

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



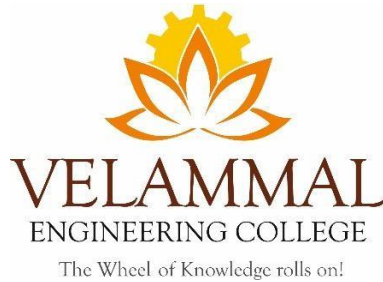
VELAMMAL ENGINEERING COLLEGE, CHENNAI-66.

(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

Mrs. DOLLY IRENE

ASSISTANT PROFESSOR I

Department of Electronics and
Communication Engineering
Velammal Engineering College
Chennai-600 066

HEAD OF THE DEPARTMENT

Dr. S. MARY JOANS

PROFESSOR & HEAD

Department of Electronics and
Communication Engineering
Velammal Engineering College
Chennai-600 066

TABLE OF CONTENT

S.NO	TITLE	PAGE NO.
1	INTRODUCTION	1
	1.1 Project overview	1
	1.2 Purpose	2
2	LITERATURE SURVEY	3
	2.1 Existing problems	3
	2.3 Problem statement and definition	3
3	IDEATION & PROPOSED SOLUTION	5
	3.1 Empathy Map Canvas	5
	3.2 Ideation & Brainstorming	6
	3.3 Proposed Solution	6
	3.4 Problem Solution fit	7
4	REQUIREMENT ANALYSIS	8
	4.1 Functional requirement	8
	4.2 Non-Functional requirements	9
5	PROJECT DESIGN	11
	5.1 Data Flow Diagrams	11
	5.2 Solution & Technical Architecture	12
	5.3 User Stories	13
6	PROJECT PLANNING & SCHEDULING	15
	6.1 Sprint Planning & Estimation	15
	6.2 Sprint Delivery Schedule	16
	6.3 Reports from JIRA	17
7	CODING & SOLUTIONING	20
	(Explain the features added in the project along with code)	
	7.1 Feature 1	20
	7.2 Feature 2	21

8	TESTING	23
	8.1 Test Cases	23
	8.2 User Acceptance Testing	24
9	RESULTS	25
	9.1 Performance Metrics	25
10	ADVANTAGES & DISADVANTAGES	26
11	CONCLUSION	27
12	FUTURE SCOPE	28
13	REFERENCE	30
14	APPENDIX	31
	SOURCE CODE	31
	GITHUB & PROJECT DEMO LINK	46

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:

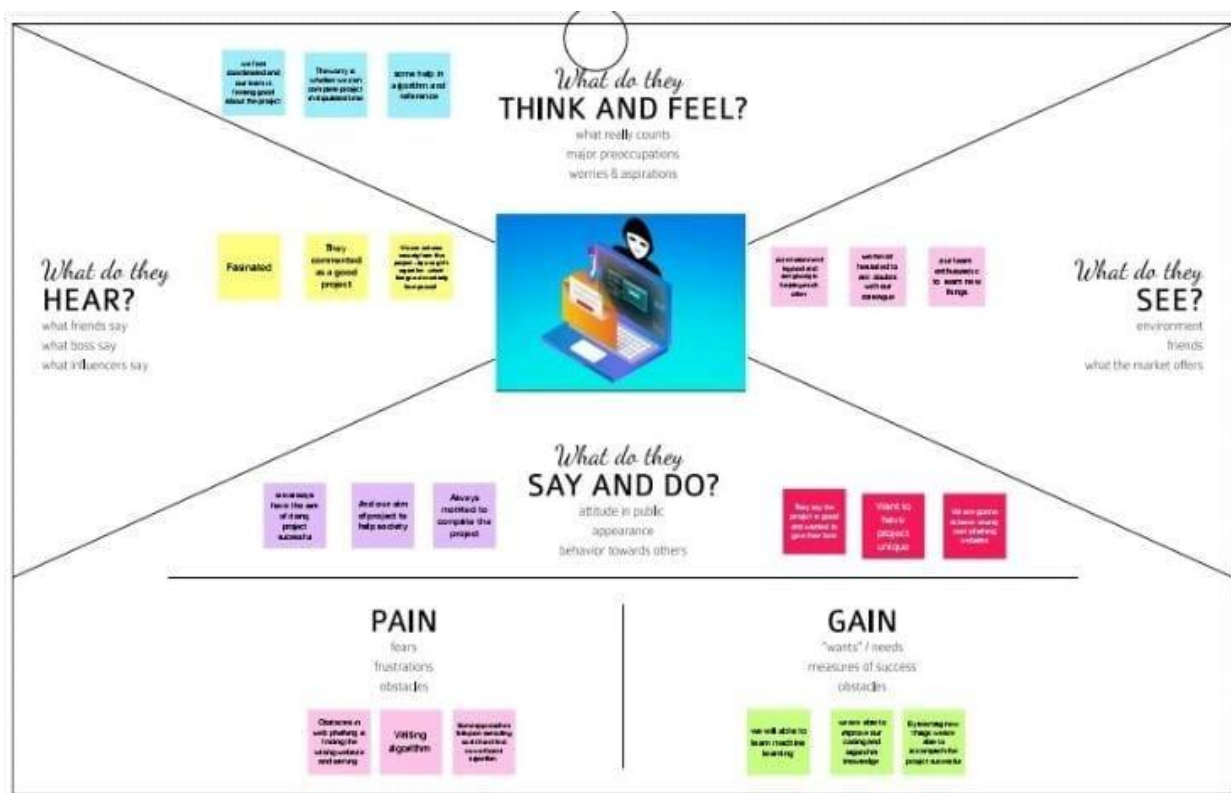
Email phishing is a numbers game. An attacker sending out thousands of fraudulent messages can net significant information and sums of money, even if only a small percentage of recipients fall for the scam. As seen above, there are some techniques attackers use to increase their success rates. For one, 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

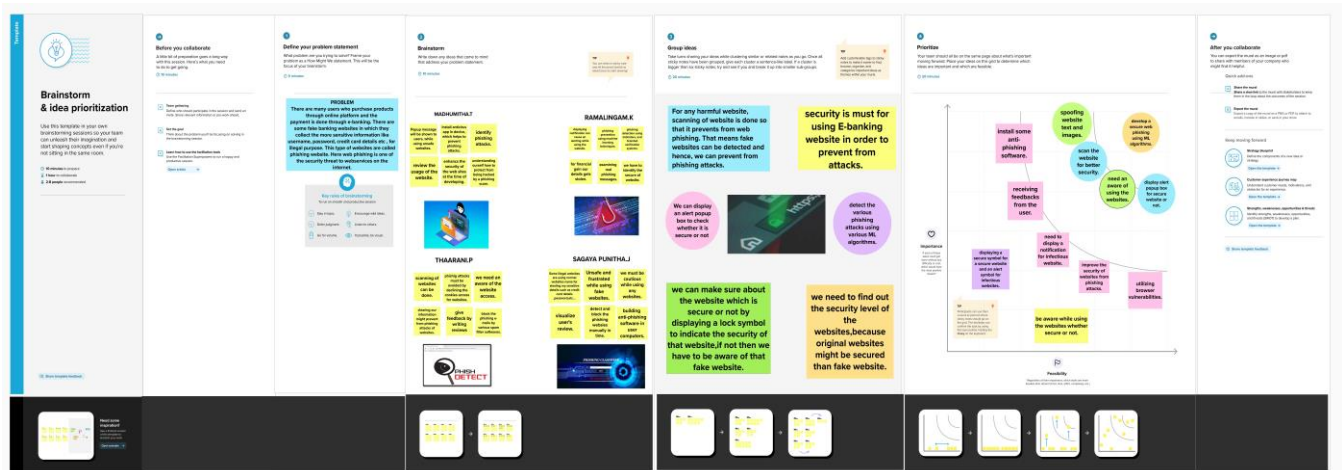
IDEATION & PROPOSED SOLUTION

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 displays an alert box which leads to an awareness 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:

Define CS, fit into CC	1. CUSTOMER SEGMENT(S) CS Who is your customer? i.e. working parents of 0-5 y.o. kids <ul style="list-style-type: none"> User who uses online shopping websites. The one who make money transaction through e banking websites. 	6. CUSTOMER CONSTRAINTS CC What constraints prevent your customers from taking action or limit their choices of solutions? i.e. spending power, budget, no cash, network connection, available devices. <ul style="list-style-type: none"> The customer don't know where to report the issue They are not ready to lose their information They were not aware of the person behind these attacks 	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? i.e. pen and paper is an alternative to digital not taking. <ul style="list-style-type: none"> The user must get an alert in prior while they visit the website The website can be scanned so that the virus is prevented in user's mobile and computer 	Explore AS, differentiate
	Focus on J&P, tap into BE, understand RC	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 phishing websites must be detected in prior. The user while visiting the website can be warned prior while they get into it. 	9. PROBLEM ROOT CAUSE RC What is the real reason that this problem exists? What is the back story behind the need to do this job? i.e. customers have to do it because of the change in regulations. <ul style="list-style-type: none"> The hackers use new techniques for creating a fake website Not having prior knowledge to the users The ML prediction accuracy is less. There were not that much research are carried out in this field 	
Identify strong TR & EM		3. TRIGGERS TR What triggers customers to act? i.e., seeing their neighbour installing solar panels, reading about a more efficient solution in the news. <ul style="list-style-type: none"> It receives alert messages in the link the user clicks They might have no prior knowledge about the kind of attacks done while clicking the websites 	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 proposal, then keep it blank until you fill 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 can install anti phishing website in order to prevent virus attack We can give prior alert box while using the website to predict that the website we are using is secure or not User must be aware of the phishing websites and they can prevent the loss of their personal information 	8. CHANNELS of BEHAVIOUR CH 8.1 ONLINE What kind of actions do customers take online? Extract a new channel from #7. <ul style="list-style-type: none"> They provide all their personal details including credit card information to some websites 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"> They try to research more information regarding attacks through books or from public
	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"> They may feel insecure while using the website. They lose all their details and credit card information and because of that they feel frustrated. 			

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:

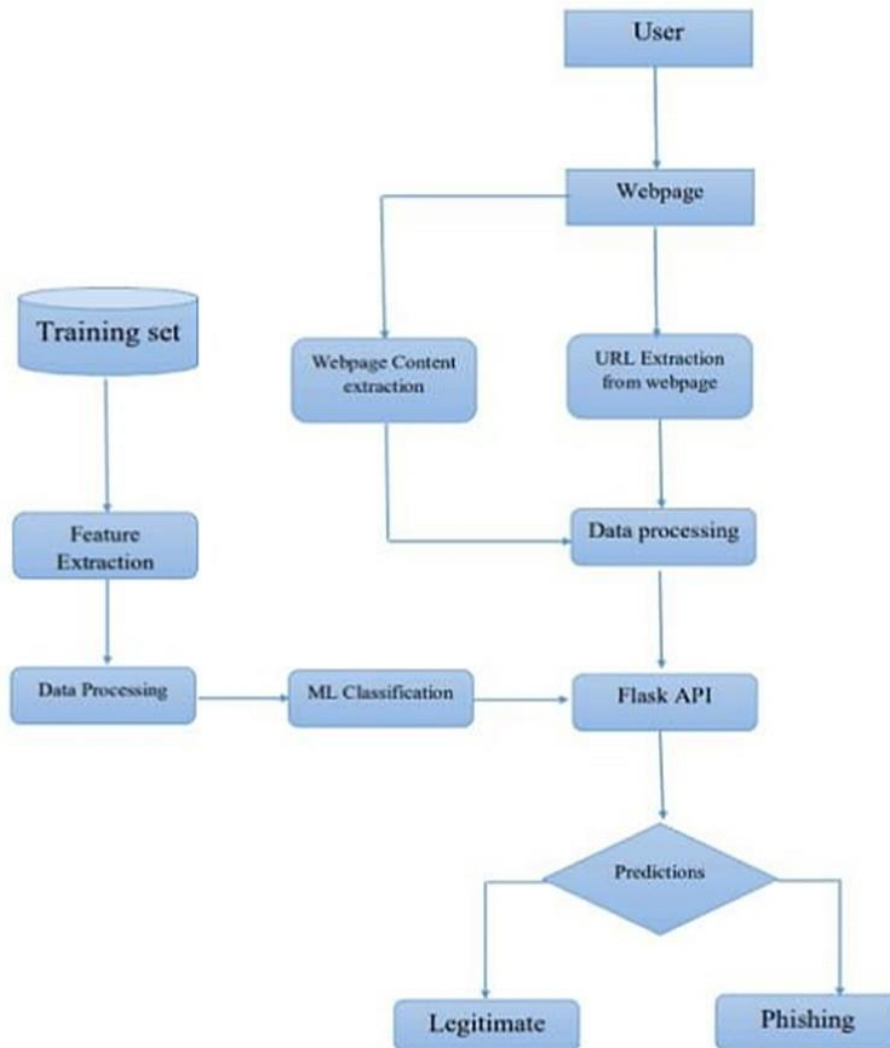
FR No.	Non-Functional Requirement	Description
NFR-1	Usability	User can have full access to the particular websites they using must proceed some certain user friendly websites so that it does not affect the datas.
NFR-2	Security	To check whether the particular website is secure or not we can notify it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

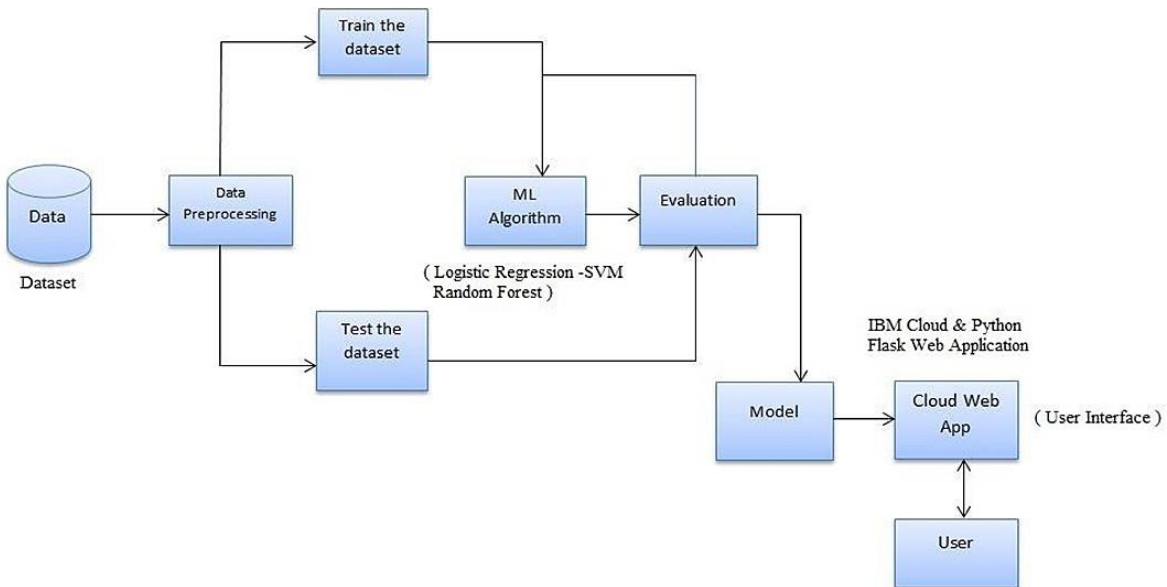
PROJECT DESIGN

5.1 DATA FLOW DIAGRAMS:

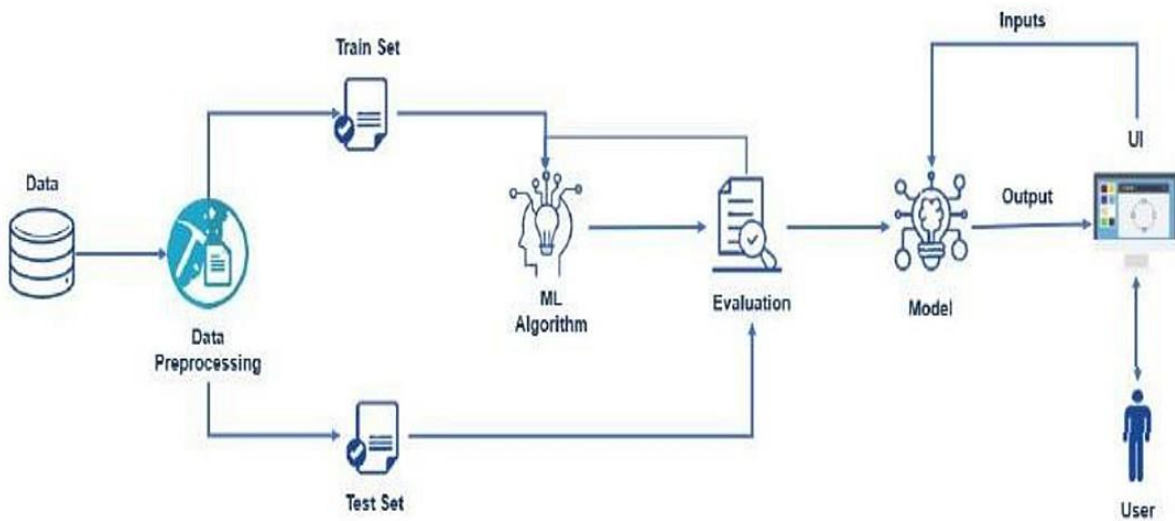


5.2 SOLUTION & TECHNICAL ARCHITECTURE:

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

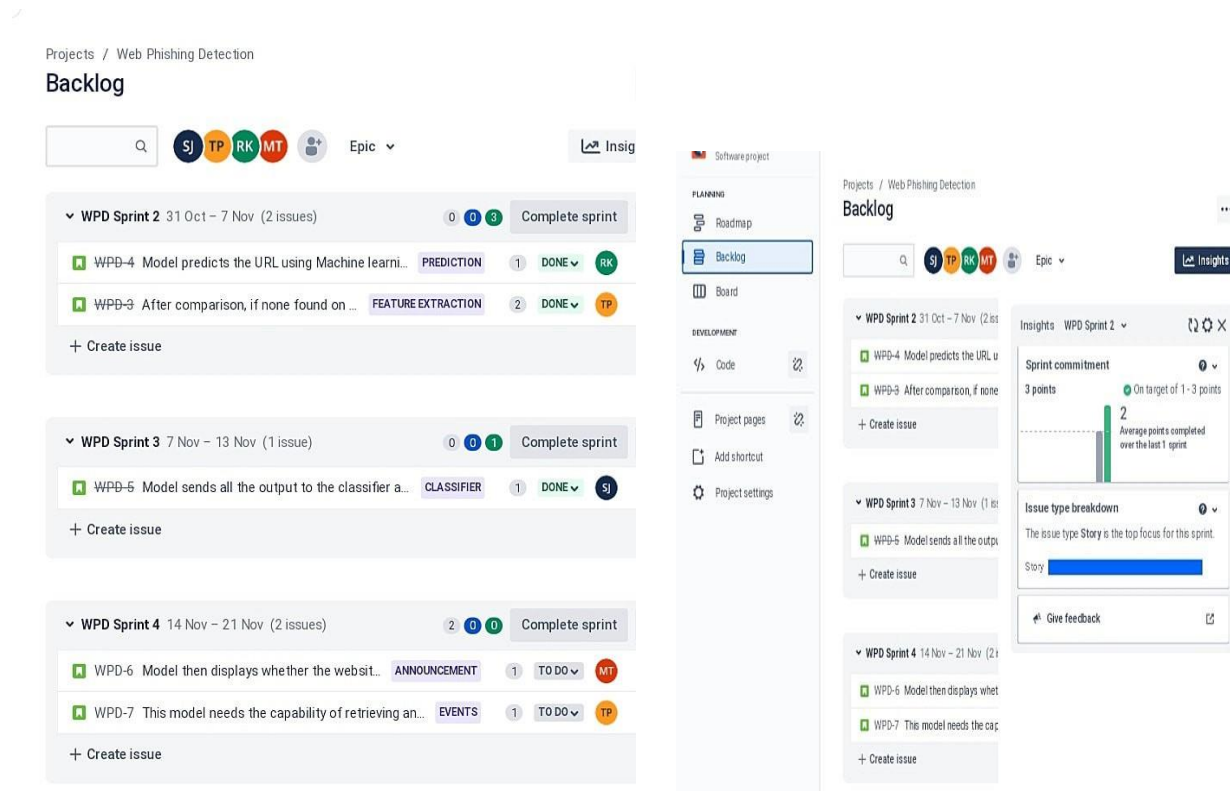
Sprint-3	Classifier	USN-5	Model sends all the output to the classifier and produces the final result.	1	Medium	Sagaya Punitha J
Sprint-4	Announcement	USN-6	Model then displays whether the website is legal site or a phishing site.	1	High	Madhumitha T
Sprint-4	Events	USN-7	This model needs the capability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

6.2 SPRINT DELIVERY SCHEDULE:

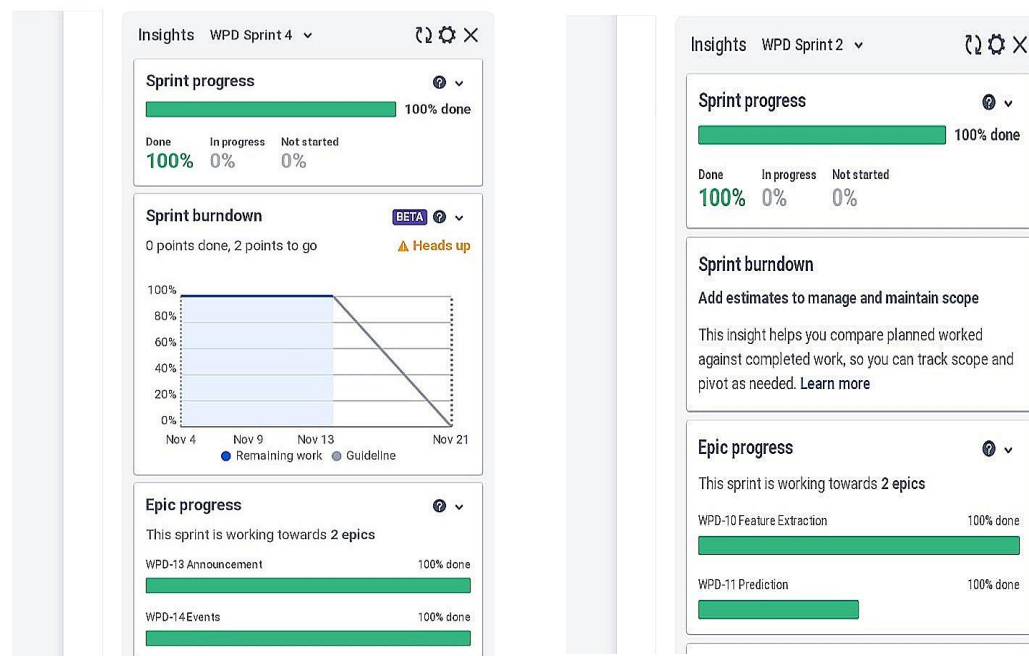
Sprint	Total Story Points	Duration	Sprint Start Date	Sprint End Date (Planned)	Story Points Completed (as on Planned End Date)	Sprint Release Date (Actual)
Sprint-1	20	6 Days	24 Oct 2022	29 Oct 2022	20	29 Oct 2022
Sprint-2	20	6 Days	31 Oct 2022	05 Nov 2022	20	05 Nov 2022
Sprint-3	20	6 Days	07 Nov 2022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

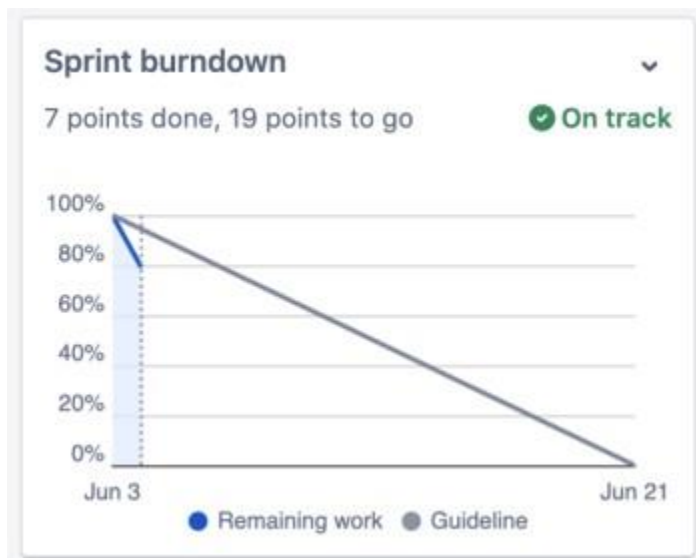
1.BACKLOG:(BACKLOG-1 & BACKLOG-2)



2.SPRINT PROGRESS(FINAL PROGRESS & INSIGHTS):

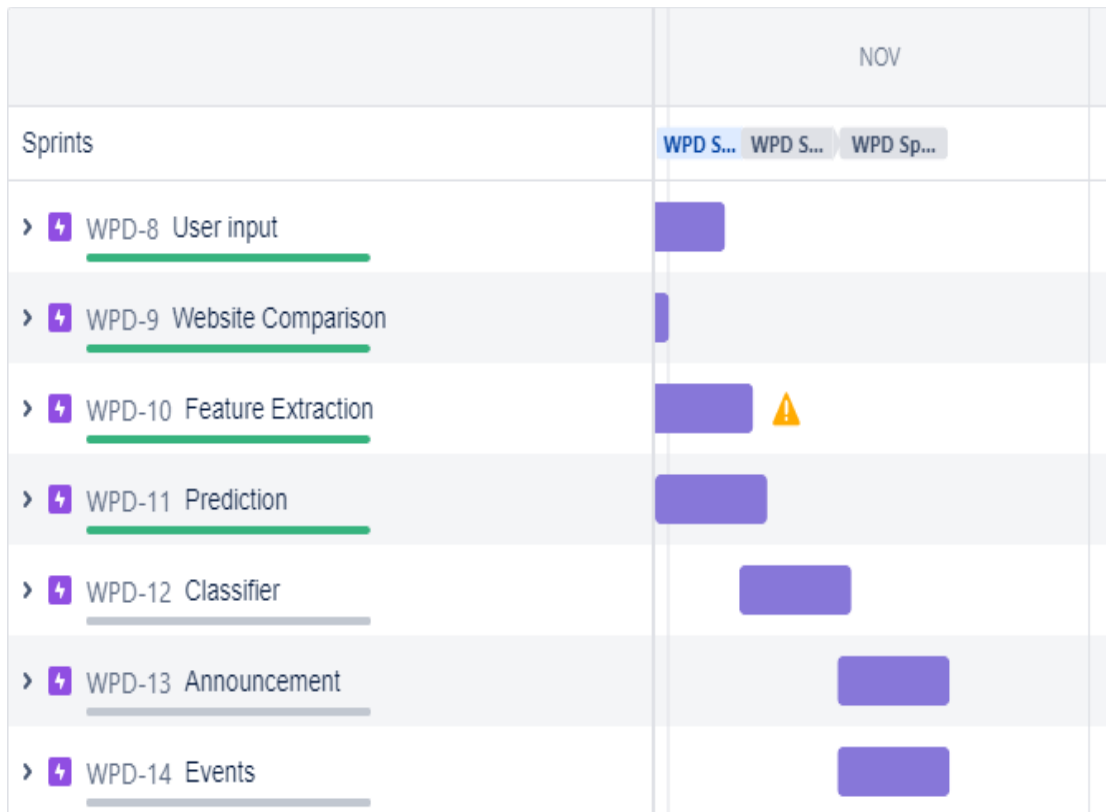


3.SPRINT BURNDOWN:

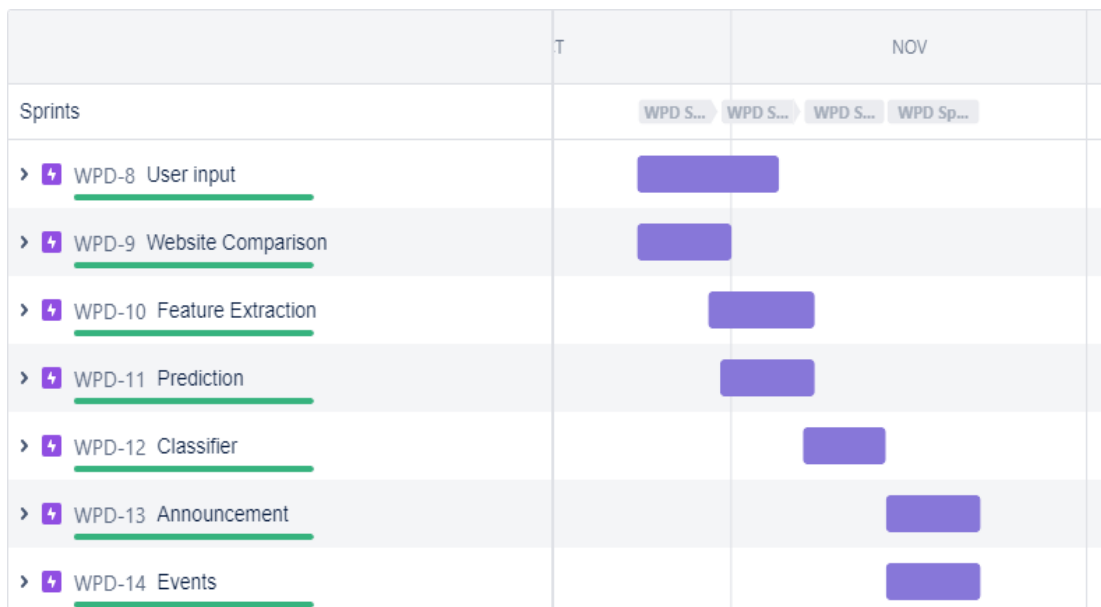


4.SPRINT 1 &2:

		NOV
Sprints	WPD Sp...	
> ⚡ WPD-8 User input		
> ⚡ WPD-9 Website Comparison		
> ⚡ WPD-10 Feature Extraction		
> ⚡ WPD-11 Prediction		



5.SPRINT 4&5:



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	SSL
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_registration_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   SPH                    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   popuupidnow            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	Shortning_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	11055.000000
mean	5528.000000	0.313798	-0.853168	0.738761	0.700688	0.741474	-0.734882	0
std	3191.447947	0.046534	0.786095	0.873966	0.713698	0.871011	0.878139	0
min	1.000000	-1.000000	-1.000000	-1.000000	-1.000000	-1.000000	-1.000000	-1
25%	2784.500000	-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 x 9 columns

```
In [25]: ds.shape
Out[25]: (11855, 32)
```

```
In [32]: #no null values
ds.isnull().any()
Out[32]:
```

	index	having_IPhaving_IP_Address	URLURL_Length	Shortning_Service	having_At_Symbol	double_slash_redirecting	Prefix_Suffix	having_Sub	Domain	registration_length	Favicon	port	HTTPS_token	Request_URL	URL_of_Anchor	Link_in_tags	SFH	Submitting_to_email	Abnormal_URL	Redirect	on_mouseover	RightClick	popupwindow	Iframe	age_of_domain	DNSRecord	web_traffic	Page_Rank	Google_Index	Link_pointing_to_page	Statistical_report	Result
dtype:	bool	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	

```
In [33]: #split data as indep(x-all cols) and dep(y-Result col)
#remove index col in indep ds(31 cols)
x = ds.iloc[:,1:32].values
y = ds.iloc[:,0].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',1,inplace=True)
ds.head()
```

```
Out[47]:
```

	index	having_IPhaving_IP_Address	URLURL_Length	shortning_Service	having_At_Symbol	double_slash_redirecting	Prefix_Suffix	having_Sub	Domain
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 x 10 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	shortning_service	having_at_symbol	double_main_redrecting	Prefix_Suffix	having_Sub_Domain	is
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 x 11 columns

```
< |
```

```
In [55]: x.mean()
Out[55]: -4.9760072533946e-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, 11)

In [59]: x_test.shape
Out[59]: (2211, 11)

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_001	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_002	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_003	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_004	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

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_005	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_006	Functional	Phishing website detection page	Verify it shows whether the URL entered by the user is safe or unsafe.	https://portalnaamudhavan.in.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://portalnaamudhavan.in.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_007	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) 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_008	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 expected	Fail	user enter URL without http	Yes		Automatic
LoginPage_TC_009	Functional	Phishing website detection page	Verify it shows whether the URL entered by the user is safe or unsafe.	portalnaamudhavan.in.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	portalnaamudhavan.in.gov.in/login	Application should display "you are safe! This is a legitimate website"	Not Working as expected	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 while using the particular website from the web browser.
- By using our project URL the user can easily 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 identification.
- 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 than 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 :

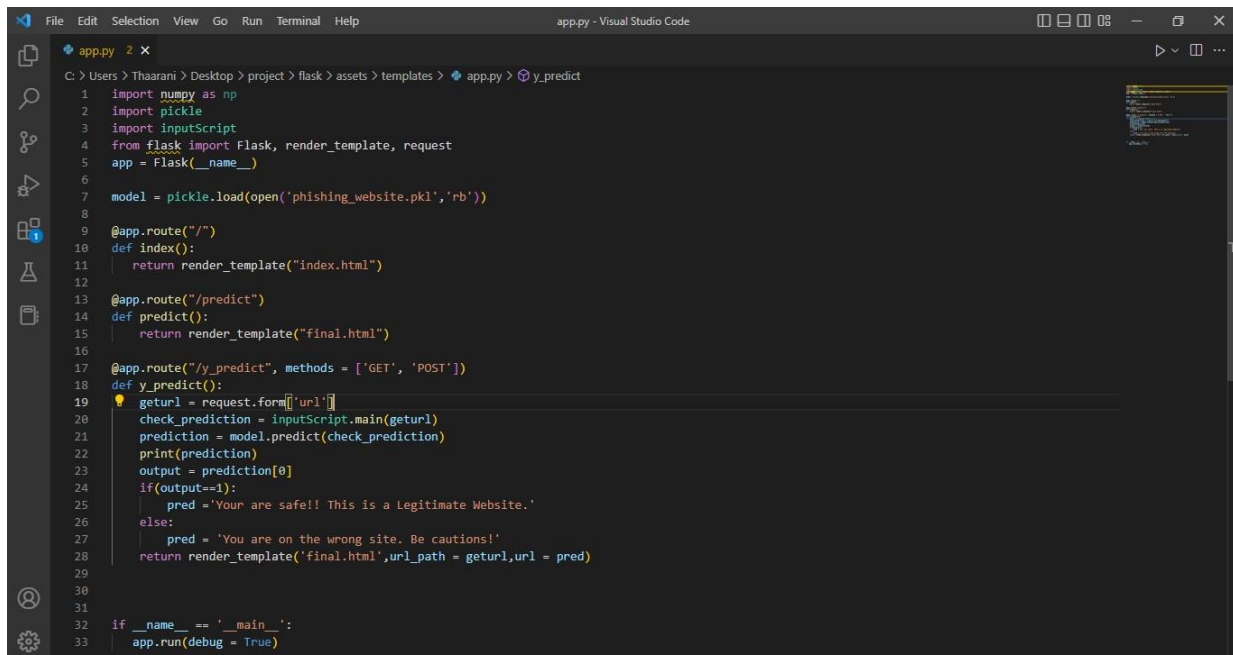
In the application, the user provides any website URL to check and the corresponding parameter values are generated by analysing the URL using which legitimate websites are detected.

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:

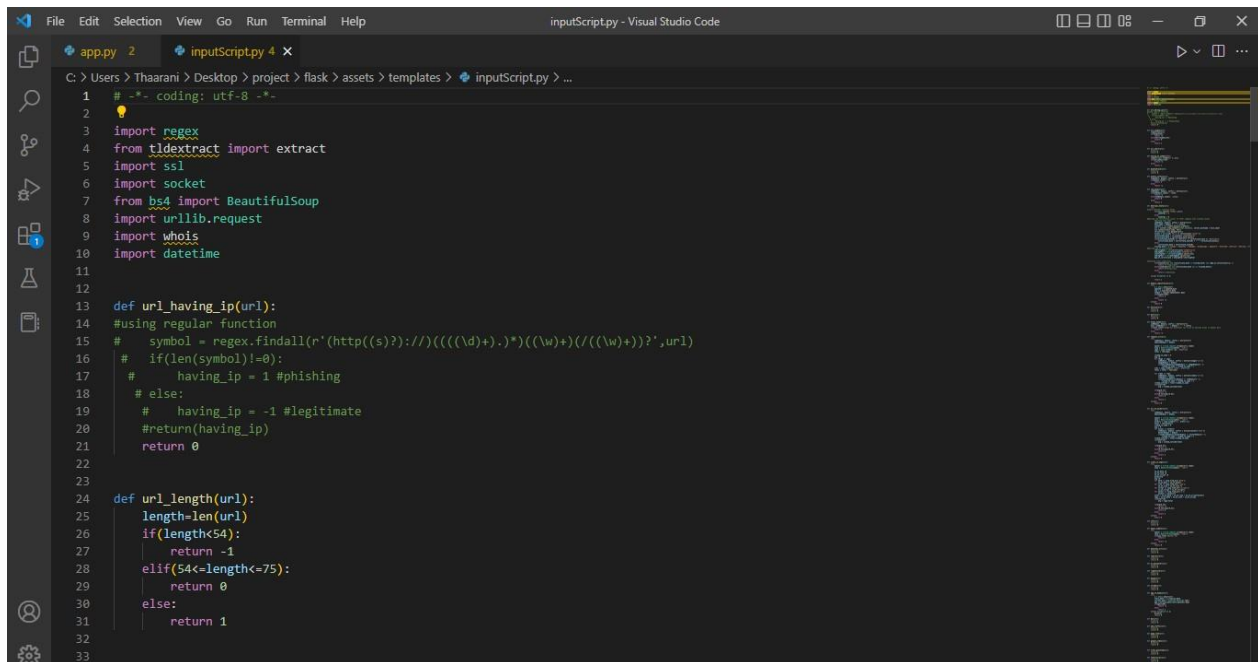
Here we have done the app.py code inorder to execute the Flask application.



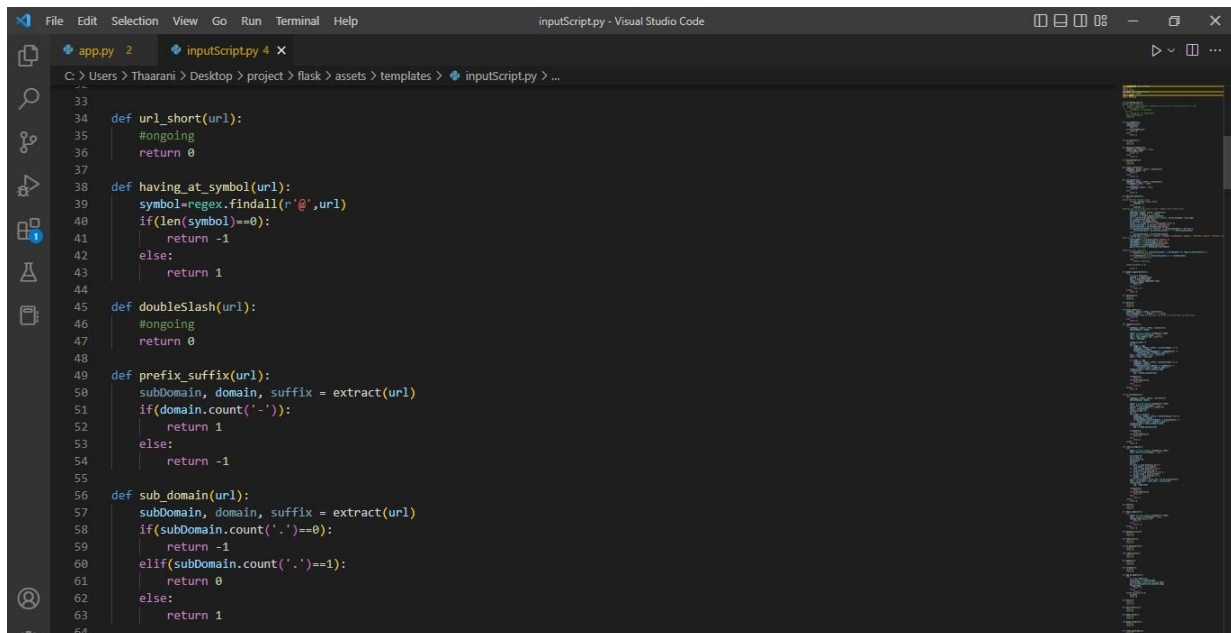
```
1 import numpy as np
2 import pickle
3 import inputScript
4 from flask import Flask, render_template, request
5 app = Flask(__name__)
6
7 model = pickle.load(open('phishing_website.pkl', 'rb'))
8
9 @app.route("/")
10 def index():
11     return render_template("index.html")
12
13 @app.route("/predict")
14 def predict():
15     return render_template("final.html")
16
17 @app.route("/y_predict", methods = ['GET', 'POST'])
18 def y_predict():
19     geturl = request.form['url']
20     check_prediction = inputScript.main(geturl)
21     prediction = model.predict(check_prediction)
22     print(prediction)
23     output = prediction[0]
24     if(output==1):
25         pred = 'Your are safe!! This is a Legitimate Website.'
26     else:
27         pred = 'You are on the wrong site. Be cautions!'
28     return render_template('final.html', url_path = geturl, url = pred)
29
30
31
32 if __name__ == '__main__':
33     app.run(debug = True)
```

INPUTSCRIPT.PY SOURCE CODE:

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



```
1  # -*- coding: utf-8 -*-
2
3  import regex
4  from tldextract import extract
5  import ssl
6  import socket
7  from bs4 import BeautifulSoup
8  import urllib.request
9  import whois
10 import datetime
11
12
13 def url_having_ip(url):
14     #using regular function
15     symbol = regex.findall(r'(http(s)?://)((\d+)+.*)((\w+)?/((\w+)))?',url)
16     # if(len(symbol)!=0):
17     #     having_ip = 1 #phishing
18     # else:
19     #     having_ip = -1 #legitimate
20     #return(having_ip)
21     return 0
22
23
24 def url_length(url):
25     length=len(url)
26     if(length<54):
27         return -1
28     elif(54<length<=75):
29         return 0
30     else:
31         return 1
32
33
```



```
33
34 def url_short(url):
35     #ongoing
36     return 0
37
38 def having_at_symbol(url):
39     symbol=regex.findall(r'@',url)
40     if(len(symbol)==0):
41         return -1
42     else:
43         return 1
44
45 def doubleSlash(url):
46     #ongoing
47     return 0
48
49 def prefix_suffix(url):
50     subDomain, domain, suffix = extract(url)
51     if(domain.count('.')==0):
52         return 1
53     else:
54         return -1
55
56 def sub_domain(url):
57     subDomain, domain, suffix = extract(url)
58     if(subDomain.count('.')==0):
59         return -1
60     elif(subDomain.count('.')==1):
61         return 0
62     else:
63         return 1
64
```

```
File Edit Selection View Go Run Terminal Help
inputScript.py - Visual Studio Code

app.py 2 inputScript.py 4 X
C:\Users> Thaarani > Desktop > project > flask > assets > templates > inputScript.py > ...

64
65 def SSIfinal_State(url):
66     try:
67         #check wheather contains https
68         if(regex.search('https',url)):
69             usehttps = 1
70         else:
71             usehttps = 0
72         #getting the certificate issuer to later compare with trusted issuer
73         #getting host name
74         subDomain, domain, suffix = extract(url)
75         host_name = domain + "." + suffix
76         context = ssl.create_default_context()
77         sct = context.wrap_socket(socket.socket(), server_hostname = host_name)
78         sct.connect((host_name, 443))
79         certificate = sct.getpeercert()
80         issuer = dict(x[0] for x in certificate['issuer'])
81         certificate_Auth = str(issuer['commonName'])
82         certificate_Auth = certificate_Auth.split()
83         if(certificate_Auth[0] == "Network" or certificate_Auth == "Deutsche"):
84             certificate_Auth = certificate_Auth[0] + " " + certificate_Auth[1]
85         else:
86             certificate_Auth = certificate_Auth[0]
87         trusted_Auth = ['Comodo','Symantec','GoDaddy','GlobalSign','DigiCert','StartCom','Entrust','Verizon','Trustwave','Unizeto','Buypass','Quo
88         #getting age of certificate
89         startingDate = str(certificate['notBefore'])
90         endingDate = str(certificate['notAfter'])
91         startingYear = int(startingDate.split()[3])
92         endingYear = int(endingDate.split()[3])
93         Age_of_certificate = endingYear-startingYear
94
```

```
File Edit Selection View Go Run Terminal Help
inputScript.py - Visual Studio Code

app.py 2 inputScript.py 4 X
C:\Users> Thaarani > Desktop > project > flask > assets > templates > inputScript.py > ...

93         Age_of_certificate = endingYear-startingYear
94
95     #checking final conditions
96     if((usehttps==1) and (certificate_Auth in trusted_Auth) and (Age_of_certificate>=1)):
97         return -1 #legitimate
98     elif((usehttps==1) and (certificate_Auth not in trusted_Auth)):
99         return 0 #suspicious
100     else:
101         return 1 #phishing
102
103     except Exception as e:
104
105         return 1
106
107 def domain_registration(url):
108     try:
109         w = whois.whois(url)
110         updated = w.updated_date
111         exp = w.expiration_date
112         length = (exp[0]-updated[0]).days
113         if(length<=365):
114             return 1
115         else:
116             return -1
117     except:
118         return 0
119
120 def favicon(url):
121     #ongoing
122     return 0
123
124 def port(url):
125     #ongoing
126     return 0

IDLE (Python 3.8 32-bit)
```

```
File Edit Selection View Go Run Terminal Help inputScript.py - Visual Studio Code
inputScript.py 4 X
C:\Users> Thaarani > Desktop > project > flask > assets > templates > inputScript.py > ...
127
128 def https_token(url):
129     subDomain, domain, suffix = extract(url)
130     host = subDomain + '.' + domain + '.' + suffix
131     if(host.count('https')): #attacker can trick by putting https in domain part
132         return 1
133     else:
134         return -1
135
136 def request_url(url):
137     try:
138         subDomain, domain, suffix = extract(url)
139         websiteDomain = domain
140
141         opener = urllib.request.urlopen(url).read()
142         soup = BeautifulSoup(opener, 'lxml')
143         imgs = soup.findAll('img', src=True)
144         total = len(imgs)
145
146         linked_to_same = 0
147         avg = 0
148         for image in imgs:
149             subDomain, domain, suffix = extract(image['src'])
150             imageDomain = domain
151             if(websiteDomain==imageDomain or imageDomain==''):
152                 linked_to_same = linked_to_same + 1
153         vids = soup.findAll('video', src=True)
154         total = total + len(vids)
155
156         for video in vids:
157             subDomain, domain, suffix = extract(video['src'])
158             vidDomain = domain
159             if(websiteDomain==vidDomain or vidDomain==''):
160                 linked_to_same = linked_to_same + 1
```

```
File Edit Selection View Go Run Terminal Help inputScript.py - Visual Studio Code
inputScript.py 4 X
C:\Users> Thaarani > Desktop > project > flask > assets > templates > inputScript.py > ...
160         linked_to_same = linked_to_same + 1
161     linked_outside = total-linked_to_same
162     if(total!=0):
163         avg = linked_outside/total
164
165     if(avg<0.22):
166         return -1
167     elif(0.22<=avg<=0.61):
168         return 0
169     else:
170         return 1
171 except:
172     return 0
173
174
175 def url_of_anchor(url):
176     try:
177         subDomain, domain, suffix = extract(url)
178         websiteDomain = domain
179
180         opener = urllib.request.urlopen(url).read()
181         soup = BeautifulSoup(opener, 'lxml')
182         anchors = soup.findAll('a', href=True)
183         total = len(anchors)
184         linked_to_same = 0
185         avg = 0
186         for anchor in anchors:
187             subDomain, domain, suffix = extract(anchor['href'])
188             anchorDomain = domain
189             if(websiteDomain==anchorDomain or anchorDomain==''):
190                 linked_to_same = linked_to_same + 1
191         linked_outside = total-linked_to_same
192         if(total!=0):
193             avg = linked_outside/total
```

```
File Edit Selection View Go Run Terminal Help
inputScript.py - Visual Studio Code

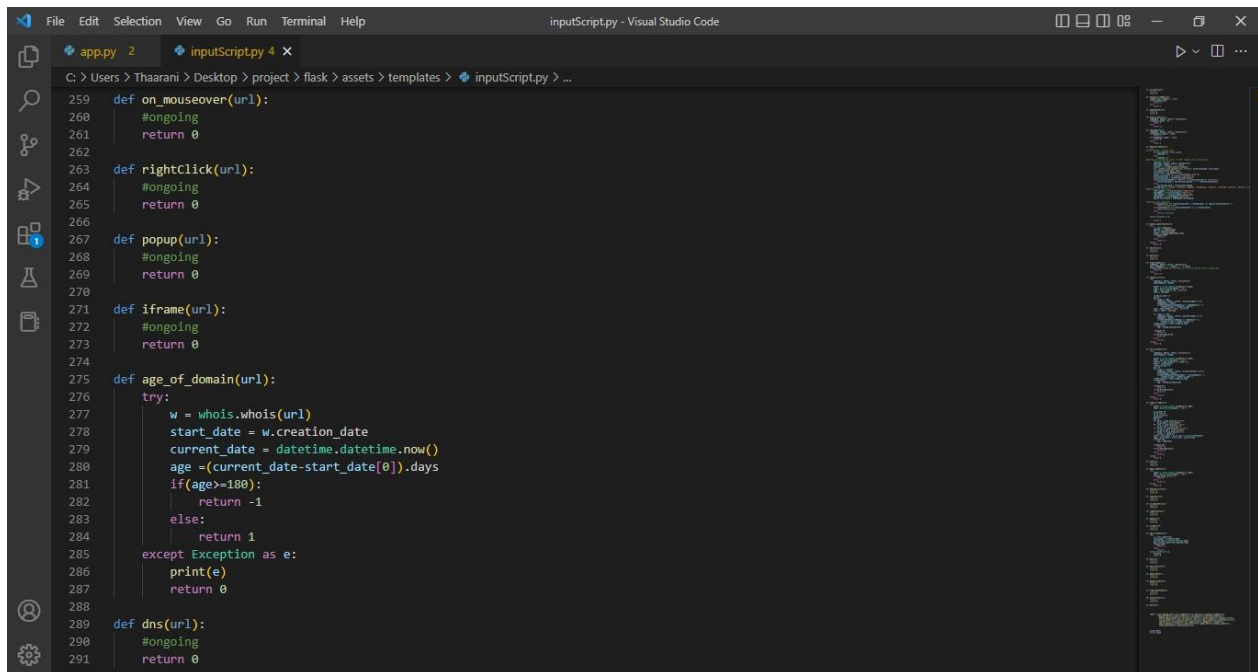
app.py 2 inputScript.py 4 X
C:\Users> Thaarani > Desktop > project > flask > assets > templates > inputScript.py > ...

194
195     if(avg<0.31):
196         return -1
197     elif(0.31<=avg<=0.67):
198         return 0
199     else:
200         return 1
201 except:
202     return 0
203
204 def Links_in_tags(url):
205     try:
206         opener = urllib.request.urlopen(url).read()
207         soup = BeautifulSoup(opener, 'lxml')
208
209         no_of_meta =0
210         no_of_link =0
211         no_of_script =0
212         anchors=0
213         avg =0
214         for meta in soup.find_all('meta'):
215             no_of_meta = no_of_meta+1
216         for link in soup.find_all('link'):
217             no_of_link = no_of_link +1
218         for script in soup.find_all('script'):
219             no_of_script = no_of_script+1
220         for anchor in soup.find_all('a'):
221             anchors = anchors+1
222         total = no_of_meta + no_of_link + no_of_script+anchors
223         tags = no_of_meta + no_of_link + no_of_script
224         if(total!=0):
225             avg = tags/total
226
227     if(avg<0.25):
```

```
File Edit Selection View Go Run Terminal Help
inputScript.py - Visual Studio Code

app.py 2 inputScript.py 4 X
C:\Users> Thaarani > Desktop > project > flask > assets > templates > inputScript.py > ...

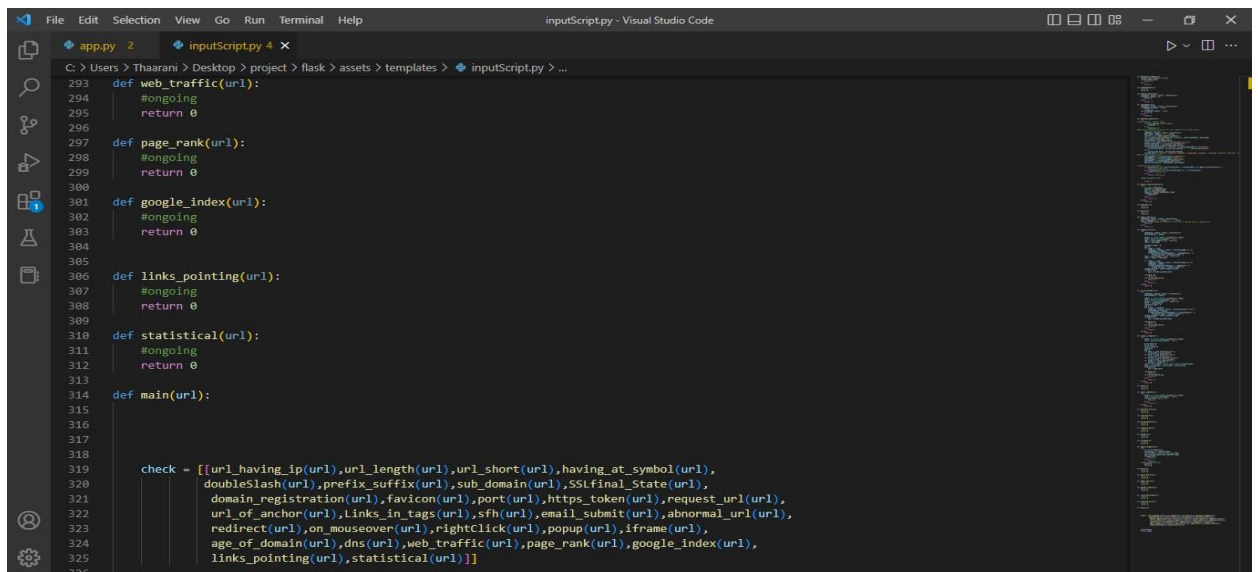
227     if(avg<0.25):
228         return -1
229     elif(0.25<=avg<=0.81):
230         return 0
231     else:
232         return 1
233 except:
234     return 0
235
236 def sfh(url):
237     #ongoing
238     return 0
239
240 def email_submit(url):
241     try:
242         opener = urllib.request.urlopen(url).read()
243         soup = BeautifulSoup(opener, 'lxml')
244         if(soup.find('mailto:')):
245             return 1
246         else:
247             return -1
248     except:
249         return 0
250
251 def abnormal_url(url):
252     #ongoing
253     return 0
254
255 def redirect(url):
256     #ongoing
257     return 0
258
259 def on_mouseover(url):
```



```
File Edit Selection View Go Run Terminal Help
inputScript.py - Visual Studio Code

app.py 2 inputScript.py 4 x
C:\Users> Thaarani > Desktop > project > flask > assets > templates > inputScript.py > ...

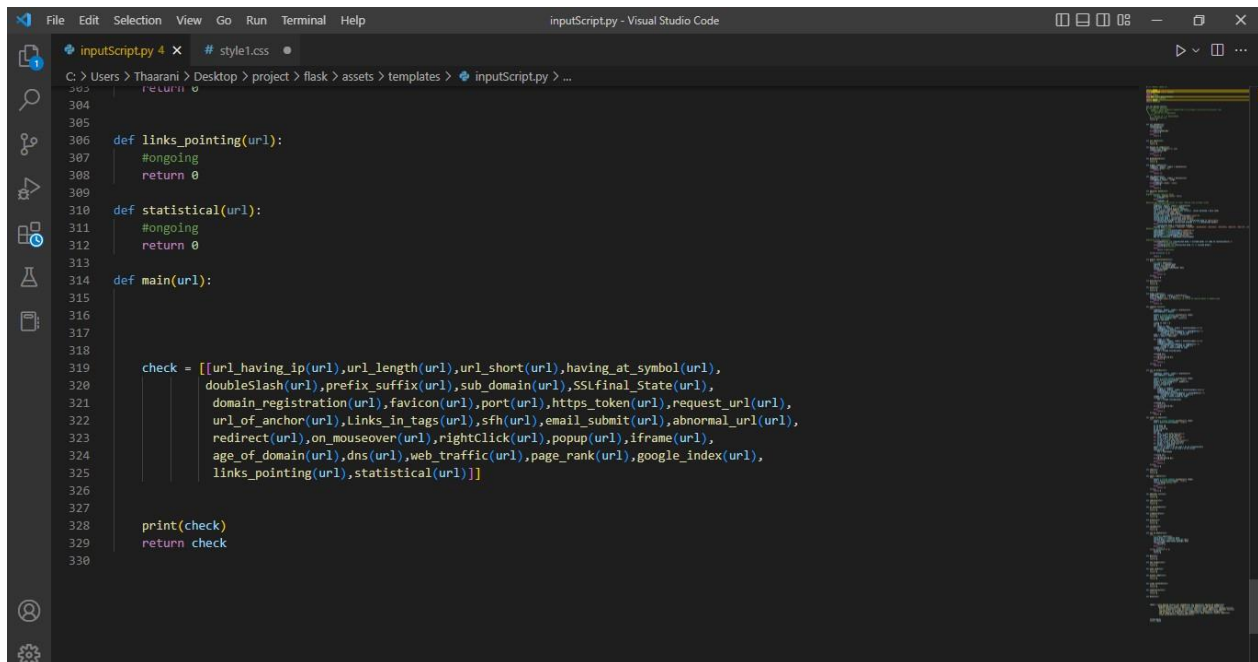
259 def on_mouseover(url):
260     #ongoing
261     return 0
262
263 def rightClick(url):
264     #ongoing
265     return 0
266
267 def popup(url):
268     #ongoing
269     return 0
270
271 def iframe(url):
272     #ongoing
273     return 0
274
275 def age_of_domain(url):
276     try:
277         w = whois.whois(url)
278         start_date = w.creation_date
279         current_date = datetime.datetime.now()
280         age =(current_date-start_date[0]).days
281         if(age>=180):
282             return -1
283         else:
284             return 1
285     except Exception as e:
286         print(e)
287         return 0
288
289 def dns(url):
290     #ongoing
291     return 0
292
```



```
File Edit Selection View Go Run Terminal Help
inputScript.py - Visual Studio Code

app.py 2 inputScript.py 4 x
C:\Users> Thaarani > Desktop > project > flask > assets > templates > inputScript.py > ...

293 def web_traffic(url):
294     #ongoing
295     return 0
296
297 def page_rank(url):
298     #ongoing
299     return 0
300
301 def google_index(url):
302     #ongoing
303     return 0
304
305
306 def links_pointing(url):
307     #ongoing
308     return 0
309
310 def statistical(url):
311     #ongoing
312     return 0
313
314 def main(url):
315
316
317
318
319     check = [[url_having_ip(url),url_length(url),url_short(url),having_at_symbol(url),
320 doubleSlash(url),prefix_suffix(url),sub_domain(url),SSLfinal_State(url),
321 domain_registration(url),favicon(url),port(url),https_token(url),request_url(url),
322 url_of_anchor(url),links_in_tags(url),sfh(url),email_submit(url),abnormal_url(url),
323 redirect(url),on_mouseover(url),rightClick(url),popup(url),iframe(url),
324 age_of_domain(url),dns(url),web_traffic(url),page_rank(url),google_index(url),
325 links_pointing(url),statistical(url)]]
326
```

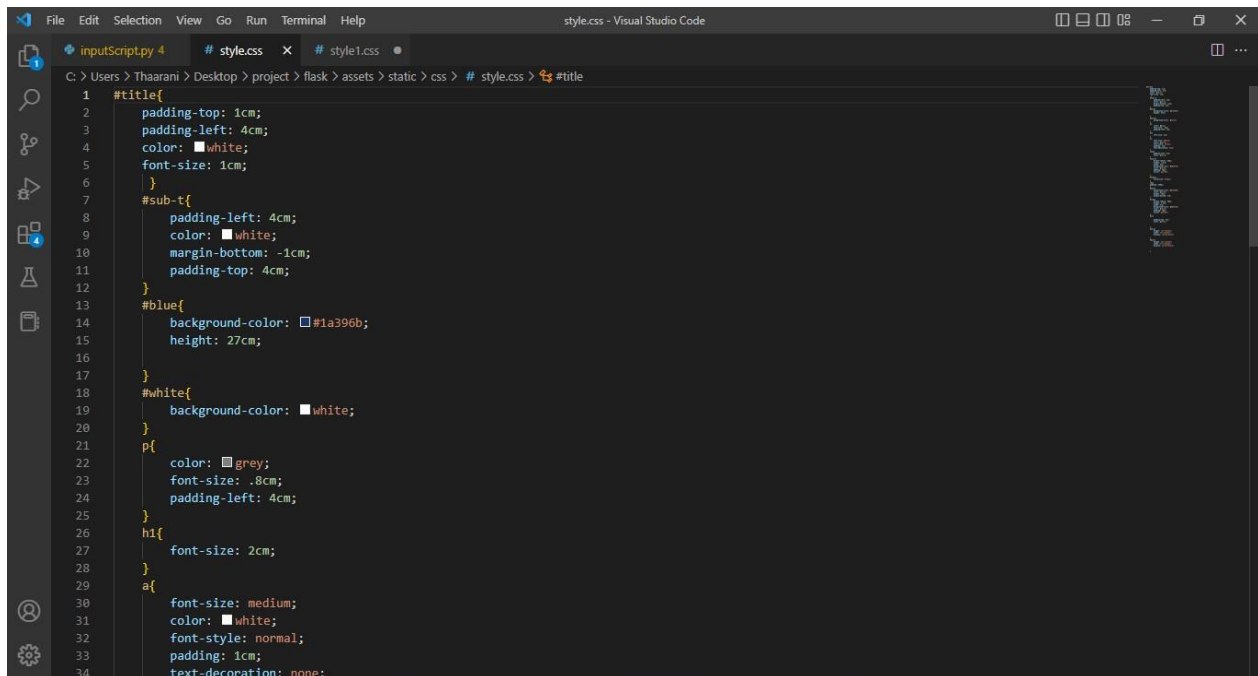



```
303     return 0
304
305
306 def links_pointing(url):
307     #ongoing
308     return 0
309
310 def statistical(url):
311     #ongoing
312     return 0
313
314 def main(url):
315
316
317
318
319     check = [[url_having_ip(url),url_length(url),url_short(url),having_at_symbol(url),
320 doubleSlash(url),prefix_suffix(url),sub_domain(url),SSLfinal_State(url),
321 domain_registration(url),favicon(url),port(url),https_token(url),request_url(url),
322 url_of_anchor(url),Links_in_tags(url),sfh(url),email_submit(url),abnormal_url(url),
323 redirect(url),on_mouseover(url),rightClick(url),popup(url),iframe(url),
324 age_of_domain(url),dns(url),web_traffic(url),page_rank(url),google_index(url),
325 links_pointing(url),statistical(url)]]
326
327
328     print(check)
329     return check
330
```

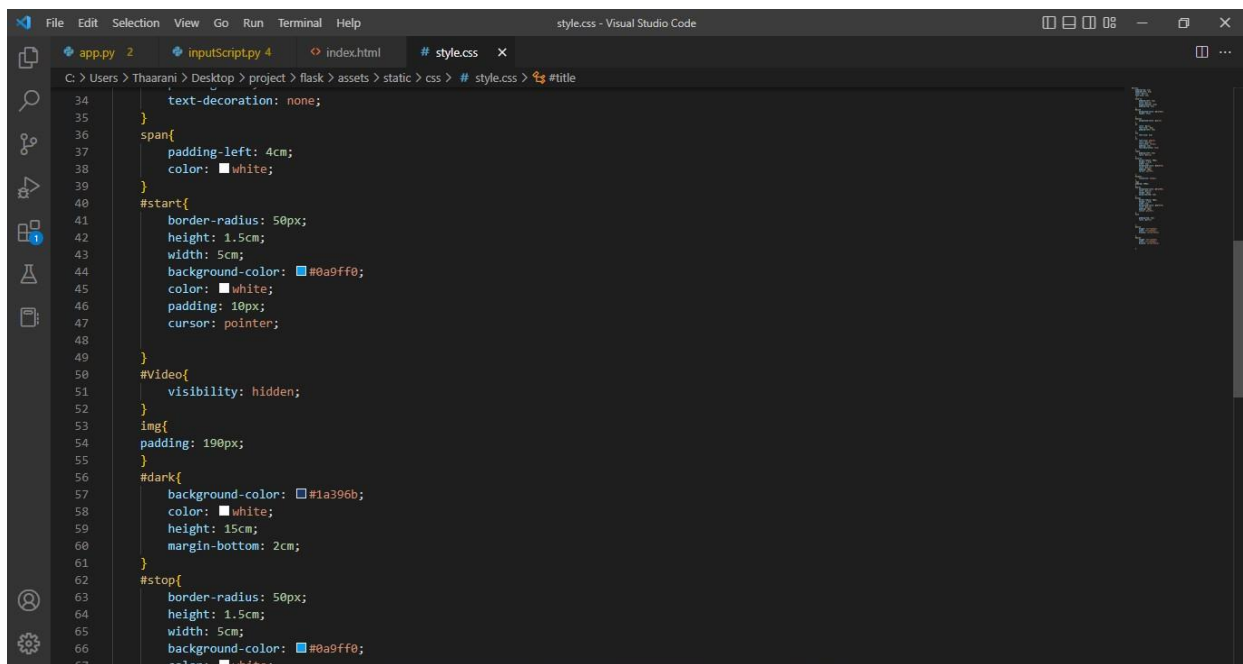
Build An HTML Page

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.

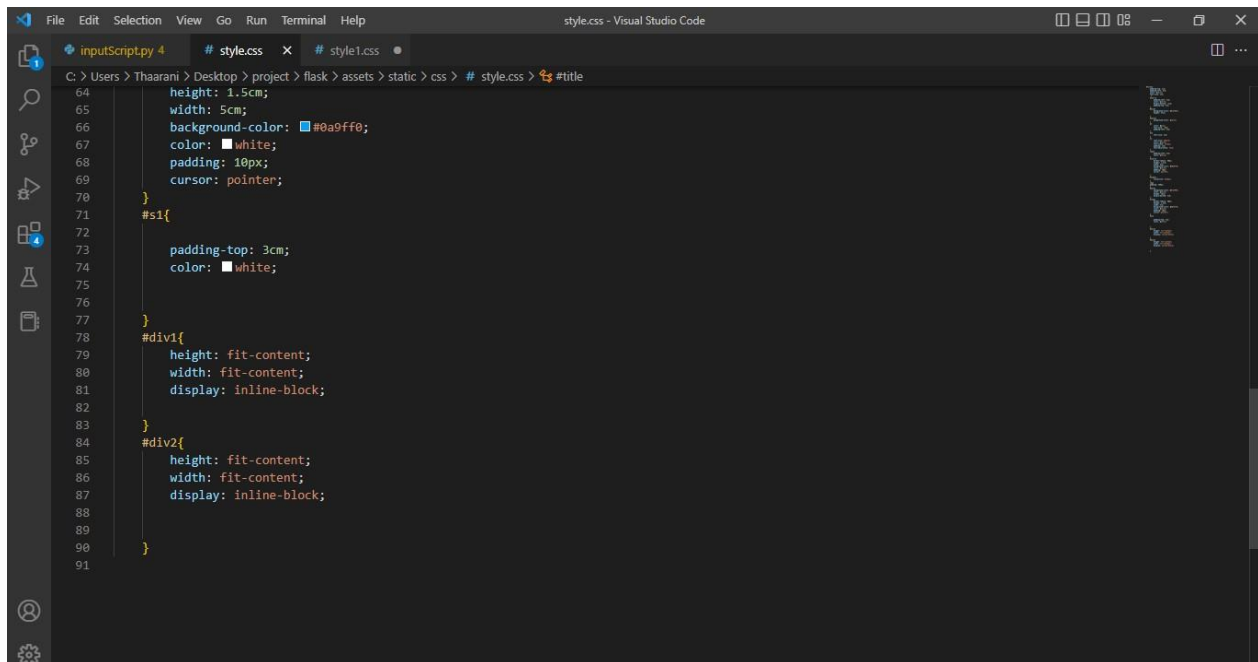
here we first built a css file called style.css file



```
1 #title{
2   padding-top: 1cm;
3   padding-left: 4cm;
4   color: white;
5   font-size: 1cm;
6 }
7 #sub-t{
8   padding-left: 4cm;
9   color: white;
10  margin-bottom: -1cm;
11  padding-top: 4cm;
12 }
13 #blue{
14   background-color: #1a396b;
15   height: 27cm;
16 }
17
18 #white{
19   background-color: white;
20 }
21 p{
22   color: grey;
23   font-size: .8cm;
24   padding-left: 4cm;
25 }
26 h1{
27   font-size: 2cm;
28 }
29 a{
30   font-size: medium;
31   color: white;
32   font-style: normal;
33   padding: 1cm;
34   text-decoration: none;
```

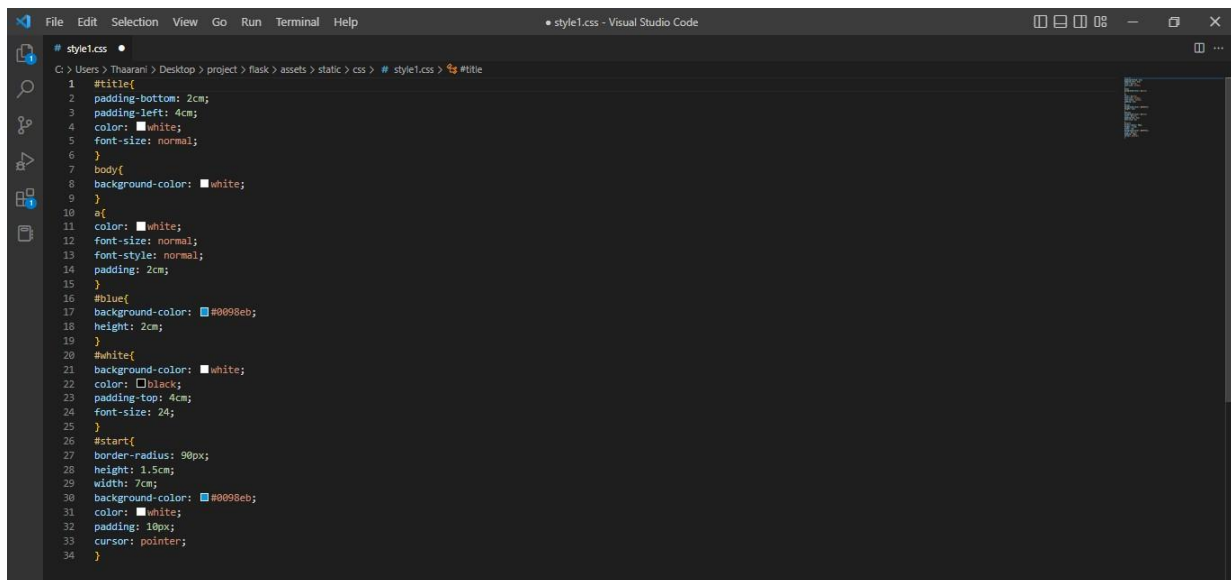


```
34   text-decoration: none;
35 }
36 span{
37   padding-left: 4cm;
38   color: white;
39 }
40 #start{
41   border-radius: 50px;
42   height: 1.5cm;
43   width: 5cm;
44   background-color: #0a9ff0;
45   color: white;
46   padding: 10px;
47   cursor: pointer;
48 }
49
50 #Video{
51   visibility: hidden;
52 }
53 img{
54   padding: 190px;
55 }
56 #dark{
57   background-color: #1a396b;
58   color: white;
59   height: 15cm;
60   margin-bottom: 2cm;
61 }
62 #stop{
63   border-radius: 50px;
64   height: 1.5cm;
65   width: 5cm;
66   background-color: #0a9ff0;
```



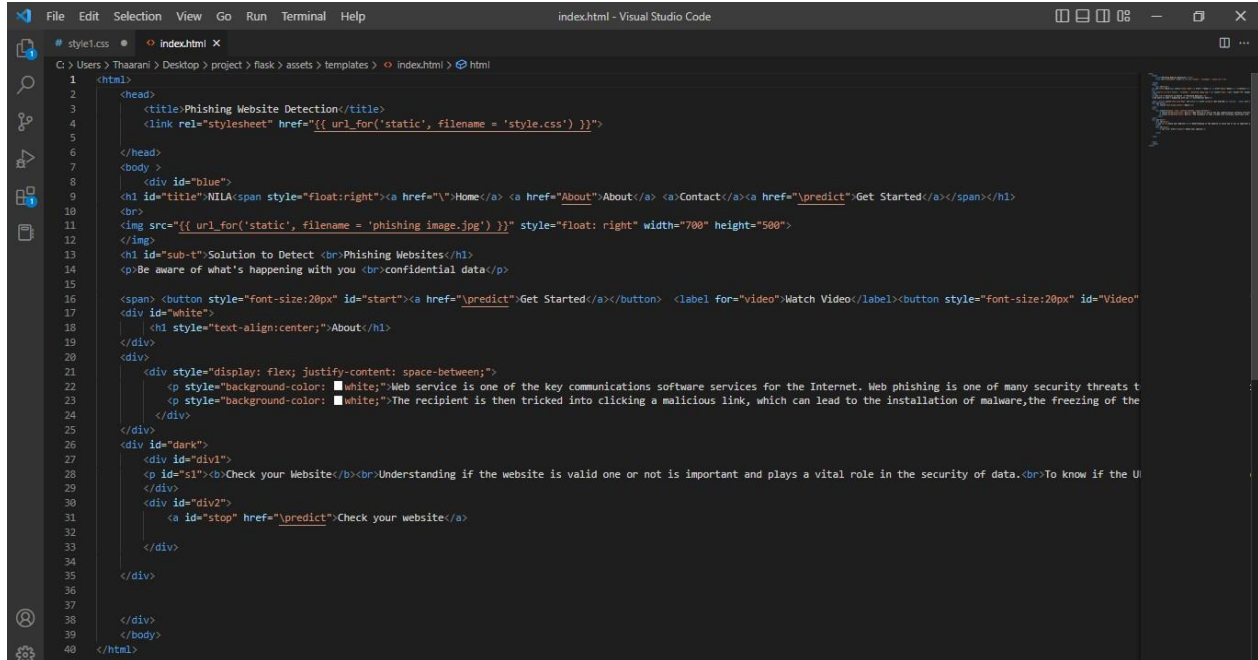
```
64 height: 1.5cm;
65 width: 5cm;
66 background-color: #0a9ff0;
67 color: white;
68 padding: 10px;
69 cursor: pointer;
70
71 #s1{
72
73 padding-top: 3cm;
74 color: white;
75
76
77 }
78 #div1{
79 height: fit-content;
80 width: fit-content;
81 display: inline-block;
82
83 }
84 #div2{
85 height: fit-content;
86 width: fit-content;
87 display: inline-block;
88
89 }
90
91 }
```

style1.css



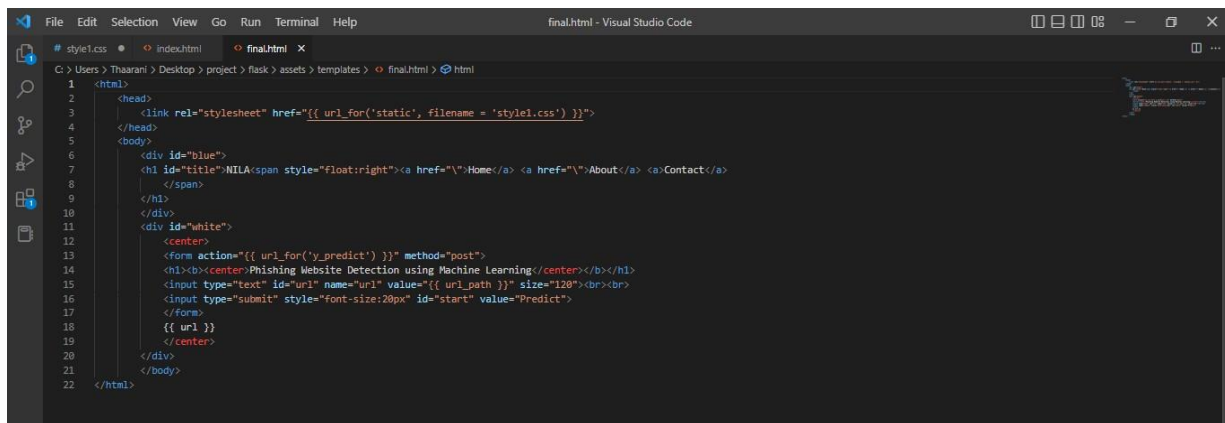
```
1 #title{
2 padding-bottom: 2cm;
3 padding-left: 4cm;
4 color: white;
5 font-size: normal;
6 font-style: normal;
7 }
8 body{
9 background-color: white;
10 }
11 #blue{
12 color: white;
13 font-size: normal;
14 font-style: normal;
15 padding: 2cm;
16 }
17 #blue{
18 background-color: #0098eb;
19 height: 2cm;
20 width: 7cm;
21 }
22 #white{
23 background-color: white;
24 color: black;
25 padding-top: 4cm;
26 font-size: 24;
27 font-style: normal;
28 }
29 #start{
30 border-radius: 90px;
31 height: 1.5cm;
32 width: 7cm;
33 background-color: #0098eb;
34 color: white;
35 padding: 10px;
36 cursor: pointer;
37 }
```

now moving on to the index.html source file



```
1 <html>
2   <head>
3     <title>Phishing Website Detection</title>
4     <link rel="stylesheet" href="{{ url_for('static', filename = 'style.css') }}">
5   </head>
6   <body>
7     <div id="blue">
8       <h1 id="title">NILA<span style="float:right"><a href="/">Home</a> <a href="/about">About</a> <a href="/contact">Contact</a> <a href="/predict">Get Started</a></span></h1>
9       <br>
10      
11    </img>
12    <h1 id="sub-t">Solution to Detect <br>Phishing Websites</h1>
13    <p>Be aware of what's happening with you <br>confidential data</p>
14
15    <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">
16    <div id="white">
17      <h1 style="text-align:center">About</h1>
18    </div>
19    <div>
20      <div style="display: flex; justify-content: space-between;">
21        <p style="background-color: #000000; color: white;">Web service is one of the key communications software services for the Internet. Web phishing is one of many security threats t
22        <p style="background-color: #000000; color: white;">The recipient is then tricked into clicking a malicious link, which can lead to the installation of malware,the freezing of the
23      </div>
24    </div>
25    <div id="dark">
26      <div id="div1">
27        <p id="s1"><b>Check your Website</b><br>Understanding if the website is a valid one or not is important and plays a vital role in the security of data.<br>To know if the U
28      </div>
29      <div id="div2">
30        <a id="stop" href="/predict">Check your website</a>
31      </div>
32    </div>
33  </div>
34  </div>
35  </div>
36  </div>
37  </div>
38  </div>
39  </div>
40 </html>
```

now final.html source file

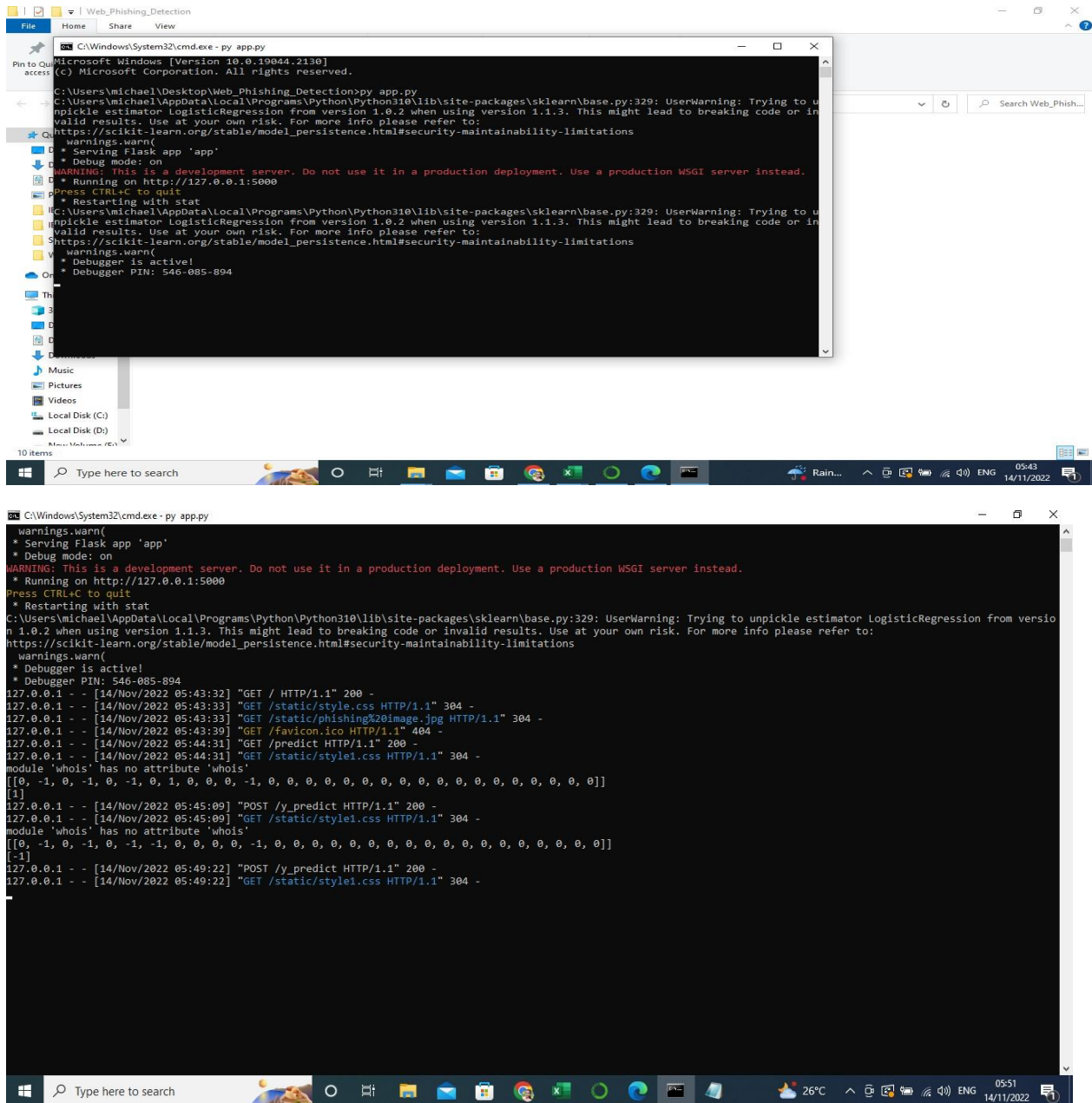


```
1 <html>
2   <head>
3     <link rel="stylesheet" href="{{ url_for('static', filename = 'style1.css') }}">
4   </head>
5   <body>
6     <div id="blue">
7       <h1 id="title">NILA<span style="float:right"><a href="/">Home</a> <a href="/about">About</a> <a href="/contact">Contact</a>
8       </span>
9     </h1>
10    </div>
11    <div id="white">
12      <center>
13        <form action="{{ url_for('y_predict') }}" method="post">
14          <h1><b><center>Phishing Website Detection using Machine Learning</center></b></h1>
15          <input type="text" id="url" name="url" value="{{ url_path }}" size="120"><br>
16          <input type="submit" style="font-size:20px" id="start" value="Predict">
17        </form>
18        <div>
19          <div id="div1">
20            <p id="s1"><b>Check your Website</b><br>Understanding if the website is a valid one or not is important and plays a vital role in the security of data.<br>To know if the U
21          </div>
22          <div id="div2">
23            <a id="stop" href="/predict">Check your website</a>
24          </div>
25        </div>
26      </center>
27    </div>
28  </div>
29  </div>
30  </div>
31  </div>
32  </div>
33  </div>
34  </div>
35  </div>
36  </div>
37  </div>
38  </div>
39  </div>
40 </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.

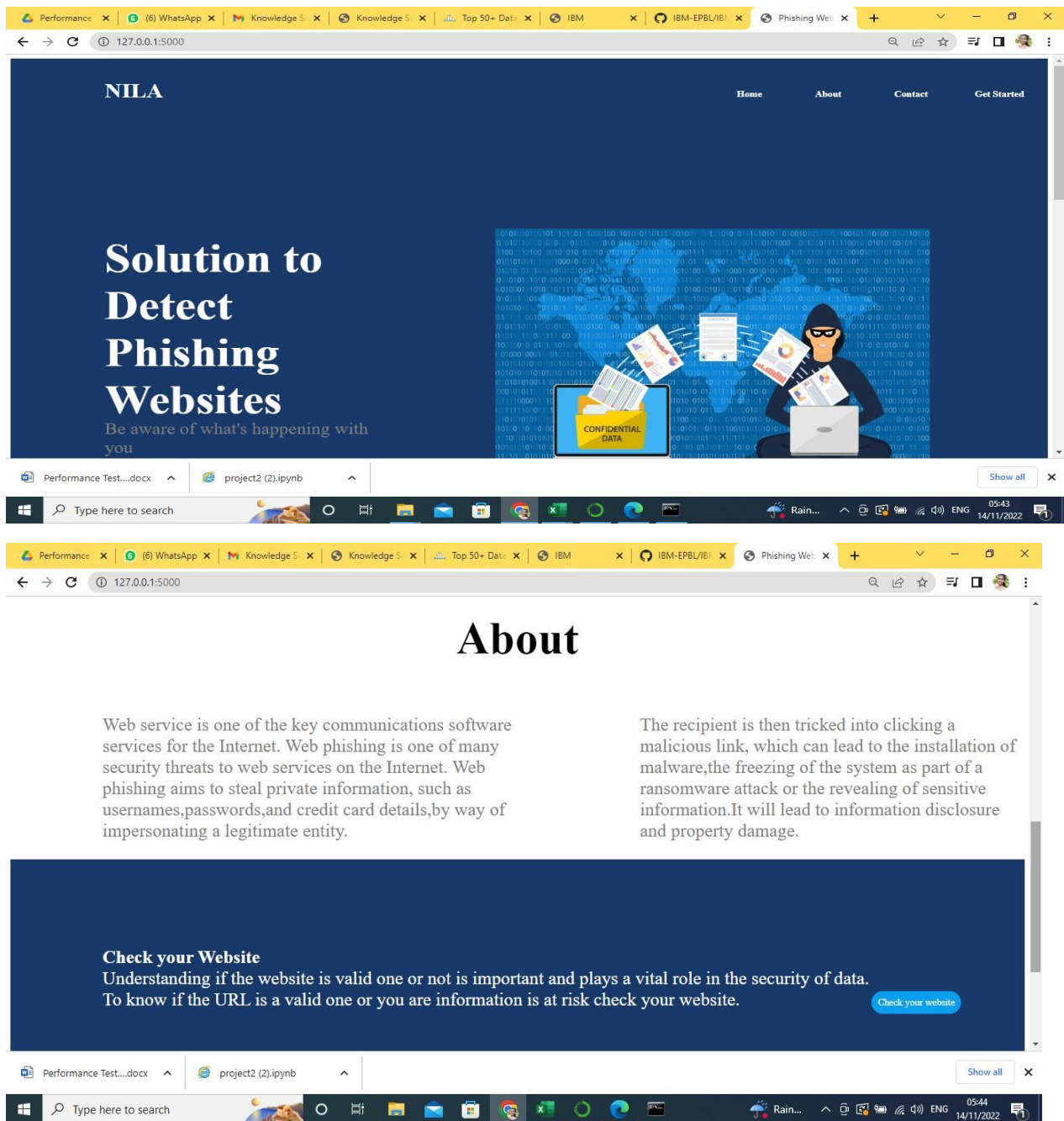


```
C:\Windows\System32\cmd.exe - py app.py
Microsoft Windows [Version 10.0.19044.2130]
(c) Microsoft Corporation. All rights reserved.

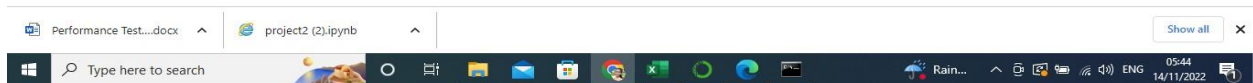
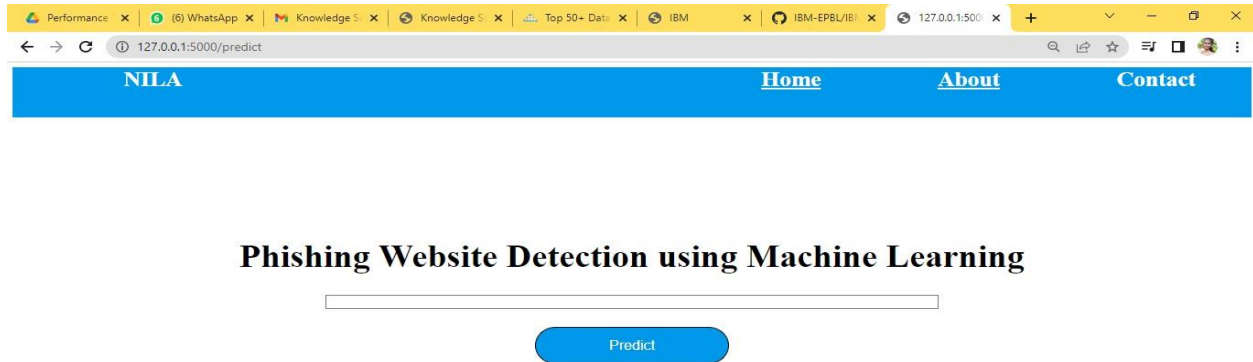
C:\Users\michael\Desktop\Web_Phishing_Detection>py app.py
C:\Users\michael\AppData\Local\Programs\Python\Python310\lib\site-packages\sklearn\base.py:329: UserWarning: Trying to unpickle estimator LogisticRegression from version 1.0.2 when using version 1.1.3. This might lead to breaking code or invalid results. Use at your own risk. For more info please refer to:
https://scikit-learn.org/stable/model_persistence.html#security-maintainability-limitations
  warnings.warn(
 * Serving Flask app 'app'
 * Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on http://127.0.0.1:5000
Press CTRL+C to quit
 * Restarting with stat
C:\Users\michael\AppData\Local\Programs\Python\Python310\lib\site-packages\sklearn\base.py:329: UserWarning: Trying to unpickle estimator LogisticRegression from version 1.0.2 when using version 1.1.3. This might lead to breaking code or invalid results. Use at your own risk. For more info please refer to:
https://scikit-learn.org/stable/model_persistence.html#security-maintainability-limitations
  warnings.warn(
 * Debugger is active!
 * Debugger PIN: 546-085-894

C:\Windows\System32\cmd.exe - py app.py
 * Serving Flask app 'app'
 * Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on http://127.0.0.1:5000
Press CTRL+C to quit
 * Restarting with stat
C:\Users\michael\AppData\Local\Programs\Python\Python310\lib\site-packages\sklearn\base.py:329: UserWarning: Trying to unpickle estimator LogisticRegression from version 1.0.2 when using version 1.1.3. This might lead to breaking code or invalid results. Use at your own risk. For more info please refer to:
https://scikit-learn.org/stable/model_persistence.html#security-maintainability-limitations
  warnings.warn(
 * Debugger is active!
 * Debugger PIN: 546-085-894
127.0.0.1 - - [14/Nov/2022 05:43:32] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [14/Nov/2022 05:43:33] "GET /static/style.css HTTP/1.1" 304 -
127.0.0.1 - - [14/Nov/2022 05:43:33] "GET /static/phishing20image.jpg HTTP/1.1" 304 -
127.0.0.1 - - [14/Nov/2022 05:43:39] "GET /favicon.ico HTTP/1.1" 404 -
127.0.0.1 - - [14/Nov/2022 05:44:31] "GET /predict HTTP/1.1" 200 -
127.0.0.1 - - [14/Nov/2022 05:44:31] "GET /static/style1.css HTTP/1.1" 304 -
module 'whois' has no attribute 'whois'
[[0, -1, 0, -1, 0, -1, 0, 1, 0, 0, 0, 0, -1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]]
[1]
127.0.0.1 - - [14/Nov/2022 05:45:09] "POST /y_predict HTTP/1.1" 200 -
127.0.0.1 - - [14/Nov/2022 05:45:09] "GET /static/style1.css HTTP/1.1" 304 -
module 'whois' has no attribute 'whois'
[[0, -1, 0, -1, 0, -1, -1, 0, 0, 0, 0, 0, -1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]]
[-1]
127.0.0.1 - - [14/Nov/2022 05:49:22] "POST /y_predict HTTP/1.1" 200 -
127.0.0.1 - - [14/Nov/2022 05:49:22] "GET /static/style1.css HTTP/1.1" 304 -
```

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

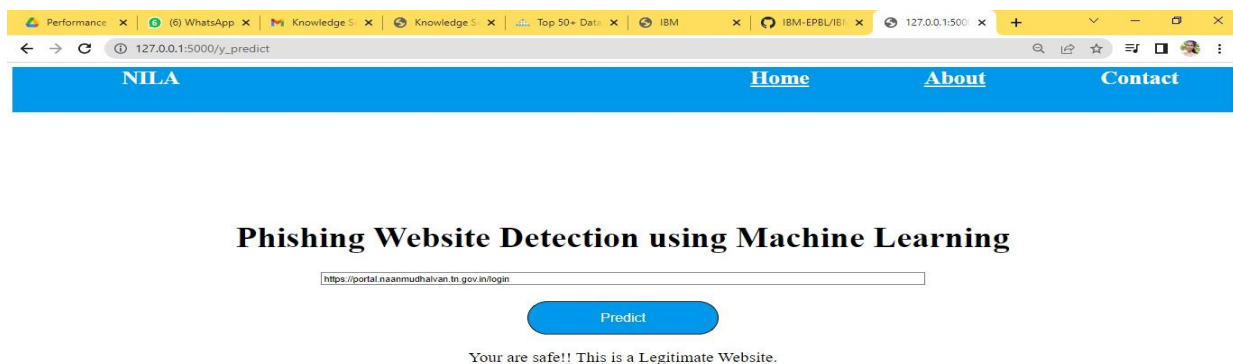


Here we have two options to predict the website one is we can click on check your website or you can click on get started to proceed your website prediction the display screen is as follows



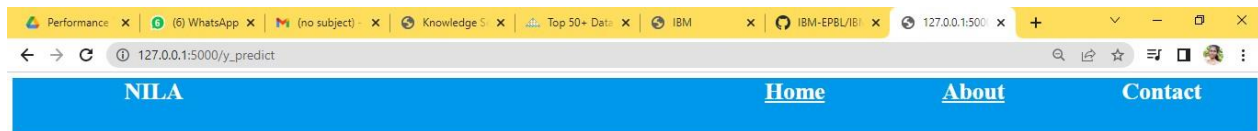
PREDICTION OF SAFE WEBSITE:

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!"**



Phishing Website Detection using Machine Learning

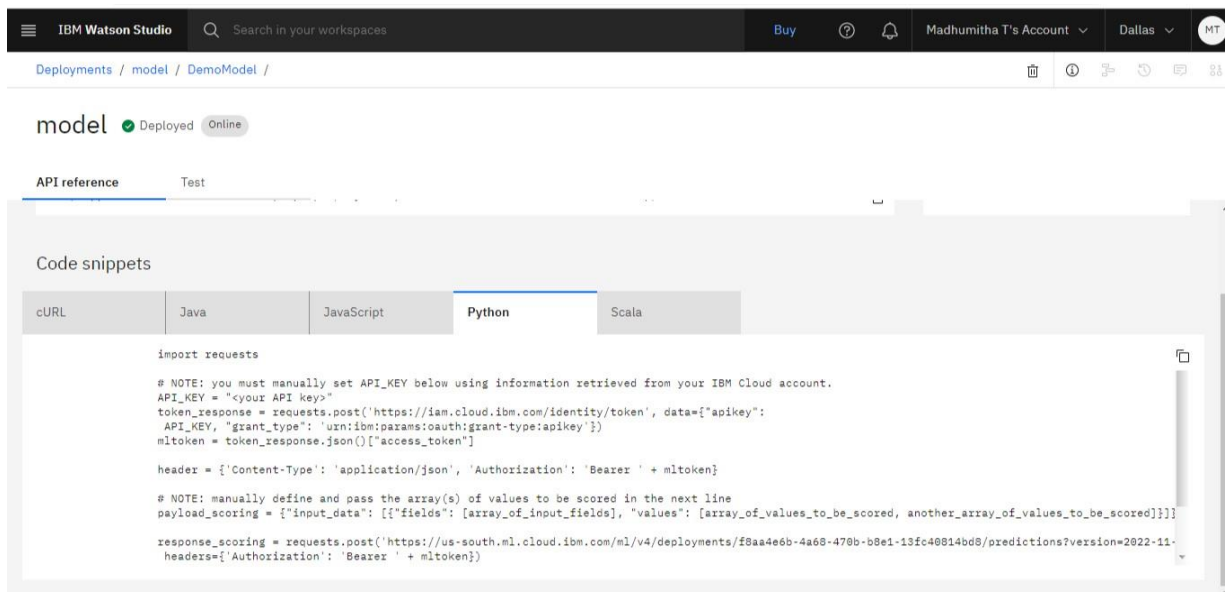
<https://s.hovstuffworks.com/t/bedroom>

Predict

You are on the wrong site. Be cautions!



IBM CLOUD DEPLOYMENT:



IBM Watson Studio

Search in your workspaces

Buy

Madhumitha T's Account

Dallas

MT

Deployments /

model

OverviewAssetsDeploymentsJobsManage

Search

Name	Type	Status	Asset	Last modified	
model	Online	Deployed	DemoModel	10 minutes ago Madhumitha T (You)	

Items per page: 201-1 of 1 items

1 of 1 pages

Drop files here or browse for files to upload.

Stay on the page until upload completes. Incomplete uploads are cancelled.

SCORING ENDPOINTS IN IBM CLOUD:

IBM Watson Studio

Search in your workspaces

Buy

Madhumitha T's Account

Dallas

MT

Projects / web phishing detection / SCORING ENDPOINT IN IBM CL...

File Edit View Insert Cell Kernel Help

Not Trusted Python 3.9

Run

Format

Code

SCORING ENDPOINT IN IBM CLOUD

```

In [8]: import requests

# NOTE: you must manually set API_KEY below using information retrieved from your IBM Cloud account.
API_KEY = "zpsRv4398uHro5tYq8C56nQvBd_ViESitemprbuY"
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}

# NOTE: manually define and pass the array(s) of values to be scored in the next line
payload_scoring = {"input_data": [{"fields": [{"usingip", "LongURL", "ShortURL", "Symbol", "Redirecting//", "PrefixSuffix-", "Subdomains", "HTTPS", "DomainRegLen", "Favicon", "NonStdPort", "HTTPSDomainURL", "RequestURL", "An
}], "values": [[1,1,1,1,1,-1,-1,-1,-1,1,1,1,-1,-1,1,1,1,1,1,1,1,1,-1,-1,-1,-1,1,1,1]]}]

response_scoring = requests.post('https://us-south.ml.cloud.ibm.com/ml/v4/deployments/f8aa4e6b-4868-470b-b8e1-13fc486140d8/predictions?version=2022-11-18', json=payload_scoring,
headers={'Authorization': 'Bearer ' + mltoken})
print("Scoring response")

predictions_response_scoring.json()
#print(predictions)
pred_print(predictions['predictions'])[0]['values'][0][0]
if(pred != 1):
    print("The Website is not Legitimate... BEWARE!!")
else:
    print("The Website is secure.. Continue")

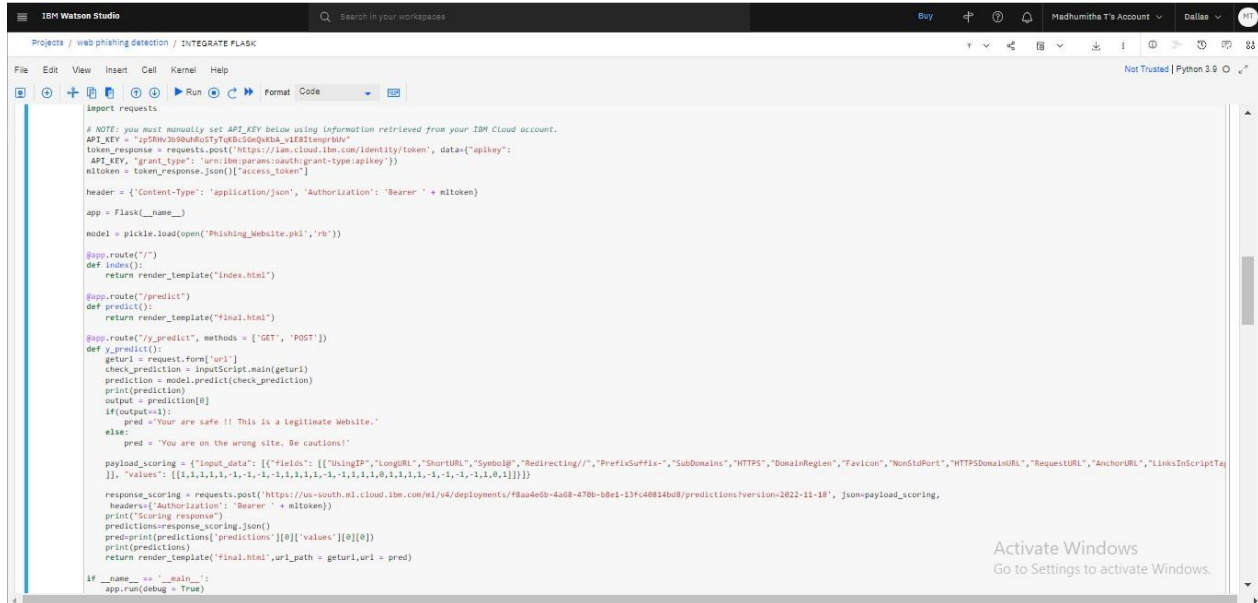
Scoring response
-1
The Website is not Legitimate... BEWARE!!

```

In []:

Activate Windows
Go to Settings to activate Windows.

INTEGRATING FLASK WITH IBM CLOUD:



```
import requests

# NOTE: you must manually set API_KEY below using information retrieved from your IBM Cloud account.
API_KEY = "xpb8w308uhk0tyt0Bic5Gqk8h_vif8t5mpt0k"
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__)

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 = "You are safe !! This is a legitimate Website."
    else:
        pred = "You are on the wrong site. Be cautious!"

    payload_scoring = {"input_data": [{"fields": [{"UsingIP", "LongURL", "ShortURL", "Symbol", "Redirecting//", "PrefixSuffix", "SubDomains", "HTTPS", "DomainReglen", "Favicon", "NonStdPort", "HTTPSDomainURL", "RequestURL", "AnchorURL", "LinksInScriptTag"}], "values": [[1,1,1,1,1,-1,-1,-1,1,1,1,-1,-1,1,1,0,1,1,1,-1,-1,-1,1,0,1]]}}
    response_scoring = requests.post("https://us-south.ml.cloud.ibm.com/ml/v4/deployments/98aade0b-4a68-470b-b861-13fc40814b08/predictions?version=2022-11-18", json=payload_scoring,
    headers={'Authorization': 'Bearer ' + mltoken})
    print("Scoring response")
    predictions=response_scoring.json()
    pred=print(predictions["predictions"][0][["values"]][0][0])
    print(predictions)
    return render_template("final.html",url_path = geturl,url = pred)

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

LINKS:

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

VIDEO LINK: [click here](#)