NALAIYA THIRAN

APPLIED DATA SCIENCE

WEB PHISHING DETECTION

TEAM ID : PNT2022TMID35546

Submitted By

BRINDHA B (2019115027)

ANSILIN VINCY (2019115015)

PREETHI S (2019115072)

VELLAI KUMARAPAN (2019115119)

in the partial fulfillment of the award of the degree

of

BACHELOR OF TECHNOLOGY

in

INFORMATION TECHNOLOGY

ANNA UNIVERSITY , CEG

Chennai – 25

TABLE OF CONTENTS

1. INTRODUCTION

1.1 PROJECT OVERVIEW

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 property damage. Large organizations may get trapped in different kinds of scams.
In order to detect and predict e-banking phishing websites, we proposed an intelligent, flexible and effective system that is based on using classification algorithms. We implemented classification algorithms and techniques to extract the phishing datasets criteria to classify their legitimacy. The e-banking phishing website can be detected based on some important characteristics like URL and domain identity, and security and encryption criteria in the final phishing detection rate.

1.2 PURPOSE

The main purpose of this project is to help people to identify the fake websites that is the phishing sites and to avoid them. By our project, people who are unaware about this phishing can protect their private information such as usernames, passwords, credit card details and like. They can prevent property damage and any kind of black mile.

2. LITERATURE SURVEY

2.1 EXISTING PROBLEM

This project focuses on identifying the quality of website before using it so that we don’t fall into web phishing websites and lose our data, money or any other resource.

2.2 REFERENCES

The following papers were referred to for the purpose knowing the preexisting methods of building the model.

- Detection and Prevention of Phishing Websites Using Machine Learning Approach [1] makes use of visual appearance-based analysis for checking genuineness of website.

- Detecting Phishing Websites Using Machine Learning [2] makes use of a machine learning method, particularly supervised learning - Random Forest technique.

- Detection of Phishing Websites by Using Machine Learning-Based URL Analysis [3] uses eight different algorithms to analyse the URLs, and three different datasets to compare the results with other works

- URL-based Phishing Websites Detection via Machine Learning [4] employs machine learning techniques such as neural networks and decision trees to learn data patterns in websites URLs. They evaluate the system on a recent phishing websites dataset using classification accuracy as a performance indicator.

- Detection of Phishing Websites from URLs by using Classification Techniques on WEKA [5] using 4 classification algorithms with this dataset to detect phishing sites.

2.3 PROBLEM STATEMENT DEFINITION

To detect the safety and quality of websites while preventing the user from accessing sites that may be harmful by using ML techiniques to build model based on the URL of the website.

3. IDEATION & PROPOSED SOLUTION

3.1 EMPATHY MAP CANVAS

- An empathy map is a straightforward, simple-to-understand picture that summarizes information about a user's actions and views.
- Teams can utilize an empathy map as a collaborative tool to obtain a deeper understanding of their customer.
- An empathy map is a popular visualization tool in the UX and HCI fields of practice.
- An empathy map's main objective in empathetic design is to bridge the understanding of the end user.

- A rectangle divided into four quadrants, with the user or client in the center, is an empathy map. A category is included in each of the four quadrants to assist us in better understanding the user's perspective.
- The four empathy map quadrants examine the user's actions, thoughts, and feelings.

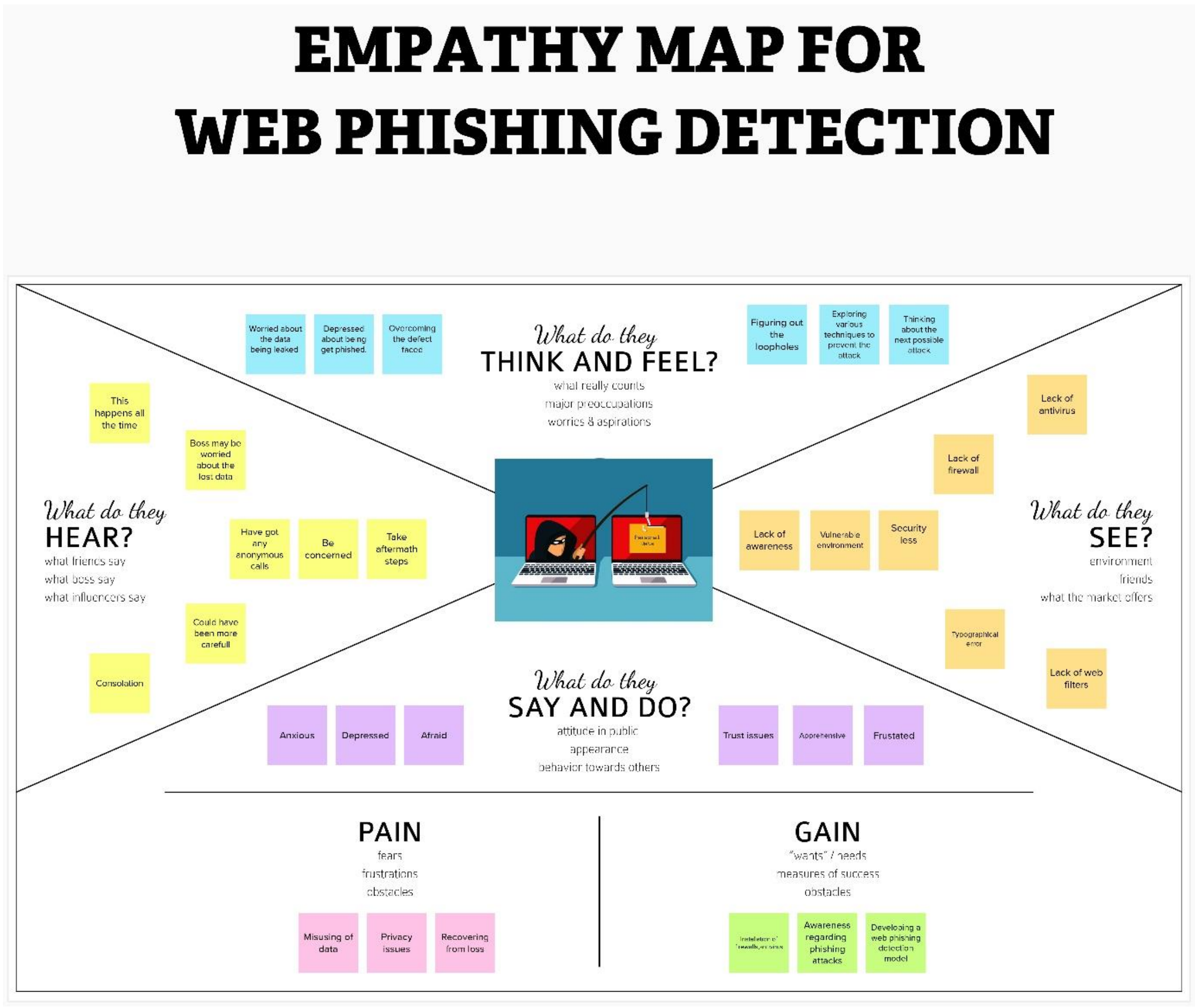


Figure 3.1 Empathy Map

From the empathy map we can infer about user's thought about web phishing like they might feel secured and confident while browsing online with the support of the proposed site.

3.2 IDEATION & BRAINSTORMING

The following diagram 3.2 a illustrates the brainstorming done for the web phishing detection system.

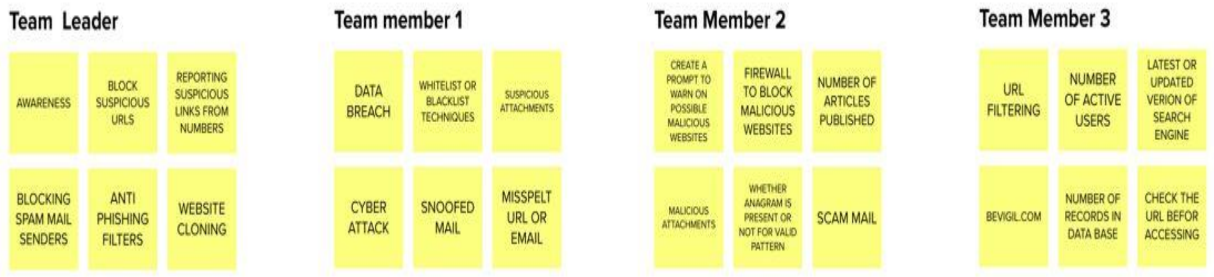
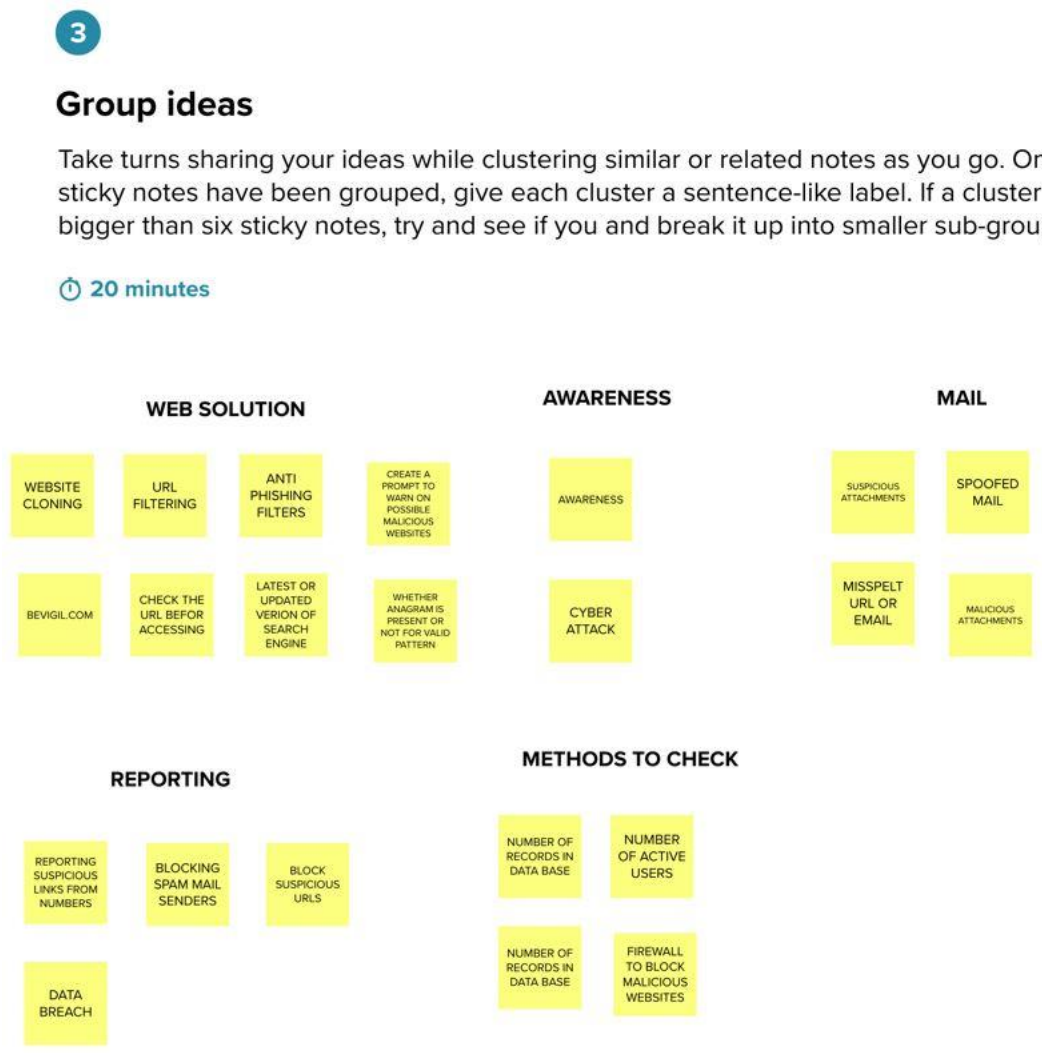

Team Leader
AWARENESS
BLOCK SUSPICIOUS URLS
REPORTING SUSPICIOUS LINKS FROM NUMBERS
BLOCKING SPAM MAIL SENDERS
ANTI PHISHING FILTERS
WEBSITE CLONING
Team member 1
DATA BREACH
WHITELIST OR BLACKLIST TECHNIQUES
SUSPICIOUS ATTACHMENTS
CYBER ATTACK
SNOOFED MAIL
MISSPELT URL OR EMAIL
Team Member 2
CREATE A PROMPT TO WARN ON POSSIBLE MALICIOUS WEBSITES
FIREWALL TO BLOCK MALICIOUS WEBSITES
NUMBER OF ARTICLES PUBLISHED
MALICIOUS ATTACHMENTS
WHETHER ANAGRAM IS PRESENT OR NOT FOR VALID PATTERN
SCAM MAIL
Team Member 3
URL FILTERING
NUMBER OF ACTIVE USERS
LATEST OR UPDATED VERION OF SEARCH ENGINE
BEVIGIL.COM
NUMBER OF RECORDS IN DATA BASE
CHECK THE URL BEFOR ACCESSING

Figure 3.2 a Ideation & Brainstorming



3

Group ideas

Take turns sharing your ideas while clustering similar or related notes as you go. Once all sticky notes have been grouped, 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

WEB SOLUTION
WEBSITE CLONING
URL FILTERING
ANTI PHISHING FILTERS
CREATE A PROMPT TO WARN ON POSSIBLE MALICIOUS WEBSITES
BEVIGIL.COM
CHECK THE URL BEFOR ACCESSING
LATEST OR UPDATED VERION OF SEARCH ENGINE
WHETHER ANAGRAM IS PRESENT OR NOT FOR VALID PATTERN
AWARENESS
AWARENESS
CYBER ATTACK
MAIL
SUSPICIOUS ATTACHMENTS
SPOOFED MAIL
MISSPELT URL OR EMAIL
MALICIOUS ATTACHMENTS
REPORTING
REPORTING SUSPICIOUS LINKS FROM NUMBERS
BLOCKING SPAM MAIL SENDERS
BLOCK SUSPICIOUS URLS
DATA BREACH
METHODS TO CHECK
NUMBER OF RECORDS IN DATA BASE
NUMBER OF ACTIVE USERS
NUMBER OF RECORDS IN DATA BASE
FIREWALL TO BLOCK MALICIOUS WEBSITES

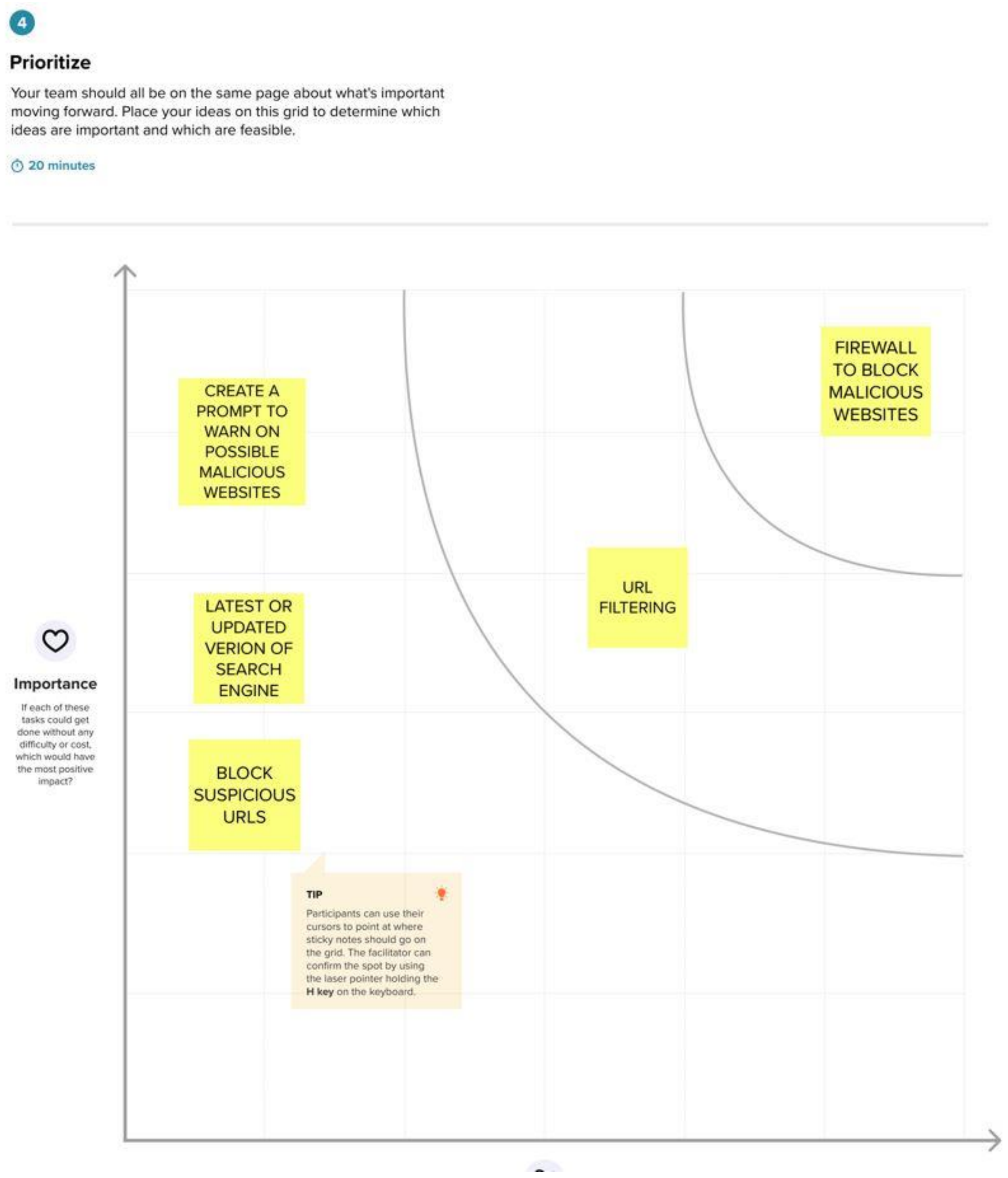

4
Prioritize
Your team should all be on the same page about what's important moving forward. Place your ideas on this grid to determine which ideas are important and which are feasible.
20 minutes
CREATE A PROMPT TO WARN ON POSSIBLE MALICIOUS WEBSITES
FIREWALL TO BLOCK MALICIOUS WEBSITES
URL FILTERING
LATEST OR UPDATED VERION OF SEARCH ENGINE
BLOCK SUSPICIOUS URLS
Importance
If each of these tasks could get done without any difficulty or cost, which would have the most positive impact?
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.

Figure 3.2 b Ideation & Brainstorming

Some of the highlights of Ideation and Brainstorming are:

Collecting dataset with necessary attributes for prediction of security of the website.
Building model using the dataset for prediction.
Creating an user interface for the users.

3.3 PROPOSED SOLUTION

The increase in the number of online phishing dramatically over the years have led the spark to build the phishing detection.

Idea / solution description:

To secure the users from phishers to avoid the loss of personal details like geo location, banking credentials etc. through various kind of engineered tools to detect the unusual activities in the user's machine and alert them.

Novelty / uniqueness:

Our application is very easy to use with a simple UI and very user friendly which is well designed for the intended purpose and very light weight which you cannot even compare with other similar services.

Social Impact / Customer Satisfaction:

It will create awareness among users to be secure online. The users will be very satisfied that they are in safe hands and need not worry about the problems described earlier.

Business Model (Revenue Model):

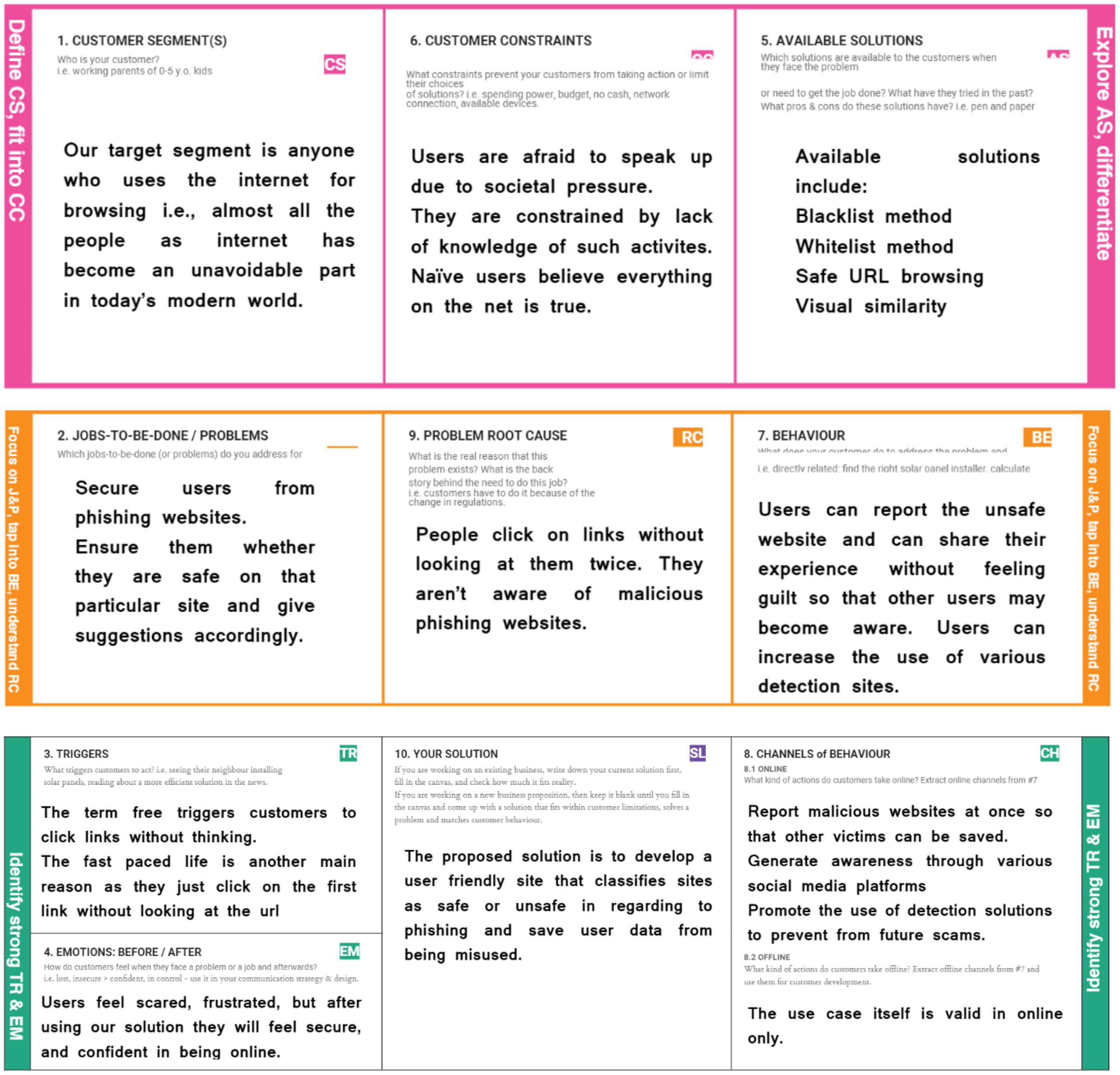
Revenue will be generated through charging a price (subscription) to the advanced features and basic features will be free of cost

Scalability of the Solution:

By securing the user base and earning their trust securing enterprises an scaling the business and capturing the market-share

3.4 PROBLEM SOLUTION FIT

The following diagram 3.4 a illustrates the Solution Fit.



Define CS, fit into CC

1. CUSTOMER SEGMENT(S)

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

CS

Our target segment is anyone who uses the internet for browsing i.e., almost all the people as internet has become an unavoidable part in today's modern world.

6. CUSTOMER CONSTRAINTS

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

Users are afraid to speak up due to societal pressure.
They are constrained by lack of knowledge of such activites.
Naïve users believe everything on the net is true.

5. AVAILABLE SOLUTIONS

Which solutions are available to the customers when they face the problem

or need to get the job done? What have they tried in the past?
What pros & cons do these solutions have? i.e. pen and paper

Available solutions include:
Blacklist method
Whitelist method
Safe URL browsing
Visual similarity

Explore AS, differentiate

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

2. JOBS-TO-BE-DONE / PROBLEMS

Which jobs-to-be-done (or problems) do you address for

Secure users from phishing websites.
Ensure them whether they are safe on that particular site and give suggestions accordingly.

9. PROBLEM ROOT CAUSE

RC

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

People click on links without looking at them twice. They aren't aware of malicious phishing websites.

7. BEHAVIOUR

BE

i.e. directly related: find the right solar panel installer, calculate

Users can report the unsafe website and can share their experience without feeling guilt so that other users may become aware. Users can increase the use of various detection sites.

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

Identify strong TR & EM

3. TRIGGERS

TR

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

The term free triggers customers to click links without thinking.
The fast paced life is another main reason as they just click on the first link without looking at the url

4. EMOTIONS: BEFORE / AFTER

EM

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

Users feel scared, frustrated, but after using our solution they will feel secure, and confident in being online.

10. YOUR SOLUTION

SL

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

The proposed solution is to develop a user friendly site that classifies sites as safe or unsafe in regarding to phishing and save user data from being misused.

8. CHANNELS of BEHAVIOUR

CH

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

Report malicious websites at once so that other victims can be saved.
Generate awareness through various social media platforms
Promote the use of detection solutions to prevent from future scams.

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

The use case itself is valid in online only.

Identify strong TR & EM

Figure 3.4 Solution Fit

4. REQUIREMENT ANALYSIS

4.1 FUNCTIONAL REQUIREMENTS

Following are the functional requirements of the proposed solution.

4.1 a) User Input

The user should give the url of the website which they want to check.

4.1 b) Eligibility of URL

The url can be either safe to user or unsafe.

4.1 c) Redirection

Either way the site provides option to redirect to that site.

4.2 NON-FUNCTIONAL REQUIREMENTS

The following are non-functional requirements that are essential.

4.2 a) Usability

The system ensures intuitiveness in the user by providing an easy-to-use and self-explainable website. The site allows users to navigate between pages. The website also has a simple structure thereby enabling faster access.

4.2 b) Reliability

The results produced are obtained from ensemble the outputs of various machine learning models. Thereby the system is highly reliable.

4.2 c) Performance

The proposed web-based application has the ability to indicate if the user inputs erroneous data types.

4.2 d) Availability

A simple web browser is enough to access the website.

4.2 e) Scalability

The site can be extended for any type of url checking and also can be developed into a web extension for easy access.

5. PROJECT DESIGN

5.1 SOLUTION & TECHNICAL ARCHITECTURE

The following diagram 5.1 shows the technical architecture of the system.

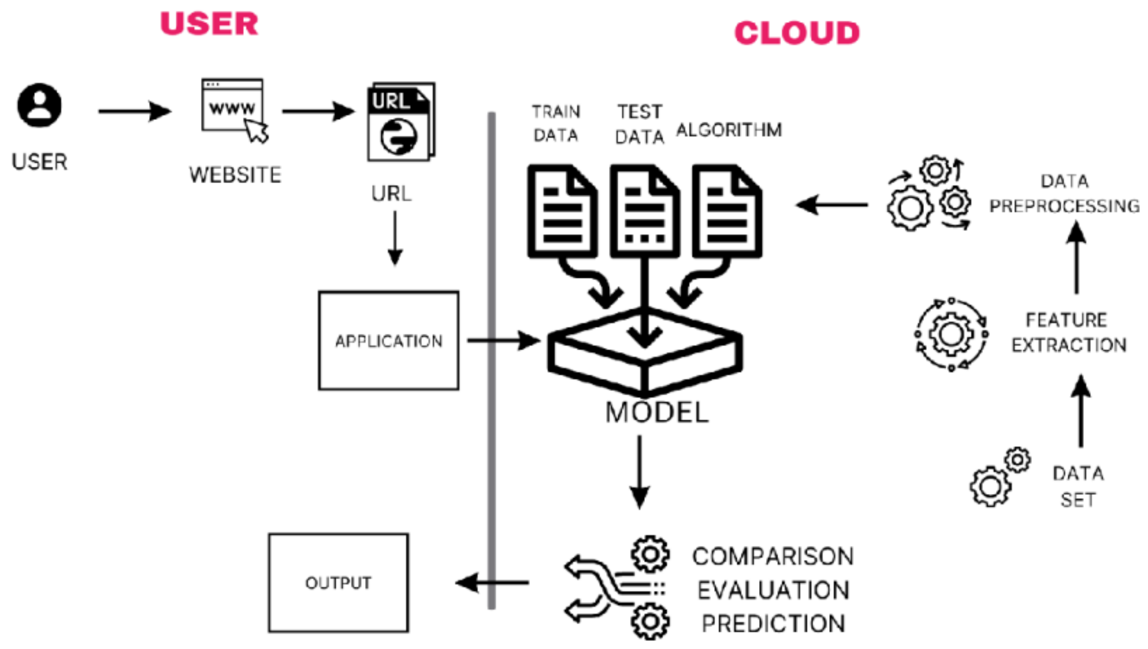
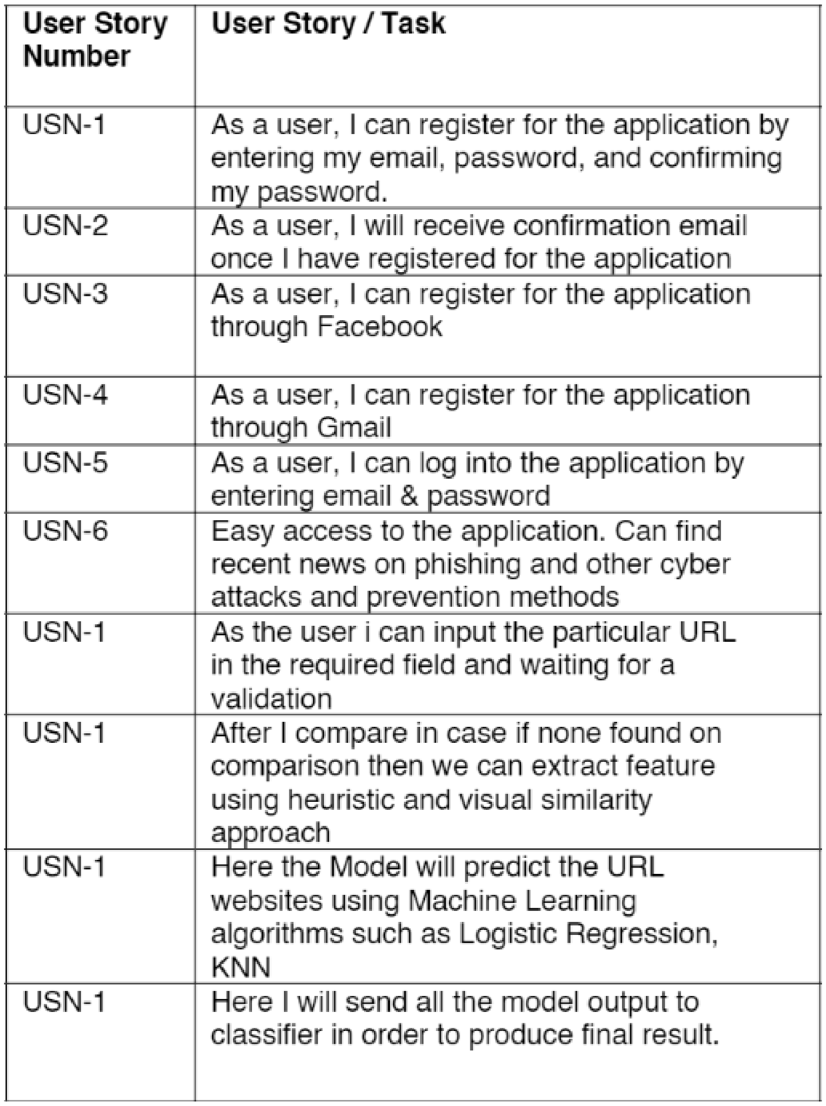


Figure 5.2 Technical Architecture

5.2 USER STORIES



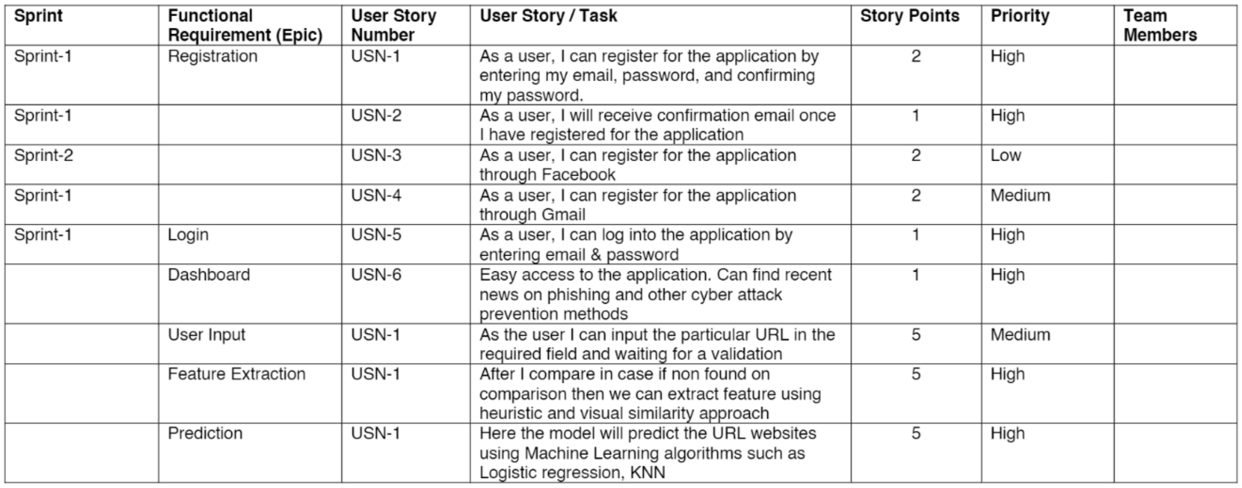
User Story Number	User Story / Task
USN-1	As a user, I can register for the application by entering my email, password, and confirming my password.
USN-2	As a user, I will receive confirmation email once I have registered for the application
USN-3	As a user, I can register for the application through Facebook
USN-4	As a user, I can register for the application through Gmail
USN-5	As a user, I can log into the application by entering email & password
USN-6	Easy access to the application. Can find recent news on phishing and other cyber attacks and prevention methods
USN-1	As the user i can input the particular URL in the required field and waiting for a validation
USN-1	After I compare in case if none found on comparison then we can extract feature using heuristic and visual similarity approach
USN-1	Here the Model will predict the URL websites using Machine Learning algorithms such as Logistic Regression, KNN
USN-1	Here I will send all the model output to classifier in order to produce final result.

Figure 5.2 User Stories.

6. PROJECT PLANNING & SCHEDULING

6.1 SPRINT PLANNING & ESTIMATION

The following figure 6.1 gives the sprint planning.

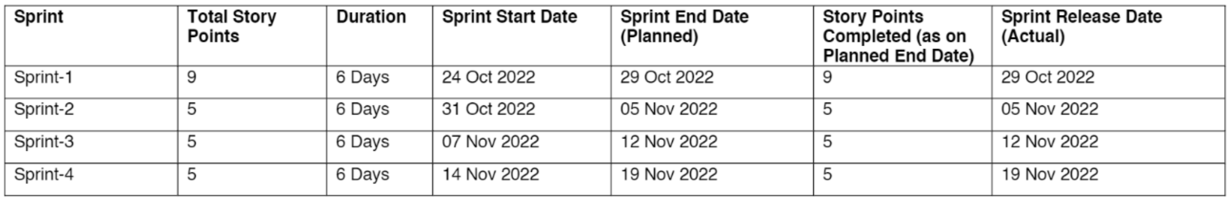


Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-1	Registration	USN-1	As a user, I can register for the application by entering my email, password, and confirming my password.	2	High	
Sprint-1		USN-2	As a user, I will receive confirmation email once I have registered for the application	1	High	
Sprint-2		USN-3	As a user, I can register for the application through Facebook	2	Low	
Sprint-1		USN-4	As a user, I can register for the application through Gmail	2	Medium	
Sprint-1	Login	USN-5	As a user, I can log into the application by entering email & password	1	High	
	Dashboard	USN-6	Easy access to the application. Can find recent news on phishing and other cyber attack prevention methods	1	High	
	User Input	USN-1	As the user I can input the particular URL in the required field and waiting for a validation	5	Medium	
	Feature Extraction	USN-1	After I compare in case if non found on comparison then we can extract feature using heuristic and visual similarity approach	5	High	
	Prediction	USN-1	Here the model will predict the URL websites using Machine Learning algorithms such as Logistic regression, KNN	5	High	

Figure 6.1 Sprint Planning

6.2 SPRINT DELIVERY SCHEDULE

The following figure 6.2 gives the sprint delivery schedule.



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	9	6 Days	24 Oct 2022	29 Oct 2022	9	29 Oct 2022
Sprint-2	5	6 Days	31 Oct 2022	05 Nov 2022	5	05 Nov 2022
Sprint-3	5	6 Days	07 Nov 2022	12 Nov 2022	5	12 Nov 2022
Sprint-4	5	6 Days	14 Nov 2022	19 Nov 2022	5	19 Nov 2022

Figure 6.2 Sprint Delivery Schedule

6.3 REPORTS FROM JIRA

JIRA Tool was used for project monitoring and management.The tasks for the project had been defined and divided into four sprints . The storypoints and priority for each task is assigned.The start and end time for each sprint is added. Once the task is started it will be in progress state. After it is completed it will be in review state. Similarly all the four sprints had been completed.

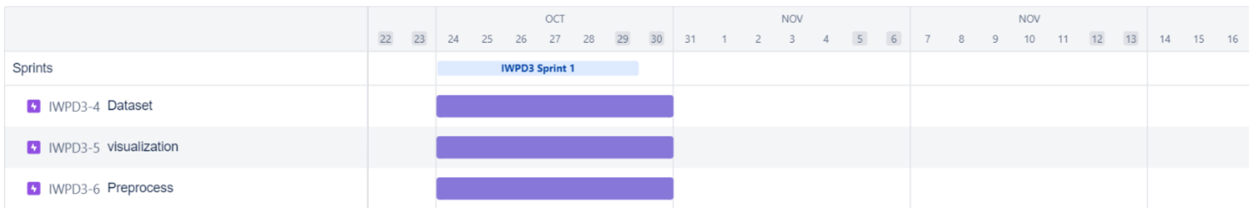


Figure 6.3 Sprint Monitoring

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

7.1 WEBSITE SAFENESS PREDICTION

1. LOGISTIC REGRESSION:

Logistic regression predicts the output of a categorical dependent variable. Therefore the outcome must be a categorical or discrete value. Logistic Regression is much similar to the Linear Regression except that how they are used. Linear Regression is used for solving Regression problems, whereas Logistic regression is used for solving the classification problems.

```
# Linear regression model
from sklearn.linear_model import LogisticRegression
#from sklearn.pipeline import Pipeline

# instantiate the model
log = LogisticRegression()

# fit the model
log.fit(X_train,y_train)
```

2. K- NEAREST NEIGHBOUR CLASSIFICATION:

K-Nearest Neighbour is one of the simplest Machine Learning algorithms based on Supervised Learning technique. K-NN algorithm assumes the similarity between the new case/data and available cases and put the new case into the category that is most similar to the available categories.

```
# K-Nearest Neighbors Classifier model
from sklearn.neighbors import KNeighborsClassifier

# instantiate the model
knn = KNeighborsClassifier(n_neighbors=1)

# fit the model
knn.fit(X_train,y_train)
```

3. **SUPPORT VECTOR MACHINE CLASSIFIER**:

Support Vector Machine or SVM is one of the most popular Supervised Learning algorithms, which is used for Classification as well as Regression problems. The goal of the SVM algorithm is to create the best line or decision boundary that can segregate n-dimensional space into classes so that we can easily put the new data point in the correct category in the future.

```
# Support Vector Classifier model
from sklearn.svm import SVC
from sklearn.model_selection import GridSearchCV

# defining parameter range
param_grid = {'gamma': [0.1],'kernel': ['rbf','linear']}

svc = GridSearchCV(SVC(), param_grid)

# fitting the model for grid search
svc.fit(X_train, y_train)
```

4. NAÏVE BAYES CLASSIFIER:

Naïve Bayes algorithm is a supervised learning algorithm, which is based on Bayes theorem and used for solving classification problems.It is mainly used in text, image classification that includes a high-dimensional training dataset. Naïve Bayes Classifier is one of the simple and most effective Classification algorithms which helps in building the fast machine learning models that can make quick predictions.

```
# Naive Bayes Classifier Model
```

```
from sklearn.naive_bayes import GaussianNB
from sklearn.pipeline import Pipeline

# instantiate the model
nb=  GaussianNB()

# fit the model
nb.fit(X_train,y_train)
```

5. DECISION TREE:

Decision Tree is a Supervised learning technique that can be used for both classification and Regression problems, but mostly it is preferred for solving Classification problems. It is a tree-structured classifier, where internal nodes represent the features of a dataset, branches represent the decision rules and each leaf node represents the outcome.

```
# Decision Tree Classifier model
from sklearn.tree import DecisionTreeClassifier

# instantiate the model
tree = DecisionTreeClassifier(max_depth=30)

# fit the model
tree.fit(X_train, y_train)
```

8. TESTING

8.1TEST CASES

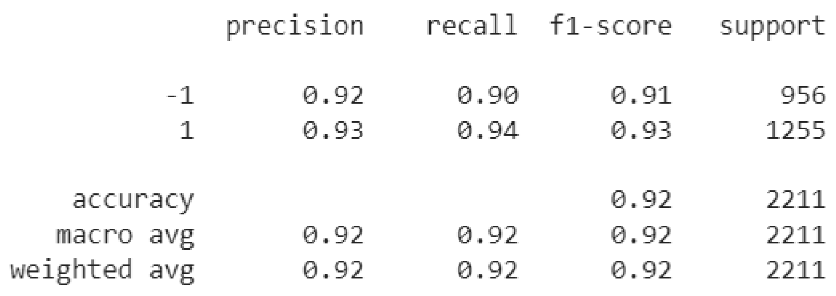
Test Scenario	Steps To Execute	Test Data	Expected Result
Verify user is able to see the links for both login and register	1.Enter URL 2.Verify login/Singup displayed or not		Login/Signup link should display
Verify login link redirects to login page	Click login button, it will redirect to login page		On clicking login button it should redirect to login page
Verify register redirects to signup page	Click register button, it will redirect to register page		On clicking register button it should redirect to register page
Verify user's registered credentials alone logins and shows invalid credentials for wrong username or password	Enter valid username in username box and valid password in password box and clicl login	Username: IBMPROJECT password: 12345678	Should login only the correct registered credentials
Verify after register redirects to login page	Click register button, it will redirect to login page		On clicking register button it should redirect to login page
Verify after login redirects to home page	Click login button, it will redirect to home page		On clicking login button it should redirect to home page
Verify safe link given, the url is predicted as safe	1.Enter URL and click go 2. Copy paste the URL 3. Check whether the website is legitimate or not. 4. Continue if the website is legitimate or be cautious if it is not.		Given correct url it should show it is safe
Verify when phishing link given predicted as unsafe	1.Enter URL and click go 2. Copy paste the URL 3. Check whether the website is legitimate or not. 4. Continue if the website is legitimate or be cautious if it is not.		Given incorrect url it should show it is unsafe
Verify the continue button redirects to the given url	Click on ontinue button redirects to the given url		On clicking the continue button it should redirect to the url given by the user
Verify the logout button log outs the user and redirects to welcome page	Click on the logout button continues to the welcome page		On clicking logout button it should redirect to welcome page

Table 8.1 Test Cases

9. RESULTS

9.1 PERFORMANCE METRICS

Logistic regression:

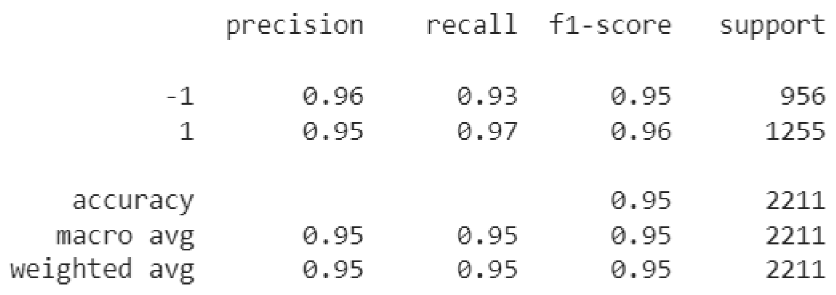


```
              precision    recall  f1-score   support

          -1       0.92      0.90      0.91       956
           1       0.93      0.94      0.93      1255

    accuracy                           0.92      2211
   macro avg       0.92      0.92      0.92      2211
weighted avg       0.92      0.92      0.92      2211
```

KNN:

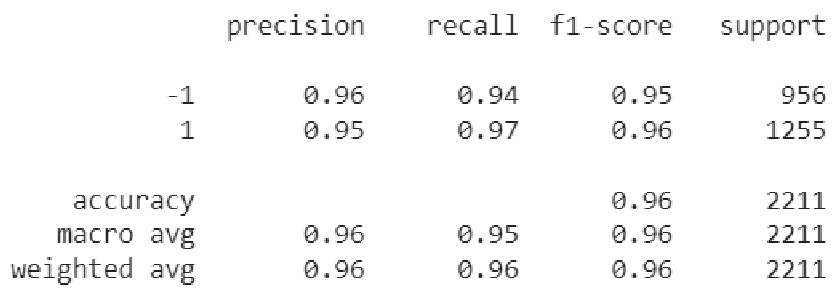


```
              precision    recall  f1-score   support

          -1       0.96      0.93      0.95       956
           1       0.95      0.97      0.96      1255

    accuracy                           0.95      2211
   macro avg       0.95      0.95      0.95      2211
weighted avg       0.95      0.95      0.95      2211
```

Support Vector Machine:

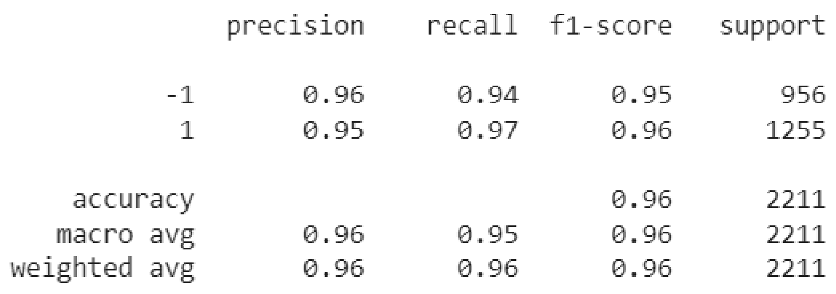


```
              precision    recall  f1-score   support

          -1       0.96      0.94      0.95       956
           1       0.95      0.97      0.96      1255

    accuracy                           0.96      2211
   macro avg       0.96      0.95      0.96      2211
weighted avg       0.96      0.96      0.96      2211
```

Naïve bayes classifier:

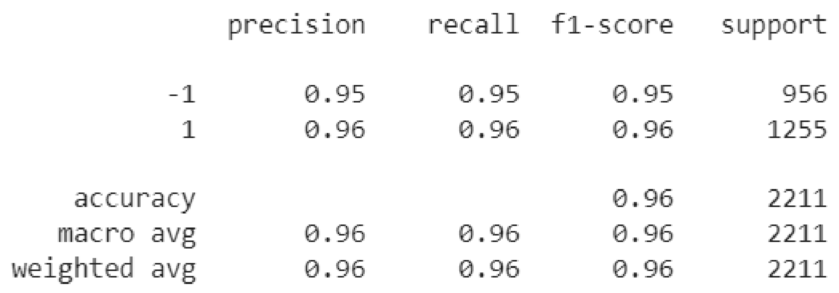


```
              precision    recall  f1-score   support

          -1       0.96      0.94      0.95       956
           1       0.95      0.97      0.96      1255

    accuracy                           0.96      2211
   macro avg       0.96      0.95      0.96      2211
weighted avg       0.96      0.96      0.96      2211
```

Decision tree:



```
              precision    recall  f1-score   support

          -1       0.95      0.95      0.95       956
           1       0.96      0.96      0.96      1255

    accuracy                           0.96      2211
   macro avg       0.96      0.96      0.96      2211
weighted avg       0.96      0.96      0.96      2211
```

10. ADVANTAGES & DISADVANTAGES

The proposed system has the following advantages:

1. The system provides accurate result as to whether the site is safe to use or not.
2. The system prevents users from falling into phishing attacks.

The system also has a few disadvantages as follows,

1. The system has no means to check a complicated url.
2. The system takes some time to give accurate results.

11. CONCLUSION

The proposed system provides the percent of accuracy of the site url given and advices the user whether or not to proceed browsing to that site.

12. FUTURE SCOPE

The website can be extended to store the user's history and make an analysis on what source does most of the phishing links come and can also be made into an extension.

APPENDIX

SOURCE CODE:

FLASK:

```
from flask import Flask, render_template, url_for, redirect,request
from flask_sqlalchemy import SQLAlchemy
from flask_login import UserMixin, login_user, LoginManager, login_required, logout_user,
    current_user
from flask_wtf import FlaskForm
from wtforms import StringField, PasswordField, SubmitField
from wtforms.validators import InputRequired, Length, ValidationError
from flask_bcrypt import Bcrypt

import numpy as np
import pandas as pd
from sklearn import metrics
import warnings
import pickle
warnings.filterwarnings('ignore')
from feature import FeatureExtraction

app = Flask(__name__)
db = SQLAlchemy(app)
bcrypt = Bcrypt(app)
app.config['SQLALCHEMY_DATABASE_URI'] = 'sqlite:///database.db'
app.config['SECRET_KEY'] = 'thisisasecretkey'

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

login_manager = LoginManager()
login_manager.init_app(app)
login_manager.login_view = 'login'



@login_manager.user_loader
```

```
def load_user(user_id):
    return User.query.get(int(user_id))


class User(db.Model, UserMixin):
    id = db.Column(db.Integer, primary_key=True)
    username = db.Column(db.String(20), nullable=False, unique=True)
    password = db.Column(db.String(80), nullable=False)


class RegisterForm(FlaskForm):
    username = StringField(validators=[
                           InputRequired(), Length(min=4, max=20)], render_kw={"placeholder":
      "Username"})

    password = PasswordField(validators=[
                             InputRequired(), Length(min=8, max=20)], render_kw={"placeholder":
      "Password"})

    submit = SubmitField('Register')

    def validate_username(self, username):
        existing_user_username = User.query.filter_by(
            username=username.data).first()
        if existing_user_username:
            raise ValidationError(
                'That username already exists. Please choose a different one.')


class LoginForm(FlaskForm):
    username = StringField(validators=[
                        InputRequired(), Length(min=4, max=20)], render_kw={"placeholder":
      "Username"})

    password = PasswordField(validators=[
                        InputRequired(), Length(min=8, max=20)], render_kw={"placeholder":
      "Password"})

    submit = SubmitField('Login')



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



@app.route('/login', methods=['GET', 'POST'])
def login():
    form = LoginForm()
    if request.method == "POST":
        if form.validate_on_submit():
            user = User.query.filter_by(username=form.username.data).first()
            if user:
                if bcrypt.check_password_hash(user.password, form.password.data):
                    login_user(user)
                    return redirect(url_for('index'))
                    #return redirect(url_for('index'))
                    #return render_template("index.html", form=form)
    return render_template('login.html', form=form)



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

```
def dashboard():
    return render_template('index.html')



@app.route("/predict", methods=['GET', 'POST'])
def predictFunction():
    if request.method == "POST":
        url = request.form["url"]
        print("URL = " + str(url))
        obj = FeatureExtraction(url)
        x = np.array(obj.getFeaturesList()).reshape(1,-1)

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



@app.route('/logout', methods=['GET', 'POST'])
@login_required
def logout():
    logout_user()
    return redirect(url_for('login'))
```

```
@ app.route('/register', methods=['GET', 'POST'])
def register():
    form = RegisterForm()

    if form.validate_on_submit():
        hashed_password = bcrypt.generate_password_hash(form.password.data)
        new_user = User(username=form.username.data, password=hashed_password)
        db.session.add(new_user)
        db.session.commit()
        return redirect(url_for('login'))

    return render_template('register.html', form=form)
if __name__ == "__main__":
    app.run(debug=True)
```

FEATURE.PY:

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

class FeatureExtraction:
```

```
features = []
def __init__(self,url):
    self.features = []
    self.url = url
    self.domain = ""
    self.whois_response = ""
    self.urlparse = ""
    self.response = ""
    self.soup = ""

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

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

    try:
        self.whois_response = whois.whois(self.domain)
    except:
        pass




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


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

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

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



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

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

# 3.shortUrl
def shortUrl(self):
    match                                                                                    =
  re.search('bit\.ly|goo\.gl|shorte\.st|go2l\.ink|x\.co|ow\.ly|t\.co|tinyurl|tr\.im|is\.gd|cli\.gs|'

  'yfrog\.com|migre\.me|ff\.im|tiny\.cc|url4\.eu|twit\.ac|su\.pr|twurl\.nl|snipurl\.com|'

  'short\.to|BudURL\.com|ping\.fm|post\.ly|Just\.as|bkite\.com|snipr\.com|fic\.kr|loopt\.us|'

  'doiop\.com|short\.ie|kl\.am|wp\.me|rubyurl\.com|om\.ly|to\.ly|bit\.do|t\.co|lnkd\.in|'
              'db\.tt|qr\.ae|adf\.ly|goo\.gl|bitly\.com|cur\.lv|tinyurl\.com|ow\.ly|bit\.ly|ity\.im|'
 'q\.gs|is\.gd|po\.st|bc\.vc|twitthis\.com|u\.to|j\.mp|buzurl\.com|cutt\.us|u\.bb|yourls\.org|'

 'x\.co|prettylinkpro\.com|scrnch\.me|filoops\.info|vzturl\.com|qr\.net|1url\.com|tweez\.me|
 v\.gd|tr\.im|link\.zip\.net', self.url)
    if match:
        return -1
    return 1

# 4.Symbol@
def symbol(self):
    if re.findall("@",self.url):
        return -1
    return 1

# 5.Redirecting//
def redirecting(self):
    if self.url.rfind('//')>6:
        return -1
    return 1

# 6.prefixSuffix
def prefixSuffix(self):
    try:
        match = re.findall('\-', self.domain)
        if match:
            return -1
        return 1
    except:
        return -1
```

```
# 7.SubDomains
def SubDomains(self):
    dot_count = len(re.findall("\.", self.url))
    if dot_count == 1:
        return 1
    elif dot_count == 2:
        return 0
    return -1

# 8.HTTPS
def Hppts(self):
    try:
        https = self.urlparse.scheme
        if 'https' in https:
            return 1
        return -1
    except:
        return 1

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

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

# 10. Favicon
def Favicon(self):
    try:
        for head in self.soup.find_all('head'):
            for head.link in self.soup.find_all('link', href=True):
                dots = [x.start(0) for x in re.finditer('\.', head.link['href'])]
                if self.url in head.link['href'] or len(dots) == 1 or domain in head.link['href']:
                    return 1
        return -1
    except:
        return -1

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

# 12. HTTPSDomainURL
def HTTPSDomainURL(self):
    try:
        if 'https' in self.domain:
            return -1
        return 1
    except:
        return -1

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

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

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

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

        try:
            percentage = success/float(i) * 100
            if percentage < 22.0:
                return 1
            elif((percentage >= 22.0) and (percentage < 61.0)):
                return 0
            else:
                return -1
        except:
            return 0
    except:
        return -1

# 14. AnchorURL
def AnchorURL(self):
    try:
        i,unsafe = 0,0
        for a in self.soup.find_all('a', href=True):
            if "#" in a['href'] or "javascript" in a['href'].lower() or "mailto" in a['href'].lower() or
 not (url in a['href'] or self.domain in a['href']):
                unsafe = unsafe + 1
            i = i + 1
        try:
            percentage = unsafe / float(i) * 100
            if percentage < 31.0:
                return 1
            elif ((percentage >= 31.0) and (percentage < 67.0)):
                return 0
            else:
                return -1
        except:
            return -1

    except:
        return -1

# 15. LinksInScriptTags
def LinksInScriptTags(self):
    try:
        i,success = 0,0

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

        for script in self.soup.find_all('script', src=True):
            dots = [x.start(0) for x in re.finditer('\.', script['src'])]
            if self.url in script['src'] or self.domain in script['src'] or len(dots) == 1:
                success = success + 1
            i = i+1
        try:
            percentage = success / float(i) * 100
            if percentage < 17.0:
                return 1
            elif((percentage >= 17.0) and (percentage < 81.0)):
                return 0
            else:
                return -1
        except:
            return 0
    except:
        return -1

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

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

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

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

# 20. StatusBarCust
```

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

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

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

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

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

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

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

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

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

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

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


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

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

```
# 30. StatsReport
def StatsReport(self):
    try:
        url_match = re.search(

  'at\.ua|usa\.cc|baltazarpresentes\.com\.br|pe\.hu|esy\.es|hol\.es|sweddy\.com|myjino\.ru|96\
  .lt|ow\.ly', url)
        ip_address = socket.gethostbyname(self.domain)
        ip_match                                                                              =
  re.search('146\.112\.61\.108|213\.174\.157\.151|121\.50\.168\.88|192\.185\.217\.116|78\.4
  6\.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\.10
  8|107\.151\.148\.109|119\.28\.52\.61|54\.83\.43\.69|52\.69\.166\.231|216\.58\.192\.225|'

  '118\.184\.25\.86|67\.208\.74\.71|23\.253\.126\.58|104\.239\.157\.210|175\.126\.123\.219|
  141\.8\.224\.221|10\.10\.10\.10|43\.229\.108\.32|103\.232\.215\.140|69\.172\.201\.153|'

  '216\.218\.185\.162|54\.225\.104\.146|103\.243\.24\.98|199\.59\.243\.120|31\.170\.160\.61
  |213\.19\.128\.77|62\.113\.226\.131|208\.100\.26\.234|195\.16\.127\.102|195\.16\.127\.157
  |'

  '34\.196\.13\.28|103\.224\.212\.222|172\.217\.4\.225|54\.72\.9\.51|192\.64\.147\.141|198\.
  200\.56\.183|23\.253\.164\.103|52\.48\.191\.26|52\.214\.197\.72|87\.98\.255\.18|209\.99\.1
  7\.27|'

  '216\.38\.62\.18|104\.130\.124\.96|47\.89\.58\.141|78\.46\.211\.158|54\.86\.225\.156|54\.8
  2\.156\.19|37\.157\.192\.102|204\.11\.56\.48|110\.34\.231\.42', ip_address)
        if url_match:
            return -1
        elif ip_match:
            return -1
        return 1
    except:
        return 1

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

WEBSITE:

INDEX.HTML

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="This website is develop for identify the safety
      of url.">
    <meta name="keywords" content="phishing url,phishing,cyber security,machine
      learning,classifier,python">
    <meta name="author" content="VAIBHAV BICHAVE">

    <!-- BootStrap -->
    <link                                                    rel="stylesheet"
      href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.0/css/bootstrap.min.css"
      integrity="sha384-
     9aIt2nRpC12Uk9gS9baDl411NQApFmC26EwAOH8WgZl5MYYxFfc+NcPb1d
     KGj7Sk" crossorigin="anonymous">

    <link href="{{ url_for("static" , filename="styles.css") }}" rel="stylesheet" />
    <title>URL detection</title>

</head>

<body>

<div class=" container">
    <div class="row">
        <div class="form col-md" id="form1">
            <h2>PHISHING URL DETECTION</h2>

            <br>
            <form action="/predict" method ="post">
                <input    type="text"    class="form__input"    name    ='url'    id="url"
     placeholder="Enter URL" required="" />
                <label for="url" class="form__label">URL</label>
                <button class="button" role="button" >Check here</button>
            </form>

    </div>

    <div class="col-md" id="form2">

        <br>
        <h6 class = "right "><a href= {{ url }} target="_blank">{{ url }}</a></h6>


        <br>
        <h3 id="prediction"></h3>
        <button            class="button2"            id="button2"            role="button"
      onclick="window.open('{{url}}')"      target="_blank"      >Still      want      to
      Continue</button>
        <button            class="button1"            id="button1"            role="button"
      onclick="window.open('{{url}}')" target="_blank">Continue</button>
    </div>
</div>
<br>

</div>

    <!-- JavaScript -->
    <script src="https://code.jquery.com/jquery-3.5.1.slim.min.js"
        integrity="sha384-
      DfXdz2htPH0lsSSs5nCTpuj/zy4C+OGpamoFVy38MVBnE+IbbVYUew+OrCXa
      Rkfj"
        crossorigin="anonymous"></script>
    <script src="https://cdn.jsdelivr.net/npm/popper.js@1.16.0/dist/umd/popper.min.js"
        integrity="sha384-
      Q6E9RHvbIyZFJoft+2mJbHaEWldlvI9IOYy5n3zV9zzTtmI3UksdQRVvoxMfoo
      Ao"
        crossorigin="anonymous"></script>
    <script src="https://stackpath.bootstrapcdn.com/bootstrap/4.5.0/js/bootstrap.min.js"
```

```
      integrity="sha384-
    OgVRvuATP1z7JjHLkuOU7Xw704+h835Lr+6QL9UvYjZE3Ipu6Tp75j7Bh/kR0
    JKI"
      crossorigin="anonymous"></script>


    <script>

          let x = '{{xx}}';
          let num = x*100;
          if (0<=x && x<0.50){
              num = 100-num;
          }
          let txtx = num.toString();
          if(x<=1 && x>=0.50){
              var label = "Website is "+txtx +"% safe to use...";
              document.getElementById("prediction").innerHTML = label;
              document.getElementById("button1").style.display="block";
          }
          else if (0<=x && x<0.50){
              var label = "Website is "+txtx +"% unsafe to use..."
              document.getElementById("prediction").innerHTML = label ;
              document.getElementById("button2").style.display="block";
          }

    </script>

</body>
```

```
</html>
LOGIN.HTML
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
     <link href="{{ url_for("static" , filename="styles.css") }}" rel="stylesheet" />
     <title>URL detection</title>
</head>

<body>
    <h1>Login Page</h1>

    <form method="POST" action="/index">
        {{ form.hidden_tag() }}
        {{ form.username }}
        {{ form.password }}
        {{ form.submit }}
    </form>

    <a href="{{ url_for('register') }}">Don't have an account? Sign Up</a>
</body>

</html>
REGISTER.HTML
<!DOCTYPE html>
```

```
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Register</title>
</head>

<body>
    <h1>Register Page</h1>

    <form method="POST" action="">
        {{ form.hidden_tag() }}
        {{ form.username }}
        {{ form.password }}
        {{ form.submit }}
    </form>

    <a href="{{ url_for('login') }}">Already have an account? Log In</a>
</body>

</html>
```

HOME.HTML

```
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link rel="stylesheet" href="static/style.css">
    <title>Web Phishing Detection Site</title>


</head>

<body background-color="blue">

<div class="heading" >
    <h1 class="mb-3">WEB PHISHING DETECTION SITE</h1>
</div>

<div class="container d-flex align-items-center justify-content-center">
    <!--<a href="{{ url_for('login') }}">Login Page</a><br>-->
    <a href="/login">Login Page</a><br>
    <a href="{{ url_for('register') }}">Register Page</a><br>
</div>
</body>

</html>
```

GITHUB LINK:

https://github.com/IBM-EPBL/IBM-Project-9309-1658993148

PROJECT DEMO LINK:

https://drive.google.com/file/d/1cFwZmx2rSe1eM8Qpzf3InMhw4uc5c2xz/view?usp=sharing

REFERENCES:

1. V. Patil, P. Thakkar, C. Shah, T. Bhat and S. P. Godse, "Detection and Prevention of Phishing Websites Using Machine Learning Approach," 2018 Fourth International Conference on Computing Communication Control and Automation (ICCUBEA), 2018, pp. 1-5, doi: 10.1109/ICCUBEA.2018.8697412.

2. A. Alswailem, B. Alabdullah, N. Alrumayh and A. Alsedrani, "Detecting Phishing Websites Using Machine Learning," 2019 2nd International Conference on Computer Applications & Information Security (ICCAIS), 2019, pp. 1-6, doi: 10.1109/CAIS.2019.8769571.

3. M. Korkmaz, O. K. Sahingoz and B. Diri, "Detection of Phishing Websites by Using Machine Learning-Based URL Analysis," 2020 11th International Conference on Computing, Communication and Networking Technologies (ICCCNT), 2020, pp. 1-7, doi: 10.1109/ICCCNT49239.2020.9225561.

4. Q. A. Al-Haija and A. A. Badawi, "URL-based Phishing Websites Detection via Machine Learning," 2021 International Conference on Data Analytics for Business and Industry (ICDABI), 2021, pp. 644-649, doi: 10.1109/ICDABI53623.2021.9655851.

5. B. Geyik, K. Erensoy and E. Kocyigit, "Detection of Phishing Websites from URLs by using Classification Techniques on WEKA," 2021 6th International Conference on Inventive Computation Technologies (ICICT), 2021, pp. 120-125, doi: 10.1109/ICICT50816.2021.9358642