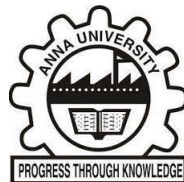


IBM
NALAIYA THIRAN
PROJECT REPORT
ON
WEB PHISHING DETECTION

TEAM ID: PNT2022TMID29369

UNIVERSITY COLLEGE OF ENGINEERING, VILLUPURAM



Team members

CHANDRU S
ARUNKUMAR A
ASHOK E
SRIHARI S
PURSOOTHAMAN R

TABLE OF CONTENTS

1. INTRODUCTION

- 1.1 Project Overview
- 1.2 Purpose

2. LITERATURE SURVEY

- 2.1 Existing Problem
- 2.2 References
- 2.3 Problem Statement Definition

3. IDEATION AND PROPOSED SOLUTION

- 3.1 Empathy Map Canvas
- 3.2 Ideation & Brainstroming
- 3.3 Proposed Solution
- 3.4 Problem Solution fit

4. REQUIREMENT ANALYSIS

- 4.1 Functional Requirements
- 4.2 Non-Functional Requirements

5. PROJECT DESIGN

- 5.1 Data Flow Diagrams
- 5.2 Solution & Technical Architecture
- 5.3 User Stories

6. PROJECT PLANNING AND SCHEDULING

- 6.1 Sprint Planning & Estimation
- 6.2 Sprint Delivery Schedule
- 6.3 Reports from JIRA

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

- 7.1 Feature 1
- 7.2 Feature 2
- 7.3 Database Schema(If Applicable)

8. TESTING

- 8.1 Test Cases
- 8.2 User Acceptance Testing

9. RESULTS

- 9.1 Performance Metrics

10. ADVANTAGES AND DISADVANTAGES

11. CONCLUSION

12. FUTURE SCOPE

13. APPENDIX

- Source code
- Github and demo link

ABSTRACT

Phishing is the most commonly used social engineering and cyber attack. Through such attacks, the phisher targets naive online users by tricking them into revealing confidential information, with the purpose of using it fraudulently. In order to avoid getting phished, Users should have awareness of phishing websites. Have a blacklist of phishing websites which requires the knowledge of website being detected as phishing. Detect them in their early appearance, using machine learning and deep neural network algorithms. Of the above three, the machine learning based method is proven to be most effective than the other methods. A phishing website is a common social engineering method that mimics trustful uniform resource locators (URLs) and webpages. The objective of this project is to train machine learning models and deep neural nets on the dataset created to predict phishing websites. Both phishing and benign URLs of websites are gathered to form a dataset and from them required URL and website content-based features are extracted. The performance level of each model is measured and compared.

Keywords: Deep learning, Machine learning, Phishing website attack, Phishing website detection, Anti-phishing website, Legitimate website , Phishing website datasets, Phishing website features.

WEB PHISHING DETECTION

1 INTRODUCTION

1.1 Project Overview

Social engineering attack is a common security threat used to reveal private and confidential information by simply tricking the users without being detected. The main purpose of this attack is to gain sensitive information such as username, password and account numbers. According to, phishing or web spoofing technique is one example of social engineering attack. Phishing attack may appear in many types of communication forms such as messaging, SMS, VOIP and fraudster emails. Users commonly have many user accounts on various websites including social network, email and also accounts for banking. Therefore, the innocent web users are the most vulnerable targets towards this attack since the fact that most people are unaware of their valuable information, which helps to make this attack successful. Typically phishing attack exploits the social engineering to lure the victim through sending a spoofed link by redirecting the victim to a fake web page. The spoofed link is placed on the popular web pages or sent via email to the victim. The fake webpage is created similar to the legitimate webpage. Thus, rather than directing the victim request to the real web server, it will be directed to the attacker server. The current solutions of antivirus, firewall and designated software do not fully prevent the web spoofing attack. The implementation of Secure Socket Layer (SSL) and digital certificate (CA) also does not protect the web user against such attack. In web spoofing attack, the attacker diverts the request to fake web server. In fact, a certain type of SSL and CA can be forged while everything appears to be legitimate. According to, secure browsing connection does virtually nothing to protect the users especially from the attackers that have knowledge on how the “secure” connections actually work. This paper develops an anti-web spoofing solution based on inspecting the URLs of fake web pages. This solution developed series of steps to check characteristics of websites Uniform Resources Locators (URLs).

1.2 Purpose

This section describes the proposed model of phishing attack detection. The proposed model focuses on identifying the phishing attack based on checking phishing websites features, Blacklist and WHOIS database. According to few selected features can be used to differentiate between legitimate and spoofed web pages. These selected features are many such as URLs, domain identity, security & encryption, source code, page style and contents, web address bar

and social human factor. This study focuses only on URLs and domain name features. Features of URLs and domain names are checked using several criteria such as IP Address, long URL address, adding a prefix or suffix, redirecting using the symbol “//”, and URLs having the symbol “@”. These features are inspected using a set of rules in order to distinguish URLs of phishing webpages from the URLs of legitimate websites.

2. LITERATURE SURVEY

A literature survey is an insightful article that presents the existing information including considerable discoveries just as theoretical and methodological commitments to a specific topic. A very effective detection of phishing website model which is focused on optimal feature selection technique and also based on neural network (OFS-NN) is proposed [23]. In this proposed model, an index called feature validity value(FVV) has been generated to check the effects of all those features on the detection of such websites. Now, based on this newly generated index, an algorithm is developed to find from the phishing websites, the optimal features. This selected algorithm will be able to overcome the problem of over-fitting of the neural network to a great extent. These optimal features are then used to build an optimal classifier that detects phishing URLs by training the neural network. A theory called Fuzzy Rough Set(FRS) [24] was devised to a tool that finds the most appropriate features from a few standardised dataset. These features are then sent to a few classifiers for detection of phishing. To investigate the feature selection for FRS in building a generalized detection of phishing, the models by a different dataset of 14,000 website samples are trained. Feature engineering plays a vital role in finding solutions for detection of phishing websites, although the accuracy of the model greatly will be based on knowledge of the features. though the features taken from all these various dimensions are understandable, the limitation lies in the time taken to collect these features. To fix this drawback, the authors have proposed a multidimensional phishing detection feature [25] approach that concentrates on a rapid detection technique by making use of deep learning (MFPD) To detect phishing occurrence accurately, a three phase detection called Web Crawler based Phishing Attack Detector (WC-PAD) [26] has been proposed. This takes the web's content, traffic and URL as input features. Now considering these features, classification is done. PhishingNet [27], is an approach based on deep learning for detecting phishing URLs in a timely manner.

PhishingNet [27], is an approach based on deep learning for detecting phishing URLs in a timely manner. A detection system was developed which can match the dynamic environment and phishing websites. Because the approach considers various types of distinctive features

from source code of webpages and URLs [28], this is a fully client side solution and needs no support of a third party. A method called parse tree validation [29] has been proposed to find if a webpage is phishing or legitimate. This is an innovative approach to find such web sites by intercepting every hyperlinks of a present page through API of Google, and developing a parse tree from all those hyperlinks that were intercepted. In this, parsing begins from the root node. It goes by the Depth-FirstSearch (DFS) algorithm to determine if any child node has the same value as the root node. A model as a solution was the focus in a study [30] that uses Random Forest classifier for detection of phishing websites by URL method. An approach that combines to form an online tool, the collection, validation and detection of phishing websites. [31]. This online tool monitors in real-time the blacklist of PhishTank, validates and detects phishing website. A framework was developed, known as "Fresh-Phish" [32], that generates for phishing websites, present machine learning data. By using 30 various features of website which can be queried using Python, a very large dataset is built and the various ML classifiers are analyzed against this generated dataset to find out which has highest accuracy. This model analyzes both the accuracy as well as the time taken by the model to train. A determined bond was built between the content-based heuristics and the authenticity of the website by evaluating both the phishing and legitimate websites' training set. A framework called Phishing-Detective is presented [33] which detects the websites as phishing based on existing heuristics as well as new heuristics. An productive way using C4.5 decision tree classifier [34] as well as certain features of the URL was proposed to detect websites that are phishing. There are many schemes for detection of phishing websites, among which the visual similarity scheme is collecting glances. The screenshot of the website is taken and stored in a database. It checks if the input screenshot of the website is same as the one stored in the database. If yes, then that website is predicted as phishing. But, if there are several similar websites, which ever is the first website that is given as input is taken as legitimate. Hence, it cannot predict correctly the authentic website and therefore recognising the goal website becomes tedious [35]. This detection method is proposed with target website finder by making use of images and CSS.

2.2 References

1. Phishing Activity Trends Report: 4rd Quarter 2020. *Anti-Phishing Work. Group*. Retrieved April **2021**, 30, 2020.
2. FBI. 2019 Internet Crime Report Released-FBI. Available online: <https://www.fbi.gov/news/stories/2019-internet-crime-report-released-021120>. (accessed on 11 February 2020).
3. Mohammad, R.M.; Thabtah, F.; McCluskey, L. Tutorial and critical analysis of phishing websites methods. *Comput. Sci. Rev.* **2015**, *17*, 1–24. [[Google Scholar](#)] [[CrossRef](#)] [[Green Version](#)]
4. Almomani, A.; Wan, T.C.; Altaher, A.; Manasrah, A.; ALmomani, E.; Anbar, M.; ALomari, E.; Ramadass, S. Evolving fuzzy neural network for phishing emails detection. *J. Comput. Sci.* **2012**, *8*, 1099. [[Google Scholar](#)]
5. Prakash, P.; Kumar, M.; Kompella, R.R.; Gupta, M. Phishnet: Predictive blacklisting to detect phishing attacks. In Proceedings of the 2010 Proceedings IEEE INFOCOM, San Diego, CA, USA, 14–19 March 2010; pp. 1–5. [[Google Scholar](#)]
6. Zhang, J.; Porras, P.A.; Ullrich, J. Highly Predictive Blacklisting. In Proceedings of the USENIX Security Symposium, San Jose, CA, USA, 28 July–1 August 2008; pp. 107–122. [[Google Scholar](#)]
7. Cao, Y.; Han, W.; Le, Y. Anti-phishing based on automated individual white-list. In Proceedings of the 4th ACM Workshop on Digital Identity Management, Alexandria, VA, USA, 31 October 2008; pp. 51–60. [[Google Scholar](#)]
8. Srinivasa Rao, R.; Pais, A.R. Detecting phishing websites using automation of human behavior. In Proceedings of the 3rd ACM Workshop on Cyber-Physical System Security, Abu Dhabi, United Arab Emirates, 2–4 April 2017; pp. 33–42. [[Google Scholar](#)]

2.3 Problem Statement Definition

The problem is derived after making a thorough observation and study about the method of classification of phishing websites that makes use of machine learning techniques. We must design a system that should allow us to:

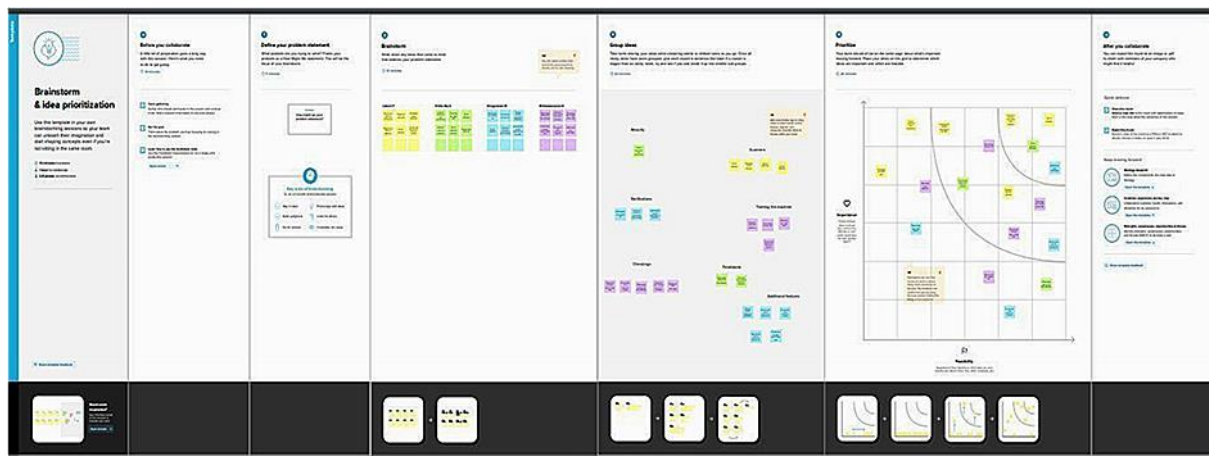
- Accurately and efficiently classify the websites into legitimate or phishing.
- Time consumed for detection should be less and should be cost effective.

3 IDEATION & PROPOSED SOLUTION

3.1 Empathy Map Canvas



3.2 Ideation & Brainstorming:



3.3 Proposed Solution:

S.NO	Parameter	Description
1.	Problem Statement (Problem to be solved)	Phishing is a major problem, which uses both social engineering and technical deception to get users' important information such as financial data, emails, and other private information. Phishing exploits human vulnerabilities; therefore, most protection protocols cannot prevent the whole phishing attacks. Many of them use the blacklist whitelist approach, however, this cannot detect zero-hour phishing attacks, and they are not able to detect new types of phishing attacks.
2.	Idea/Solution Description	A common way to obtain phishing detection measurements is to perform an assessment. To continue with the phishing detection example, a measurement of how many suspicious e-mails were reported to security would be collected at the end of each phishing assessment. If you're using a commercial tool, the number of e-mails sent during each assessment is available from the reporting screen.
3.	Novelty/uniqueness	Using the Machine learning technology to detect the phishing attacks Machine learning is one of the critical mechanisms working in tandem with Artificial Intelligence (AI). It is based on algorithms focused on understanding and recognizing patterns from enormous piles of data to create a system that can predict unusual behaviours and anomalies. It evolves with time while learning patterns of normal behaviours. These characteristics make it helpful in identifying phishing emails, spam, and malware.
4.	social Impact / Customer Satisfaction	Millions of people are being affected and billions of dollars are getting stolen. Phishing is a technique used to extract personal information from victims by means of deceptive and fraudulent emails for identity theft. As a result of this, the organizations as well consumers

		are facing enormous social effects. Phishing is causing two-way damage.
5.	Business model(Revenue model)	Tool has been used to import Machine learning algorithms. Each classifier is trained using training set and testing set is used to evaluate performance of classifiers. Performance of classifiers has been evaluated by calculating classifier's accuracy score. improve the accuracy of our models with better feature extraction
6.	Scalability of the solution	For future enhancements, we intend to build the phishing detection system as a scalable web service which will incorporate online learning so that new phishing attack patterns can easily be learned and improve the accuracy of our models with better feature extraction.

3.4 Problem Solution Fit:

Essentially, the problem solution fit means that you have found a problem with a customer and that the solution you have realized solves it for them. By Identifying behavioural patterns, it helps entrepreneurs , marketers and corporate innovators identify what and why.

Purpose:

- ☐ Find a way to solve complex problems in a manner that is appropriate for the customer's situation
- ☐ Utilize existing channels and media to gain more adoption of your solution.
- ☐ Use the right triggers and messaging to refine your communication and marketing strategy
- ☐ Understand the existing situation in order to improve it for your target group

Define CS, fit into CC	1. CUSTOMER SEGMENT(S) a. Users who buy products on-line and make payments via e-banking. b. Sensitive data will be shared through these kind of websites.	6. CUSTOMER CONSTRAINTS a. Not being able to see the main process of the transaction site, they will not be able to know the real nature of the site. b. sense of insecurity when faced with constraints. c. Not knowing how to protect them and identify malicious	5. AVAILABLE SOLUTIONS a. The above solutions check if the website is available in the legitimate websites list, but have property limitations such as exact name and adding items to the list frequently. b. Other ML model solution predictions are based on the content of the URL instead of its properties	Explore AS, differentiate
Focus on JSP, JS, HTML, CSS, JavaScript	2. JOBS-TO-BE-DONE / PROBLEMS a. Websites with link that contain malware. b. Saying that they've noticed some suspicious activity or log-in attempts. c. Claim there is a problem with your account or your payment information. d. Want you to click on a link to make a payment, but the link has malware.	9. PROBLEM ROOT CAUSE a. Attackers keep fooling people by spoofing original sites. b. They use their knowledge on the domain for cheating and other bad intentions. c. Common people will not have much knowledge on this domain. They find it harder just to use the web service.	7. BEHAVIOUR a. Users need to be more aware about what information they provide to the sites. b. They should not believe any site they visit even if they look the legitimate ones.	Focus on JSP, JS, HTML, CSS, JavaScript

3. TRIGGERS a. Loss of money b. Loss of intellectual property. c. Damage to reputation d. Disruption of operational activities	10. YOUR SOLUTION A deep learning-based framework by implementing it as a browser plug-in capable of determining whether there is a phishing risk in real-time when the user visits a web page and gives a warning message. The real-time prediction includes whitelist filtering, blacklist interception, and machine learning (ML) prediction. To deal with phishing attacks and distinguishing the phishing webpages automatically, Blacklist based detection technique keeps a list of websites' URLs that are categorized as phishing sites. If a web-page requested by a user exists in the formed list, the connection to the queried website is blocked. Machine Learning (ML) based approaches rely on classification algorithms such as Support Vector Machines (SVM) and Decision Trees (DT) to train a model that can later automatically classify the fraudulent websites at run-time without any human	8. CHANNELS of BEHAVIOUR 8.1 ONLINE Enter the URL and predicts the user 8.2 OFFLINE a. Offline b. Checks the site already available legitimate sites list. c. Stores the phishing site to another list..
4. EMOTIONS: BEFORE / AFTER BEFORE: ✓ Stressed ✓ Fear ✓ Frustrated ✓ Confused AFTER : ✓ Confident ✓ Safe ✓ Peace ✓ Happy		

4. REQUIREMENT ANALYSIS

4.1 Functional Requirements

A function of software system is defined in functional requirement and the behavior of the system is evaluated when presented with specific inputs or conditions which may include calculations, data manipulation and processing and other specific functionality.

Our system should be able to load air quality data and preprocess data.

- It should be able to analyze the air quality data.
- It should be able to group data based on hidden patterns.
- It should be able to assign a label based on its data groups.
- It should be able to split data into trainset and testset.
- It should be able to train model using trainset.

- It must validate trained model using testset.
- It should be able to display the trained model accuracy.
- It should be able to accurately predict the air quality on unseen data.

4.2 Non-Functional Requirements

Non-functional requirements describe how a system must behave and establish constraints of its functionality. This type of requirements is also known as the system's quality attributes. Attributes such as performance, security, usability, compatibility are not the feature of the system, they are a required characteristic. They are "developing" properties that emerge from the whole arrangement and hence we can't compose a particular line of code to execute them. Any attributes required by the customer are described by the specification. We must include only those requirements that are appropriate for our project.

Some Non-Functional Requirements are as follows:

- Reliability
- Maintainability
- Performance
- Portability
- Scalability
- Flexibility

Some of the quality attributes are as follows:

ACCESSIBILITY

Availability is a general term used to depict how much an item, gadget, administration, or condition is open by however many individuals as would be prudent. In our venture individuals who have enrolled with the cloud can get to the cloud to store and recover their information with the assistance of a mystery key sent to their email ids. UI is straightforward and productive and simple to utilize.

MAINTAINABILITY

In programming designing, viability is the simplicity with which a product item can be altered so as to:

- Correct absconds

• Meet new necessities New functionalities can be included in the task based the client necessities just by adding the proper documents to existing venture utilizing ASP.net and C# programming dialects.

Since writing computer programs is extremely straightforward, it is simpler to discover and address the imperfections and to roll out the improvements in the undertaking.

SCALABILITY

Framework is fit for taking care of increment all out throughput under an expanded burden when assets (commonly equipment) are included. Framework can work ordinarily under circumstances, for example, low data transfer capacity and substantial number of clients.

PORTABILITY

Convey ability is one of the key ideas of abnormal state programming. Convenient is the product code base component to have the capacity to reuse the current code as opposed to making new code while moving programming from a domain to another. Venture can be executed under various activity conditions to meet its base setups. Just framework records and dependent congregations would need to be designed in such a case. The functional requirements for a system describe what the system should do. Those requirements depend on the type of software being developed, the expected users of the software. These are the statements of services the system should provide, how the system should react to particular inputs and how the system should behave in particular situation.

- Extracting data from CSV files
- Cleaning the data.
- Vector Representation.

Non-functional requirements is not about functionality or behaviour of system, but rather are used to specify the capacity of a system. They are more related to properties of system such as quality, reliability and quick response time. Non- functional requirements come up via customer needs, because of budget, interoperability need such as software and hardware requirement, organizational policies or due to some external factors such as:-

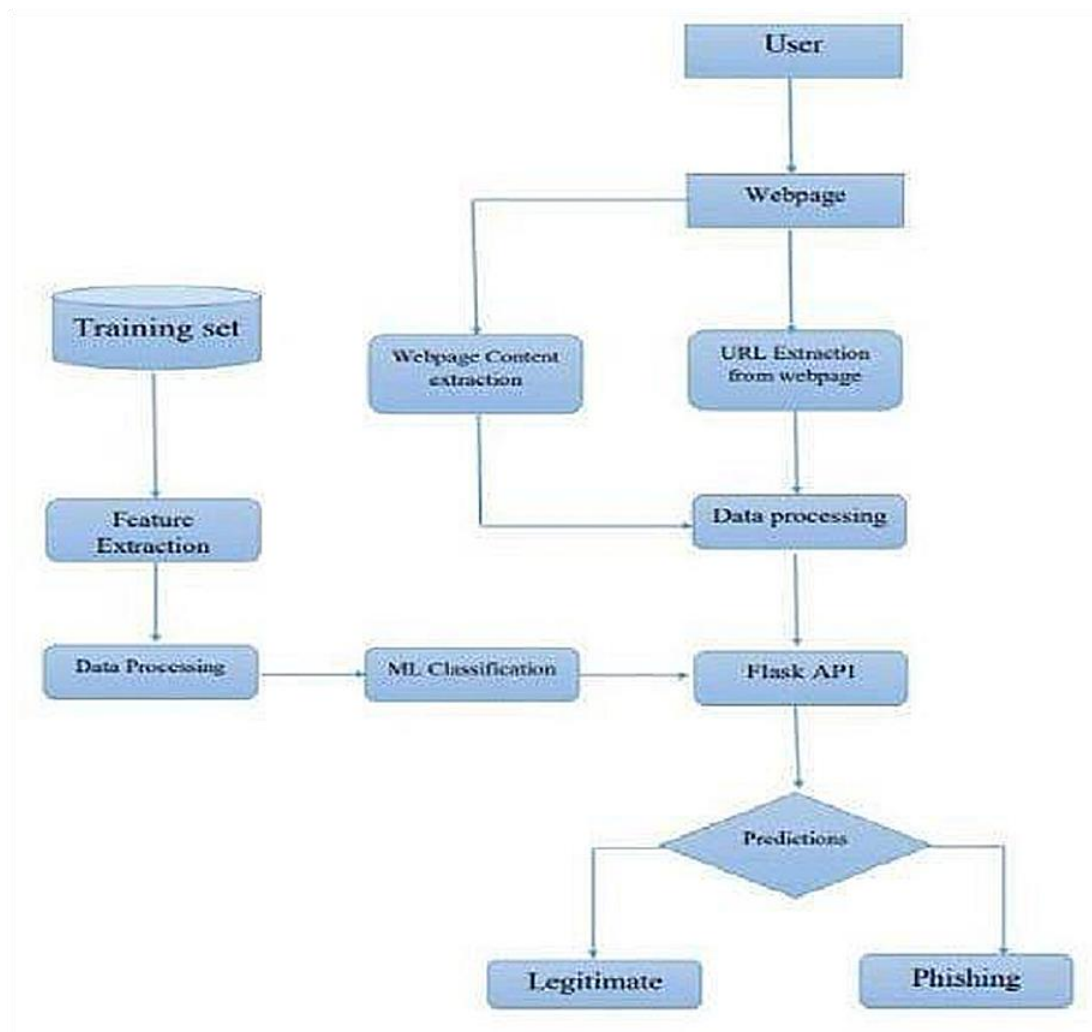
- Basic Operational Requirement
- Organizational Requirement
- Product Requirement
- User Requirement

5. PROJECT DESIGN

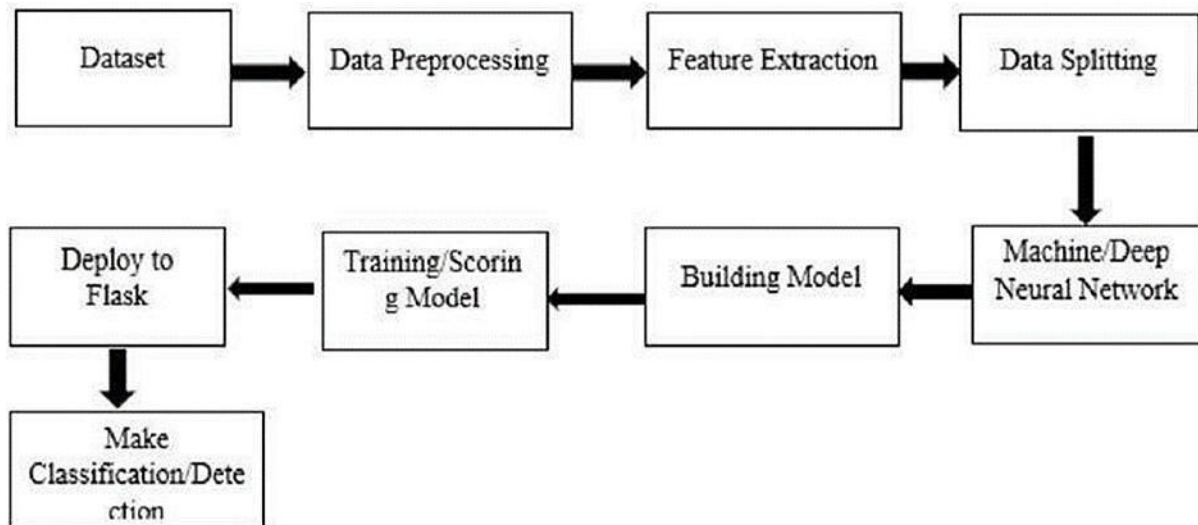
5.1 Data Flow Diagrams

A Data Flow Diagram (DFD) is a traditional visual representation of the information flows within a system. A neat and clear DFD can depict the right amount of the system requirement graphically. It shows how data enters and leaves the system, what changes the information, and where data is stored.

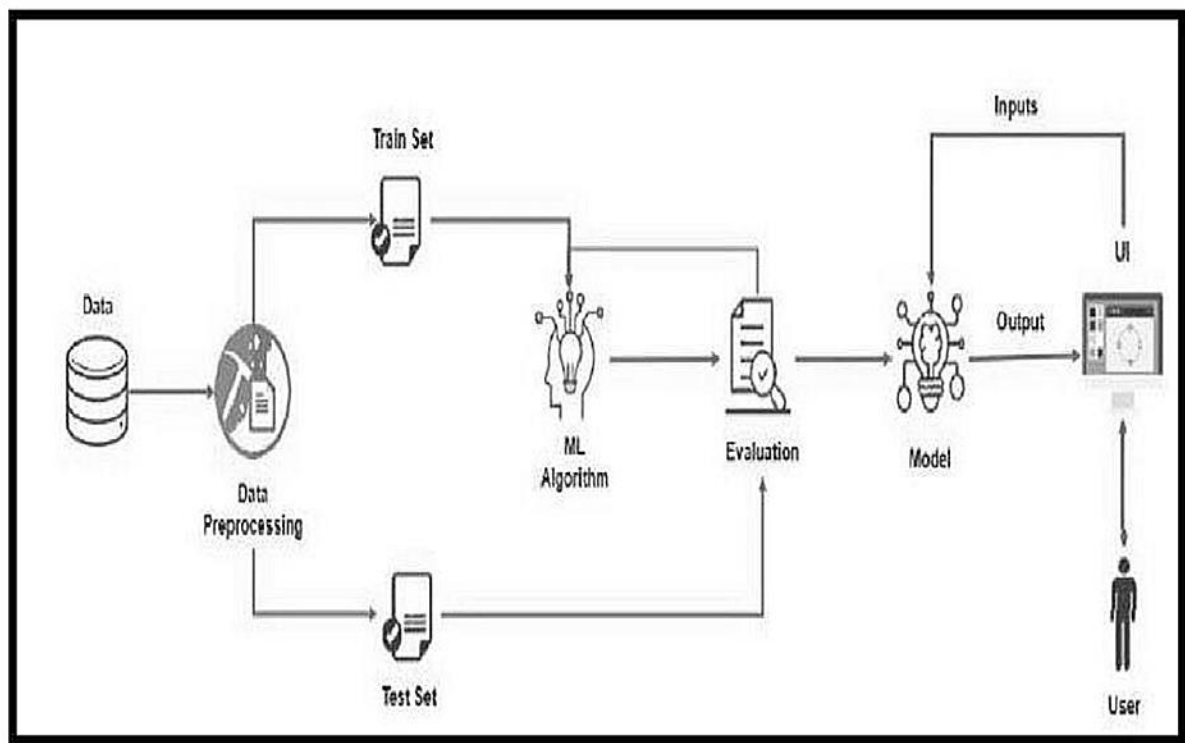
5.2 Solution And Technical Architecture



Solution Architecture



MODEL FOR WEB PHISHING DETECTION



5.3 User Stories

Use the below template to list all the user stories for the product.

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	After i compare in case if none found on comparison then we can extract feature using heuristic and visual similarity approach.	As a User i can have comparison between websites for security.	High	Sprint-1
Administrator	Prediction	USN-1	Here the Model will predict the URL websites using Machine Learning algorithms such as Logistic Regression, KNN	In this i can have correct prediction on the particular algorithms	High	Sprint-1
	Classifier	USN-2	Here i will send all the model output to classifier in order to produce final result.	I this i will find the correct classifier for producing the result	Medium	Sprint-2

6 PROJECT PLANNING AND SCHEDULING

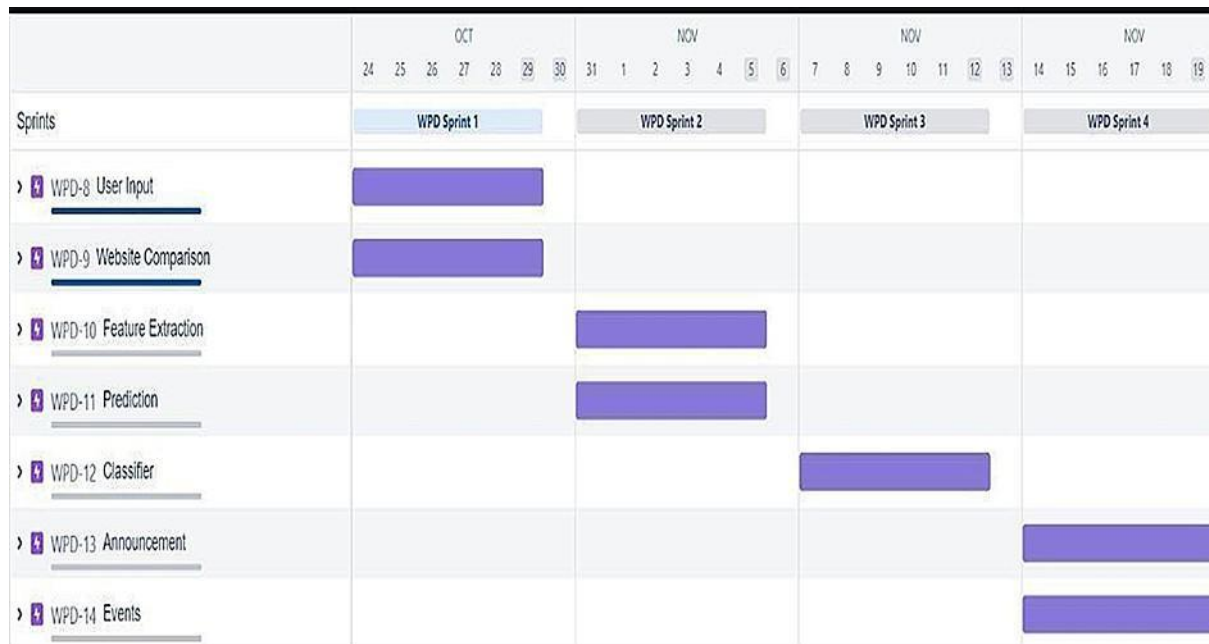
6.1 Sprint Planning And Estimation

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	High	VIJAY N
Sprint-1	Website Comparison	USN-2	Model compares the websites using Blacklist and Whitelist approach.	1	High	BAVYABAI S
Sprint-2	Feature Extraction	USN-3	After comparison, if none found on comparison then it extract feature using heuristic and visual similarity.	2	High	ARUN R
Sprint-2	Prediction	USN-4	Model predicts the URL using Machine learning algorithms such as logistic Regression, KNN.	1	Medium	AJITH D
Sprint-3	Classifier	USN-5	Model sends all the output to the classifier and produces the final result.	1	Medium	VIJAY N
Sprint-4	Announcement	USN-6	Model then displays whether the website is legal site or a phishing site.	1	High	BAVYABAI S
Sprint-4	Events	USN-7	This model needs the capability of retrieving and displaying accurate result for a website.	1	High	AJITH D

6.2 Sprint Delivery Schedule

Sprint	Total Story Points	Duration	Sprint Start Date	Sprint End Date (Planned)	Story Points Completed (as on Planned End Date)	Sprint 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



7 CODING AND SOLUTION

7.1 Feature 1

#app.py:

```
import numpy as np
import pandas as pd
from flask import Flask, request, jsonify, render_template, redirect
from sklearn.preprocessing import MinMaxScaler
from sklearn.metrics import accuracy_score, confusion_matrix
import pickle
import feature
import sys
import joblib
import logging

app = Flask(__name__)

model = pickle.load(open('./Phishing.pkl', 'rb'))
@app.route('/') #decorator
def phishing_detection():
    return render_template('index.html')

@app.route('/y_predict', methods=['POST'])
def y_predict():
    url = request.form['url']
    checkprediction = feature.main(url)
    prediction = model.predict(checkprediction)
    print(prediction)
    output=prediction[0]
    if(output==1):
        pred="your website is legitimate."
    elif(output==-1):
        pred="Wrong Website"
```

```

return render_template('./index.html',prediction_text='{ }'.format(pred),url=url)

@app.route('/predict_api',methods=['POST'])
def predict_api():
    data = request.get_json(force=True)
    prediction = model.y_predict([np.array(list(data.values()))])

    output= prediction[0]
    return jsonify(output)

# if __name__ == '__main__':
#     app.debug=True
#     app.run(host='0.0.0.0', port=5000)
# @app.route('/predict/', methods=['GET','POST'])
# @login_required
# def predict():
#     if request.method == 'POST':
#         url = request.form['url']
#         checkprediction = inputScript.main(url)
#         print(url)
#         print(checkprediction)
#         prediction = model.predict(checkprediction)
#         print(prediction)
#         output=prediction[0]
#         if(output==1):
#             pred="Safe,legitimate link"

#         else:
#             pred="Malicious URL alert!"
#         if(session and session['logged_in']):
#             if(session['logged_in']==True):

```

```

#         return render_template('index.html',userInfo=session['user'],pred=pred)
#     else:
#         return render_template('./templates/prediction-result.html',pred=pred)
#     else:
#         return render_template('./templates/prediction-result.html',pred=pred)
#     elif request.method == 'GET':
#         return render_template('index.html',userInfo=session['user'])
if __name__ == '__main__':
    app.run(host='127.0.0.1', debug=True)

```

#feature.py:

```

# import regex
# from tldextract import extract
# import ssl
# import socket
# from bs4 import BeautifulSoup
# import urllib.request
# import whois
# import datetime
# import requests
# import favicon
# import re
# import google
# from googlesearch import search

# #checking 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


# #Checking for the URL length. Returns 1 (Legitimate) if the URL length is less than 54
characters

# #Returns 0 if the length is between 54 and 75

# #Else returns -1;

# def URLURL_Length (url):

#     length=len(url)

#     if(length<=75):

#         if(length<54):

#             return 1

#         else:

#             return 0

#     else:

#         return -1


# #Checking with the shortening URLs.

# #Returns -1 if any shortening URLs used.

# #Else returns 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|lurl\.com|tweez\.me|v\.g
d|tr\.im|link\.zip\.net',url)

# if match:
#     return -1
# else:
#     return 1

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

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

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

# #Checking for - in Domain. Returns -1 if '-' is found else returns 1.
# def Prefix_Suffix(url):

```

```

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

# #checking the Subdomain. Returns 1 if the subDomain contains less than 1 '.'
# #Returns 0 if the subDomain contains less than 2 '.'
# #Returns -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 response 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  $\leq 1$  year returns -1, otherwise returns 1

# def Domain_registration_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.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, 'xml')
#     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, 'xml')

#     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, 'xml')
#         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  $\leq 1$  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, 'xml')

#         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(), "xml").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, 'xml')
#     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\|swedd
y\|com\|myjino\|ru\|96\|lt\|ow\|ly',url)
#     try:
#         ip_address = socket.gethostbyname(hostname)
#
# ip_match=regex.search('146\|112\|61\|108\|213\|174\|157\|151\|121\|50\|168\|88\|192\|185\|217\|
116\|78\|46\|211\|158\|181\|174\|165\|13\|46\|242\|145\|103\|121\|50\|168\|40\|83\|125\|22\|219\|4
6\|242\|145\|98\|107\|151\|148\|44\|107\|151\|148\|107\|64\|70\|19\|203\|199\|184\|144\|27\|107\|1
51\|148\|108\|107\|151\|148\|109\|119\|28\|52\|61\|54\|83\|43\|69\|52\|69\|166\|231\|216\|58\|192\|
225\|118\|184\|25\|86\|67\|208\|74\|71\|23\|253\|126\|58\|104\|239\|157\|210\|175\|126\|123\|219\|

```

```
141\,8\,224\,221|10\,10\,10\,10|43\,229\,108\,32|103\,232\,215\,140|69\,172\,201\,153|216\,21
8\,185\,162|54\,225\,104\,146|103\,243\,24\,98|199\,59\,243\,120|31\,170\,160\,61|213\,19\,12
8\,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\,25
3\,164\,103|52\,48\,191\,26|52\,214\,197\,72|87\,98\,255\,18|209\,99\,17\,27|216\,38\,62\,18|1
04\,130\,124\,96|47\,89\,58\,141|78\,46\,211\,158|54\,86\,225\,156|54\,82\,156\,19|37\,157\,19
2\,102|204\,11\,56\,48|110\,34\,231\,42',ip_address)
```

```
# except:
```

```
#     return -1
```

```
# if url_match:
```

```
#     return -1
```

```
# else:
```

```
#     return 1
```

```
# #returning scrapped data to calling function in app.py
```

```
# def main(url):
```

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

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

```
#
Domain_registration_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
```

```
import regex
```

```
from tldextract import extract
```

```
import ssl
import socket
from bs4 import BeautifulSoup
import urllib.request
import whois
import datetime
```

```
def url_having_ip(url):
    symbol = regex.findall(r'(http((s)?):/)(((((\d)+).)*)((\w)+)/((\w)+))?'@',url)
    if(len(symbol)!=0):
        having_ip = 1 #phishing
    else:
        having_ip = -1 #legitimate
    return(having_ip)
    return 0
```

```
def url_length(url):
    length=len(url)
    if(length<54):
        return -1
    elif(54<=length<=75):
        return 0
    else:
        return 1
```

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

```
def having_at_symbol(url):
    symbol=regex.findall(r'@',url)
```

```
if(len(symbol)==0):
```

```
    return -1
```

```
else:
```

```
    return 1
```

```
def doubleSlash(url):
```

```
    #ongoing
```

```
    return 0
```

```
def prefix_suffix(url):
```

```
    subDomain, domain, suffix = extract(url)
```

```
    if(domain.count('-')):
```

```
        return 1
```

```
    else:
```

```
        return -1
```

```
def sub_domain(url):
```

```
    subDomain, domain, suffix = extract(url)
```

```
    if(subDomain.count('.')==0):
```

```
        return -1
```

```
    elif(subDomain.count('.')==1):
```

```
        return 0
```

```
    else:
```

```
        return 1
```

```
def SSLfinal_State(url):
```

```
    try:
```

```
#check wheather contains https
```

```
    if(regex.search('^https',url)):
```

```
        usehttps = 1
```

```
    else:
```

```
        usehttps = 0
```

```

#getting the certificate issuer to later compare with trusted issuer

#getting host name
subDomain, domain, suffix = extract(url)
host_name = domain + "." + suffix
context = ssl.create_default_context()
sct = context.wrap_socket(socket.socket(), server_hostname = host_name)
sct.connect((host_name, 443))
certificate = sct.getpeercert()
issuer = dict(x[0] for x in certificate['issuer'])
certificate_Auth = str(issuer['commonName'])
certificate_Auth = certificate_Auth.split()
if(certificate_Auth[0] == "Network" or certificate_Auth == "Deutsche"):
    certificate_Auth = certificate_Auth[0] + " " + certificate_Auth[1]
else:
    certificate_Auth = certificate_Auth[0]

trusted_Auth =
['Comodo','Symantec','GoDaddy','GlobalSign','DigiCert','StartCom','Entrust','Verizon','Trustw
ave','Unizeto','Buypass','QuoVadis','Deutsche Telekom','Network
Solutions','SwissSign','IdenTrust','Secom','TWCA','GeoTrust','Thawte','Doster','VeriSign']

#getting age of certificate
startingDate = str(certificate['notBefore'])
endingDate = str(certificate['notAfter'])
startingYear = int(startingDate.split()[3])
endingYear = int(endingDate.split()[3])
Age_of_certificate = endingYear-startingYear

#checking final conditions
if((usehttps==1) and (certificate_Auth in trusted_Auth) and (Age_of_certificate>=1) ):
    return -1 #legitimate
elif((usehttps==1) and (certificate_Auth not in trusted_Auth)):
    return 0 #suspicious
else:
    return 1 #phishing

```

```
except Exception as e:
```

```
    return 1
```

```
def domain_registration(url):
```

```
    try:
```

```
        w = whois.whois(url)
```

```
        updated = w.updated_date
```

```
        exp = w.expiration_date
```

```
        length = (exp[0]-updated[0]).days
```

```
        if(length<=365):
```

```
            return 1
```

```
        else:
```

```
            return -1
```

```
    except:
```

```
        return 0
```

```
def favicon(url):
```

```
    #ongoing
```

```
    return 0
```

```
def port(url):
```

```
    #ongoing
```

```
    return 0
```

```
def https_token(url):
```

```
    subDomain, domain, suffix = extract(url)
```

```
    host =subDomain + '.' + domain + '.' + suffix
```

```
    if(host.count('https')): #attacker can trick by putting https in domain part
```

```
        return 1
```

```
    else:
```

```
return -1
```

```
def request_url(url):
```

```
    try:
```

```
        subDomain, domain, suffix = extract(url)
```

```
        websiteDomain = domain
```

```
        opener = urllib.request.urlopen(url).read()
```

```
        soup = BeautifulSoup(opener, 'xml')
```

```
        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
    elif(0.22<=avg<=0.61):
        return 0
    else:
        return 1
except:
    return 0
```

```
def url_of_anchor(url):
```

```
    try:
        subDomain, domain, suffix = extract(url)
        websiteDomain = domain

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

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

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

```
def Links_in_tags(url):
```

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

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

        if(avg<0.25):
            return -1
        elif(0.25<=avg<=0.81):
```

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

def sfh(url):
    #ongoing
    return 0

def email_submit(url):
    try:
        opener = urllib.request.urlopen(url).read()
        soup = BeautifulSoup(opener, 'xml')
        if(soup.find('mailto:')):
            return 1
        else:
            return -1
    except:
        return 0

def abnormal_url(url):
    #ongoing
    return 0

def redirect(url):
    #ongoing
    return 0

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

```
def rightClick(url):
```

```
    #ongoing
```

```
    return 0
```

```
def popup(url):
```

```
    #ongoing
```

```
    return 0
```

```
def iframe(url):
```

```
    #ongoing
```

```
    return 0
```

```
def age_of_domain(url):
```

```
    try:
```

```
        w = whois.whois(url)
```

```
        start_date = w.creation_date
```

```
        current_date = datetime.datetime.now()
```

```
        age =(current_date-start_date[0]).days
```

```
        if(age>=180):
```

```
            return -1
```

```
        else:
```

```
            return 1
```

```
    except Exception as e:
```

```
        print(e)
```

```
        return 0
```

```
def dns(url):
```

```
    #ongoing
```

```
    return 0
```

```
def web_traffic(url):
```

```
#ongoing
```

```
return 0
```

```
def page_rank(url):
```

```
#ongoing
```

```
return 0
```

```
def google_index(url):
```

```
#ongoing
```

```
return 0
```

```
def links_pointing(url):
```

```
#ongoing
```

```
return 0
```

```
def statistical(url):
```

```
#ongoing
```

```
return 0
```

```
def main(url):
```

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

```
return check
```

8 TESTING:

8.1 Test Cases:

				Date	15-Nov-22								
				Team ID	PNT2022TMD40752								
				Project Name	Project - Web Phishing Detection								
				Maximum Marks	4 marks								
Component						Actual TC for BU							
Test case ID	Feature Type	S	Test Scenario	Pre-Requirement	Steps To Execute	Test Data https://phishingshield.herokuapp.com/	Expected Result	Result	Status	Comments	Automation(Y/N)	ID	Executed By
LoginPage_TC_OO 1	Functional	Home Page	Verify user is able to see the Landing Page when user can type the URL in the box		1.Enter URL and click go 2.Type the URL 3.Verify whether it is processing or not.		Should Display the Webpage	Working as expected	Pass		N		N Vijay
LoginPage_TC_OO 2	UI	Home Page	Verify the UI elements is Responsive		1.Enter URL and click go 2. Type or copy paste the URL 3. Check whether the button is responsive or not 4. Reload and Test Simultaneously	https://phishingshield.herokuapp.com/	Should Wait for Response and then gets Acknowledge	Working as expected	Pass		N		S Bavya bai
LoginPage_TC_OO 3	Functional	Home page	Verify whether the link is legitimate or not		1.Enter URL and click go 2. Type or copy paste the URL 3. Check the website is legitimate or not 4. Observe the results	https://phishingshield.herokuapp.com/	User should observe whether the website is legitimate or not.	Working as expected	Pass		N		R Arun
LoginPage_TC_OO 4	Functional	Home Page	Verify user is able to access the legitimate website or not		1.Enter URL and click go 2. Type or copy paste the URL 3. Check the website is legitimate or not 4. Continue if the website is legitimate or be cautious if it is not legitimate.	https://phishingshield.herokuapp.com/	Application should show that Safe Webpage or Unsafe.	Working as expected	Pass		N		D Ajith
LoginPage_TC_OO 5	Functional	Home Page	Testing the website with multiple URLs		1.Enter URL (https://phishingshield.herokuapp.com/) and click go 2. Type or copy paste the URL to test 3. Check the website is legitimate or not 4. Continue if the website is secure or be cautious if it is not secure	1. https://vishalaw.github.io/websocketstaged.com 2. https://www.500px.com 3. safescript.info 4. https://www.google.com/delights.com	User can able to identify the websites whether it is secure or not	Working as expected	Pass		N		N Vijay

8.2 User Acceptance Testing:

UAT Execution & Report Submission

1.Purpose of Document:

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

2 .Defect Analysis:

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

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

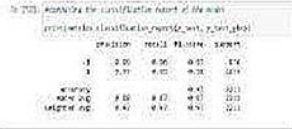

Test Case Analysis:

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

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

9 RESULTS

9.1 Performance Metrics:

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

1. METRICS:

CLASSIFICATION REPORT:

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

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

```

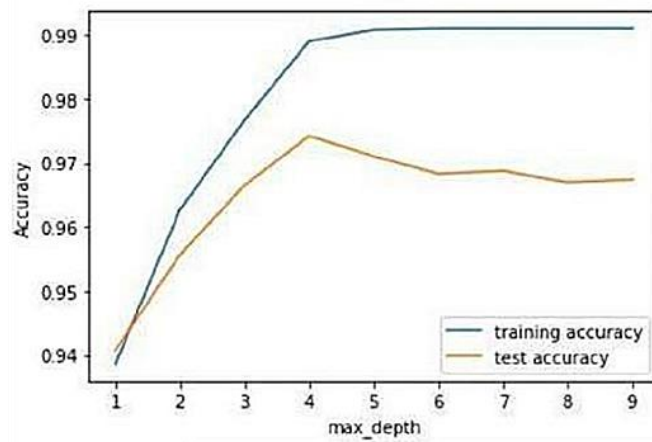
              precision    recall  f1-score   support

-1          0.99         0.96         0.97         976
1           0.97         0.99         0.98        1235

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

```


PERFORMANCE :



Out[83]:

	ML Model	Accuracy	f1_score	Recall	Precision
0	Gradient Boosting Classifier	0.974	0.977	0.994	0.986
1	CatBoost Classifier	0.972	0.975	0.994	0.989
2	Random Forest	0.969	0.972	0.992	0.991
3	Support Vector Machine	0.964	0.968	0.980	0.965
4	Decision Tree	0.958	0.962	0.991	0.993
5	K-Nearest Neighbors	0.956	0.961	0.991	0.989
6	Logistic Regression	0.934	0.941	0.943	0.927
7	Naive Bayes Classifier	0.605	0.454	0.292	0.997
8	XGBoost Classifier	0.548	0.548	0.993	0.984
9	Multi-layer Perceptron	0.543	0.543	0.989	0.983

TUNE THE MODEL - HYPERPARAMETER TUNING:

2. TUNE THE MODEL – HYPERPARAMETER TUNING

```
In [58]: #HYPERPARAMETER TUNING  
grid.fit(X_train, y_train)
```

```
Out[58]: 

GridSearchCV  
GridSearchCV(cv=5,  
             estimator=GradientBoostingClassifier(learning_rate=0.7,  
                                                    max_depth=4),  
             param_grid={'max_features': array([1, 2, 3, 4, 5]),  
                         'n_estimators': array([ 10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 110, 120, 130,  
140, 150, 160, 170, 180, 190, 200])})  
             estimator: GradientBoostingClassifier  
             GradientBoostingClassifier(learning_rate=0.7, max_depth=4)  
             GradientBoostingClassifier  
             GradientBoostingClassifier(learning_rate=0.7, max_depth=4)


```

```
In [59]: print("The best parameters are %s with a score of %0.2f"  
              % (grid.best_params_, grid.best_score_))  
  
The best parameters are {'max_features': 5, 'n_estimators': 200} with a score of 0.97
```

VALIDATION METHODS: KFOLD & Cross Folding

Wilcoxon signed-rank test

```
In [78]: #KFOLD and Cross Validation Model

from scipy.stats import wilcoxon
from sklearn.datasets import load_iris
from sklearn.ensemble import GradientBoostingClassifier
from xgboost import XGBClassifier
from sklearn.model_selection import cross_val_score, KFold

# Load the dataset
X = load_iris().data
y = load_iris().target

# Prepare models and select your CV method
model1 = GradientBoostingClassifier(n_estimators=100)
model2 = XGBClassifier(n_estimators=100)
kf = KFold(n_splits=20, random_state=None)
# Extract results for each model on the same folds
results_model1 = cross_val_score(model1, X, y, cv=kf)
results_model2 = cross_val_score(model2, X, y, cv=kf)
stat, p = wilcoxon(results_model1, results_model2, zero_method='zsplit');
stat
```

Out[78]: 95.0

5x2CV combined F test

```
In [89]: from mlxtend.evaluate import combined_ftest_5x2cv
from sklearn.tree import DecisionTreeClassifier, ExtraTreeClassifier
from sklearn.ensemble import GradientBoostingClassifier
from mlxtend.data import iris_data

# Prepare data and clfs
X, y = iris_data()
clf1 = GradientBoostingClassifier()
clf2 = DecisionTreeClassifier()

# Calculate p-value
f, p = combined_ftest_5x2cv(estimator1=clf1,
                           estimator2=clf2,
                           X=X, y=y,
                           random_seed=1)

print('f-value:', f)
print('p-value:', p)

f-value: 1.727272727272733
p-value: 0.2840135734291782
```

10 ADVANTAGES

1. Improve on Inefficiencies of SEG and Phishing Awareness Training
2. It Takes a Load off the Security Team
3. It Offers a Solution, Not a Tool
4. Separate You from Your Competitors
5. This system can be used by many e-commerce websites in order to have good customer relationships.
6. If internet connection fails this system will work

DISADVANTAGES

1. All website related data will be stored in one place.
2. It is a very time-consuming process.

11 CONCLUSION:

It is outstanding that a decent enemy of phishing apparatus ought to anticipate the phishing assaults in a decent timescale. We accept that the accessibility of a decent enemy of phishing device at a decent time scale is additionally imperative to build the extent of anticipating phishing sites. This apparatus ought to be improved continually through consistent retraining. As a matter of fact, the accessibility of crisp and cutting-edge preparing dataset which may gained utilizing our very own device [30, 32] will help us to retrain our model consistently and handle any adjustments in the highlights, which are influential in deciding the site class. Albeit neural system demonstrates its capacity to tackle a wide assortment of classification issues, the procedure of finding the ideal structure is very difficult, and much of the time, this structure is controlled by experimentation. Our model takes care of this issue via computerizing the way toward organizing a neural system conspire; hence, on the off chance that we construct an enemy of phishing model and for any reasons we have to refresh it, at that point our model will encourage this procedure, that is, since our model will mechanize the organizing procedure and will request scarcely any client defined parameters.

12 FUTURE SCOPE:

There is a scope for future development of this project. We will implement this using advanced deep learning method to improve the accuracy and precision. Enhancements can be done in an efficient manner. Thus, the project is flexible and can be enhanced at any time with more advanced features.

13 APPENDIX:

- Application Building
- Collection of Dataset
- Data Pre-processing
- Integration of Flask App with IBM Cloud
- Model Building
- Performance Testing
- Training the model on IBM
- User Acceptance Testing
- Ideation Phase
- Preparation Phase

- Project Planning
- Performance Testing
- User Acceptance Testing

PROJECT LINK:

<https://github.com/IBM-EPBL/IBM-Project-21978-1659800383>