3.7 CODE DESIGN :

#index.html html>

{% load static %}

<html>

<head>

<title>URL Analyzer - Detect Fake URLs</title>

<link rel="icon" href=" { % static 'urlanalyzer/favicon.ico' %} " type="image/x-icon">

<style> body { font-family: Arial, sans-serif; max-width: 800px; margin: 0 auto; padding: 20px;

.container { margin-top: 30px;

#url-input { width: 100%; padding: IOpx;

font-size: 16px; margin-bottom: IOpx;

#analyze-btn {

padding: 10px 20px; background: #4CAF50; color: white; border: none; cursor: pointer;

font-size: 16px;

#analyze-btn:hover { background: #45a049;

#results { margin-top: 20px; padding: 15px; border-radius: 5px; .safe { background-color: #dffDd8; color: #3c763d; border: Ipx solid #d6e9c6;

.warning {

background-color: #fcße3 ; color: #8a6d3b; border: Ipx solid #faebcc;

.danger {

background-color: #f2dede; color: #a94442; border: Ipx solid #ebccdl ;

.warning-item { margin: 5px 0;

padding: 5px;

.loading { display: none;

margin-top: IOpx;

</style> </head>

<body>



<p>Check if a URL might be fake or malicious</p>

<div class="container">

<form id="url-form">

{ % csrf\_token %}

<input type="text" id="url-input" placeholder="Enter URL to analyze (e.g., https://example.com)" required>

<button type="submit" id="analyze-btn">Analyze URL</button> </form>

<div id—"loading" class="loading">Analyzing URL, please wait...</div>

<div id—"results" style="display: none;"></div>

</div>

<script> document.getElementById('url-form').addEventListener('submit', function(e) {

e.preventDefault(); const url = document.getElementById('url-input').value.trim(); const resultsDiv = document.getElementById('results'); const loadingDiv = document.getElementById('loading');

if (!url) { resultsDiv.style.display = 'block'; resultsDiv.className = 'danger'; resultsDiv.innerHTML = '<strong>Error:</strong> Please enter a URL'; return;

// Show loading indicator loadingDiv.style.display = 'block'; resultsDiv.style.display = 'none'; fetchC/analyze/', { method: 'POST', headers: {

'Content-Type': 'application/x-www-form-urlencoded',

'X-CSRFToken': document.querySelector('[name=csrfmiddlewaretoken]').value

body: 'url=$ {encodeURIComponent(url)}

.then(response => response.json())

.then(data => { loadingDiv.style.display = 'none'; resultsDiv.style.display = 'block'; if (data.error) { resultsDiv.className = 'danger'; resultsDiv.innerHTML = '<strong>Error:</strong> $ {data.error} return;

if (data.is\_malicious) { resultsDiv.className = 'danger';

|  |  |
| --- | --- |
| resultsDiv.innerHTML =  } else { resultsDiv.className = 'safe'; | Warning: This URL appears suspicious</h3>'; |
| resultsDiv.innerHTML = | This URL appears safe</h3>'; |

if (data.warnings && data.warnings.length) {

resultsDiv.innerHTML '<h4>Potential issues:</h4><div



const warningsDiv = document.getElementById('warnings'); data.warnings.forEach(warning => {

warningsDiv.innerHTML <div class="warning-item">•

$ {warning}</div>' ;

if (data.analysis) { resultsDiv.innerHTML += '<h4>Analysis details :</h4><pre>' +

JSON.stringify(data.analysis, null, 2) +

.catch(error => { loadingDiv.style.display = 'none'; resultsDiv.style.display = 'block'; resultsDiv.className = 'danger';

resultsDiv.innerHTML = 'An error occurred while analyzing the URL:

error.message;

console.error('Error:', error);

</script> </body>

#urlanalyzer/urls.py from django.urls import path from . import views urlpatterns = [ path(", views.index, name='index'), path('analyze/', views.analyze url, name='analyze url'),

#utils.py import re import requests import tldextract import whois from urllib.parse import urlparse from datetime import datetime class URLAnalyzer:

def \_init\_(self, url):

self.url = url self.parsed\_url = urlparse(url) self.domain info = tldextract.extract(url) self.results =

furl': url,

'is malicious': False,

'warnings': [J'

'analysis': { }

def :

Long URLs are often suspicious if len(self.url) > 75:

self.results['warnings'].append("URL is unusually long") self.results['analysis']['url length'] = len(self.url) def check shortened(self):

Check if URL uses shortening services shorteners = ['bit.ly', 'goo.gl', 'tinyurl', 'ow.ly', 't.co', 'is.gd'] for service in shorteners:

if service in self.url:

uses shortening service ( {service})") self.results['analysis']['shortener'] = service break def :

Excessive special characters can be suspicious special\_chars = re.findall(r'[A\w\-\.\A:]', self.url) if len(special\_chars) > 3 : self.results['warnings'].append("URL contains excessive special characters") self.results['analysis']['special\_chars'] = len(special\_chars) def check ip\_address(self):

URLs with IP addresses instead of domains are suspicious

ip\_pattern = {1,3 '

if re.search(ip\_pattern, self.url):

self.results['warnings'].append("URL contains IP address instead of domain") self.results['analysis']['contains\_ip'] = True

def check\_domain\_age(self) :

'Newly registered domains are more likely to be malicious try:

domain = self.domain info.registered domain if not domain: # If we can't extract domain return domain info = whois.whois(domain) if domain info.creation date:

if isinstance(domain info.creation\_date, list):

creation date = domain info.creation\_date[0] else:

creation date = domain info.creation date

age = (datetime.now() - creation\_date).days if age < 30: # Less than 30 days old self.results['warnings'] .append(f'Domain is very new ( {age} days)") self.results['analysis']['domain age\_days'] = age except Exception as e:

not check domain age: {str(e)} def check https(self):

Check if URL uses HTTPS if self.parsed\_url.scheme != 'https':

self.results['warnings'].append("URL does not use HTTPS") self.results['analysis']['uses\_https'] = False

def check suspicious\_keywords(self):

Check for common phishing keywords in domain keywords = ['login', 'account', 'verify', 'secure', 'bank', 'paypal', 'update'] for keyword in keywords:

if keyword in self.url.lower():

contains suspicious keyword: {keyword} ") self.results['analysis']['suspicious\_keywords'] = True break def check subdomains(self):

Excessive subdomains can be suspicious if len(self.domain\_info.subdomain.split('.')) > 2:

self.results['warnings'].append("URL has excessive subdomains") self.results['analysis']['subdomain\_count'] = len(self.domain\_info.subdomain.split('.')) def analyze(self):

Run all checks self.check url length() self.check shortened() self.check\_special\_chars() self.check ip\_address() self.check domain\_age() self.check\_https() self. check\_suspicious\_keywords() self. check\_subdomains()

# If we have warnings, mark as potentially malicious if self.results['warnings'] :

self.results['is malicious'] True return self.results

#views.py from django.shortcuts import render from django.http import JsonResponse from .utils import URLAnalyzer from django. template.loader import render\_to\_string from django.http import HttpResponse def index(request):

return render(request, 'urlanalyzer/index.html') def analyze\_url(request):

if request.method == 'POST':

url = request.POST.get('url', ").strip() if not url:

return JsonResponse({'error': 'URL is required'} , status=400)

# Add http:// if missing if not url.startswith(('http://', 'https://')):

url = 'http://' + url analyzer = URLAnalyzer(url) results = analyzer.analyze() return JsonResponse(results)

return JsonResponse({'error': 'Only POST method allowed'}, status—405)

#main url from django.contrib import admin from django.urls import path, include urlpatterns = [ path('admin/', admin.site.urls), path(", include('urlanalyzer.urls')),