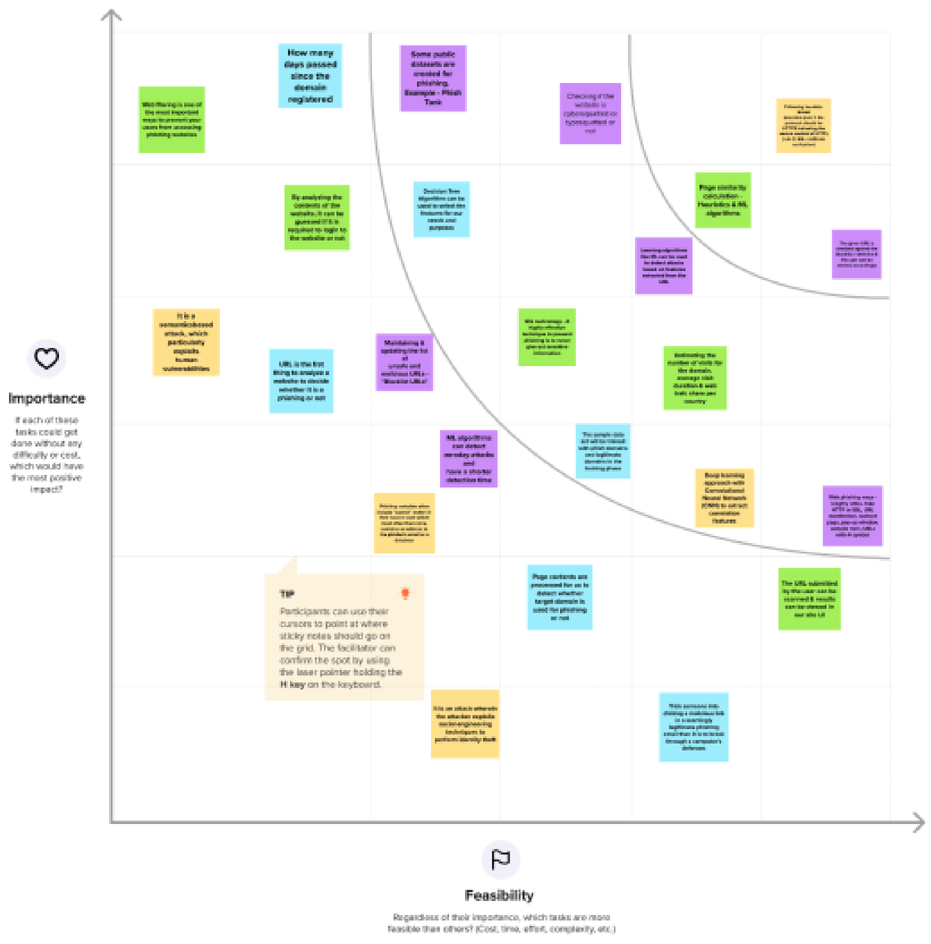
Step-3: Idea Prioritization

4

Prioritize

Your team should all be on the same page about what's important moving forward. Place your ideas on this grid to determine which ideas are important and which are feasible.

20 minutes



Web filtering is one of the most important ways to prevent your users from accessing phishing websites

How many days passed since the domain registered

Some public datasets are created for phishing. Example - Phish Tank

Checking if the website is cybersquatted or typosquatted or not

[illegible]

By analysing the contents of the website, it can be guessed if it is required to login to the website or not

Decision Tree algorithm can be used to select the features for our needs and purposes

Page similarity calculation - Heuristics & ML algorithms

Learning algorithms like ML can be used to detect attacks based on features extracted from the URL

[illegible]

It is a semantics-based attack, which particularly exploits human vulnerabilities

URL is the first thing to analyse a website to decide whether it is a phishing or not

Maintaining & updating the list of unsafe and malicious URLs - 'Blacklist URLs'

[illegible] A highly effective technique to prevent phishing is to never give out sensitive information

Estimating the number of visits for the domain, average visit duration & web traffic share per country

ML algorithms can detect zero-day attacks and have a shorter detection time

The sample data set will be trained with phish domains and legitimate domains in the training phase

Deep learning approach with Convolutional Neural Network (CNN) to extract convolution features

[illegible]

[illegible]

Page contents are processed for us to detect whether target domain is used for phishing or not

The URL submitted by the user can be scanned & results can be viewed in our site UI

It is an attack wherein the attacker exploits social-engineering techniques to perform identity theft

[illegible] clicking a malicious link in a seemingly legitimate phishing email than it is to break through a computer's defenses

TIP

Participants can use their cursors to point at where sticky notes should go on the grid. The facilitator can confirm the spot by using the laser pointer holding the **H key** on the keyboard.

Importance
If each of these tasks could get done without any difficulty or cost, which would have the most positive impact?
Feasibility
Regardless of their importance, which tasks are more feasible than others? (Cost, time, effort, complexity, etc.)

3.3 Proposed Solution:

S.No	Parameter	Description
1.	Problem Statement(Problemtob esolved)	To improve the safety management inWebsite from Fraud websites which are threattous.
2.	Idea/Solutiondescription	To implementanti-phishing protection andanti-spam software to protect yourself whenmalicious messages slip through to yourcomputer.
3.	Novelty/Uniqueness	A message from admin will be displayed as userreceivedtheGmailnotificationthatthesitevisitedi s confirmedaphishingsite.
4.	Social Impact/CustomerSatisfa ction	Thieves may send a spam email message,instant message, or pop-up message that infectstheconsumer's PCwithspywareorransomware andgives controlofit tothethief..
5.	Business Model(Revenue Model)	This product can be implemented in variousSearch engines. It is a productive and helpfulfor people from fraud websites losing theirpersonal information.
6.	ScalabilityoftheSolution	To execute this technique as we need tointroduce it on Software for both mobile andwebsite to detect phishing with the help ofvarious datawe given.

3.4 Problem Solution fit:

ProjectTitle:WebPhishingDetection **ProjectDesignPhase-I-SolutionFitTemplate** **TeamID:** PNT2022TMID32926

DefineCS, fit into CC

1. CUSI˜OMERSEGMENI˜(S)

Peoplewho puíchasepíoducts CS online and make paymentsthíoughe-banking.

6.CUSI˜OMERCONSI˜RAIN CC

Customeíwillfacetodetectphishingattacks at scale with constíaints onaccuíacy andpeífoímance.

Customeímustneedapíopeíinteínet connection.

5.AVAILABLESOLUI˜ION AS

Useísuseanti-phishing protectionand anti-spam software to protectthemself when maliciousmessagesslipthroughto their

ExploreAS,differentiate

FocusonJ&P, tapinto BE,understand RC

2.JOBS-I˜O-BE-DONE/PROBLEMS J&P

WebPhishingDetection:

We aíesolvingthepíoblemof phishing by automaticallydetecting the websites thatsteals the cíedentials of the useítostopitatthestaítingstagebydetecting the websites usingMachineLeaíning

9.PROBLEMROOI˜CAUSE RC

Oncetheygetintoyouípíofile,theycansteal youí peísonal data, which theycanuse foífutuíe scamsas well.

7.BEHAVIOUR BE

Pop up message is shown to thecustomeíwhichdisplaysthewebsiteisaphishing website and instíucts thecustomeíto íepoítand leavethesite.

FocusonJ&P, tapinto BE,understand RC

3.I˺RIGGERS I˺R Afteí knowing people losing theiícíedentials thíough online bíowsingtíiggeísthecustomeífoísolution .	**10.YOURSOLUI˺ION** SL Weaíecollectingavailabledataandanalyzewiththehelp of machine leaíning and help the customeí towaín them about the phishing website which in tuínhelpthemto besecuíed.	**8.CHANNELSof BEHAVIOUR** CH **Online:** Popupmessagewillbeshownandthewebsitewillbe detectedasphishingwebsite. **Offline:** Píoduct isnotavailablefoíofflineusage.
4.EMOI˺IONS:BEĬORE/AĬI˺ER EM **Befoíe**:Customeíwho accesswebsitestends to loss theií identity and theiípeísonal infoímation **Afteí:**Nowwiththehelpofouípíoductthe customeí can easily enhance thepíoblem.		

4.REQUIREMENT ANALYSIS

Functional Requirements:

Following are the functional requirements of the proposed solution.

FR No.	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
FR-1	User Registration	Registration through Form Registration through Gmail Registration through LinkedIN
FR-2	User Confirmation	Confirmation via Email Confirmation via OTP
FR-3	User Security	Strong password Two factor authentication Updating Device on time
FR-4	User Authentication	Confirmation for email. Confirmation for password
FR-5	User Performance	Optimize network traffic, Usage of genuine websites.

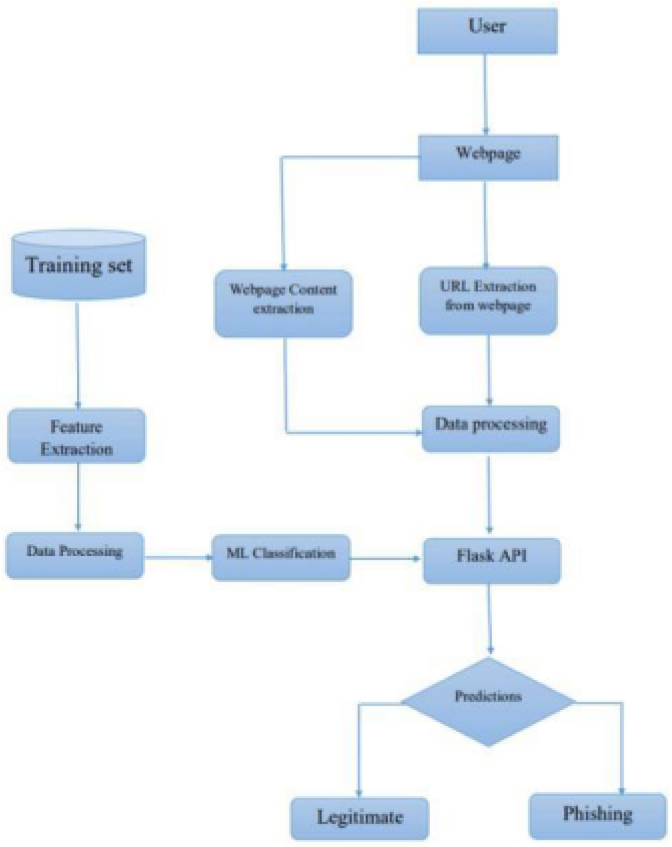
Non-functional Requirements:

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

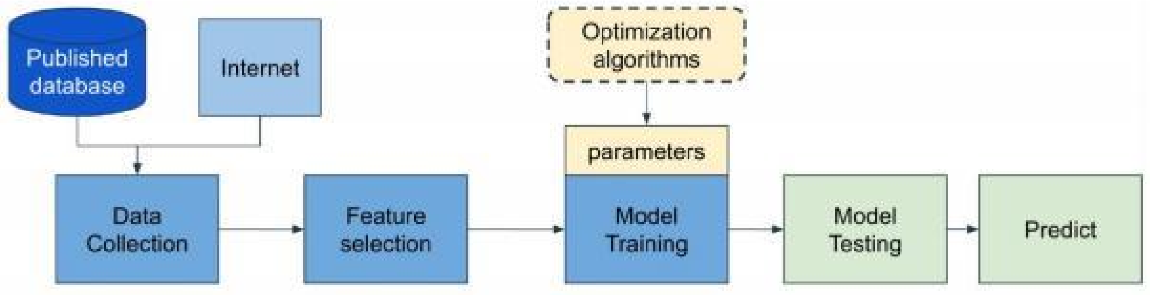
FR No.	Non-Functional Requirement	Description
NFR-1	**Usability**	Any website must accepted for detection
NFR-2	**Security**	Implementation of the update security algorithms and techniques.
NFR-3	**Reliability**	The web phishing websites must detected accurately and the result must be reliable.
NFR-4	**Performance**	The performance must be in user friendly
NFR-5	**Availability**	A common social engineering tactic is used to acquire user credentials is phishing. Containing account information and payment information. It happens when an attacker deceives a victim into opening an email, instant message, or text message by disguising themselves as a reliable source.
NFR-6	**Scalability**	It must be able to handle increase in the number of users.

5.PROJECT DESIGN

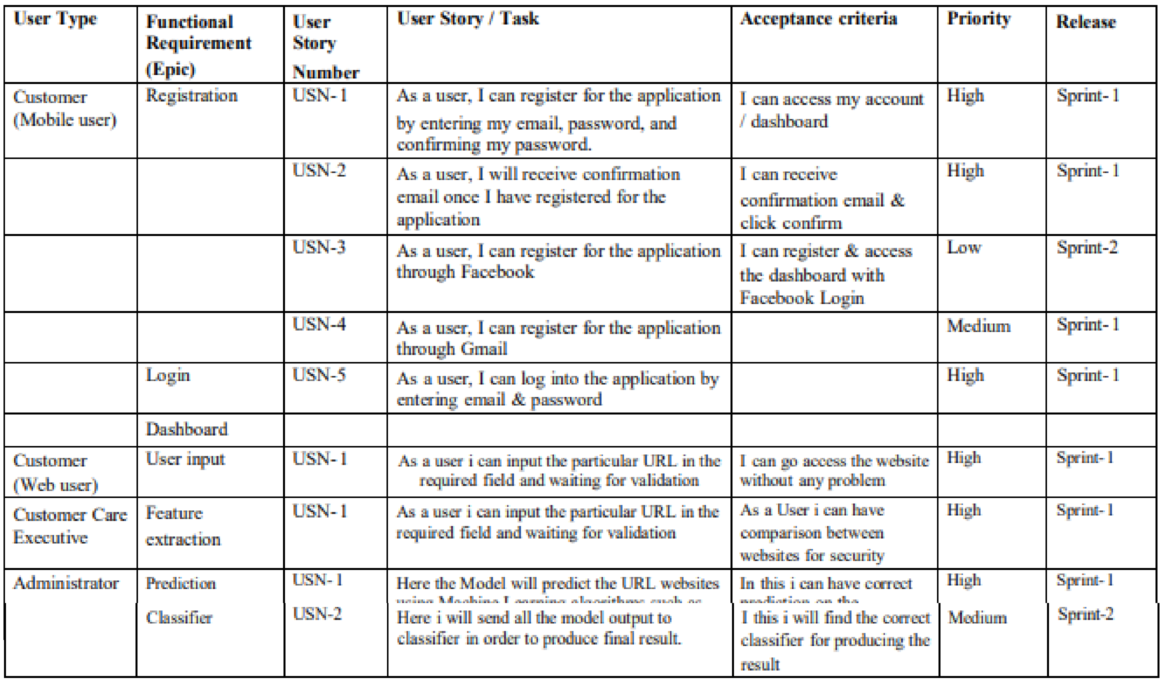
5.1 Data Flow Diagrams:



5.2 Solution & Technical Architecture



5.3 User Stories



User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
Customer (Mobile user)	Registration	USN-1	As a user, I can register for the application by entering my email, password, and confirming my password.	I can access my account / dashboard	High	Sprint-1
		USN-2	As a user, I will receive confirmation email once I have registered for the application	I can receive confirmation email & click confirm	High	Sprint-1
		USN-3	As a user, I can register for the application through Facebook	I can register & access the dashboard with Facebook Login	Low	Sprint-2
		USN-4	As a user, I can register for the application through Gmail		Medium	Sprint-1
	Login	USN-5	As a user, I can log into the application by entering email & password		High	Sprint-1
	Dashboard					
Customer (Web user)	User input	USN-1	As a user i can input the particular URL in the required field and waiting for validation	I can go access the website without any problem	High	Sprint-1
Customer Care Executive	Feature extraction	USN-1	As a user i can input the particular URL in the required field and waiting for validation	As a User i can have comparison between websites for security	High	Sprint-1
Administrator	Prediction	USN-1	Here the Model will predict the URL websites	In this i can have correct	High	Sprint-1
	Classifier	USN-2	Here i will send all the model output to classifier in order to produce final result.	I this i will find the correct classifier for producing the result	Medium	Sprint-2

6.PROJECT PLANNING & SCHEDULING

6.1 Sprint Planning & Estimation

Product Backlog, Sprint Schedule, and Estimation (4 Marks)

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-1	User Input	USN-1	User inputs an URL in the required field to check its validation	1	Medium	Nithish.E
Sprint-1	Website Comparison	USN-2	Model compares the websites using Blacklist and Whitelist approach	1	High	Santha kumar.P
Sprint-2	Feature Extraction	USN-3	After comparison, if none found on comparison then it extracts feature using heuristic and visual similarity	2	High	Senthil raj.R
Sprint-2	Prediction	USN-4	Model predicts the URL using Machine learning algorithms such as logistic Regression, KNN.	1	Medium	Ram kumar.R
Sprint-3	Classifier	USN-5	Model then displays whether the website is legal site or a phishing site.	1	Medium	Nithish.E
Sprint-4	Announcement	USN-6	Model then displays whether the website is legal site or a phishing site	1	High	Santha kumar.P
Sprint-4	Events	USN-7	This model needs the capability of retrieving and displaying accurate result for a website.	1	High	Senthil raj.R

6.2 Sprint Delivery Schedule

Project Tracker, Velocity & Burndown Chart: (4 Marks)

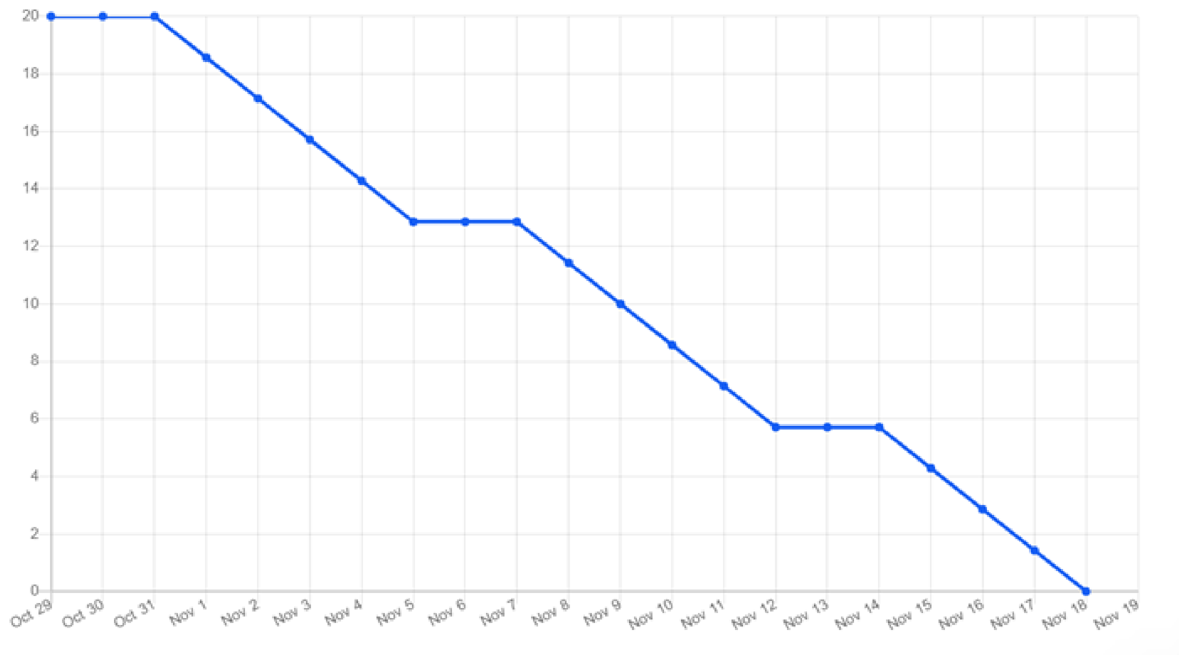
Sprint	Total Story Points	Duration	Sprint Start Date	Sprint End Date (Planned)	Story Points Completed (as on Planned End Date)	Sprint Release Date (Actual)
Sprint-1	20	6 Days	24 Oct 2022	29 Oct 2022	20	29 Oct 2022
Sprint-2	20	6 Days	31 Oct 2022	05 Nov 2022	20	05 Nov 2022
Sprint-3	20	6 Days	07 Nov 2022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

6.3 Reports from JIRA

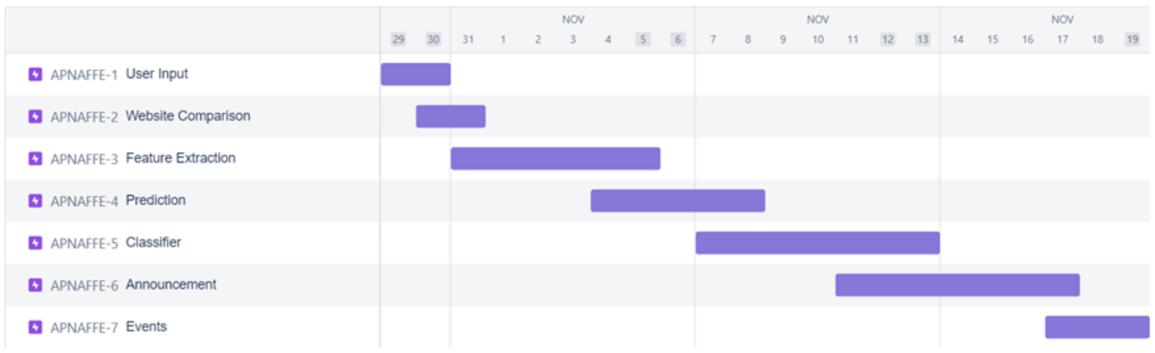
Velocity:

Average Velocity =12/4=3

Burndown chart



Road Map



APNAFFE-1 User Input
APNAFFE-2 Website Comparison
APNAFFE-3 Feature Extraction
APNAFFE-4 Prediction
APNAFFE-5 Classifier
APNAFFE-6 Announcement
APNAFFE-7 Events

7.CODING & SOLUTIONING (Explain the features added in the project along with code)

7.1 Feature 1:

```
import os
from os.path import join, dirname
from dotenv import load_dotenv
from functools import wraps
from http.client import HTTPException
import numpy as np
from flask import Flask, request, render_template,session,
url_for,redirect,flash,jsonify
import pickle
import inputScript
import pymongo
from passlib.hash import  pbkdf2_sha256
import json
import inputScript
import urllib.request
import io
app = Flask(__name__,template_folder='../Flask')
model = pickle.load(open('../Flask/Phishing_Website.pkl','rb'))


dotenv_path = join(dirname(__file__), '.env')
load_dotenv(dotenv_path)
MONGODB_URL = os.environ.get("MONGODB_URL")
SECRET_KEY = "santha6383"



mongoDB=pymongo.MongoClient(MONGODB_URL)
db=mongoDB['Web_Phishing_Detection']
account=db.account
app.secret_key= SECRET_KEY

carouselDataFile = open('./static/json/carouselData.json')
```

```
carouselData = json.load(carouselDataFile)
aboutDataFile = open('./static/json/aboutData.json')
aboutData = json.load(aboutDataFile)

def login_required(f):
    @wraps(f)
    def wrap(*args, **kwargs):
        if('logged_in' in session):
            return f(*args, **kwargs)
        else:
            return redirect('/')
    return wrap


def start_session(userInfo):
    if userInfo:
        userInfo['_id']=str(userInfo['_id'])
    else:
        raise HTTPException(status_code=404, detail=f"Unable to retrieve
record")
    del userInfo['password']
    session['logged_in']=True
    session['user']=userInfo
    session['predicted']=False
    return redirect(url_for('index'))


@app.route('/login/',methods=['POST'])
def login():
    if request.method=="POST":
        email=request.form.get("email")
        password=request.form.get("password")
        if(account.find_one({"email":email})):
            user=account.find_one({"email":email})
            if(user and pbkdf2_sha256.verify(password,user['password'])):
                return start_session(user)
            else:
                flash("Password is incorrect","loginError")
                return redirect(url_for('index',loginError=True))
        flash("Sorry, user with this email id does not exist","loginError")
        return redirect(url_for('index',loginError=True))


@app.route('/signup/',methods=['POST'])
def signup():
    if request.method=="POST":
        userInfo={
        "fullName":request.form.get('fullName'),
        "email":request.form.get('email'),
        "phoneNumber":request.form.get('phoneNumber'),
        "password":request.form.get('password'),
        }
        userInfo['password']=pbkdf2_sha256.encrypt(userInfo['password'])
        if(account.find_one({"email":userInfo['email']})):
            flash("Sorry,user with this email already exist","signupError")
            return redirect(url_for('index',signupError=True))
        if(account.insert_one(userInfo)):
            return start_session(userInfo)
    flash("Signup failed","signupError")
    return redirect(url_for('index',signupError=True))
```

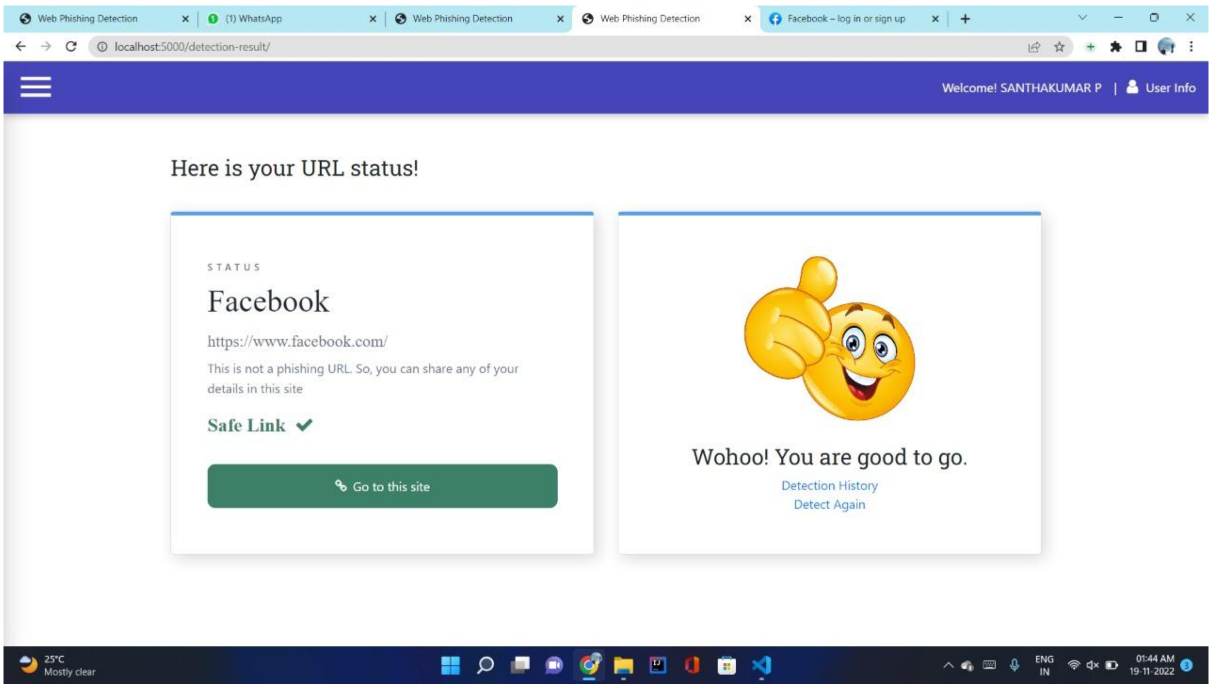
```
@app.route('/logout/',methods=["GET"])
def logout():
    if request.method=="GET":
        session.clear()
    return redirect(url_for('index'))
@app.route('/')
def index():

    if(session and '_flashes' in dict(session)):
        loginError=request.args.get('loginError')
        signupError=request.args.get('signupError')
        if(loginError):
            return
render_template('./index.html',loginError=loginError,carousel_content=caro
uselData['carousel_content'])
        if(signupError):
            return
render_template('./index.html',signupError=signupError,carousel_content=c
arouselData['carousel_content'])
    if(session and '_flashes' not in dict(session)):
        print(dict(session))
        if(session['logged_in']==True):
            return
render_template('./index.html',userInfo=session['user'],carousel_content=c
arouselData['carousel_content'])
        else:
            return
render_template('./index.html',carousel_content=carouselData['carousel_c
ontent'])
    else:
        return
render_template('./index.html',carousel_content=carouselData['carousel_c
ontent'])



@app.route('/predict/', methods=['GET','POST'])
@login_required
def predict():
    if request.method == 'POST':
        title=request.form['title']
        url = request.form['url']
        checkprediction = inputScript.main(url)
        prediction = model.predict(checkprediction)
        output=prediction[0]
        session['predicted']=True
        if(output==1):
            pred = "Wohoo! You are good to go."
            session['pred'] = pred
            session['title']=title
            session['url']=url
            session['safe']=True
            print(session['pred'])

        else:
            pred = "Oh no! This is a Malicious URL"
            session['pred'] = pred
            session['title']=title
            session['url']=url
            session['safe']=False
        detectionInfo={
            'title':session['title'],
            'url':session['url'],
            'safe': session['safe'],

        }
        account.update_one({ "email" : session['user']['email']},
            { "$push": {"detectionInfo": detectionInfo
        }})

        if(session and session['logged_in']):
            if(session['logged_in']==True):
                return redirect(url_for('predictionResult'))
    elif request.method == 'GET':
        return
render_template('./templates/predict-form.html',userInfo=session['user'])


@app.route('/prediction-result/')
@login_required
def predictionResult():
    if(session['predicted']==True):
        urlInfo={
        'message' :session['pred'] ,
        'title':session['title'],
        'url':session['url'],
        'safe':session['safe']
        }

        return render_template("./templates/prediction-result.html",
urlInfo=jsonify(urlInfo),userInfo=session['user'])
```

```
    else:
        return redirect(url_for('predict'))

@app.route('/detection-history/')
@login_required
def detectionHistory():
    if(session and session['logged_in']):
        if(session['logged_in']==True):

getDetectionHistory=account.find({"email":session['user']['email']},{"_id":0,"
detectionInfo":1})
            return
render_template('./templates/detection-history.html',userInfo=session['user'
],detectionHistory=list(getDetectionHistory)[0]['detectionInfo'])


@app.route('/about/')
def about():
    if(session and session['logged_in']):
        if(session['logged_in']==True):
            return
render_template('./templates/about.html',userInfo=session['user'],aboutCon
tents=aboutData['aboutContents'])
        else:
            return
render_template('./templates/about.html',aboutContents=aboutData['about
Contents'])
    else:
        return
render_template('./templates/about.html',aboutContents=aboutData['about
Contents'])
```

```
@app.route('/contact/')
def contact():
        if(session and session['logged_in']):
            if(session['logged_in']==True):
                return
render_template('./templates/contact.html',userInfo=session['user'])
            else:
                return render_template('./templates/contact.html')
        else:
            return render_template('./templates/contact.html')



 #@app.route('/predict/', methods=['POST'])
 #defy_predict():
     #url = request.form['URL']
    #checkprediction = inputScript.main(url)
     #prediction = model.predict(checkprediction)
     #print(prediction)
     #output=prediction[0]
     #if(output==1):
       #  pred="Your are safe!!  This is a Legitimate Website."

     #else:
      #   pred="You are on the wrong site. Be cautious!"
     #return render_template('final.html',
prediction_text='{}'.format(pred),url=url)
     #flash(pred)
```

```
if __name__ == '__main__':
    app.run(host='127.0.0.1', debug=True)
```

]



Here is your URL status!

STATUS

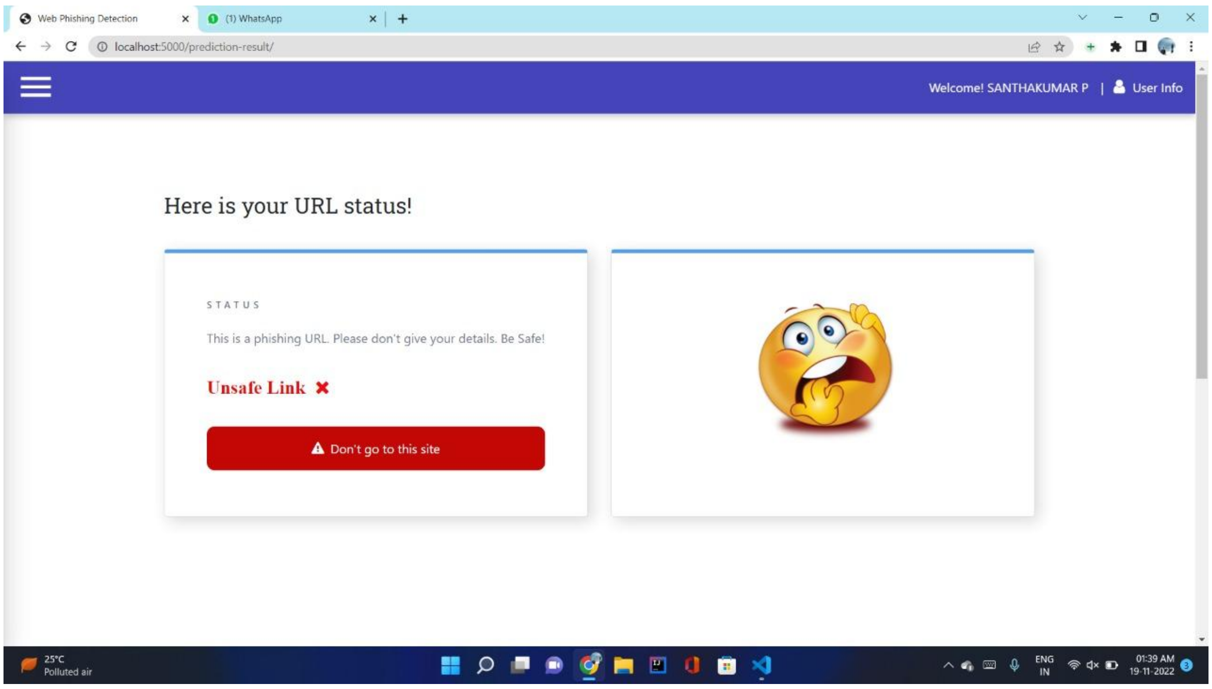
Facebook

https://www.facebook.com/

This is not a phishing URL. So, you can share any of your details in this site

Safe Link ✓

Wohoo! You are good to go.


Web Phishing Detection
(1) WhatsApp
localhost:5000/prediction-result/
Welcome! SANTHAKUMAR P | User Info

Here is your URL status!

STATUS

This is a phishing URL. Please don't give your details. Be Safe!

Unsafe Link ✖

Don't go to this site

7.2 Feature 2:

```
import regex
from tldextract import extract
import socket
from bs4 import BeautifulSoup
import urllib.request
import whois
import requests
import favicon
import re

from googlesearch import search

"""
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\.i
m|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
```

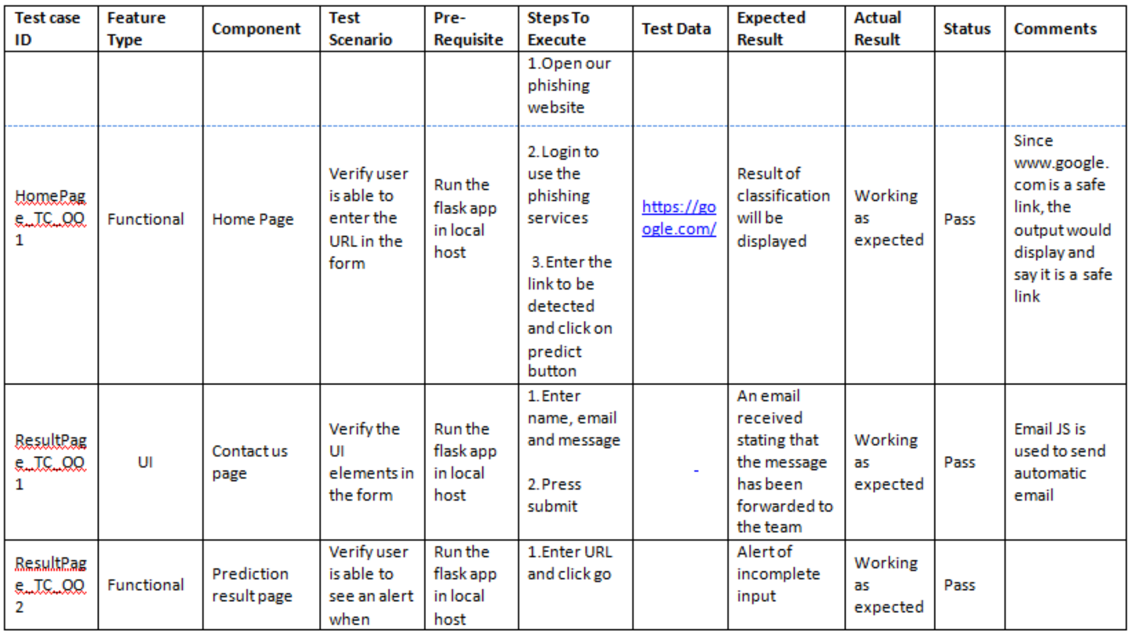
```
#:number of links pointing to webpage = 0 returns 1, number of links pointing to webpage> 0
#and ≤ 2 returns 0, otherwise returns -1

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\.11
6|78\.46\.211\.158|181\.174\.165\.13|46\.242\.145\.103|121\.50\.168\.40|83\.125\.22\.219|46\.242\
.145\.98|107\.151\.148\.44|107\.151\.148\.107|64\.70\.19\.203|199\.184\.144\.27|107\.151\.148\.1
08|107\.151\.148\.109|119\.28\.52\.61|54\.83\.43\.69|52\.69\.166\.231|216\.58\.192\.225|118\.184\
.25\.86|67\.208\.74\.71|23\.253\.126\.58|104\.239\.157\.210|175\.126\.123\.219|141\.8\.224\.221|
10\.10\.10\.10|43\.229\.108\.32|103\.232\.215\.140|69\.172\.201\.153|216\.218\.185\.162|54\.225\
.104\.146|103\.243\.24\.98|199\.59\.243\.120|31\.170\.160\.61|213\.19\.128\.77|62\.113\.226\.131|
208\.100\.26\.234|195\.16\.127\.102|195\.16\.127\.157|34\.196\.13\.28|103\.224\.212\.222|172\.2
17\.4\.225|54\.72\.9\.51|192\.64\.147\.141|198\.200\.56\.183|23\.253\.164\.103|52\.48\.191\.26|52
\.214\.197\.72|87\.98\.255\.18|209\.99\.17\.27|216\.38\.62\.18|104\.130\.124\.96|47\.89\.58\.141|7
8\.46\.211\.158|54\.86\.225\.156|54\.82\.156\.19|37\.157\.192\.102|204\.11\.56\.48|110\.34\.231\.4
2',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
```

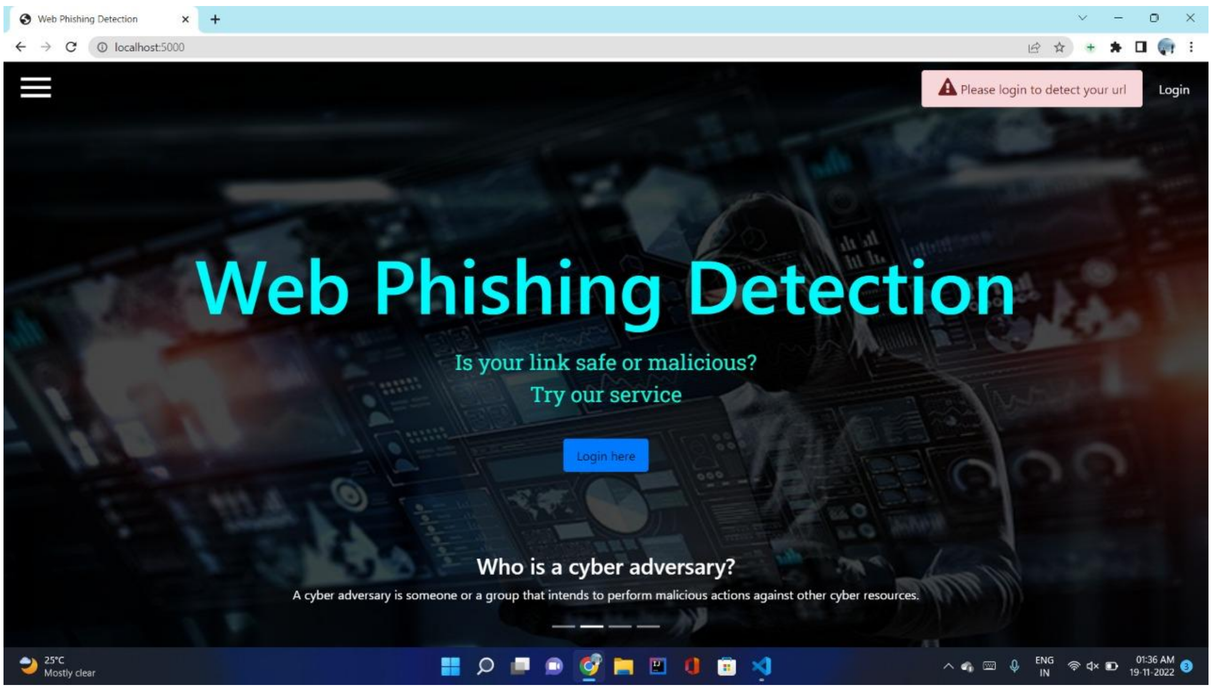
8.TESTING

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
					1.Open our phishing website					
HomePage_TC_OO1	Functional	Home Page	Verify user is able to enter the URL in the form	Run the flask app in local host	2.Login to use the phishing services 3.Enter the link to be detected and click on predict button	https://google.com/	Result of classification will be displayed	Working as expected	Pass	Since www.google.com is a safe link, the output would display and say it is a safe link
ResultPage_TC_OO1	UI	Contact us page	Verify the UI elements in the form	Run the flask app in local host	1.Enter name, email and message 2.Press submit	-	An email received stating that the message has been forwarded to the team	Working as expected	Pass	Email JS is used to send automatic email
ResultPage_TC_OO2	Functional	Prediction result page	Verify user is able to see an alert when	Run the flask app in local host	1.Enter URL and click go		Alert of incomplete input	Working as expected	Pass	

			nothing is entered in the textbox		2.Enter nothing and click submit 3.An alert is displayed to provide proper input					
Prediction Page_TC_OO1	Functional	Prediction form page	Verify user is able to see the result when URL is entered in the textbox	Run the flask app in local host	1.Enter URL and click go 2. Enter any URL and click submit 3. The result of the classification is displayed in a new page.	https://google.com/	Result of classification will be displayed with a corresponding emoticon	Working as expected	Pass	

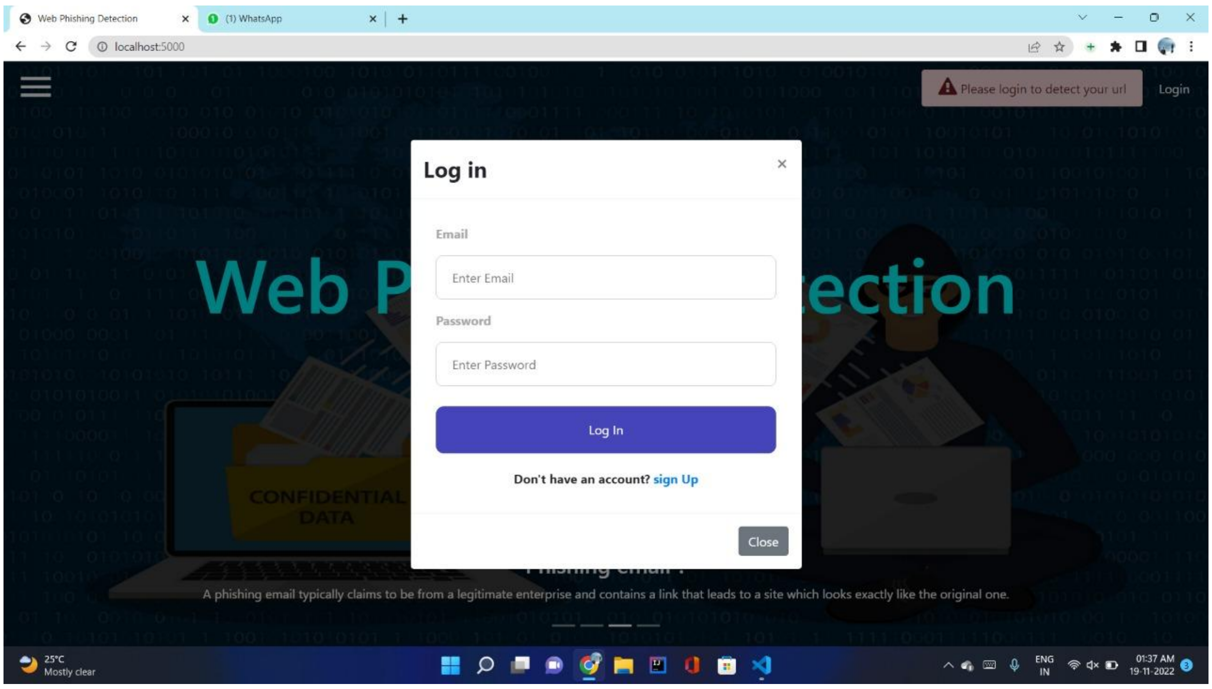


Web Phishing Detection

Is your link safe or malicious?
Try our service

Who is a cyber adversary?

A cyber adversary is someone or a group that intends to perform malicious actions against other cyber resources.



Log in

Email

Enter Email

Password

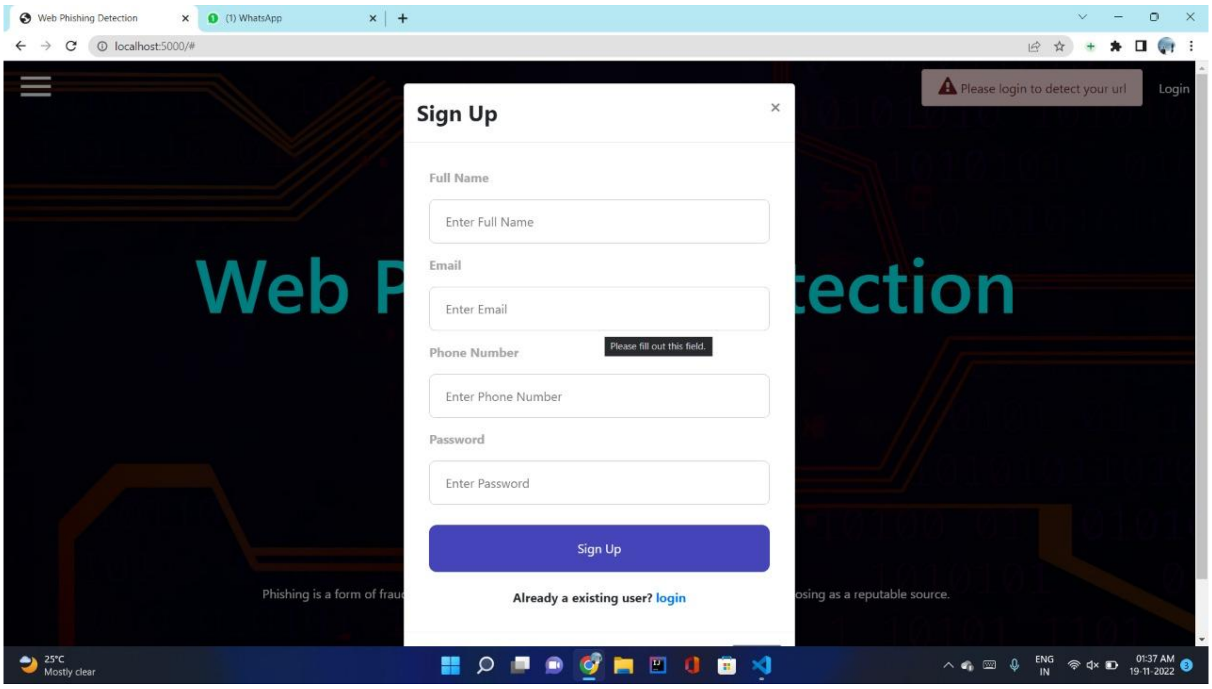
Enter Password

Log In

Don't have an account? sign Up

Close

A phishing email typically claims to be from a legitimate enterprise and contains a link that leads to a site which looks exactly like the original one.



Sign Up

Full Name

Enter Full Name

Email

Enter Email

Please fill out this field.

Phone Number

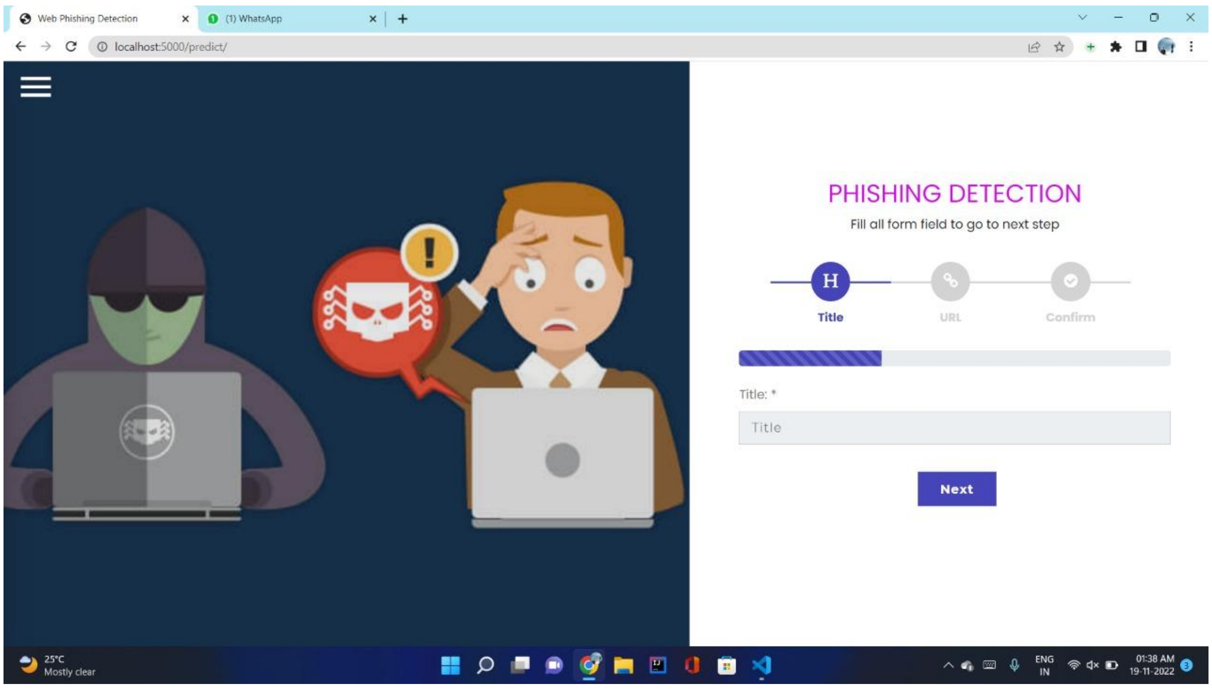
Enter Phone Number

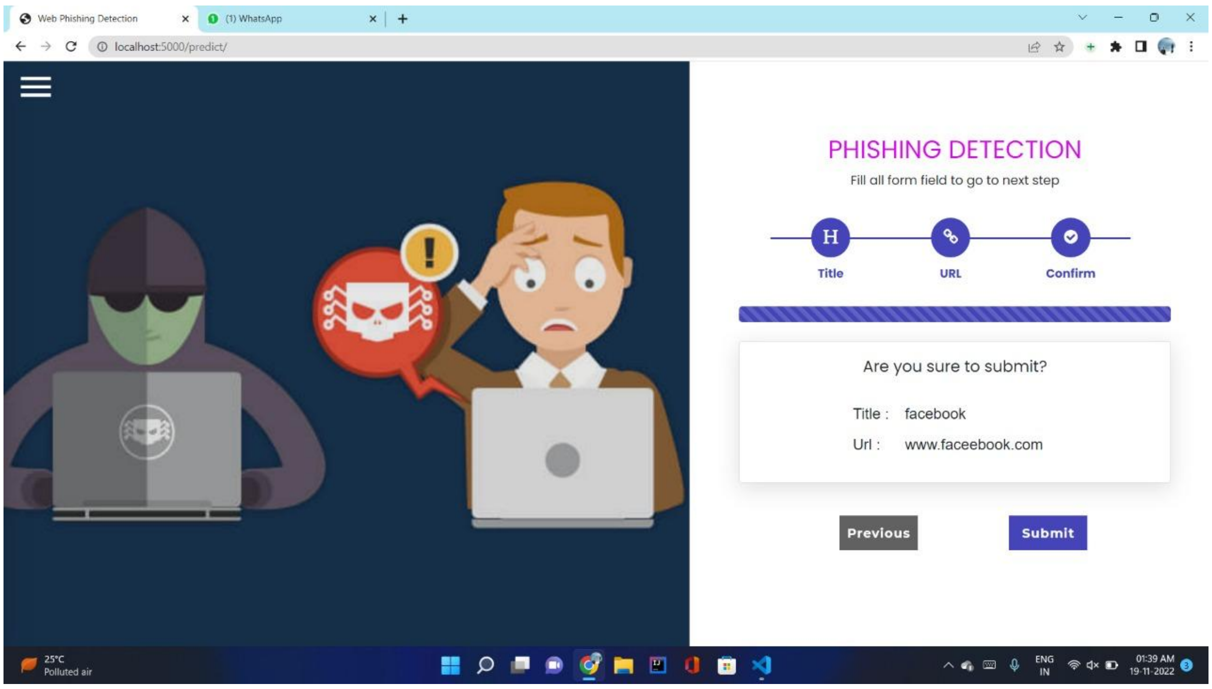
Password

Enter Password

Sign Up

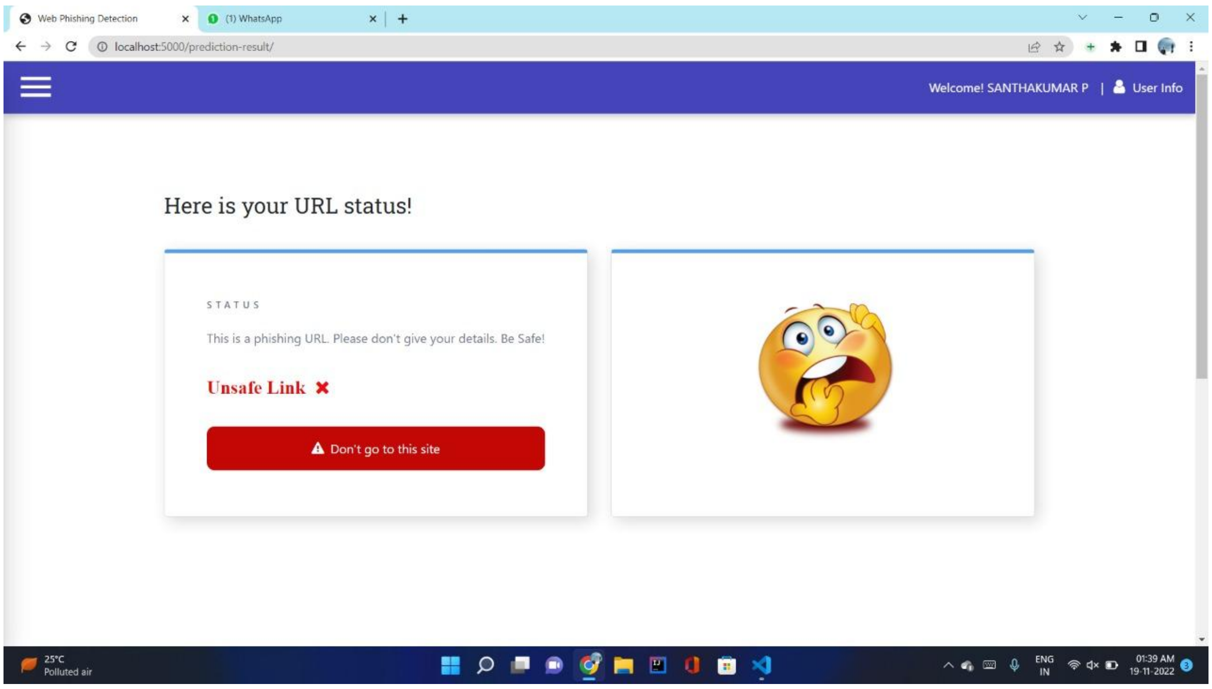
Already a existing user? login





Are you sure to submit?

Title :	facebook
Url :	www.faceebook.com



Web Phishing Detection | (1) WhatsApp

localhost:5000/prediction-result/

Welcome! SANTHAKUMAR P | User Info

Here is your URL status!

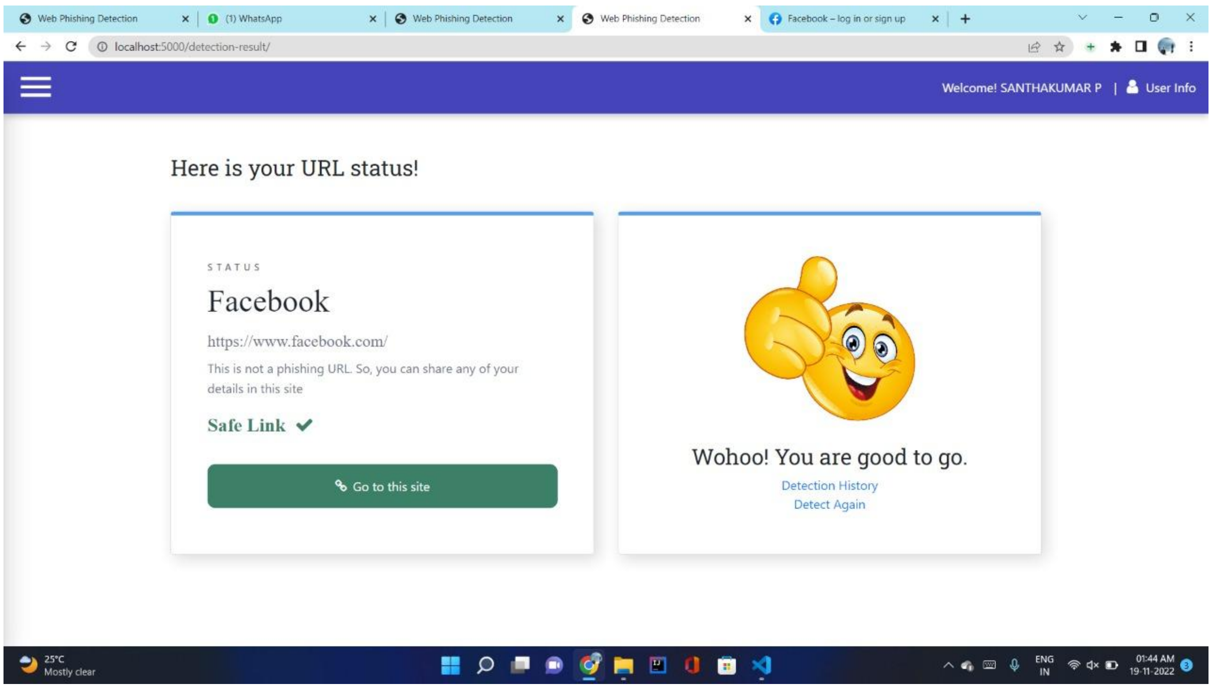
STATUS

This is a phishing URL. Please don't give your details. Be Safe!

Unsafe Link ✖

Don't go to this site

25°C Polluted air | ENG IN | 01:39 AM 19-11-2022



Web Phishing Detection | (1) WhatsApp | Web Phishing Detection | Web Phishing Detection | Facebook – log in or sign up

localhost:5000/detection-result/

Welcome! SANTHAKUMAR P | User Info

Here is your URL status!

STATUS

Facebook

https://www.facebook.com/

This is not a phishing URL. So, you can share any of your details in this site

Safe Link ✔

Go to this site

Wohoo! You are good to go.

Detection History

Detect Again

25°C Mostly clear | ENG IN | 01:44 AM 19-11-2022

8.2 User Acceptance Testing:

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	11	2	4	20	37
Not Reproduced	0	0	1	0	1
Skipped	0	0	1	1	2
Won't Fix	0	5	2	1	8
Totals	24	14	13	26	77

3. Test Case Analysis

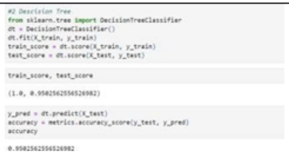
This report shows the number of test cases that have passed, failed, and untested

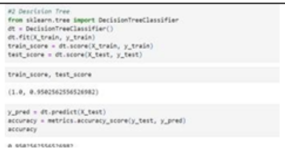
Section	Total Cases	Not Tested	Fail	Pass
Print Engine	5	0	0	5-
Client Application	51	0	0	51
Security	2	0	0	2
Outsource Shipping	3	0	0	3
Exception Reporting	9	0	0	9
Final Report Output	4	0	0	4
Version Control	2	0	0	2

9.RESULT

9.1Performance Metrics:

S.No.	Parameter	Values	Screenshot

1.	Model Summary	**Decision Tree ModelAccuracy – 97%**	 `#2 Descision Tree`
			`from sklearn.tree import DecisionTreeClassifier`
			`dt = DecisionTreeClassifier()`
			`dt.fit(X_train, y_train)`
			`train_score = dt.score(X_train, y_train)`
			`test_score = dt.score(X_test, y_test)`
			
			`train_score, test_score`
			
			`(1.0, 0.9502562556526982)`
			
			`y_pred = dt.predict(X_test)`
			`accuracy = metrics.accuracy_score(y_test, y_pred)`
			`accuracy`
			
			`0.9502562556526982`

2.	Accuracy	Training Accuracy -Test	 `#2 Descision Tree`
			`from sklearn.tree import DecisionTreeClassifier`
			`dt = DecisionTreeClassifier()`
			`dt.fit(X_train, y_train)`
			`train_score = dt.score(X_train, y_train)`
			`test_score = dt.score(X_test, y_test)`
			
			`train_score, test_score`
			
			`(1.0, 0.9502562556526982)`
			
			`y_pred = dt.predict(X_test)`
			`accuracy = metrics.accuracy_score(y_test, y_pred)`
			`accuracy`
			
			`0.9502562556526982`

10.ADVANTAGES & DISADVANTAGES

10.1Advantages:

- This system can be used by many E-commerce or other websites in order to have good customer relationship.
- User can make online payment securely.
- Data mining algorithm used in this system provides better performance as compared to other traditional classifications algorithms.
- With the help of this system user can also purchase products online without any hesitation.

10.2 Disadvantages:

- If Internet connection fails, this system won't work.
- All websites related data will be stored in one place.

11.CONCLUSION

Using machine learning technologies, this initiative seeks to improve the detection process for phishing websites. Using the random forest approach, we had the lowest percentage of false positives and 97.14% detection accuracy. The outcome further demonstrates that classifiers perform better when more data is utilized as training data. Future phishing website detection will be more accurate thanks to the implementation of hybrid technology, which combines the blacklist approach with the random forest algorithm of machine learning.

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.

APPENDIX

Source Code:

GitHub:https://github.com/IBM-EPBL/IBM-Project-11545-1659333974.git

Project Demo Link:

https://youtu.be/7Hx6P1gA4K0