

WEB PHISHING DETECTION

IBM PROJECT REPORT

Submitted by

DARATHI J (913119104015)

BARUNI PRIYA TS(913119104010)

SORNA V (913119104100)

SWATHEKA R (913119104111)

In partial fulfillment for the award of the degree

of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE ENGINEERING

**VELAMMAL COLLEGE OF ENGINEERING AND TECHNOLOGY
MADURAI**

ANNA UNIVERSITY : CHENNAI 600 025

JUNE 2022

CHAPTER NO.	TITLE	PAGE NO*
1.	INTRODUCTION	4
	1.1 PROJECT OVERVIEW	5
	1.2 PURPOSE	5
2.	LITERATURE SURVEY	6
	2.1 EXISTING PROBLEM	6
	2.2 REFERENCES	7
	2.3 PROBLEM STATEMENT DEFINITION	8
3.	IDEATION AND PROPOSED SOLUTION	
	3.1 EMPATHY MAP CANVAS	8
	3.2 IDEATION AND BRAINSTORMING	10
	3.3 PROPOSED SOLUTION	11
	3.4 PROBLEM SOLUTION FIT	12
4.	REQUIREMENT ANALYSIS	
	4.1 FUNCTIONAL REQUIREMENTS	14
	4.2 NON-FUNCTIONAL REQUIREMENTS	14
5.	PROJECT DESIGN	
	5.1 DATA FLOW DIAGRAM	15
	5.2 SOLUTION AND TECHNICAL ARCHITECTURE	16
	5.3 USER STORIES	17
6.	PROJECT PLANNING AND SCHEDULING	
	6.1 SPRINT PLANNING AND ESTIMATION	19
	6.2 SPRINT DELIVERY AND SCHEDULE	19
	6.3 REPORTS FROM JIRA	20
7.	CODING AND SOLUTIONING	
	7.1 FEATURE 1	20
	7.2 FEATURE 2	21
	7.3 DATABASE SCHEMA	21

8.	TESTING	
	8.1 TEST CASES	23
	8.2 USER ACCEPTANCE TESTING	23
9.	RESULTS	
	9.1 PERFORMANCE METRICS	25
10.	ADVANTAGES AND DISADVANTAGES	26
11.	CONCLUSIONS	27
12.	FUTURE SCOPE	28
13.	APPENDIX	28
	SOURCE CODE	28
	GITHUB AND PROJECT DEMO LINK	45

Abstract

Phishing attack is a simplest way to obtain sensitive information from innocent users. Aim of the phishers is to acquire critical information like username, password and bank account details. Cyber security persons are now looking for trustworthy and steady detection techniques for phishing websites detection. This paper deals with machine learning technology for detection of phishing URLs by extracting and analyzing various features of legitimate and phishing URLs. Decision Tree, random forest and Support vector machine algorithms are used to detect phishing websites. Aim of the paper is to detect phishing URLs as well as narrow down to best machine learning algorithm by comparing accuracy rate, false positive and false negative rate of each algorithm

1. INTRODUCTION

Nowadays Phishing becomes a main area of concern for security researchers because it is not difficult to create the fake website which looks so close to legitimate website. Experts can identify fake websites but not all the users can identify the fake website and such users become the victim of phishing attack. Main aim of the attacker is to steal banks account credentials. In United States businesses, there is a loss of US\$2billion per year because their clients become victim to phishing [1]. In 3rd Microsoft Computing Safer Index Report released in February 2014, it was estimated that the annual worldwide impact of phishing could be as high as \$5 billion Phishing attacks are becoming successful because lack of user awareness. Since phishing attack exploits the weaknesses found in users, it is very difficult to mitigate them but it is very important to enhance phishing detection techniques. The general method to detect phishing websites by updating blacklisted URLs, Internet Protocol (IP) to the antivirus database which is also known as "blacklist" method. To evade blacklists attackers uses creative techniques to fool users by modifying the URL to appear legitimate via obfuscation and many other simple techniques including: fast-flux, in which proxies are automatically generated to host the web-page; algorithmic generation of new

URLs; etc. Major drawback of this method is that, it cannot detect zero-hour phishing attack. Heuristic based detection which includes characteristics that are found to exist in phishing attacks in reality and can detect zero-hour phishing attack, but the characteristics are not guaranteed to always exist in such attacks and false positive rate in detection is very high [3]. To overcome the drawbacks of blacklist and heuristics based method, many security researchers now focused on machine learning techniques. Machine learning technology consists of a many algorithms which requires past data to make a decision or prediction on future data. Using this technique, algorithm will analyze various blacklisted and legitimate URLs and their features to accurately detect the phishing websites including zero- hour phishing websites.

1.1. Project Overview

In this project, we built a mechanism to detect phishing websites. Our methodology uses not just traditional URL based or content based rules but rather employs the machine learning technique to identify not so obvious patterns and relations in the data. We have used features from various domain spanning from URL to HTML tags of the webpage, from embedded URLs to favicon.

Classifying most websites correctly and proving the effectiveness of the machine learning based technique to attack the problem of phishing websites. We provided the output as a user-friendly web platform which can further be extended to a browser extension to provide safe and healthy online space to the users.

This project employs machine learning technique for prediction task and supervised learning algorithm namely random forest technique for exploring results

1.2 Purpose

The purpose of this project is to design an intelligent system for detecting phishing websites. Phishing is one of the social attack which aims in stealing sensitive information of the users such as login credentials, credit card numbers etc. Here we have collected phishing dataset from phish Tanks as well as from phishing sites and are compared with the algorithms which

classifies the phishing dataset into phishing or legitimate. The goal is to retrieve personal information of the recipient by making them believe that the message is something they want or need i.e., request from the bank etc. The aim is to develop a model to safeguard users from phishing attacks. This can be done using unique features of phishing websites. The goal of our paper is to develop a web application which notifies the user when it detects a phishing site.

2 . LITERATURE SURVEY

2.1 Existing Problem

Malicious links will lead to a website that often steals login credentials or financial information like credit card numbers. Attachments from phishing emails can contain malware that once opened can leave the door open to the attacker to perform malicious behavior from the user's computer. Phishing targets individuals and private citizens each day. Additionally, cyber criminals will target businesses. Business email compromise (BEC) scams accounted for over \$12 million in losses last year, according to [Retruster](#). U.S. drug company Upsher-Smith Laboratories lost over \$50 million in just three weeks in 2014. Attackers impersonated the company's CEO and were able to convince the company's accounts payable coordinator to make nine wire transfers. Higher education wasn't immune from phishing. MacEwan University in Canada had \$11.8 million taken in 2017 when phishers imitated construction companies and sent fake invoices. Beyond monetary damages, businesses who are breached lose public trust and must work to secure their databases. Many companies are required to notify their customers of a breach, pay regulatory fines, and lose customers as a result. The long answer is that it is a growing problem for businesses each day which requires greater defense. Phishing is the most popular attack vector for criminals and has grown 65% in the last year, according to [Retruster](#). DataShield is here to explain phishing, how attacks have affected businesses, how this form of cybercrime is growing, and how to defend against them

2.2 References

1. Paper 1

Authors: Said Salloum, Tarek Gaber, Sunil Vadera and Khaled Shaalan

Title: A Systematic Literature Review on Phishing Email Detection Using
Natural Language Processing Techniques

Published Journal:IEEE Access (Volume: 10)

Published Date: 14 June 2022

2. Paper 2

Authors:Jian Mao, Wenqian Tian, Pei Li, Tao Wei and Zhenkai Liang

Title:Phishing-Alarm: Robust and Efficient Phishing Detection via Page component.

Published Journal :IEEE Access (Volume: 5)

Published Date: 23 August 2017

3. Paper 3

Authors:Peng Yang, Guangzhen Zhao and Peng Zeng

Title:Phishing Website Detection Based on Multidimensional Features
Driven by Deep Learning

Published Journal:IEEE Access (Volume: 7)

Published Date: 11 January 2019

4. Paper 4

Authors:Saad Al-Ahmadi, Afrah Alotaibi and Omar Alsaleh

Title:PDGAN: Phishing Detection With Generative Adversarial Networks

Published Journal:IEEE Access (Volume: 10)

Published Date: 18 April 2022

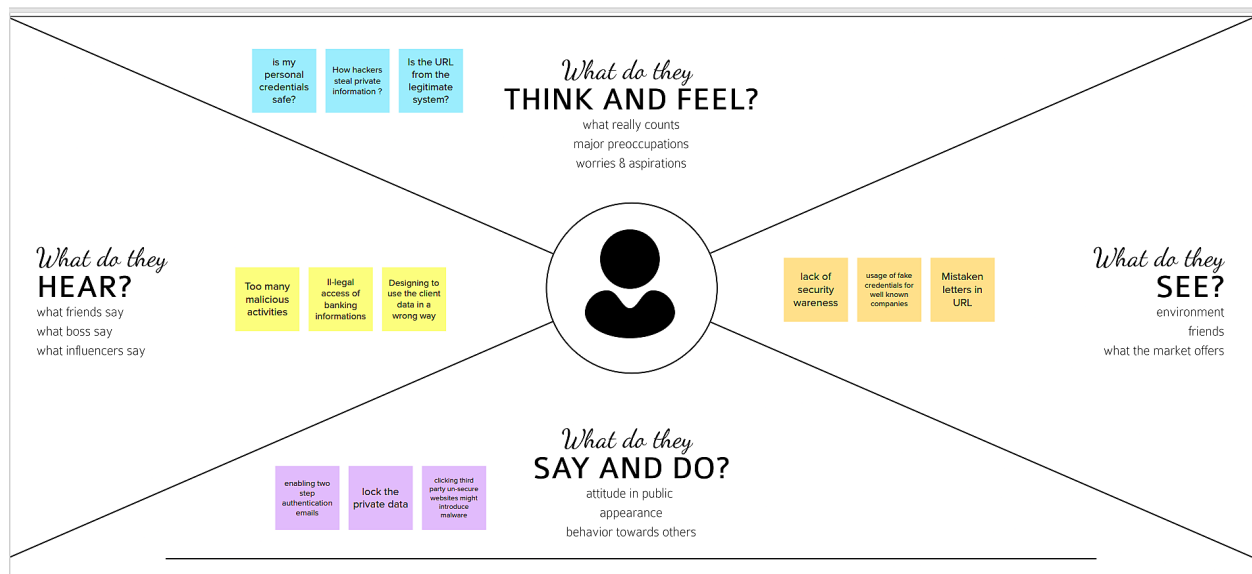
2.3 Problem Statement Definition

There are a number of users who purchase products online and make payments through e-banking. There are e-banking websites that ask users to provide sensitive data such as username, password & credit card details, etc., often for malicious reasons. This type of e-banking website is known as a phishing website. Web phishing is one of many security threats to web services on the Internet.

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 and techniques to extract the phishing datasets criteria to classify their legitimacy. The e-banking phishing website can be detected based on some important characteristics like URL and domain identity, and security and encryption criteria in the final phishing detection rate.

3. IDEATION & PROPOSED SOLUTION

3.1 Empathy Map Canvas



- ✓ Enabling two step authentication emails
- ✓ Lock the private data
- ✓ Clicking third party unsecure website might introduce malware

PAIN:

- DNS poisoning
- Malware
- Keyloggers

GAIN:

- Hackers can easily obtain user's money
- Getting some privileged access
- private email domain for personal use

3.2 Ideation and Brainstorming

Brainstorm & idea prioritization

Use this template in your own brainstorming sessions or your team can utilize their imagination and start shaping concepts even if you're not sitting in the same room.

- 10 minutes to prepare
- 1 hour to ideate
- 15 minutes to prioritize

Before you collaborate

1. Select all participants to join a meeting with this session. Review what you need to do to get going.

2. [Get started](#)

3. [Get started](#)

4. [Get started](#)

5. [Get started](#)

6. [Get started](#)

7. [Get started](#)

8. [Get started](#)

9. [Get started](#)

10. [Get started](#)

Define your problem statement

What problem are you trying to solve? Frame your problem as a clear, specific statement. This will be the focus of your brainstorm.

1. [Get started](#)

2. [Get started](#)

3. [Get started](#)

4. [Get started](#)

5. [Get started](#)

6. [Get started](#)

7. [Get started](#)

8. [Get started](#)

9. [Get started](#)

10. [Get started](#)

Brainstorm

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

1. [Get started](#)

2. [Get started](#)

3. [Get started](#)

4. [Get started](#)

5. [Get started](#)

6. [Get started](#)

7. [Get started](#)

8. [Get started](#)

9. [Get started](#)

10. [Get started](#)

Group ideas

Take your ideas and group them into clusters or related ideas on your page. To do so, it's important to give each cluster a name. The cluster is larger than the ideas it contains, so it's important to give it a name that is larger than the ideas it contains.

1. [Get started](#)

2. [Get started](#)

3. [Get started](#)

4. [Get started](#)

5. [Get started](#)

6. [Get started](#)

7. [Get started](#)

8. [Get started](#)

9. [Get started](#)

10. [Get started](#)

Prioritize

Now that you have a list of ideas, it's time to prioritize them. Use the grid below to rank each idea based on its importance and feasibility.

1. [Get started](#)

2. [Get started](#)

3. [Get started](#)

4. [Get started](#)

5. [Get started](#)

6. [Get started](#)

7. [Get started](#)

8. [Get started](#)

9. [Get started](#)

10. [Get started](#)

After you collaborate

Now that you have a list of ideas, it's time to prioritize them. Use the grid below to rank each idea based on its importance and feasibility.

1. [Get started](#)

2. [Get started](#)

3. [Get started](#)

4. [Get started](#)

5. [Get started](#)

6. [Get started](#)

7. [Get started](#)

8. [Get started](#)

9. [Get started](#)

10. [Get started](#)

Brainstorm & idea prioritization

Use this template in your own brainstorming sessions or your team can utilize their imagination and start shaping concepts even if you're not sitting in the same room.

- 10 minutes to prepare
- 1 hour to ideate
- 15 minutes to prioritize

Before you collaborate

1. Select all participants to join a meeting with this session. Review what you need to do to get going.

2. [Get started](#)

3. [Get started](#)

4. [Get started](#)

5. [Get started](#)

6. [Get started](#)

7. [Get started](#)

8. [Get started](#)

9. [Get started](#)

10. [Get started](#)

Define your problem statement

What problem are you trying to solve? Frame your problem as a clear, specific statement. This will be the focus of your brainstorm.

1. [Get started](#)

2. [Get started](#)

3. [Get started](#)

4. [Get started](#)

5. [Get started](#)

6. [Get started](#)

7. [Get started](#)

8. [Get started](#)

9. [Get started](#)

10. [Get started](#)

Brainstorm

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

1. [Get started](#)

2. [Get started](#)

3. [Get started](#)

4. [Get started](#)

5. [Get started](#)

6. [Get started](#)

7. [Get started](#)

8. [Get started](#)

9. [Get started](#)

10. [Get started](#)

Group ideas

Take your ideas and group them into clusters or related ideas on your page. To do so, it's important to give each cluster a name. The cluster is larger than the ideas it contains, so it's important to give it a name that is larger than the ideas it contains.

1. [Get started](#)

2. [Get started](#)

3. [Get started](#)

4. [Get started](#)

5. [Get started](#)

6. [Get started](#)

7. [Get started](#)

8. [Get started](#)

9. [Get started](#)

10. [Get started](#)

Prioritize

Now that you have a list of ideas, it's time to prioritize them. Use the grid below to rank each idea based on its importance and feasibility.

1. [Get started](#)

2. [Get started](#)

3. [Get started](#)

4. [Get started](#)

5. [Get started](#)

6. [Get started](#)

7. [Get started](#)

8. [Get started](#)

9. [Get started](#)

10. [Get started](#)

After you collaborate

Now that you have a list of ideas, it's time to prioritize them. Use the grid below to rank each idea based on its importance and feasibility.

1. [Get started](#)

2. [Get started](#)

3. [Get started](#)

4. [Get started](#)

5. [Get started](#)

6. [Get started](#)

7. [Get started](#)

8. [Get started](#)

9. [Get started](#)

10. [Get started](#)

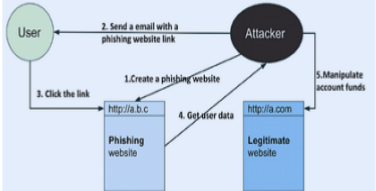
3.3 Proposed Solution

S. No.	Parameter	Description
1	Problem Statement (Problem to be solved)	<ul style="list-style-type: none">● It is important to be cautious while we provide sensitive data such as username, password, credit card details, personal information, etc.● As the number of users who purchase products through online and make payments through e-banking increases, the number of fraudulent e-banking websites also increases who try to get sensitive information from the users for malicious reasons.● Such a kind of an e-banking website is an example of a phishing website.● They impersonate as a legitimate entity to steal private information from us.● These activities lead to information disclosure and property damage.● Web phishing is becoming one of many security threats to web services on the Internet.

2	Idea / Solution description	<ul style="list-style-type: none"> ● 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 and techniques to extract the phishing datasets criteria to classify their legitimacy. ● The e-banking phishing website can be detected based on some important characteristics like URL and domain identity, and security and encryption criteria in the final phishing detection rate. ● The real-time prediction will include whitelist filtering, blacklist interception and Machine Learning (ML) prediction.
---	-----------------------------	---

3.4 Problem Solution Fit

- The objective of phishing website URLs is to purloin the personal information like user name, passwords and online banking transactions
- Phishers use the websites which are visually and semantically similar to those real websites. As technology continues to grow, phishing techniques started to progress rapidly and this needs to be prevented by using anti-phishing mechanisms to detect phishing.
- We would create an interactive and responsive website that will be used to detect whether a website is legitimate or phishing.

Define CS, fit into CC	1. CUSTOMER SEGMENT(S) CS Who is your customer? <ul style="list-style-type: none"> Protect yourself and your family against malicious websites with the platform for free. With the platform, protecting your staff, data, brand, and your customer from malicious websites has never been easier. Proactively protect multiple customers against malicious websites at once with all-in-one platform. The platform can be used for government embeds to provide 100% security and privacy. 	6. CUSTOMER CONSTRAINTS CC What constraints prevent your customers from taking action or limit their choices of solutions? <ul style="list-style-type: none"> The limitations of the web phishing detection approaches are explored by means of detection time, detection rate, and storage complexity to verify the level of robustness against the phishing attack. Thus most of the recent web phishing detection approaches lag in feature selection mechanism as they use handcrafted features to detect the attack. 	5. AVAILABLE SOLUTIONS AS Which solutions are available to the customers when they face the problem or need to get the job done? What have they tried in the past? What pros & cons do these solutions have? <ul style="list-style-type: none"> Legitimate websites prevent web scraping by several techniques in respect to obfuscation using CSS sprites to display important data, replacing text with images. Spam filtering techniques are used to identify unsolicited emails before the user reads or clicks the link. When users visit a phishing web page that looks like a legitimate website, many people do not remember the legitimate website's domain name, particularly for some start-ups or unknown companies, so users cannot recognise the phishing website based on the URL. Some web browsers integrate a security component to detect phishing or malware sites, such as Chrome, which will display warning messages when one visits an unsafe web page. When the website detects that the IP address and device information of the user who is logging in does not match the commonly used information, it is necessary to verify the authenticity of the user 	Explore AS, differentiate
	2. JOBS-TO-BE-DONE / PROBLEMS J&P Which jobs-to-be-done (or problems) do you address for your customers? There could be more than one; explore different sides. <ul style="list-style-type: none"> The objective of phishing website URLs is to purloin the personal information like user name, passwords and online banking transactions. Phishers use the websites which are visually and semantically similar to those real websites. As technology continues to grow, phishing techniques started to progress rapidly and this needs to be prevented by using anti-phishing mechanisms to detect phishing. 	9. PROBLEM ROOT CAUSE RC What is the real reason that this problem exists? What is the back story behind the need to do this job?  <ul style="list-style-type: none"> A phishing attack is a type of cybersecurity threat that targets users directly through email, text or direct messages. During one of these scams, a cybercriminal will pose as a trusted contact to steal data from an unsuspecting user such as login information, account numbers and credit card information. While there are several types of phishing, the main purpose behind all of them is it to steal sensitive information or transfer malware. 	7. BEHAVIOUR BE What does your customer do to address the problem and get the job done? i.e. directly related: find the right solar panel installer, calculate usage and benefits; indirectly associated: customers spend free time on volunteering work <ul style="list-style-type: none"> Customers should take a "trust no one" approach when opening email. Check and verify the "From" address of the email. By carefully reading the email copy, users can typically spot something that seems "off" including: <ul style="list-style-type: none"> An email with an "urgent" request or An email offering the user something that's "too good to be true". Check grammar and spelling. Poor grammar and misspelled words in an email can be red flags. Be wary of generic salutations in an email. Legitimate companies, especially those with which you have accounts or have done business typically will address you by name versus by a generic greeting. Encourage your clients to look for any unusual or odd requests in their emails. Most fraudulent emails contain a request to respond to the email or click a link in it. Avoid clicking links or attachments in emails from unfamiliar sources. 	Focus on J&P, tap into BE, understand RC
Identify strong TR & EM	3. TRIGGERS TR What triggers customers to act? <ul style="list-style-type: none"> Your users lack security awareness . Criminals are (unsurprisingly) following the money . You're not performing sufficient due diligence . Low-cost phishing and ransomware tools are easy to get hold of . Malware is becoming more sophisticated . 	10. YOUR SOLUTION SL If you are working on an existing business, write down your current solution first, fill in the canvas, and check how much it fits reality. If you are working on a new business proposition, then keep it blank until you fill in the canvas and come up with a solution that fits within customer limitations, solves a problem and matches customer behaviour. <ul style="list-style-type: none"> We would create an interactive and responsive website that will be used to detect whether a website is legitimate or phishing. This website is made using different web designing languages which include HTML, CSS, JavaScript and Python. This website is more useful to the user and it is user friendly also. 	8. CHANNELS of BEHAVIOUR CH 8.1 ONLINE What kind of actions do customers take online? Extract online channels from #1 <ul style="list-style-type: none"> Nothing teaches like experience. When employees click on a link or an attachment in a simulated phishing email, it's important to communicate to them that they have potentially put both themselves and the organisation at risk. 8.2 OFFLINE What kind of actions do customers take offline? Extract offline channels from #7 and use them for customer development. <ul style="list-style-type: none"> Phishing awareness training starts with educating your employees on why phishing is harmful, and empowering them to detect and report phishing attempts. Simulated phishing campaigns reinforce employee training, and to understand risk and improve workforce resiliency as these can take many forms, such as mass phishing, spear phishing, and whaling. 	Extract online & offline CH of BE
	4. EMOTIONS: BEFORE / AFTER EM How do customers feel when they face a problem or a job and afterwards? <ul style="list-style-type: none"> Greed- Clicking on fake successful messages. Urgency-Hackers use fake security alerts with exclamation marks. Helpfulness-Hackers and cyber criminals use major tragedies to appeal for help but they are only helping themselves. Fear- Emails that spread fear and phishing links go hand in hand. 			

4. REQUIREMENT ANALYSIS

4.1. Functional Requirements

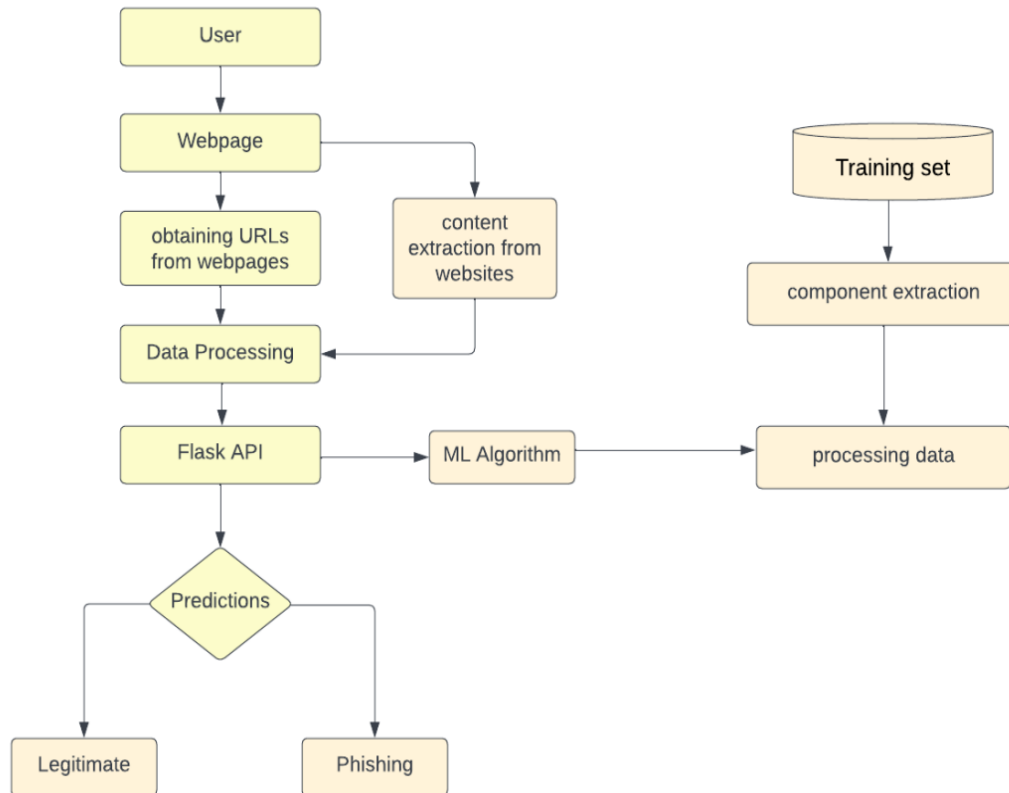
- User Registration- Register by providing information such as your name, email address, password, phone number, and so on.
- User Confirmation - Log in with your registered email address and password. After registering, the user/admin may enter his details to access.
- Building requirements - Create and compare multiple machine learning models for detecting web phishing.
- Verify the website - Here, the user enters the URL to search the banned website.
- Integration - linking the frontend with the created ML model.

4.2. Non – Functional Requirements

- Usability - For detection, any URL must be permitted.
- Security - Notify the user about the harmful website via phone or email.
- Reliability - The online phishing domains must be correctly recognised, and the output must be accurate.
- Performance - Measures how quickly the pages of a website load and display in the result
- Availability - It must be possible for everyone to sign up and log in at any time.

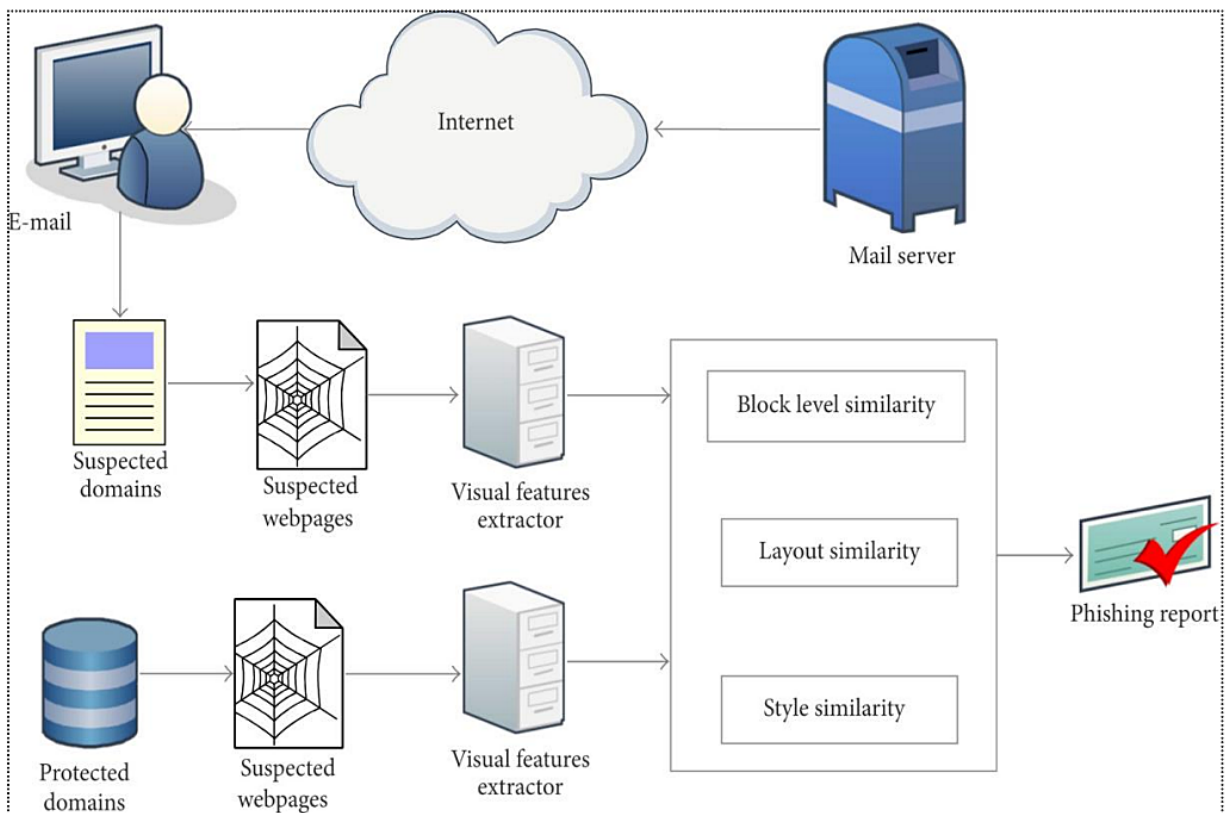
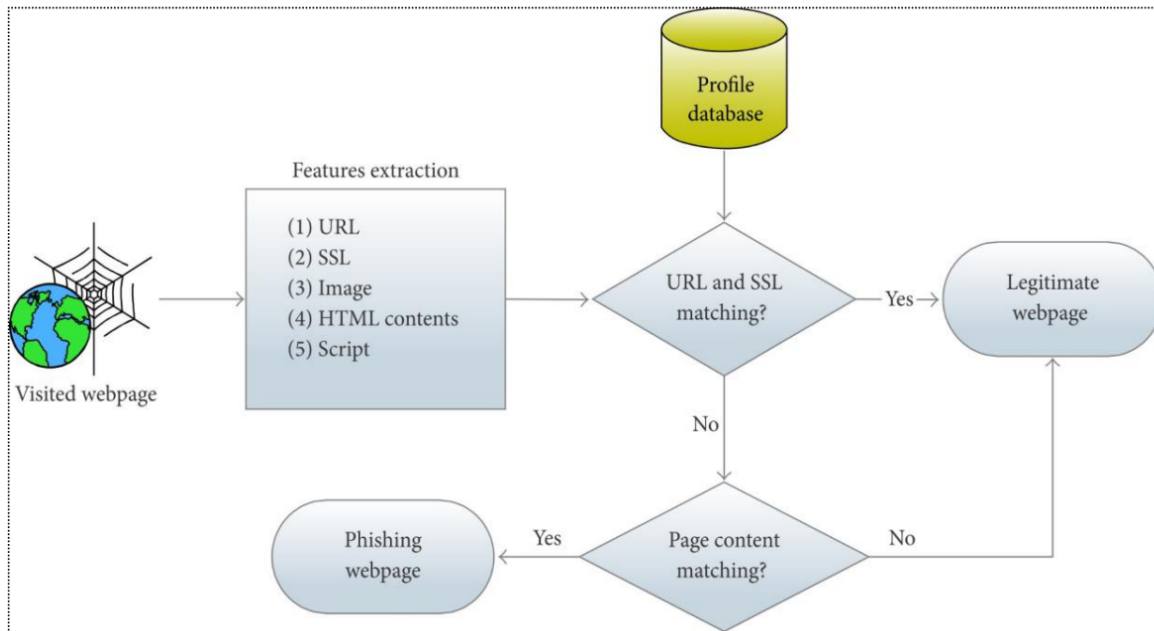
5. PROJECT DESIGN

5.1. Data Flow Diagrams



5.2. Solution & Technical Architecture

- Solution architecture is a discipline of enterprise architecture that seeks to define and describe the application.
- It is a structural design that addresses a set of functional and non-functional requirements.



5.3. User Stories

USER STORIES -> 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	I can sign up for the application as a user by providing my email, password, and password confirmation.	I can access my account / dashboard	High	Sprint-1
		USN-2	When I register for the application as a user, I will get a confirmation email.	I can get a confirmation email and confirm it.	High	Sprint-1
		USN-3	I can sign up for the application as a user through Facebook.	I can register & access the dashboard with Facebook Login	Low	Sprint-2
		USN-4	Using Gmail, I may register as a user for the application.		Medium	Sprint-1

	Login	USN-5	As a user, I can log into the application by inputting email &password		High	Sprint-1
	Dashboard					

6. PROJECT PLANNING & SCHEDULING

6.1. Sprint Planning & Estimation

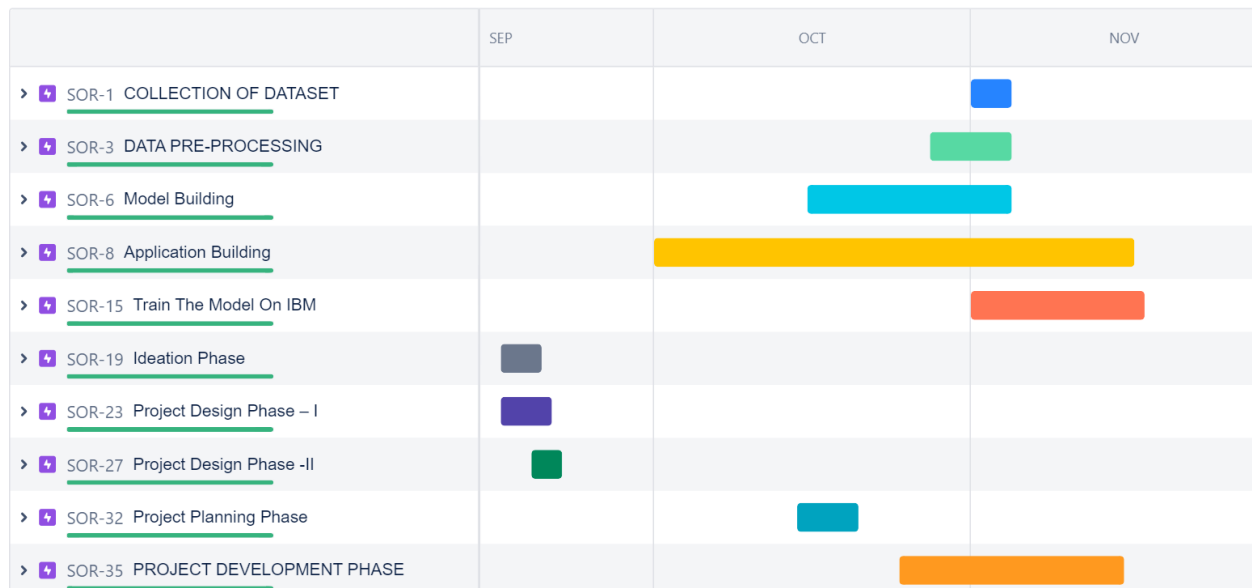
Product Backlog, Sprint Schedule, and Estimation (4 Marks)

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-1	Registration	USN-1	As a user, I can register for the application by entering my email, password, and confirming my password.	1	Low	Sorna V Swatheka R
Sprint-1	Login	USN-2	As a user, I can login with my user credentials	1	Low	Swatheka R Baruni Priya T S
Sprint-2	Dashboard	USN-3	As a user, I will be re-directed to the dashboard after registering or login.	2	High	Darathi J Sorna V
Sprint-2		USN-4	As a user, I can know about the usage of the website via help section in the dashboard	1	Medium	Sorna V Baruni Priya T S
Sprint-3	Feature Extraction	USN-5	In the event, we can extract the features using heuristics and a visual similarity technique.	2	Medium	Darathi J Baruni Priya T S
Sprint-3		USN-6	As a user, I can enter the website link to check if it is phishing or not.	2	High	Darathi J Swatheka R
Sprint-4	Result	USN-7	As a user, I can view the result in the result page with some informations about the entered website.	2	High	Baruni Priya T S Darathi J

6.2. Sprint Delivery Schedule**Project Tracker, Velocity & Burndown Chart: (4 Marks)**

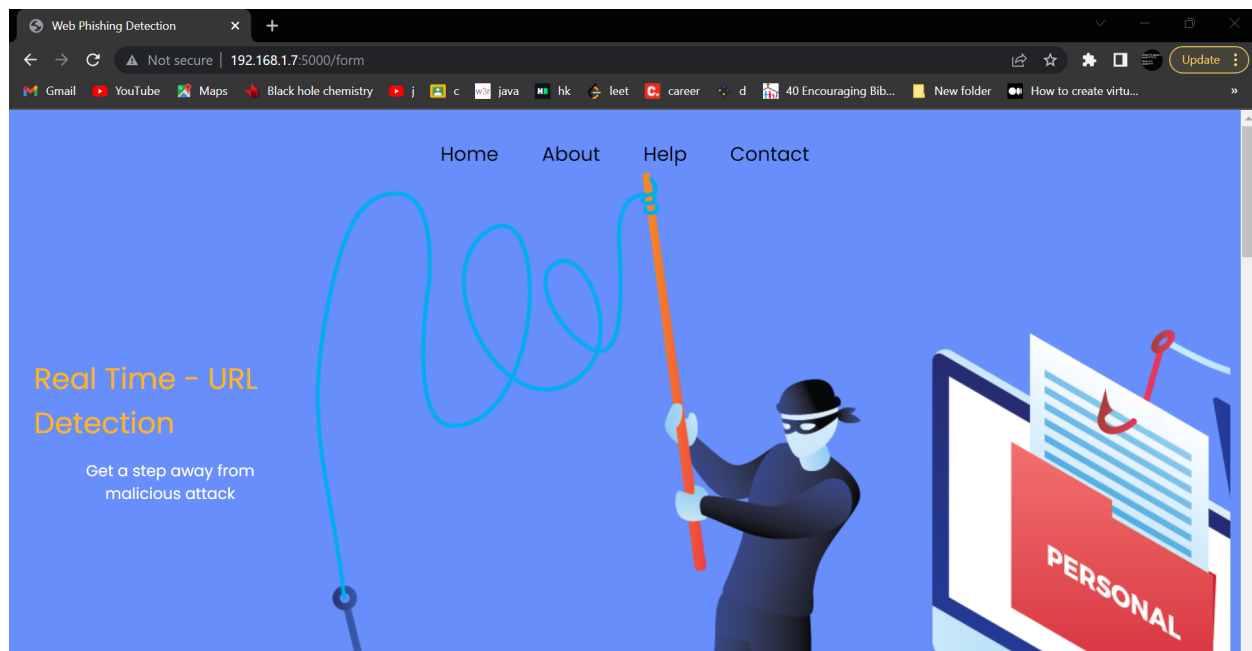
Sprint	Total Story Points	Duration	Sprint Start Date	Sprint End Date (Planned)	Story Points Completed (as on Planned End Date)	Sprint Release Date (Actual)
Sprint-1	20	6 Days	24 Oct 2022	29 Oct 2022	2	29 Oct 2022
Sprint-2	20	6 Days	31 Oct 2022	05 Nov 2022	3	05 Nov 2022
Sprint-3	20	6 Days	07 Nov 2022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

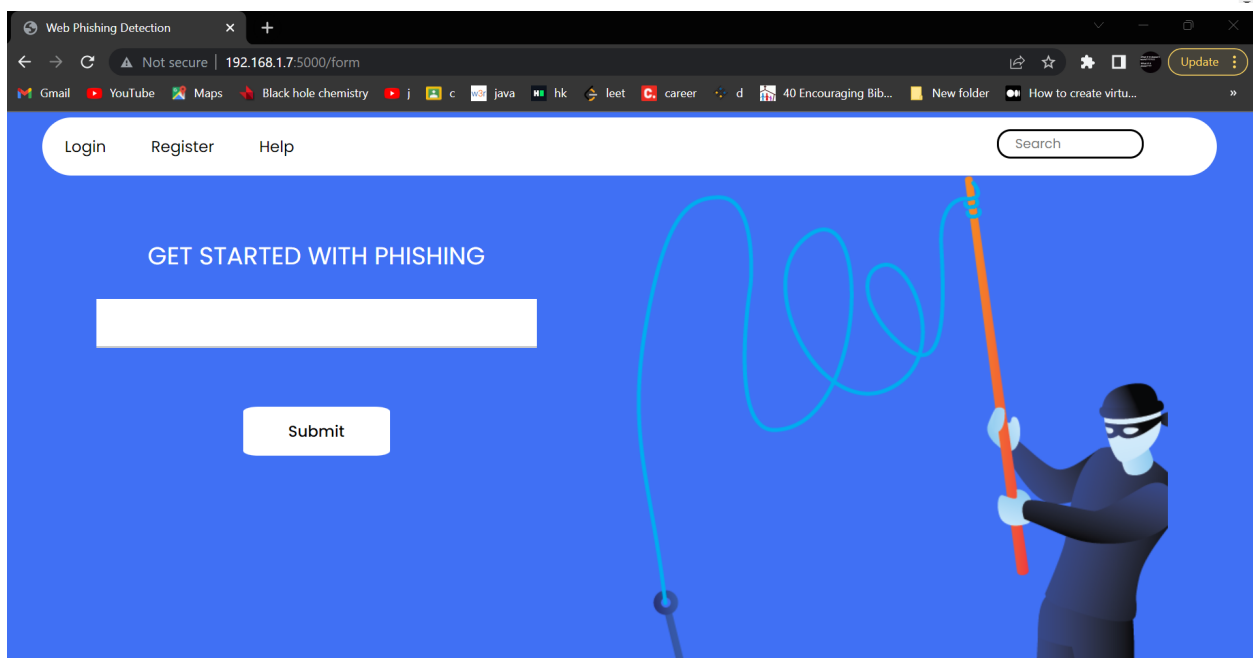
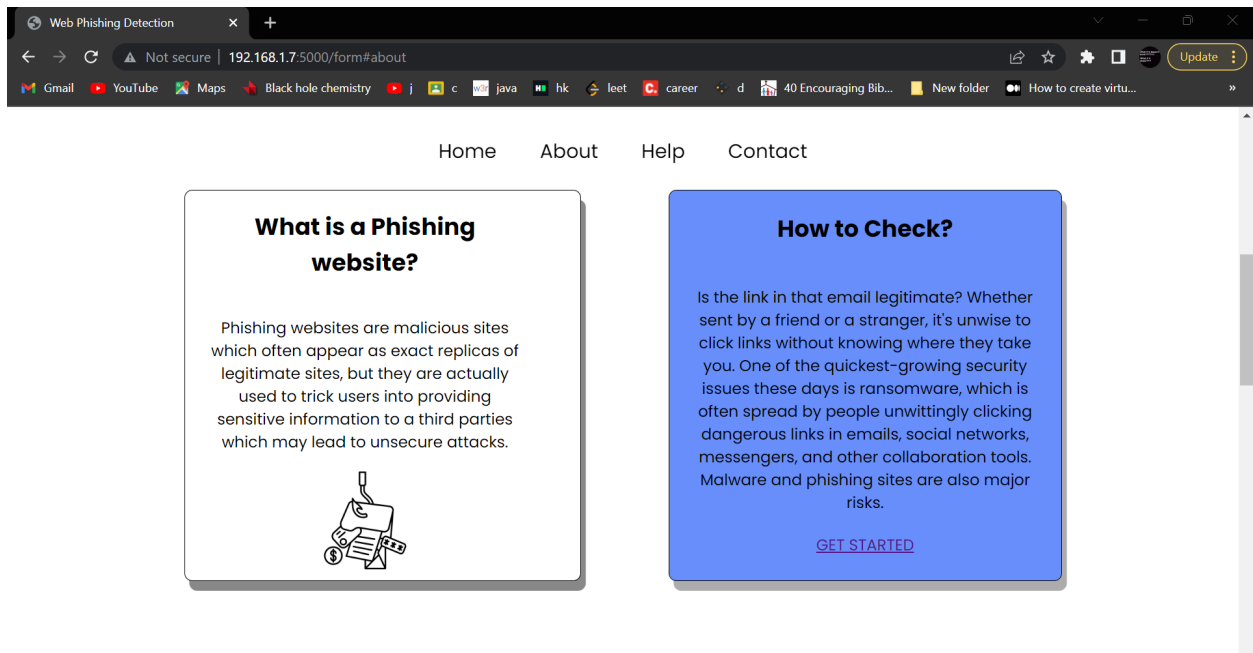
6.3. Reports from JIRA



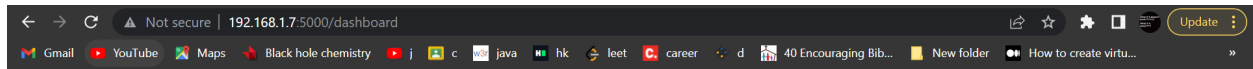
7. CODING & SOLUTIONING

7.1 Feature 1





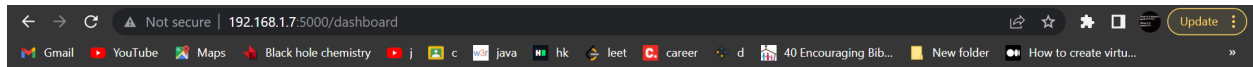
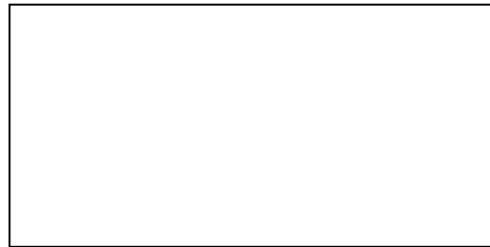
7.2 Feature 2



Home About Help

GET STARTED WITH PHISHING

Submit

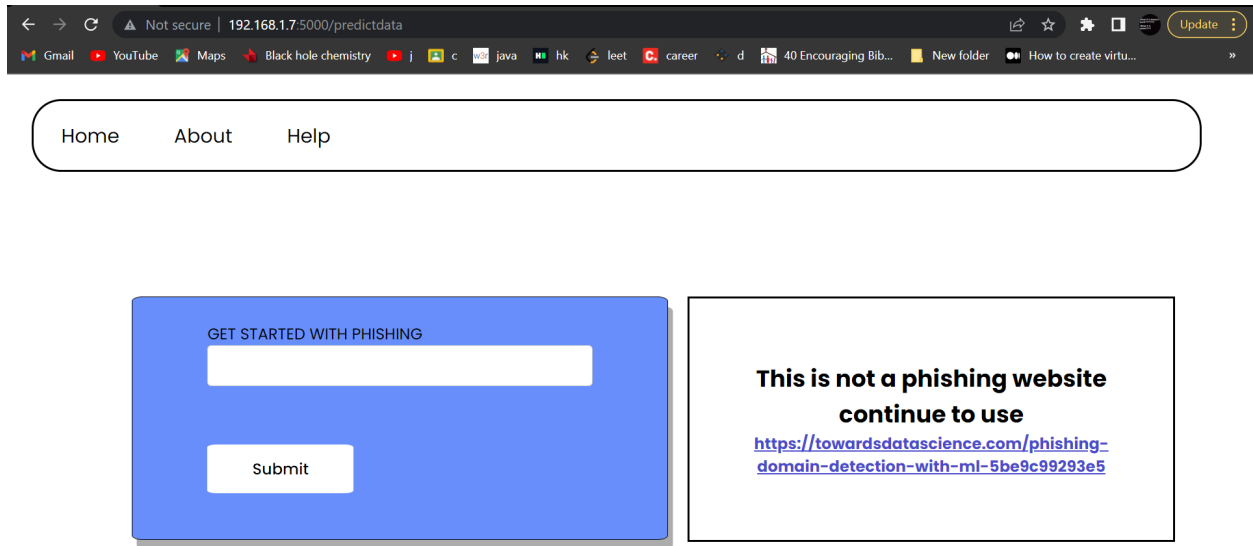


Home About Help

GET STARTED WITH PHISHING

Submit





7.3 Database Schema

8. TESTING

8.1 Test Cases

8.2 User Acceptance Testing

User acceptance testing (UAT), also called application testing or end-user testing, is a phase of software development in which the software is tested in the real world by its intended audience.

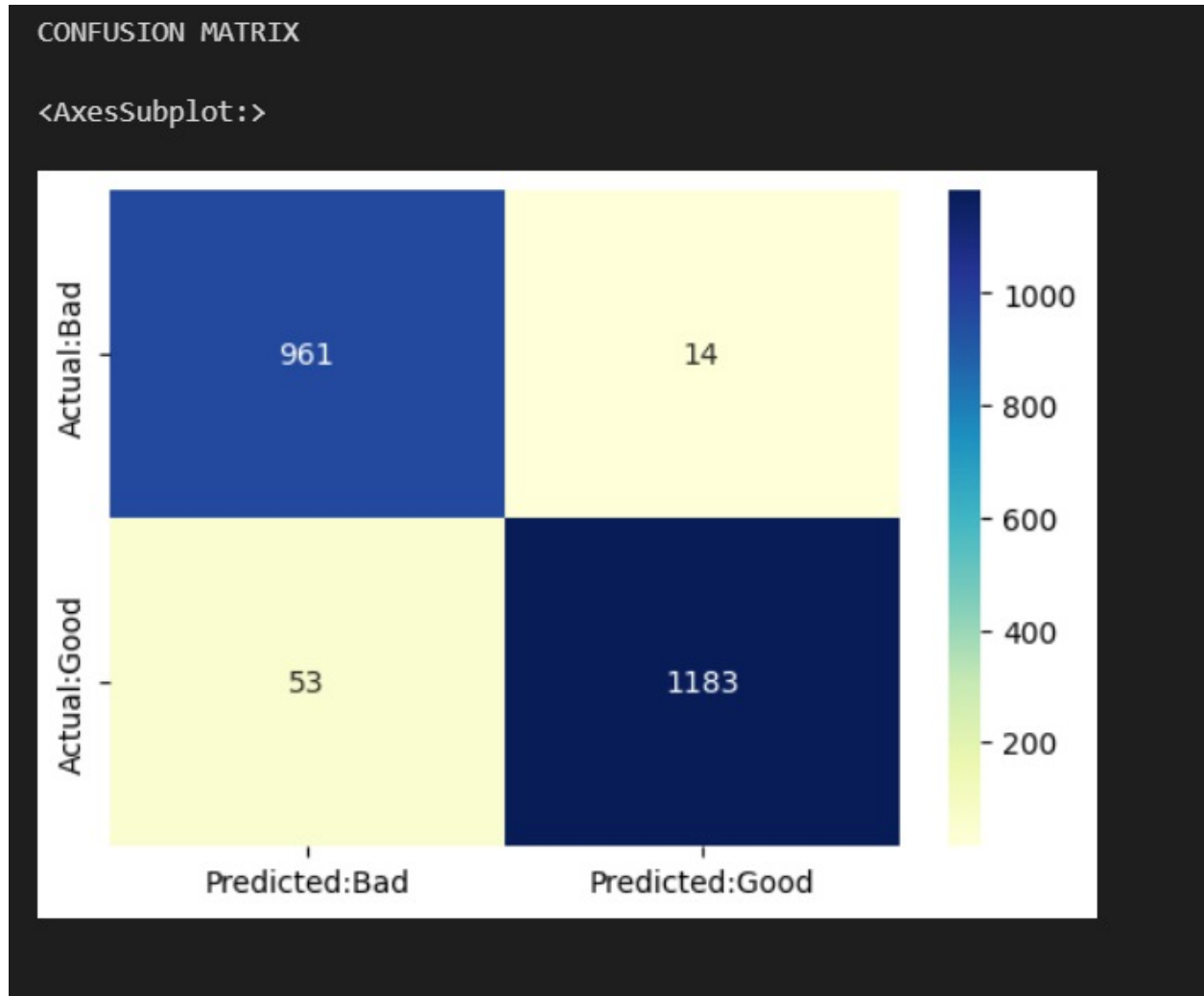
UAT is often the last phase of the **software testing** process and is performed before the tested software is released to its intended market. The goal of UAT is to ensure software can handle real-world tasks and perform up to development specifications.

8.3 Performance Testing

By classification Report,we have

CLASSIFICATION REPORT				
	precision	recall	f1-score	support
Bad	0.95	0.99	0.97	975
Good	0.99	0.96	0.97	1236
accuracy			0.97	2211
macro avg	0.97	0.97	0.97	2211
weighted avg	0.97	0.97	0.97	2211

By plotting it,



9. RESULTS

9.1 Performance Metrics

To evaluate the efficiency of a system, we use certain parameters. For each machine learning model, we calculate the Accuracy, Precision, Recall, F1 Score and ROC curve to determine its performance. Each of these metrics is calculated based on True Positive (TP), True Negative (TN), False Positive (FP) and False Negative (FN).

In the case of URL classification, True Positive (TP) is the number of phishing URLs that are correctly classified as phishing. True Negative (TN) is the number of legitimate URLs that are correctly classified as legitimate. False Positive (FP) is the number of legitimate URLs that are

classified as phishing. False Negative (FN) is the number of phishing URLs that are classified as legitimate. These values are summarized in Table IV called Confusion Matrix.

	Predicted Phishing	Predicted Legitimate
Actual Phishing	TP	FN
Actual Legitimate	FP	TN

	Predicted Phishing	Predicted Legitimate
Actual Phishing	TP	FN
Actual Legitimate	FP	TN

10. ADVANTAGES & DISADVANTAGES

Advantages:

- This system can be used by many E-commerce or other websites in order to have good customer relationship.
- User can make online payment securely.
- Measure the degrees of corporate and employee vulnerability
- Protect valuable corporate and personal data
- Meet industry compliance obligations.

Disadvantages:

Unfortunately, many of the existing phishing-detection tools, especially those that depend on an existing blacklist, suffer limitations such as low detection accuracy and high false alarm that is often caused by either a delay in blacklist update as a result of human verification process involved in classification or perhaps, it can be attributed to human error in classification which may lead to improper classification of the classes.

These critical issues have drawn many researchers to work on various approaches to improve detection accuracy phishing attacks and to minimize false alarm rate. This may cause an error in generating efficient classification results when the suspicious webpage includes language other than English.

11. CONCLUSION

Phishing website attacks are a massive challenge for researchers, and they continue to show a rising trend in recent years.

The importance to safeguard online users from becoming victims of online fraud, divulging confidential information to an attacker among other effective uses of phishing as an attacker's tool, phishing detection tools play a vital role in ensuring a secure online experience for users.

The inconsistent nature of attacks behaviors and continuously changing URL phish patterns require timely updating of the reference model. Therefore, it requires an effective technique to regulate retraining as to enable machine learning algorithm to actively adapt to the changes in phish patterns.

12. FUTURE SCOPE

Phishing is a growing problem for internet users. There are a number of anti-phishing tools available to cope against this problem. Still there are limitation on accuracy because detection techniques are time consuming. Among several machine learning algorithm, Random-forest gives the better result. This work become unique from other existing work by proposing a group of features that can be extracted automatically using our own software tool. In future we can make the system available in mobile devices. In future hybrid technology will be implemented to detect phishing websites more accurately, for which random forest algorithm of machine learning technology and blacklist method will be used.

13. APPENDIX

Source Code:

IBM_app.py

```
from flask import *
import numpy as np
from markupsafe import escape
import pickle
from inputScript import main

import requests

# NOTE: you must manually set API_KEY below using information retrieved from your IBM
Cloud account.
API_KEY = "zWW8uDkQyOqWvJ5xmqqNAX325IF5qYdACzS8KeyX6Gk1"
token_response = requests.post('https://iam.cloud.ibm.com/identity/token', data={"apikey":
    API_KEY, "grant_type": 'urn:ibm:params:oauth:grant-type:apikey'})
mltoken = token_response.json()["access_token"]
```

```
header = {'Content-Type': 'application/json', 'Authorization': 'Bearer ' + mltoken}
```

```
app = Flask(__name__)
```

```
[00:42, 20/11/2022] Darathi Vcet: #use the model
```

```
model=pickle.load(open('phishing.pkl','rb'))
```

```
@app.route('/dashboard')
```

```
def predict_data():
```

```
    return render_template('dashboard.html')
```

```
@app.route('/form')
```

```
def single_page():
```

```
    return render_template('index.html')
```

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

```
def get_prediction():
```

```
    url = request.form['url']
```

```
    checking=main(url)
```

```
    print(url)
```

```
    # X=np.array(checking.getFeatureList()).reshape(1,30)
```

```
    print(type(url))
```

```
    # NOTE: manually define and pass the array(s) of values to be scored in the next line
```

```
    payload_scoring = {"input_data": [{"field": 'url', "values": checking}]}
```

```
    response_scoring = requests.post('https://us-south.ml.cloud.ibm.com/ml/v4/deployments/a16a9f12-741c-40d9-a9f...
```

```
[00:42, 20/11/2022] Darathi Vcet: if value==-1:
```

```
    val=1
```

```
    txt="This is not a phishing website continue to use"
```

```
if value==1:
```

```
    txt="You are in a phishing site."
```

```
else:
```

```
    val=0
```

```

        txt="This is a suspicious website"
        return render_template("dashboard.html",predicted='{}'.format(txt),url=url)

if __name__ == "__main__":
    app.run(host='0.0.0.0', debug=True)

```

second:app.py

```

from flask import *
import numpy as np
from markupsafe import escape
import pickle
from inputScript import main

app = Flask(__name__)

#use the model
model=pickle.load(open('phishing.pkl','rb'))

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

@app.route('/form')
def single_page():
    return render_template('index.html')

@app.route('/predictdata', methods=["GET","POST"])
def get_prediction():
    url = request.form['url']
    print(url)
    print(type(url))
    checking=main(url)
    predicted_value=model.predict(checking)
    value=predicted_value[0]

```

```

    val=0
    if value==1:
        val=1
        txt="This is not a phishing website continue to use"
    else:
        val=0
        txt="This is a phishing website"
    return render_template("dashboard.html",predicted='{}'.format(txt),url=url)

if __name__ == "__main__":
    app.run(host='0.0.0.0', debug=True)

:feature extraction of URL

import regex
from tldextract import extract
import ssl
import socket
from bs4 import BeautifulSoup
import urllib.request
import whois
import datetime
import re
import requests
import validators

#1
def url_having_ip(url):
    return 0

#2
def chec_ip_regularity(url):
    return 0

```

#3

#URL based Features

#length of the URL

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

#4

#shortening service

```
def url_sort(url):  
    return 0
```

#5

```
def check_symbol_specification(url):
```

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

#6

```
def having_double_slash(url):
```



```
return 0
```

```
#7
```

```
#domain based Feature
```

```
#count for domain
```

```
def check_prefix_suffix(url):
```

```
    print(type(url))
```

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

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

```
        return 1
```

```
    else:
```

```
        return -1
```

```
#8
```

```
#URL based Feature
```

```
#number of sub-domain in URL
```

```
def check_subdomain(url):
```

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

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

```
        return -1
```

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

```
        return 0
```

```
    else:
```

```
        return 1
```

```
#9
```

```
#check the ssl certificate final state to avoid phishing
```

```
def check_SSL_finalstate(url):
```

```

try:
    if regex.search('^https',url):
        used=1
    else:
        used=0
    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','Veriz
on','Trustwave','Unizeto','Buypass','QuoVadis','Deutsche Telekom','Network
Solutions','SwissSign','IdenTrust','Secom','TWCA','Geotrust','Thawte','Doster','VeriSign']

    startingdate=str(certificate['noBefore'])
    endingdate=str(certificate['noAfter'])
    startingYear=int(startingdate.split()[3])
    endingYear=int(endingdate.split()[3])
    Age_of_certificate=endingYear-startingYear
    if (used==1) and (certificate_Auth in trusted_Auth) and (Age_of_certificate>=1):
        return 1

```

```
        elif (used==1) and (certificate_Auth not in trusted_Auth):  
            return 0  
        else:  
            return 1  
    except Exception as e:  
        return 1
```

#10

#Domain Based Feature

#How many days passed since the domain was registered?

```
def check_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
```

#11

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

#12

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

```
return 0
```

```
#13
```

```
#check for https token in the url
```

```
def check_https_token(url):
```

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

```
    host = subdomain + "." + domain + suffix
```

```
    if host.count('https'):
```

```
        return 1
```

```
    else:
```

```
        return -1
```

```
#14
```

```
def check_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 = 0
```

```
        avg = 0
```

```
        for image in imgs:
```

```
            subdomain, domain, suffix = extract(image['src'])
```

```
            imageDomain = domain
```

```
            if websiteDomain == imageDomain or imageDomain == ":
```

```

        linked=linked+1
    vids=soup.findAll('vedio',src=True)
    total=total+len(vids)
    for video in vids:
        subdomain, domain, suffix = extract(video['src'])
        vidDomain=domain
        if websiteDomain == vidDomain or vidDomain == "":
            linked = linked + 1
    linked2=total-linked
    if total!=0:
        ag=linked2/total
        if avg<0.22:
            return -1
        elif(0.22<=avg<=0.61):
            return 0
        else:
            return 1
    except:
        return 0

```

#15

#shows the achor linking of that page

```

def chec_anchor(url):
    try:
        subdomain,domain,suffix=extract(url)
        websiteDomain=domain

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

```

```

soup=BeautifulSoup(opener,'lxml')
anchors=soup.findAll('a',href=True)
total=len(anchors)
linked=0
avg=0
for anchor in anchors:
    subdomain,domain,suffix=extract(anchors['href'])
    anchorDomain=domain
    if websiteDomain==anchorDomain or anchorDomain=="":
        linked=linked+1
    linked2=total-linked
    if total!=0:
        avg=linked2/total
    if avg<0.31:
        return -1
    elif 0.31<=avg<=0.67:
        return 0
    else:
        return 1
except:
    return 0

```

#16

#checking for links an tags

def check_tags_links(url):

try:

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

soup=BeautifulSoup(opener,'lxml')

```

meta=0
link=0
script=0
anchors=0
avg=0
for meta in soup.find_all('meta'):
    meta=meta+1
for link in soup.find_all('link'):
    link=link+1
for script in soup.find_all('script'):
    script=script+1
for anchor in soup.find_all('a'):
    anchors = anchors + 1
total=meta+link+script+anchors
tags=meta+link+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 1

```

#17

```
def sfh(url):
```

```
    return 0
```

#18

```
def check_email_submit(url):
```

```
    try:
```

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

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

```
        if soup.find('mailyo'):
```

```
            return 1
```

```
        else:
```

```
            return -1
```

```
    except:
```

```
        return 0
```

#19

```
def abnormal_url(url):
```

```
    if validators.url(url)==True:
```

```
        return 1
```

```
    else:
```

```
        return -1
```

#20

```
#redirect
```

```
def redirect(url):
```

```
    responses = requests.get(url)
```

```
    if responses.history==" ":
```



```
        return 1
    else:
        return -1

#21
def on_mouse_over(url):
    return 0

#22
def right_click(url):
    return 0

#23
def pop_up(url):
    return 0

#24
def iframe(url):
    return 0

#25
def checK_age_domain(url):
    try:
        w=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
```

#26

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

#27

#Page Based Features

#Web traffic share per country

```
def web_traffic(url):
    try:
        r = requests.head(url,verify=False,timeout=5)
        if r.status_code==200:
            return 1
        else:
            return -1
    except:
        return 0
```

#28

```
def page_run(url):
    return 0
```

#29

#page based Feature

```

#checking if the url is indexed by google or not
def google_index(url):
    google = "https://www.google.com/search?q=site:" + url + "&hl=en"
    response = requests.get(google, cookies={"CONSENT": "YES+1"})
    soup = BeautifulSoup(response.content, "html.parser")
    not_indexed = re.compile("did not match any documents")

    if soup(text=not_indexed):
        return -1
    else:
        return 1

#30

def link_pointing(url):
    return 0

#31

def statistical(url):
    return 0

def main(url):
    print(url)
    check=[[url_having_ip(url),
            chec_ip_regularity(url),
            url_sort(url),
            check_symbol_specification(url),
            having_double_slash(url),
            check_prefix_suffix(url),

```

```
check_subdomain(url),
check_SSL_finalstate(url),
check_domain_registration(url),
favicon(url),
port(url),
check_https_token(url),
check_request_url(url),
chec_anchor(url),
check_tags_links(url),
sfh(url),
check_email_submit(url),
abnormal_url(url),
redirect(url),
on_mouse_over(url),
right_click(url),
pop_up(url),
iframe(url),
chechK_age_domain(url),
dns(url),
web_traffic(url),
page_run(url),
google_index(url),
link_pointing(url),
statistical(url)
]]
return check
```

```
# ,check_symbol_specification(url)
```

Github and project Demo Link

Github link: <https://github.com/IBM-EPBL/IBM-Project-22797-1659858444>

project demo Link:

