NALAIYA THIRAN - IBM PROJECT REPORT

(19EC406T - Professional Readiness for Innovation, Employability and Entrepreneurship)

ON

WEB PHISHING DETECTION

Submitted by

TEAM ID: PNT2022TMID23445

DURGA DEVI M	**(113219041026)**
SINDHU J	**(113219041009)**
SUBHIKSHA P	**(113219041117)**
SURTHIKA G	**(113219041119)**

in partial fulfillment for the award of the degree of

BACHELOR OF ENGINEERING

IN

ELECTRONICS AND COMMUNICATION ENGINEERING

VELAMMAL ENGINEERING COLLEGE, CHENNAI-66.

(An Autonomous Institution, Affiliated to Anna University, Chennai)

2022-2023

VELAMMAL ENGINEERING COLLEGE
CHENNAI -66

(An Autonomous Institution, Affiliated to Anna University, Chennai)

BONAFIDE CERTIFICATE

Certified that this NALAIYA THIRAN – IBM PROJECT REPORT **"WEB PHISHING DETECTION"** is the Bonafide work of "DURGA DEVI M (113219041026), SINDHU J (11321901009), SUBHIKSHA P (113219041117), and SURTHIKA G (113219041119)" carried out in "PROFESSIONAL READINESS FOR INNOVATION, EMPLOYABILITY AND ENTREPRENEURSHIP (NALAIYA THIRAN-IBM PROJECT)**"** during the Academic Year 2022-2023.

FACULTY EVALUATOR	**HEAD OF THE DEPARTMENT**
Mrs. DOLLY IRENE	**Dr. S. MARY JOANS**
ASSISTANT PROFESSOR I	**PROFESSOR & HEAD**
Department of Electronics and Communication Engineering Velammal Engineering College Chennai-600 066	Department of Electronics and Communication Engineering Velammal Engineering College Chennai-600 066

TABLE OF CONTENT

CHAPTER 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:

- Web phishing aims to steal private information, such as usernames, passwords, and credit card details, by way of impersonating a legitimate entity.
- It will lead to information disclosure and property damage.
- Large organizations may get trapped in different kinds of scams.
- This Guided Project mainly focuses on applying a machine-learning algorithm to detect Phishing websites.

In order to detect and predict e-banking phishing websites, we proposed an intelligent, flexible and effective system that is based on using classification algorithms. We implemented classification algorithms and techniques to extract the phishing datasets criteria to classify their legitimacy. The e-banking phishing website can be detected based on some important characteristics like URL and domain identity, and security and encryption criteria in the final phishing detection rate. 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 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. We propose a web application for detection. The algorithm used is random forest in order to get better performance and accuracy. This system uses a database in order to store phishing websites which are already tested and can be used as blacklist, which makes the classification even faster, as it reduces repetition.

CHAPTER 2
LITERATURE SURVEY

2.1 EXISTING PROBLEM:

There are many users who purchase products through online platform and the payment is done through e-banking. There are some fake banking websites in which they collect the more sensitive information like username, password, credit card details etc , for illegal purpose. This type of websites are called phishing website. Here web phishing is one of the security threat to webservices on the internet. we people are highly dependent on the internet. For performing online shopping and online activities like banking, mobile recharge and more activities are done only through internet. Here phishing is nothing but a type of website threat which illegally collects the original website information such as login id, password and credit card information.

2.2 PROBLEM STATEMENT DEFINITION:

An online user needed to purchase something through an online. So he entered into the online website through internet. It takes some time to display the product. He started to see all the products. He search the necessary things in online website. At last he found the needed products. After that he entered all the credit card details, username and password for purchasing the things through online. Then he received the message "Your order is placed and transaction is successfully completed. You will receive the ordered product within 2 days". After that within 24 hours he got a message in mobile and the bank account was empty then the customer shocked . Then only he realized that was a fake website and his bank account details was stolen by hacker.

Web phishing aims to steal private information, such as usernames, passwords, and credit card details, by way of impersonating a legitimate entity. It will lead to information disclosure and property damage. This paper mainly focuses on applying a deep learning framework to detect phishing websites.

PROBLEMS:

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 attack examples

A spoofed email ostensibly from myuniversity.edu is massdistributed to as many faculty members as possible.

The email claims that the user’s password is about to expire. Instructions are given to go to myuniversity.edu/renewal to renew their password within 24 hours.

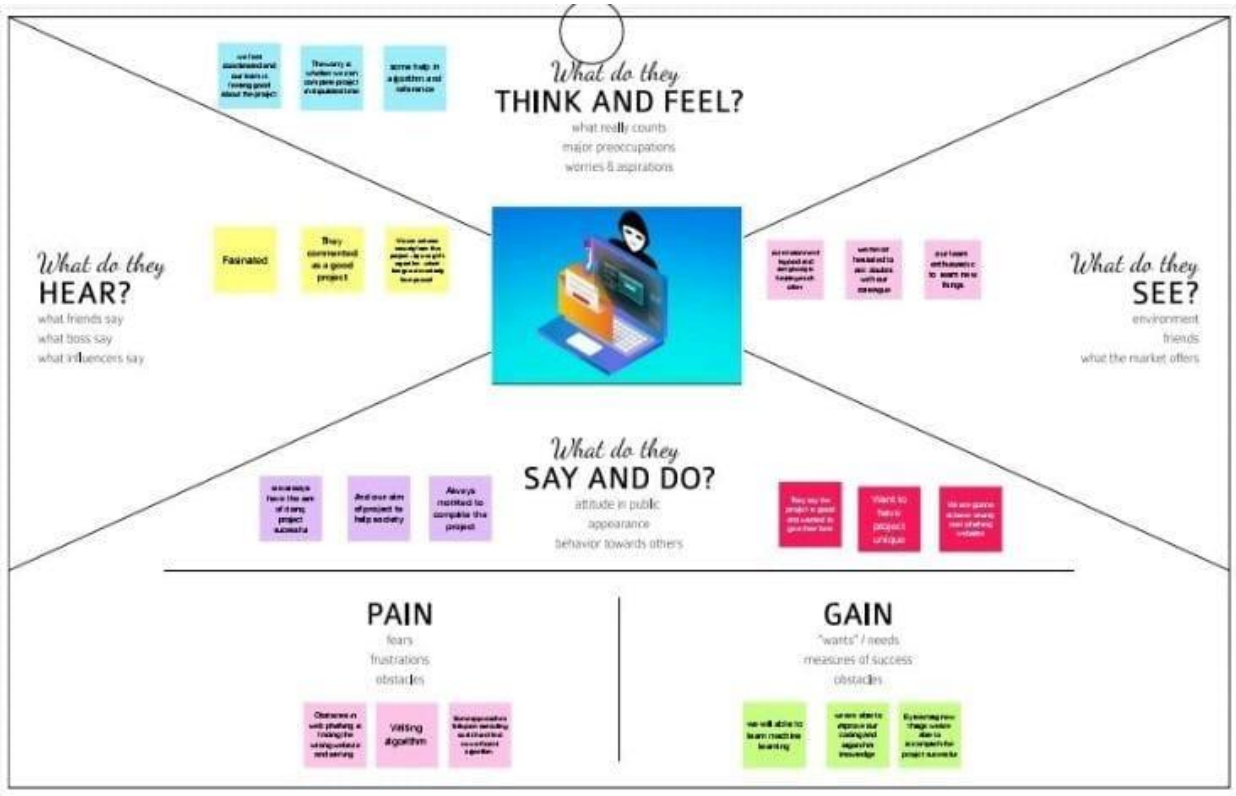
Email phishing scams:

Email phishing is a numbers game. An attacker sending out thousands of fraudulent messages can net significant information and sums of money, even if only a small percentage of recipients fall for the scam. As seen above, there are some techniques attackers use to increase their success rates.For one, they will go to great lengths in designing phishing messages to mimic actual emails from a spoofed organization. Using the same phrasing, typefaces, logos, and signatures makes the messages appear legitimate.In addition, attackers will usually try to push users into action by creating a sense of urgency. For example, as previously shown, an email could threaten account expiration and place the recipient on a timer. Applying such pressure causes the user to be less diligent and more prone to error.Lastly, links inside messages resemble their legitimate counterparts, but typically have a misspelled domain name or extra subdomains.In the above example,the myuniversity.edu/renewal URL was changed to myuniversity.edurenewal.com. To overcome this we came with the solution of web phishing detection.

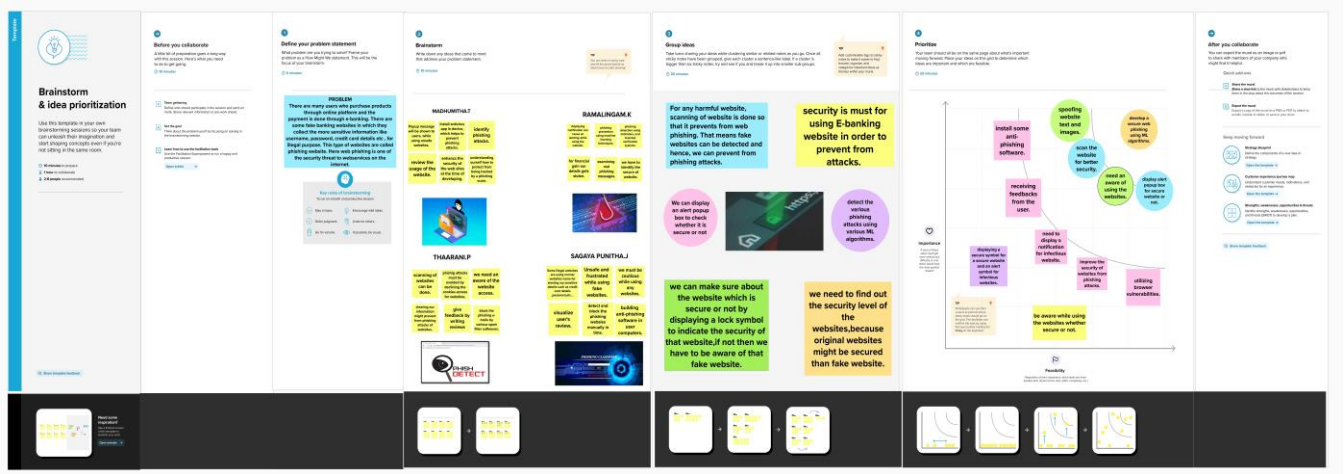
CHAPTER 3
IDEATION & PROPOSED SOLUTION

3.1 EMPATHY MAP CANVAS:

An empathy map is a simple, easy-to-digest visual that captures knowledge about a user's behaviours and attitudes. It is a useful tool to helps teams better understand their users. Creating an effective solution requires understanding the true problem and the person who is experiencing it. The exercise of creating the map helps participants consider things from the user's perspective along with his or her goals and challenges.



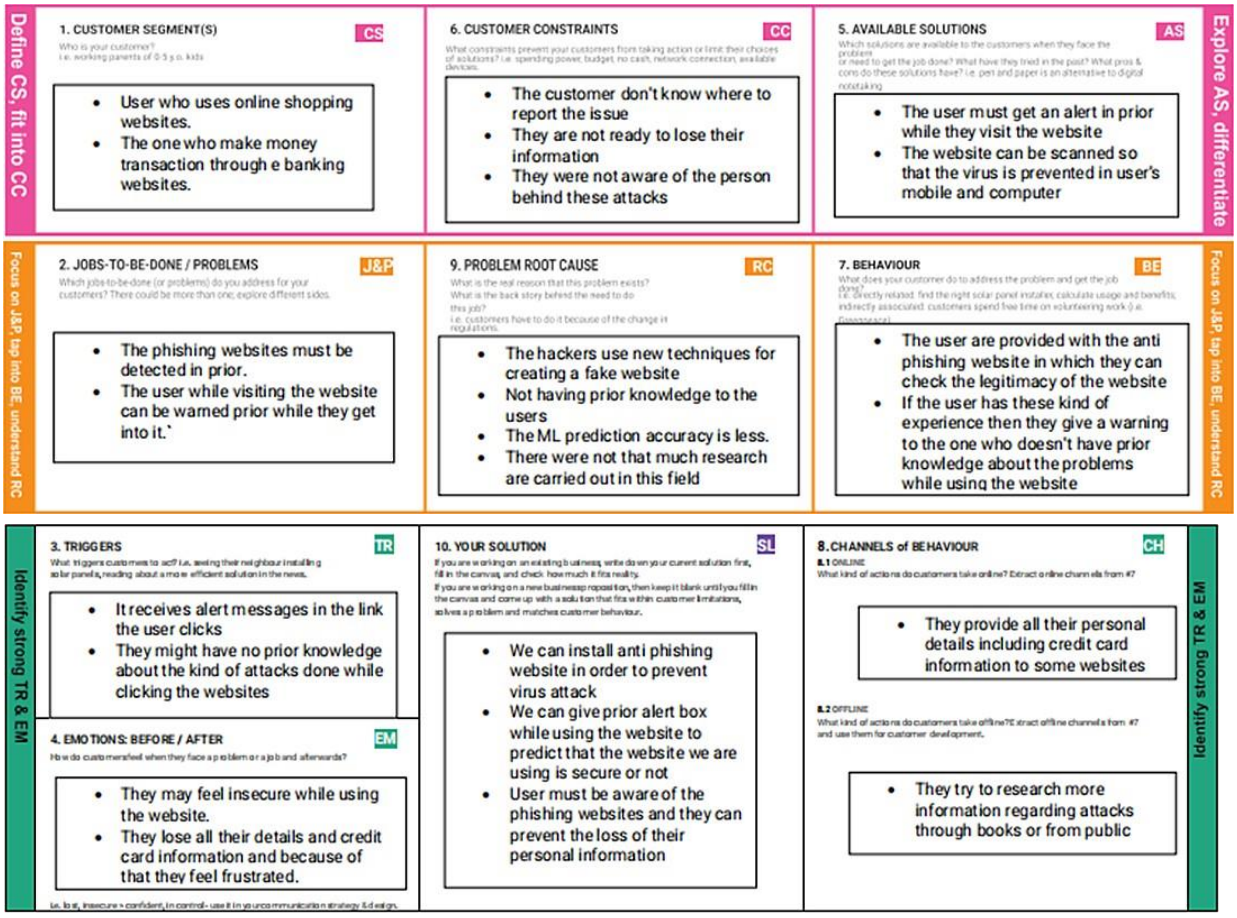
3.2 IDEATION & BRAINSTROMING:



3.3 PROPOSED SOLUTION:

To overcome the problem of phishing website whenever we are clicking on one website it must show an alert box like it is a secure website or it is not a secure website Then another way is that we can scan the website in order to prevent our system or mobile from the phishing attack. Even though technologies are there we as the user have to be aware of the websites whether it is secure or not. We should not click any unwanted websites. The proposed approach has divided the hyperlink specific features into 12 different categories and used these features to train the machine learning algorithms. We have evaluated the performance of our proposed phishing detection approach on various classification algorithms using the phishing and non-phishing websites dataset.As we are using some websites but while clicking that website it display an alert box which leads to an aware of the customer which results in satisfaction of the user while using the websites, And another way is that we can scan the website in order to prevent the hacking of the information which makes even more satisfaction to the customer.

3.4 PROBLEM SOLUTION FIT:



CHAPTER 4
REQUIREMENT ANALYSIS

4.1 FUNCTIONAL REQUIREMENTS:

FR No.	**Functional Requirement(Epic)**	**Sub Requirement (Story/sub-Task)**
FR-1	User Input	User inputs an URL in required field to check its validation.
FR-2	Website Comparison	Model compares the websites using Blacklist and Whitelist approach
FR-3	Feature Extraction	After comparing, if none found on comparison then it extracts feature using heuristic and visual similarity approach.
FR-4	Prediction	Model predicts the URL using Machine Learning algorithms such as Logistic Regression.
FR-5	Classifier	Model sends all output to classifier and produces final result.
		Model then displays

FR-6	Announcement	whether website is a legal site or a phishing site.
FR-7	Events	This model needs the capability of retrieving and displaying accurate result for a website

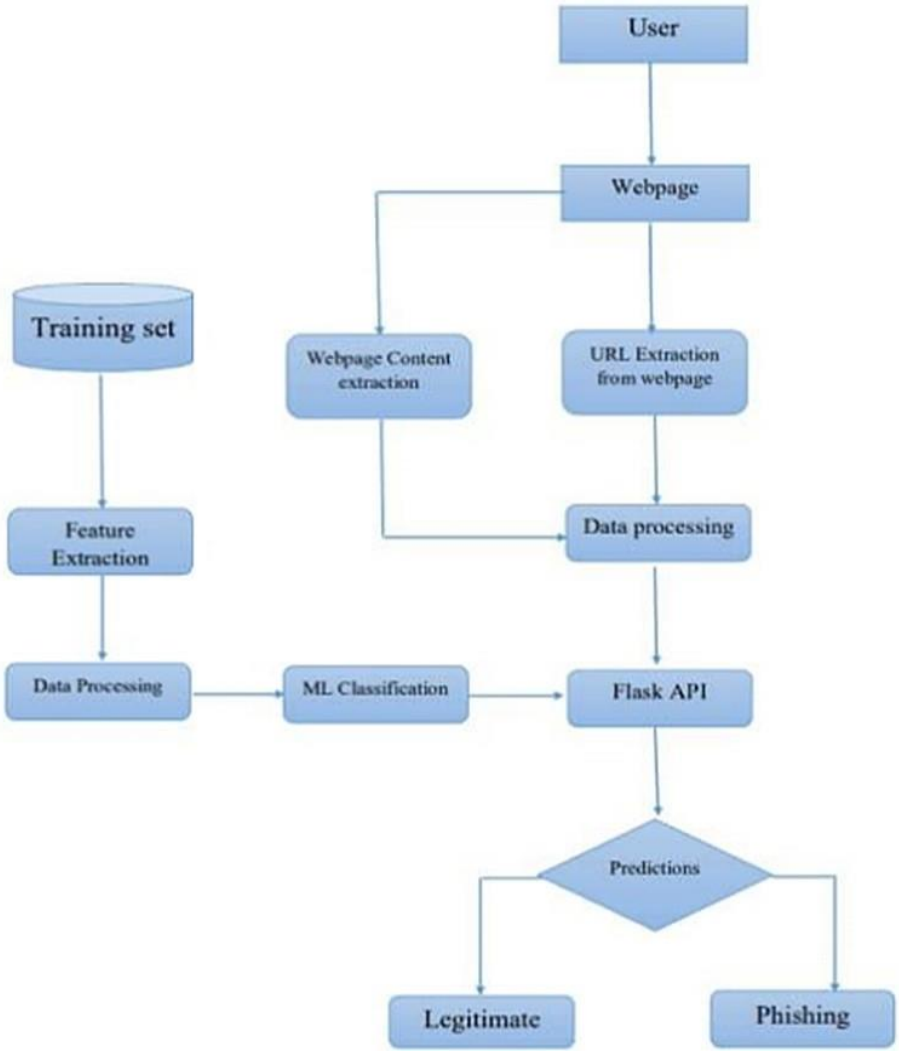
4.2 NON-FUNCTIONAL REQUIREMENTS:

FR No.	**Non-Functional Requirement**	**Description**
NFR-1	**Usability**	User can have full access to the particular websites they using must proceed some certain user friendly websites so that it does not affect the datas.
NFR-2	**Security**	To check whether the particular website is secure or not we can notify it by displaying an alert box while using the websites.
		It must be a reliable

NFR-3	**Reliability**	source to the users while they using the websites.
NFR-4	**Performance**	The performance must be good while using the websites which the users proceeds the website.
NFR-5	**Availability**	The website availability must be valid for the users to access the resources.
NFR-6	**Scalability**	It must be able to handle an increase in users and loads without disrupting the end users.

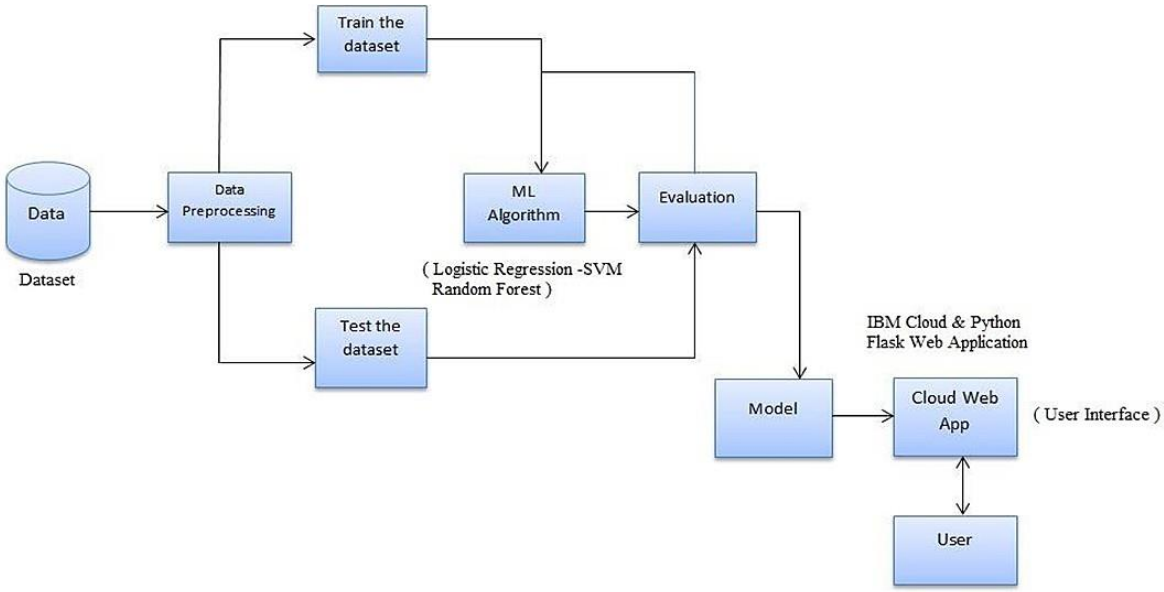
CHAPTER 5
PROJECT DESIGN

5.1 DATA FLOW DIAGRAMS:

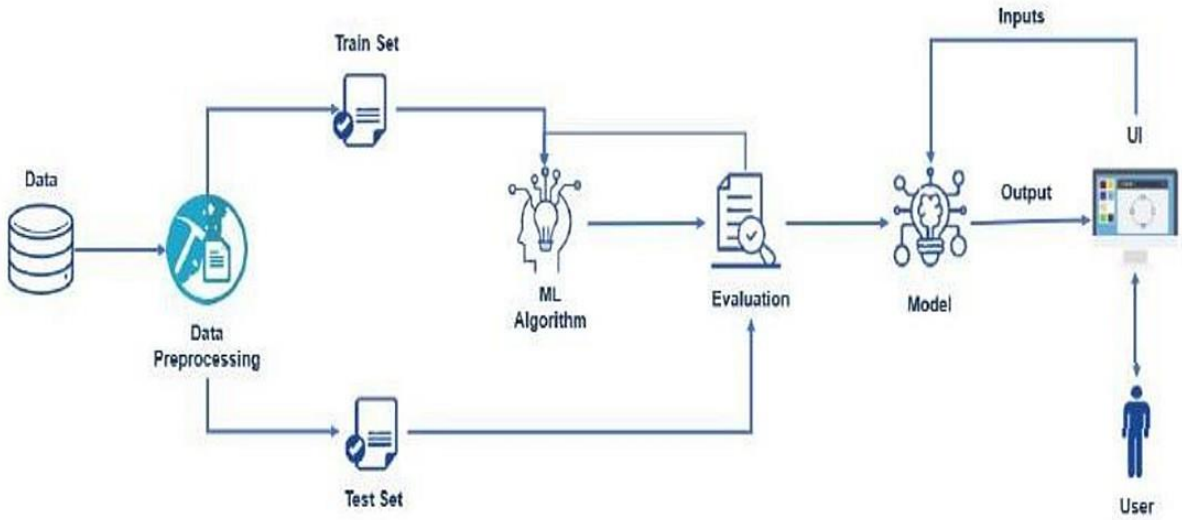


5.2 SOLUTION &TECHNICAL ARCHITECTURE:

SOLUTION ARCHITECTURE:



TECHNICAL ARCHITECTURE:



5.3 USER STORIES:

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

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

CHAPTER 6
PROJECT PLANNING &SCHEDULING

6.1 SPRINT PLANNING & ESTIMATION:

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-1	User input	USN-1	User inputs an URL in the required field to check its validation.	1	Medium	Sagaya Punitha J
Sprint-1	Website Comparison	USN-2	Model compares the websites using Blacklist and Whitelist approach.	1	High	Madhumitha T
Sprint-2	Feature Extraction	USN-3	After comparison, if none found on comparison then it extract feature using heuristic and visual similarity.	2	High	Thaarani P
Sprint-2	Prediction	USN-4	Model predicts the URL using Machine learning algorithms such as logistic Regression.	1	Medium	Ramalingam K

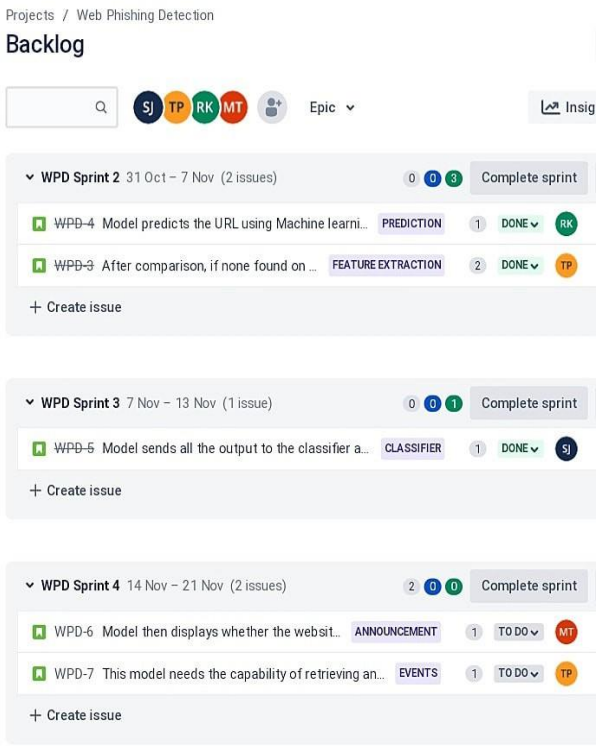
Sprint-3	Classifier	USN-5	Model sends all the output to the classifier and produces the final result.	1	Medium	Sagaya Punitha J
Sprint-4	Announcement	USN-6	Model then displays whether the website is legal site or a phishing site.	1	High	Madhumitha T
Sprint-4	Events	USN-7	This model needsthe capability of retrieving and displaying accurate result for a website.	1	High	Thaarani P

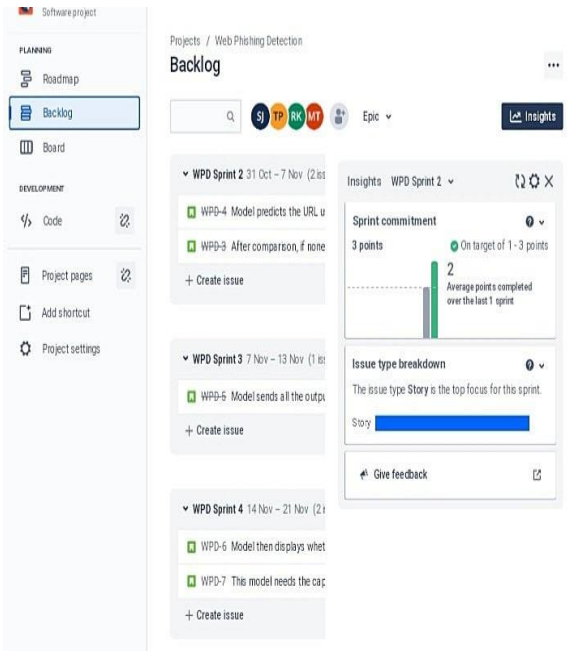
6.2 SPRINT DELIVERY SCHEDULE:

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

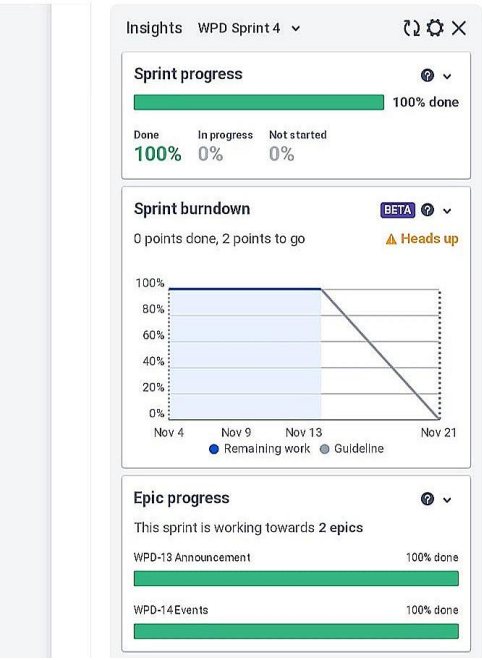
6.3 REPORTS FROM JIRA:

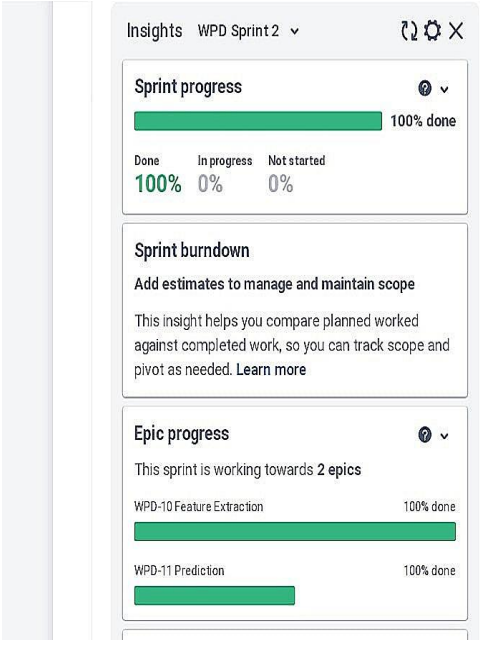
1.BACKLOG:(BACKLOG-1 & BACKLOG-2)



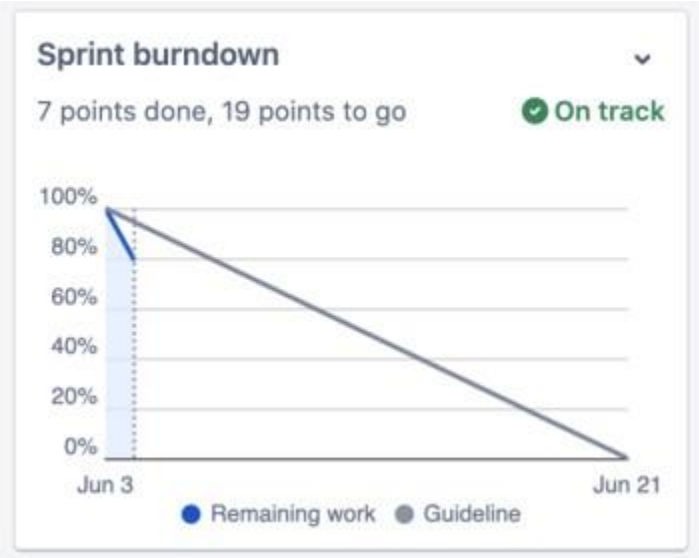


2. SPRINT PROGRESS(FINAL PROGRESS & INSIGHTS):

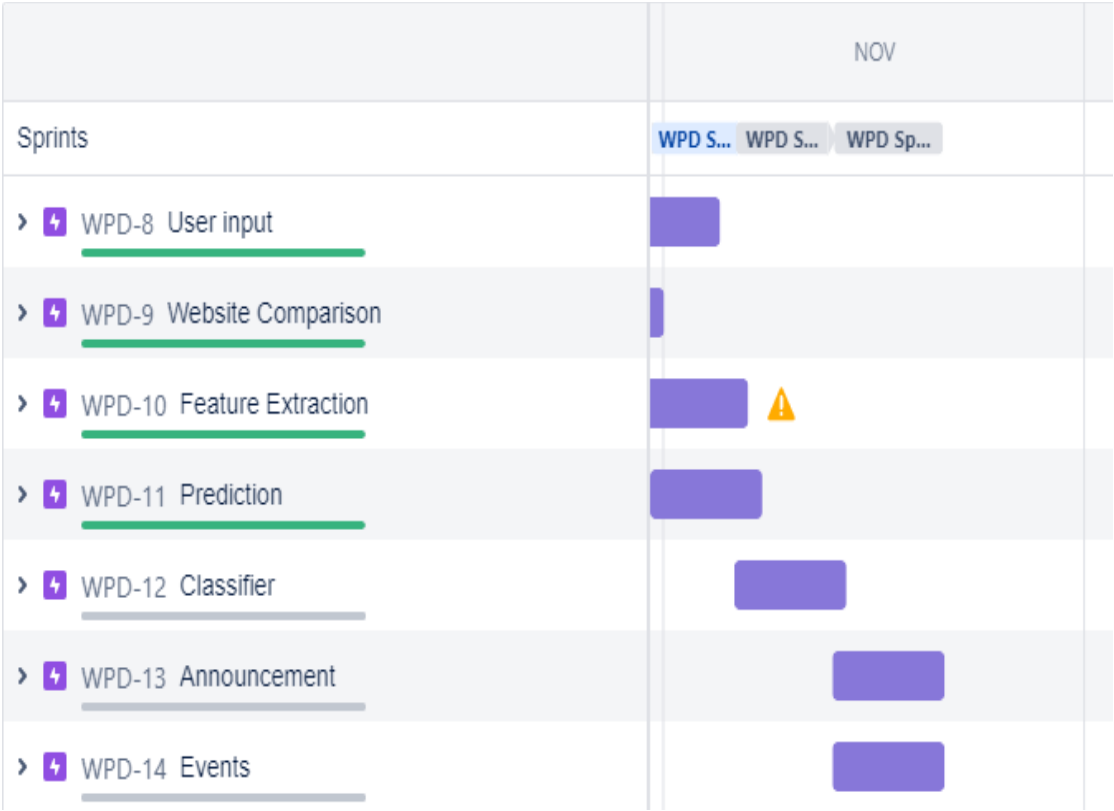




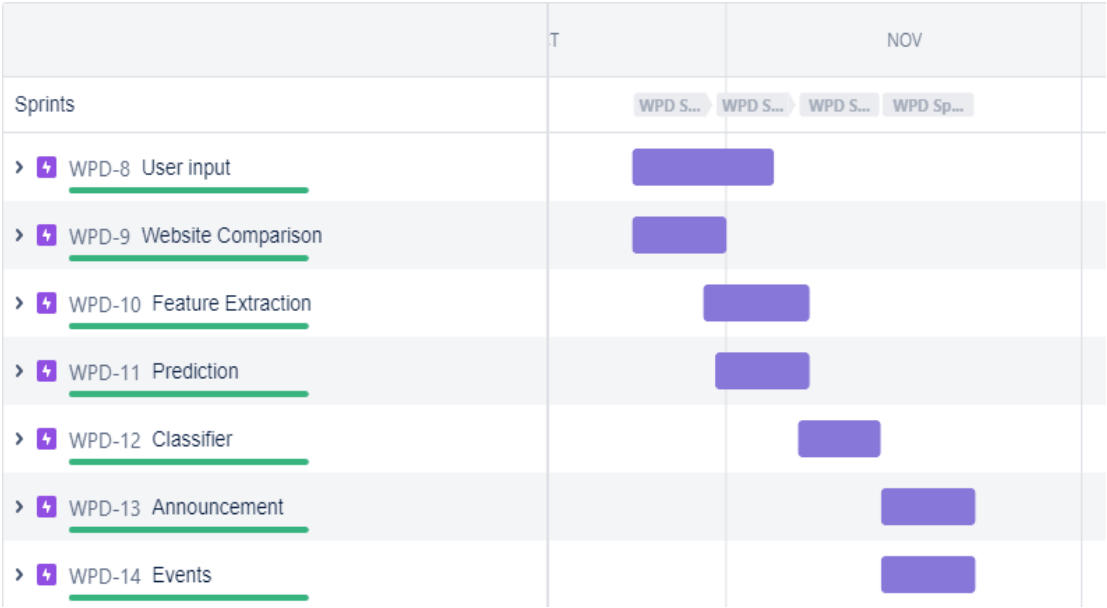
3. SPRINT BURNDOWN:



4. SPRINT 1 &2:


NOV
Sprints
WPD S...
WPD S...
WPD Sp...
WPD-8 User input
WPD-9 Website Comparison
WPD-10 Feature Extraction
WPD-11 Prediction
WPD-12 Classifier
WPD-13 Announcement
WPD-14 Events

5. SPRINT 4&5:


T
NOV
Sprints
WPD S...
WPD S...
WPD S...
WPD Sp...
WPD-8 User input
WPD-9 Website Comparison
WPD-10 Feature Extraction
WPD-11 Prediction
WPD-12 Classifier
WPD-13 Announcement
WPD-14 Events

CHAPTER 7
CODING & SOLUTIONING

7.1 FEATURE 1:

As a user before using any website they bringing that website into our project URL and typing it . As the user types,it results whether it is a safe website or unsafe website in that porject URL so that before they entering into that website and giving thier personal information they may get aware of usage of that particular website to prevent the fake websites to be used by the user .By entering the website it predicts the safe or unsafe website.

```
In [21]: #import dataset
         ds=pd.read_csv("dataset_website .csv")
         ds.head()
```

Out[21]:

	index	having_IPhaving_IP_Address	URLURL_Length	Shortining_Service	having_At_Symbol	double_slash_redirecting	Prefix_Suffix	having_Sub_Domain	SSL
0	1	-1	1	1	1	-1	-1	-1	
1	2	1	1	1	1	1	-1	0	
2	3	1	0	1	1	1	-1	-1	
3	4	1	0	1	1	1	-1	-1	
4	5	1	0	-1	1	1	-1	1	

5 rows × 32 columns

```
In [22]: ds.tail()
```

Out[22]:

	index	having_IPhaving_IP_Address	URLURL_Length	Shortining_Service	having_At_Symbol	double_slash_redirecting	Prefix_Suffix	having_Sub_Domain
11050	11051	1	-1	1	-1	1	1	1
11051	11052	-1	1	1	-1	-1	-1	1
11052	11053	1	-1	1	1	1	-1	1
11053	11054	-1	-1	1	1	1	-1	-1
11054	11055	-1	-1	1	1	1	-1	-1

5 rows × 32 columns

```
In [23]: ds.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 11055 entries, 0 to 11054
Data columns (total 32 columns):
 #   Column                       Non-Null Count  Dtype
---  ------                       --------------  -----
 0   index                        11055 non-null  int64
 1   having_IPhaving_IP_Address   11055 non-null  int64
 2   URLURL_Length                11055 non-null  int64
 3   Shortining_Service           11055 non-null  int64
 4   having_At_Symbol             11055 non-null  int64
 5   double_slash_redirecting     11055 non-null  int64
 6   Prefix_Suffix                11055 non-null  int64
 7   having_Sub_Domain            11055 non-null  int64
 8   SSLfinal_State               11055 non-null  int64
 9   Domain_registeration_length  11055 non-null  int64
 10  Favicon                      11055 non-null  int64
 11  port                         11055 non-null  int64
 12  HTTPS_token                  11055 non-null  int64
 13  Request_URL                  11055 non-null  int64
 14  URL_of_Anchor                11055 non-null  int64
 15  Links_in_tags                11055 non-null  int64
 16  SFH                          11055 non-null  int64
 17  Submitting_to_email          11055 non-null  int64
 18  Abnormal_URL                 11055 non-null  int64
 19  Redirect                     11055 non-null  int64
 20  on_mouseover                 11055 non-null  int64
 21  RightClick                   11055 non-null  int64
 22  popupwidnow                  11055 non-null  int64
 23  Iframe                       11055 non-null  int64
 24  age_of_domain                11055 non-null  int64
 25  DNSRecord                    11055 non-null  int64
 26  web_traffic                  11055 non-null  int64
 27  Page_Rank                    11055 non-null  int64
 28  Google_Index                 11055 non-null  int64
 29  Links_pointing_to_page       11055 non-null  int64
 30  Statistical_report           11055 non-null  int64
 31  Result                       11055 non-null  int64
dtypes: int64(32)
memory usage: 2.7 MB
```

7.2 FEATURE 2:

Here the user enters the URL in our project workspace, it predicts the safe or unsafe website in which the user will get alert of entering into that website as it alerts with a wordings namely **"it is safe website,and it is secure! or it is an unsafe website be cautious!".**By displaying the message user will surely be alert of using the website.

```
In [24]: ds.describe()
```

Out[24]:

	index	having_IPhaving_IP_Address	URLURL_Length	Shortining_Service	having_At_Symbol	double_slash_redirecting	Prefix_Suffix	having_Sub_
count	11055.000000	11055.000000	11055.000000	11055.000000	11055.000000	11055.000000	11055.000000	11055
mean	5528.000000	0.313795	-0.633198	0.738761	0.700588	0.741474	-0.734962	0
std	3191.447947	0.949534	0.766095	0.673998	0.713598	0.671011	0.678139	0
min	1.000000	-1.000000	-1.000000	-1.000000	-1.000000	-1.000000	-1.000000	-1
25%	2764.500000	-1.000000	-1.000000	1.000000	1.000000	1.000000	-1.000000	-1
50%	5528.000000	1.000000	-1.000000	1.000000	1.000000	1.000000	-1.000000	0
75%	8291.500000	1.000000	-1.000000	1.000000	1.000000	1.000000	-1.000000	1
max	11055.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1

8 rows × 32 columns

```
In [25]: ds.shape
Out[25]: (11055, 32)
```

```
In [32]: #no null values
         ds.isnull().any()
Out[32]: index                        False
         having_IPhaving_IP_Address   False
         URLURL_Length                False
         Shortining_Service           False
         having_At_Symbol             False
         double_slash_redirecting     False
         Prefix_Suffix                False
         having_Sub_Domain            False
         SSLfinal_State               False
         Domain_registeration_length  False
         Favicon                      False
         port                         False
         HTTPS_token                  False
         Request_URL                  False
         URL_of_Anchor                False
         Links_in_tags                False
         SFH                          False
         Submitting_to_email          False
         Abnormal_URL                 False
         Redirect                     False
         on_mouseover                 False
         RightClick                   False
         popUpWidnow                  False
         Iframe                       False
         age_of_domain                False
         DNSRecord                    False
         web_traffic                  False
         Page_Rank                    False
         Google_Index                 False
         Links_pointing_to_page       False
         Statistical_report           False
         Result                       False
         dtype: bool
```

```
In [33]: #split data as indep(x-all cols) and dep(y-Resut col)
         #remove index col in indep ds(32 cols)
         x = ds.iloc[:,1:31].values
         y = ds.iloc[:,-1].values
         print(x,y)

         [[-1  1  1 ...  1  1 -1]
          [ 1  1  1 ...  1  1  1]
          [ 1  0  1 ...  1  0 -1]
          ...
          [ 1 -1  1 ...  1  0  1]
          [-1 -1  1 ...  1  1  1]
          [-1 -1  1 ... -1  1 -1]] [-1 -1 -1 ... -1 -1 -1]
```

Check for Categorical columns and perform encoding.

```
In [47]: #manually handling the categorical columns

         ds['Google_Index'].replace( {'1:0'},inplace=True)

         ds.head()
```

Out[47]:

	Index	having_IPhaving_IP_Address	URLURL_Length	Shortining_Service	having_At_Symbol	double_slash_redirecting	Prefix_Suffix	having_Sub_Domain	SS
0	1	-1	1	1	1	-1	-1	-1	
1	2	1	1	1	1	1	-1	0	
2	3	1	0	1	1	1	-1	-1	
3	4	1	0	1	1	1	-1	-1	
4	5	1	0	-1	1	1	-1	1	

5 rows × 32 columns

Split the data into dependent and independent variables.

In [48]:
```
#target variable

y=ds['Google_Index']
y.head()
```

Out[48]:
```
0    1
1    1
2    1
3    1
4    1
Name: Google_Index, dtype: int64
```

In [49]:
```
#independent

x=ds.drop(columns=['Google_Index'],axis=1)

x.head()
```

Out[49]:

	index	having_IPhaving_IP_Address	URLURL_Length	Shortining_Service	having_At_Symbol	double_slash_redirecting	Prefix_Suffix	having_Sub_Domain	$S
0	1	-1	1	1	1	-1	-1	-1	
1	2	1	1	1	1	1	-1	0	
2	3	1	0	1	1	1	-1	-1	
3	4	1	0	1	1	1	-1	-1	
4	5	1	0	-1	1	1	-1	1	

5 rows × 31 columns

In [55]:
```
x.mean()
```

Out[55]:
```
-4.976007253539460e-18
```

In [56]:
```
x.std()
```

Out[56]:
```
0.9999999999999998
```

splitting data into train and test sets

In [57]:
```
#splitting data into train and test
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_state=0)
```

In [58]:
```
x_train.shape
```

Out[58]:
```
(8844, 31)
```

In [59]:
```
x_test.shape
```

Out[59]:
```
(2211, 31)
```

In [60]:
```
y_train.shape
```

Out[60]:
```
(8844,)
```

In [61]:
```
y_test.shape
```

Out[61]:
```
(2211,)
```

Saving the model

In [75]:
```
import pickle
pickle.dump(lr,open('Phishing_Website.pkl','wb'))
```

In [ ]:

CHAPTER-8
TESTING

8.1 TEST CASES:

Test case ID	Feature Type	Component	Test Scenario	Pre-Requisite	Steps To Execute	Test Data	Expected Result	Actual Result	Status	Comments	TC for Automation(Y/N	BUG ID	Executed By
LoginPage_TC_OO1	Functional	Home Page	Verify user is able to see the home page, when user enter the link in URL	1.Internet connection 2. Web browser such as Google 3. User know the link http://127.0.0.1:5000 4.Mobile ,Laptop, or System... needed	1.Enter the URL in web browser and click go 2.Verify home page displayed or not	http://127.0.0.1:5000	Home page should display	Working as expected	Pass		Yes		manual
LoginPage_TC_OO2	UI	Home Page	Verify the UI elements in home page	1. Internet connection 2. Web browser such as Google 3. User know the link http://127.0.0.1:5000 4.Mobile ,Laptop, or System... needed	1.Enter URL and click go 2.Verify home page display below UI elements: a.Home b.About c.Contact d.Get started	http://127.0.0.1:5000	Application should show below UI elements: a.Home b.About c.Contact d.Get started	Working as expected	pass		Yes		manual
LoginPage_TC_OO3	Functional	Home page	Verify user is redirected to the about page, when the user click the "About " button	1. Internet connection 2. Web browser such as Google 3. User know the link http://127.0.0.1:5000 4.Mobile ,Laptop, or System... needed	1.Enter the URL (http://127.0.0.1:5000) 2.Click the About button 3.Verify About page displayed or not	http://127.0.0.1:5000	User should navigate to about page	Working as expected	pass		Yes		manual
LoginPage_TC_OO4	Functional	Home page	Verify user is redirected to phishing website detection page when user click the "Get started" button in the home page.	1. Internet connection 2. Web browser such as Google 3. User know the link http://127.0.0.1:5000 4.Mobile ,Laptop, or System... needed	1.Enter the URL (http://127.0.0.1:5000) 2.Click the "Get started " button 3.verify phishing website detection page displayed or not	click the get started button	User should navigate to phishing website detection page	Working as expected	Pass		Yes		manual

				Maximum Marks	4 marks								
Test case ID	Feature Type	Component	Test Scenario	Pre-Requisite	Steps To Execute	Test Data	Expected Result	Actual Result	Status	Comments	TC for Automation(Y/N	BUG ID	Executed By
LoginPage_TC_OO5	Functional	About page	Verify user is redirected to phishing website detection page when user click the "check your website" button in the about page.	1. Internet connection 2. Web browser such as Google 3. User know the link http://127.0.0.1:5000 4.Mobile ,Laptop, or System... needed	1.Enter the URL (http://127.0.0.1:5000) 2.Click the About button 3.Click the "Check your website" button in the About page 4.Verify phishing website	click the "check your website" button	user should navigate to phishing website detection page	Working as expected	Pass	Here user click the "check your website" button in about page	Yes		manual
LoginPage_TC_OO6	Functional	Phishing website detection page	Verify it shows whether the URL entered by the user is safe or unsafe.	https://portal.naanmudhalvan.tn.gov.in/login	1.Enter the URL (http://127.0.0.1:5000) 2.Click the About button 3.Click the "Check your website" button in the About page 4.enter the URL in the Phishing website detection page 5.click the predict button 6.verify it shows whether the URL entered by the user is safe	https://portal.naanmudhalvan.tn.gov.in/login	Application should display "you are safe!! This is a legitimate website"	Working as expected	Pass	user enter the URL in correct format	Yes		Automatic
LoginPage_TC_OO7	Functional	Phishing website detection page	Verify it shows whether the URL entered by the user is safe or unsafe.	https://www.searchonlineinfo.com/	1.Enter the URL (http://127.0.0.1:5000) 2.Click the About button 3.Click the "Check your website" button in the About page 4.enter the URL in the Phishing website detection page 5.click the predict button 6.verify it shows whether the URL entered by the user is safe	https://www.searchonlineinfo.com/	Application should display "you are on the wrong site. Be cautious!"	Working as expected	Pass	User entered the URL in correct format	Yes		Automatic

Test case ID	Feature Type	Component	Test Scenario	Pre-Requisite	Steps To Execute	Test Data	Expected Result	Actual Result	Status	Comments	TC for Automation(Y/N	BUG ID	Executed By
LoginPage_TC_OO8	Functional	Phishing website detection page	Verify it shows whether the URL entered by the user is safe or unsafe.	www.searchonlineinfo.com/	1.Enter the URL (http://127.0.0.1:5000) 2.Click the About button 3.Click the "Check your website" button in the About page 4.enter the URL in the Phishing website detection page 5.click the predict button 6.verify it shows whether the URL entered by the user is safe	www.searchonlineinfo.com/	Application should display "you are on the wrong site. Be cautious!"	Not Working as expecte	Fail	e user enter URL without http	Yes		Automatic
LoginPage_TC_OO9	Functional	Phishing website detection page	Verify it shows whether the URL entered by the user is safe or unsafe.	portal.naanmudhalvan.tn.gov.in/login	1.Enter the URL (http://127.0.0.1:5000) 2.Click the About button 3.Click the "Check your website" button in the About page 4.enter the URL in the Phishing website detection page 5.click the predict button 6.verify it shows whether the URL entered by the user is safe	portal.naanmudhalvan.tn.gov.in/login	Application should display "you are safe!! This is a legitimate website"	Not Working as expecte	Fail	User enter the URL in correct format	Yes		Automatic

8.2 USER ACCEPTANCE TESTING:

Here we briefly explained the test coverage and open issues of the Web Phishing Detection project at the time of the release to User Acceptance Testing (UAT).

8.2.1 DEFECT ANALYSIS:

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

Resolution	Severity 1	Severity 2	Severity 3	Severity 4	Subtotal
By Design	10	4	2	3	20
Duplicate	1	0	3	0	4
External	2	3	0	1	6
Fixed	11	2	4	20	37
Not Reproduced	0	0	1	0	1
Skipped	0	0	1	1	2
Won't Fix	0	5	2	1	8
Totals	24	14	13	26	77

8.2.2 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	7	0	0	7
Client Application	51	0	0	51
Security	2	0	0	2
Outsource Shipping	3	0	0	3
Exception Reporting	9	0	0	9
Final Report Output	4	0	0	4
Version Control	2	0	0	2

CHAPTER 9
RESULTS

9.1 PERFORMANCE METRICS:

In this performance metrics,we have built a decision tree model classifier we predicted the accuracy of the training data and also the testing data.

building the Decision Tree Classifier model

In [63]:
```
# Decision Tree model
from sklearn.tree import DecisionTreeClassifier

# instantiate the model
tree = DecisionTreeClassifier(max_depth = 5)
# fit the model
tree.fit(x_train, y_train)
```

Out[63]:
```
DecisionTreeClassifier(max_depth=5)
```

In [64]:
```
#predicting the target value from the model for the samples
y_test_tree = tree.predict(x_test)
y_train_tree = tree.predict(x_train)
```

Performance Evaluation:

In [66]:
```
#computing the accuracy of the model performance
acc_train_tree = accuracy_score(y_train,y_train_tree)
acc_test_tree = accuracy_score(y_test,y_test_tree)
print("Decision Tree: Accuracy on training Data: {:.3f}".format(acc_train_tree))
print("Decision Tree: Accuracy on test Data: {:.3f}".format(acc_test_tree))
```

```
Decision Tree: Accuracy on training Data: 0.885
Decision Tree: Accuracy on test Data: 0.876
```

CHAPTER 10
ADVANTAGES & DISADVANTAGES

ADVANTAGES:

- Here we can easily predict the website security of the user whlie using the particular website from the web browser.
- By using our project URL the user can easily can get aware of the secure level of the particular website.
- Sometimes the user need not get scare about giving their personal information if they predicted the particular website to be safe or unsafe if it is a safe website they can proceed further or if it's unsafe means they can be even more cautious.

DISADVANTAGES:

- As it depends on internet connectivity,we have to make sure of a proper internet connection to proceed our project URL platform.
- Here we have to enter the original website for identifcation.
- According to this prediction might take a longer due to the availability of the internet services.

CHAPTER 11
CONCLUSION

Our project aims to enhance detection method to detect phishing website using machine learning technology. Also , classifiers generated by machine learning algorithms identify legitimate phishing websites.The proposed technique can detect new temporary phishing sites and reduce the damage caused by phishing attacks. The performance of the proposed technique based on machine learning is more effective that previous phishing detection technologies. In the future, it will be useful to investigate the impact of feature selection using various algorithms.

CHAPTER 12
FUTURE SCOPE

In future in our project we try to inbuilt our URL with google for a safer access for the user to maintain a proper functioning of our project and also we will be using a pop up box for the alert of the user in order to make sure about the security level of the particular website in the user system or mobile so that they can get aware of the usage of the website.

In future we can develop an application for mobile phones and browser extension so it will automatically detect the legitimacy of the websites and warn the user if website is suspicious.

CHAPTER 13
REFERENCES

[1] Higashino, M., Et Al. An Anti-phishing Training System For Security Awareness And Education Considering Prevention Of Information Leakage. In 2019 5th International Conference On Information Management (icim). 2019.

[2] H. Bleau, Global Fraud And Cybercrime Forecast,. 2017.

[3] Michel Lange, V., Et Al., Planning And Production Of Grammatical And Lexical Verbs In Multiword Messages. Plos One, 2017. 12(11): P. E0186685-e018668.

[4] S. Sheng, M. Holbrook, P. Kumaraguru, L. F. Cranor, and J. Downs, "Who falls for phish?: a demographic
analysis of phishing susceptibility and effectiveness of interventions," in Proceedings of the 28th international conference
on Human factors in computing systems, ser. CHI '10. New York, NY, USA: ACM, 2010, pp. 373–382.

[5] B. Krebs, "HBGary Federal hacked by Anonymous," http: //krebsonsecurity.com/2011/02/hbgary-federal-hacked-by-anonymous/, 2011, accessed December 2011.

[6] B. Schneier, "Lockheed Martin hack linked to RSA's SecurID breach," http://www.schneier.com/blog/archives/2011/05/lockheed martin.html, 2011, accessed December 2011.

[7] C. Whittaker, B. Ryner, and M. Nazif, "Large-scale automatic classification of phishing pages," in NDSS '10, 2010.

[8] X. Dong, J. Clark, and J. Jacob, "Modelling user-phishing interaction," in Human System Interactions, 2008 Conference on, may 2008, pp. 627 –632.

CHAPTER 14
APPENDIX

SOURCE CODES :

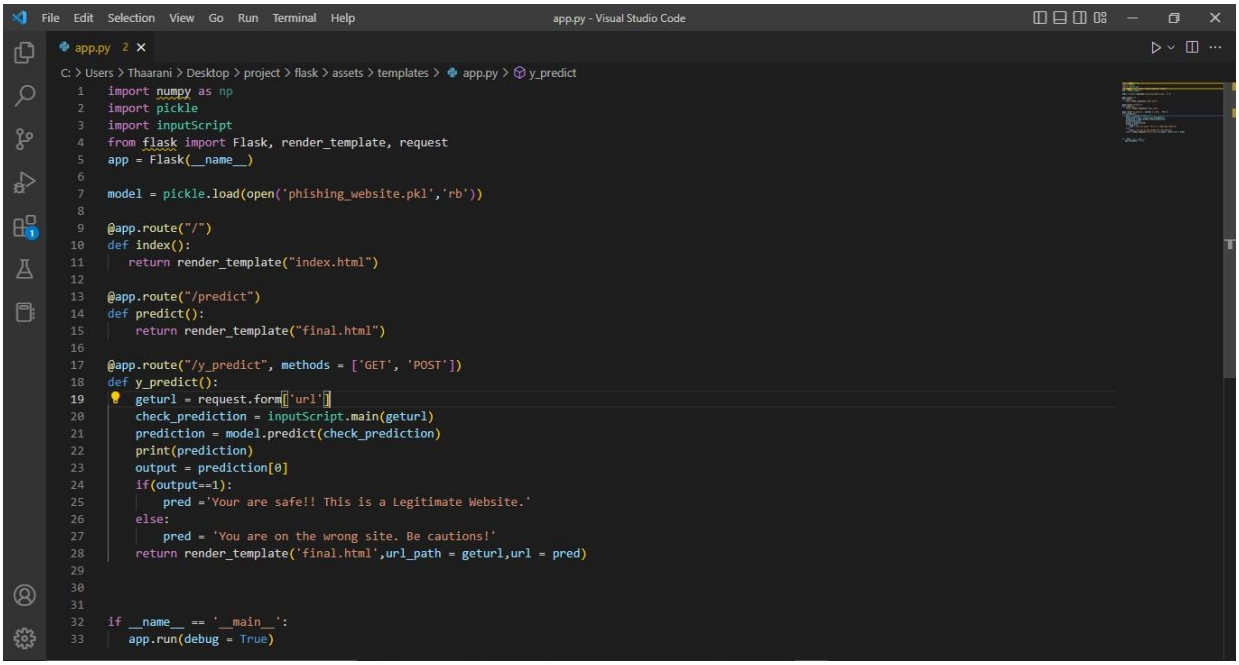
In the application, the user provides any website URL to check and the corresponding parameter values are generated by analysing the URL using which legitimate websites are detected.

BUILDING THE PYTHON FLASK APPLICATION:

In the flask application, the URL is taken from the HTML page and it is scraped to get the different factors or the behavior of the URL. These factors are then given to the model to know if the URL is phishing or safe and is sent back to the HTML page to notify the user.

APP.PY SOURCE CODE:

Here we have done the app.py code inorder to execute the Flask application.

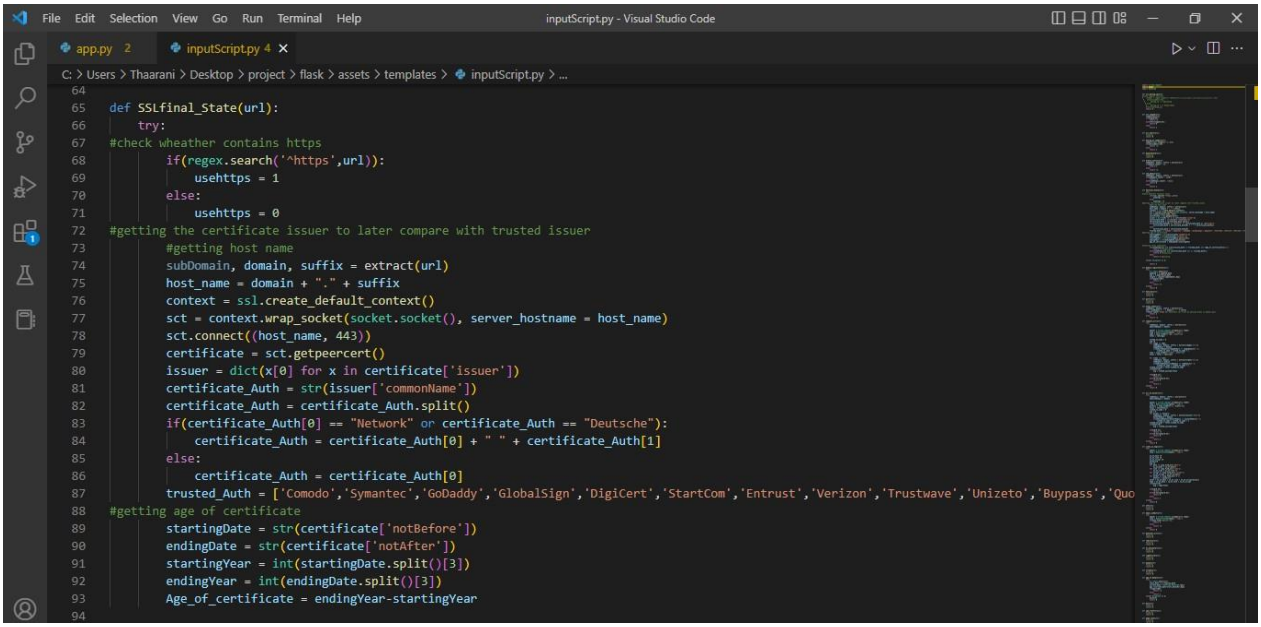



```
import numpy as np
import pickle
import inputScript
from flask import Flask, render_template, request
app = Flask(__name__)

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

@app.route("/")
def index():
    return render_template("index.html")

@app.route("/predict")
def predict():
    return render_template("final.html")

@app.route("/y_predict", methods = ['GET', 'POST'])
def y_predict():
    geturl = request.form['url']
    check_prediction = inputScript.main(geturl)
    prediction = model.predict(check_prediction)
    print(prediction)
    output = prediction[0]
    if(output==1):
        pred ='Your are safe!! This is a Legitimate Website.'
    else:
        pred = 'You are on the wrong site. Be cautions!'
    return render_template('final.html',url_path = geturl,url = pred)


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

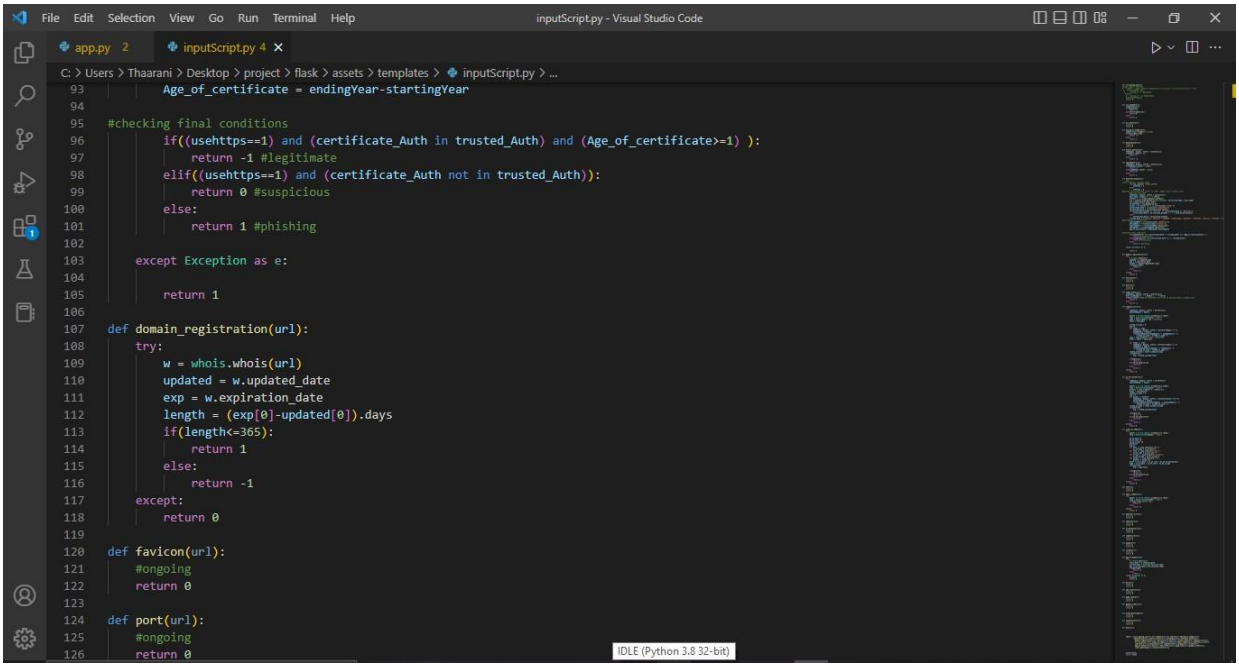
INPUTSCRIPT.PY SOURCE CODE:

After executing the flask application, we have executed the inputscript.py file.

```python
# -*- coding: utf-8 -*-

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


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


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


def url_short(url):
    #ongoing
    return 0

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

def doubleSlash(url):
    #ongoing
    return 0

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

def sub_domain(url):
    subDomain, domain, suffix = extract(url)
    if(subDomain.count('.')==0):
        return -1
    elif(subDomain.count('.')==1):
        return 0
    else:
        return 1
```



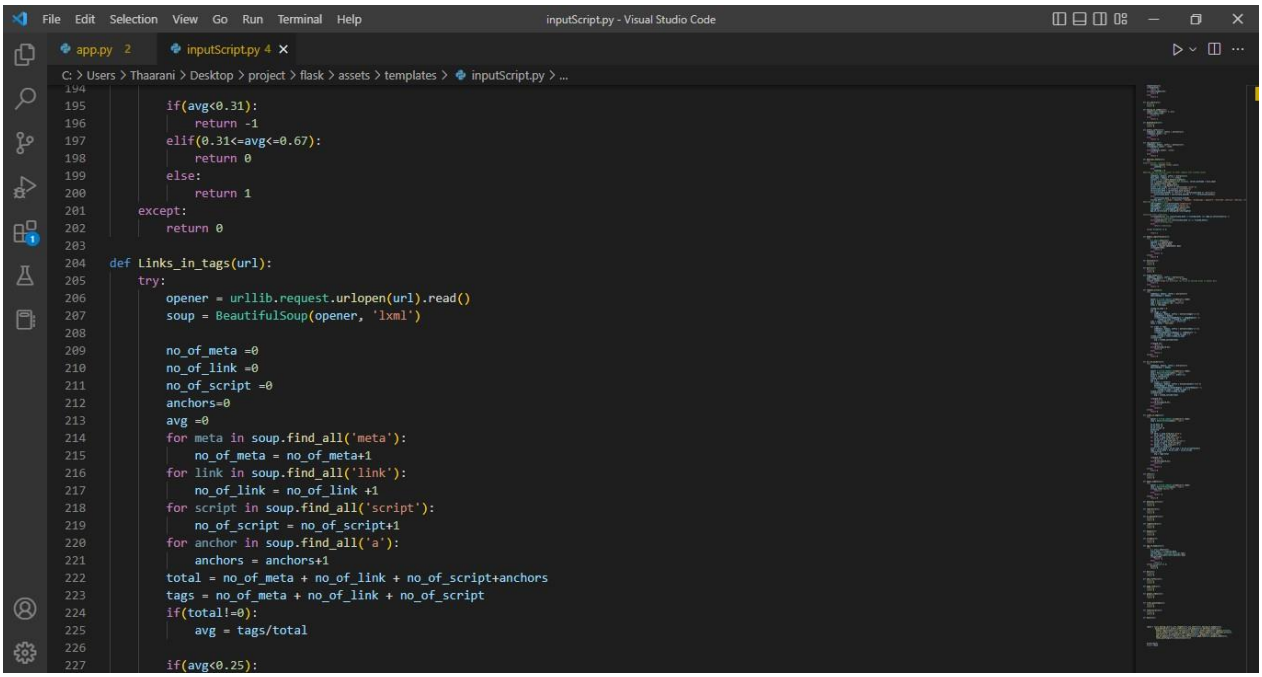

```python
def SSLfinal_State(url):
    try:
#check wheather contains https
        if(regex.search('^https',url)):
            usehttps = 1
        else:
            usehttps = 0
#getting the certificate issuer to later compare with trusted issuer
        #getting host name
        subDomain, domain, suffix = extract(url)
        host_name = domain + "." + suffix
        context = ssl.create_default_context()
        sct = context.wrap_socket(socket.socket(), server_hostname = host_name)
        sct.connect((host_name, 443))
        certificate = sct.getpeercert()
        issuer = dict(x[0] for x in certificate['issuer'])
        certificate_Auth = str(issuer['commonName'])
        certificate_Auth = certificate_Auth.split()
        if(certificate_Auth[0] == "Network" or certificate_Auth == "Deutsche"):
            certificate_Auth = certificate_Auth[0] + " " + certificate_Auth[1]
        else:
            certificate_Auth = certificate_Auth[0]
        trusted_Auth = ['Comodo','Symantec','GoDaddy','GlobalSign','DigiCert','StartCom','Entrust','Verizon','Trustwave','Unizeto','Buypass','Quo
#getting age of certificate
        startingDate = str(certificate['notBefore'])
        endingDate = str(certificate['notAfter'])
        startingYear = int(startingDate.split()[3])
        endingYear = int(endingDate.split()[3])
        Age_of_certificate = endingYear-startingYear
```



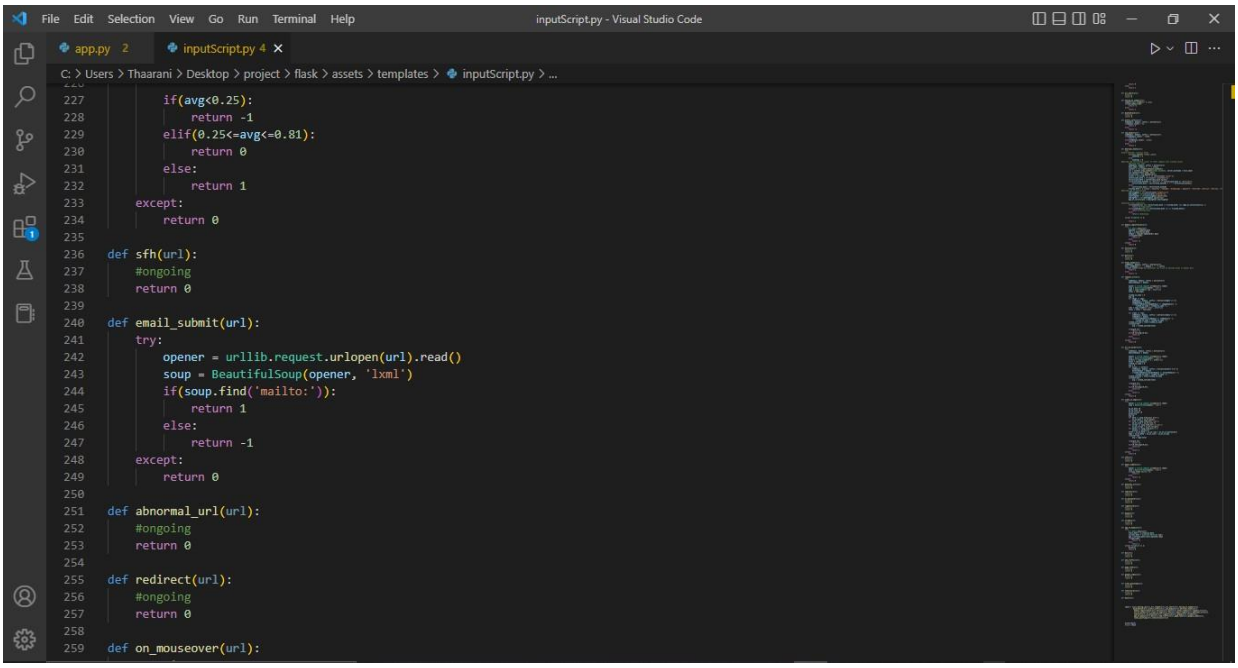

```python
        Age_of_certificate = endingYear-startingYear

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

    except Exception as e:

        return 1

def domain_registration(url):
    try:
        w = whois.whois(url)
        updated = w.updated_date
        exp = w.expiration_date
        length = (exp[0]-updated[0]).days
        if(length<=365):
            return 1
        else:
            return -1
    except:
        return 0

def favicon(url):
    #ongoing
    return 0

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

```python
def https_token(url):
    subDomain, domain, suffix = extract(url)
    host =subDomain +'.' + domain + '.' + suffix
    if(host.count('https')): #attacker can trick by putting https in domain part
        return 1
    else:
        return -1

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

        opener = urllib.request.urlopen(url).read()
        soup = BeautifulSoup(opener, 'lxml')
        imgs = soup.findAll('img', src=True)
        total = len(imgs)

        linked_to_same = 0
        avg =0
        for image in imgs:
            subDomain, domain, suffix = extract(image['src'])
            imageDomain = domain
            if(websiteDomain==imageDomain or imageDomain==''):
                linked_to_same = linked_to_same + 1
        vids = soup.findAll('video', src=True)
        total = total + len(vids)

        for video in vids:
            subDomain, domain, suffix = extract(video['src'])
            vidDomain = domain
            if(websiteDomain==vidDomain or vidDomain==''):
                linked_to_same = linked_to_same + 1
```

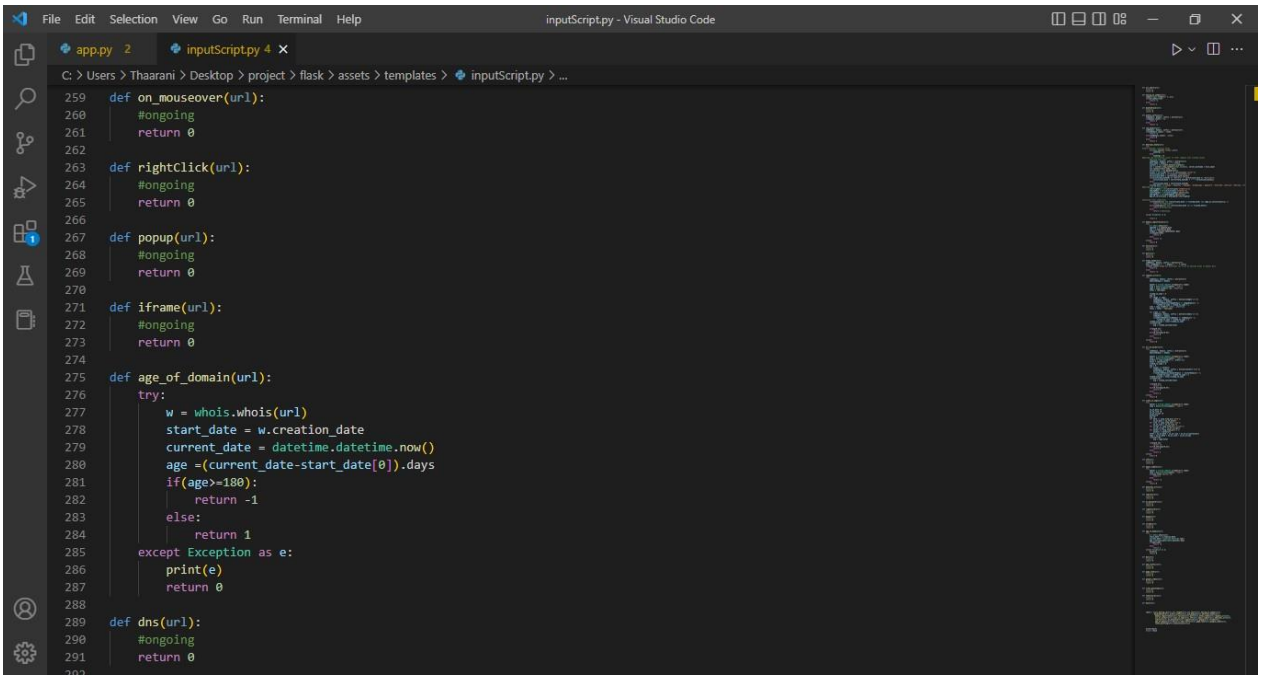
```python
        linked_outside = total-linked_to_same
        if(total!=0):
            avg = linked_outside/total

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


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

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



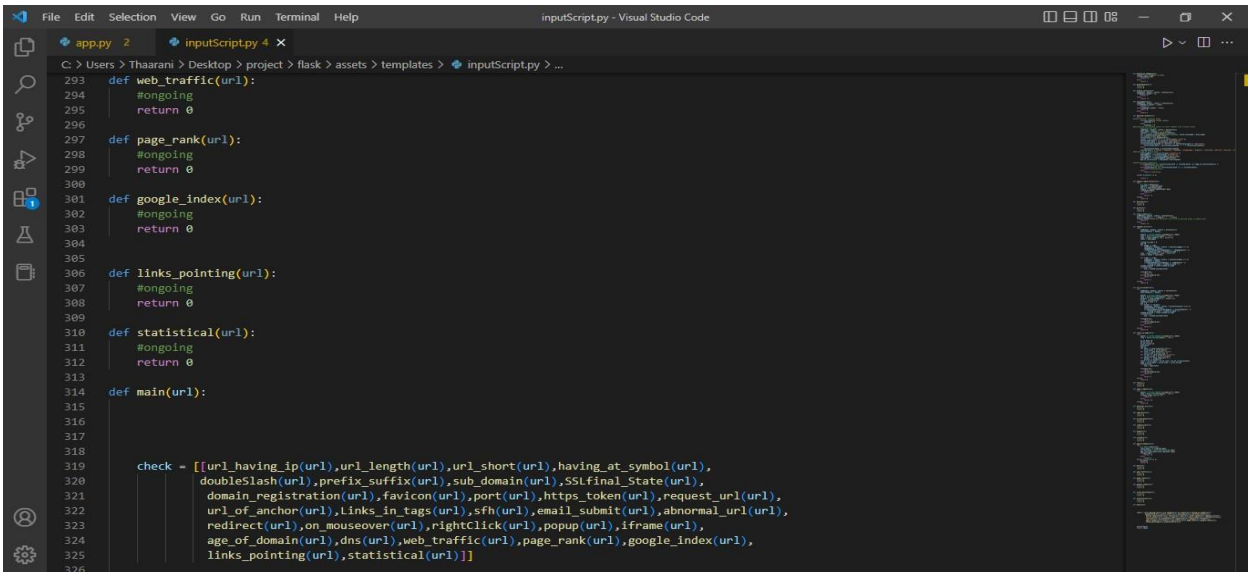

```
        if(avg<0.31):
            return -1
        elif(0.31<=avg<=0.67):
            return 0
        else:
            return 1
    except:
        return 0

def Links_in_tags(url):
    try:
        opener = urllib.request.urlopen(url).read()
        soup = BeautifulSoup(opener, 'lxml')

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

        if(avg<0.25):
```




```
        if(avg<0.25):
            return -1
        elif(0.25<=avg<=0.81):
            return 0
        else:
            return 1
    except:
        return 0

def sfh(url):
    #ongoing
    return 0

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

def abnormal_url(url):
    #ongoing
    return 0

def redirect(url):
    #ongoing
    return 0

def on_mouseover(url):
```




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

def rightClick(url):
    #ongoing
    return 0

def popup(url):
    #ongoing
    return 0

def iframe(url):
    #ongoing
    return 0

def age_of_domain(url):
    try:
        w = whois.whois(url)
        start_date = w.creation_date
        current_date = datetime.datetime.now()
        age =(current_date-start_date[0]).days
        if(age>=180):
            return -1
        else:
            return 1
    except Exception as e:
        print(e)
        return 0

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




```python
def web_traffic(url):
    #ongoing
    return 0

def page_rank(url):
    #ongoing
    return 0

def google_index(url):
    #ongoing
    return 0


def links_pointing(url):
    #ongoing
    return 0

def statistical(url):
    #ongoing
    return 0

def main(url):



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

```python
def links_pointing(url):
    #ongoing
    return 0

def statistical(url):
    #ongoing
    return 0

def main(url):



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


    print(check)
    return check
```

Build An HTML Page

We Build an HTML page to take the URL as a text and upon clicking on the button for submission it has to redirect to the URL for “y_predict” which returns if the URL given is phishing or safe. The output is to be then displayed on the page. The HTML pages are put under the templates folder and any style sheets if present is kept in the static folder.

here we first built a css file called style.css file

```
#title{
    padding-top: 1cm;
    padding-left: 4cm;
    color: white;
    font-size: 1cm;
     }
    #sub-t{
        padding-left: 4cm;
        color: white;
        margin-bottom: -1cm;
        padding-top: 4cm;
    }
    #blue{
        background-color: #1a396b;
        height: 27cm;

    }
    #white{
        background-color: white;
    }
    p{
        color: grey;
        font-size: .8cm;
        padding-left: 4cm;
    }
    h1{
        font-size: 2cm;
    }
    a{
        font-size: medium;
        color: white;
        font-style: normal;
        padding: 1cm;
        text-decoration: none;
```

```
        text-decoration: none;
    }
    span{
        padding-left: 4cm;
        color: white;
    }
    #start{
        border-radius: 50px;
        height: 1.5cm;
        width: 5cm;
        background-color: #0a9ff0;
        color: white;
        padding: 10px;
        cursor: pointer;

    }
    #Video{
        visibility: hidden;
    }
    img{
    padding: 190px;
    }
    #dark{
        background-color: #1a396b;
        color: white;
        height: 15cm;
        margin-bottom: 2cm;
    }
    #stop{
        border-radius: 50px;
        height: 1.5cm;
        width: 5cm;
        background-color: #0a9ff0;
```

```
        height: 1.5cm;
        width: 5cm;
        background-color: #0a9ff0;
        color: white;
        padding: 10px;
        cursor: pointer;
    }
    #s1{

        padding-top: 3cm;
        color: white;


    }
    #div1{
        height: fit-content;
        width: fit-content;
        display: inline-block;

    }
    #div2{
        height: fit-content;
        width: fit-content;
        display: inline-block;


    }

```

style1.css

```
#title{
padding-bottom: 2cm;
padding-left: 4cm;
color: white;
font-size: normal;
}
body{
background-color: white;
}
a{
color: white;
font-size: normal;
font-style: normal;
padding: 2cm;
}
#blue{
background-color: #0098eb;
height: 2cm;
}
#white{
background-color: white;
color: black;
padding-top: 4cm;
font-size: 24;
}
#start{
border-radius: 90px;
height: 1.5cm;
width: 7cm;
background-color: #0098eb;
color: white;
padding: 10px;
cursor: pointer;
}
```

now moving on to the index.html source file

```html
<html>
    <head>
        <title>Phishing Website Detection</title>
        <link rel="stylesheet" href="{{ url_for('static', filename = 'style.css') }}">

    </head>
    <body >
        <div id="blue">
    <h1 id="title">NILA<span style="float:right"><a href="\">Home</a> <a href="About">About</a> <a>Contact</a><a href="\predict">Get Started</a></span></h1>
    <br>
    <img src="{{ url_for('static', filename = 'phishing image.jpg') }}" style="float: right" width="700" height="500">
    </img>
    <h1 id="sub-t">Solution to Detect <br>Phishing Websites</h1>
    <p>Be aware of what's happening with you <br>confidential data</p>

    <span> <button style="font-size:20px" id="start"><a href="\predict">Get Started</a></button>  <label for="video">Watch Video</label><button style="font-size:20px" id="Video"
    <div id="white">
        <h1 style="text-align:center;">About</h1>
    </div>
    <div>
        <div style="display: flex; justify-content: space-between;">
            <p style="background-color: white;">Web service is one of the key communications software services for the Internet. Web phishing is one of many security threats t
            <p style="background-color: white;">The recipient is then tricked into clicking a malicious link, which can lead to the installation of malware,the freezing of the
          </div>
    </div>
    <div id="dark">
        <div id="div1">
        <p id="s1"><b>Check your Website</b><br>Understanding if the website is valid one or not is important and plays a vital role in the security of data.<br>To know if the U
        </div>
        <div id="div2">
            <a id="stop" href="\predict">Check your website</a>

        </div>

    </div>


    </div>
    </body>
</html>
```

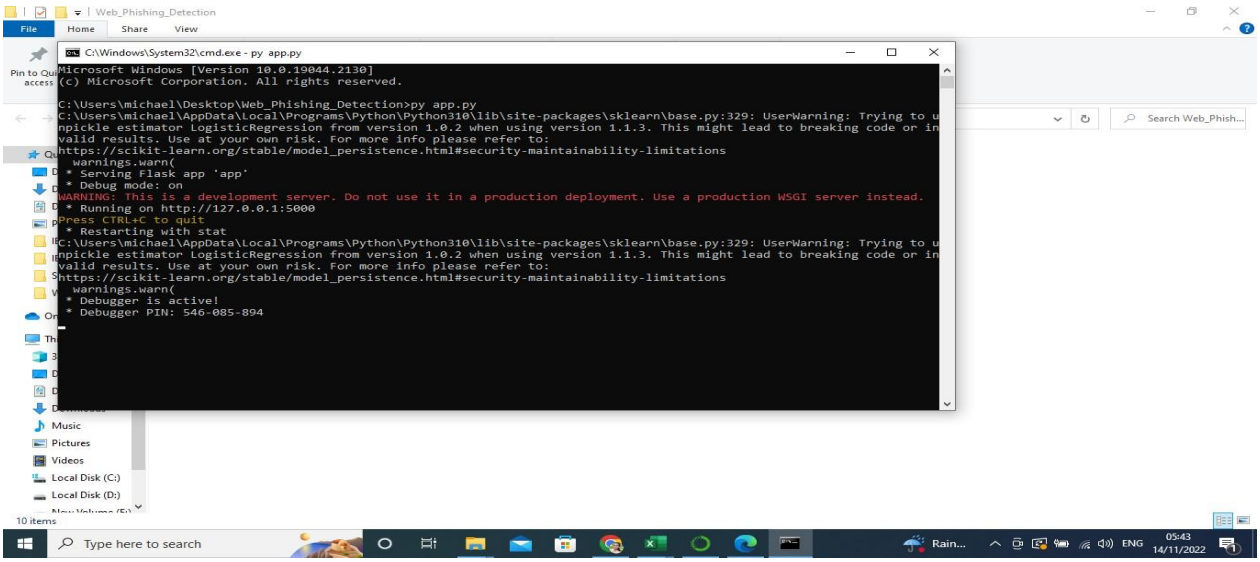
now final.html source file

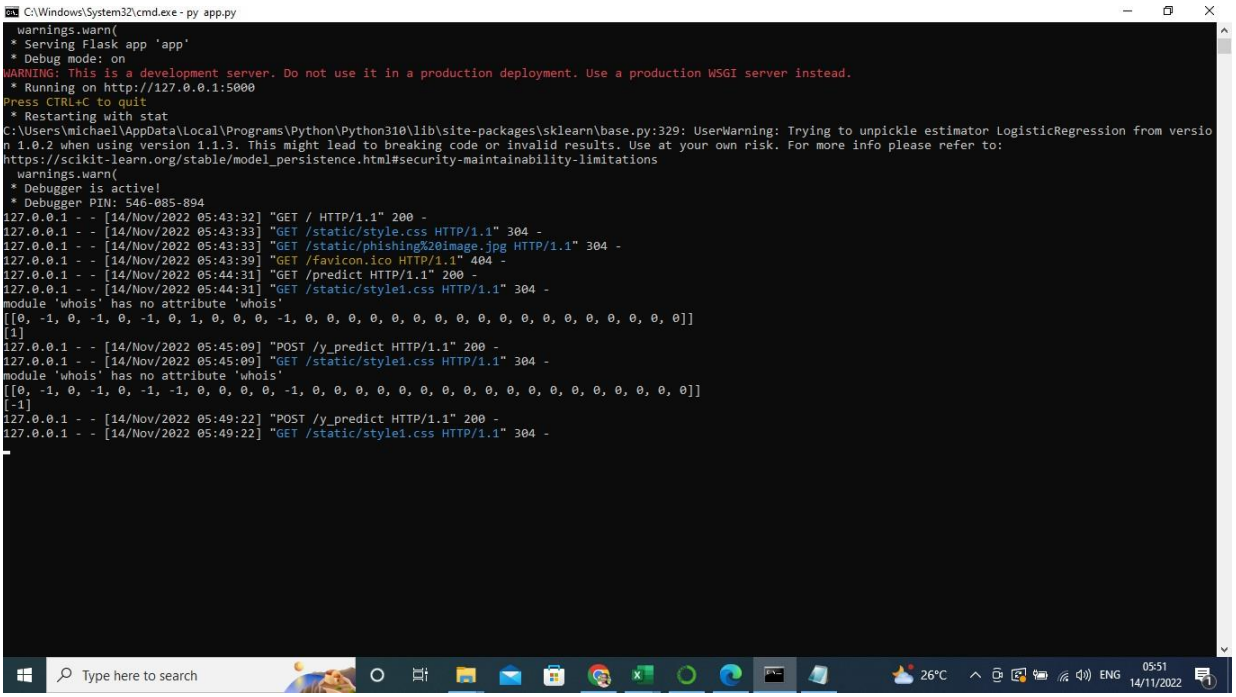
```html
<html>
    <head>
        <link rel="stylesheet" href="{{ url_for('static', filename = 'style1.css') }}">
    </head>
    <body>
        <div id="blue">
        <h1 id="title">NILA<span style="float:right"><a href="\">Home</a> <a href="\">About</a> <a>Contact</a>
            </span>
        </h1>
        </div>
        <div id="white">
            <center>
            <form action="{{ url_for('y_predict') }}" method="post">
            <h1><b><center>Phishing Website Detection using Machine Learning</center></b></h1>
            <input type="text" id="url" name="url" value="{{ url_path }}" size="120"><br><br>
            <input type="submit" style="font-size:20px" id="start" value="Predict">
            </form>
            {{ url }}
            </center>
        </div>
        </body>
</html>
```

FINAL OUTPUT:

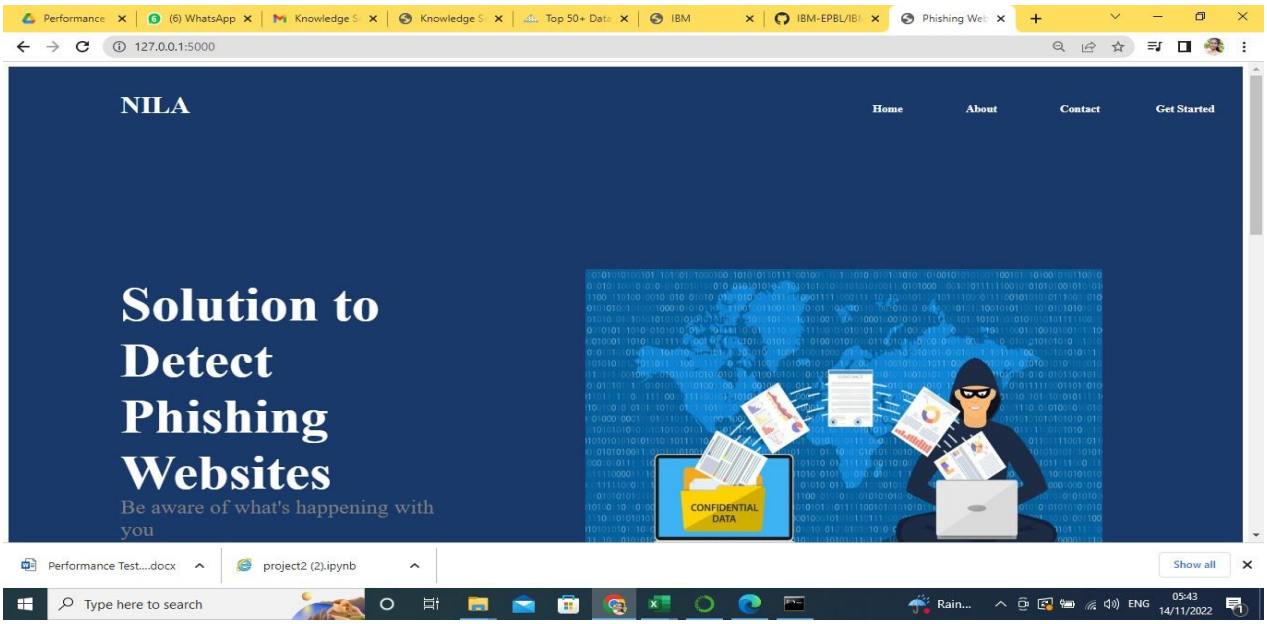
After executing the source codes we get the following outputs as follows

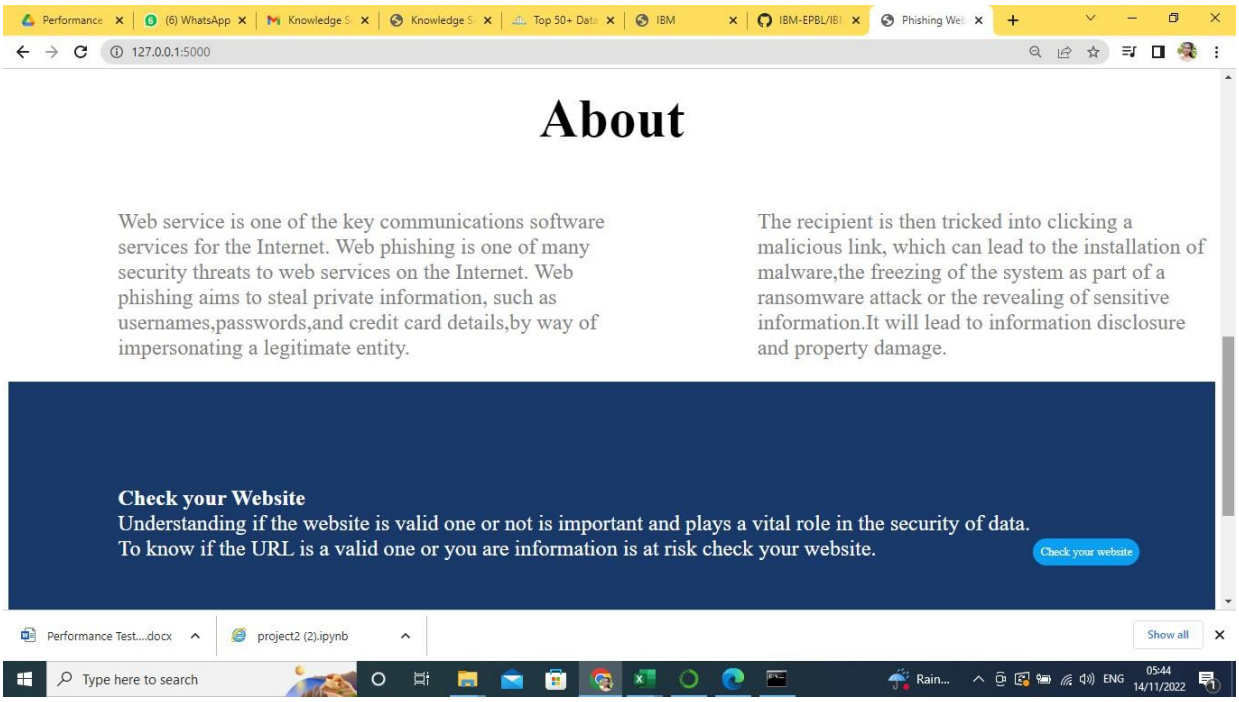
first we get a command prompt output with the URL over there we have to copy that particular URL and paste it in your web browser. In our project we have used google chrome web browser for the execution.



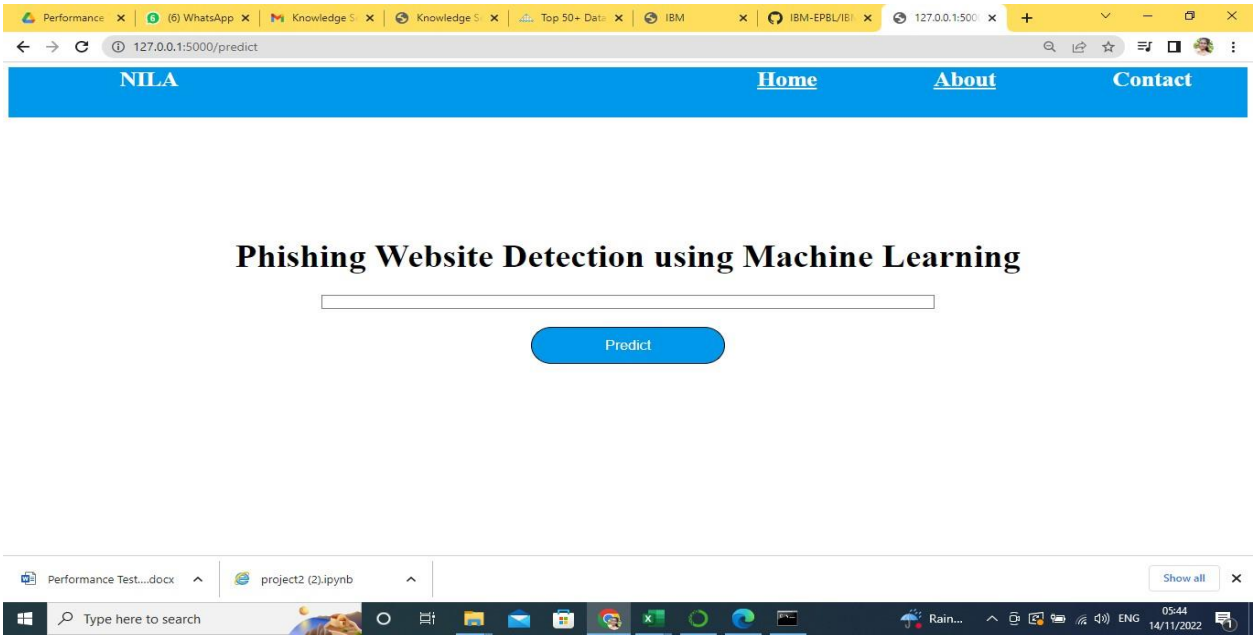


after the URL has been pasted we get the home page of our project web phishing detection



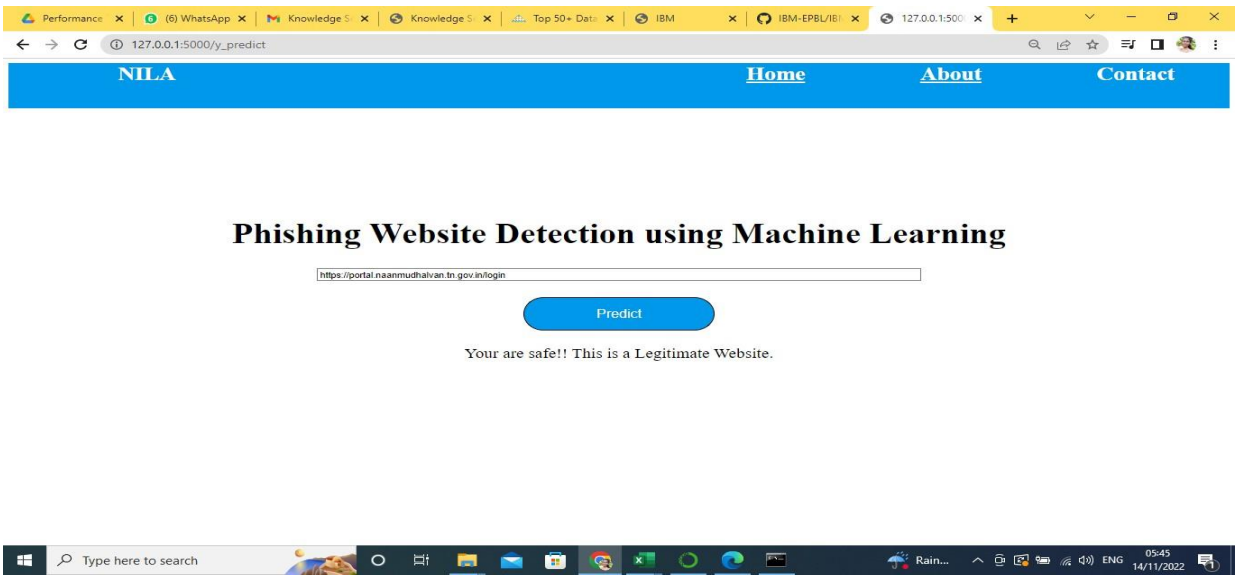


Here we have two options to predict the website one is we can click on check your website or you can click on get started to proceed your website prediction the diplay screen is as follows



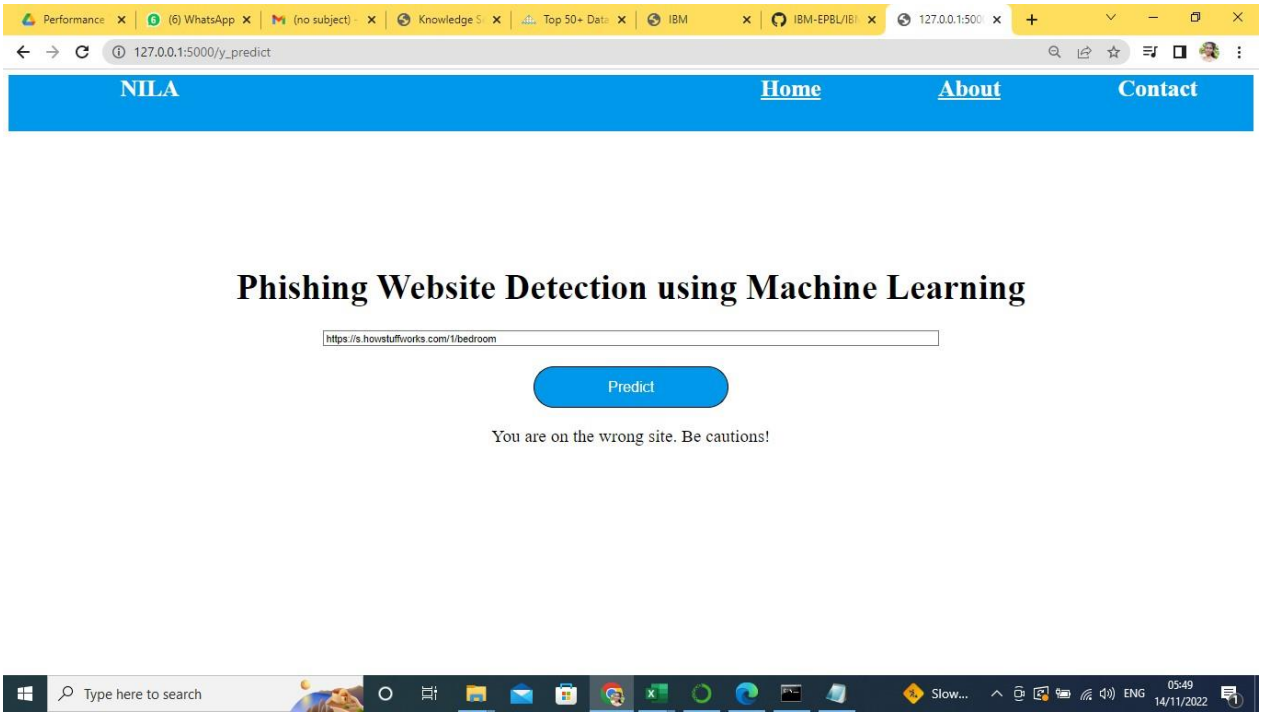
PREDICTION OF SAFE WEBSITE:

Here we have used the original website as a prediction so now we have pasted that URL in our project web page and we clicked on Predict and here the result will be produced as **"YOU ARE SAFE! THIS IS A LEGITIMATE WEBSITE."**

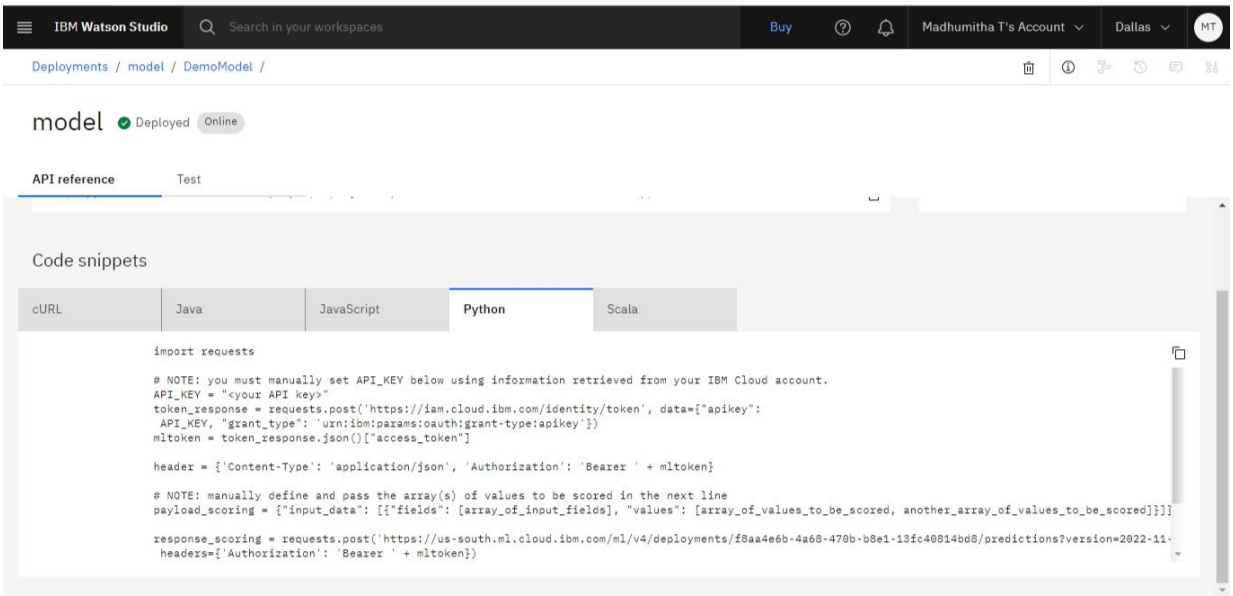


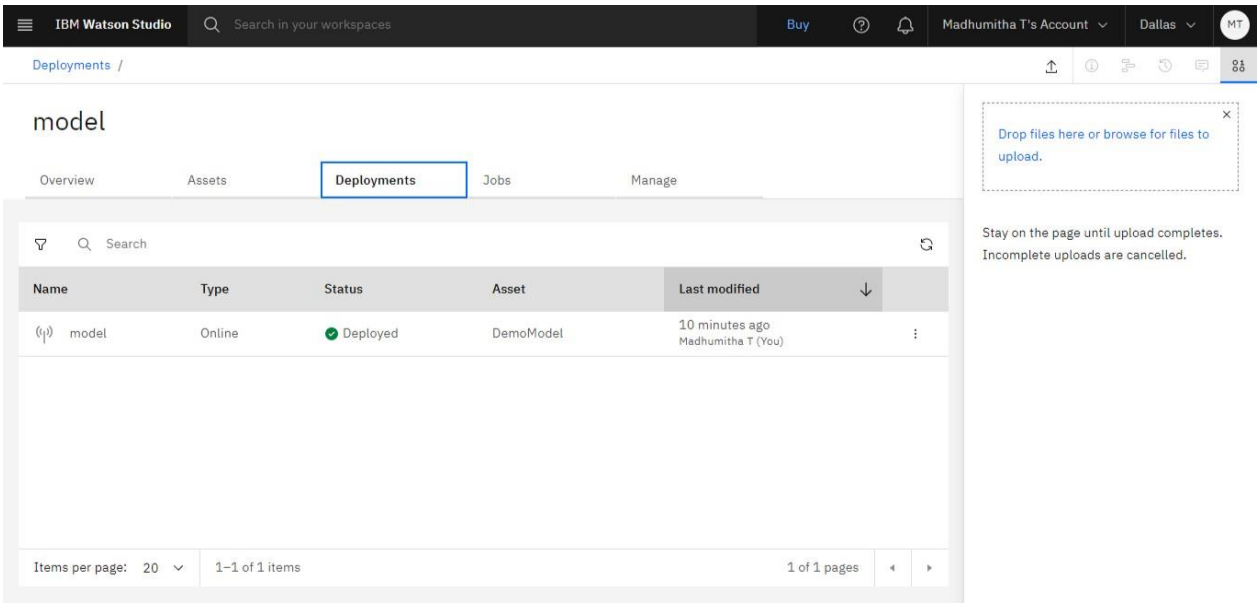
PREDICTION OF UNSAFE WEBSITE:

Here we have used a fake website as a prediction so now we again pasted the fake URL in our project web page and we clicked on Predict and here the result will be produced as **"YOU ARE ON THE WRONG SITE BE CAUTIONS!"**

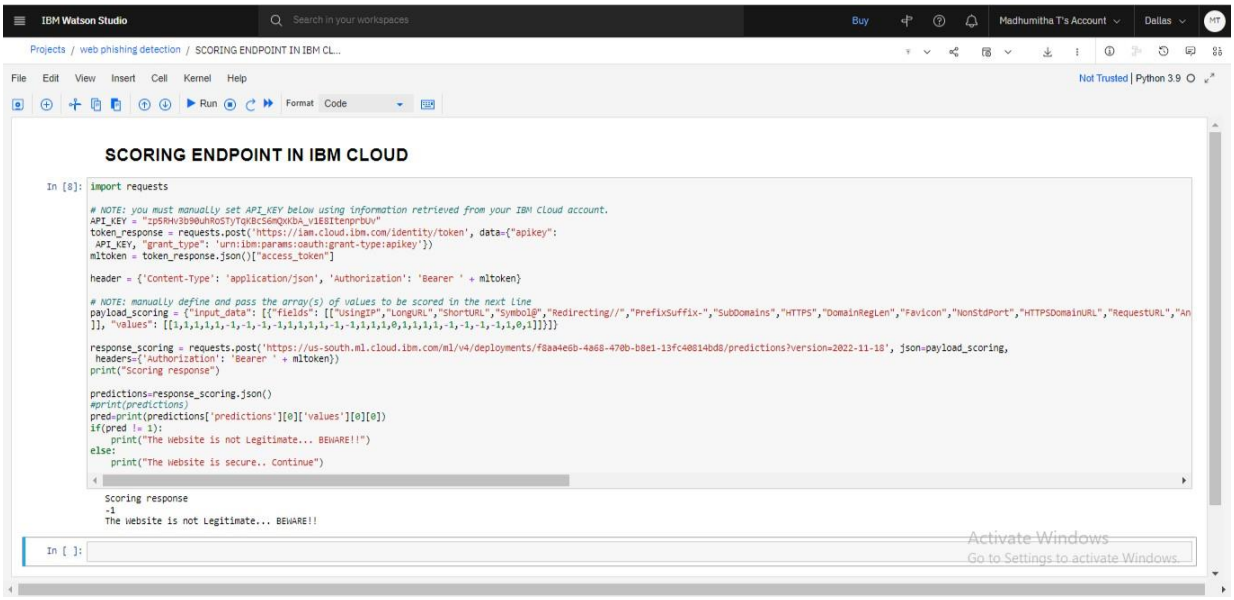


IBM CLOUD DEPLOYMENT:

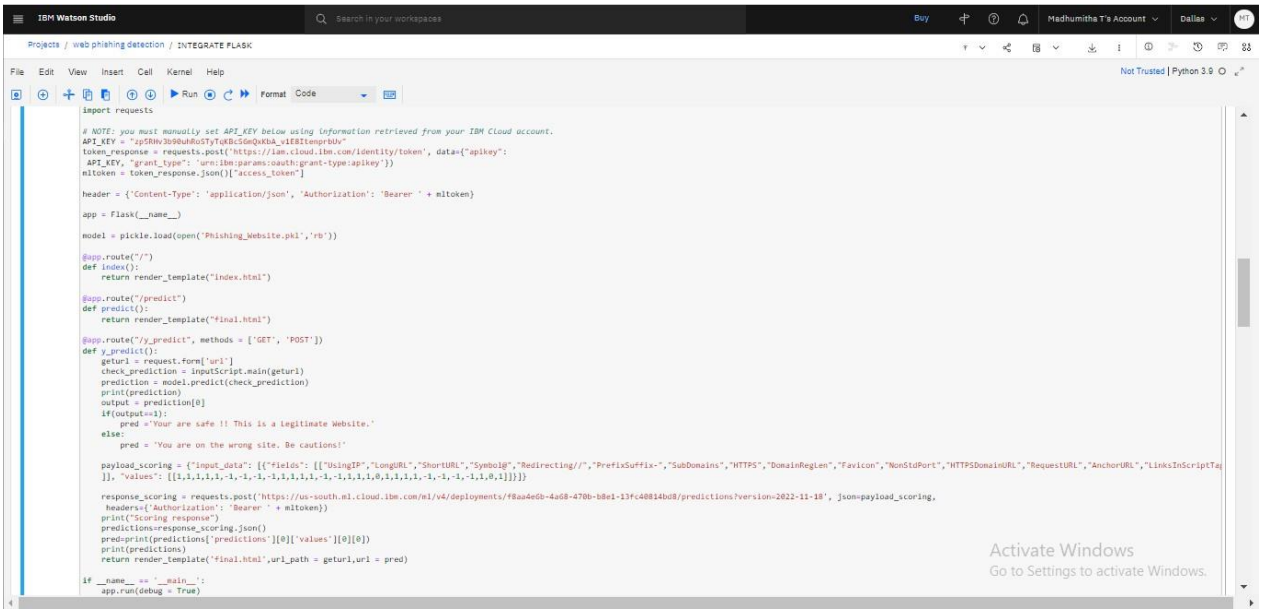




SCORING ENDPOINTS IN IBM CLOUD:



INTEGRATING FLASK WITH IBM CLOUD:



LINKS:

GITHUB LINK: [IBM-EPBL/IBM-Project-22344-1659849670: Web Phishing Detection (github.com)]

VIDEO LINK: **click here**