WEB PHISHING DETECTION

A PROJECT REPORT

Submitted by

Team ID: PNT2022TMID50410

KALAI SELVI G
ANUSHYA M
PAUL RESHMIN M
KARTHIGA DEVI K

OF

INFORMATION TECHNOLOGY

PSN ENGINEERING COLLEGE, TIRUNELVELI - 627152

ANNA UNIVERSITY: CHENNAI 600 025

Project Report Format

CHAPTER .NO **TITTLE**

CHAPTER 1

INTRODUCTION

1.1Project Overview

The criminals, who want to obtain sensitive data, first create unauthorized replicas of a real website and e-mail. The e-mail will be created using logos and slogans of a legitimate company. The nature of website creation is one of the reasons that the Internet has grown so rapidly as a communication medium. Phisher then send the "spoofed" e-mails to as many people as possible in an attempt to lure them into the scheme. When these e-mails are opened or when a link in the mail is clicked, the consumers are redirected to a spoofed website, appearing to be from the legitimate entity. We discuss the methods used for detection of phishing Websites based on URL importance properties. Phishing has been accounted for many fraudulent incidents on the internet in the recent years, and it is showing no sign of stopping anytime soon. So, what is phishing? It is a term that is used to describe a malicious individual or a group of individuals who scam users. This is done by sending emails or creating web pages that are designed to collect an individual's online credentials,credit card details or other login information's. The concept of detecting phishing websites is usually done by looking through a huge database or a directory that contains all the malicious sites that has been logged by internet users or community members. An effective way for end users to benefit from phishing detection is by having the option to use an extension plugin that works on real time, as it gives them real time indication of what they are surfing and as well as if they are safe while browsing.

1.2 PROJECT PURPOSE

The main purpose of the project is to detect the fake or phishingwebsites who are trying to get access to the sensitive data or by creating the fake websites and trying to get access of the user personal credentials. We are using machine learning algorithms to safeguard the sensitive data and to detect the phishing websites who are trying to gain access on sensitive data. This research mainly will focus on implementing machine learning in JavaScript for it to run on a browser as an extension since JavaScript does not have much library support towards Machine Learning and also to keep in mind of the users’ machines performance. This approach should be made with the intention of having it lite in order to achieve the capability to allow as much users as possible to use it. Random forest classifier for this project will be trained traditionally based on the phishing dataset 2 using Python scikit, and parameters of this model will then be exported in a JSON format to beused together with JavaScript. Phishing is a form of fraud in which the attacker tries to learn sensitive information such as login credentials or account information by sending as a reputable entity or person in email or other communication channels. Typically a victim receives a message that appears to have beensent by a known contact or organization. The message contains malicious software targeting the user’s computer or has links to direct victims to malicious websites in order to trick them into divulging personal and financial information, such as passwords, account IDs orcredit card details. The main reason is the lack of awareness of users.

CHAPTER 2

LITERATURE SURVEY

2.1 Existing Problem

There are e-banking websites that requests the users to provide more sensitive information such as credit card details, password etc., for malicious reasons. These websites that mimics trustful URLs and webpages are known as phishing websites.

Common causes for web phishing attacks involve:

- Users lack of security awareness
- Not performing sufficient due diligence
- Low-cost phishing and Ransomware tools are easy to get hold of
- Malware is becoming more sophisticated and so on

Web phishing is considered to be a threat in various aspects of security on the internet, which might involve scams and private information disclosure.

Some of the common threats of web phishing are:

- Attempt to fraudulently solicit personal information from an individual or organization.
- Attempt to deliver malicious software by posing as a trustworthy organization or entity.
- Installing those malwares infects the data that cause a data breach or even nature's forces that takes down your company's data headquarters, disrupting access.

For this purpose, the objective of our project involves building an efficient and intelligent system to detect such websites by applying a machine-learning algorithm which implements classification algorithms and techniques to extract the phishing datasets criteria to classify their legitimacy and as a result of which whenever a user makes a transaction online and makes payment through an e-banking website our system will use a data mining algorithm to detect whether the e-banking website is phishing website or not.

2.2References

[1] Farashazillah Yahya,Ryan Isaac W Mahibol,Chong Kim Ying,Magnus Bin Anai,Sidney Allister Frankie,Eric Ling Nin Wei and Rio Guntur Utomo,”Detection of Phising Websites using Machine
Learning Approaches”,2021 International Conference on Data Science and Its Applications (ICoDSA).

[2] Prajakta Patil,Rashmi Rane and Madhuri Bhalekar,”Detecting spam and phishing mails using SVM and obfuscation URL detection algorithm”,2017 International Conference on Inventive Systems and Control (ICISC).

[3] Gaurav Varshney, Manoj Mishra and Pradeep K. Atrey, "A phish detector using lightweight search features", Computers & Security, 2016.

[4] Antonio Hernández Dominguez and Walter Baluja García, "Updated Analysis of Detection Methods for Phishing Attacks", Futuristic Trends in Network and Communication Technologies, vol.1395, pp.56, 2021.

[5] Anggit Ferdita Nugraha and Luthfia Rahman, "Meta-Algorithms for Improving Classification Performance in the Web-phishing Detection Process", 2019 4th International Conference on Information Technology, Information Systems and Electrical Engineering (ICITISEE), pp.271-275, 2019.

[6] Yoga Pristyanto and Akhmad Dahlan, "Hybrid Resampling for Imbalanced Class Handling on Web Phishing Classification Dataset", 2019 4th International Conference on Information Technology, Information Systems and Electrical Engineering (ICITISEE), pp.401-406, 2019.

[7] Athulya A.A and Praveen K, ”Towards the Detection of Phishing Attacks”,2020 4th International
Conference on Trends in Electronics and Informatics (ICOEI)(48184)

[8] Miyamoto D, Hazeyama H and Kadobayashi Y,” An evaluation of machine learning-based methods for detection of phishing sites” , International Conference on Neural Information Processing pp. 539-546. Springer, Berlin, Heidelberg. (2008)

[9] K S Swarnalatha,K C Ramchandra,Kaushar Ansari,Love Ojha and Sanjok Subedi Sharma,”Real-Time Threat Intelligence-Block Phising Attacks”,2021 IEEE International Conference on Computation System and Information Technology for Sustainable Solutions (CSITSS)

2.3 Problem Statement Definition

Web Phishing is a form of cyber fraud, which implies that fraudsters use various means to impersonate the URL address and page content of a real website or use vulnerabilities in the server program of a real website to insert dangerous HTML code in certain pages of the site.

It is a threat in various aspects of security on the internet, which might involve scams and private information disclosure. Some of the common threats of web phishing are:

- Obtaining personal information from an individual or organization.
- Impersonating as a trustworthy organization to deliver malicious websites.

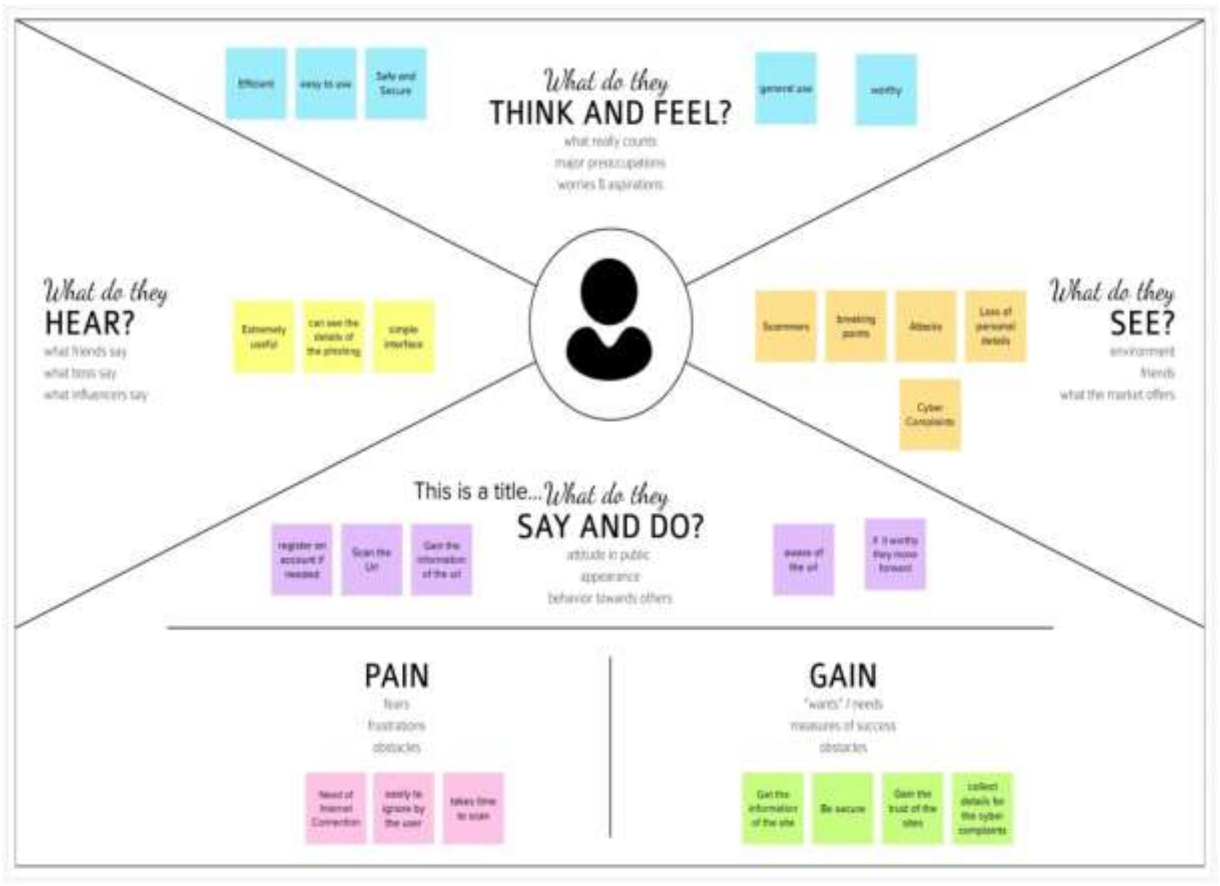
To avoid these threats, we build an efficient and intelligent system to detect such websites using machine-learning algorithms which implements classification algorithms and techniques to extract the phishing datasets criteria to classify their legitimacy.

CHAPTER 3

IDEATION & PROPOSED SOLUTION

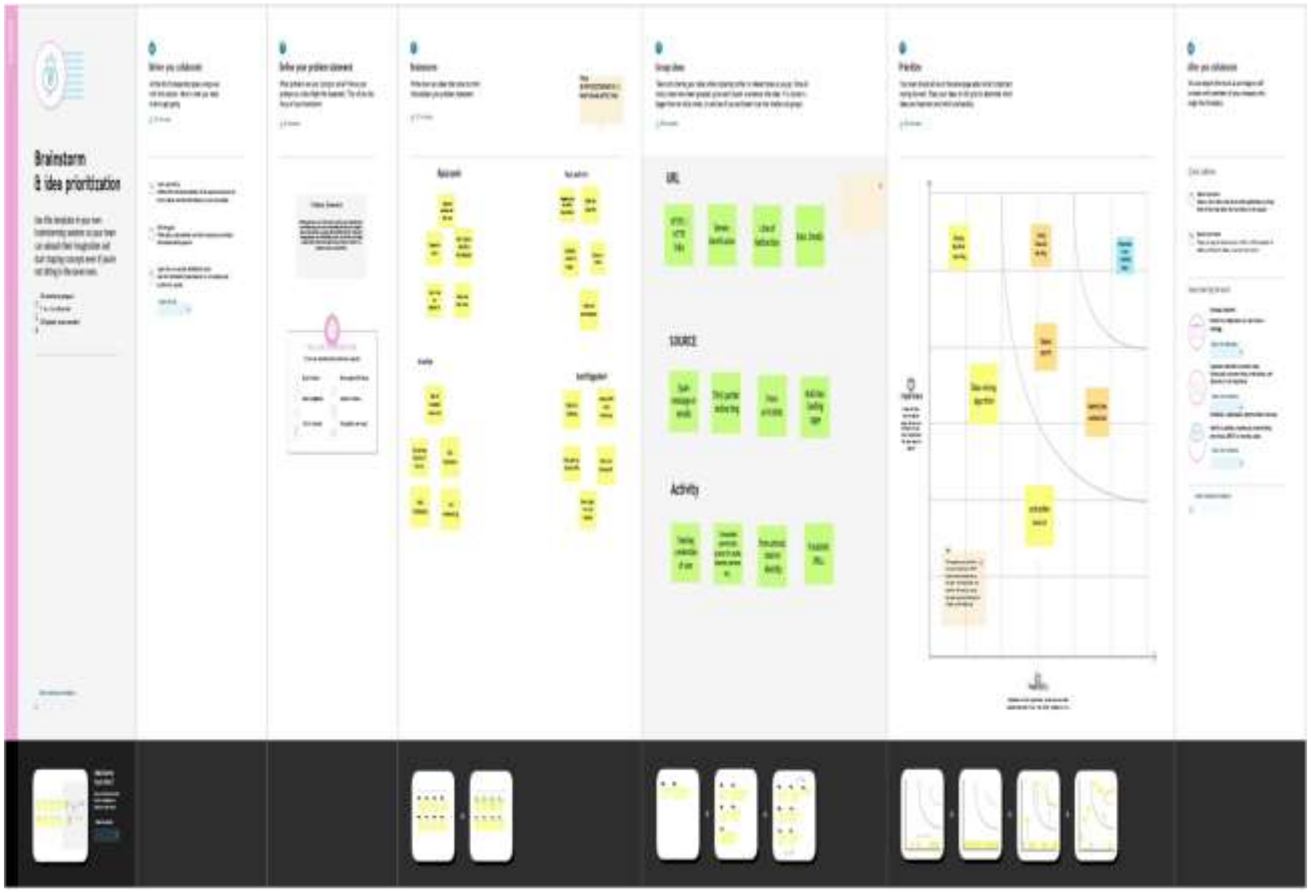
3.1 Empathy Map Canvas

An empathy map is a collaborative tool teams can use to gain a deeper insight into their customers. Much like a user persona, an empathy map can represent a group of users, such as a customer segment. Empathy maps should be used throughout any UX process to establish common ground among team members and to understand and prioritize user needs. In user-centered design, empathy maps are best used from the very beginning of the design process.



3.2 Ideation & Brainstrom

- Ideation essentially refers to the whole creative process of coming up with and communicating new ideas. Ideation is innovative thinking, typically aimed at solving a problem or providing a more efficient means of doing or accomplishing something.
- Ideation is often closely related to the practice of brainstorming, a specific technique that is utilized to generate new ideas. A principal difference between ideation and brainstorming is that ideation is commonly more thought of as being an individual pursuit, while brainstorming is almost always a group activity.



3.3 Proposed Solution

S.no	Parameter	Description
1.	Problem Statement (Problem to be solved)	As opposed to software vulnerabilities, "phishing sites" are a particular kind of internet security problems that primarily target human vulnerabilities. Phishing sites are harmful websites that pretend to be trustworthy websites or web pages in order to steal users' personal information, including their user name, password, and credit card number. Since phishing is mostly a semantics-based attack that focuses on human vulnerabilities, identifying these phishing websites can be difficult. The main goal of this project is to classify phishing websites using a variety of machine learning approaches in order to produce a model with the highest level of accuracy and simplicity.
2.	Idea / Solution description	• The method includes the extraction of lexical features from collected webpages as well as host- and page-based feature extraction. The first stage is gathering phishing and legitimate websites. In the host-based technique, attribute extractions based on admiration and lexical bases are carried out to create a database of attribute value. This database contains knowledge that has been extracted using various machine learning methods. A selective classifier is chosen after comparing the methods, and it is put into practice in Python. • The suggested approach gathered URLs of safe websites from sites like www.alexa.com, www.dmoz.org, and browsing history. We gathered the phishing URLs from www.phishtak.com. 20000 benign URLs and 17000 phishing URLs

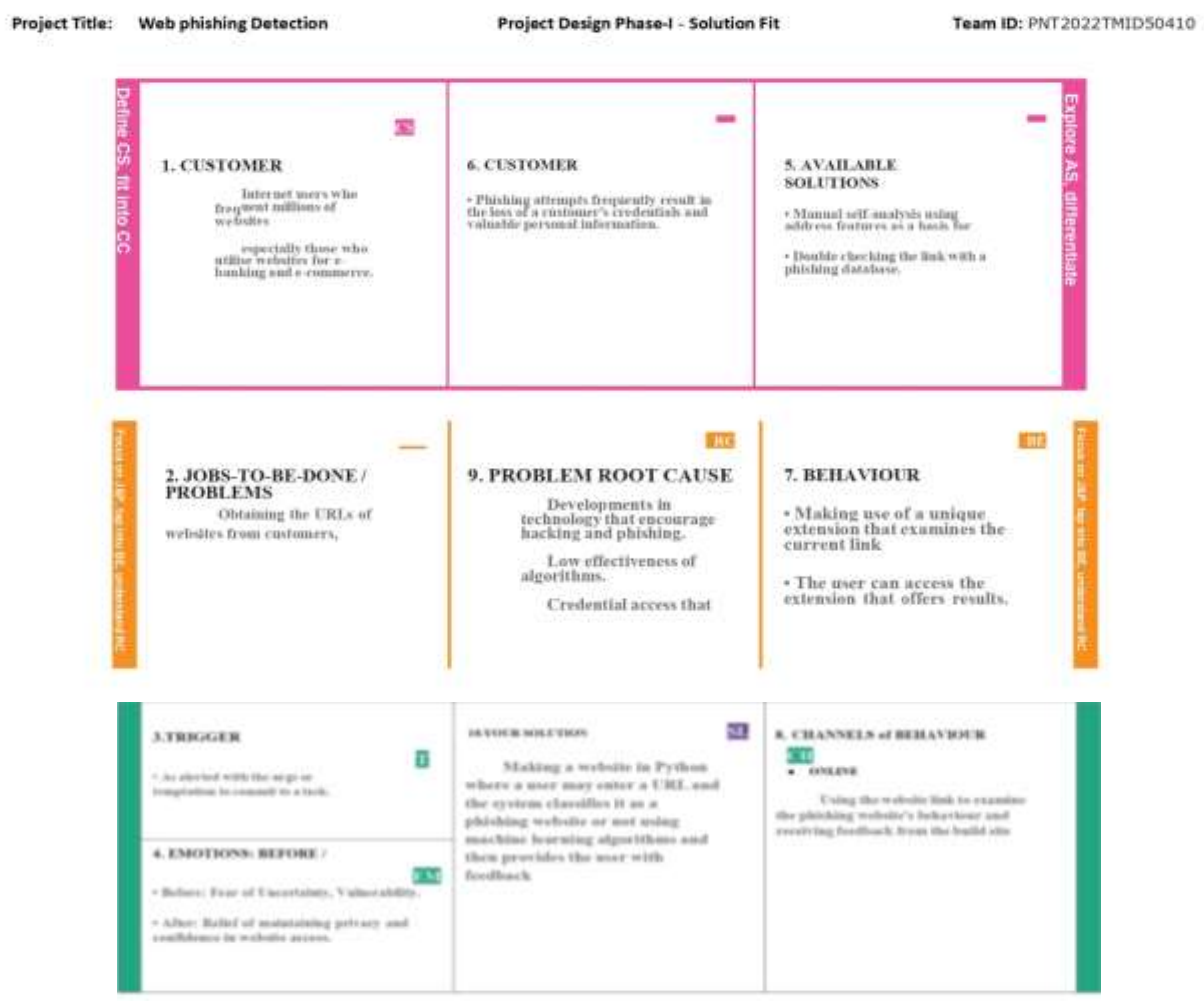
		make up the data collection.
3.	Novelty / Uniqueness	The dataset provided by UCI Machine Learning repository4 and compiled by Mohammad et al3 was used by the suggested system. The dataset contains 6157 legal URLs and 4898 phishing URLs across 11055 datapoints. Each data point had 30 features that were sorted into the three categories below: ● Features extracted from the URL ● Features based on the page's source code, such as URLs that are incorporated into the webpage and HTML and Javascript-based features. ● Features based on domains.
4.	Social Impact / Customer Satisfaction	The majority of the public (users) were assisted by the project in determining if a website was a phishing website or not. It assisted them in classifying the hazardous locations. Machine learning methods were employed in this research. The URL is entered, and it will recognize it and provide users with precise results.
5.	Business Model (Revenue Model)	In the literature, a number of methods for phishing attack detection and filtering have been suggested. Researchers are still looking for a solution that can protect consumers from phishing attacks and produce better outcomes. It might be easier to spot phishing websites if we can recognize the specific traits and patterns they exhibit. The classification problem of identifying such traits can be resolved using machine learning approaches.

3.4 Problem Solution fit

The Problem-Solution Fit simply means that you have found a problem with your customer and that the solution you have realized for it solves the customer’s problem. It helps entrepreneurs, marketers and corporate innovators identify behavioural patterns and recognize what would work and why.

Purpose:

- Solve complex problems in a way that fits the state of your customers.
- Succeed faster and increase your solution adoption by tapping into existing mediums and channels of behaviour.
- Sharpen your communication and marketing strategy with the right triggers and messaging.
- Increase touch points with your company by finding the right problem-behaviour fit and building trust by solving frequent annoyances, or urgent or costly problems.
- Understand the existing situation in order to improve it for your target group.



Project Title: Web phishing Detection **Project Design Phase-I - Solution Fit** **Team ID:** PNT2022TMID50410

Define CS, fit into CC
Explore AS, differentiate

1. CUSTOMER

Internet users who frequent millions of websites

especially those who utilise websites for e-banking and e-commerce.

6. CUSTOMER

• Phishing attempts frequently result in the loss of a customer's credentials and valuable personal information.

5. AVAILABLE SOLUTIONS

• Manual self-analysis using address features as a basis for

• Double checking the link with a phishing database.

2. JOBS-TO-BE-DONE / PROBLEMS

Obtaining the URLs of websites from customers,

9. PROBLEM ROOT CAUSE

Developments in technology that encourage hacking and phishing.

Low effectiveness of algorithms.

Credential access that

7. BEHAVIOUR

• Making use of a unique extension that examines the current link

• The user can access the extension that offers results.

3.TRIGGER

• As alerted with the urge or temptation to commit to a task.

4. EMOTIONS: BEFORE /

• Before: Fear of Uncertainty, Vulnerability.

• After: Relief of maintaining privacy and confidence in website access.

10.YOUR SOLUTION

Making a website in Python where a user may enter a URL and the system classifies it as a phishing website or not using machine learning algorithms and then provides the user with feedback

8. CHANNELS of BEHAVIOUR

- ONLINE

Using the website link to examine the phishing website's behaviour and receiving feedback from the build site

CHAPTER 4

REQUIREMENT ANALYSIS

4.1 Functional Requirement

FR NO	Functional Requirements	Classification
FR-1	Fetch Electronic Mail Messages	Core
FR-2	Extract URLS	Core
FR-3	Extract Header Information	Core
FR-4	Classify Email	Core
FR-5	Static or Dynamic (Inbox)	Core
FR-6	Provide User Feedback	Core

4.2 Non-functional Requirements

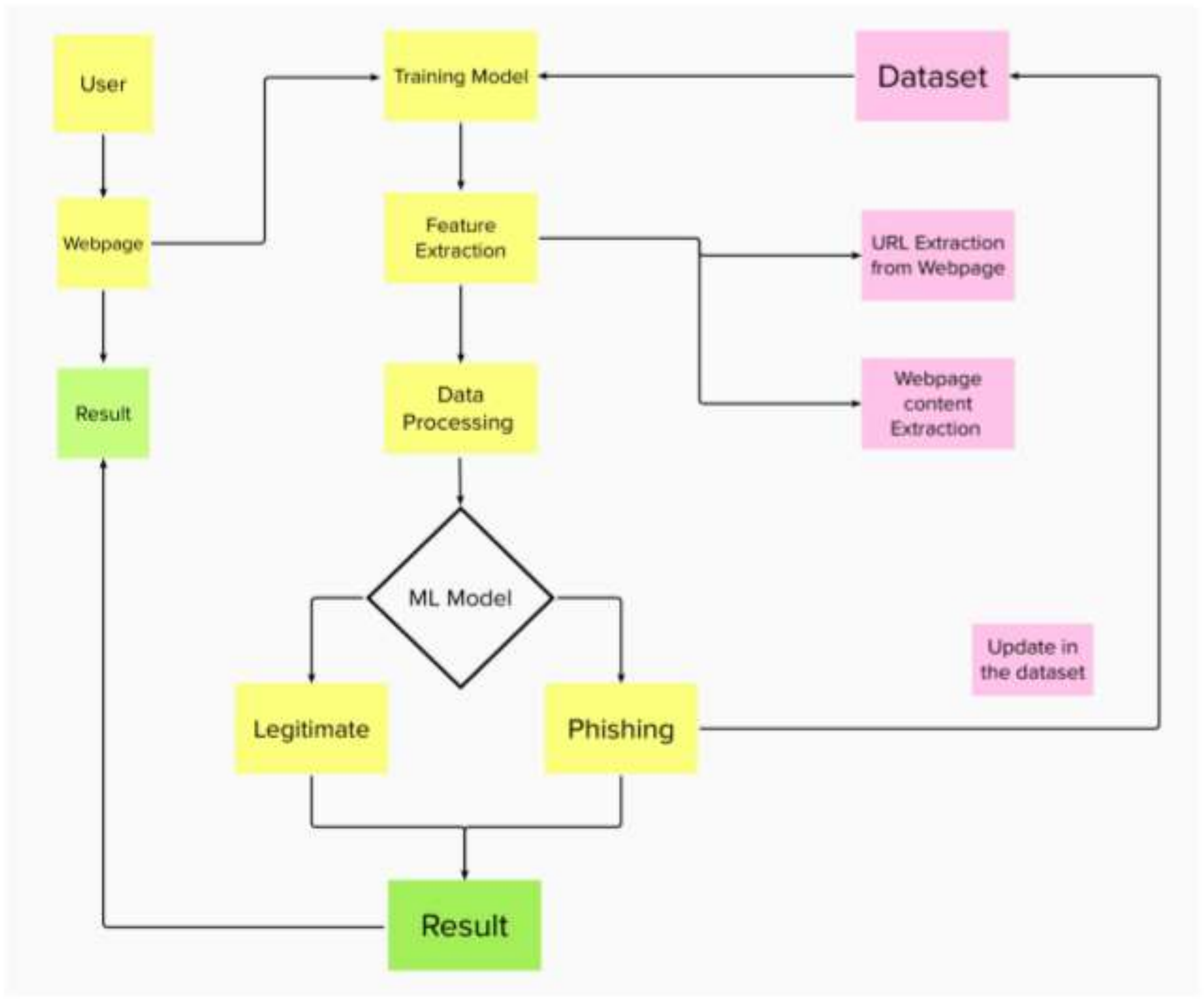
NFR NO	Non-Functional Requirements	Description
NFR-1	Usability	System is easy to configure and is efficient in carrying out user tasks.
NFR-2	Availability	System is available to work as required when it is required.
NFR-3	Reliability	System will perform the tasks it was designed to do.
NFR-4	Performance	System will perform tasks in a fashion that complies with predetermined criteria.
NFR-5	Security	System will protect all data manipulated internally from unauthorized access and threats.

CHAPTER 5

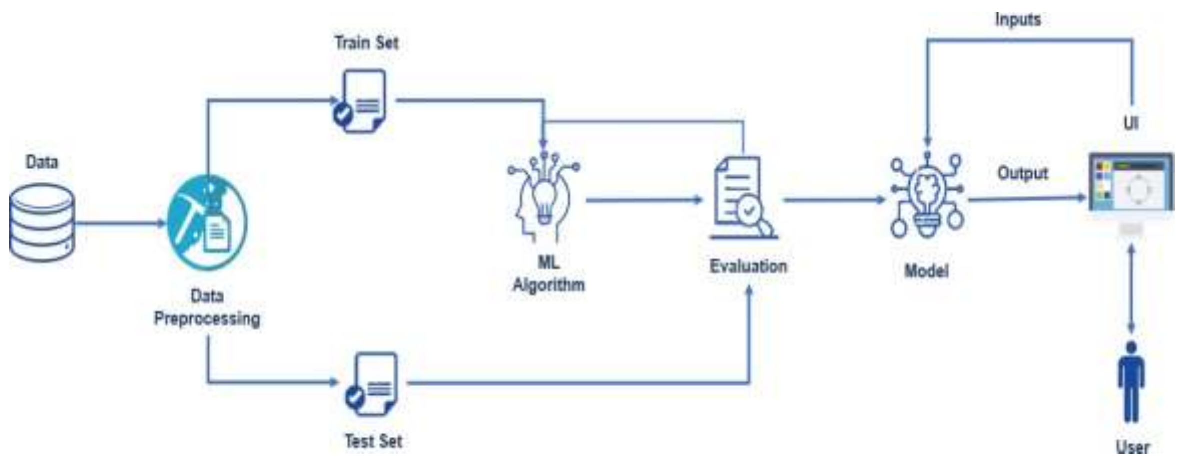
PROJECT DESIGN

5.1Data Flow Diagrams

A Data Flow Diagram (DFD) is a traditional visual representation of the information flows within a system. A neat and clear DFD can depict the right amount of the system requirement graphically. It shows how data enters and leaves the system, what changes the information, and where data is stored.



5.2 Solution & Technical Architecture



5.3User Stories

User Type	Functional Requirement (Epic)	User Story Number	User Story /Task	Acceptance Criteria	Priority	Release
Customer (Mobile user)	Registration	USN-1	As a user, I can register for the application by entering my email, password, and confirming my password.	I can access my account / dashboard	High	Sprint-1
		USN-2	As a user, I will receive confirmation email once I have registered for the application	I can receive confirmation email & click confirm	High	Sprint-1
		USN-3	As a user, I can register for the application through Facebook	I can register & access the dashboard with Facebook Login	Low	Sprint-2
		USN-4	As a user, I can		Medium	Sprint-1

			register for the application through Gmail			
	Login	USN-5	As a user, I can log into the application by entering email & password		High	Sprint-1
	Dashboard					Sprint-1
Customer (Web user)	User Input	USN-1	As a user, I can enter the required URL in the box while awaiting validation.	I can access the website without any problem	High	Sprint-1
Customer Care Executive	Feature Extraction	USN-1	In the event that nothing is discovered during comparison, we can extract features using a heuristic and a visual similarity technique	As a user I can have comparison between websites for security	High	Sprint-1
Administ rator	Prediction	USN-1	The model will use machine learning algorithms like a logistics regression and KNN to forecast the URLs of the websites.	I can accurately forecast the specific algorithms in this way.	High	Sprint-1
	Classifier	USN-2	To create the final product, I will now feed all of the model output to classifier.	I'll use this to identify the appropriate classifier for generating the outcome.	Medium	Sprint-2

CHAPTER 6

PROJECT PLANNING & SCHEDULING

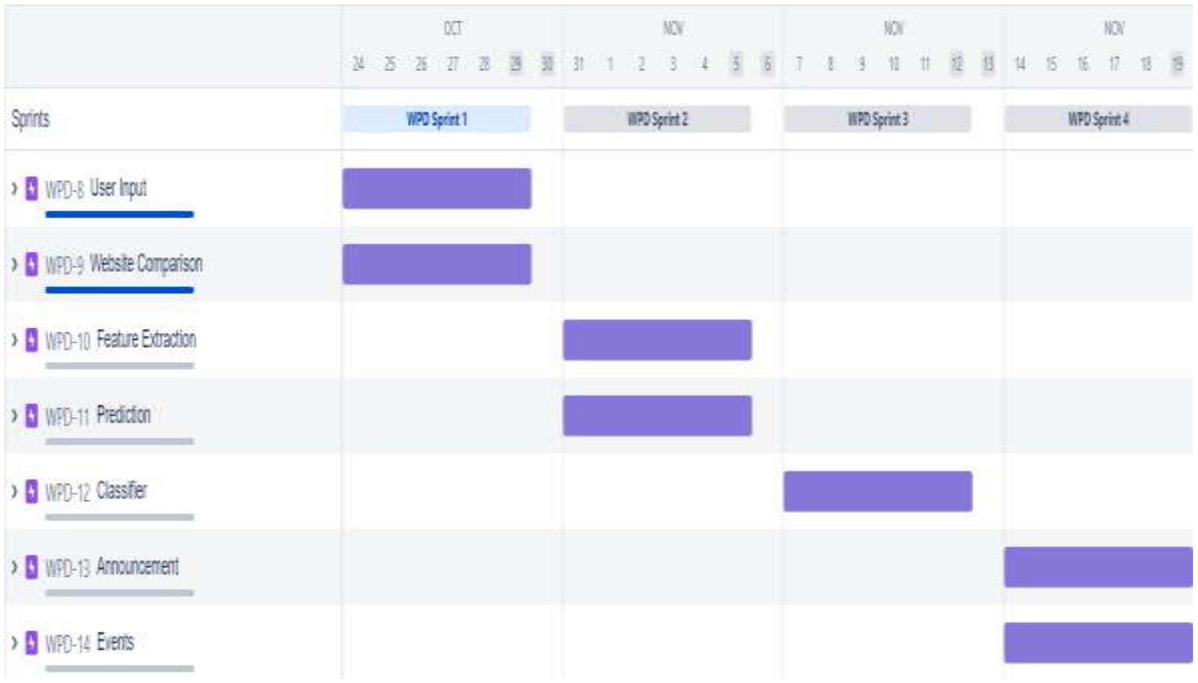
6.1 Sprint Planning & Estimation

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-1	Home page	USN-1	As a user, first have an good impression upon the homepage and I can explore and view the funtioning of the website.	20	medium	Kalai Selvi, Paul Reshmin
Sprint-2	Registration	USN-2	As a user, I will receive confirmation email once I have registered for the application.	10	High	Anushya, Karthiga Devi
Sprint-2		USN-3	As a user, I can register for the application through google.		Medium	Karthiga Devi
Sprint-2	Login	USN-4	As a user, I can register for the application through Gmail.	10	Medium	Paul Reshmin
Sprint-2		USN-5	As a user, I can log into the application by entering email & password.		Low	Anushya
Sprint-3	Dashboard	USN-6	User would go through the funtionalities and the uses of the website.	5	Low	Kalai Selvi

6.2 Sprint Delivery Schedule

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

6.3 Reports from JIRA

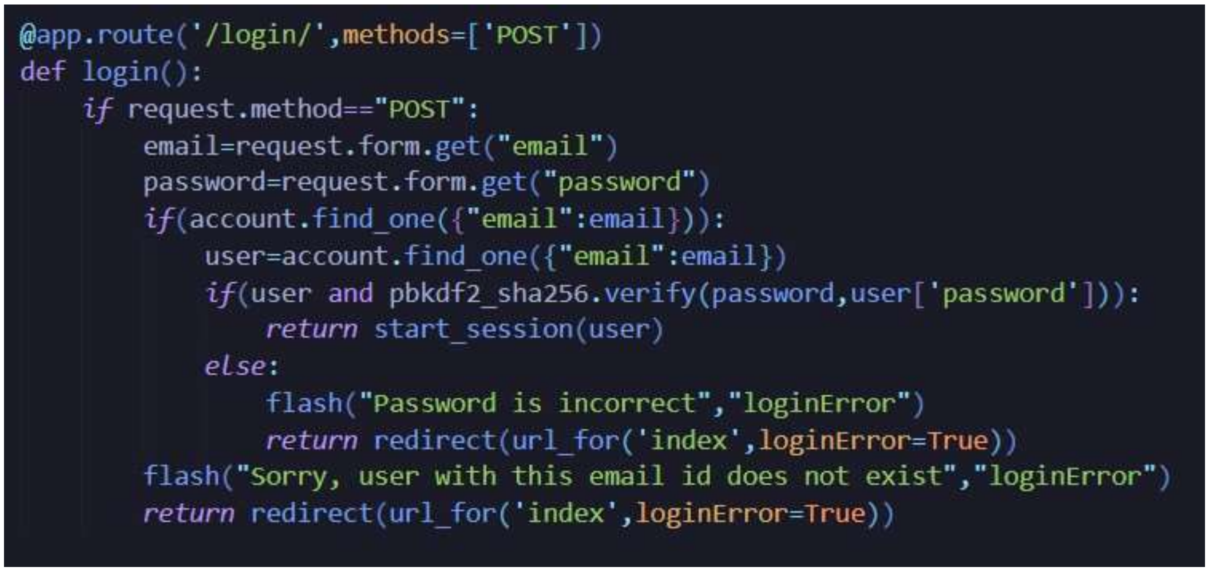


CHAPTER 7

CODING & SOLUTIONING

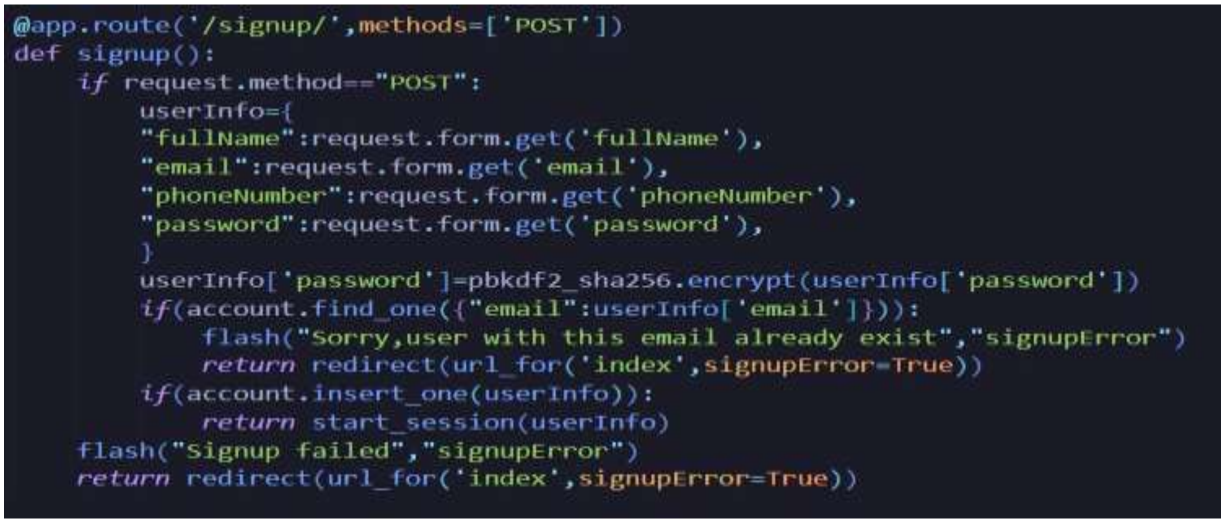
7.1 Feature 1

LOGIN



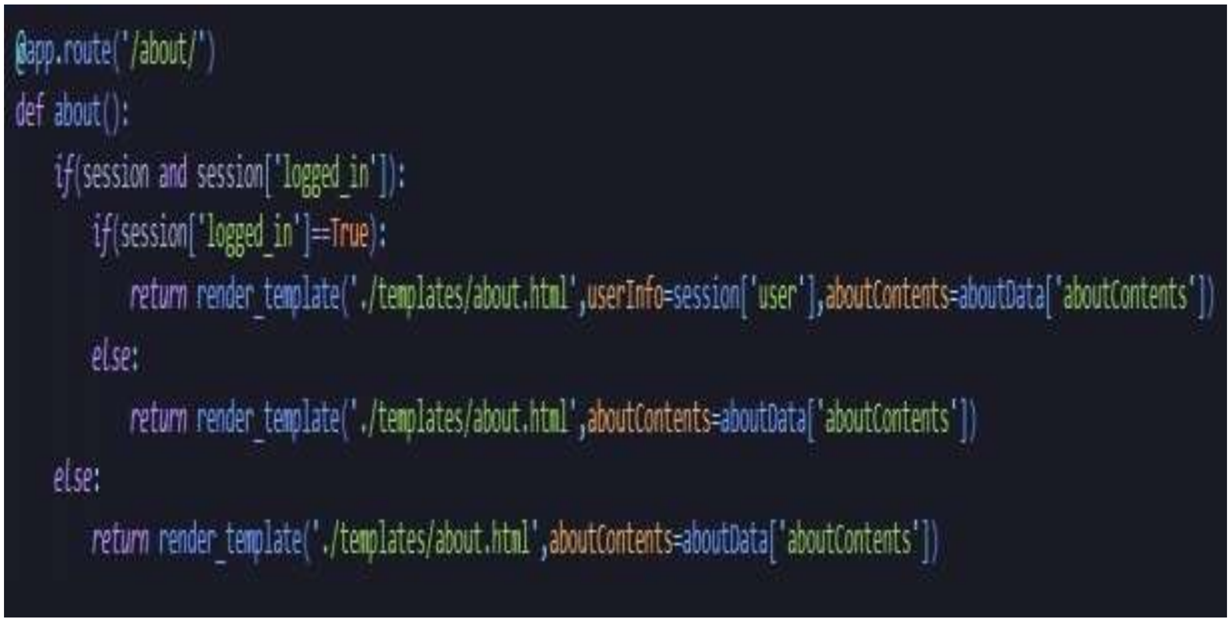
```
@app.route('/login/',methods=['POST'])
def login():
    if request.method=="POST":
        email=request.form.get("email")
        password=request.form.get("password")
        if(account.find_one({"email":email})):
            user=account.find_one({"email":email})
            if(user and pbkdf2_sha256.verify(password,user['password'])):
                return start_session(user)
            else:
                flash("Password is incorrect","loginError")
                return redirect(url_for('index',loginError=True))
        flash("Sorry, user with this email id does not exist","loginError")
        return redirect(url_for('index',loginError=True))
```

SIGN UP



```
@app.route('/signup/',methods=['POST'])
def signup():
    if request.method=="POST":
        userInfo={
        "fullName":request.form.get('fullName'),
        "email":request.form.get('email'),
        "phoneNumber":request.form.get('phoneNumber'),
        "password":request.form.get('password'),
        }
        userInfo['password']=pbkdf2_sha256.encrypt(userInfo['password'])
        if(account.find_one({"email":userInfo['email']})):
            flash("Sorry,user with this email already exist","signupError")
            return redirect(url_for('index',signupError=True))
        if(account.insert_one(userInfo)):
            return start_session(userInfo)
    flash("Signup failed","signupError")
    return redirect(url_for('index',signupError=True))
```

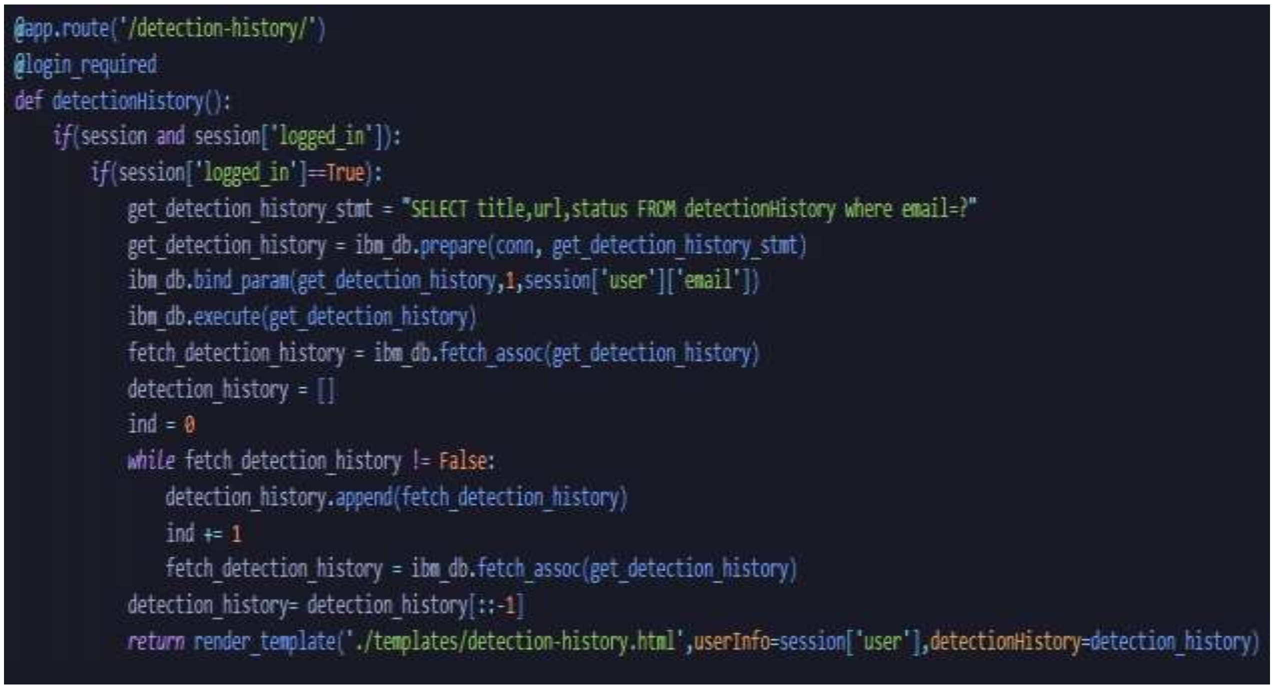
ABOUT US



```python
@app.route('/about/')
def about():
    if(session and session['logged_in']):
        if(session['logged_in']==True):
            return render_template('./templates/about.html',userInfo=session['user'],aboutContents=aboutData['aboutContents'])
        else:
            return render_template('./templates/about.html',aboutContents=aboutData['aboutContents'])
    else:
        return render_template('./templates/about.html',aboutContents=aboutData['aboutContents'])
```

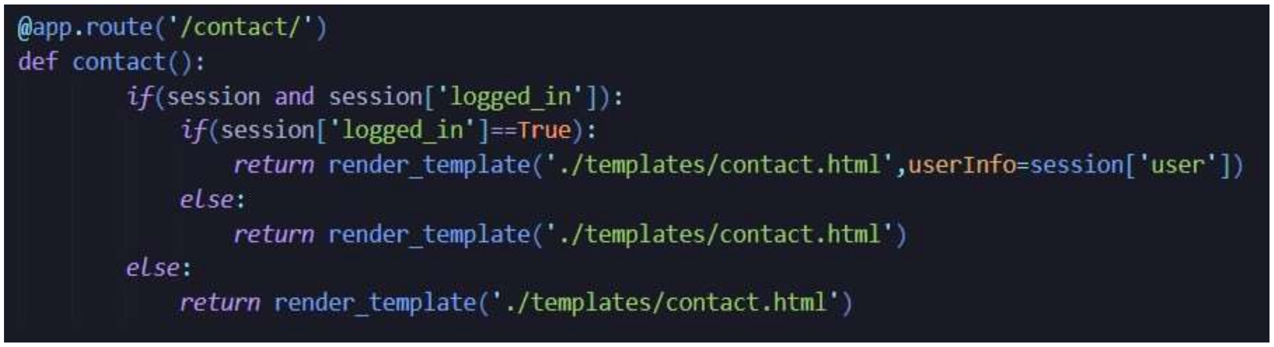
7.2 Feature2

HISTORY PAGE



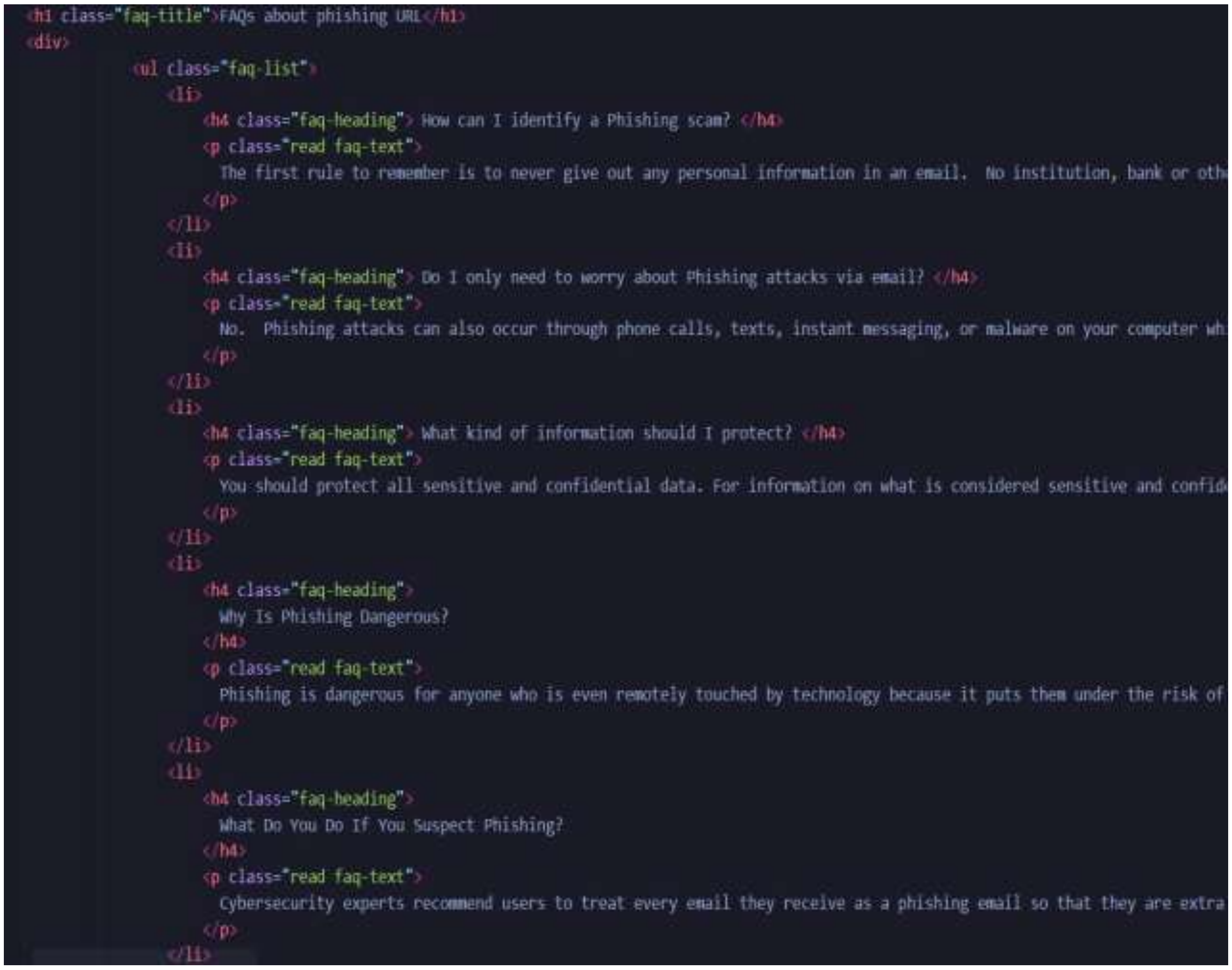
```python
@app.route('/detection-history/')
@login_required
def detectionHistory():
    if(session and session['logged_in']):
        if(session['logged_in']==True):
            get_detection_history_stmt = "SELECT title,url,status FROM detectionHistory where email=?"
            get_detection_history = ibm_db.prepare(conn, get_detection_history_stmt)
            ibm_db.bind_param(get_detection_history,1,session['user']['email'])
            ibm_db.execute(get_detection_history)
            fetch_detection_history = ibm_db.fetch_assoc(get_detection_history)
            detection_history = []
            ind = 0
            while fetch_detection_history != False:
                detection_history.append(fetch_detection_history)
                ind += 1
                fetch_detection_history = ibm_db.fetch_assoc(get_detection_history)
            detection_history= detection_history[::-1]
            return render_template('./templates/detection-history.html',userInfo=session['user'],detectionHistory=detection_history)
```

CONTACT US PAGE



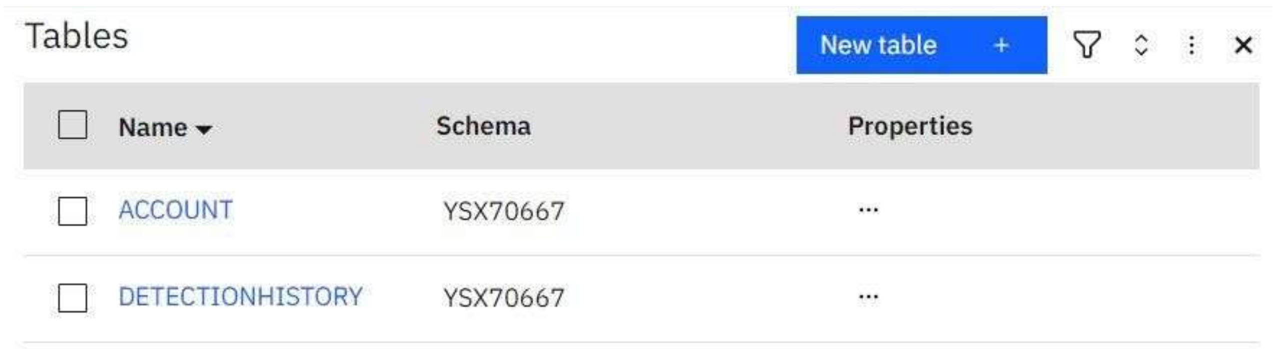
```python
@app.route('/contact/')
def contact():
        if(session and session['logged_in']):
            if(session['logged_in']==True):
                return render_template('./templates/contact.html',userInfo=session['user'])
            else:
                return render_template('./templates/contact.html')
        else:
            return render_template('./templates/contact.html')
```

FAQ



```html
<h1 class="faq-title">FAQs about phishing URL</h1>
<div>
        <ul class="faq-list">
            <li>
                <h4 class="faq-heading"> How can I identify a Phishing scam? </h4>
                <p class="read faq-text">
                  The first rule to remember is to never give out any personal information in an email.  No institution, bank or oth
                </p>
            </li>
            <li>
                <h4 class="faq-heading"> Do I only need to worry about Phishing attacks via email? </h4>
                <p class="read faq-text">
                  No.  Phishing attacks can also occur through phone calls, texts, instant messaging, or malware on your computer wh
                </p>
            </li>
            <li>
                <h4 class="faq-heading"> What kind of information should I protect? </h4>
                <p class="read faq-text">
                  You should protect all sensitive and confidential data. For information on what is considered sensitive and confid
                </p>
            </li>
            <li>
                <h4 class="faq-heading">
                  Why Is Phishing Dangerous?
                </h4>
                <p class="read faq-text">
                  Phishing is dangerous for anyone who is even remotely touched by technology because it puts them under the risk of
                </p>
            </li>
            <li>
                <h4 class="faq-heading">
                  What Do You Do If You Suspect Phishing?
                </h4>
                <p class="read faq-text">
                  Cybersecurity experts recommend users to treat every email they receive as a phishing email so that they are extra
                </p>
            </li>
```

7.3 Database Schema



Name	Schema	Properties
ACCOUNT	YSX70667	...
DETECTIONHISTORY	YSX70667	...

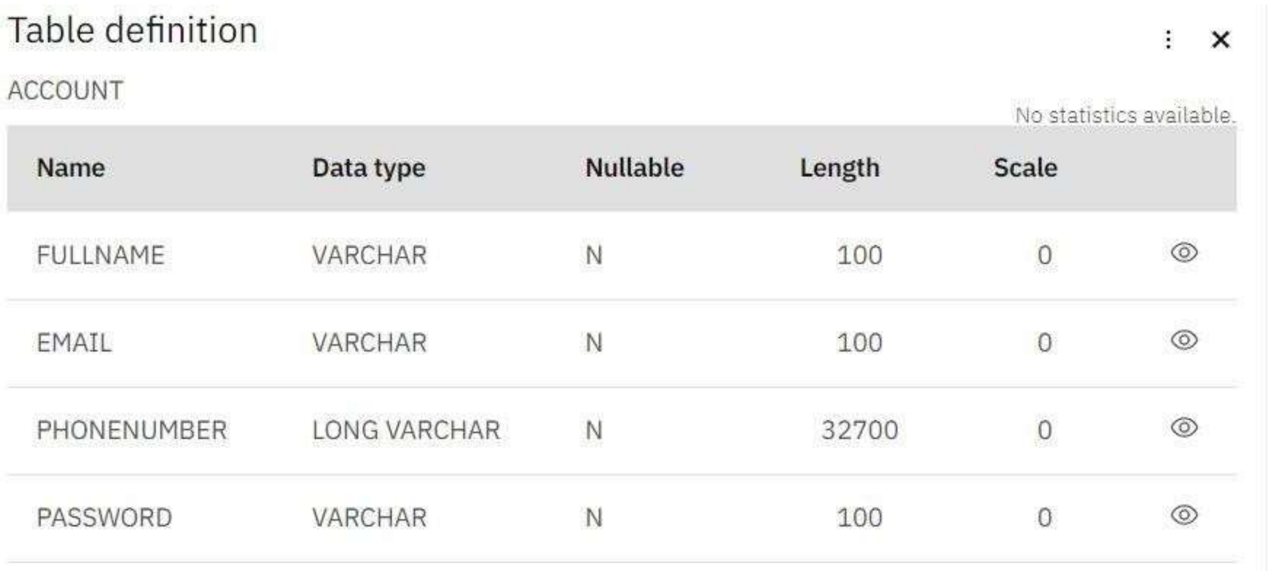


Table definition

ACCOUNT

No statistics available.

Name	Data type	Nullable	Length	Scale
FULLNAME	VARCHAR	N	100	0
EMAIL	VARCHAR	N	100	0
PHONENUMBER	LONG VARCHAR	N	32700	0
PASSWORD	VARCHAR	N	100	0

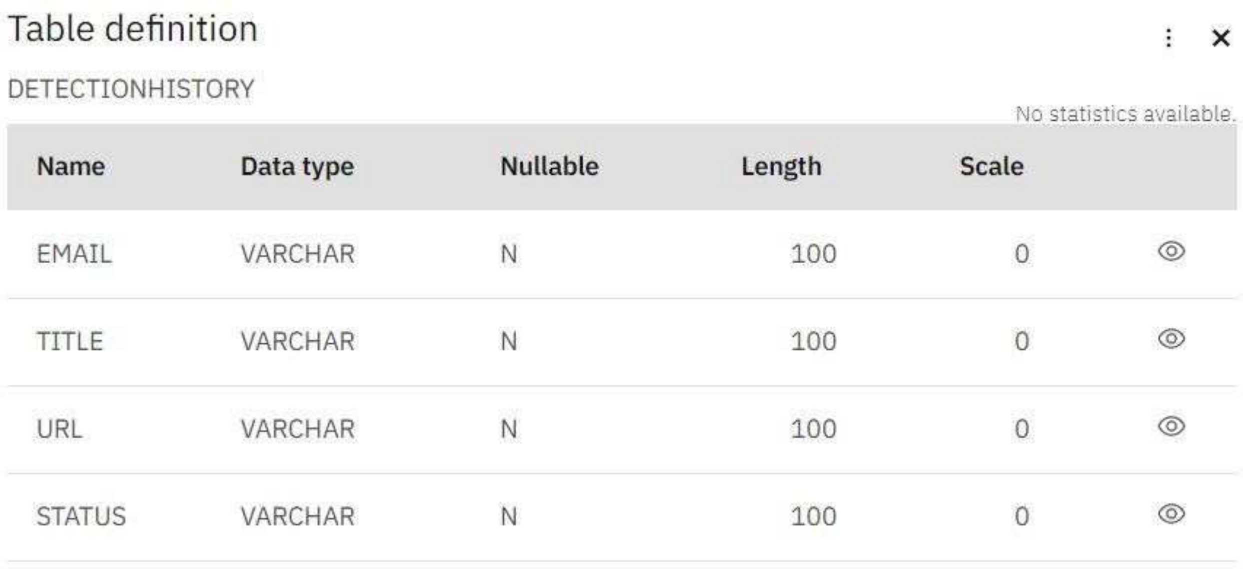


Table definition

DETECTIONHISTORY

No statistics available.

Name	Data type	Nullable	Length	Scale
EMAIL	VARCHAR	N	100	0
TITLE	VARCHAR	N	100	0
URL	VARCHAR	N	100	0
STATUS	VARCHAR	N	100	0

CHAPTER 8

TESTING

8.1 Test cases

Test case ID	Feature Type	Component	Test Scenario	Pre-Requisite	Steps ToExecute	Test Data	Result	Status
HomePage_TC_OO1	Functional	Home Page	Verify useris able to enter the URL in the form	Run the flask app in local host	1.Open our phishing website 2. Login to use the phishing services 3. Enter the link to be detected and click onpredict button	https://google.com/	Working as expected	Pass
ResultPage_TC_OO1	UI	Contact us page	Verify the UI elements inthe form	Run the flask app in local host	1. Enter name, email and message 2. Presssubmit		Working as expected	Pass
ResultPage_TC_OO2	Functional	Prediction result page	Verify user is able to see an alert when	Run the flask app in local host	1.Enter URLand click go		Working as expected	Pass

8.2Acceptance 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	10	4	2	3	20
Duplicate	1	0	3	0	4
External	2	3	0	1	6
Fixed	11	2	4	20	37
Not Reproduced	0	0	1	0	1
Skipped	0	0	1	1	2
Won't Fix	0	5	2	1	8
Totals	24	14	13	26	77

- Test Case Analysis

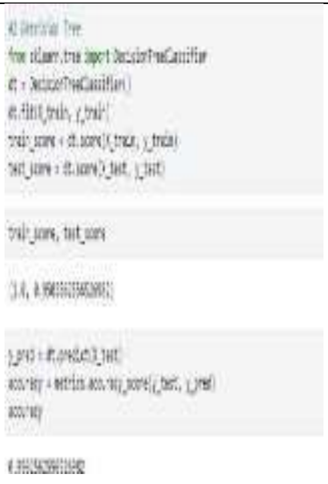
This report shows the number of test cases that have passed, failed, and untested

Section	Total Cases	Not Tested	Fail	Pass
Print Engine	5	0	0	5-
Client Application	51	0	0	51
Security	2	0	0	2
Outsource Shipping	3	0	0	3
Exception Reporting	9	0	0	9
Final Report Output	4	0	0	4
Version Control	2	0	0	2

CHAPTER 9

RESULTS

9.1 Performance Metrics

S.no	Parameter	Values	Screenshot
1.	Model Summary	Decision Tree Model Accuracy - 97%	
2.	Accuracy	Training Accuracy- Test	

```
#2 Decision Tree
from sklearn.tree import DecisionTreeClassifier
dt = DecisionTreeClassifier()
dt.fit(X_train, y_train)
train_score = dt.score(X_train, y_train)
test_score = dt.score(X_test, y_test)
```

```
train_score, test_score
```

```
(1.0, 0.9[illegible])
```

```
y_pred = dt.predict(X_test)
accuracy = metrics.accuracy_score(y_test, y_pred)
accuracy
```

```
0.9[illegible]
```

```
#2 Decision Tree
from sklearn.tree import DecisionTreeClassifier
dt = DecisionTreeClassifier()
dt.fit(X_train, y_train)
train_score = dt.score(X_train, y_train)
test_score = dt.score(X_test, y_test)
```

```
train_score, test_score
```

```
(1.0, 0.9[illegible])
```

```
y_pred = dt.predict(X_test)
accuracy = metrics.accuracy_score(y_test, y_pred)
accuracy
```

```
0.9[illegible]
```

CHAPTER 10

ADVANTAGES & DISADVANTAGES

Advantages:

- Increases user alertness to phishing risks Whenever the user navigates into the website and provide the URL of the website that needs to be verified for legitimacy, the system detects phishing sites by applying a machine learning algorithm which implements classification algorithms and techniques to extract the phishing datasets criteria to classify their legitimacy which in turn helps the customers to eliminate the risks of cyber threat and protect their valuable corporate or personal data.
- Users will also be able to pose any query to the admin through the report page designed our system is also provided with an option for the clients to report to the administrator which helps them to ask their questions significantly improving their experience on our site.

Disadvantages

- Not a generalized model
- Huge number of rules.
- Needs feed continuously.
- Hackers Find New Way To Attack,
- May not be able to detect all the websites.
- Could possibly bypass the detection.

CHAPTER 11

CONCLUSION

The importance to safeguard online users from becoming victims of online fraud, divulging confidential information to an attacker among other effective uses of phishing as an attacker's tool, phishing detection tools play avital role in ensuring a secure online experience for users. Unfortunately, many of the existing phishing-detection tools, especially those that depend on an existing blacklist, suffer limitations such as low detection accuracy and high false alarm that is often caused by either a delay in blacklist update as a result of human verification process involved in classification or perhaps, it can be attributed to human error in classification which may lead to improper classification of the classes. These critical issues have drawn many researchers to work on various approaches to improve detection accuracy ofphishing attacks and to minimize false alarm rate. The inconsistent nature of attacks behaviors and continuously changing URL phish patterns require timely updating of the reference model. Therefore, it requires an effective technique to regulate retraining as to enable machine learning algorithm to activelyadapt to the changes in phish patterns.

CHAPTER 12

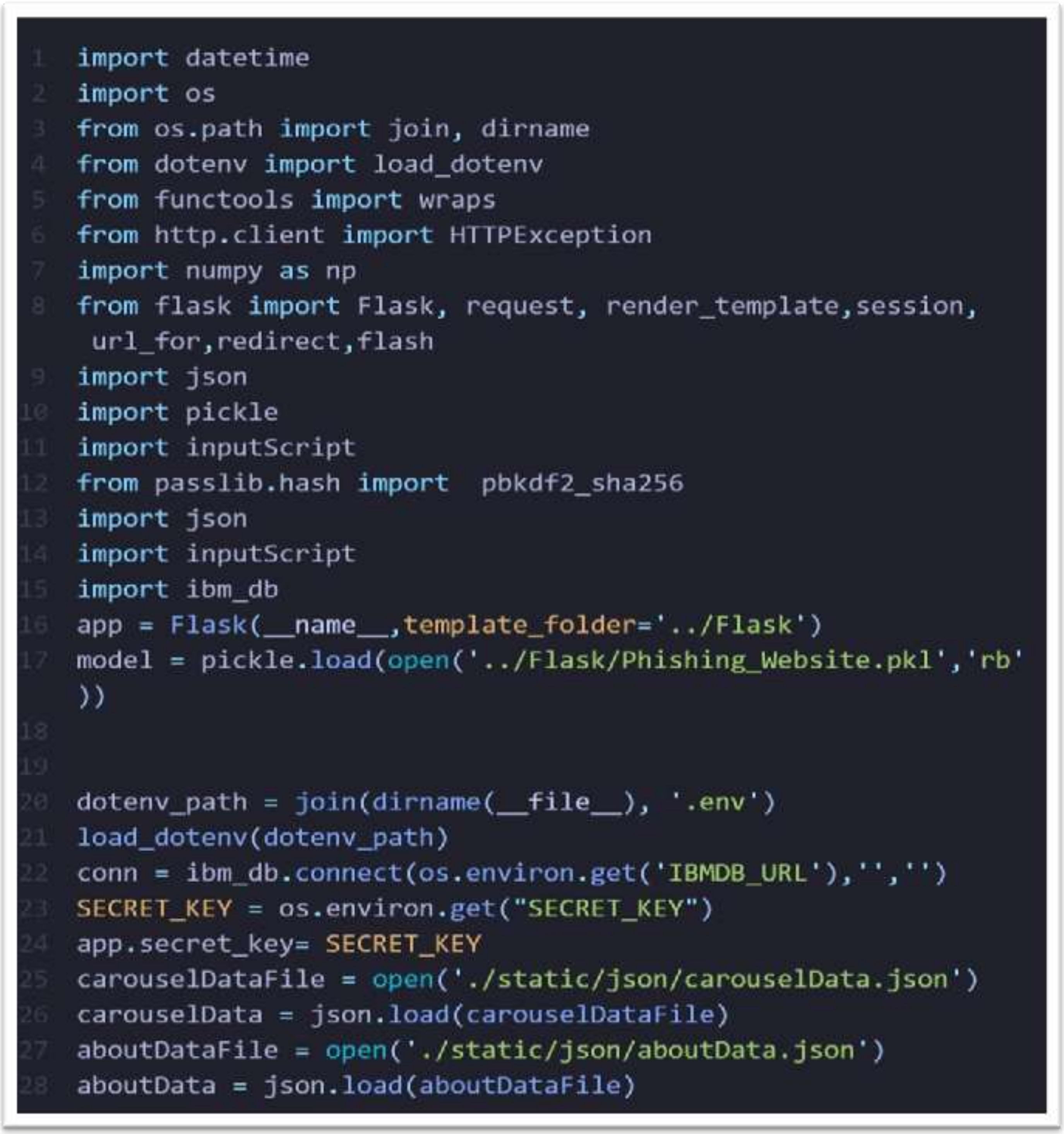
FUTURE SCOPE

In future we intend to build an add-ons for our system and if we get a structured dataset of phishing, we can perform phishing detection much faster than any other technique. We can also use a combination of any two or more classifiers to get maximum accuracy. We plan to explore various phishing techniques which use Network based features, Content based features webpage based features and HTML and JavaScript features of web pages which will improve the performance of the system. In particular, we extract features from URLs and pass it through the various classifiers.

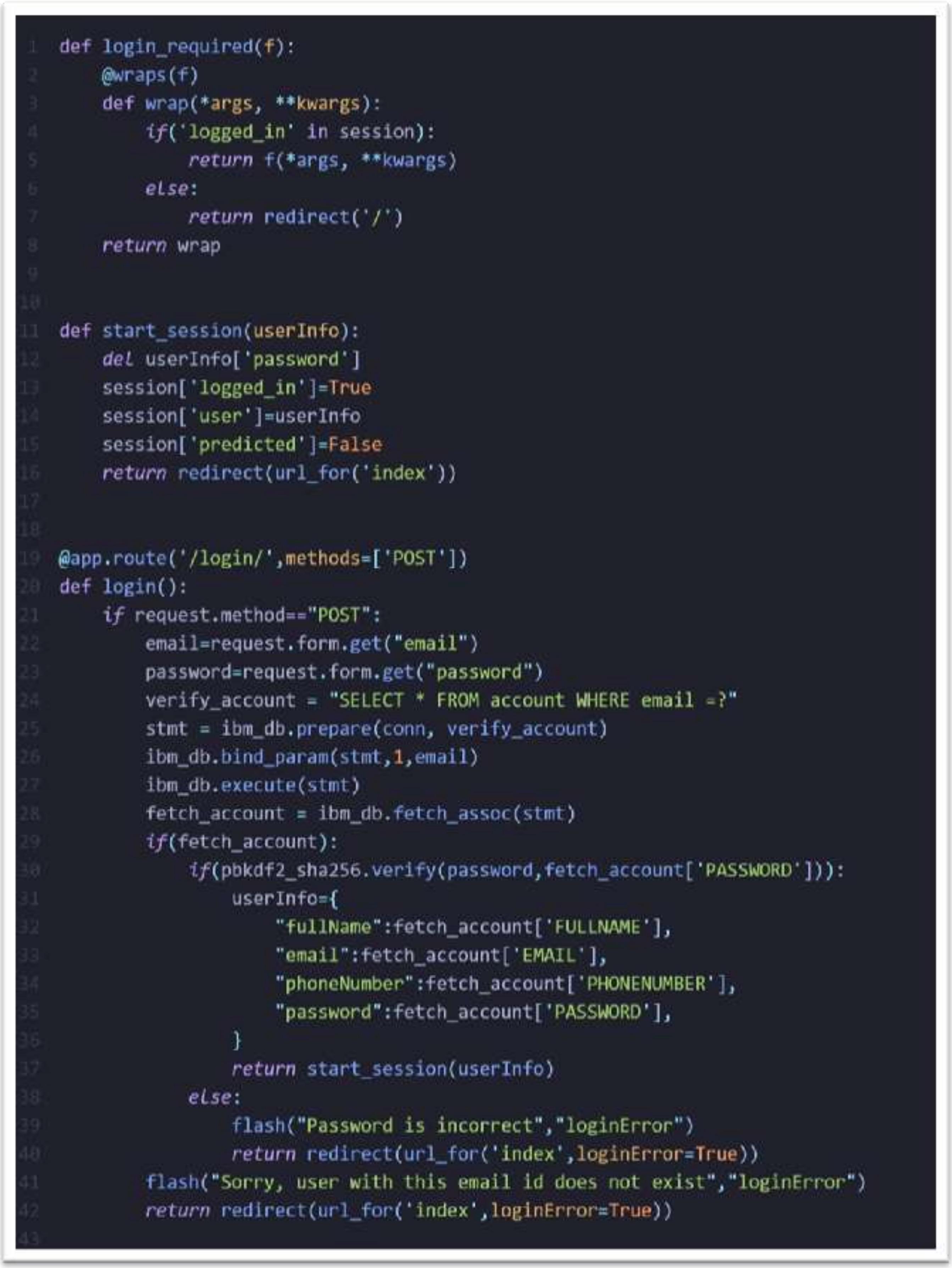
CHAPTER 13

APPENDIX

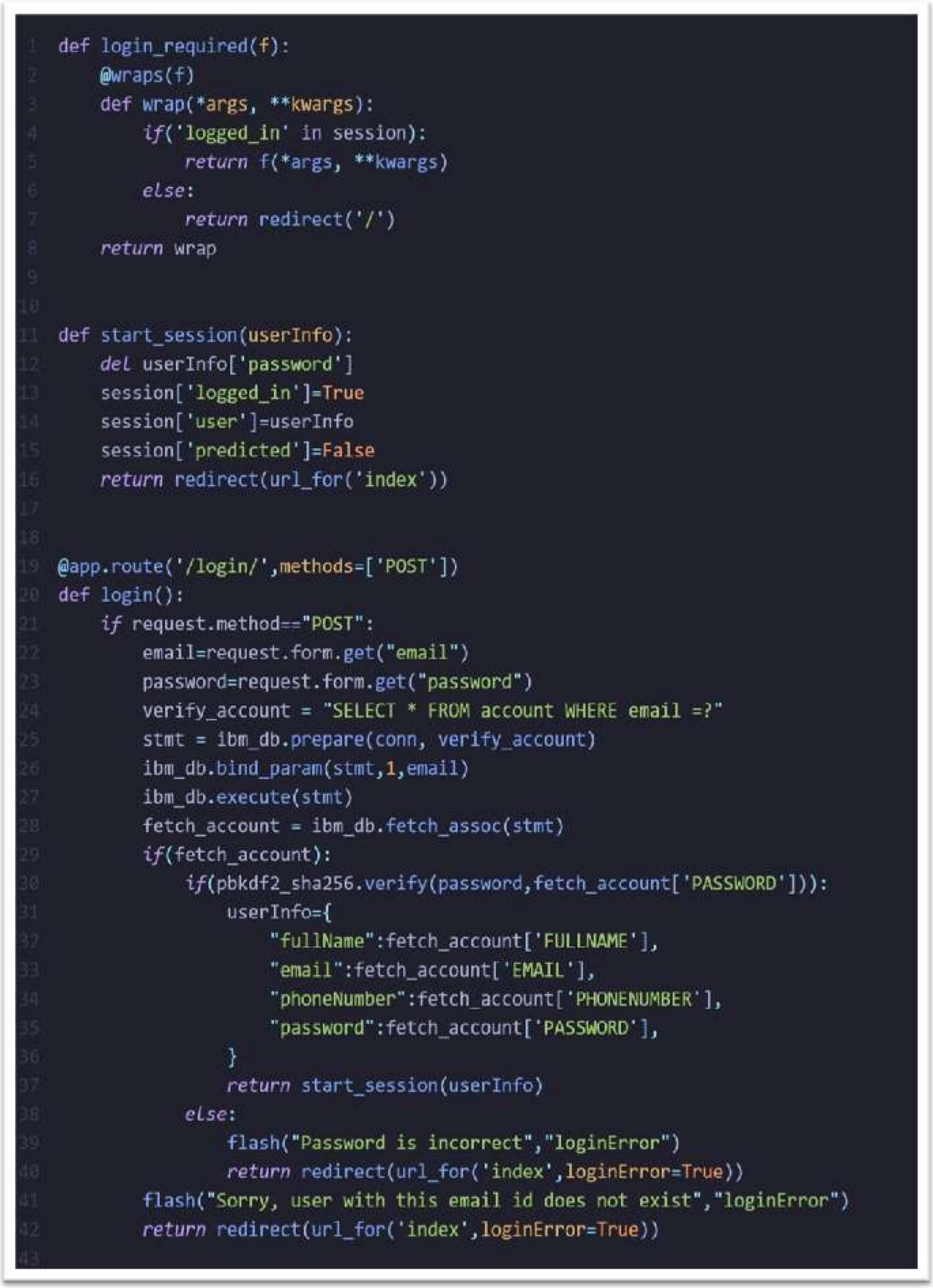
13.1 Source code



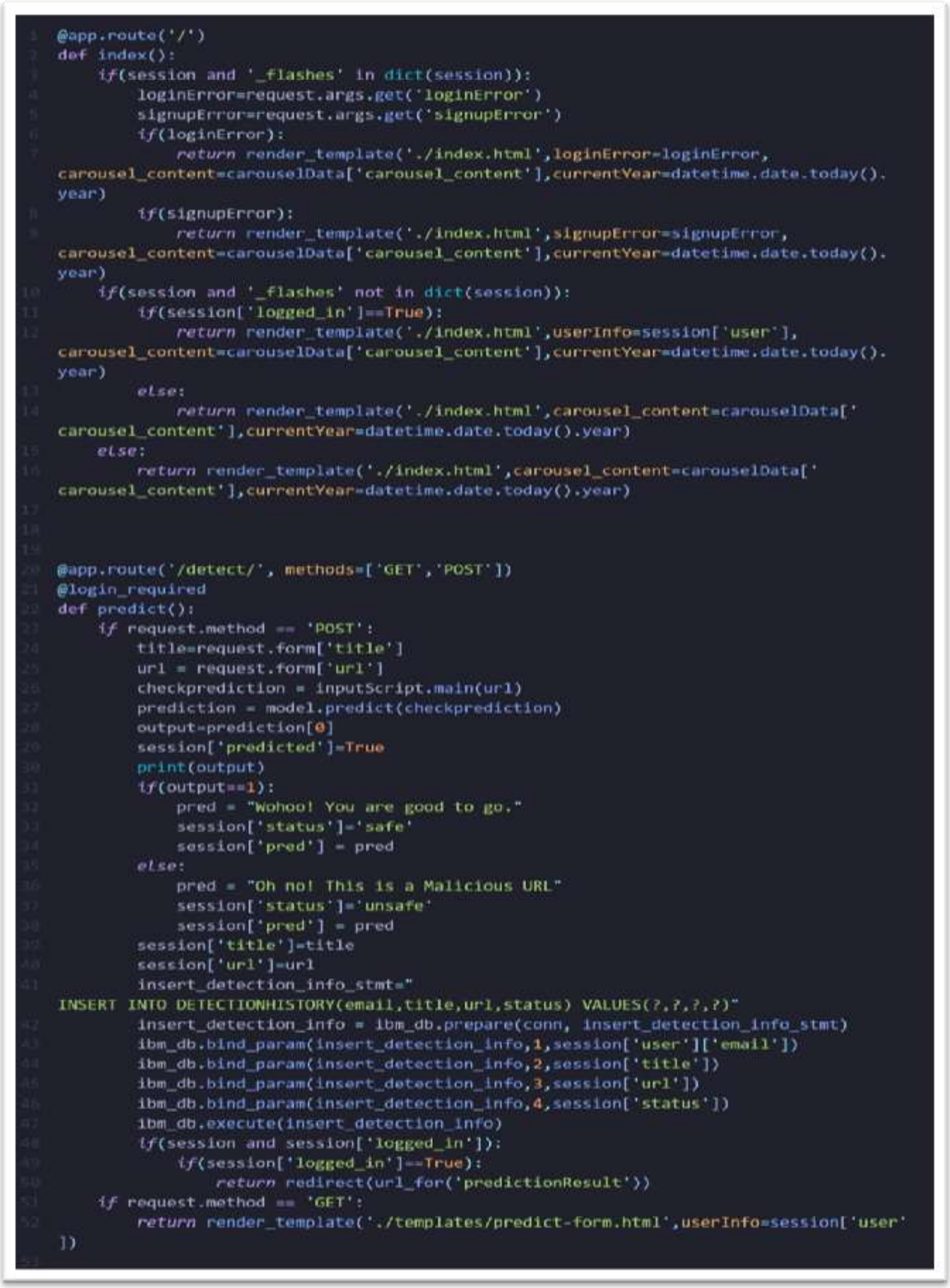
```
import datetime
import os
from os.path import join, dirname
from dotenv import load_dotenv
from functools import wraps
from http.client import HTTPException
import numpy as np
from flask import Flask, request, render_template,session,
 url_for,redirect,flash
import json
import pickle
import inputScript
from passlib.hash import  pbkdf2_sha256
import json
import inputScript
import ibm_db
app = Flask(__name__,template_folder='../Flask')
model = pickle.load(open('../Flask/Phishing_Website.pkl','rb'
))


dotenv_path = join(dirname(__file__), '.env')
load_dotenv(dotenv_path)
conn = ibm_db.connect(os.environ.get('IBMDB_URL'),'','')
SECRET_KEY = os.environ.get("SECRET_KEY")
app.secret_key= SECRET_KEY
carouselDataFile = open('./static/json/carouselData.json')
carouselData = json.load(carouselDataFile)
aboutDataFile = open('./static/json/aboutData.json')
aboutData = json.load(aboutDataFile)
```



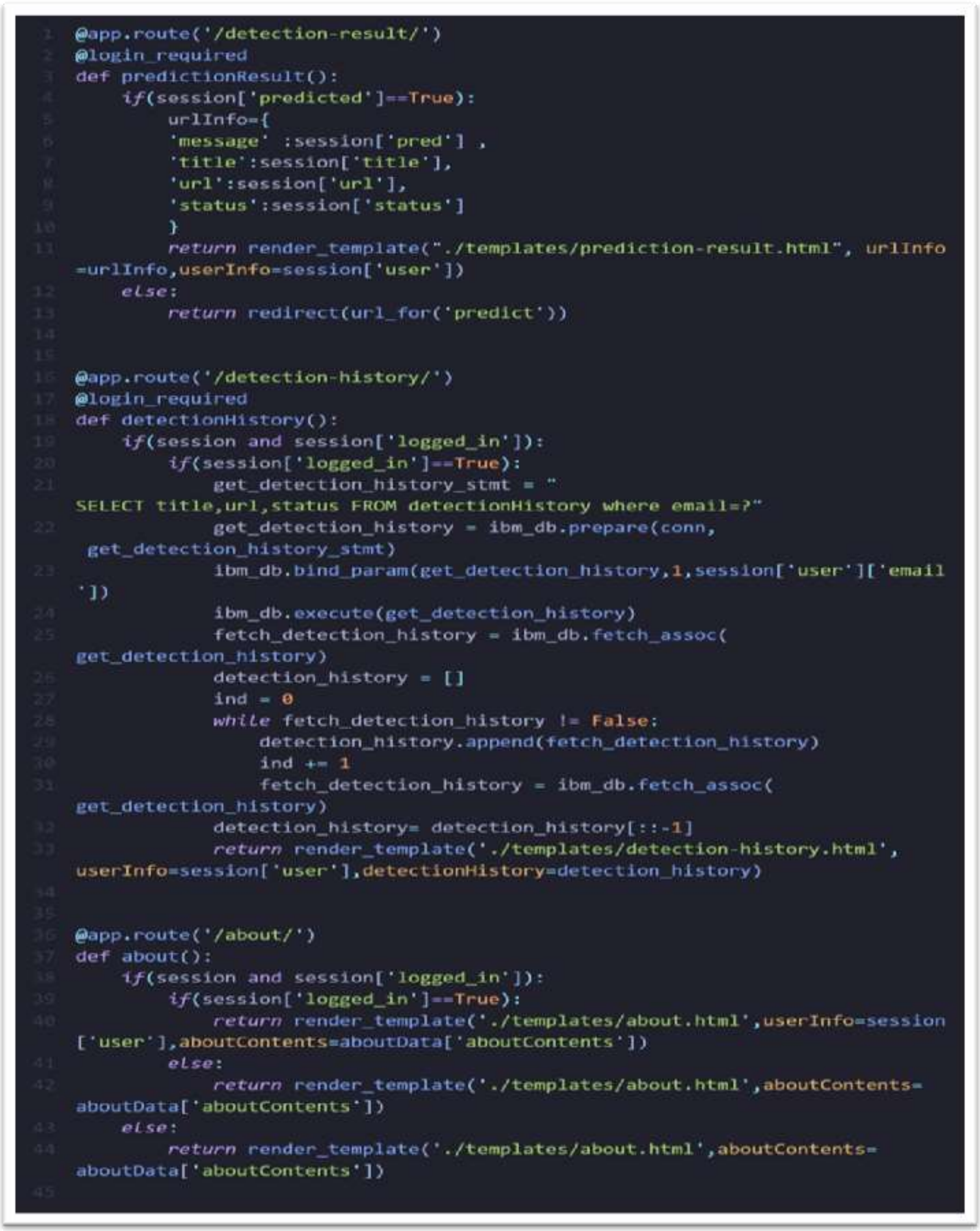
```python
def login_required(f):
    @wraps(f)
    def wrap(*args, **kwargs):
        if('logged_in' in session):
            return f(*args, **kwargs)
        else:
            return redirect('/')
    return wrap


def start_session(userInfo):
    del userInfo['password']
    session['logged_in']=True
    session['user']=userInfo
    session['predicted']=False
    return redirect(url_for('index'))


@app.route('/login/',methods=['POST'])
def login():
    if request.method=="POST":
        email=request.form.get("email")
        password=request.form.get("password")
        verify_account = "SELECT * FROM account WHERE email =?"
        stmt = ibm_db.prepare(conn, verify_account)
        ibm_db.bind_param(stmt,1,email)
        ibm_db.execute(stmt)
        fetch_account = ibm_db.fetch_assoc(stmt)
        if(fetch_account):
            if(pbkdf2_sha256.verify(password,fetch_account['PASSWORD'])):
                userInfo={
                    "fullName":fetch_account['FULLNAME'],
                    "email":fetch_account['EMAIL'],
                    "phoneNumber":fetch_account['PHONENUMBER'],
                    "password":fetch_account['PASSWORD'],
                }
                return start_session(userInfo)
            else:
                flash("Password is incorrect","loginError")
                return redirect(url_for('index',loginError=True))
        flash("Sorry, user with this email id does not exist","loginError")
        return redirect(url_for('index',loginError=True))
```



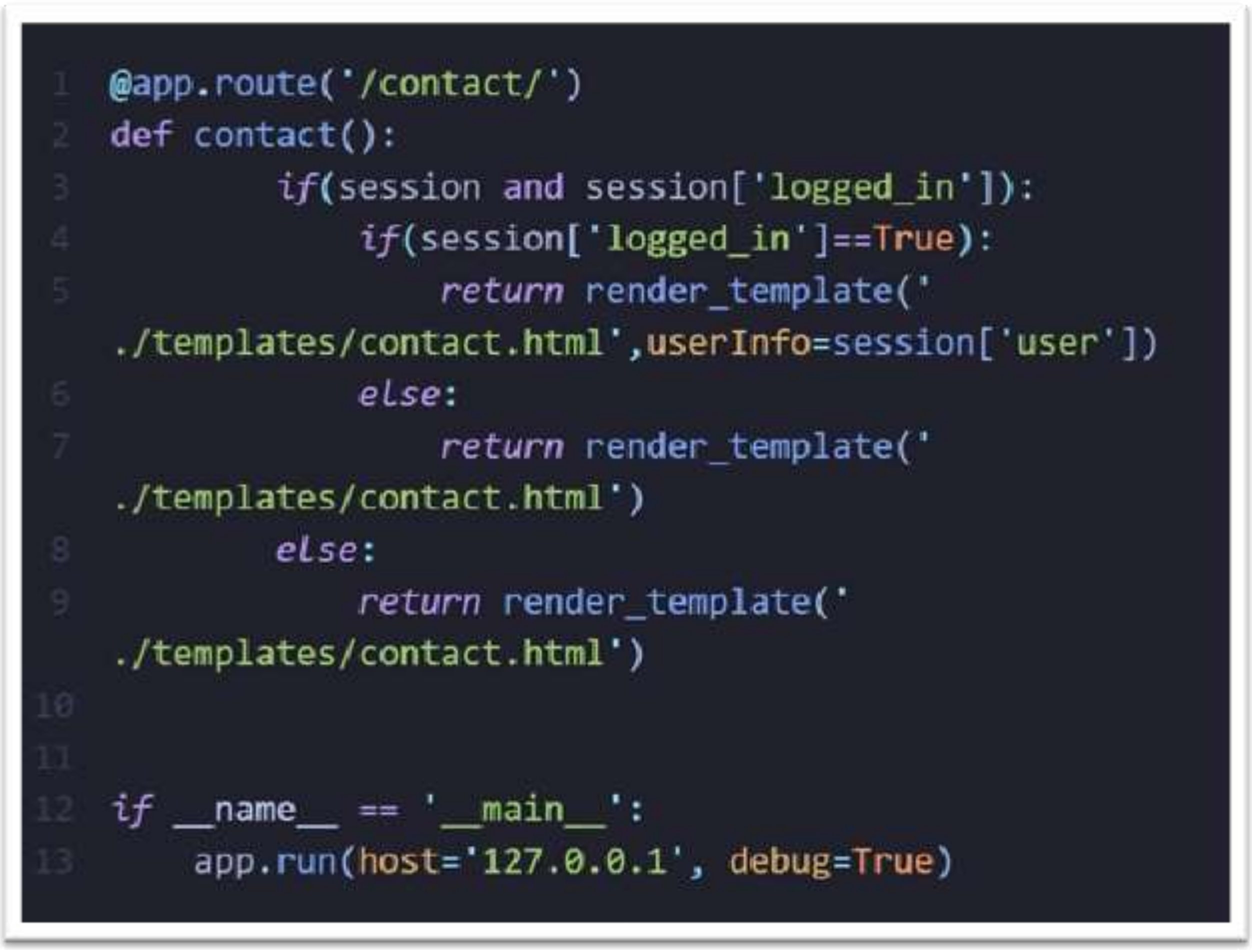
```python
def login_required(f):
    @wraps(f)
    def wrap(*args, **kwargs):
        if('logged_in' in session):
            return f(*args, **kwargs)
        else:
            return redirect('/')
    return wrap


def start_session(userInfo):
    del userInfo['password']
    session['logged_in']=True
    session['user']=userInfo
    session['predicted']=False
    return redirect(url_for('index'))


@app.route('/login/',methods=['POST'])
def login():
    if request.method=="POST":
        email=request.form.get("email")
        password=request.form.get("password")
        verify_account = "SELECT * FROM account WHERE email =?"
        stmt = ibm_db.prepare(conn, verify_account)
        ibm_db.bind_param(stmt,1,email)
        ibm_db.execute(stmt)
        fetch_account = ibm_db.fetch_assoc(stmt)
        if(fetch_account):
            if(pbkdf2_sha256.verify(password,fetch_account['PASSWORD'])):
                userInfo={
                    "fullName":fetch_account['FULLNAME'],
                    "email":fetch_account['EMAIL'],
                    "phoneNumber":fetch_account['PHONENUMBER'],
                    "password":fetch_account['PASSWORD'],
                }
                return start_session(userInfo)
            else:
                flash("Password is incorrect","loginError")
                return redirect(url_for('index',loginError=True))
        flash("Sorry, user with this email id does not exist","loginError")
        return redirect(url_for('index',loginError=True))
```



```
@app.route('/')
def index():
    if(session and '_flashes' in dict(session)):
        loginError=request.args.get('loginError')
        signupError=request.args.get('signupError')
        if(loginError):
            return render_template('./index.html',loginError=loginError,
carousel_content=carouselData['carousel_content'],currentYear=datetime.date.today().
year)
        if(signupError):
            return render_template('./index.html',signupError=signupError,
carousel_content=carouselData['carousel_content'],currentYear=datetime.date.today().
year)
    if(session and '_flashes' not in dict(session)):
        if(session['logged_in']==True):
            return render_template('./index.html',userInfo=session['user'],
carousel_content=carouselData['carousel_content'],currentYear=datetime.date.today().
year)
        else:
            return render_template('./index.html',carousel_content=carouselData['
carousel_content'],currentYear=datetime.date.today().year)
    else:
        return render_template('./index.html',carousel_content=carouselData['
carousel_content'],currentYear=datetime.date.today().year)



@app.route('/detect/', methods=['GET','POST'])
@login_required
def predict():
    if request.method == 'POST':
        title=request.form['title']
        url = request.form['url']
        checkprediction = inputScript.main(url)
        prediction = model.predict(checkprediction)
        output=prediction[0]
        session['predicted']=True
        print(output)
        if(output==1):
            pred = "Wohoo! You are good to go."
            session['status']='safe'
            session['pred'] = pred
        else:
            pred = "Oh no! This is a Malicious URL"
            session['status']='unsafe'
            session['pred'] = pred
        session['title']=title
        session['url']=url
        insert_detection_info_stmt="
INSERT INTO DETECTIONHISTORY(email,title,url,status) VALUES(?,?,?,?)"
        insert_detection_info = ibm_db.prepare(conn, insert_detection_info_stmt)
        ibm_db.bind_param(insert_detection_info,1,session['user']['email'])
        ibm_db.bind_param(insert_detection_info,2,session['title'])
        ibm_db.bind_param(insert_detection_info,3,session['url'])
        ibm_db.bind_param(insert_detection_info,4,session['status'])
        ibm_db.execute(insert_detection_info)
        if(session and session['logged_in']):
            if(session['logged_in']==True):
                return redirect(url_for('predictionResult'))
    if request.method == 'GET':
        return render_template('./templates/predict-form.html',userInfo=session['user'
])
```



```python
@app.route('/detection-result/')
@login_required
def predictionResult():
    if(session['predicted']==True):
        urlInfo={
        'message' :session['pred'] ,
        'title':session['title'],
        'url':session['url'],
        'status':session['status']
        }
        return render_template("./templates/prediction-result.html", urlInfo=urlInfo,userInfo=session['user'])
    else:
        return redirect(url_for('predict'))


@app.route('/detection-history/')
@login_required
def detectionHistory():
    if(session and session['logged_in']):
        if(session['logged_in']==True):
            get_detection_history_stmt = "SELECT title,url,status FROM detectionHistory where email=?"
            get_detection_history = ibm_db.prepare(conn, get_detection_history_stmt)
            ibm_db.bind_param(get_detection_history,1,session['user']['email'])
            ibm_db.execute(get_detection_history)
            fetch_detection_history = ibm_db.fetch_assoc(get_detection_history)
            detection_history = []
            ind = 0
            while fetch_detection_history != False:
                detection_history.append(fetch_detection_history)
                ind += 1
                fetch_detection_history = ibm_db.fetch_assoc(get_detection_history)
            detection_history= detection_history[::-1]
            return render_template('./templates/detection-history.html', userInfo=session['user'],detectionHistory=detection_history)


@app.route('/about/')
def about():
    if(session and session['logged_in']):
        if(session['logged_in']==True):
            return render_template('./templates/about.html',userInfo=session['user'],aboutContents=aboutData['aboutContents'])
        else:
            return render_template('./templates/about.html',aboutContents=aboutData['aboutContents'])
    else:
        return render_template('./templates/about.html',aboutContents=aboutData['aboutContents'])
```



```
@app.route('/contact/')
def contact():
        if(session and session['logged_in']):
            if(session['logged_in']==True):
                return render_template('
./templates/contact.html',userInfo=session['user'])
            else:
                return render_template('
./templates/contact.html')
        else:
            return render_template('
./templates/contact.html')


if __name__ == '__main__':
    app.run(host='127.0.0.1', debug=True)
```

13.2 Github & Project Demo Link

Github link

https://github.com/IBM-EPBL/IBM-Project-49479-1660819950

Project demo link

https://youtu.be/j9w-XH7qmdM