WEB PHISHING DETECTION

A PROJECT REPORT

Submitted by

Sanjai kumar N
Sivaprakash M
Vignesh T.P
Sathish R

of

NANDHA ENGINEERING COLLEGE

CHAPTER NO	TITLE	PAGE NO
1.	**INTRODUCTION**	
	1.1 Project Overview	4
	1.2 Purpose	4
2.	**LITERATURE SURVEY**	
	2.1 Existing problem	5
	2.2 References	5
	2.3 Problem Statement Definition	6
3.	**IDEATION & PROPOSED SOLUTION**	
	3.1 Empathy Map Canvas	7
	3.2 Ideation & Brainstorming	8
	3.3 Proposed Solution	9
	3.4 Problem Solution fit	10
4.	**REQUIREMENT ANALYSIS**	
	4.1 Functional requirement	11
	4.2 Non-Functional requirements	11
5.	**PROJECT DESIGN**	
	5.1 Data Flow Diagrams	12
	5.2 Solution & Technical Architecture	13
	5.3 User Stories	13
6.	**PROJECT PLANNING & SCHEDULING**	
	6.1 Sprint Planning & Estimation	14
	6.2 Sprint Delivery Schedule	14
	6.3 Reports from JIRA	15
7.	**CODING & SOLUTIONING**	
	7.1 Feature 1	15
	7.2 Feature 2	24

8.	**TESTING**		
		8.1 Test Cases	33
		8.2 User Acceptance Testing	35
9.	**RESULTS**		
		9.1 Performance Metrics	36
10.	**ADVANTAGES AND DISADVANTAGES**		37
11.	**CONCLUSION**		37
12.	**FUTURE SCOPE**		37
13.	**APPENDIX**		
		13.1 Source Code	38
		13.2 GitHub Link	49

CHAPTER 1
INTRODUCTION

1.1 Project Overview:

Web service is one of the most important internet communication software services. The project mainly focuses on applying a machine-learning algorithm to detect Phishing websites. In order to detect and predict the 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 algorithms applied include Logistic Regression, Support Vector Machine, Decision Tree, Naïve Bayes, Random Forest and Stacking. The 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. When the URL of the website is entered, the machine learning algorithm is used to detect whether the website is a phishing website or not.

1.2 Purpose:

There are several users who purchase products online and make payments through e-banking. There are websites that ask users to provide sensitive data such as username, password & credit card details, etc., often for malicious reasons. This type of website is known as a phishing 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.

The main purpose of the project is to detect phishing sites to improve the customer's sense of safety whenever he/she attempts to provide any sensitive information to a site. This awareness will help people to not access the phishing sites, which will reduce the revenue of malicious site owners. This application can be accessed online without paying instead, can be accessed via any browser of the customer's choice to detect any site with high accuracy.

CHAPTER 2
LITERATURE SURVEY

2.1 Existing Problem:

There are websites online that detect phishing. However, once a usage cap is reached, users are charged. A significant number of them come with a simple foundation of features. Several criteria that could be utilized to recognize a phishing site have been extensively examined and discovered by us. These elements are classified as address bar-based features, domain-based features, HTML-based features, and JavaScript-based features. These features allow us to construct an intelligent system that is highly accurate and effective at detecting phishing sites. Furthermore, it is an open-source website that will be simple for all users to use.

2.2 References:

[1] H. Huang et al., (2009) proposed the frameworks that distinguish the phishing utilizing page section similitude that breaks down universal resource locator tokens to create forecast preciseness phishing pages normally keep its CSS vogue like their objective pages.

[2] S. Singh, M. P. Singh and R. Pandey, "Phishing Detection from URLs Using Deep Learning Approach," 2020 5th International Conference on Computing, Communication and Security (ICCCS), 2020, pp. 1-4, doi: 10.1109/ICCCS49678.2020.9277459.

[3] M. Abutaha, M. Ababneh, K. Mahmoud and S. A. -H. Baddar, "URL Phishing Detection using Machine Learning Techniques based on URLs Lexical Analysis," 2021 12th International Conference on Information and Communication Systems (ICICS), 2021, pp. 147-152, doi: 10.1109/ICICS52457.2021.9464539.

[4] A. K. Singh and N. Goyal, "Detection of Malicious Webpages Using Deep Learning," 2021 IEEE International Conference on Big Data (Big Data), 2021, pp. 3370-3379, doi: 10.1109/BigData52589.2021.9671622.

[5] C. -Y. Wu, C. -C. Kuo and C. -S. Yang, "A Phishing Detection System based on Machine Learning," 2019 International Conference on Intelligent Computing and its Emerging Applications (ICEA), 2019, pp. 28-32, doi: 10.1109/ICEA.2019.8858325.

[6] A. Lakshmanarao, P. S. P. Rao and M. M. B. Krishna, "Phishing website detection using novel machine learning fusion approach," 2021 International Conference on Artificial Intelligence and Smart Systems (ICAIS), 2021, pp. 1164-1169, doi: 10.1109/ICAIS50930.2021.9395810.

[7] M. M. Yadollahi, F. Shoeleh, E. Serkani, A. Madani and H. Gharaee, "An Adaptive Machine Learning Based Approach for Phishing Detection Using Hybrid Features," 2019 5th International Conference on Web Research (ICWR), 2019, pp. 281-286, doi: 10.1109/ICWR.2019.8765265.

[8] S. -J. Bu and S. -B. Cho, "Integrating Deep Learning with First-Order Logic Programmed Constraints for Zero-Day Phishing Attack Detection," ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2021, pp. 2685-2689, doi: 10.1109/ICASSP39728.2021.9414850.

[9] L. Zhang, P. Zhang, L. Liu and J. Tan, "Multiphish: Multi-Modal Features Fusion Networks for Phishing Detection," ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2021, pp. 3520-3524, doi: 10.1109/ICASSP39728.2021.9415016.

[10] S. Yu, C. An, T. Yu, Z. Zhao, T. Li and J. Wang, "Phishing Detection Based on Multi-Feature Neural Network," 2022 IEEE International Performance, Computing, and Communications Conference (IPCCC), 2022, pp. 73-79

2.3 Problem statement definition:

Phishing is a form of social engineering assault that is frequently employed to obtain user information, such as user credentials and credit card data. It happens when an attacker deludes a victim into opening an email, instant message, or text message by disguising themselves as a reliable source. It poses a risk to numerous elements of online security, including the potential for scams and the release of sensitive data. Following are typical hazards posed by web phishing:

● Getting personal information from a person or business.

● Posing as a reliable company to distribute harmful web pages.

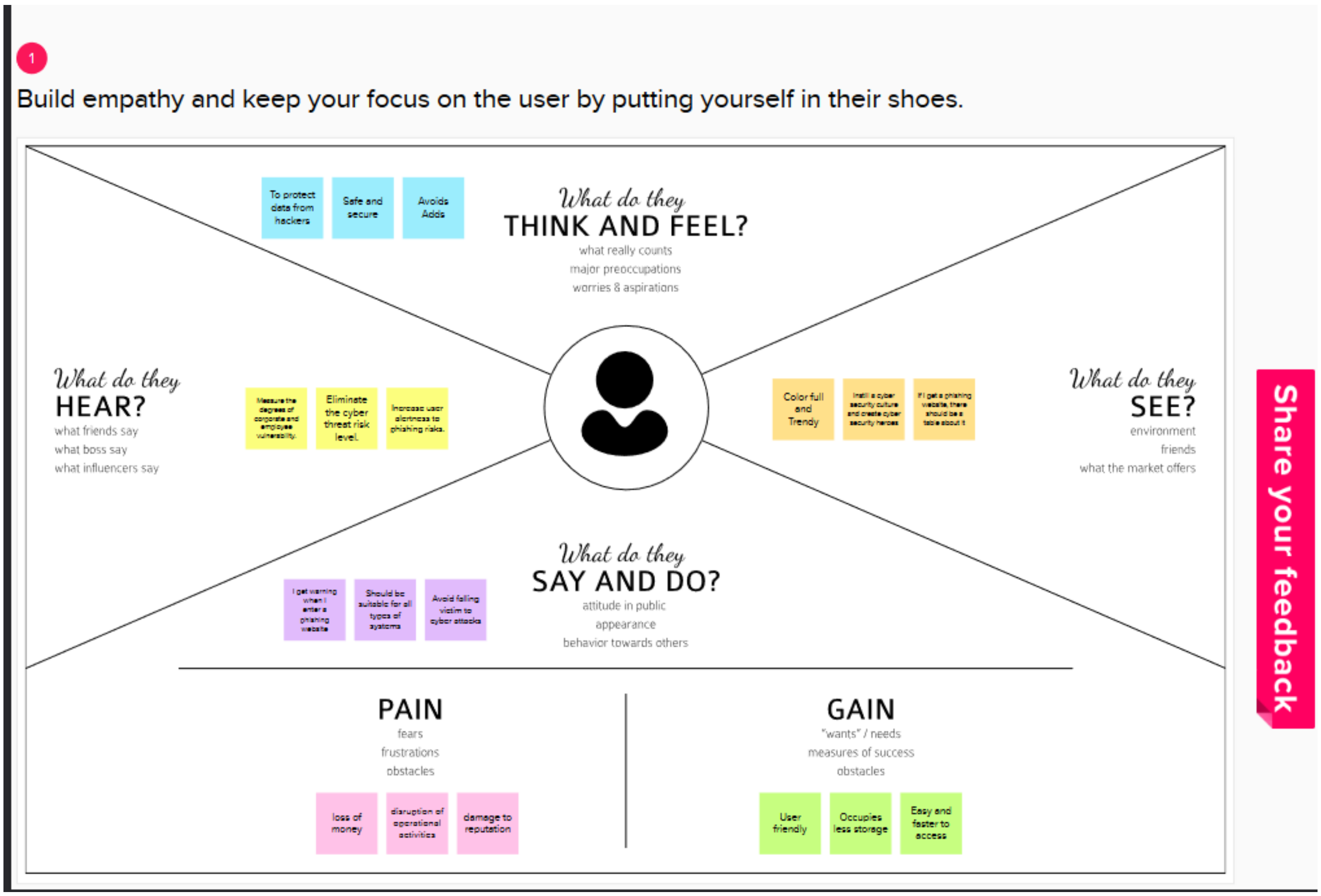
Aim is classification of a phishing website with the aid of various machine learning techniques to achieve maximum accuracy and a concise model. By implementing classification algorithms and approaches to extract the phishing datasets criteria to define their authenticity, we construct an effective and intelligent system to detect such websites in work to circumvent these dangers.

CHAPTER 3

IDEATION & PROPOSED SOLUTION

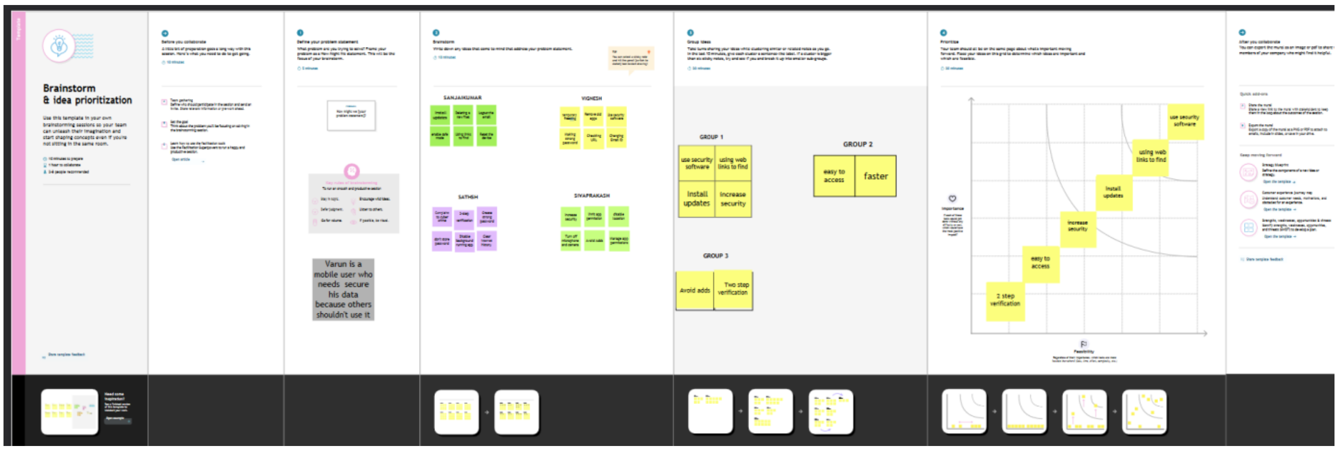
3.1 Empathy Map Canvas:

Empathy maps are useful tool that designers use to not only analyze users' behavior but also to visually convey their results to colleagues, bringing the team together around a common knowledge of the user. In user-centered design, empathy maps are best used from the very beginning of the design process.



3.2 Ideation and Brainstorming:

Ideation is a general term that refers to the process of coming up with and expressing new ideas. It is an imaginative thought that seeks to resolve a dilemma or offer a more effective means of carrying out an action. It includes creating fresh concepts, upgrading existing ones, and figuring out how to put fresh concepts into action. Brainstorming is the most frequently practiced form of ideation. The intention of brainstorming is to leverage the collective thinking of the group, by engaging with each other, listening, and building on other ideas.



3.3 Proposed Solution:

Proposed Solution Template:

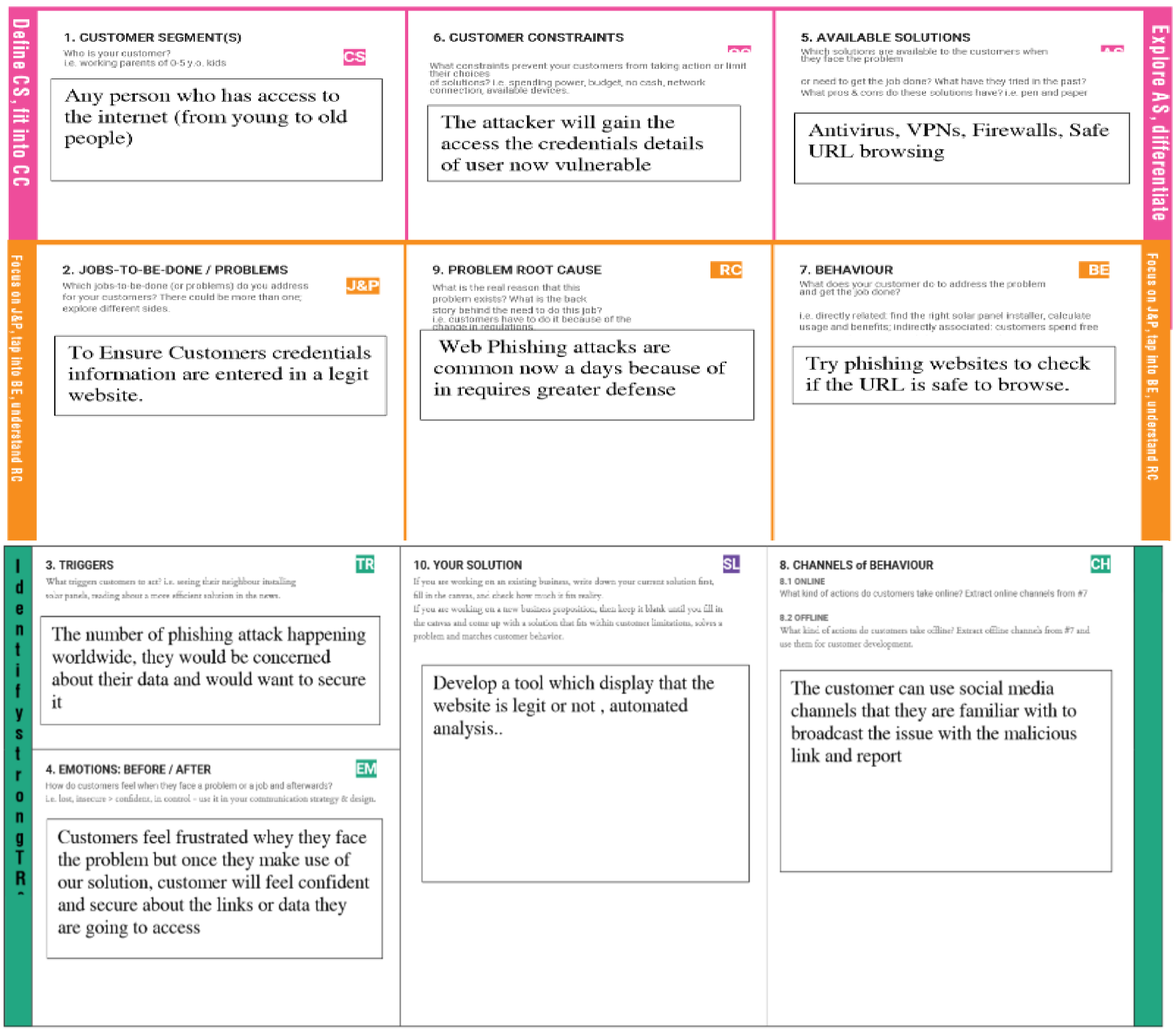
S.No.	Parameter	Description
1.	**Problem Statement (Problem to be solved)**	❖ Phishing sites are malicious websites that imitate legitimate websites or web pages and aim to steal user's personal credentials like user id, password, and financial information ❖ To reduce the people falling for web phishing scams by creating a sophisticated tool that classifies a website as malicious or safe to use
2.	**Idea / Solution description**	❖ Identify web phishing, classify whether it is an attack and prevent malicious intrusive websites.
3.	**Novelty / Uniqueness**	❖ Uses an Ensemble model ❖ Explores weighted features for Neural Network approaches ❖ Extensive feature extraction strategy from the URL ❖ Simple, Easy-to-Understand UI
4.	**Social Impact / Customer Satisfaction**	❖ This is a very hands off approach, the user does not have to do any work and let the extension inform the user about the legitimacy of the website. ❖ Users need not fear of losing their money to phishing scams. ❖ Customers don't need to rely on offline transactions because of the fear of initiating transactions online.
5.	**Business Model (Revenue Model)**	❖ Site can charge a one time fee for a device/user based on demographic surveys (Rs. 50 per year) ❖ Companies can be charged a discounted fee due to bulk purchase of the Application Programming Interface (API) ❖ Premium users will have access to details of the URL and reasonings for why a site has been classified 'unsafe'
6.	**Scalability of the Solution**	❖ Solution can use additional hardware resources when the amount of users and activity is increased ❖ The API can ensure that multiple requests at the same time are handled in a parallel fashion

3.4 Problem Solution Fit:

Problem-Solution Fit happens when there is proof that customers are interested in particular tasks, challenges, and benefits. You've established that a problem exists and created a value offer that takes into account the tasks, challenges, and gains of your clients at this point.

A problem-solution-fit occurs when such a solution is discovered and a business develops a strategy that, from a variety of angles, offers a game changer for customers.

However, if businesses miss evaluating the Problem-Solution Fit they developed, they face a risk of finding that no one wants their solution, which is unfortunate considering the effort and money invested.



CHAPTER 4
REQUIREMENT ANALYSIS

4.1 Functional Requirements:

FR No.	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
FR-1	User Registration	Registration through Gmail
FR-2	User Confirmation	Confirmation via Email
FR-3	User Input	User enter the URL in required field for validation
FR-4	URL Processing	Model compares the websites using the blacklist and whitelist approach.
FR-5	Prediction	Model predicts the URL using Machine Learning algorithm.
FR-6	Classifier	The URL will be identified as Malicious or not
FR - 7	Result	Result predicted by the model is displayed to the user.

4.2 Non – Functional Requirements:

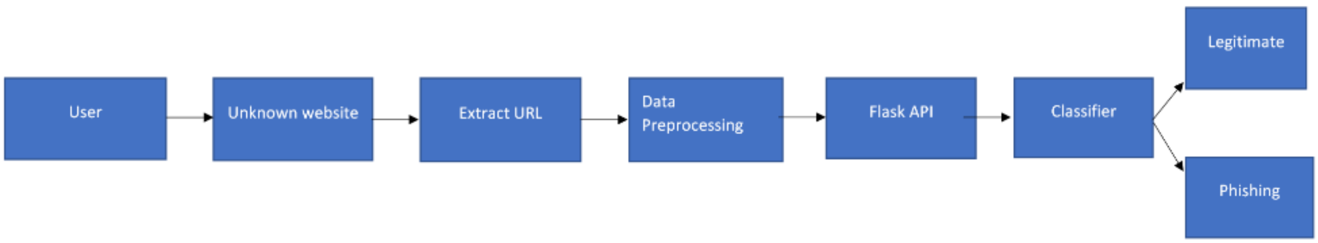
FR No.	Non-Functional Requirement	Description
NFR-1	**Usability**	With an efficient, user-friendly UI, users will not have difficulty in using the solution and navigating through the system
NFR-2	**Security**	By enabling google authentication which provides multi factor authentication
NFR-3	**Reliability**	Probability of failure free operations in the specified environment of usage
NFR-4	**Performance**	The performance should be faster and user friendly for efficiency
NFR-5	**Availability**	The model should be available for use always, it can be exported to users and can be run in the local machine
NFR-6	**Scalability**	This can be developed into an API which can be incorporated by others who can make use of **i**

CHAPTER 5

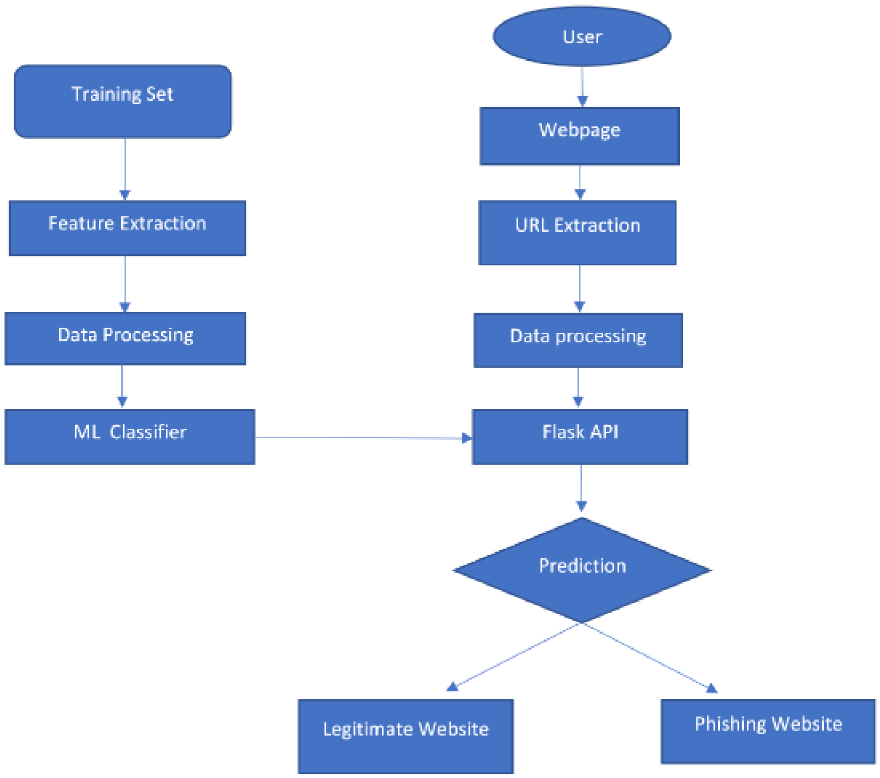
PROJECT DESIGN

5.1 Data Flow Diagram:

A data flow diagram is a visualization tool used to illustrate the flow of processes in a company or a specific project within it. It highlights the movement of information as well as the sequence of steps or events required to complete a work task.



5.2 Solution and Technical Architecture



5.3 USER STORIES

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
Customer (Web user)	Website	USN-1	As a user, I need a website to check whether the URL is safe to enter or not.	Website should be user-friendly and responsive	High	Sprint-3
	Alert Notification	USN-2	If I enter into some Malicious Link , Notification has to be sent to me	Receive notification in mobile or to my mail id	Low	Sprint-3
	Blocking	USN-3	If the link is not safe to enter, It should block me to use that site.		High	Sprint-2
	Allowing	USN-4	If I wish to use that website then ,iT should also allow me to enter into that website		Medium	Sprint-2
	Accurate Prediction	USN-5	As a User, I need a correct result. There shouldn't be any anomaly	The phishing website has to be determined correctly.	High	Sprint-1

CHAPTER 6

PROJECT PLANNING & SCHEDULING

6.1 Sprint planning and estimation

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-1		USN-1	As a user, I can able to view the home page	5	High	Sivaprakash, Vignesh
Sprint-3		USN-3	As a user, I can register for the application through LinkedIn	10	Low	Sanjaikuma, Sathish
Sprint-2		USN-4	As a user, I can register for the application through G mail	5	Medium	Sanjaikuma,
Sprint-2	Dashboard	USN-6	As a user, I paste the Link that needs to be Verified as a Phishing site or not	5	High	Sanjaikuma, Sathish
Sprint-2		USN-7	As a user,I can see the Result	10	High	Sanjaikuma, Sathish
Sprint-3		USN-8	As a user,I can see the Result	10	Medium	Sanjaikuma, Sathish
Sprint-4	Result	USN-9	As a Administrator, I can Answer the User Queries	10	Low	Sanjaikuma
Sprint-4		USN-10	As a Administrator, I can Improve the Accuracy	10	High	Sanjaikuma,

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	15	05 Nov 2022
Sprint-3	20	6 Days	07 Nov 2022	12 Nov 2022	10	12 Nov 2022
Sprint-4	20	6 Days	14 Nov 2022	19 Nov 2022	20	19 Nov 2022

CHAPTER 7

CODING & SOLUTION

7.1 Model Building

7.1.1 Data Collection & Exploratory Data Analysis

The dataset contains 32 features and 11055 records. All the data columns are of the type int64. EDA is the process of performing initial investigation on the dataset. The features that are present in the data set include:

- IP Address in URL
- Length of URL
- Using URL Shortening Services
- "@" Symbol in URL
- Redirection "//" in URL
- Prefix or Suffix "-" in Domain
- Having Sub Domain
- Length of Domain Registration
- Favicon
- Port Number
- HTTPS Token
- Request URL
- URL of Anchor
- Links in Tags
- SFH
- Email Submission
- Abnormal URL
- Status Bar Customization (on mouse over)
- Disabling Right Click
- Presence of Popup Window
- IFrame Redirection
- Age of Domain
- DNS Record
- Web Traffic
- Page Rank
- Google Index
- Links pointing to the page
- Statistical Report
- Result

```python
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)]]
```

7.1.2 Data Visualization

Data visualization helps to understand the data and also explain the data to others. Histogram, box plot, correlation matrix plot, scatter matrix plot, pair plot has been plotted.

Univariate analysis: Univariate analysis provides an understanding in the characteristics of each feature in the data set. Different characteristics are computed for numerical and categorical data. For the numerical features characteristics are standard deviation, skewness, kurtosis, percentile, interquartile range (IQR) and range. For the categorical features characteristics are count, cardinality, list of unique values, top and freq.

	count	mean	std	min	25%	50%	75%	max
index	11055.0	5528.000000	3191.447947	1.0	2764.5	5528.0	8291.5	11055.0
having_IPhaving_IP_Address	11055.0	0.313795	0.949534	-1.0	-1.0	1.0	1.0	1.0
URLURL_Length	11055.0	-0.633198	0.766095	-1.0	-1.0	-1.0	-1.0	1.0
Shortining_Service	11055.0	0.738761	0.673998	-1.0	1.0	1.0	1.0	1.0
having_At_Symbol	11055.0	0.700588	0.713598	-1.0	1.0	1.0	1.0	1.0
double_slash_redirecting	11055.0	0.741474	0.671011	-1.0	1.0	1.0	1.0	1.0
Prefix_Suffix	11055.0	-0.734962	0.678139	-1.0	-1.0	-1.0	-1.0	1.0
having_Sub_Domain	11055.0	0.063953	0.817518	-1.0	-1.0	0.0	1.0	1.0
SSLfinal_State	11055.0	0.250927	0.911892	-1.0	-1.0	1.0	1.0	1.0
Domain_registeration_length	11055.0	-0.336771	0.941629	-1.0	-1.0	-1.0	1.0	1.0
Favicon	11055.0	0.628584	0.777777	-1.0	1.0	1.0	1.0	1.0
port	11055.0	0.728268	0.685324	-1.0	1.0	1.0	1.0	1.0
HTTPS_token	11055.0	0.675079	0.737779	-1.0	1.0	1.0	1.0	1.0
Request_URL	11055.0	0.186793	0.982444	-1.0	-1.0	1.0	1.0	1.0
URL_of_Anchor	11055.0	-0.076526	0.715138	-1.0	-1.0	0.0	0.0	1.0
Links_in_tags	11055.0	-0.118137	0.763973	-1.0	-1.0	0.0	0.0	1.0
SFH	11055.0	-0.595749	0.759143	-1.0	-1.0	-1.0	-1.0	1.0
Submitting_to_email	11055.0	0.635640	0.772021	-1.0	1.0	1.0	1.0	1.0
Abnormal_URL	11055.0	0.705292	0.708949	-1.0	1.0	1.0	1.0	1.0
Redirect	11055.0	0.115694	0.319872	0.0	0.0	0.0	0.0	1.0
on_mouseover	11055.0	0.762099	0.647490	-1.0	1.0	1.0	1.0	1.0
RightClick	11055.0	0.913885	0.405991	-1.0	1.0	1.0	1.0	1.0
popUpWidnow	11055.0	0.613388	0.789818	-1.0	1.0	1.0	1.0	1.0
Iframe	11055.0	0.816915	0.576784	-1.0	1.0	1.0	1.0	1.0
age_of_domain	11055.0	0.061239	0.998168	-1.0	-1.0	1.0	1.0	1.0
DNSRecord	11055.0	0.377114	0.926209	-1.0	-1.0	1.0	1.0	1.0
web_traffic	11055.0	0.287291	0.827733	-1.0	0.0	1.0	1.0	1.0
Page_Rank	11055.0	-0.483673	0.875289	-1.0	-1.0	-1.0	1.0	1.0
Google_Index	11055.0	0.721574	0.692369	-1.0	1.0	1.0	1.0	1.0
Links_pointing_to_page	11055.0	0.344007	0.569944	-1.0	0.0	0.0	1.0	1.0
Statistical_report	11055.0	0.719584	0.694437	-1.0	1.0	1.0	1.0	1.0
Result	11055.0	0.113885	0.993539	-1.0	-1.0	1.0	1.0	1.0

7.1.3 Data Preprocessing & Splitting the dataset

In the first step data preprocessing is done. Preprocessing is the method by which we perform data cleaning i.e., raw dataset is converted into cleaned dataset. There are no missing values in the dataset. The dataset is divided into 80:20 ratio where 80% is for training data and 20% for testing data

```
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]
```

```
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)
```

7.1.4 Model Building and Hyper parameter tuning

From the dataset above, it is clear that this is a supervised machine learning task. There are two major types of supervised machine learning problems, called classification and regression. This data set comes under classification problem, as the input URL is classified as phishing (-1) or legitimate (1). In this we have done 5 supervised classification algorithm namely Logistic regression, support vector machine, Random Forest, Decision tree, Naïve bayes.

Hyper parameter tuning is the process through which we choose a set of optimal hyper parameters for learning algorithm. Hyper parameters are model argument whose value is set before the learning process begins. Not all hyper parameters are equally important. GridSearchCV is a method to find the best set of optimal hyper parameters from a grid and this method will go through all the possible intermediate combinations. As a result, accuracy can be improved.

Evaluation is done using classification accuracy. Confusion matrix and classification report is also plotted.

Logistic Regression

```
#Logistic Regression
from sklearn.linear_model import LogisticRegression
lr = LogisticRegression()
lr.fit(x_train,y_train)
```

```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=100,
                   multi_class='auto', n_jobs=None, penalty='l2',
                   random_state=None, solver='lbfgs', tol=0.0001, verbose=0,
                   warm_start=False)
```

```
#accuracy
y_pred = lr.predict(x_test)
log_reg = accuracy_score(y_test,y_pred)

print(classification_report(y_test, y_pred))
print(f"{round(log_reg*100,2)}% Accurate")
```

```
              precision    recall  f1-score   support

          -1       0.92      0.89      0.91      1014
           1       0.91      0.94      0.92      1197

    accuracy                           0.92      2211
   macro avg       0.92      0.91      0.92      2211
weighted avg       0.92      0.92      0.92      2211

91.68% Accurate
```

Support Vector Machine

```python
#Support Vector Machine
from sklearn import svm
sv = svm.SVC()
sv.fit(x_train,y_train)
```

```
SVC(C=1.0, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma='scale', kernel='rbf',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False)
```

```python
#accuracy
y_pred5 = sv.predict(x_test)
supp_vec = accuracy_score(y_test,y_pred5)

print(classification_report(y_test, y_pred5))
print(f"{round(supp_vec*100,2)}% Accurate")
```

```
              precision    recall  f1-score   support

          -1       0.95      0.92      0.93      1014
           1       0.93      0.96      0.95      1197

    accuracy                           0.94      2211
   macro avg       0.94      0.94      0.94      2211
weighted avg       0.94      0.94      0.94      2211

94.08% Accurate
```

Decision Tree

```
#Decision Tree Classifier
from sklearn.tree import DecisionTreeClassifier
dt = DecisionTreeClassifier()
dt.fit(x_train,y_train)
```

```
DecisionTreeClassifier(ccp_alpha=0.0, class_weight=None, criterion='gini',
                       max_depth=None, max_features=None, max_leaf_nodes=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, presort='deprecated',
                       random_state=None, splitter='best')
```

```
#accuracy
y_pred3 = dt.predict(x_test)
des_class = accuracy_score(y_test,y_pred3)

print(classification_report(y_test, y_pred3))
print(f"{round(des_class*100,2)}% Accurate")
```

```
              precision    recall  f1-score   support

          -1       0.96      0.95      0.96      1014
           1       0.96      0.97      0.96      1197

    accuracy                           0.96      2211
   macro avg       0.96      0.96      0.96      2211
weighted avg       0.96      0.96      0.96      2211

96.07% Accurate
```

Random Forest

```
#Random Forest Classifier
from sklearn.ensemble import RandomForestClassifier
rf = RandomForestClassifier()
rf.fit(x_train,y_train)
```

```
RandomForestClassifier(bootstrap=True, ccp_alpha=0.0, class_weight=None,
                       criterion='gini', max_depth=None, max_features='auto',
                       max_leaf_nodes=None, max_samples=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, n_estimators=100,
                       n_jobs=None, oob_score=False, random_state=None,
                       verbose=0, warm_start=False)
```

```
#accuracy
y_pred2 = rf.predict(x_test)
ran_for = accuracy_score(y_test,y_pred2)

print(classification_report(y_test, y_pred2))
print(f"{round(ran_for*100,2)}% Accurate")
```

```
              precision    recall  f1-score   support

          -1       0.98      0.95      0.97      1014
           1       0.96      0.99      0.97      1197

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

96.92% Accurate
```

Naïve Bayes

```
#K Neighbors Classifier
from sklearn.neighbors import KNeighborsClassifier
kc = KNeighborsClassifier()
kc.fit(x_train,y_train)
```

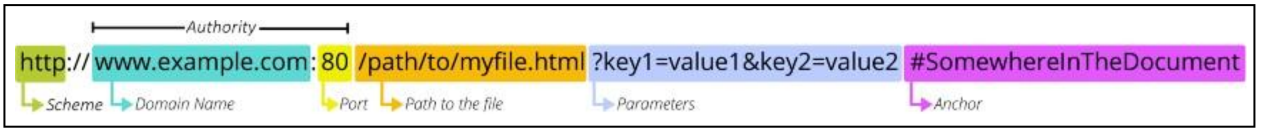
```
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
                     metric_params=None, n_jobs=None, n_neighbors=5, p=2,
                     weights='uniform')
```

```
#accuracy
y_pred4 = kc.predict(x_test)
kn_class = accuracy_score(y_test,y_pred4)

print(classification_report(y_test, y_pred4))
print(f"{round(kn_class*100,2)}% Accurate")
```

```
              precision    recall  f1-score   support

          -1       0.95      0.92      0.94      1014
           1       0.94      0.96      0.95      1197

    accuracy                           0.94      2211
   macro avg       0.94      0.94      0.94      2211
weighted avg       0.94      0.94      0.94      2211

94.17% Accurate
```

7.2 Feature Extraction from URL

The address of a specific unique resource on the Web is all that is contained in a URL, also known as a uniform resource locator. Theoretically, every legitimate URL leads to a different resource.



URL Features

Deep learning techniques offer a predictive strategy that is independent of prior knowledge of well-known signatures and generalization across platforms. ML approaches will extract features of well-known good and bad URLs and generalize these features to identify new and previously undiscovered good or bad URLs given a sample of legitimate and malicious malware samples.

URL having IP

From URL, we are checking whether IP address is present or not. If URL has IP address, then there is a chance that the URL has some malicious link.

Code Snippet:

```
def url_having_ip(url):
  if len(url) < 54:
        return 1
  if len(url) >= 54 and len(url) <= 75:
        return 0
  return -1
```

URL Length

If Length of the URL is less than 54 , then the website can be phishing website.Malicious URLs are generally shorter in length than benign URLs.

Code Snippet:

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

Having @ symbol

If an website contain @ symbol then the website may be malicious.

Code Snippet:

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

Extract prefix-suffix

If domain of the website contain '-' ,then the website is malicious. For instance,"https://www.binance-co.com/", this website contain hypen in domain , and it is classified as malicious link.

Code Snippet:

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

Extract subdomain

If there is more than one subdomain present in the URL , then the website may be malicious. For instance,the URL ‘amazon.com’ do not look suspicious, however, the same sub-string looks malicious in ‘amazon.com.support.info’.

Code Snippet:

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

SSL final state

A secure connection can be established over the internet with the aid of the Secure socket layer (SSL) or Transport level security (TLS) protocols. But it does more than just collect information. Its purpose is to securely verify the identities of the websites. Check whether website has http connection in secure way by https.

Code Snippet:

```
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','QuoVadis','Deutsche Telekom','Network
Solutions','SwissSign','IdenTrust','Secom','TWCA','GeoTrust','Thawte','Doster','VeriSign']
#getting age of certificate
        startingDate = str(certificate['notBefore'])
        endingDate = str(certificate['notAfter'])
        startingYear = int(startingDate.split()[3])
        endingYear = int(endingDate.split()[3])
        Age_of_certificate = endingYear-startingYear

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

    except Exception as e:

        return 1
```

Domain registration

From the url , website domain registration date is found, if it is less than 365 days then the website is phishing website.

Code Snippet:

```
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
```

HTTP Token

If sometimes , the attacker can include https part in domain of the website to make that website look like secure one.

Code Snippet:

```
def https_token(url):
    subDomain, domain, suffix = extract(url)
    host =subDomain +'.' + domain + '.' + suffix
if(host.count('https')):
    return 1
    else:
        return -1
```

Url of Anchor

When one hits a link (anchor tag) on a web page, and it opens in a new browser tab, there are chances that a hacker might have taken control over your original tab web page.while the link is opening in another tab, the attacker can redirect the original tab's URL location to a phishing page in the background, designed to look like the real original page, asking for login credentials

Code Snippet:

```
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
```

Links in tags

Code Snippet:

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

Email submits

Extracting the webpage html file and check whether the webpage is try to send mail to any other website, then the website is malicious.

Code Snippet:

```
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
```

Age of domain

The WHOIS database can be used to extract this characteristic. The majority of phishing websites are only active for a little time. For this initiative, a legal domain must have a minimum age of 12 months. Age in this context simply refers to the interval between creation and expiration times.

Code Snippet:

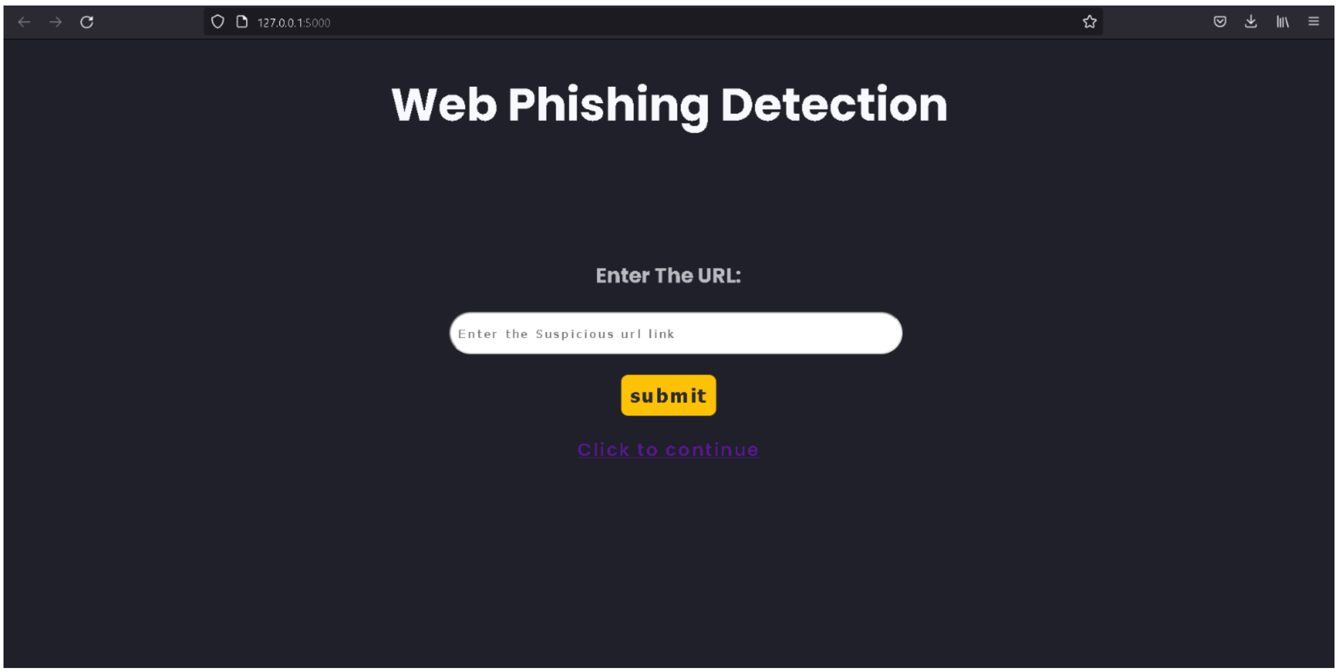
```
def age_of_domain(url):
    try:
        w = whois.whois(url)
        start_date = w.creation_date
        current_date = datetime.datetime.now()
        age =(current_date-start_date[0]).days
```

```
        if(age>=180):
            return -1
        else:
            return 1
    except Exception as e:
        print(e)
        return 0
```

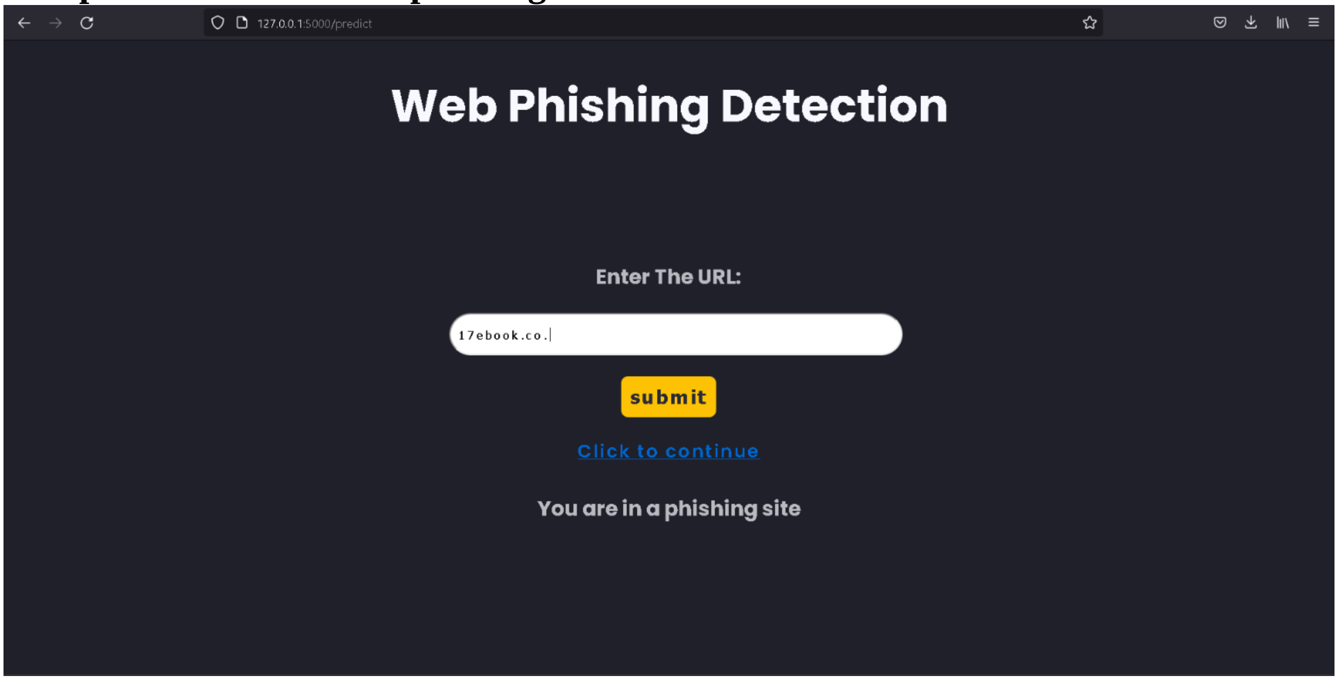
7.2.1 Phishing Website

Home page

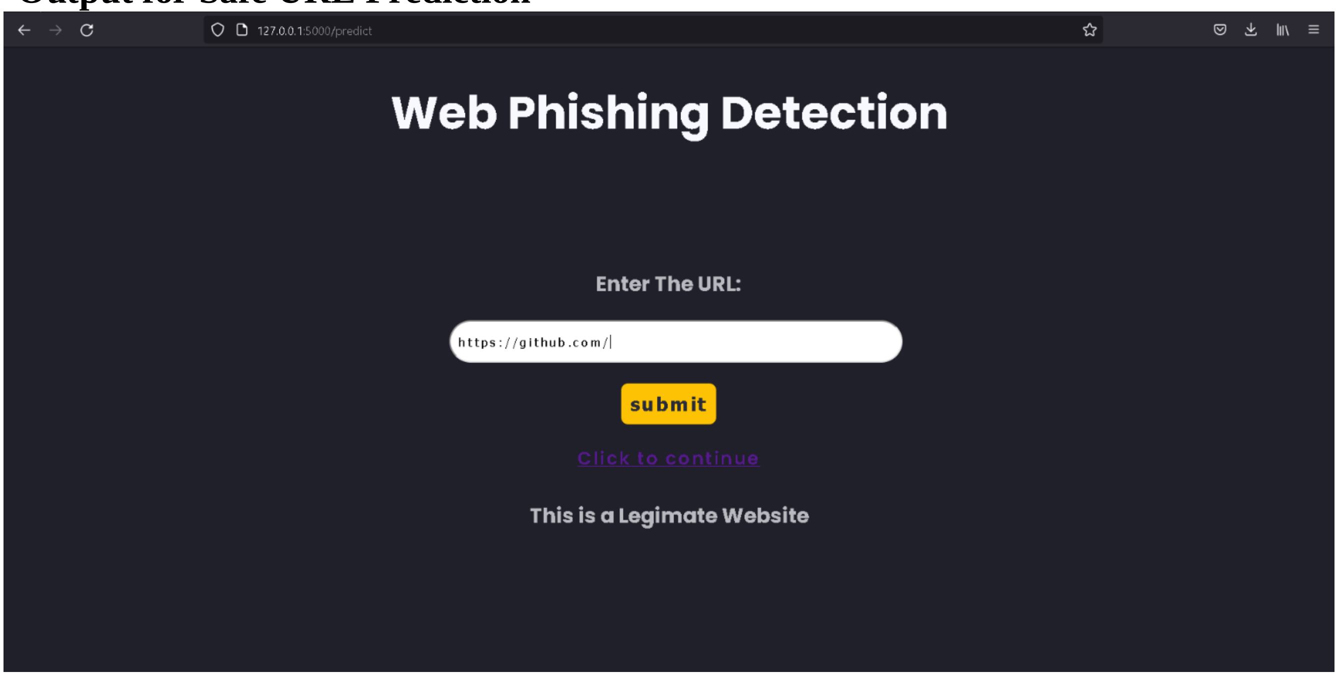
Landing page of the phishing detection page where user can find check url button to check whether the URL is good or not.



Output for detection for phishing URL



Output for Safe URL Prediction



Chapter 8

TESTING

8.1 Test Case

Test case ID	Feature Type & Component	Test Scenario	Steps To Execute	Test Data	Expected Result	Actual Result	Status
HomePage_TC_OO1	Functional & Home Page	Verify user is able to see check URL button to predict the trustworthiness of the URL	1.Enter URL and click go 2.In Check section , user can view the check URLbutton. 3.Redirects to the phishing detection page.	https://github.com/	Application should predict whether the URL is good or bad	Working as expected	Pass

Phishing Page_TC _OO2	Functional & Phishing page	Verify user is able to enter url and check whether url is good or bad	1.Enter URL(https://subhiksha.pythonanywhere.com/) and click go 2.Click on check url in check Section 3.Redirects user to phishing detection page 4.Enter url in input field 5.Click on predict button	https://17ebook.co.cutesta.com/	User should get result the url is not safe to enter	Working as expected	Pass
Phishing Page_TC _OO4	Functional & Phishing page	Verify user is not getting result when url is not entered	1.Enter URL and click go 2.Click on check URL button 3.Redirects to the phishing page 4.Click on predict button without entering anything to the input field 6. The message is displayed to enter URL in input field		Application should show 'Please fill out Enter URL Field'	Working as expected	Pass

8.2 User Acceptance Testing

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	5	3	2	2	15
Duplicate	1	0	1	0	2
External	2	2	0	0	4
Fixed	9	4	5	10	20
Not Reproduced	0	0	1	0	1
Skipped	0		0	1	1
Won't Fix	0	4	1	1	6
Totals	18	13	10	14	49

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	9	0	0	9
Client Application	45	0	0	45
Security	2	0	0	2
Outsource Shipping	2	0	0	2
Exception Reporting	10	0	0	10
Final Report Output	3	0	0	3
Version Control	2	0	0	2

CHAPTER 9

RESULTS

9.1 Performance Metrics

	Classfication Algorithms	Accuracy
1	Random Forest Classifier	0.969697
2	Decision Tree Classifier	0.963817
3	K Neighbors Classifier	0.943464
4	Support Vector Machine	0.940751
0	Logistic Regression	0.916780

CHAPTER 10

ADVANTAGES & DISADVANTAGES

ADVANTAGES:

- Identify the phishing URL’s
- Differentiate legitimate and phishing links
- User friendly

DISADVANTAGES:

- Not a generalized model
- Huge number of rules

CHAPTER 11

CONCLUSION

Due to its importance in preserving privacy and ensuring security, experts are currently very interested in the detection of phishing. There are numerous ways to detect phishing. By applying machine learning, our technology seeks to improve the detection process for phishing websites. We were successful in achieving a high detection accuracy, and the findings demonstrate that the classifiers work better as we use more training data. Maximum accuracy of 97.46% is achieved for testing data with random forest classifier and we used that to deploy in the cloud. Ensemble methods like stacking also have been tried to improve the accuracy of weak learners along with hyper parameter tuning to improve the classification accuracy for both testing and training data.

CHAPTER 12

FUTURE SCOPE

We plan to develop system add-ons in the future, and if we can obtain a structured dataset of phishing, we will be able to detect it much more quickly than with any other method. In the future work a web extension can be made so that the working will be much simplified for the end users / customers.

Chapter 13

APPENDIX

13.1 Source Code

Flask Integration with scoring end point

```
import numpy as np
from flask import Flask, render_template, request, redirect, jsonify
from markupsafe import escape
import pickle
import inputScript
import requests
import os
from dotenv import load_dotenv

load_dotenv()


token_response = requests.post('https://iam.cloud.ibm.com/identity/token',
data={"apikey":"vSzdKylG1_OAEEHPigvybydUqEMPOlvQ0j1c85gq3hzd", "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__)

# user-inputs the URL in this page
@app.route('/')
def predict():
    return render_template("index.html")

#  fetches given URL and passes to inputScript
@app.route('/predict',methods=["POST"])
def y_predict():
    url = request.form['url']
    check_predic = inputScript.main(url)
```

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

    response_scoring = requests.post('https://us-south.ml.cloud.ibm.com/ml/v4/deployments/abd0569c-
f8e9-45e7-bd7b-8bd2977876f8/predictions?version=2022-11-20',
json=payload_scoring,headers={'Authorization': 'Bearer ' + mltoken})


    predic = response_scoring.json()

    result = predic['predictions'][0]['values'][0][0]


    if(result==-1):
        pred = "This is a Legimate Website"
    elif(result==1):
        pred = "You are in a phishing site"

    return render_template("index.html", pred_text = '{}'.format(pred), url = url)

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

InputScript.py

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




def url_having_ip(url):
   if len(url) < 54:
         return 1
   if len(url) >= 54 and len(url) <= 75:
         return 0
   return -1



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):
   match =
re.search('bit\.ly|goo\.gl|shorte\.st|go2l\.ink|x\.co|ow\.ly|t\.co|tinyurl|tr\.im|is\.gd|cli\.gs|'
                 'yfrog\.com|migre\.me|ff\.im|tiny\.cc|url4\.eu|twit\.ac|su\.pr|twurl\.nl|snipurl\.com|'

'short\.to|BudURL\.com|ping\.fm|post\.ly|Just\.as|bkite\.com|snipr\.com|fic\.kr|loopt\.us|'
```

```
'doiop\.com|short\.ie|kl\.am|wp\.me|rubyurl\.com|om\.ly|to\.ly|bit\.do|t\.co|lnkd\.in|'
                    'db\.tt|qr\.ae|adf\.ly|goo\.gl|bitly\.com|cur\.lv|tinyurl\.com|ow\.ly|bit\.ly|ity\.im|'

'q\.gs|is\.gd|po\.st|bc\.vc|twitthis\.com|u\.to|j\.mp|buzurl\.com|cutt\.us|u\.bb|yourls\.org|'

'x\.co|prettylinkpro\.com|scrnch\.me|filoops\.info|vzturl\.com|qr\.net|1url\.com|tweez\.me|v\.gd|t
r\.im|link\.zip\.net',
                    url)
    if match:
        return -1
    return 1

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

def doubleSlash(url):
    if url.rfind('//') > 6:
        return -1
    return 1

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

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

def SSLfinal_State(url):
    try:
#check wheather contains https
        if(regex.search('^https',url)):
```

```
            usehttps = 1
        else:
            usehttps = 0
#getting the certificate issuer to later compare with trusted issuer
        #getting host name
        subDomain, domain, suffix = extract(url)
        host_name = domain + "." + suffix
        context = ssl.create_default_context()
        sct = context.wrap_socket(socket.socket(), server_hostname = host_name)
        sct.connect((host_name, 443))
        certificate = sct.getpeercert()
        issuer = dict(x[0] for x in certificate['issuer'])
        certificate_Auth = str(issuer['commonName'])
        certificate_Auth = certificate_Auth.split()
        if(certificate_Auth[0] == "Network" or certificate_Auth == "Deutsche"):
            certificate_Auth = certificate_Auth[0] + " " + certificate_Auth[1]
        else:
            certificate_Auth = certificate_Auth[0]
        trusted_Auth =
['Comodo','Symantec','GoDaddy','GlobalSign','DigiCert','StartCom','Entrust','Verizon','Trustwa
ve','Unizeto','Buypass','QuoVadis','Deutsche Telekom','Network
Solutions','SwissSign','IdenTrust','Secom','TWCA','GeoTrust','Thawte','Doster','VeriSign']
#getting age of certificate
        startingDate = str(certificate['notBefore'])
        endingDate = str(certificate['notAfter'])
        startingYear = int(startingDate.split()[3])
        endingYear = int(endingDate.split()[3])
        Age_of_certificate = endingYear-startingYear

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

    except Exception as e:

        return 1

def domain_registration(url):
    try:
        w = whois.whois(url)
```

```
        updated = w.updated_date
        exp = w.expiration_date
        length = (exp[0]-updated[0]).days
        if(length<=365):
            return 1
        else:
            return -1
    except:
        return 0

def favicon(url):
    return 0

def port(url):
    return 0

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

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

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

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

```
        for video in vids:
            subDomain, domain, suffix = extract(video['src'])
            vidDomain = domain
            if(websiteDomain==vidDomain or vidDomain==''):
                linked_to_same = linked_to_same + 1
        linked_outside = total-linked_to_same
        if(total!=0):
            avg = linked_outside/total

        if(avg<0.22):
            return -1
        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):
            return -1
        elif(0.25<=avg<=0.81):
            return 0
        else:
            return 1
    except:
        return 0

def sfh(url):
    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):
        return 0


def redirect(url):
    responses = requests.get(url)
    if responses.history == " ":
        return 1
    else:
        return -1

def on_mouseover(url):
    #ongoing
    return 0

def rightClick(url):
    #ongoing
    return 0

def popup(url):
    #ongoing
    return 0

def iframe(url):
    #ongoing
    return 0

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

```
        print(e)
        return 0

def dns(url):
    #ongoing
    return 0

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

def page_rank(url):
    #ongoing
    return 0

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

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


def links_pointing(url):
    return 0

def statistical(url):
    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
```

Home.html

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>

    <link rel="stylesheet" href="../static/style.css">

    <!-- <link rel="preconnect" href="https://fonts.googleapis.com">
    <link rel="preconnect" href="https://fonts.gstatic.com" crossorigin>
    <link href="https://fonts.googleapis.com/css2?family=Patrick+Hand&display=swap"
rel="stylesheet"> -->

</head>
<body>

<div class="center">
  <h1>Web Phishing Detection </h1><br>
   <form action="/predict" method="post">
      <div class="con">
        <label for="url">Enter The URL:</label><br><br>
        <input type="text" placeholder="Enter the Suspicious url link" name="url">
        <br><br>
        <button type="submit" >submit</button>
        <br><br>
        <a href="{{url}}" class="url">Click to continue</a>
```

```
      <br>
      <h4 >{{ pred_text }}</h4>
    </div>
  </form>

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

GitHub and project link

GitHub link - https://github.com/IBM-EPBL/IBM-Project-31763-1660204777