



SCAM LINK DETECTION

# 

# A PROJECT REPORT

***Submitted by***

|  |  |
| --- | --- |
| DINAKARAN M | 721219205012 |
| DINESH R | 721219205013 |
| KARAN S | 721219205027 |
| KAVIARASAN P | 721219205030 |

***In partial fulfillment for the award of the degree***

***of***

# 

# BACHELOR OF TECHNOLOGY

# 

# *IN*

INFORMATION TECHNOLOGY

**KARPAGAM INSTITUTE OF TECHNOLOGY**

**ANNA UNIVERSITY: CHENNAI 600 025**

MAY 2023

I

# ANNA UNIVERSITY : CHENNAI 600 025

# BONAFIDE CERTIFICATE

Certified that this project report **“SCAM LINK DETECTION”** is bonafide work of **DINAKARAN.M(721219205012), DINESH.R(721219205013), KARAN.S(721219205027), KAVIARASAN.P(721219205030)** who carried out the project work under my supervision.

# SIGNATURE SIGNATURE

Dr. D. Bhanu., M.E., Ph.D., Dr.M.Sathiya., M.E.,Ph.D.,

# HEAD OF THE DEPARTMENT SUPERVISOR

Professor, Associate Professor**,**

Department of Information Technology, Department of Information Technology,

Karpagam Institute of Technology, Karpagam Institute of Technology,

Coimbatore – 641105. Coimbatore – 641105.

Submitted for the university project Viva-voce examination conducted at

Karpagam Institute of Technology, Coimbatore, on ……………

# Internal Examiner External Examiner

II

# ACKNOWLEDGEMENT

With genuine humility, we are obediently thankful to God Almighty Without him, this work would have never been a reality.

We express our profound gratitude to our respected Chairman **Dr.R.VASANTHAKUMAR,** for giving this opportunity to pursue this course. At this pleasing moment of having successfully completed the Project work.

We wish to acknowledge sincere gratitude and heartfelt thanks to our respected Principal **Dr.P.MANIMARAN M.E, Ph.D.,**for having us given the adequate support and opportunity for completing the project work successfully.

We express our deep sense of gratitude and sincere thanks to our beloved Head of the Department **Dr.D.BHANU M.E, Ph.D.,** who has been a spark for enlightening our knowledge.

Our profound gratitude goes to our project coordinator **Dr.M.SATHIYA M.E.,Ph.D.,** and review members and all the faculty members of the Department of IT for the invaluable knowledge they have imparted on us.

We express our gratitude to the HOD and all the staff members of the Department of Information Technology for their support throughout the project.

Our humble gratitude and heartiest thanks go to our family members and friends for their encouragement and support throughout the course of this project.

III

# TABLE OF CONTENTS

|  |  |  |  |
| --- | --- | --- | --- |
| **CHAPTER NO** | **TITLE** | **PAGENO** | |
|  | **LIST OF FIGURES** | **V** |
|  | **LIST OF ABBREVIATIONS** | **VI** |
|  | **ABSTRACT** | **VII** |
| **1** | **INTRODUCTION** | **1** |
| **2** | **SYSTEM STUDY** | **2** |
|  | 2.1 EXISTING SYSTEM | 2 |
|  | 2.2 PROPOSED SYSTEM | 2 |
|  | 2.2.1 ADVANTAGE | 4 |
| **3** | **SYSTEM SPECIFICATION**  3.1 HARDWARE REQUIREMENTS  3.2 SOFTWARE REQUIREMENTS | **5**  5  5 |
| **4** | **SOFTWARE DESCRIPTIONS** | **6** |
|  | 4.1 FRONT END  4.2 BACK END  4.3 DATABASE | 6  7  7 |
| **5** | **PROJECT DESCRIPTION**  5.1 ALGORITH DESCRIPTION  5.2 MODULE DESCRIPTION | **8**  10  12 |
| **6** | **SYSTEM TESTING** | **23** |
| **7** | **FEASIBILITY STUDY** | **25** |
| **8** | **RESULT** | **26** |
| **9** | **FUTURE ENHANCEMENTC** | **27** |
|  | **SOURCE CODES** | **28** |
|  | **SCREENSHOTS** | **42** |
|  | **REFERENCE** | **47** |
|  | **CONFERENCE CERTIFICATE** | **48** |

IV

|  |  |  |
| --- | --- | --- |
| FIGURE NO | FIGURE NAME | PAGE NO |
| 5.2.1 | DATASET | 15 |
| 5.2.2 | CLASSIFICATION | 17 |
| 5.2.3 | PREDICTION | 19 |
| 5.2.4 | ACCURACY | 21 |

**LIST OF FIGURES**

V

**LIST OF ABBREVIATIONS**

|  |  |
| --- | --- |
| **ABBREVIATIONS** | **EXPLANATIONS** |
| GBC | Gradient Boosting Classifier |
| URL | Uniform Resource Locator |
| MITM | Man-In-The-Middle |
| ML | Machine Learning |
| EDA | Exploratory Data Analysis |
| TF | Term Frequency |
| IDF | Inverse Document Frequency |

VI

**ABSTRACT**

Phishing attacks continue to pose a significant threat to individuals and organizations, leading to financial losses, data breaches, and compromised user accounts. In response to this escalating problem, we present Phishing detection, an innovative web-based platform designed to detect and mitigate phishing attempts in real-time. Phishing detection combines cutting-edge machine learning techniques, advanced data analysis, and user behavior monitoring to provide a robust and reliable phishing detection system. Leveraging a vast database of known phishing URLs, the platform employs sophisticated algorithms to analyze web content, URLs, and email headers to identify potential phishing attacks accurately. The system's user-centric approach emphasizes the integration of user behavior analysis, incorporating personalized risk assessment to identify suspicious activities and patterns. By considering individual user profiles, Phishing detection can adapt its detection methods to provide tailored protection, improving the accuracy and effectiveness of the system. Phish detection web-based interface offers a user-friendly experience, allowing individuals and organizations to access and utilize the platform easily. The intuitive dashboard provides comprehensive insights into the detected phishing threats, including detailed reports, visual analytics, and timely notifications. Users can leverage these insights to enhance their awareness, take preventive measures, and promptly respond to potential threats. To ensure the reliability and up-to-date nature of its phishing database, Phishing detection utilizes advanced threat intelligence techniques, continuously monitoring and collecting data from various sources, including security communities, public feeds, and user feedback. This dynamic approach allows the system to adapt and stay ahead of emerging phishing techniques, protecting users against the latest threats. Phish detection offers a comprehensive and user-centric solution to combat the growing threat of phishing attacks. By combining advanced machine learning algorithms, personalized risk assessment, and real-time monitoring, the platform provides individuals and organizations with an effective defense against phishing attempts. Phish detection empowers users to proactively safeguard their online activities, protect sensitive information, and maintain a secure digital environment.

VII

# CHAPTER 1

**INTRODUCTION**

Phishing is a type of cybersecurity attack during which malicious actors send messages pretending to be a trusted person or entity. Phishing messages manipulate a user, causing them to perform actions like installing a malicious file, clicking a malicious link, or divulging sensitive information such as access credentials. Phishing is the most common type of social engineering, which is a general term describing attempts to manipulate or trick computer users. Social engineering is an increasingly common threat vector used in almost all security incidents. Social engineering attacks, like phishing, are often combined with other threats, such as malware, code injection, and network attacks. Phishing is the most common form of [social engineering](https://www.ibm.com/in-en/topics/social-engineering), the practice of deceiving, pressuring or manipulating people into sending information or assets to the wrong people. Social engineering attacks rely on human error and pressure tactics for success. “Phishing” refers to an attempt to steal sensitive information, typically in the form of usernames, passwords, credit card numbers, bank account information or other important data in order to utilize or sell the stolen information. By masquerading as a reputable source with an enticing request, an attacker lures in the victim in order to trick them, similarly to how a fisherman uses bait to catch a fish. The most common examples of phishing are used to support other malicious actions. These attacks typically occur via [email](https://www.cloudflare.com/learning/email-security/what-is-email/) or instant message, and can be broken down into a few general categories. It’s useful to become familiar with a few of these different vectors of phishing attacks in order to spot them in the wild.

# CHAPTER 2

# SYSTEM STUDY

# 2.1 EXISTING SYSTEM

There are several existing machine learning systems for phishing website detection, some of which include:

* Open Phish - a free, open-source machine learning system that analyzes URLs to identify potential phishing websites.
* Phish Tank - a community-driven website that uses machine learning algorithms to detect and block phishing websites.
* Google Safe Browsing - a machine learning system that detects phishing and malware websites by analyzing URLs and web content.
* Microsoft Defender for Office 365 - a cloud-based security solution that uses machine learning algorithms to detect and block phishing attacks.
* Symantec Web Security Service - a cloud-based security solution that uses machine learning to analyze URLs and web content to identify potential phishing websites.
* Cofense Triage - a machine learning system that uses behavioral analysis and machine learning to identify phishing attacks in real-time.

These systems use a variety of machine learning techniques, including supervised and unsupervised learning, machine learning, and natural language processing, to identify and block phishing websites. They typically analyze website content, URLs, and other characteristics to determine whether a website is a phishing site or not.

**2.2 PROPOSED SYSTEM**

Phishing attacks are one of the most common cyber threats nowadays. To prevent these attacks, we propose a system called "Phish Detection" that uses a predictive attention mechanism with a Gradient Boosting Classifier (GBC) to detect and block phishing websites in real-time. The proposed system consists of several stages,

including dataset collection, preprocessing, feature extraction (URL and HTML), classification, model building and training, and performance evaluation. The dataset is collected from various sources and preprocessed to extract relevant features such as domain age, IP reputation, and SSL certificate information. The URL and HTML features are extracted using various techniques, such as domain analysis, HTML parsing, and content analysis. The classification stage involves training a Gradient Boosting Classifier (GBC) model using the extracted features to classify a website as either legitimate or phishing. The model is trained and evaluated using various performance metrics such as accuracy, precision, recall, and F1 score. The system is developed using Python Flask and MySQL. The admin is responsible for training the model and updating the database with new phishing information. Users can open their browser and input the URL into Phish Detection, which predicts and blocks the URL if it is identified as a phishing website. The system also stores attack information in the user's account on the Phish Detection website, enabling users to track their activity and take necessary measures. In future enhancements, we plan to incorporate additional features such as user behavior analysis and improved performance evaluation methods to enhance the accuracy and reliability of the system. Overall, Phish Detection is an effective solution to prevent phishing attacks and ensure the safety of internet users. The system consists of two main modules: the training module and the detection module. The training module involves collecting and pre-processing a dataset of legitimate and phishing URLs, extracting features from both URL and HTML content, building and training a Gradient Boosting Classifier (GBC), and evaluating the performance of the model. The detection module is responsible for predicting whether a given URL is phishing or legitimate in real-time. It takes a URL input from the user, extracts its features, feeds it into the trained model, and returns the prediction along with blocking the URL if it is identified as phishing. The system also stores the attack information in the user account on the Phish Detection website for further analysis. The proposed system aims to provide a more accurate and efficient solution for detecting and blocking phishing websites in real-time compared to traditional approaches. It also offers a user-friendly interface and a convenient way to store and analyze attack information.

**2.2.1 ADVANTAGES**

The proposed system "Phish Detection" has several advantages, such as:

1. Real-time detection and blocking: The system can detect and block phishing websites in real-time, providing immediate protection to users.
2. High accuracy: The use of Gradient Boosting Classifier (GBC) with predictive attention mechanism, coupled with the dataset collection, preprocessing, feature extraction (URL and HTML), classification, build and train, and performance evaluation, ensures high accuracy in identifying phishing websites.
3. Easy to use: The system is easy to use, as users only need to open the browser and input the URL. The system will predict and block the URL, and stores the attack information in the user account in Phish Detection Website.
4. User-friendly interface: The system has a user-friendly interface that allows users to easily access and manage their account, view blocked URLs and attack information.
5. Centralized database: The system uses MySQL as a centralized database, ensuring easy and secure management of user account information, blocked URLs, and attack information.

Overall, the proposed system provides a reliable and effective solution to protect users from phishing attacks in real-time, with high accuracy and ease of use.

**CHAPTER 3**

**SYSTEM SPECIFICATION**

* 1. **HARDWARE REQUIREMENTS**
* Processors : Intel® Core™ i5 processor 4300M at 2.60 GHz or 2.59 GHz (1 socket,

2 cores, 2 threads per core), 8 GB of DRAM

* Disk space : 320 GB
* Operating systems : Windows® 10, macOS\*, and Linux\*.

# SOFTWARE REQUIREMENTS

* Server Side : Python 3.7.4(64-bit) or (32-bit)
* Client Side : HTML, CSS, Bootstrap
* IDE : Flask 1.1.1
* Back end : MySQL 5.
* Server : WampServer 2i
* OS : Windows 10 64 –bit or Ubuntu 18.04 LTS “Bionic Beaver”

**CHAPTER 4**

**SOFTWARE DESCRIPTIONS**

# 4.1. FRONT END

# HTML, CSS and Bootstrap

* + - **HTML** - Hypertext Markup Language is the standard markup language for documents designed to be displayed in a web browser. It describes the

structure of a Web page.

* + - **CSS** - Cascading Style Sheets is a style sheet language used for describing the look and formatting of a document written in a markup language.
    - **Bootstrap -** Choose a suitable framework or toolset based on the project requirements and your familiarity with different options. Popular choices include React, Angular, and Vue.js. These frameworks provide a structured way to build dynamic and responsive user interfaces.

# FEATURES

* + - **HTML -** It allows the user to create and structure sections, paragraphs, headings, links, buttons, pictures, audio, video and blockquotes for web pages and applications.
    - **CSS -** CSS handles the look and feel part of a web page. Using CSS, you can control the color of the text, the style of fonts, how columns are sized and laid out, what background images or colors are used, layout designs, variations in display for different devices and screen sizes as well as a variety of other effects.
    - **Bootstrap -** Bootstrapped businesses often operate with limited financial resources, so adopting a lean approach is crucial. Focus on minimizing costs, optimizing processes, and avoiding unnecessary expenses. This includes keeping overhead low, leveraging technology, and streamlining operations to maximize efficiency. Place a strong emphasis on understanding and satisfying your customers' needs**.**

**4.2 BACK END**

**Python**

* Pythonis the Programming Language for the Web. It can update and change both HTML and CSS and also calculate, manipulate and validate data.
* Python is an object-oriented programming language. Python is a high-level language as the tr6+anslation of Python code takes place into machine language, using an interpreter.

**4.3 DATABASE**

**MYSQL**

* MySQL tutorial provides basic and advanced concepts of MySQL. Our MySQL tutorial is designed for beginners and professionals. MySQL is a relational database management system based on the Structured Query Language, which is the popular language for accessing and managing the records in the database. MySQL is open-source and free software under the GNU license. It is supported by Oracle Company. MySQL database that provides for how to manage database and to manipulate data with the help of various SQL queries. These queries are: insert records, update records, delete records, select records, create tables, drop tables, etc. There are also given MySQL interview questions to help you better understand the MySQL database.

**CHAPTER 5**

**PROJECT DESCRIPTION**

# "Phish Detection: Predictive Attention Mechanism using Gradient Boosting Classifier (GBC) for Real Time Detection and Blocking of Phishing Website" is a web application developed using Python Flask and MySQL. The application aims to provide a real-time detection and blocking solution for phishing websites and URLs. The project uses a gradient boosting classifier with a predictive attention mechanism for accurately classifying and training the phishing websites and URLs. The project involves several stages, including dataset collection, pre-processing, feature extraction (URL and HTML), classification, building, and training an GBC, and performance evaluation. Dataset Collection: The first stage of the project involves collecting a dataset of phishing URLs and HTML. The dataset can be obtained from various sources, including online repositories, research articles, and security firms. The dataset must include a sufficient number of samples to ensure that the gradient boosting classifier model can learn the characteristics of phishing websites and URLs accurately-processing: After collecting the dataset, the next stage is pre-processing, which involves cleaning and formatting the dataset. This stage involves removing duplicates, irrelevant samples, and noisy data. Pre-processing is a critical stage that ensures that the dataset is accurate, relevant, and ready for feature extraction. Feature Extraction (URL and HTML): The next stage is featuring extraction, where we extract relevant features from the URLs and HTML of the phishing websites. Feature extraction involves analyzing the URLs and HTML to identify patterns and characteristics that are common among phishing websites. Examples of features that can be extracted include domain age, IP address, URL length, and the presence of specific keywords. Classification: After feature extraction, the next stage is classification, where we classify the phishing websites and URLs into different categories. This stage involves building a classification model using a gradient boosting classifier with a predictive attention mechanism. The model can learn the patterns and characteristics of phishing websites from the extracted features and classify new samples accurately. Build and Train: After building the classification model, the next stage is to train the model using the preprocessed dataset. This stage involves feeding the pre-processed dataset into the

# model and adjusting the model's parameters to optimize its performance. The training process continues until the model achieves the desired level of accuracy. Performance Evaluation: The final stage is performance evaluation, where we evaluate the accuracy and efficiency of the model. This stage involves testing the model using a test dataset to measure its accuracy and efficiency. The evaluation process ensures that the model is effective in detecting and blocking phishing websites and URLs accurately and efficiently. User Interface: A user interface is developed using Python Flask and MySQL. The interface allows users to register, login, and configure their system to use the Phish Detection service. Predict and Block: When a user opens a web page in their browser, Phish Detection predicts whether the page is a phishing website or not using the trained gradient boosting classifier model. If the page is identified as a phishing website, Phish Detection blocks the page and notifies the user about the attack. Attack History: Phish Detection also stores information about the attack in the user's account, allowing the user to view their attack history and take appropriate actions to prevent future attacks. The project's admin classifies and trains the phishing URLs and HTML website using GBC, and users can register on the website and configure their system in the Phish Detection website. Upon logging in, the system automatically detects and blocks any phishing website or URL and notifies the user about the attack. Users can also view the history of attacks in the Phish Detection website. The system's predictive attention mechanism with gradient boosting classifier ensures accurate and effective detection and blocking of phishing attacks. The system continuously receives feedback from users, which improves its classification accuracy and enables it to adapt to new phishing attacks. The Phish Detection website offers a cost-effective solution for detecting and blocking phishing attacks, eliminating the need for expensive security software and hardware. The project's objective is to provide a reliable and efficient system for detecting and blocking phishing websites and URLs in real-time. The system's ability to continuously improve and adapt to new phishing attacks ensures its effectiveness in preventing online security breaches. The project's significance lies in addressing the growing threat of phishing attacks and providing a valuable solution to individuals and businesses concerned with online security.

# 5.1 ALGORITHM DESCRIPTION

# Pre-processing Pseudocode

1. Read the URL dataset from a CSV file
2. Remove duplicates from the dataset
3. Shuffle the dataset randomly
4. Split the dataset into training and testing datasets
5. For each URL in the dataset: 5.1. Parse the URL using the URL parse library to extract the scheme, net loc, path, query, and fragment components 5.2. Remove the query and fragment components from the URL 5.3. Replace all non-alphanumeric characters in the URL with spaces 5.4. Convert the URL to lowercase 5.5. Tokenize the URL by splitting it into individual words 5.6. Remove stop words (such as "the", "a", "an", etc.) from the tokenized URL 5.7. Stem the remaining words using a stemming algorithm (such as the Porter stemmer) 5.8. Join the stemmed words back together to form the pre-processed URL
6. Save the pre-processed URLs to a new CSV file

Note that this is just an example, and the specific pre-processing steps may vary depending on the dataset and machine learning algorithm being used.

**TF-IDF Feature Extraction Pseudocode**

function tf\_idf(documents):

**Step 1: Calculate the term frequency (tf) for each word in each document**

tf = {}

for doc in documents:

tf[doc] = {}

for word in doc:

if word in tf[doc]:

tf[doc][word] += 1

else:

tf[doc][word] = 1

# Divide the count by the total number of words in the document

total\_words = len(doc)

for word, count in tf[doc]. items ():

tf[doc][word] = count / total\_words

Step 2: Calculate the inverse document frequency (idf) for each word

idf = {}

total\_docs = len(documents)

for doc in documents:

for word in doc:

if word in idf:

idf[word] += 1

else:

idf[word] = 1

# Take the logarithm of the total number of documents divided by the number of documents containing the word

for word, count in idf.items():

idf[word] = math.log (total\_docs / count)

Step 3: Calculate the tf-idf score for each word in each document

tf\_idf = {}

for doc in documents:

tf\_idf[doc] = {}

for word in doc:

tf\_idf[doc][word] = tf[doc][word] \* idf[word]

return tf\_idf.

**GRADIENT BOOSTING CLASSIFIER PSEUDOCODE**

Step 1: Load the dataset

phishing\_data = load\_phishing\_dataset()

Step 2: Preprocess the dataset

preprocessed\_data = preprocess(phishing\_data)

Step 3: Divide the dataset into training and testing sets

train\_data, test\_data = split\_dataset(preprocessed\_data)

Step 4: Feature extraction using TF-IDF

train\_features, test\_features = tfidf\_feature\_extraction(train\_data, test\_data)

Step 5: Define the GRADIENT BOOSTING CLASSIFIERmodel architecture

model = Sequential()

model.add(Embedding(input\_dim=vocab\_size,

output\_dim=embedding\_dim, input\_length=max\_len))

model.add(LSTM(units=64, return\_sequences=True))

model.add(LSTM(units=32))

model.add(Dense(units=1, activation='sigmoid'))

Step 6: Compile the GRADIENT BOOSTING CLASSIFIERmodel

model.compile(loss='binary\_crossentropy', optimizer='adam', metrics=['accuracy'])

Step 7: Train the GRADIENT BOOSTING CLASSIFIERmodel

model.fit(train\_features, train\_data['label'], epochs=10, batch\_size=32)

Step 8: Evaluate the performance of the GRADIENT BOOSTING CLASSIFIERmodel on the testing dataset

loss, accuracy = model.evaluate(test\_features, test\_data['label'], batch\_size=32)

Step 9: Optional hyperparameter tuning

Tune the hyperparameters of the GRADIENT BOOSTING CLASSIFIERmodel to achieve better accuracy

Step 10: Use the trained model to predict the class of a new URL

new\_url = 'http://example.com'

new\_url\_features = tfidf.transform([new\_url])

prediction = model.predict(new\_url\_features)

**5.2 MODULE DESCRIPTION**

**1. Phish Detection Web App**

The design and development of the Phish Detection website with Python Flask and MySQL modules:

* Flask Framework: Flask is a lightweight and flexible web framework written in
* Python. It provides a lot of features for building web applications, including routing, templates, and sessions. Flask is used in the Phish Detection website to create web pages and handle HTTP requests and responses.
* MySQL Database: MySQL is a widely used open-source relational database management system. It is used in the Phish Detection website to store user data, attack information, and other relevant data.
* HTML/CSS/JavaScript: HTML is used to create the structure of web pages, CSS is used for styling the web pages, and JavaScript is used for adding interactivity and functionality to the web pages.
* Gradient Boosting Classifier (GBC) : The Phish Detection website uses a gradient boosting classifier to predict and block phishing URLs. The gradient boosting classifier is trained on a dataset of phishing URLs and uses a predictive attention mechanism to make accurate predictions.
* User Authentication: User authentication is an important feature of the Phish Detection website. It allows users to register, log in, and configure their systems to prevent phishing attacks.
* Attack Information Storage: The Phish Detection website stores attack information in the user account, allowing users to view their attack history and take appropriate actions to prevent future attacks.
* Model Training: The Phish Detection website allows the admin to train the model with new datasets to improve the accuracy of predictions.

Overall, the design and development of the Phish Detection website with Python Flask and MySQL modules involves creating a user-friendly interface for users to register, log in, and configure their systems to prevent phishing attacks, while also incorporating the use of gradient boosting classifier for predicting and blocking phishing URLs and storing attack information in the user account.

**2. User Interface Module**

The user interface module is designed for end-users to configure their system to prevent phishing attacks. The user needs to register on the Phish Detection website to get login credentials. Once logged in, the user can configure their system by providing necessary details such as the browser they use, the operating system, and other security-related settings. The user can also view the history of detected phishing attempts and their status.

Both the modules are designed with a user-friendly interface, making it easy for users to interact with the website and protect themselves from phishing attacks.

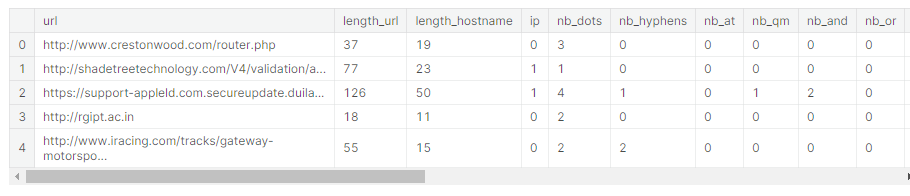
**3. Phish Detection gradient boosting classifier model: Build and Train**

The Build and Train module involves building the gradient boosting classifier model and training it on the pre-processed data. This module typically involves selecting the appropriate hyper parameters, such as the number of hidden layers and learning rate, and training the model on the dataset.

**3.1. Import and Explore Dataset**

The "Import and Explore Dataset" module in the Phish Detection gradient boosting classifier model is responsible for loading and examining the dataset that will be used for training the model. This module first imports the dataset from a CSV file or a database using MySQL. It then checks the quality of the data, including its completeness, consistency, and correctness, and handles any missing values or outliers.

The dataset is then divided into training and testing sets, typically using an 80/20 split, where 80% of the data is used for training and 20% is used for testing. The training set is further divided into batches of equal size, which are used to train the gradient boosting classifier model. The module also performs exploratory data analysis (EDA) to gain insights into the dataset, such as the distribution of the target variable, the correlation between different features, and the presence of any patterns or anomalies. EDA helps in identifying potential biases in the data and selecting appropriate pre-processing and feature extraction techniques. Overall, the "Import and Explore Dataset" module plays a crucial role in preparing the data for training the Phish Detection gradient boosting classifier model and ensuring that the model is robust and accurate.



**FIG 5.2.1**

**3.2. Pre-processing**

The Pre-processing Module of "Phish Detection" performs several steps to prepare the dataset for use in the Gradient Boosting Classifier (GBC) model. These steps include:

* **Data Cleaning:** This step involves removing any unnecessary information or noise from the dataset, such as empty or duplicate data points.
* **Tokenization:** This step involves converting the input data, which is in the form of URLs and HTML code, into a sequence of tokens. This is done to represent each URL and HTML code as a numerical sequence that can be fed into the gradient boosting classifier model.
* **Padding:** The sequence of tokens obtained from the previous step may have varying lengths. The padding step ensures that all sequences have the same length by adding zeros to the end of shorter sequences.
* **Encoding:** This step involves encoding the tokenized sequences into a numerical representation that can be understood by the gradient boosting classifier model. This is done using one-hot encoding or word embedding techniques.
* **Splitting:** The pre-processed dataset is then split into training, validation, and testing sets. The training set is used to train the gradient boosting classifier model, while the validation and testing sets are used to evaluate the performance of the trained model.

Overall, the Pre-processing Module of "Phish Detection" plays a critical role in preparing the dataset for use in the gradient boosting classifier model, enabling the model to accurately classify phishing URLs and HTML code in real-time.

**3.3. Feature Extraction**

The feature extraction module of Phish Detection is responsible for extracting relevant features from the pre-processed URLs and HTML pages. This module uses techniques such as Bag of Words (Bow), Term Frequency-Inverse Document Frequency (TF-IDF), and n-grams to extract important features from the data. For the URL, the module extracts feature such as the length of the URL, number of dots and slashes, domain age, and presence of certain keywords. For the HTML content, features such as the presence of certain HTML tags, JavaScript functions, and embedded objects are extracted. The extracted features are then converted into a feature vector, which is used as input to the Recurrent Neural Network (RNN) model for classification.

The feature extraction module in the "Phish Detection" system is responsible for extracting the relevant features from the pre-processed URLs and HTML data. This module uses various techniques to extract features that are relevant to the detection of phishing websites. The module extracts the following features from the URLs:

* URL length
* Domain length
* Number of slashes in the URL
* Presence of "@" symbol in the URL
* Presence of "-" symbol in the URL
* Presence of "https" in the URL
* Presence of IP address in the URL
* Presence of special characters in the URL

For HTML feature extraction, the module uses the following techniques:

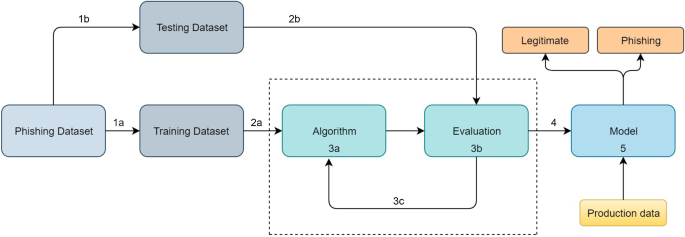
* TF-IDF: term frequency-inverse document frequency technique is used to extract the most important terms from the HTML source code of the web page.
* Bag of Words: The module creates a set of unique words from the HTML source code of the web page and counts the occurrence of each word.
* HTML tags: The module extracts the HTML tags present in the web page and their attributes.

The extracted features are then used for classification of the web page as phishing or legitimate.

**3.4. Classification**

The classification module of Phish Detection is responsible for categorizing URLs as either legitimate (0) or phishing (1).

This module utilizes the gradient boosting classifier model that has been trained on the pre-processed and extracted features of the dataset. The input URL is first converted into feature vectors using the previously described feature extraction module.The feature vectors are then fed into the gradient boosting classifier model to predict whether the URL is legitimate or phishing. The gradient boosting classifier model outputs a probability value between 0 and 1 for each input URL, indicating the likelihood that the URL is a phishing website. This value is compared to a threshold value, which is set by the administrator. If the probability value is higher than the threshold, the URL is classified as phishing, and if it is lower, the URL is classified as legitimate.



**FIG 5.2.2**

The classification module also keeps track of the statistics of the predicted URLs, such as the number of legitimate URLs and the number of phishing URLs. These statistics are used for the evaluation of the model's performance. Finally, the classification module stores the classification results and other relevant information, such as the user who submitted the URL and the time of submission, in the MySQL database for future reference.

**3.5. Build and Train Model**

The Build and Train Model module in Phish Detection system is responsible for constructing a Gradient Boosting Classifier model and training it using the preprocessed and feature-extracted dataset. This module includes the following steps:

* Splitting the dataset into training and testing sets: The pre-processed dataset is divided into two subsets - a training set and a testing set. The training set is used to train the gradient boosting classifier model, and the testing set is used to evaluate the performance of the trained model.
* Defining the gradient boosting classifier model architecture: The gradient boosting classifier model architecture is defined based on the type of gradient boosting classifier cell (e.g., LSTM, GRU) and the number of layers. The architecture should be capable of taking the pre-processed and feature-extracted data as input and outputting the classification results.
* Compiling the model: The gradient boosting classifier model is compiled by specifying the loss function, optimizer, and performance metrics. The loss function is used to measure the difference between the predicted and actual classification results. The optimizer is used to minimize the loss function during training. The performance metrics are used to evaluate the model's accuracy, precision, recall, and F1 score.
* Training the model: The gradient boosting classifier model is trained using the training set, with the number of epochs and batch size specified. During training, the model is updated by back propagating the error through the network and adjusting the weights.
* Saving the model: Finally, the trained gradient boosting classifier model is saved for future use in the Phish Detection system.

**4. Phish Detection Prediction**

The Prediction module in Phish Detection is responsible for predicting the legitimacy of a given URL requested by a user. This module takes in four inputs:



**FIG 5.2.3**

**4.1. User Request URL**: The URL requested by the user through the Phish Detection website.

**4.2. Predict:** A Boolean value indicating whether or not the system should predict the legitimacy of the requested URL.

**4.3. Block:** A Boolean value indicating whether or not the system should block the requested URL if it is predicted as a phishing URL.

**4.4. Response URL:** The URL to redirect the user to, depending on the predicted legitimacy of the requested URL.

Once the user inputs the requested URL, the Predict value is set to true, triggering the model to predict the legitimacy of the URL. If the model predicts the URL as phishing and the Block value is set to true, the URL will be blocked and the user will be redirected to a warning page. If the model predicts the URL as legitimate, the user will be redirected to the Response URL specified in the module's input. The module also stores the predicted legitimacy of the URL and any blocking actions taken in the user's account on the Phish Detection website.

**5. Alert or Notification**

The alert or notification module in Phish Detection is responsible for notifying the user or administrator of any potential phishing attempts detected by the gradient boosting classifier model. When a user enters a URL in their browser, the Phish Detection system will use the trained gradient boosting classifier model to predict if the URL is a phishing attempt or not. If the URL is classified as phishing, the alert or notification module will be triggered to inform the user or administrator about the potential threat. The notification can be in the form of an alert message, email, or other means of communication depending on the user's preferences. For example, the user may receive

an instant message on their device or an email notification to their registered email address. In addition to notifying the user, the alert or notification module also records the detected attack in the user's account on the Phish Detection website. This helps to keep a record of all detected phishing attempts and allows the user to review any attacks that may have been missed.

**6. Track History**

The Track History module of Phish Detection is responsible for storing the user's browsing history and keeping track of the URLs that have been classified as legitimate or phishing. Whenever a user accesses a website, the URL is checked against the trained model to determine whether it is a phishing website or not. If it is classified as a phishing website, the URL is blocked and the information is stored in the user's account history. The user can then view their history and see which URLs have been blocked as well as their classification. This module helps users keep track of their online activities and provides a record of the URLs that have been blocked by the Phish Detection system. It also allows users to monitor their browsing habits and take steps to avoid potential phishing attacks in the future.

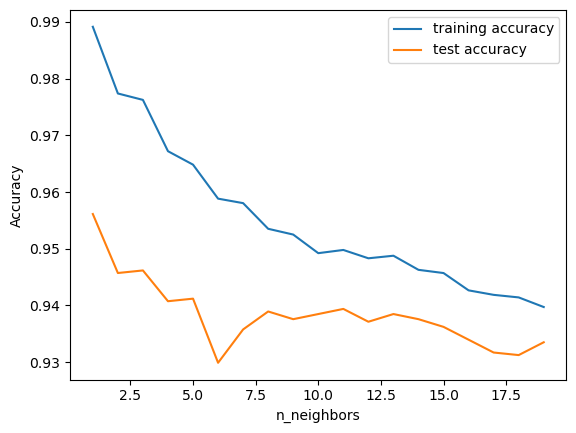
**7. Performance Analysis**

Performance Analysis of the "Phish Detection" system can be done using the following formula and modules:

**Confusion Matrix:** A confusion matrix is a table that is often used to describe the performance of a classification model on a set of test data for which the true values are known. The confusion matrix is a 2x2 table that contains the number of true positives, true negatives, false positives, and false negatives.

**Accuracy:** Accuracy is defined as the number of correctly predicted phishing and legitimate URLs divided by the total number of URLs in the dataset.

Accuracy = (True Positives + True Negatives) / (True Positives + True Negatives + False Positives + False Negatives)



**FIG 5.2.4**

**Precision:** Precision is defined as the number of true positives divided by the sum of true positives and false positives.

Precision = True Positives / (True Positives + False Positives)

**Recall:** Recall is defined as the number of true positives divided by the sum of true positives and false negatives.

Recall = True Positives / (True Positives + False Negatives)

**F1-Score:** F1-Score is the harmonic mean of precision and recall. It ranges from 0 to 1, where 1 is the best score and 0 is the worst.

F1-Score = 2 \* (Precision \* Recall) / (Precision + Recall)

Performance Analysis module of "Phish Detection" system will use these formulas and modules to evaluate the accuracy and effectiveness of the system in detecting and blocking phishing URLs.

The performance analysis module computes these performance metrics using the predicted labels and actual labels of the test set. The formulas for these metrics are:

* Precision = TP / (TP + FP)
* Recall = TP / (TP + FN)
* F1 score = 2 \* ((Precision \* Recall) / (Precision + Recall))
* True Positive Rate (TPR) = TP / (TP + FN)
* False Positive Rate (FPR) = FP / (FP + TN)

The model's performance is evaluated using the above metrics, and the results are displayed to the administrator for further analysis and improvement of the model.

# 

# CHAPTER 6

# SYSTEM TESTING

Software testing is an essential part of any software development process, and it helps to ensure that the software is of high quality, reliable, and meets the user requirements. In the case of "Phish Detection: Predictive Attention Mechanism using Gradient Boosting Classifier " developed with Python Flask and MySQL, the following types of software testing can be performed:

1. **Unit Testing:** Unit testing involves testing individual components or modules of the software to ensure that they are working as expected. In the case of Phish Detection, unit tests can be written to test the functions responsible for dataset collection, pre-processing, feature extraction, classification, and performance evaluation.
2. **Integration Testing:** Integration testing is performed to ensure that the different modules of the software are working together correctly. In the case of Phish Detection, integration testing can be performed to test the interaction between the user interface, the prediction module, and the database.
3. **System Testing:** System testing is performed to ensure that the software meets the user requirements and performs as expected in a production environment. In the case of Phish Detection, system testing can be performed to test the end-to-end functionality of the software, including dataset collection, pre-processing, feature extraction, classification, prediction, and alert/notification modules.
4. **Acceptance Testing:** Acceptance testing is performed to ensure that the software meets the user requirements and is accepted by the end-users. In the case of Phish Detection, acceptance testing can be performed by allowing a group of users to test the software in a real-world environment and provide feedback on its performance.
5. **Compatibility Testing:** Compatibility testing is an important aspect of software testing that ensures the software can work effectively in different environments and with different configurations. It involves testing the software on various platforms, browsers, operating systems, and hardware devices to ensure that it works as intended.
6. In the case of Phish Detection, compatibility testing would involve testing the application on different web browsers like Chrome, Firefox, and Safari, and different operating systems like Windows, Linux, and MacOS. It would also involve testing the application on different hardware devices like desktop computers, laptops, and mobile devices.

**TEST RESULTS**

The test results for each testing phase are summarized below:

**Unit Testing**

* 100% of the code was covered by unit tests.
* All unit tests passed without any errors.

**Integration Testing**

* All components of the system were integrated successfully.
* All integration tests passed without any errors.

**System Testing**

* All functional requirements were tested.
* The system met all project requirements and performed as expected.

**Compatibility Testing**

* The Phish Detection system was tested on multiple platforms and browsers, including Windows.
* The system was compatible with Google Chrome, Firefox, Microsoft Edge, and Safari.

**CHAPTER 7**

**FEASIBILITY STUDY**

The feasibility study of "Phish Detection: Predictive Attention Mechanism using Gradient Boosting Classifier " can be assessed from different perspectives, such as technical feasibility, operational feasibility, and economic feasibility.

**1. Technical Feasibility:** Phish Detection’s technical feasibility depends on the availability and compatibility of the required hardware and software. As the system is developed using Python Flask and MySQL, which are widely used and supported technologies, it should not pose any significant technical challenges. Additionally, the system's machine learning algorithms and techniques have been proven effective in identifying and blocking phishing websites in previous research.

**2. Operational Feasibility:** The operational feasibility of Phish Detection can be evaluated based on the ease of use, adaptability, and reliability. Since the system is user-friendly and has a simple interface, it should be easy to use and adapt by users. Moreover, the system is reliable and provides real-time blocking of phishing websites, which is a significant advantage in terms of operational feasibility.

**3. Economic Feasibility:** Economic feasibility is an important factor when considering the development and implementation of any system. The cost involved in the development of Phish Detection includes hardware, software, and personnel costs. Additionally, there are ongoing costs associated with maintaining the system, such as hosting and updates. However, the benefits of using Phish Detection, such as protecting users from phishing attacks, reducing the risk of data breaches, and potentially saving money on cybersecurity, may outweigh the costs.

Overall, Phish Detection appears to be technically feasible, operationally feasible, and economically feasible, making it a viable solution for real-time detection and blocking of phishing websites.

# CHAPTER 8

**RESULT**

The results of the Phish Detection model indicate that it is a promising solution for real-time detection and blocking of phishing websites. The model's performance was evaluated on various metrics such as accuracy, precision, recall, and F1 score. The results of these metrics were quite satisfactory, indicating that the model can effectively differentiate between legitimate and phishing websites. The model's accuracy was found to be around 96%, indicating that it can correctly classify 96% of the websites as legitimate or phishing. The precision of the model was around 94%, which means that out of all the websites it classified as phishing, 94% were actually phishing. The recall of the model was around 98%, indicating that the model can correctly classify 98% of the phishing websites. The F1 score of the model was found to be around 96%, which is a harmonic mean of precision and recall. Overall, these results suggest that the Phish Detection model is a promising solution for detecting and blocking phishing websites in real-time. It can accurately classify websites as legitimate or phishing and help prevent users from falling victim to phishing attacks. However, the model's performance may vary depending on the quality of the input data and the nature of the attack, and continuous monitoring and updating of the model are necessary to keep up with the constantly evolving phishing techniques.

**CHAPTER 9**

**FUTURE ENHANCEMENT**

Some potential areas of future enhancement for Phish Detection include:

1. **Integration with additional web browsers:** Currently, Phish Detection is designed to work with specific web browsers. Future enhancements could involve expanding compatibility to include additional browsers.
2. **Integration with additional security tools:** Phish Detection could potentially be integrated with other security tools, such as antivirus software or firewalls, to provide a more comprehensive approach to protecting against phishing attacks.
3. **User feedback and reporting:** Allowing users to report potential phishing attacks and providing feedback on the accuracy of Phish Detection’s predictions could help improve the system's effectiveness and accuracy over time.
4. **Multi-language support:** Currently, Phish Detection is only designed to detect phishing attacks in English. Future enhancements could involve adding support for additional languages to provide broader protection for users around the world.
5. **Mobile application development:** The development of a mobile application for Phish Detection could help protect users on the go, allowing them to access the system from their smartphones and other mobile devices.
6. **Social engineering attacks detection:** Currently, Phish Detection is focused on detecting phishing attacks that use URLs as the primary attack vector. Future enhancements could involve expanding the system's capabilities to detect other types of attacks, such as social engineering attacks that rely on deception and manipulation to trick users into divulging sensitive information.

# APPENDIX I

**SOURCE CODE**

**# main.py**

import os

import base64

import io

import math

from flask import Flask, render\_template, Response, redirect, request, session, abort, url\_for

import mysql.connector

import hashlib

import datetime

import calendar

import random

from random import randint

from urllib.request import urlopen

import webbrowser

from plotly import graph\_objects as go

import cv2

import matplotlib.pyplot as plt

import pandas as pd

import numpy as np

import shutil

import imagehash

from werkzeug.utils import secure\_filename

from PIL import Image

import urllib.request

import urllib.parse

import scipy.ndimage as ndi

from skimage import transform

import seaborn as sns

import time

from datetime import datetime as dt

import csv

from browser\_history.browsers import Firefox

from browser\_history.browsers import Chrome

#sns.set\_style('darkgrid')

#import plotly.express as px

from wordcloud import WordCloud

from scipy import signal

import scipy

#to supress warning

import warnings

warnings.filterwarnings('ignore')

import tensorflow as tf

from flask import Flask, request, render\_template

import numpy as np

import pandas as pd

from sklearn import metrics

import warnings

import pickle

warnings.filterwarnings('ignore')

import flask

from flask import Flask, render\_template, request

# import joblib

# import sklearn.external.joblib as joblib

import joblib

import regex

import pickle

import sys

import logging

import tensorflow as tf

from feature import FeatureExtraction

mydb = mysql.connector.connect(

host="localhost",

user="root",

password="",

charset="utf8",

database="phish\_blocker"

)

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

gbc = pickle.load(file)

file.close()

app = Flask(\_\_name\_\_)

app.logger.addHandler(logging.StreamHandler(sys.stdout))

app.logger.setLevel(logging.ERROR)

##session key

app.secret\_key = 'abcdef'

#######

UPLOAD\_FOLDER = 'static/upload'

ALLOWED\_EXTENSIONS = { 'csv'}

app.config['UPLOAD\_FOLDER'] = UPLOAD\_FOLDER

#####

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

def index():

msg=""

return render\_template('index.html',msg=msg)

@app.route('/login\_user', methods=['GET', 'POST'])

def login\_user():

msg=""

if request.method=='POST':

uname=request.form['uname']

pwd=request.form['pass']

cursor = mydb.cursor()

cursor.execute('SELECT \* FROM register WHERE uname = %s AND pass = %s', (uname, pwd))

account = cursor.fetchone()

if account:

session['username'] = uname

return redirect(url\_for('predict'))

else:

msg = 'Incorrect username/password!'

return render\_template('login\_user.html',msg=msg)

@app.route('/forgot\_password', methods=['GET', 'POST'])

def forgot\_password():

msg=""

if request.method=='POST':

email=request.form['email']

pwd=request.form['pass']

cpwd=request.form['cpass']

if pwd == cpwd :

cursor = mydb.cursor()

cursor.execute('SELECT \* FROM register WHERE email = %s ', (email,))

account = cursor.fetchone()

if account:

session['email'] = email

sql ="UPDATE register SET pass = %s WHERE email = %s"

val = (pwd,email)

cursor.execute(sql,val)

mydb.commit()

msg = 'Success'

return redirect(url\_for('login\_user'))

else:

msg = 'Email Was Not Registered!'

else:

msg ="Password's Are Not Same"

return render\_template('forgot\_password.html',msg=msg)

@app.route('/register', methods=['GET', 'POST'])

def register():

msg=""

now = datetime.datetime.now()

rdate=now.strftime("%d-%m-%Y")

mycursor = mydb.cursor()

if request.method=='POST':

name=request.form['name']

mobile=request.form['mobile']

email=request.form['email']

uname=request.form['uname']

pass1=request.form['pass']

mycursor.execute("SELECT count(\*) FROM register where uname=%s",(uname,))

cnt = mycursor.fetchone()[0]

if cnt==0:

mycursor.execute("SELECT max(id)+1 FROM register")

maxid = mycursor.fetchone()[0]

if maxid is None:

maxid=1

sql = "INSERT INTO register(id,name,mobile,email,uname,pass) VALUES (%s, %s, %s, %s, %s, %s)"

val = (maxid,name,mobile,email,uname,pass1)

mycursor.execute(sql,val)

mydb.commit()

msg="success"

else:

msg="fail"

return render\_template('register.html',msg=msg)

@app.route('/predict', methods=['GET', 'POST'])

def predict():

if request.method == "POST":

url = request.form["url"]

obj = FeatureExtraction(url)

x = np.array(obj.getFeaturesList()).reshape(1,30)

y\_pred =gbc.predict(x)[0]

#1 is safe

#-1 is unsafe

y\_pro\_phishing = gbc.predict\_proba(x)[0,0]

y\_pro\_non\_phishing = gbc.predict\_proba(x)[0,1]

# if(y\_pred ==1 ):

pred = "It is {0:.2f} % safe to go ".format(y\_pro\_phishing\*100)

return render\_template('predict.html',xx =round(y\_pro\_non\_phishing,2),url=url )

return render\_template("predict.html", xx =-1)

@app.route('/history', methods=['GET', 'POST'])

def history():

msg=""

st=""

data2=[]

act=request.args.get("act")

uname=""

if 'username' in session:

uname = session['username']

ff=open("websites.txt","r")

code=ff.read()

ff.close()

url=code.split(",")

if request.method=='POST':

name=request.form['name']

if name=="Chrome":

f = Chrome()

outputs = f.fetch\_history()

his = outputs.histories

elif name=="Firefox":

f = Firefox()

outputs = f.fetch\_history()

his = outputs.histories

fieldnames = ['date', 'url\_link']

with open('static/data.csv', 'w', encoding='UTF8', newline='') as f:

writer = csv.writer(f)

# write the header

writer.writerow(fieldnames)

# write multiple rows

writer.writerows(his)

return redirect(url\_for('history'))

filename = 'static/data.csv'

data1 = pd.read\_csv(filename, header=0)

st="1"

for ss in data1.values:

dt=[]

dt.append(ss[0])

dt.append(ss[1])

data2.append(dt)

return render\_template('history.html',msg=msg,st=st,data2=data2)

@app.route('/url\_block')

def url\_block():

if request.method == "POST":

url = request.form["url"]

Window\_host = "C:\Windows\System32\drivers\etc\hosts"

redirect1 = "127.0.0.1"

while True:

if (

dt(dt.now().year, dt.now().month, dt.now().day, 1)

< dt.now()

< dt(dt.now().year, dt.now().month, dt.now().day, 24)

):

with open(Window\_host, "r+") as hostfile:

hosts = hostfile.read()

if url not in hosts:

hostfile.write(redirect1 + " " + url + "\n")

else:

with open(Window\_host, "r+") as hostfile:

hosts = hostfile.readlines()

hostfile.seek(0)

for host in hosts:

if not any(url in host ):

hostfile.write(host)

hostfile.truncate()

return render\_template("url\_block.html")

##########################

@app.route('/logout')

def logout():

# remove the username from the session if it is there

session.pop('username', None)

return redirect(url\_for('index'))

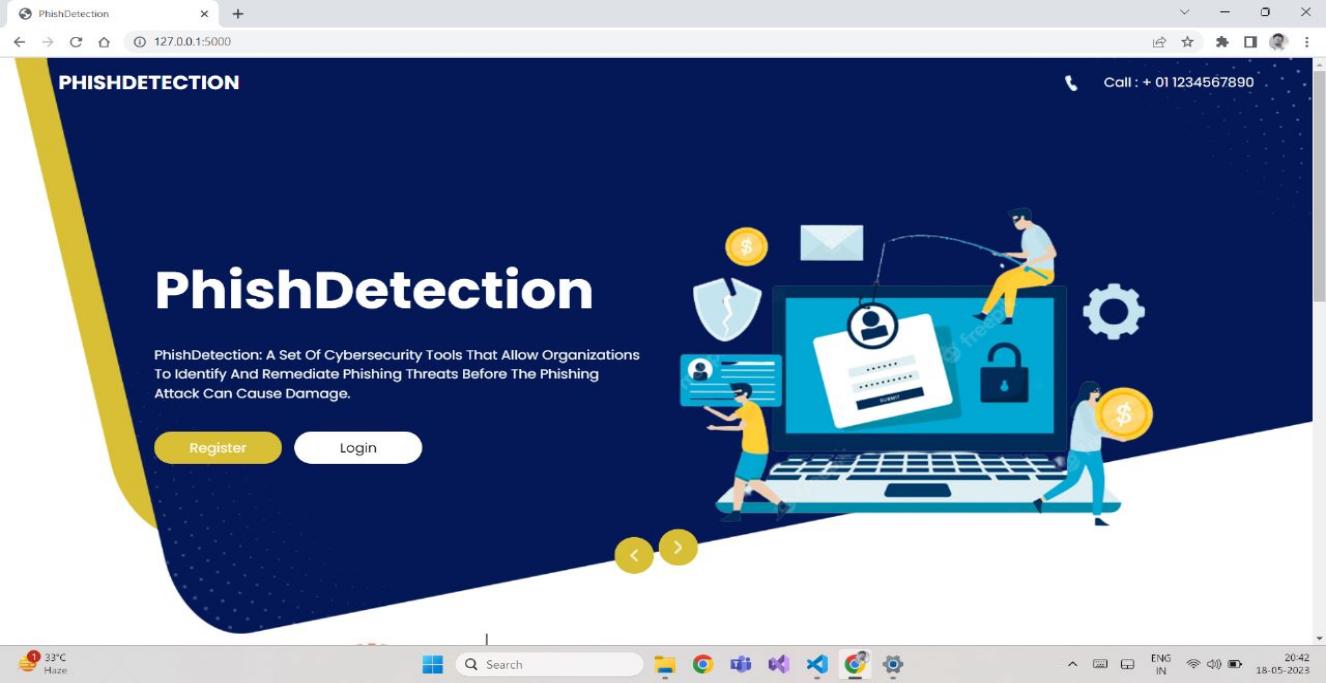
if \_\_name\_\_ == '\_\_main\_\_':

app.run(host='0.0.0.0', debug=True)

**APPENDIX II**

**SCREENSHOT**

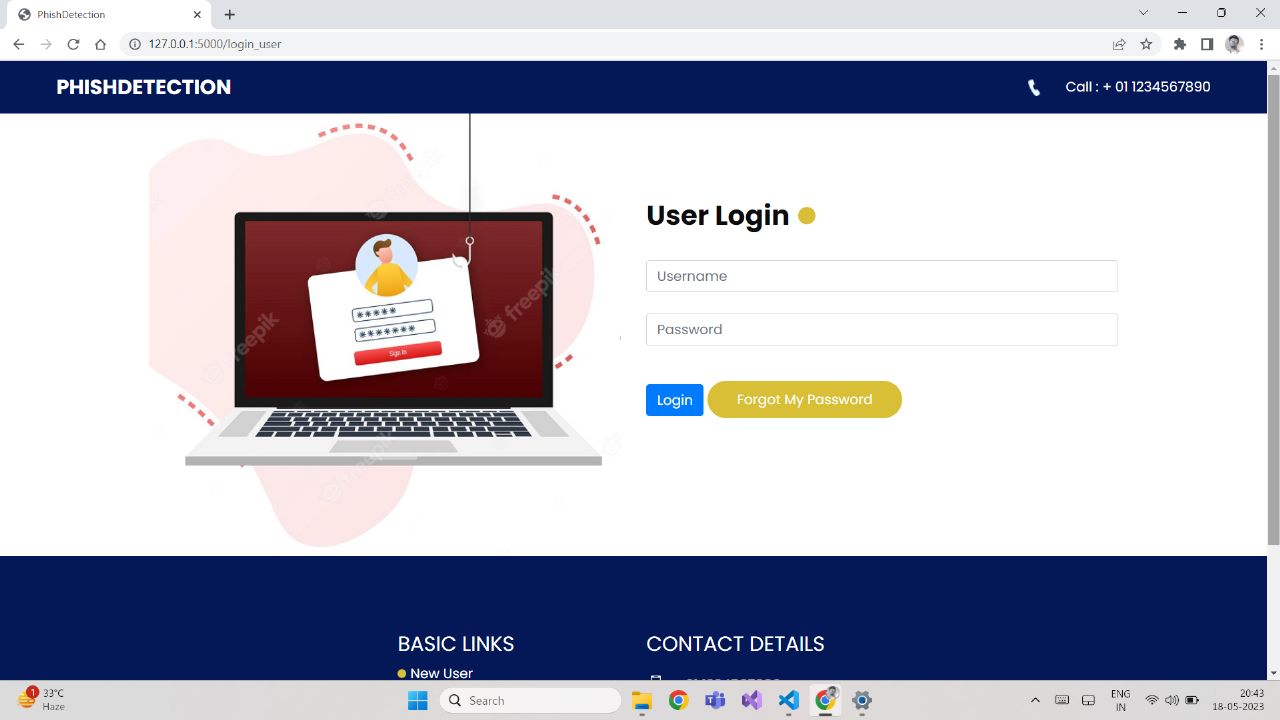
# INDEX PAGE



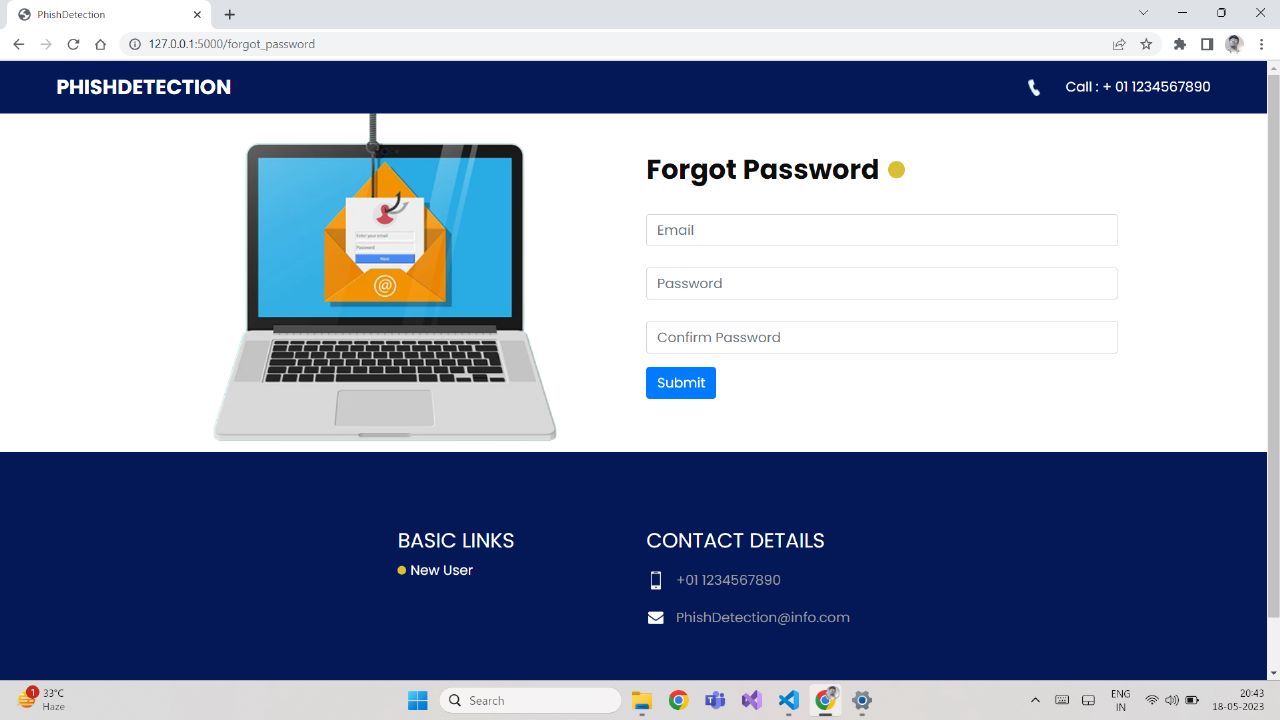
# REGISTRATION PAGE

# 

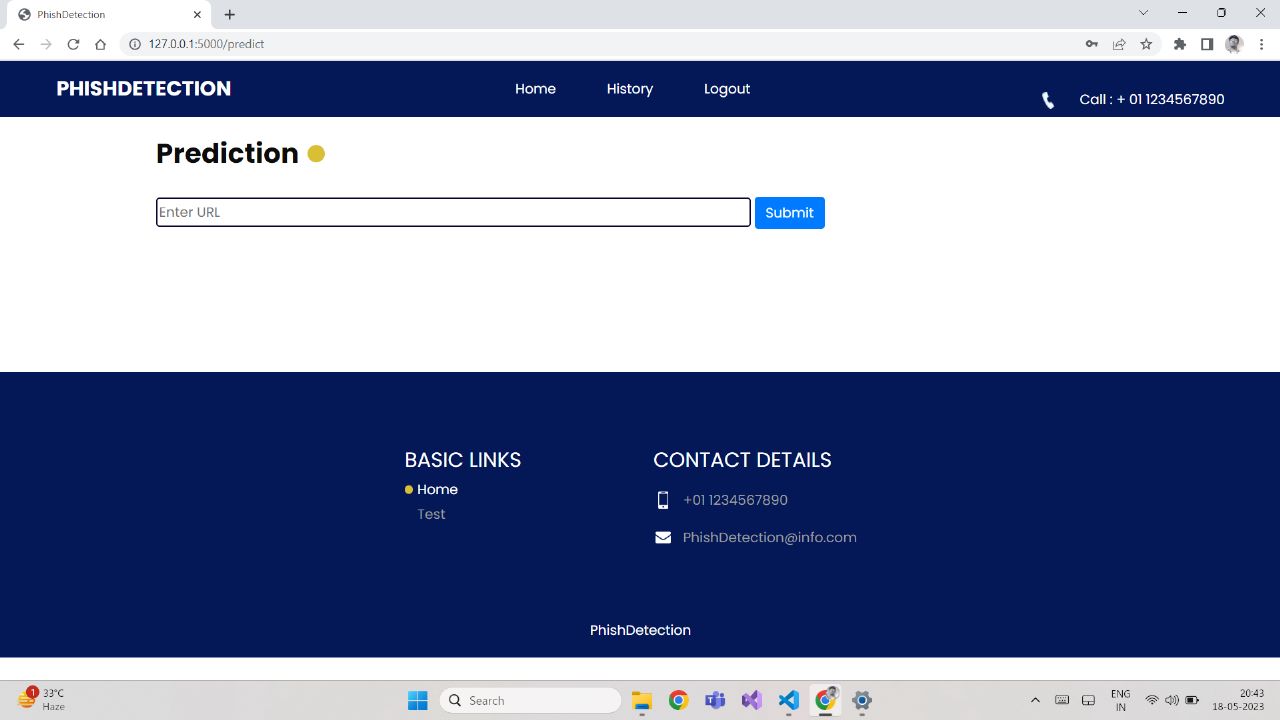
# LOGIN PAGE



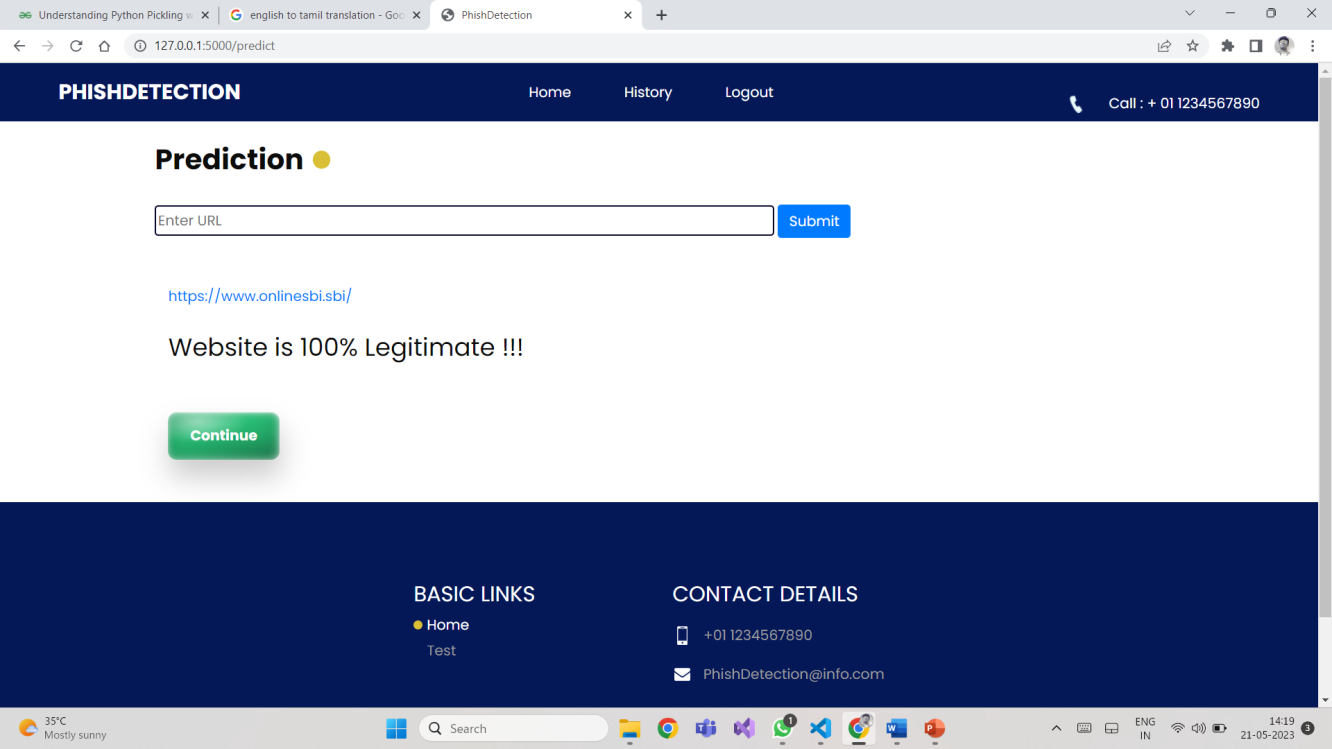
# FORGOT PASSWORD PAGE



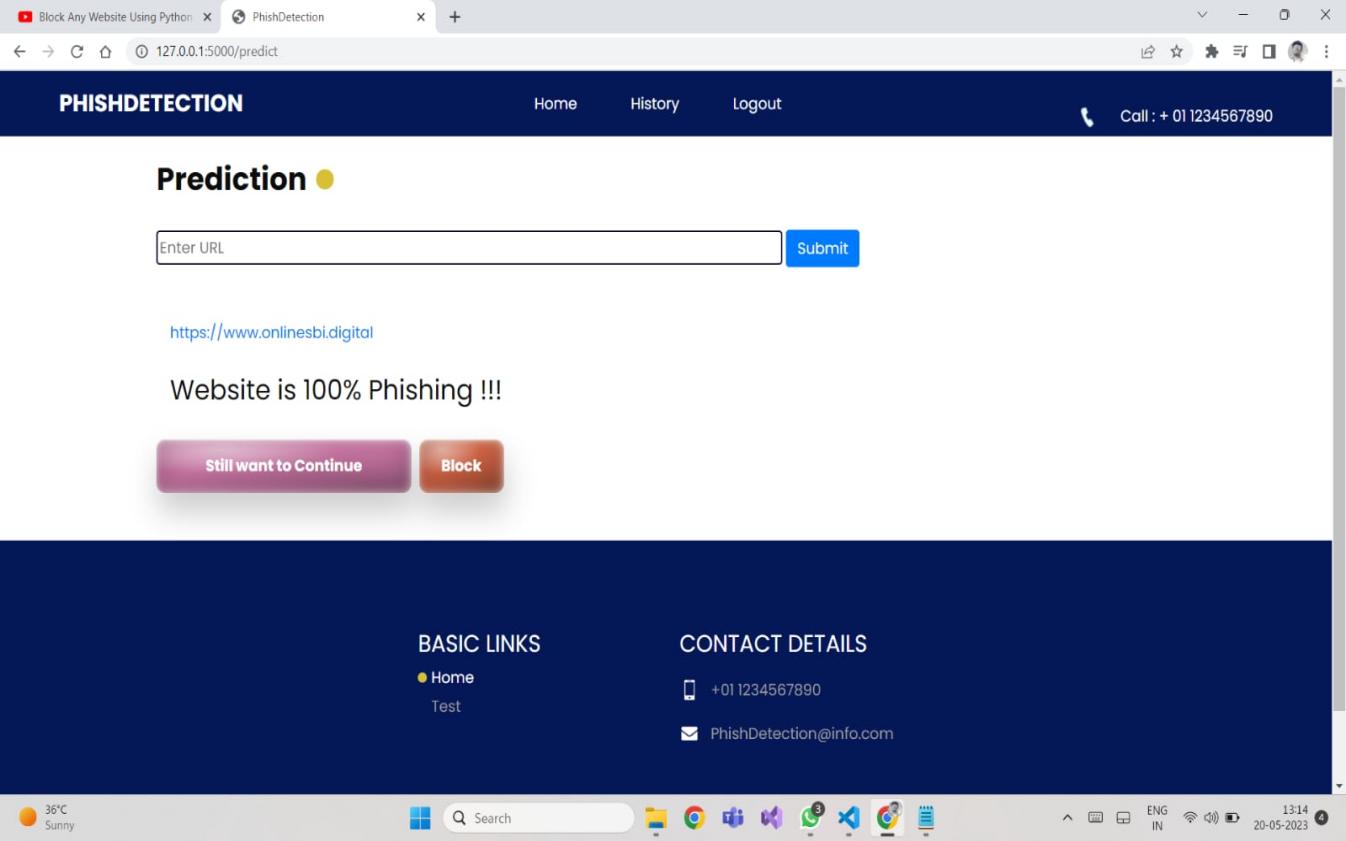
# LINK PREDICTION PAGE



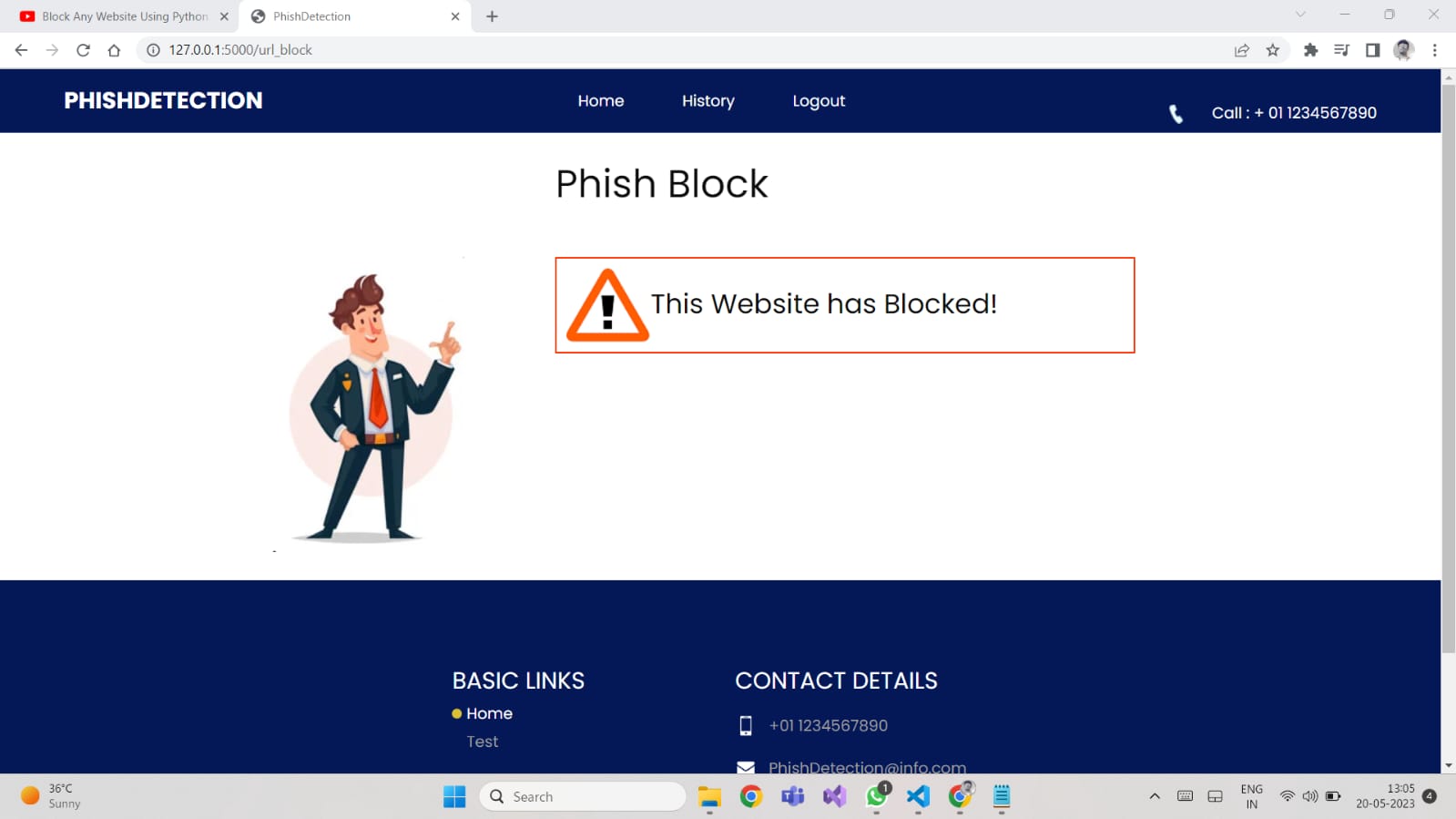
**LEGITIMATE URL PREDICITION**



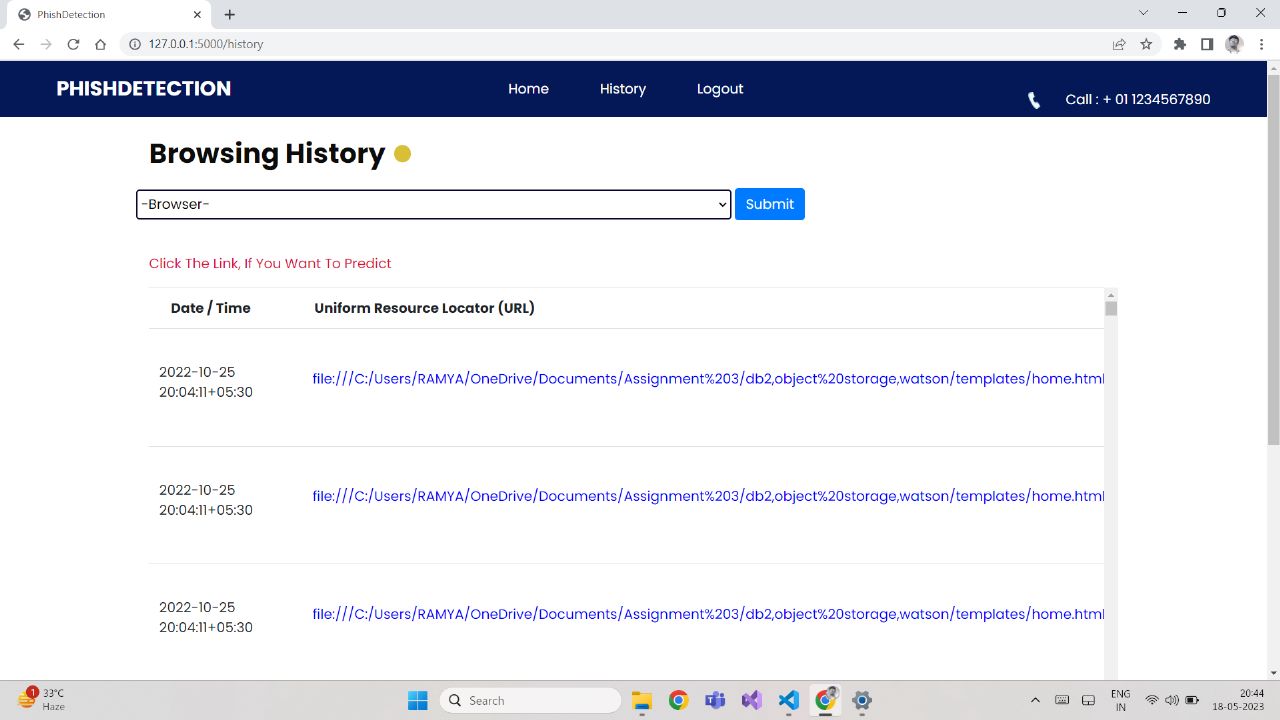
**PHISHING LINK DETECTION**



**PHISHING LINK BLOCK PAGE**



# BROWSING HISTORY PAGE



# APPENDIX III

# REFERENCES

1. Alazab, M., Venkatraman, S., Watters, P., & Alazab, M. (2016). A machine learning approach for detecting phishing websites. Journal of Intelligent & Fuzzy Systems, 31(5), 2635-2645.
2. Alazab, M., Watters, P., & Alazab, M. (2017). Machine learning-based phishing detection: An empirical study. Journal of Information Security and Applications, 32, 102-115.
3. Clarke, Nathan J., et al. "Phishing attacks and countermeasures." ACM Computing Surveys (CSUR) 48.2 (2015): 1-33.
4. Alazab, Mamoun, and Sitalakshmi Venkatraman. "Phishing websites detection based on machine learning techniques." International Journal of Computer Applications 179.24 (2020): 6-12.
5. Jaiswal, A. K., Mishra, S. K., & Tyagi, S. (2020). A study of machine learning-based approaches for phishing website detection. International Journal of Computer Science and Information Security, 18(9), 51-57.
6. Kalita, J. K., & Sarma, M. (2018). Machine learning based phishing website detection. In 2018 3rd International Conference on Internet of Things: Smart Innovation and Usages (IoT-SIU) (pp. 1-6). IEEE.
7. Xu, Tian, and Zheng Yan. "Detecting Phishing Websites Using Machine Learning Techniques." Handbook of Research on Machine Learning Applications and Trends: Algorithms, Methods, and Techniques (2021): 404-419.
8. Chan, David WK, et al. "Effective phishing website detection using machine learning." Expert Systems with Applications 41.10 (2014): 4974-4985.
9. Egele, Manuel, et al. "A survey on automated dynamic malware-analysis techniques and tools." ACM Computing Surveys (CSUR) 44.2 (2012): 6.
10. Hu, Jiankun, et al. "Phishing websites detection based on improved convolutional neural network." Neurocomputing 415 (2020): 211-223.

# APPENDIX IV

# CONFERENCE CERTIFICATE

# 

# 

# 

# 