A

Major Project

On

Phishing URL Detection
A Real Case Scenario Through Login URLs

(Submitted in partial fulfillment of the requirements for the award of Degree)

BACHELOR OF TECHNOLOGY

In

COMPUTER SCIENCE AND ENGINEERING

By

Madala Hemanth (207R1A0593)

Jupalli Venkata Sai (207R1A0582)

Nimmala Nithish (207R1A05A7)

Under the Guidance of

B.P Deepak Kumar

(Assistant Professor)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CMR TECHNICAL CAMPUS

UGC AUTONOMOUS

(Accredited by NAAC, NBA, Permanently Affiliated to JNTUH, Approved by AICTE, New Delhi) Recognized Under Section 2(f) & 12(B) of the UGCAct.1956, Kandlakoya (V), Medchal Road, Hyderabad-501401.

2020-2024

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

This is to certify that the project entitled **"Phishing URL Detection-A Real Case Scenario Through Login URLs"** being submitted by **Madala Hemanth (207R1A0593), Jupalli Venkata Sai (207R1A0582) & Nimmala Nithish (207R1A05A7)** in partial fulfillment of the requirements for the award of the degree of B.Tech in Computer Science and Engineering to the Jawaharlal Nehru Technological University Hyderabad, is a record of bonafide work carried out by them under our guidance and supervision during the year 2023-24.

The results embodied in this thesis have not been submitted to any other University or Institute for the award of any degree or diploma.

B.P Deepak Kumar
(Associate Professor)
INTERNAL GUIDE

Dr. A. Raji Reddy
DIRECTOR

Dr. K. Srujan Raju
HOD

EXTERNAL EXAMINER

Submitted for viva voice Examination held on _____

ACKNOWLEDGEMENT

Apart from the efforts of us, the success of any project depends largely on the encouragement and guidelines of many others. We take this opportunity to express our gratitude to the people who have been instrumental in the successful completion of this project.

We take this opportunity to express my profound gratitude and deep regard to my guide **Mr. B.P Deepak Kumar,** Associate Professor for his exemplary guidance, monitoring andconstant encouragement throughout the project work. The blessing, help and guidance given by him shall carry us a long way in the journey of life on which we are about to embark.

We also take this opportunity to express a deep sense of gratitude to the Project Review Committee (PRC) **G. Vinesh Shanker, Dr. J. Narasimha rao, Ms. Shilpa, & Dr. K. Maheswari** for their cordial support, valuable information and guidance, which helped us in completing this task through various stages.

We are also thankful to **Dr. K. Srujan Raju,** Head, Department of Computer Science and Engineering for providing encouragement and support for completing this project successfully.

We are obliged to **Dr. A. Raji Reddy,** Director for being cooperative throughout the course of this project. We also express our sincere gratitude to Sri. **Ch. Gopal Reddy,** Chairman for providing excellent infrastructure and a nice atmosphere throughout the course of this project.

The guidance and support received from all the members of **CMR Technical Campus** who contributed to the completion of the project. We are grateful for their constant support and help.

Finally, we would like to take this opportunity to thank our family for their constant encouragement, without which this assignment would not be completed. We sincerely acknowledge and thank all those who gave support directly and indirectly in the completion of this project.

Madala Hemanth (207R1A0593)

Jupalli Venkata Sai (207R1A0582)

Nimmala Nithish (207R1A05A7)

ABSTRACT

Phishing is a social engineering cyberattack where criminals deceive users to obtain their credentials through a login form that submits the data to a malicious server. In this paper, we compare machine learning and deep learning techniques to present a method capable of detecting phishing websites through URL analysis. In most current state-of-the-art solutions dealing with phishing detection, the legitimate class is made up of homepages without including login forms. On the contrary, we use URLs from the login page in both classes because we consider it is much more representative of a real case scenario and we demonstrate that existing techniques obtain a high false-positive rate when tested with URLs from legitimate login pages. Additionally, we use datasets from different years to show how models decrease their accuracy over time by training a base model with old datasets and testing it with recent URLs. Also, we perform a frequency analysis over current phishing domains to identify different techniques carried out by phishers in their campaigns. To prove these statements, we have created a new dataset named Phishing Index Login URL (PILU-90K), which is composed of 60K legitimate URLs, including index and login websites, and 30K phishing URLs. Finally, we present a Logistic Regression model which, combined with Term Frequency - Inverse Document Frequency (TF-IDF) feature extraction, obtains 96:50% accuracy on the introduced login URL dataset.

.

LIST OF FIGURES/TABLES

LIST OF SCREENSHOTS

TABLE OF CONTENTS

1. INTRODUCTION

1.INTRODUCTION

1.1 PROJECT SCOPE

Develop a phishing URL detection system for login pages using machine learning. Collect and analyze datasets of legitimate and phishing URLs. Train a model to distinguish between the two, considering common phishing tactics. Implement a real-time analysis and alert system with a user-friendly interface. Provide API integration and ongoing updates for adaptability.

1.2 PROJECT PURPOSE

Create a phishing URL detection system to enhance online security by identifying and preventing login page phishing attacks. Utilize machine learning to distinguish between legitimate and phishing URLs, offering real-time analysis and alerts. Develop a user-friendly interface for easy interaction and integrate an API for broader usability. The project aims to empower users to identify potential threats, reducing the risk of falling victim to phishing attacks on login pages. Ongoing updates ensure adaptability to evolving phishing tactics.

1.3 PROJECT FEATURES

The project focuses on developing a cutting-edge phishing URL detection system tailored for login pages. Leveraging machine learning, it employs advanced algorithms to discern between legitimate and phishing URLs, enhancing overall online security. The system ensures real-time analysis, offering users swift insights into potential threats. A user-friendly interface facilitates easy interaction, allowing users to input URLs effortlessly. An integrated alert system promptly notifies users of potential phishing risks, fostering proactive cybersecurity practices. Additionally, the project features API integration, enabling seamless collaboration with other applications and systems for broader utility.

2.SYSTEM ANALYSIS

2. SYSTEM ANALYSIS

SYSTEM ANALYSIS

System Analysis is the important phase in the system development process. The System is studied to the minute details and analyzed. The system analyst plays an important role of an interrogator and dwells deep into the working of the present system0.69. In analysis, a detailed study of these operations performed by the system and their relationships within and outside the system is done. A key question considered here is, “what must be done to solve the problem?” The system is viewed as a whole and the inputs to the system are identified. Once analysis is completed the analyst has a firm understanding of what is to be done.

2.1 PROBLEM DEFINITION

Addressing the escalating threat of phishing attacks on login pages, this project aims to develop a machine learning-based system for real-time detection. The challenge lies in the increasing sophistication of phishing tactics, necessitating a proactive defense mechanism. Existing solutions often lack the adaptability required to keep pace with evolving threats, emphasizing the need for a robust and agile detection system. The goal is to enhance online security by empowering users to discern between legitimate and malicious URLs effectively.

2.2 EXISTING SYSTEM

Moghimi and Vorjani proposed a system independent from third services like Google Page Rank or WHOIS. They used two handcrafted feature sets, extracted from the URL and the Document Object Model (DOM) of the website. The first set has nine legacy features including a set of keywords, while the second has eight novel features which inform of whether the website's resources are loaded using SSL protocol or not. They used Levenshtein distance to detect typo-squatting by comparing the website and resources URLs. These features were used to train an SVM classifier and obtained an accuracy of 98:65% on their banking websites dataset.

2.2.1 LIMITATIONS OF EXISTING SYSTEM

Following are the disadvantages of existing system:

- From an existing perspective, and to the best of our knowledge, publicly available datasets are not reflecting conditions that represent some real problems for phishing URL detection.
- It is observed that recent machine learning proposals obtained high accuracy using outdated datasets, i.e., typically containing URLs collected from 2009 to 2017. We demonstrate that models trained with old URLs decrease their performance when they are tested with URLs coming from recent phishing pages.

2.3 PROPOSED SYSTEM

This project presents a phishing URL dataset using legitimate login websites to obtain the URLs from such pages. Then, we evaluate machine and deep learning techniques for recommending the method with higher accuracy. Next, we show how models trained with legitimate homepages struggle to classify legitimate login URLs, demonstrating our hypothesis about phishing detection and legitimate login URLs. Additionally, we show how the accuracy decrease with the time on models trained with datasets from 2016 and evaluated on data collected in 2020. Finally, we provide an overview of current phishing encounters, explaining attacker tricks and approaches.

2.3.1 ADVANTAGES OF THE PROPOSED SYSTEM

Following are the advantages of proposed system:

- Machine learning models to detect unreported phishing encounters. Depending on their input data, these approaches can be classified into two categories: URL-based and content based.
- present an extended version of the Phishing Index Login URL (PILU-60K) dataset [20] and we name it PILU-90K. PILU-90K contains 90K URLs divided into three classes.

2.4 FEASIBILITY STUDY

The feasibility of the project is analyzed in this phase and a business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the company. Three key considerations involved in the feasibility analysis:

- Economic Feasibility
- Technical Feasibility
- Social Feasibility

2.4.1 ECONOMIC FEASIBILITY

The developing system must be justified by cost and benefit. Criteria to ensure that effort is concentrated on a project, which will give best, return at the earliest. One of the factors, which affect the development of a new system, is the cost it would require.

The following are some of the important financial questions asked during preliminary investigation:

- The costs conduct a full system investigation.
- The cost of the hardware and software.
- The benefits in the form of reduced costs or fewer costly errors.

Since the system is developed as part of project work, there is no manual cost to spend for the proposed system. Also all the resources are already available, it give an indication that the system is economically possible for development.

2.4.2 TECHNICAL FEASIBILITY

This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. The developed system must have a modest requirement, as only minimal or null changes are required for implementing this system.

2.4.3 SOCIAL FEASIBILITY

This includes the following questions:

- Is there sufficient support for the users?
- Will the proposed system cause harm?

The project would be beneficial because it satisfies the objectives when developed and installed. All behavioral aspects are considered carefully and conclude that the project is behaviorally feasible

2.5 HARDWARE & SOFTWARE REQUIREMENTS

2.5.1 HARDWARE REQUIREMENTS:

Hardware interfaces specify the logical characteristics of each interface between the software product and the hardware components of the system. The following are some hardware requirements.

- Processor : Pentium IV
- Hard disk : 20 GB
- RAM : 4 GB(min)

2.5.2 SOFTWARE REQUIREMENTS:

Software Requirements specifies the logical characteristics of each interface and software components of the system. The following are some software requirements,

- Operating system : Windows 7 Ultimate
- Languages : Python
- Front-End : Python
- Bach-End : Django-ORM
- Designing : HTML, CSS, JavaScript

3. ARCHITECTURE

3. ARCHITECTURE

3.1 PROJECT ARCHITECTURE

This project architecture shows the procedure followed for classification, starting from input to final prediction.

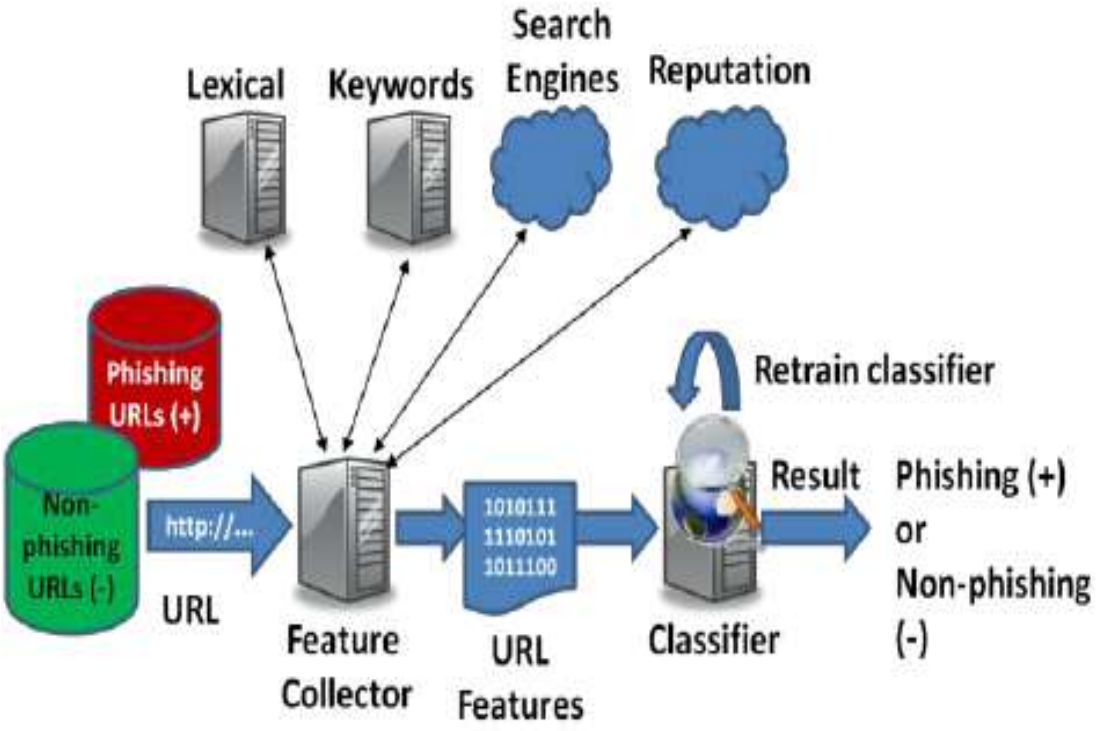


Figure 3.1: Project Architecture for Phishing URL Detection A Real-Case Scenario Through Login URLs

3.2 DESCRIPTION

This project endeavors to develop a sophisticated phishing URL detection system with a specific focus on safeguarding login pages. The escalating complexity of phishing attacks poses a severe risk to user credentials and data security. By harnessing the power of machine learning algorithms, the system aims to conduct real-time analysis of URLs, scrutinizing intricate patterns to discern between legitimate and malicious links. The user interface will be thoughtfully designed for seamless interaction, allowing users to effortlessly input URLs for analysis. An integrated alert system will provide prompt notifications, empowering users to proactively respond to potential phishing threats targeting login credentials. The system's adaptive nature is underscored, emphasizing regular updates to stay resilient against the ever-evolving landscape of phishing tactics. API integration will extend the system's functionality, fostering collaboration with a diverse array of applications and platforms. Ultimately, this initiative aspires to significantly elevate online security standards, furnishing users with a robust tool to navigate and mitigate the dynamic challenges posed by phishing attacks on login pages.

3.3 USE CASE DIAGRAM

A use case diagram is a graphical depiction of a user's possible interactions with a system. A use case diagram shows various use cases and different types of usersthe system has. The use cases are represented by either circles or ellipses. The actors are often shown as stick figures.

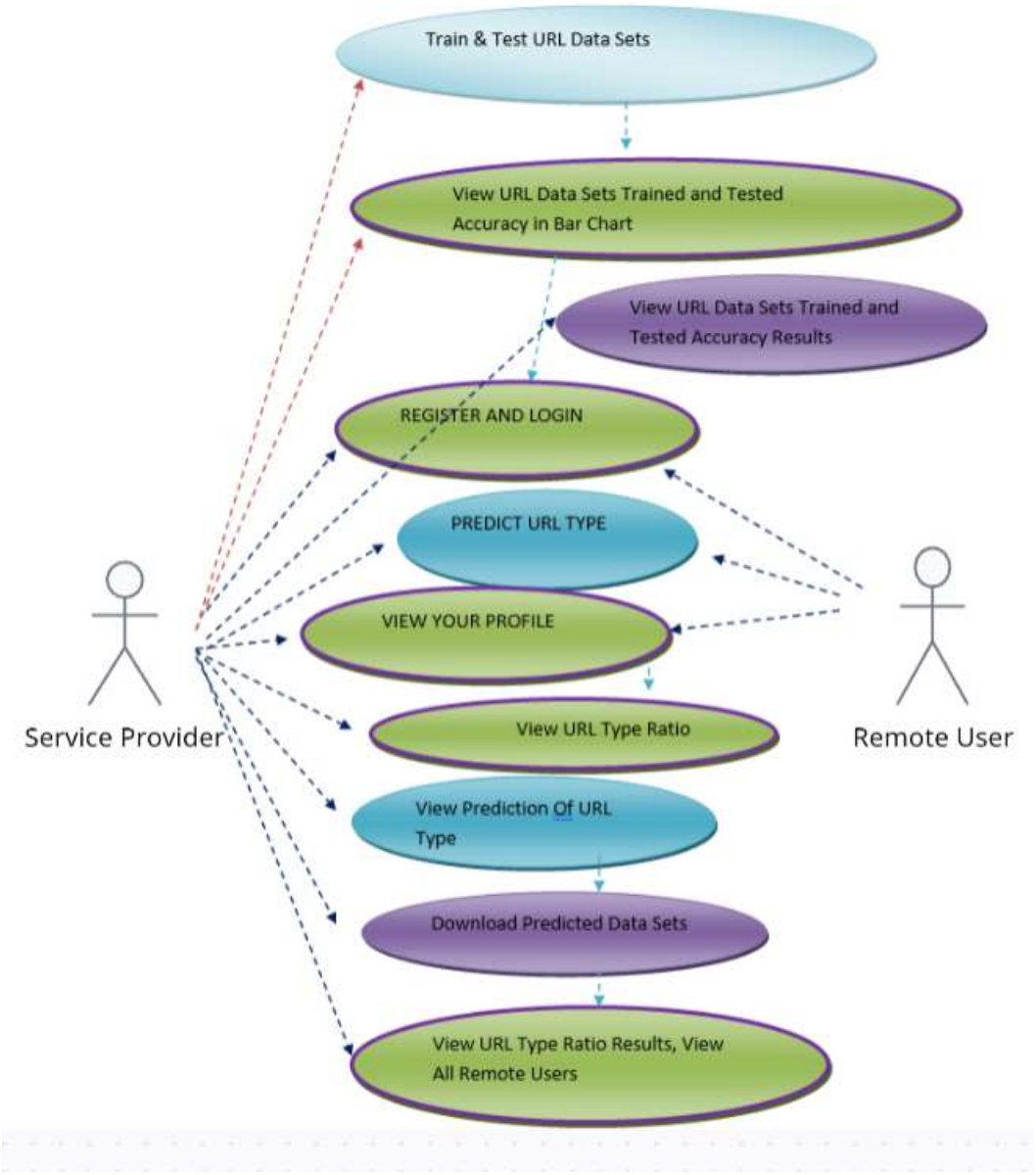


Figure 3.2: Use Case Diagram for Phishing URL Detection A Real-Case Scenario Through Login URLs

3.4 CLASS DIAGRAM

Class diagram is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations(or methods), and the relationships among objects.

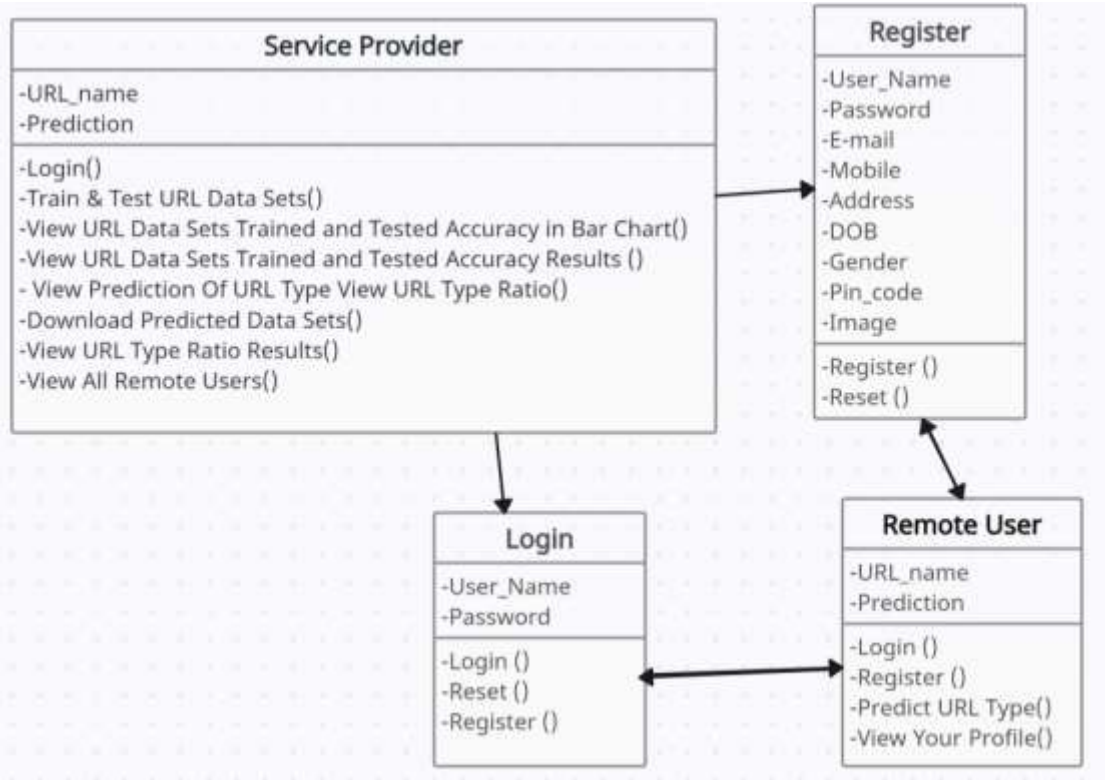


Figure 3.3: Class Diagram for Phishing URL Detection A Real-Case Scenario Through Login URLs

3.5 SEQUENCE DIAGRAM

A sequence diagram shows object interactions arranged in time sequence. It depicts the objects involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of the scenario. Sequence diagrams are typically associated with use case realizations in the logical view of the system under development.

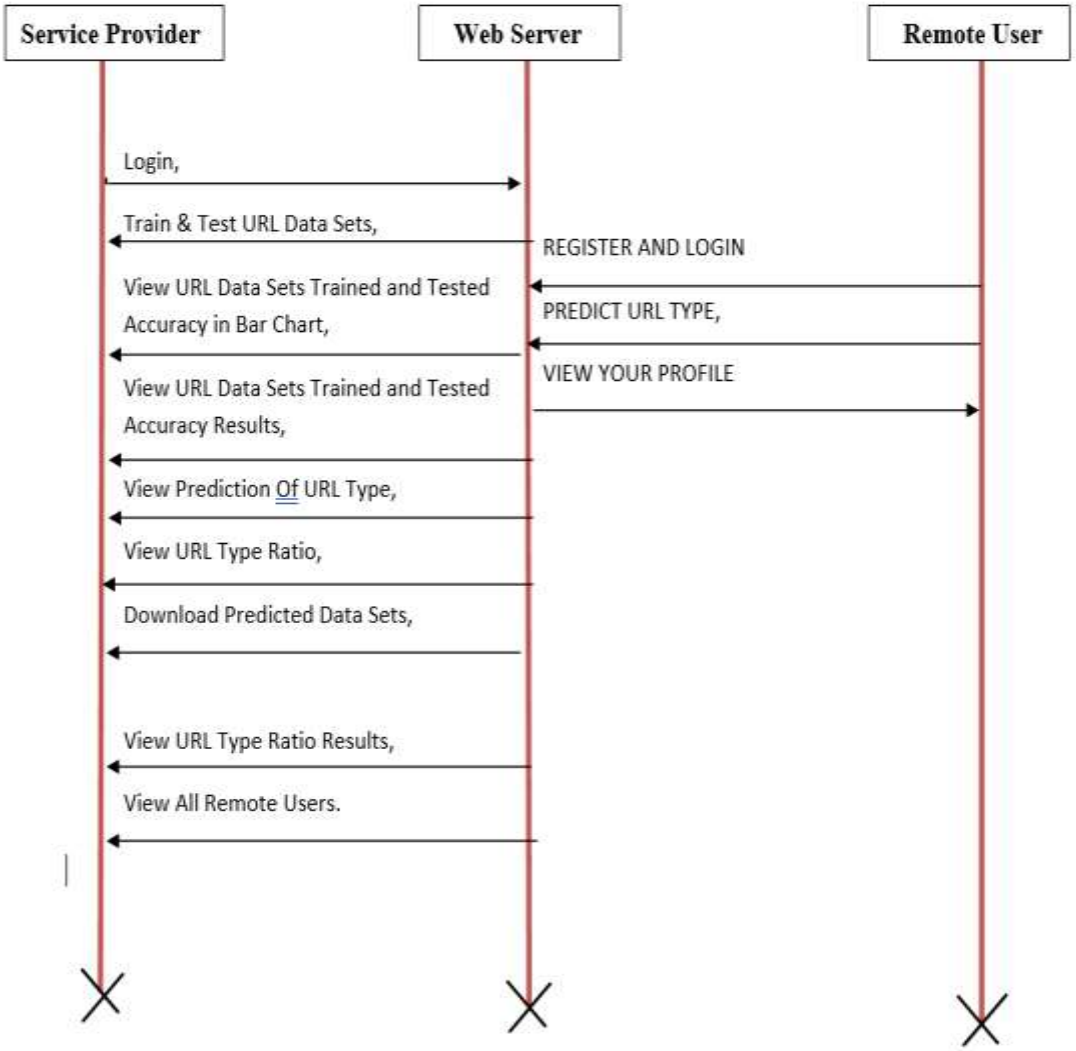


Figure 3.4: Sequence Diagram for Phishing URL Detection A Real-Case Scenario Through Login URLs

3.6 ACTIVITY DIAGRAM

Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency. They can also include elements showing the flow of data between activities through one or more data stores.

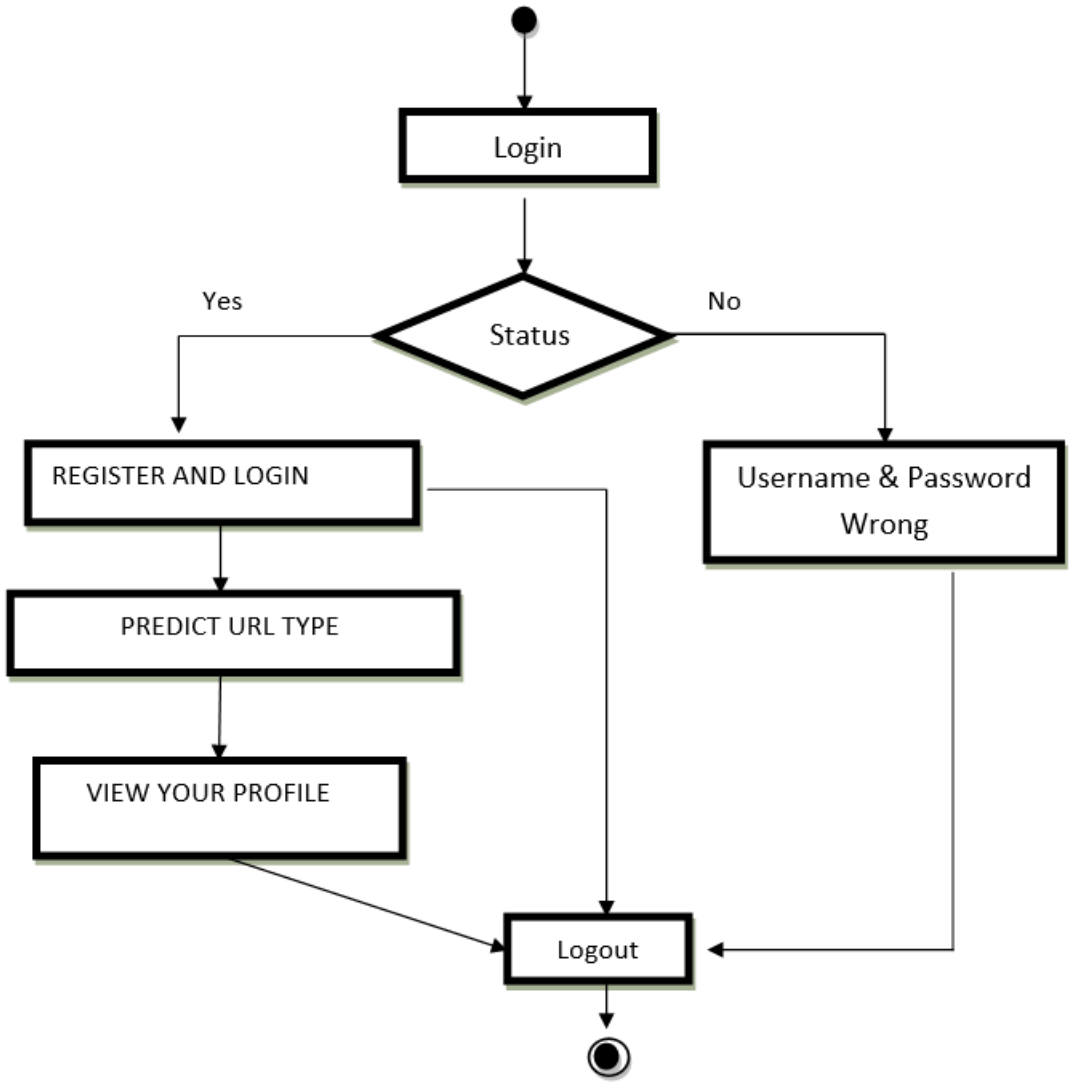


Figure 3.5: Activity Diagram for Phishing URL Detection A Real-Case Scenario Through Login URLs

4. IMPLEMENTATION

4.1 SAMPLE CODE

```
import os
import sys

def main():
    """Run administrative tasks."""
    os.environ.setdefault('DJANGO_SETTINGS_MODULE', 'phishing_url_detection.settings')
    try:
        from django.core.management import execute_from_command_line
    except ImportError as exc:
        raise ImportError(
            "Couldn't import Django. Are you sure it's installed and "
            "available on your PYTHONPATH environment variable? Did you "
            "forget to activate a virtual environment?"
        ) from exc
    execute_from_command_line(sys.argv)


if __name__ == '__main__':
    main()

from django import forms

from Remote_User.models import ClientRegister_Model


class ClientRegister_Form(forms.ModelForm):
    password = forms.CharField(widget=forms.PasswordInput())
    email = forms.EmailField(required=True)

    class Meta:
        model = ClientRegister_Model
        fields = ("username","email","password","phoneno","country","state","city")

from django.db.models import Count
from django.db.models import Q
from django.shortcuts import render, redirect, get_object_or_404
import datetime
import openpyxl

import pandas as pd

from sklearn.feature_extraction.text import CountVectorizer
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
from sklearn.metrics import accuracy_score
```

```
from sklearn.ensemble import VotingClassifier
# Create your views here.
from Remote_User.models import
ClientRegister_Model,url_detection_type,detection_accuracy,detection_ratio

def login(request):


    if request.method == "POST" and 'submit1' in request.POST:

        username = request.POST.get('username')
        password = request.POST.get('password')
        try:
            enter = ClientRegister_Model.objects.get(username=username,password=password)
            request.session["userid"] = enter.id

            return redirect('ViewYourProfile')
        except:
            pass

    return render(request,'RUser/login.html')

def Add_DataSet_Details(request):

    return render(request, 'RUser/Add_DataSet_Details.html', {"excel_data": "})


def Register1(request):
    if request.method == "POST":
        username = request.POST.get('username')
        email = request.POST.get('email')
        password = request.POST.get('password')
        phoneno = request.POST.get('phoneno')
        country = request.POST.get('country')
        state = request.POST.get('state')
        city = request.POST.get('city')
        address = request.POST.get('address')
        gender = request.POST.get('gender')
        ClientRegister_Model.objects.create(username=username, email=email, password=password,
phoneno=phoneno,
                                            country=country, state=state, city=city, address=address,
gender=gender)
        obj = "Registered Successfully"
        return render(request, 'RUser/Register1.html', {'object': obj})

    else:
        return render(request,'RUser/Register1.html')
```

```
from django.db.models import  Count, Avg
from django.shortcuts import render, redirect
from django.db.models import Count
from django.db.models import Q
import datetime
import xlwt
from django.http import HttpResponse
import numpy as np

import pandas as pd

from sklearn.feature_extraction.text import CountVectorizer
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
from sklearn.metrics import accuracy_score

# Create your views here.
from Remote_User.models import
ClientRegister_Model,url_detection_type,detection_accuracy,detection_ratio

def serviceproviderlogin(request):
    if request.method  == "POST":
        admin = request.POST.get('username')
        password = request.POST.get('password')
        if admin == "Admin" and password =="Admin":
            detection_accuracy.objects.all().delete()
            return redirect('View_Remote_Users')

    return render(request,'SProvider/serviceproviderlogin.html')

def View_URL_Type_Ratio(request):
    detection_ratio.objects.all().delete()
    ratio = ""
    kword = 'Non-Phishing'
    print(kword)
    obj = url_detection_type.objects.all().filter(Q(Prediction=kword))
    obj1 = url_detection_type.objects.all()
    count = obj.count();
    count1 = obj1.count();
    ratio = (count / count1) * 100
    if ratio != 0:
        detection_ratio.objects.create(names=kword, ratio=ratio)

    ratio1 = ""
    kword1 = 'Phishing'
    print(kword1)
    obj1 = url_detection_type.objects.all().filter(Q(Prediction=kword1))
    obj11 = url_detection_type.objects.all()
    count1 = obj1.count();
    count11 = obj11.count();
```

```
    ratio1 = (count1 / count11) * 100
    if ratio1 != 0:
        detection_ratio.objects.create(names=kword1, ratio=ratio1)

    ratio12 = ""
    kword12 = 'Defacement'
    print(kword12)
    obj12 = url_detection_type.objects.all().filter(Q(Prediction=kword12))
    obj112 = url_detection_type.objects.all()
    count12 = obj12.count();
    count112 = obj112.count();
    ratio12 = (count12 / count112) * 100
    if ratio12 != 0:
        detection_ratio.objects.create(names=kword12, ratio=ratio12)

    ratio123 = ""
    kword123 = 'Malware'
    print(kword123)
    obj123 = url_detection_type.objects.all().filter(Q(Prediction=kword123))
    obj1123 = url_detection_type.objects.all()
    count123 = obj123.count();
    count1123 = obj1123.count();
    ratio123 = (count123 / count1123) * 100
    if ratio123 != 0:
        detection_ratio.objects.create(names=kword123, ratio=ratio123)

    obj = detection_ratio.objects.all()
    return render(request, 'SProvider/View_URL_Type_Ratio.html', {'objs': obj})

def View_Remote_Users(request):
    obj=ClientRegister_Model.objects.all()
    return render(request,'SProvider/View_Remote_Users.html',{'objects':obj})

def ViewTrendings(request):
    topic = url_detection_type.objects.values('topics').annotate(dcount=Count('topics')).order_by('-
dcount')
    return  render(request,'SProvider/ViewTrendings.html',{'objects':topic})

def charts(request,chart_type):
    chart1 = detection_ratio.objects.values('names').annotate(dcount=Avg('ratio'))
    return render(request,"SProvider/charts.html", {'form':chart1, 'chart_type':chart_type})

def charts1(request,chart_type):
    chart1 = detection_accuracy.objects.values('names').annotate(dcount=Avg('ratio'))
    return render(request,"SProvider/charts1.html", {'form':chart1, 'chart_type':chart_type})

def View_Prediction_Of_URL_Type(request):
    obj =url_detection_type.objects.all()
    return render(request, 'SProvider/View_Prediction_Of_URL_Type.html', {'list_objects': obj})
```

5.SCREENSHOTS

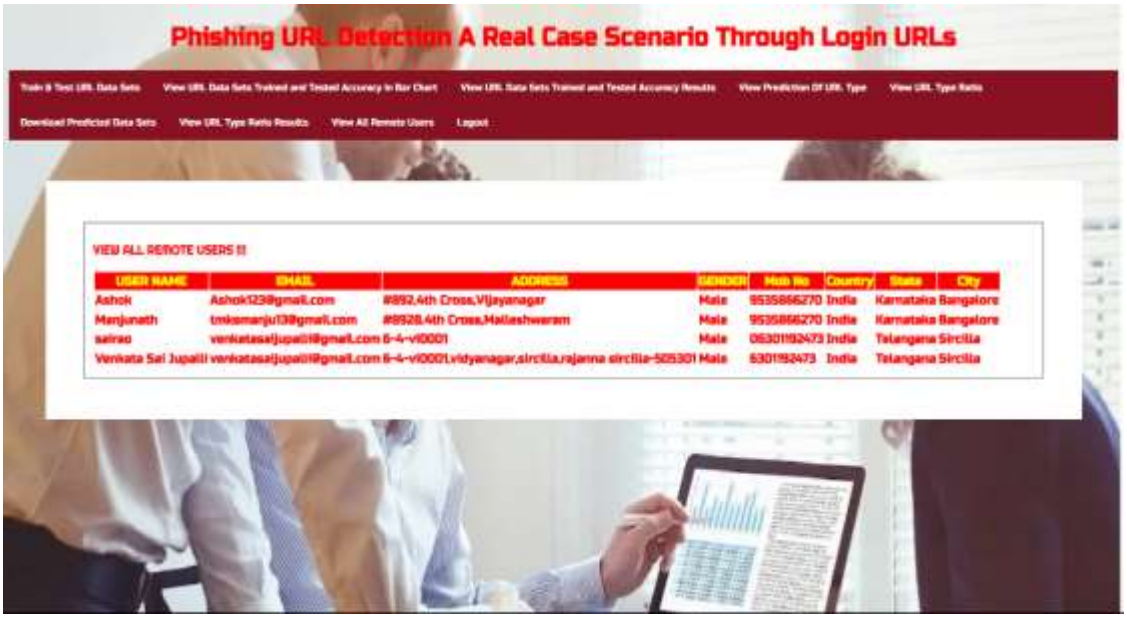
Screenshot 5.1: User Register

Screenshot 5.2: user Login

Screenshot 5.3: User Profile

Screenshot 5.4: URL Predictor

Screenshot 5.5: Admin Login



Screenshot 5.6: Registered Users

Screenshot 5.7: URL Predction Type Details

Screenshot 5.8: URL Type Ratio

6. TESTING

6. TESTING

6.1 INTRODUCTION TO TESTING

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, subassemblies, assemblies and/or a finished product. It is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of tests. Each test type addresses a specific testing requirement.

6.2 TYPES OF TESTING

6.2.1 UNIT TESTING

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application .It is done after the completion of an individual unit before integration. This is a structural testing that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

6.2.2 INTEGRATION TESTING

Integration tests are designed to test integrated software components to determine if they actually run as one program. Integration tests demonstrate thatalthough the components were individually satisfactory, as shown by successfully unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

6.2.3 FUNCTIONAL TESTING

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.
Functional testing is centered on the following items:

Valid Input : identified classes of valid input must be accepted.

Invalid : identified classes of invalid input must be rejected.

Functions : identified functions must be exercised.

Output : identified classes of application outputs must be exercised.

Systems/Procedures: interfacing systems or procedures must be invoked. Organization and preparation of functional tests is focused on requirements, key functions, or special test cases.

6.3 TEST CASES

6.3.1 CLASSIFICATION

Test case ID	Test case name	Purpose	Input	Output
1.	User Register	If User registration successfully.	Pass	If already user email exist then it fails.
2.	user Login	If Username and password is correct then it will getting valid page.	Pass	Un Register Users will not logged in.
3.	User View	Show our dataset	Pass	If Data set Not Available fail.
4.	Admin login	Admin can login with his login credential. If success he get his home page	Pass	Invalid login details will not allowed here
5.	Admin can activate the register users	Admin can activate the register user id	Pass	If user id not found then it won't login
6.	Results	For our models the accuracy	Pass	If Accuracy Not Displayed fail

7. CONCLUSION
&
FUTURE SCOPE

7. CONCLUSION & FUTURE SCOPE

7.1 PROJECT CONCLUSION

In conclusion, this study presents a novel approach to phishing URL detection, focusing on login pages for both legitimate and phishing classifications. By addressing false positives, showcasing temporal model accuracy trends, and uncovering phishing tactics, the research provides valuable insights. The creation of the PILU-90K dataset and the successful application of a Logistic Regression model underscore the significance of the proposed methodology in real-case scenarios.

7.2 FUTURE SCOPE

The feature scope encompasses a range of functionalities crucial for developing an effective phishing URL detection system. This includes analyzing URLs, employing machine learning and deep learning models for classification, utilizing various feature extraction techniques, conducting temporal analysis to detect patterns over time, analyzing phishing domains, implementing strategies for reducing false positives, managing datasets, designing user-friendly interfaces, evaluating performance metrics, integrating with existing security systems, and ensuring scalability and efficiency to handle large volumes of data in real-time or near-real-time environments. These features collectively enable the system to accurately identify phishing URLs and enhance overall cybersecurity measures.

8. BIBLIOGRAPHY

8. BIBLIOGRAPHY

8.1 REFERENCES

[1] Statista. (2020). *Adoption Rate of Emerging Technologies in Organizations Worldwide as of 2020*. Accessed: Sep. 12, 2021. [Online]. Available: https://www.statista.com/statistics/661164/worldwide-cio-surveyoperati%onal-priorities/

[2] R. De', N. Pandey, and A. Pal, ``Impact of digital surge during COVID-19 pandemic: A viewpoint on research and practice," *Int. J. Inf. Manage.*,vol. 55, Dec. 2020, Art. no. 102171.

[3] P. Patel, D. M. Sarno, J. E. Lewis, M. Shoss, M. B. Neider, and C. J. Bohil,``Perceptual representation of spam and phishing emails," *Appl. Cognit.Psychol.*, vol. 33, no. 6, pp. 1296_1304, Nov. 2019.

[4] J. A. Chaudhry, S. A. Chaudhry, and R. G. Rittenhouse, ``Phishing attacks and defenses," *Int. J. Secur. Appl.*, vol. 10, no. 1, pp. 247_256, 2016.

[5] M. Hijji and G. Alam, ``A multivocal literature review on growing social engineering based cyber-attacks/threats during the COVID-19 pandemic: Challenges and prospective solutions," *IEEE Access*, vol. 9,pp. 7152_7169, 2021.

[6] A. Alzahrani, ``Coronavirus social engineering attacks: Issues and recommendations," *Int. J. Adv. Comput. Sci. Appl.*, vol. 11, no. 5,pp. 154_161, 2020.

[7] *Phishing Activity Trends Report 3Q*, Anti-Phishing Working Group, International, 2017. Accessed: Sep. 12, 2021.

8.2 GITHUB LINK

https://github.com/venkatasairao/Phishing_URL_Detection-A_Real_Case_Scenario_Through_Login_URLs