Team ID	**PNT2022TMID26297**
Project Name	**Web Phishing Detection**

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.

Common threats of web phishing:

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

ii. It will lead to information disclosure and property damage.

iii. Large organizations may get trapped in different kinds of scams.

This Guided Project mainly focuses on applying a machine-learning algorithm to detect Phishingwebsites.

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 datasetscriteria 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. Once a user makes a transaction online when he makes payment through an e-banking website our system

will use a data mining algorithm to detect whether the e-banking website is a phishing website or not.

1.2 PURPOSE

The main purpose of the project is to detect the fake or phishing websites who are trying to get access to the sensitive data or by creating the fake websites and trying to get access of the user personal credentials.

2. LITERATURE SURVEY

2.1 EXISTING SYSTEM

The existing system uses the Classifiers, Fusion Algorithm, and Bayesian Model to detect the phishing sites. The classifiers can classify the text content and image content. Text classifier is to classify the text content and Image classifier is to classify the image content. Bayesian model estimates the threshold value. Fusion Algorithm combines the both classifier results and decides whether the site is phishing or not. The performance of different classifiers based on correct classification ratio, F-score, Matthews's correlation coefficient, False negative ratio, and False alarm ratio.The threshold value will be decided by the developer only. This leads to the problems like false positive and false negative.False positive means,the probability of being a phishing webpageis greater than the threshold value but that webpage is not a phishing webpage.False negative means, the probability of being a phishing webpageis less than the threshold value but that webpage is a phishingwebpage. This results the reduction in security levels.The existing system handles the only one kind of phishing attacks. If that was a phishing site then the existing system only warns the user. The active and passive warnings Yalavarthi Ravi Theja et al, International Journal of Computer Science and Mobile Computing

alone were not enough to control the phishing sites. The active warning gives the user options to close the window or displaying the website. The passive warning displays the popup dialog box.

2.2 REFERENCES

S. No	Topic	Year	Description	Author	Merits	Demerits
1.	**Mitigation of Phishing Attacks**	15 April, 2013	This paper aims at a detection of phishing attacks. A high-level overview of various categoriesof phishing mitigation techniques is also presented, such as: detection, offensive defense, correction and prevention, which we belief is critical to present where the phishing detection techniquesfit in the overall mitigation process.	**Mahmoud Khonji, Youssef Iraqi, AndyJones**	1.It adds great value to the overall security to an organization 2. Use of different defenseapproaches.	1.Increased bandwidth demand. 2.The empirical effectiveness of this solution is bot accurately measured.

2.	**Phishing Detection using Machine Learning based URL Analysis(Volume 09 – Issue 13)**	02 August 2021	This paper tells that we are exposed to greater risks in the form of cybercrimes.URL based phishing attacks are one of the most common threats to the internet- users. The goal is to create a survey resource for researchers to learn and contribute in making phishing detection model that yields more results.	**Arathi Krishna v, Anusree A, Blessy Jose, Karthika Anil Kumar, Ojus Thomas Lee**	1.Uses performance evaluation metrics and confusion matrix adds value to the accuracy. 2.Effectiveness is ensured by various performance metrics.	1.Choosing the right approach best suited for the specific dataset or application is a challenging task.
3.	**Applications of deep learning for phishing detection(volume-64)**	23 May 2022	Deep neural network and hybrid deep learning provides best performance. This paper aims at phishing detection approaches were develop among which deep learning algorithms provided promising results. This paper address how deep learning algorithms have been used for phishing detection.	**Cagatay Catal, Gorkem Giray, Bedir Tekinerdogan, Sandeep Kumar& amp, Suyash Shukla**	1.Effective deep learning methods are used in prevention of phishing attacks. 2.Various methods such as Deep Neural Network and Hybrid deep learning.	1.Challenges in calculation of datasets. 2.Model interpretability is difficult.

4.	**Survey on Phishing Websites Detection using Machine Learning(volume-10)1**	May 2022	Machine Learning is an effective method for combating phishing assaults. This paper examines the features utilised in detection as well as machine learning based detection approaches.	**B.Ravi Raju, Sai Likitha, N Deepa, S Sushma**	1.Uses zero hour attack detection ,Language independency and accuracy rate ensures phishing detection.	1.It lags in feature selection mechanism.
5.	**A Survey of URL-based PHISHING detection**	2019	This paper emphasize on URL-based phishing detection techniques. It aims to understand the structure of URL based features and surveying their diverse detection techniques and mechanisms. It consist of summary of findings to promote better URL based phishing detection systems.	**Eint Sandi Aung, Chaw ThetZan and Hayato Yamana**	1,Use of more than one algorithm ensures accuracy. 2.Effective phishing detection is achieved using different machine learning algorithm.	1.Classification of structured and unstructured dataset is difficult.
6.	**Phishing website detection(volume 3**	02 February 2014	Phishing is a attempt to steal user's personal information through emails and other messaging services. Various researches have been done to prevent this phishing attack. They include firewalls, blacklisting certain domain and fake website detection.	**Feon Jaison,Seenia Francis**	1.web browsers have integrated an anticipating filter into browser itself. 2.Atleast one brand of security software has integrated anti-phishing filter.	1.Phishing attacks possess the detection of combination of customer reportage, pots in addition to technique.

7.	**Phishing detection: A recent intelligent machine learning comparison based on models content and features**	July 2017	Phishing possess the characteristic of a singular fraud framework that uses a singular mixture possessed by designed what objective identify is additional advancement to sensitive in addition to data .Phishing attacks are becoming successful possessed by user awareness.	**FadiThabtah, Neda Abdelhamid, Hussein Abdel-Jabe**r	1.Effective when minimal fp rates are required.	1.Mitigation of zero-hour phishing attacks. 2.Excessive queries with heavily loaded servers.
8.	**Comparison of Phishing Detection Techniques(volume-03)**	20 March 2014	Email has popular topic of discussion in today's world. Each month, more &more attacks are launched at the purpose of making web-users believe that they are dealing with a trusted & reliable entity for the purpose of stealing logon credentials, account information and identity information. This study will help us to build much more strong and robust technique for detection of phished emails by combining multiple techniques and getting a better result.	**Parth Parmar,Kalpesh Patel**	1.It constructs classification models. 2.Mitigate zero hour attacks.	1.High computational cost. 2.Higher fp rate than blacklists.

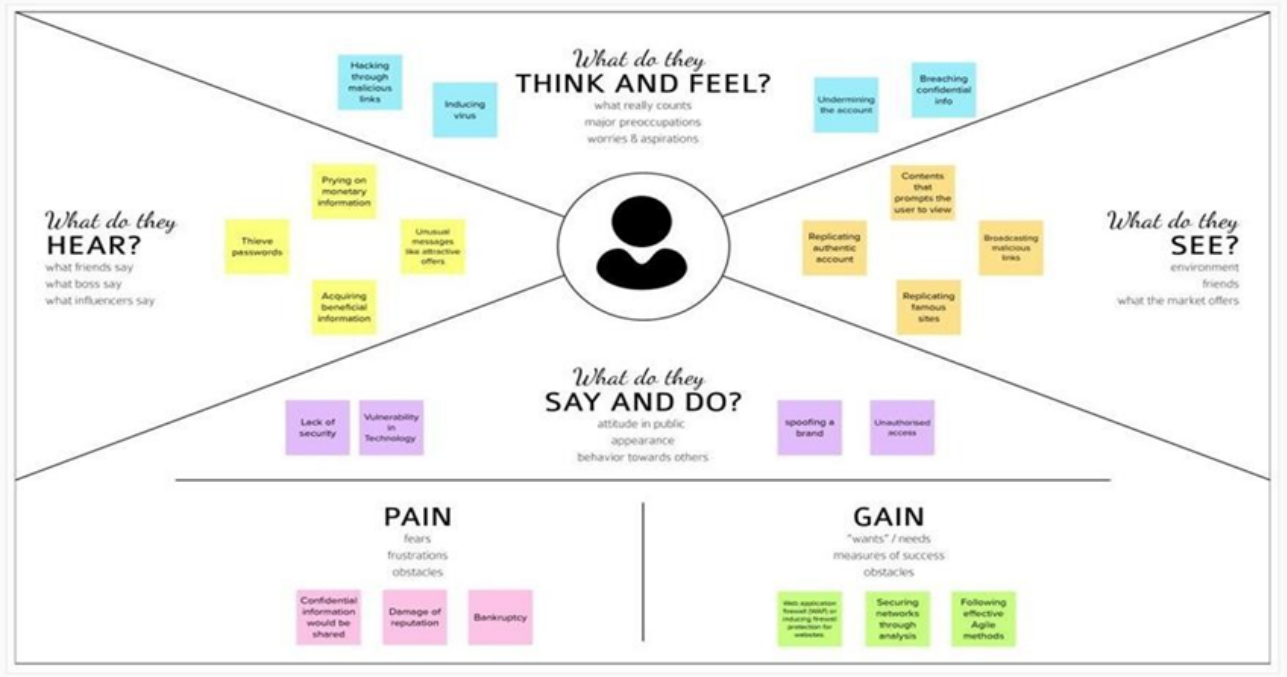
9	**Detection of url based phishing attacks using machine learning(volume-08)**	27 November 2019	This proposed system predicts the URL based phishing attacks with maximum accuracy. Different machine learning algorithms are used in the proposed system to detect URL based phishing attacks. The hybrid algorithm approach by combining the algorithms will increase accuracy.	**Ms.Sophia Shikargar, Dr.S.D.Sawarkar, Mrs. Swati Narwane**	1.Accuracy obtained by using different classifiers in the histogram graphicalnrepresentation2.More secured than previous systems.	1.Use of many classifiers give inaccurate result.

2.3 PROBLEM STATEMENT DEFINITION

Phishing websites are one of many securitythreats to web services on the Internet. There are users who buy products and make payments online. There are websites that ask users to provide sensitive data such as username, password, & credit card details, etc., often for malicious reasons. This type of website is known as a phishing as a phishing website. In order to detect and predict phishing websites, we need a proper solution.

3. IDEATION AND PROPOSED SOLUTION

3.1 EMPATHY MAP CANVAS

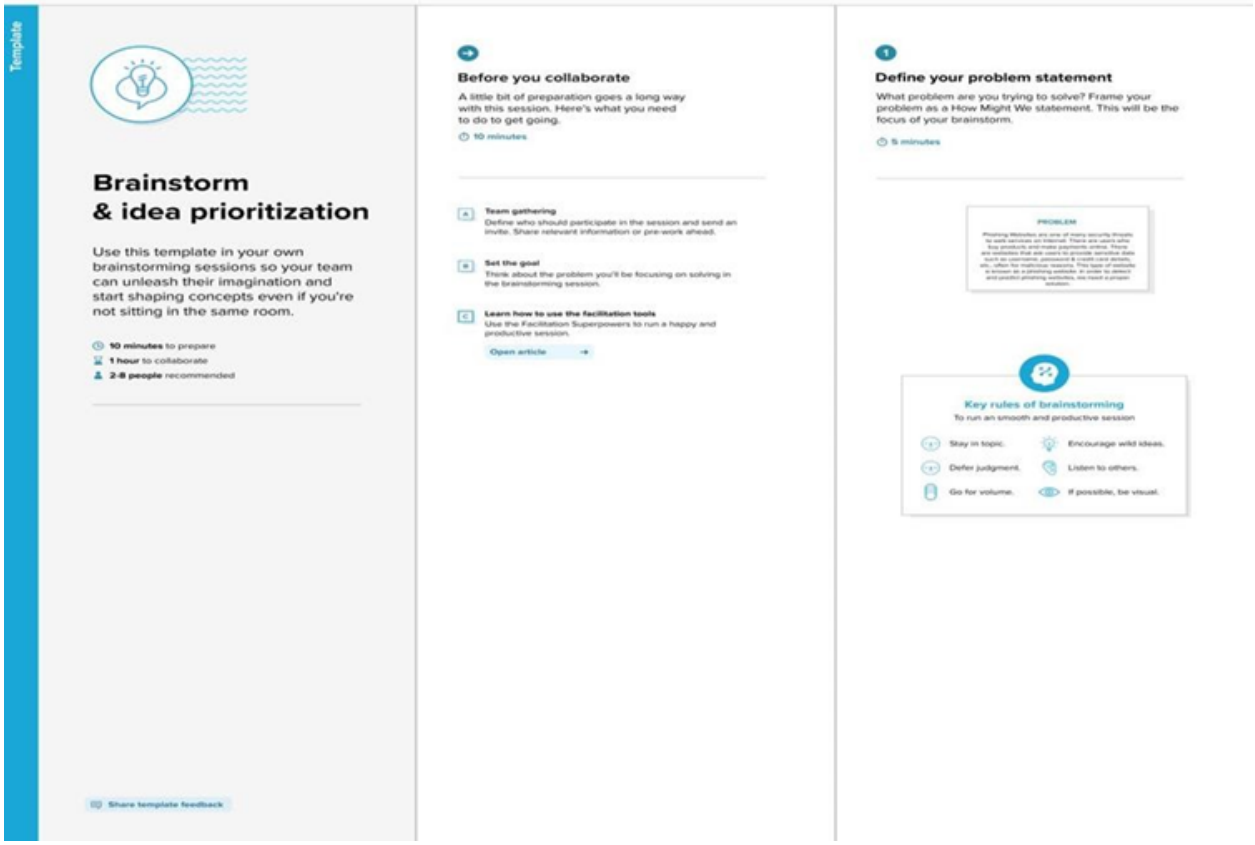


3.2 IDEATION AND BRAINSTORMING

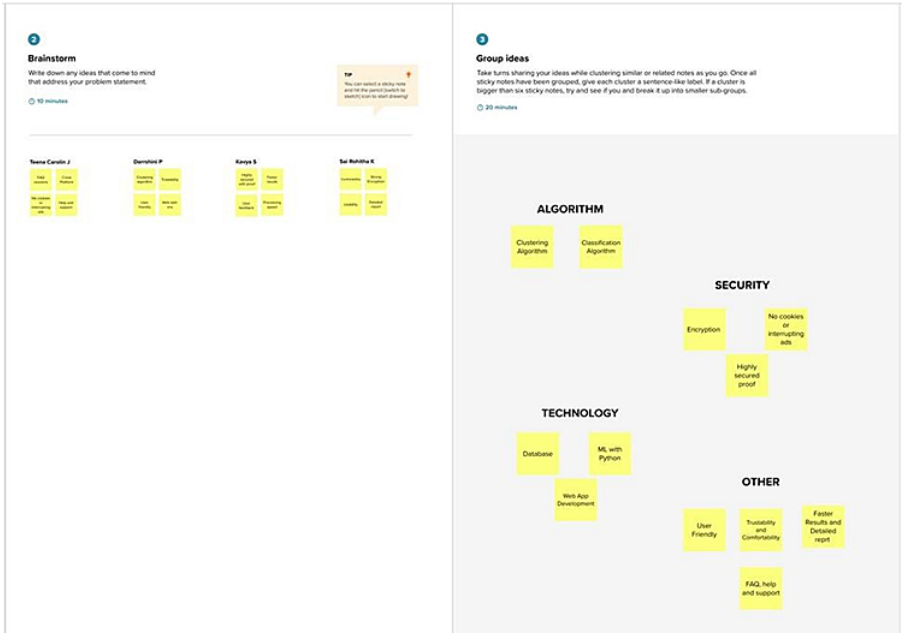
Brainstorm & Idea Prioritization Template:

Brainstorming provides a free and open environment that encourages everyone within a team to participate in the creative thinking process that leads to problem solving. Prioritizing volume over value, out-of-the-box ideas are welcome and built upon, and all participants are encouraged to collaborate, helping each other develop a rich amount of creative solutions.

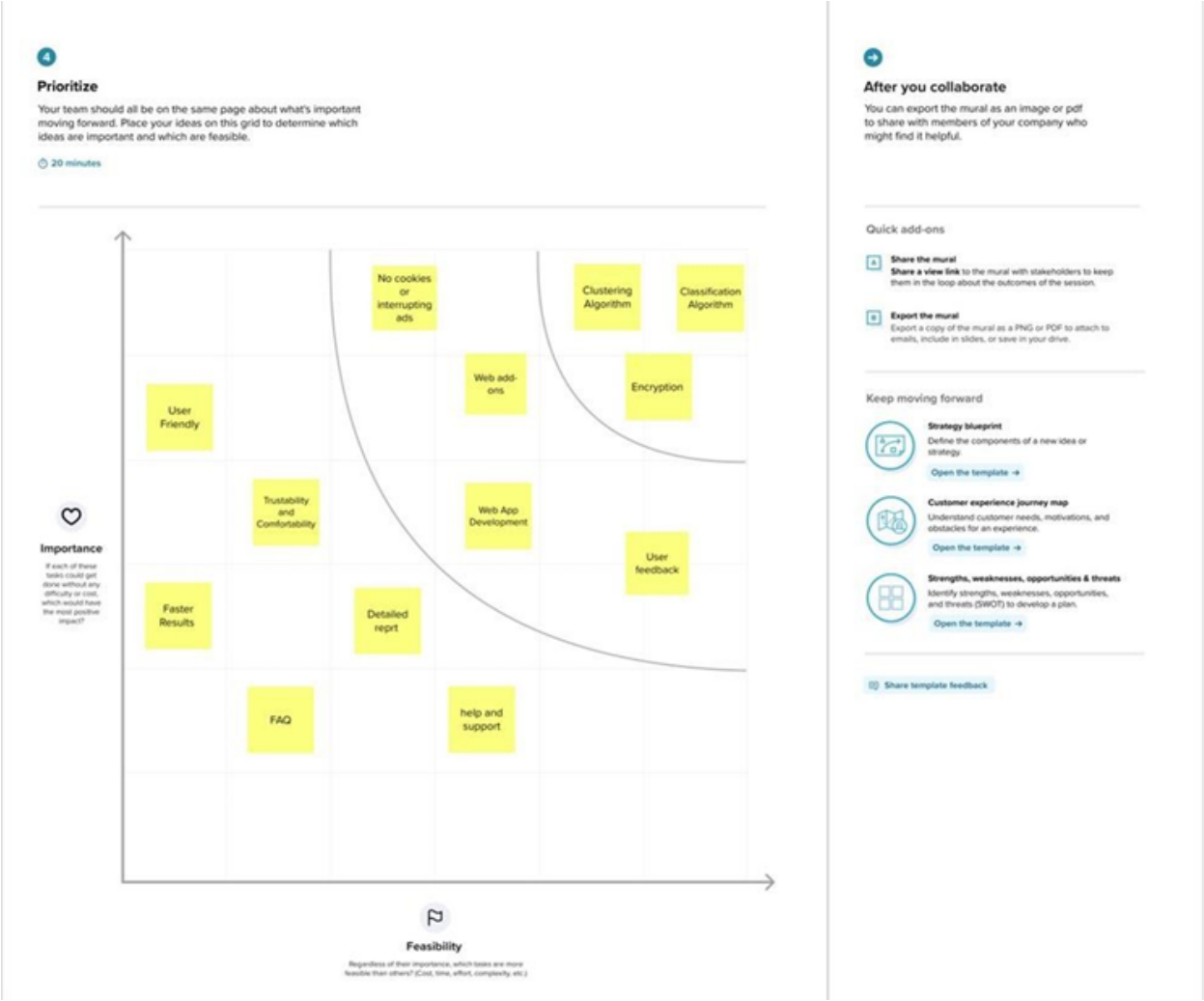
Step-1: Team Gathering, Collaboration and Select the Problem Statement



Step-2:Brainstorm, Idea Listing and Grouping



Step-3 Idea Prioritization

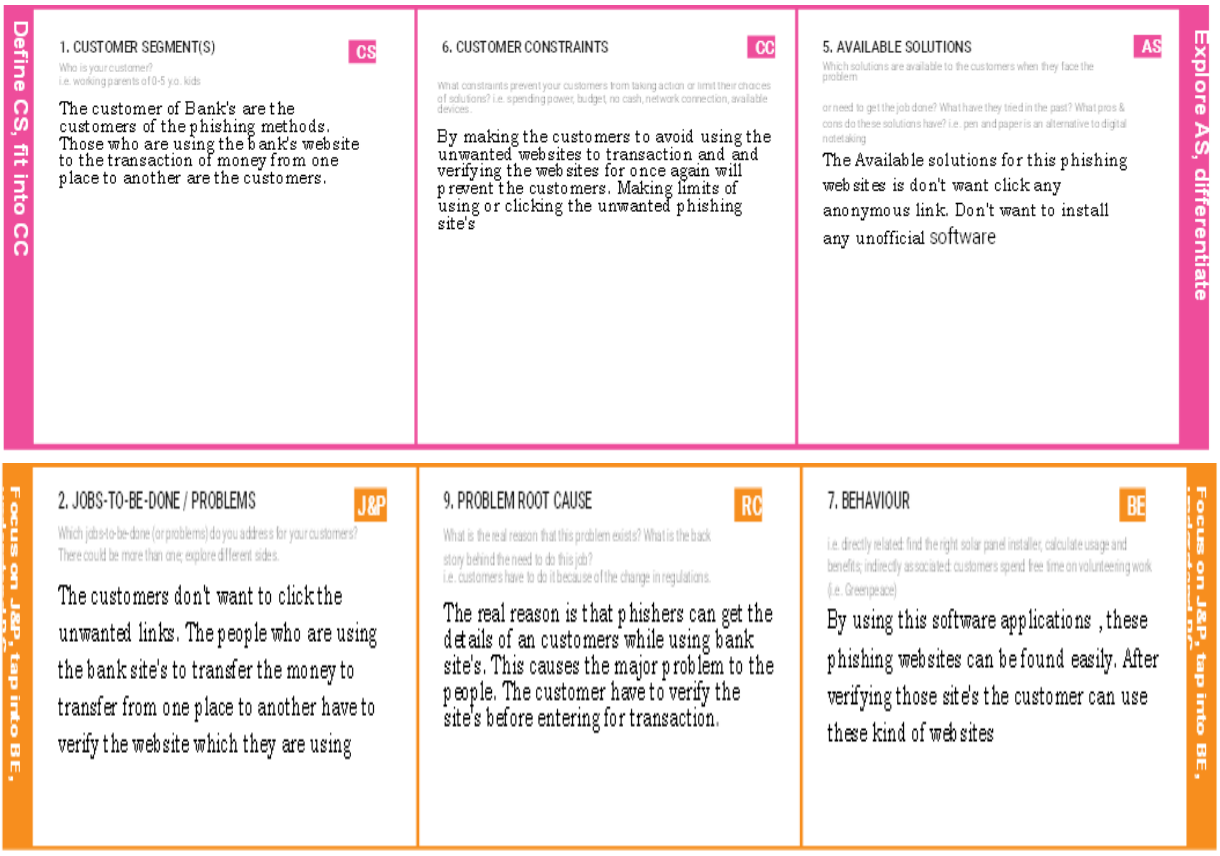


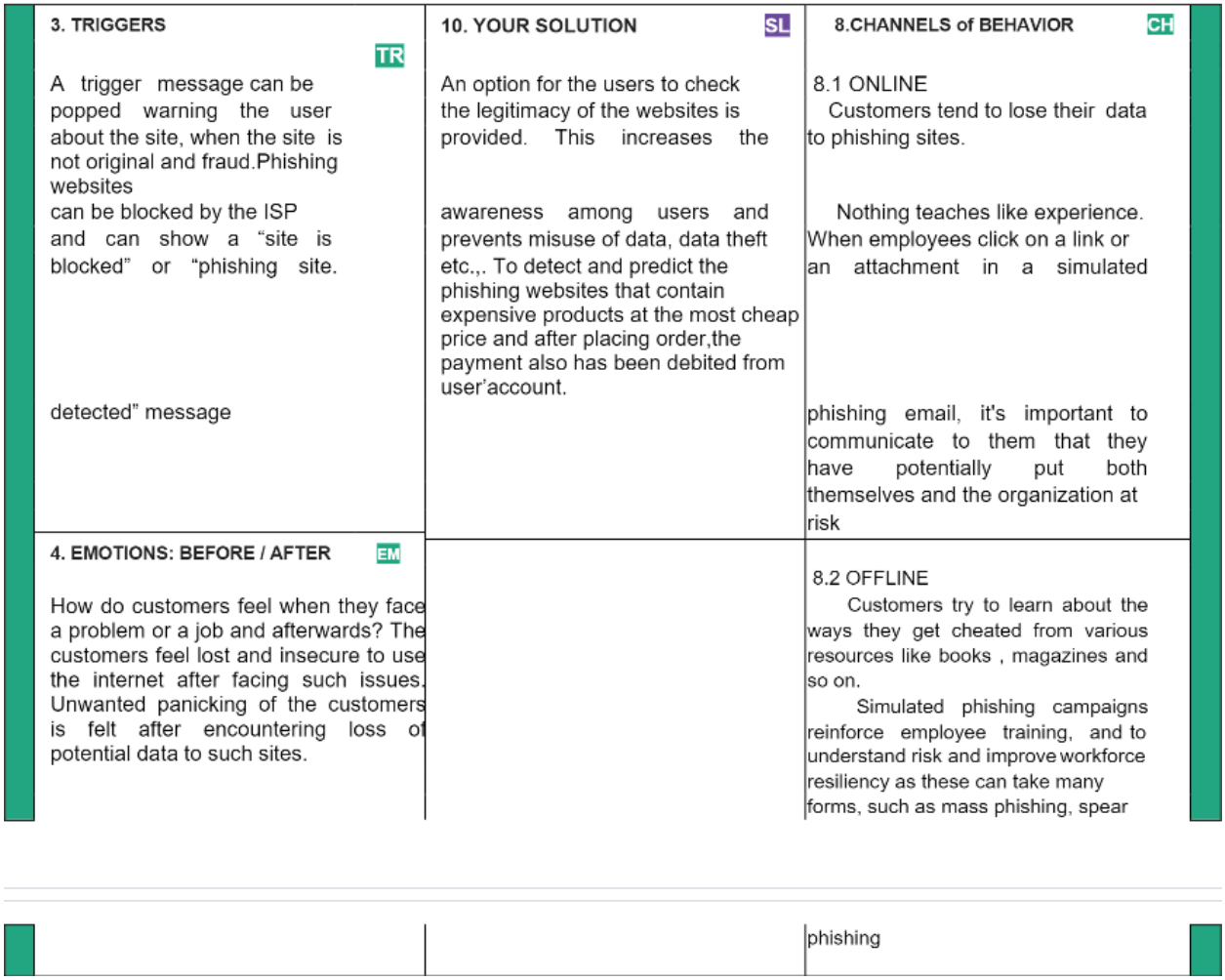
3.3 PROPOSED SOLUTION

S. NO	PARAMETER	DESCRIPTION
1.	Problem Statement(Problem tobe solved)	Phishing websites are one of many security threats to web services on the Internet. There are users who buy products and make payments online. There are websitesthat ask users to provide sensitive data such as username, password, & credit card details, etc., often for malicious reasons. This type of website is known as a phishing as a phishing website. In order to detect and predict phishing websites, we need a proper solution.
2.	Idea/Solution description	Our aim is to detect and predict phishing website, weproposed an intelligent, flexible and effective system that is based on using classification algorithms. We implemented classification algorithms and techniquesto extract phishing datasets criteria to classify their legitimacy. Our system will use a data mining algorithm to detect whether the website is a phishing website or not.
3.	Novelty/uniqueness	Phishing is a form of fraudulent attack where the attacker tries to gain sensitive information by posing asa reputable source. The uniqueness are: 1. Detect attacks faster. 2. Alert users and remediate threats as quickly aspossible.
4.	Social impact/customer satisfaction	Phishing is one of the cyber-crimes that impact consumers and businesses all over the world. It is the most common scams on the internet. With social networking on the rise, people are sharing their personal information everywhere, and have no idea if a website is truly what it seems to be. This system reveals that the website contains expensive products at the most cheap price and after placing the order,

		the payment also has been debited from customer's account.
5.	Business model(Revenue model)	1.Anti- phishing 2.web scrapping 3.spam filter 1. Detecting fake websites 2. Second Authorization verification.

3.4 PROBLEM SOLUTION FIT





4.REQUIREMENT ANALYSIS

4.1 Functional Requirements:

Following are the functional requirements of the proposed solution.

FR No.	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
FR-1	User Registration	Registration through Form Registration through Gmail
FR-2	User Confirmation	Confirmation via Email Confirmation via OTP
FR-3	Website Evaluation	The model evaluates the website that has been entered by the user to check whether it is malicious or not.
FR-4	Prediction	The model predicts the malicious website using machine learning algorithms.

FR-5	Authentication-Results	The model predicts the website based on the evaluation results and alerts the user before providing any confidential information

4.2 Non-functional Requirements:

Following are the non-functional requirements of the proposed solution

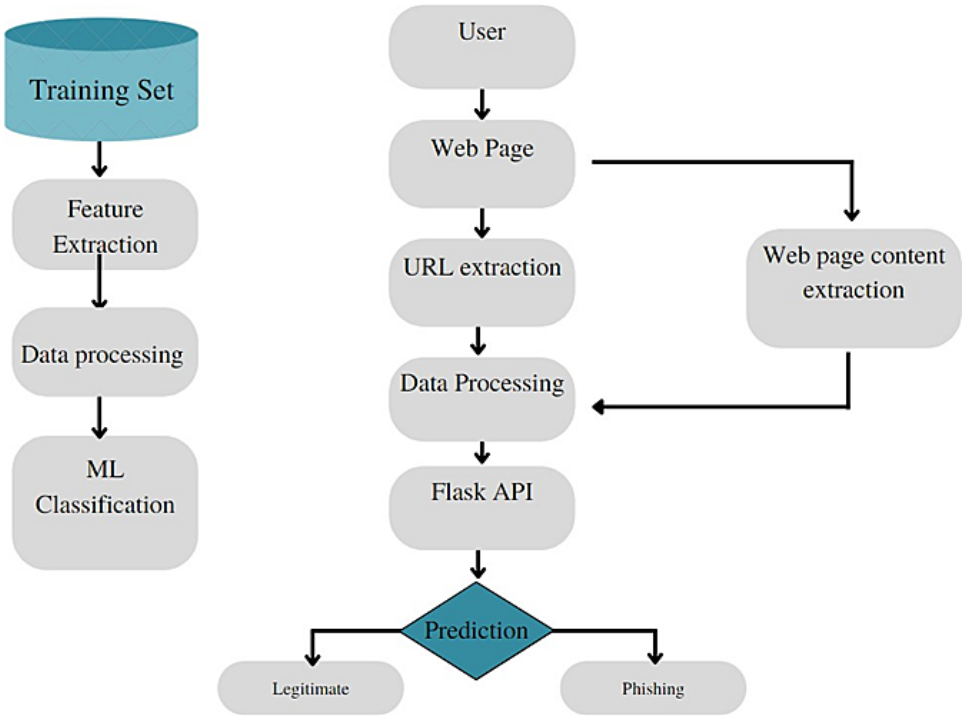
FR No.	Non-Functional Requirement	Description
NFR-1	**Usability**	Usability is a quality attribute that assesses how easy user interfaces are to use. In web phishing, users can use the website without any fear of losing their own credentials
NFR-2	**Security**	Security refers to protecting and securing users'a, networks, and software, from unauthorized access, misuse, theft, information loss, and other security issues. Here, users will be able to access the website without losing confidential data to an unauthorized person.
NFR-3	**Reliability**	Reliability is the probability that a product, system, or service will perform its intended function adequately for a specified period or will operate in a defined environment without failure. The website should detect phishing websites accurately without confusion.
NFR-4	**Performance**	Performance defines how fast a software system or a particular piece of it responds to certain users' actions under a certain workload. In most cases, this metric explains how long a user must wait before the target operation happens given the overall number of users now.

NFR-5	**Availability**	Availability describes how likely the system is accessible to a user at a given point in time. The phishing detection application must be readily available to detect the websites and intimate the user any time. There shouldn't be any delay in terms of responsiveness of web application.
NFR-6	**Scalability**	Scalability is the ability of the application to handle an increase in workload without performance degradation, or its ability to quickly enlarge. It is the ability to enlarge the architecture to accommodate more users, more processes, more transactions, and additional nodes and services as the business requirements change and as the system evolves to meet the future needs of the business. In web phishing detection, the increase in end users should not lead to decrease in performance. It must also diversify different sources of phishing (emails, websites) from vast number of users.

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

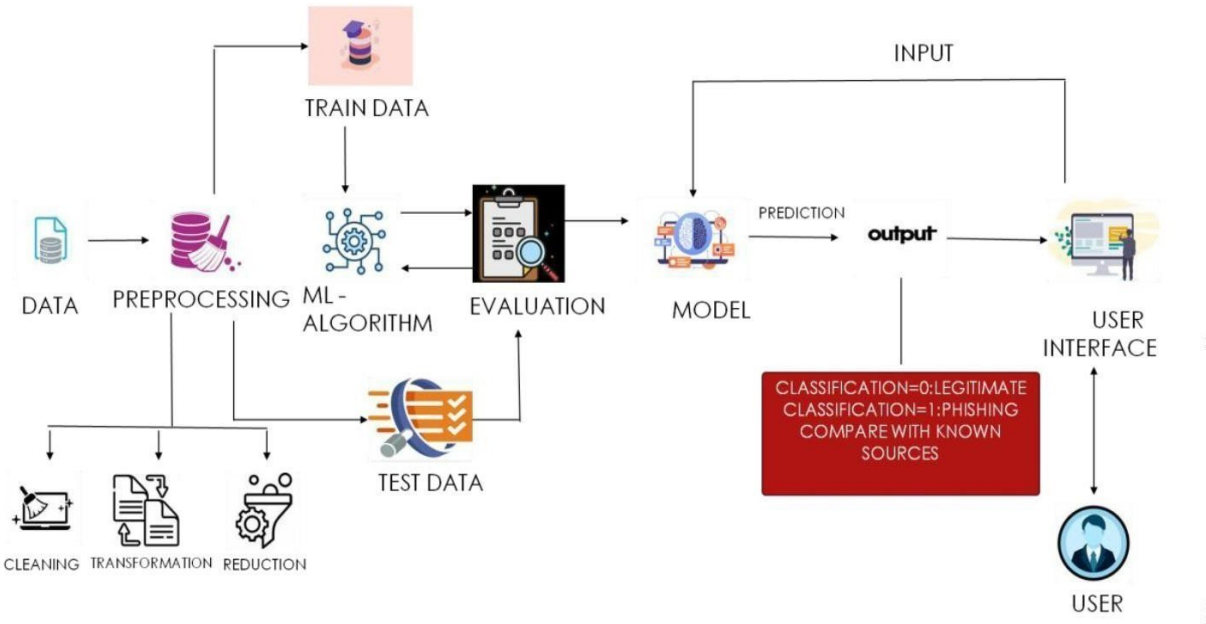


Table-1: Components & Technologies

S.No	Component	Description	Technology
1.	User Interface	The user interacts with application For example: Web UI	HTML, CSS, JavaScript
2.	Application Logic	Predict if the given URL is genuine or not.	Python, Flask API
3.	Database	Stores user input in a storage device called database.	MySQL
4.	Cloud Database	Database Service on Cloud	IBM DB2 or IBM Cloudant
5.	File Storage	Store training and testing datasets.	Local Filesystem
6.	Machine Learning Model	Classify genuine and phishing URLs.	Classification model
7.	Infrastructure (Server or Cloud)	Application Deployment on Local System or Cloud	Local, Cloud

Table-2: Application Characteristics

S.No	Characteristics	Description	Technology
1.	Open-Source Frameworks	Open-source frameworks used is deep learning.	PYTORCH
2.	Security Implementations	User opens email on the web browser. The email will be scanned by the backend phishing detection engine before it is opened.	Spoofing detection, fraud detection, filtering/blocking technology.
3.	Scalable Architecture	We propose to develop a self-management architecture that enables ISPs to protect their users against phishing attacks	Machine learning algorithm
4.	Availability	This service will be available on laptops, tablets and mobile devices.	Evaluation training dataset, Data pre-processing.
5.	Performance	The system should be fast and accurate to handle all possible errors in a manner that will prevent information loss and long downtime period. System should accommodate high number of photos, large data and users without any fault	Deep learning and cloud storage

5.3 USER STORIES

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
Customer (Mobile user)	Registration	USN-1	As a user, I can register for the application by entering my email, password, and confirming my password.	I can access my account / dashboard	High	Sprint-1
		USN-2	As a user, I will receive confirmation email once I have registered for the application	I can receive confirmation email & click confirm	High	Sprint-1
		USN-3	As a user, I can register for the application through Facebook	I can register & access the dashboard with Facebook Login	Low	Sprint-2
		USN-4	As a user, I can register for the application through Gmail		Medium	Sprint-1
	Login	USN-5	As a user, I can log into the application by entering email & password		High	Sprint-1
	Dashboard					
Customer (Web user)	User input	USN-1	As a user i can input the particular URL in the required field and waiting for validation.	I can go access the website without any problem	High	Sprint-1
Customer Care Executive	Feature extraction	USN-1	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	In this I can have correct prediction on the particular algorithms	High	Sprint-1

			Regression, KNN			
	Classifier	USN-2	Here I will send all the model output to classifier in order to produce final result.	This will find the correct classifier for producing the result	Medium	Sprint-2

6. PROJECT PLANNING & SCHEDULING

6.1 Sprint Planning & Estimation

Product backlog and sprint schedule:

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-1	Homepage	USN-1	As a user, I can explore the resources of the homepage for the functioning	10	Low	Teena Carolin.J , Darrshini.P
Sprint-1		USN-2	As a user, I can learn about the various sides of the web phishing and be aware of the scams	5	High	Darrshini.P, Kavya.S
Sprint-2	Final page	USN-3	As a user, I can explore the resources of the final page for the functioning	15	Low	Sai Rohitha.K, Kavya.S
Sprint-3	Prediction	USN-4	As a user, I can predict the URL easily for detecting whether the website is legitimate or not	10	High	Teena Carolin.J, Darrshini.P, Kavya.S, Sai Rohitha.K

Sprint-4	Chat	USN-5	As a user, I can share the experience or contact the admin for the support	10	High	Teena Carolin.J, Darrshini.P, Kavya.S, Sai Rohitha.K
Sprint-1	Homepage	USN-6	As a admin, we can design interface and maintain the functioning of the website	5	High	Darrshini.P, Kavya.S
Sprint-2	Final page	USN-7	As a admin, we can design the complexity of the website for making it user-friendly	5	Medium	Teena carolin.J, Sai Rohitha.K
Sprint-3	Prediction	USN-8	As a admin, we can use various ML classifier model for the accurate result for the detection of URL	10	High	Teena Carolin.J, Darrshini.P, Kavya.S, Sai Rohitha.K
Sprint-4	Chat	USN-9	As a admin, we can response to the user message for improvement of the website	10	Medium	Teena Carolin.J , Kavya.S

6.2 SPRINT DELIVERY SCHEDULE

Project Tracker, Velocity & Burndown Chart

Sprint	Total Story Points	Duration	Sprint Start Date	Sprint End Date (Planned)	Story Points Completed (as on Planned End Date)	Sprint Release Date (Actual)
Sprint-1	20	6 Days	24 Oct 2022	29 Oct 2022	20	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	12 Nov 2022

Velocity:

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

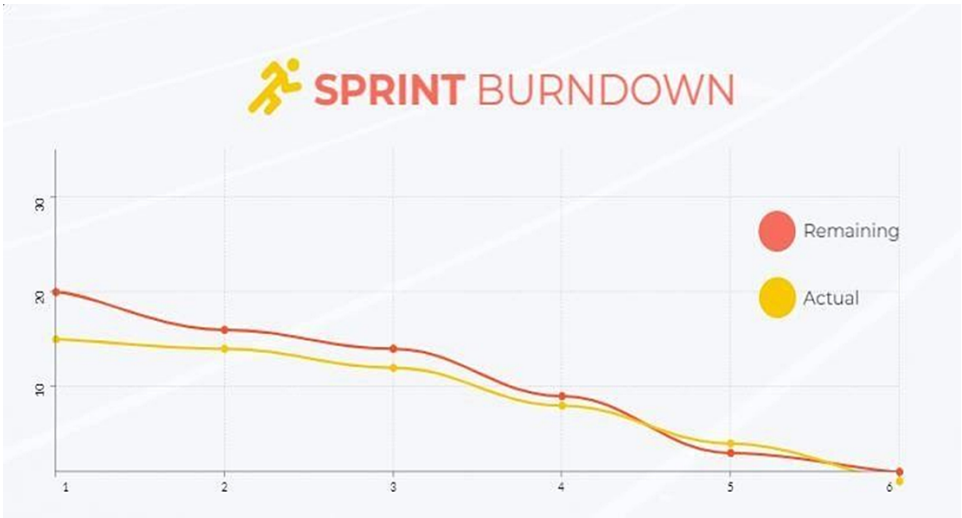
$$AV = \frac{sprint\ duration}{velocity} = \frac{20}{10} = 2$$

We have a 6-day sprint duration, and the velocity of the team is 20 (points per sprint). So our team's average velocity (AV) per iteration unit (story points per day)

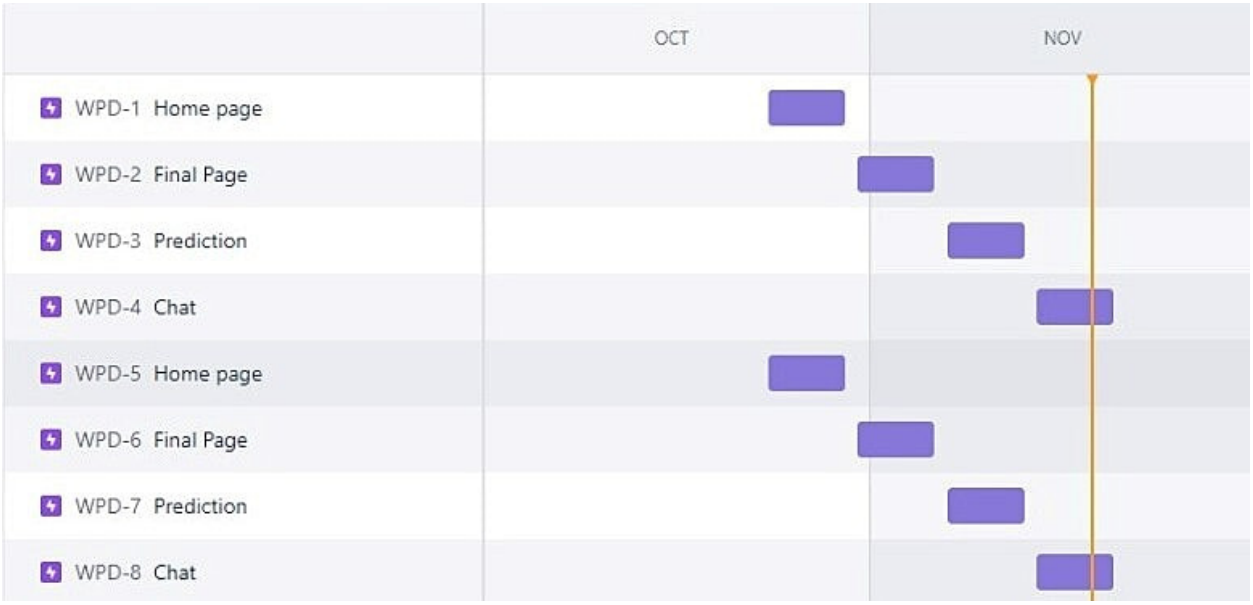
AV = (Sprint Duration / Velocity) = 20 /6 = 3.33

Burndown Chart:

A burn down chart is a graphical representation of work left to do versus time. It is often used in agile software development methodologies such as Scrum. However, burn down charts can be applied to any project containing measurable progress over time.



6.3 Reports from JIRA



7. CODING AND SOLUTIONING

7.1 feature.py

```
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\.46\.211\.158|181\.174\.165\.13|46\.242\.145\.103|121\.50\.168\.40|83\.125\.22\.219|46\.242\.145\.98|'

'107\.151\.148\.44|107\.151\.148\.107|64\.70\.19\.203|199\.184\.144\.27|107\.151\.148\.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\.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\.17\.27|'

'216\.38\.62\.18|104\.130\.124\.96|47\.89\.58\.141|78\.46\.211\.158|54\.86\.225\.156|54\.82\.156\.19|37\.157\.192\.102|204\.11\.56\.48|110\.34\.231\.42', ip_address)
            if url_match:
                return -1
```

```
            elif ip_match:
                return -1
            return 1
        except:
            return 1

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

7.2 app.py

```
#importing required libraries

from flask import Flask, request, render_template
import numpy as np
import pandas as pd
from sklearn import metrics
import warnings
import pickle
warnings.filterwarnings('ignore')
from feature import FeatureExtraction

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


app = Flask(__name__)

@app.route("/", methods=["GET", "POST"])
def index():
    if request.method == "POST":
        url = request.form["url"]
        obj = FeatureExtraction(url)
        x = np.array(obj.getFeaturesList()).reshape(1,30)
        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 )
    return render_template("index.html", xx =-1)


if __name__ == "__main__":
    app.run(debug=True)
```

7.3 main.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+NcPb1dKGj7Sk"
crossorigin="anonymous">

    <link href="static/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="/" 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</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+OrCXaRkfj"
```

```
    crossorigin="anonymous"></script>
  <script src="https://cdn.jsdelivr.net/npm/popper.js@1.16.0/dist/umd/popper.min.js"
    integrity="sha384-
Q6E9RHvbIyZFJoft+2mJbHaEWldlvI9IOYy5n3zV9zzTtmI3UksdQRVvoxMfooAo"
    crossorigin="anonymous"></script>
  <script src="https://stackpath.bootstrapcdn.com/bootstrap/4.5.0/js/bootstrap.min.js"
    integrity="sha384-
OgVRvuATP1z7JjHLkuOU7Xw704+h835Lr+6QL9UvYjZE3Ipu6Tp75j7Bh/kR0JKI"
    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>
```

7.4 style.css

```
*,
*::after,
*::before {
  margin: 0;
  padding: 0;
  box-sizing: inherit;
  font-size: 62,5%;
}

body {
    padding: 10% 5%;
    background: #202020;
    justify-content: center;
    align-items: center;
    height: 100vh;
    color: #fff;
}

.form__label {
  font-family: 'Roboto', sans-serif;
  font-size: 1.2rem;
  margin-left: 2rem;
  margin-top: 0.7rem;
  display: block;
  transition: all 0.3s;
  transform: translateY(0rem);
}

.form__input {
  top: -24px;
  font-family: 'Roboto', sans-serif;
  color: #333;
  font-size: 1.2rem;
  padding: 1.5rem 2rem;
  border-radius: 0.2rem;
  background-color: rgb(255, 255, 255);
```

```
  border: none;
  width: 75%;
  display: block;
  border-bottom: 0.3rem solid transparent;
  transition: all 0.3s;
}

.form__input:placeholder-shown + .form__label {
  opacity: 0;
  visibility: hidden;
  -webkit-transform: translateY(+4rem);
  transform: translateY(+4rem);
}


.button {
  appearance: button;
  background-color: transparent;
  background-image: linear-gradient(to bottom, #fff, #f8eedb);
  border: 0 solid #e5e7eb;
  border-radius: .5rem;
  box-sizing: border-box;
  color: #482307;
  column-gap: 1rem;
  cursor: pointer;
  display: flex;
  font-family: ui-sans-serif,system-ui,-apple-system,system-ui,"Segoe UI",Roboto,"Helvetica
Neue",Arial,"Noto Sans",sans-serif,"Apple Color Emoji","Segoe UI Emoji","Segoe UI
Symbol","Noto Color Emoji";
  font-size: 100%;
  font-weight: 700;
  line-height: 24px;
  margin: 0;
  outline: 2px solid transparent;
  padding: 1rem 1.5rem;
  text-align: center;
  text-transform: none;
```

```
  transition: all .1s cubic-bezier(.4, 0, .2, 1);
  user-select: none;
  -webkit-user-select: none;
  touch-action: manipulation;
  box-shadow: -6px 8px 10px rgba(81,41,10,0.1),0px 2px 2px rgba(81,41,10,0.2);
}

.button:active {
  background-color: #f3f4f6;
  box-shadow: -1px 2px 5px rgba(81,41,10,0.15),0px 1px 1px rgba(81,41,10,0.15);
  transform: translateY(0.125rem);
}

.button:focus {
  box-shadow: rgba(72, 35, 7, .46) 0 0 0 4px, -6px 8px 10px rgba(81,41,10,0.1), 0px 2px 2px
rgba(81,41,10,0.2);
}


.main-body{
  display: flex;
  flex-direction: row;
  width: 75%;
  justify-content:space-around;
}

.button1{
  appearance: button;
  background-color: transparent;
  background-image: linear-gradient(to bottom, rgb(160, 245, 174), #37ee65);
  border: 0 solid #e5e7eb;
  border-radius: .5rem;
  box-sizing: border-box;
  color: #482307;
  column-gap: 1rem;
  cursor: pointer;
  display: flex;
```

```
  font-family: ui-sans-serif,system-ui,-apple-system,system-ui,"Segoe UI",Roboto,"Helvetica Neue",Arial,"Noto Sans",sans-serif,"Apple Color Emoji","Segoe UI Emoji","Segoe UI Symbol","Noto Color Emoji";
  font-size: 100%;
  font-weight: 700;
  line-height: 24px;
  margin: 0;
  outline: 2px solid transparent;
  padding: 1rem 1.5rem;
  text-align: center;
  text-transform: none;
  transition: all .1s cubic-bezier(.4, 0, .2, 1);
  user-select: none;
  -webkit-user-select: none;
  touch-action: manipulation;
  box-shadow: -6px 8px 10px rgba(81,41,10,0.1),0px 2px 2px rgba(81,41,10,0.2);
  display: none;
}

.button2{
  appearance: button;
  background-color: transparent;
  background-image: linear-gradient(to bottom, rgb(252, 162, 162), #ee3737);
  border: 0 solid #e5e7eb;
  border-radius: .5rem;
  box-sizing: border-box;
  color: #482307;
  column-gap: 1rem;
  cursor: pointer;
  display: flex;
  font-family: ui-sans-serif,system-ui,-apple-system,system-ui,"Segoe UI",Roboto,"Helvetica Neue",Arial,"Noto Sans",sans-serif,"Apple Color Emoji","Segoe UI Emoji","Segoe UI Symbol","Noto Color Emoji";
  font-size: 100%;
  font-weight: 700;
  line-height: 24px;
  margin: 0;
```

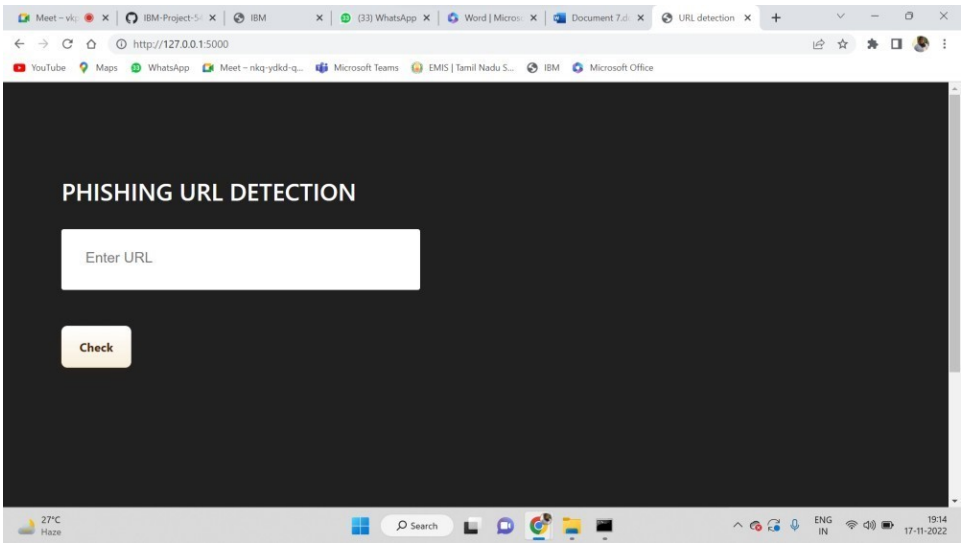
```
  outline: 2px solid transparent;
  padding: 1rem 1.5rem;
  text-align: center;
  text-transform: none;
  transition: all .1s cubic-bezier(.4, 0, .2, 1);
  user-select: none;
  -webkit-user-select: none;
  touch-action: manipulation;
  box-shadow: -6px 8px 10px rgba(81,41,10,0.1),0px 2px 2px rgba(81,41,10,0.2);
  display: none;
}

.right {
  right: 0px;
  width: 300px;
}

@media (max-width: 576px) {
  .form {
    width: 100%;
  }
}
.abc{
  width: 50%;
}
```

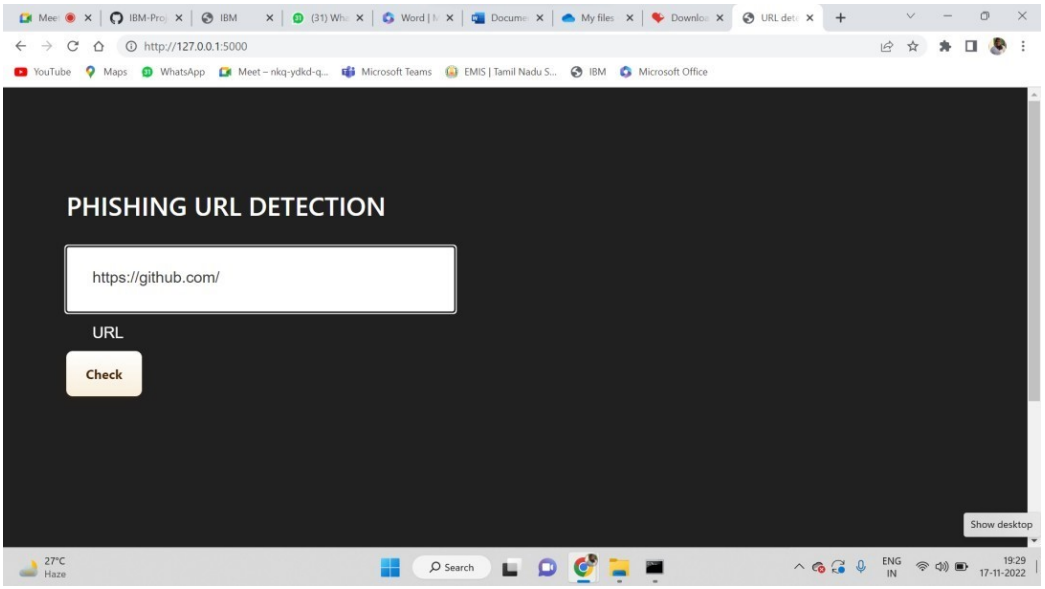
7.5 SOLUTIONING

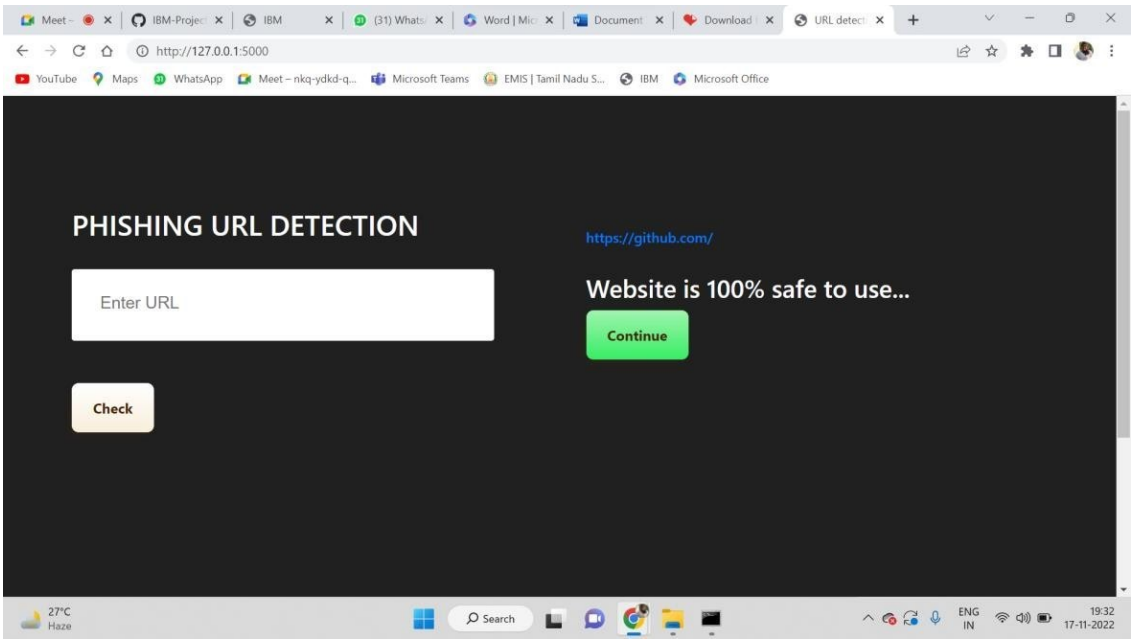
app.py in anaconda prompt

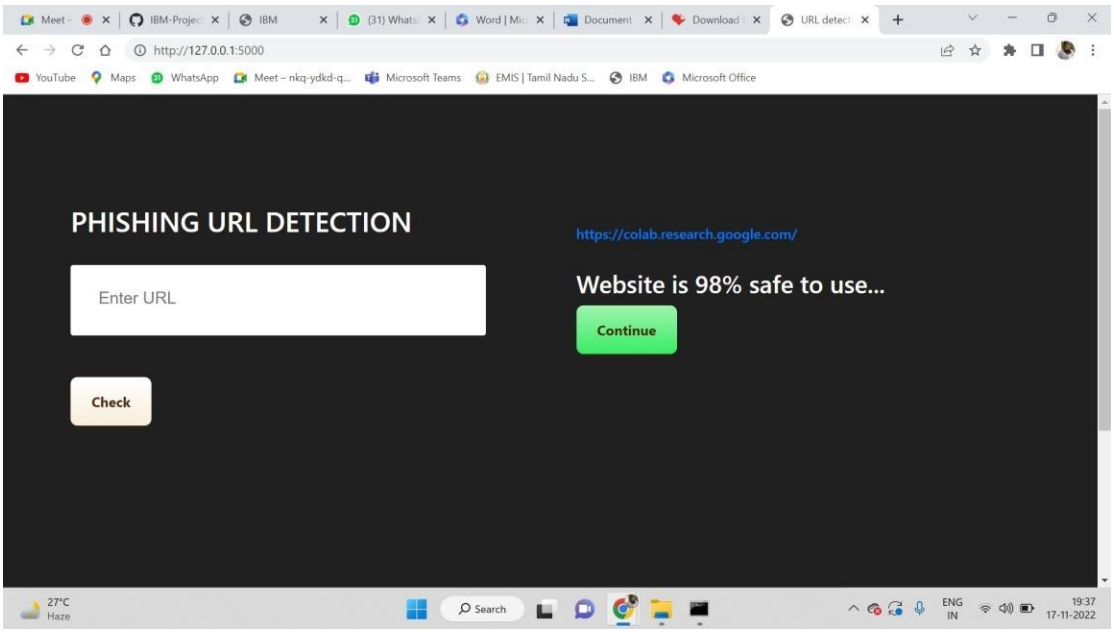
```
(base) C:\Users\Teena Carolin>cd C:\Users\Teena Carolin\Desktop\26297\Code\Project_Folder\Flask

(base) C:\Users\Teena Carolin\Desktop\26297\Code\Project_Folder\Flask>flask run
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: off
 * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```

This is the home page of the web application(main.html)



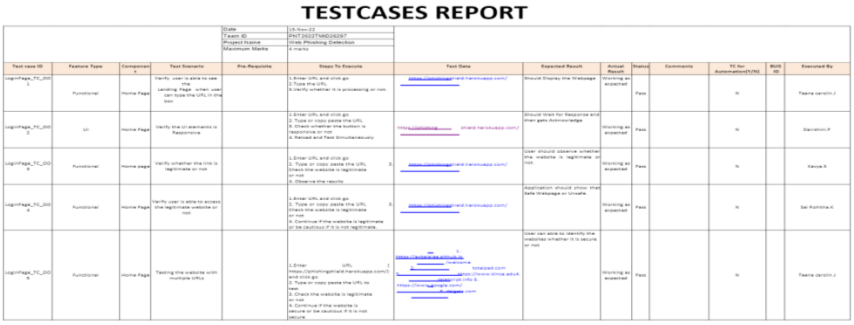

http://127.0.0.1:5000
YouTube
Maps
WhatsApp
Meet – nkq-ydkd-q...
Microsoft Teams
EMIS | Tamil Nadu S...
IBM
Microsoft Office
PHISHING URL DETECTION
https://github.com/
URL
Check
Show desktop
27°C
Haze
Search
ENG
IN
19:29
17-11-2022


http://127.0.0.1:5000
YouTube
Maps
WhatsApp
Meet – nkq-ydkd-q...
Microsoft Teams
EMIS | Tamil Nadu S...
IBM
Microsoft Office
PHISHING URL DETECTION
Enter URL
Check
https://github.com/
Website is 100% safe to use...
Continue
27°C
Haze
Search
ENG
IN
19:32
17-11-2022



8.TESTING

8.1 TEST CASES



8.2 USER ACCEPTANCE TESTING

Purpose of Document

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

Defect Analysis

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

were resolved.

Resolution	Severity 1	Severity 2	Severity 3	Severity 4	Subtotal
By Design	10	4	2	3	20
Duplicate	1	0	3	0	4
External	2	3	0	1	6
Fixed	10	2	4	20	36
Not Reproduced	0	0	1	0	1
Skipped	0	0	0	0	0
Won't Fix	0	0	2	1	3
Totals	23	9	12	25	70

Test Case Analysis

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

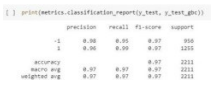
Section	Total Cases	Not Tested	Fail	Pass
Print Engine	10	0	0	10
Client Application	50	0	0	50
Security	5	0	0	4
Outsource Shipping	3	0	0	3
Exception reporting	10	0	0	9
Final Report Output	10	0	0	10
Version Control	4	0	0	4

9.RESULTS

9.1 PERFORMANCE METRICS

Model Performance Testing:

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

S.No.	Parameter	Values	Screenshot
1.	Metrics	**Classification Model: Gradient Boosting Classification** Accuray Score- 97.1%	
2.	Tune the Model	Hyperparameter Tuning - 97% Validation Method – KFOLD & Cross Validation Method	

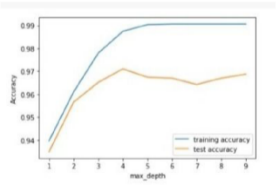
1.METRICS:

CLASSIFICATION REPORT:

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

	precision	recall	f1-score	support
-1	0.98	0.95	0.97	956
1	0.96	0.99	0.97	1255
accuracy			0.97	2211
macro avg	0.97	0.97	0.97	2211
weighted avg	0.97	0.97	0.97	2211

PERFORMANCE:



	ML Model	Accuracy	f1_score	Recall	Precision
0	Support Vector Machine	0.957	0.963	0.982	0.966
1	Logistic Regression	0.924	0.933	0.947	0.927
2	K-Nearest Neighbors	0.953	0.959	0.990	0.989
3	Decision Tree	0.958	0.963	0.992	0.991
4	Gradient Boosting Classifier	0.971	0.975	0.992	0.985
5	Random Forest	0.964	0.969	0.992	0.989

2.TUNE THE MODEL – HYPERPARAMETER TUNING

```
gbc.fit(X_train,y_train)
```

```
GradientBoostingClassifier(learning_rate=0.7, max_depth=4)
```

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)
```

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 AND DISADVANTAGES:

ADVANTAGES:

- In this system , we have used Gradient boosting algorithm which has better performance when compared to other traditional classifications algorithms.
- User can purchase products online and make payments securely without any hesitation.

DISADVANTAGES:

- This system won’t work, if the Internet connection lost.

11.CONCLUSION

The demonstration of phishing is turning into an advanced danger to this quickly developing universe of innovation. The project was carried out in Anaconda IDE and was written in Python. The proposed method uses four machine learning classifiers to achieve this and a comparative study of the six algorithms was made. A good accuracy score was also achieved. The six algorithms used are K-Nearest neighbor, Support vector Algorithm, Logistic regression Decision Tree ,Gradient Boosting Algorithm & Random Forest Classifier. All the six classifiers gave promising results with the best being Gradient Boosting Classifier with an accuracy score of 97.1%. The accuracy score might vary with datasets and other algorithms. Gradient Boosting Algorithm is an ensemble classifier and hence the high accuracy. This model can be deployed in real time to detect the URLs as phishing or legitimate.

12.FUTURE SCOPE

Further work can be done to enhance the model by using ensembling models to get greater accuracy score. Ensemble methods is a machine learning technique that combines many base models to generate an optimal predictive model. Further reaching future work would be combining multiple classifiers, trained on different aspects of the same training set, into a single classifier that may provide a more robust prediction than any of the single classifiers on their own. The project can also include other variants of phishing like smishing, vishing, etc. to complete the system. The collections will ideally grow incrementally over time so there will need to be a way to apply a classifier incrementally to the new data, but also potentially have this

classifier receive feedback that might modify it over time.

13.APPENDIX

GitHub Link: https://github.com/IBM-EPBL

Project Demo Link: https://github.com/IBM-EPBL/IBM-Project-54261658764359/blob/main/Final%20Deliverables/Demo%20Video.mp4