



slington college
(इस्लिङ्टन कलेज)

Module Code & Module Title
CS6P05NI Final Year Project

Assessment Weightage & Type
40% FYP Final Report

Year & Semester
2022/2023 Spring

Student Name: Sarthak Bikram Rana

London Met ID: 20049228

College ID: NP01NT4S210129

Assignment Due Date: 19th April 2023

Assignment Submission Date: 19th April 2023

Submitted To: (Internal Supervisor: Shishir Subedi)

(External Supervisor: Suraj Neupane)

Word Count (Where Required): 9000

I confirm that I understand my coursework needs to be submitted online via Google Classroom under the relevant module page before the deadline in order for my assignment to be accepted and marked. I am fully aware that late submissions will be treated as non-submission and a mark of zero will be awarded.

Acknowledgment

I would want to express my heartfelt gratitude and appreciation to everyone who helped me develop the "Phishing URL Detection System Using Machine Learning " project.

First and foremost, I'd want to thank my internal supervisor, Mr. Shishir Subedi for their crucial advice, encouragement, and support. Their expertise in the domain has been critical in influencing the project's path. Their continual input and constructive criticism have considerably aided in the system's refining and advancement.

I am also grateful to Mr. Suraj Neupane, who gave great insights and comments that helped to widen my grasp of the subject. Their vast expertise and experience in the field of cybersecurity and various other domain have been invaluable in fine-tuning the project and assuring its practical application.

I would also want to thank my mentor, Mr. Sameer Mainali for being a consistent source of encouragement and inspiration during the entire process. Their passion for research and unfailing faith in my skills have given me the courage to face challenges and overcome obstacles. Their guidance has not only improved my technical abilities but has also aided in my development as a researcher and as a person.

In addition, I would like to express my gratitude to my colleagues and friends, who provided me with essential advice, assistance, and companionship during the course of this project. Their willingness to assist, share ideas, and provide support has made this journey genuinely joyful and unforgettable.

Finally, I am grateful to my family, whose unfailing love, support, and understanding have served as the basis for this effort. Their faith in my ability and patience have moved me ahead and given me the strength to endure in the face of hardship. This project would not have been feasible without the combined efforts and contributions of everyone listed above. I express my heartfelt thanks and gratitude to each of you.

Abstract

The project “**Phishing URL Detection Using Machine Learning**” with Visualization Dashboard project offers a cutting-edge solution for identifying and examining phishing URLs through the power of machine learning. This report explores the development, execution, and assessment of this inventive system, designed to detect phishing URLs and present real-time insights via an accessible dashboard. By employing machine learning algorithms, the system can accurately determine whether URLs are safe or malicious, and the dashboard exhibits a comprehensive log of scanned URLs, including their domain, IP address, model prediction, and date of detection.

Moreover, the system seamlessly connects with the VirusTotal API to display vendor results, general info, and a snapshot of the analyzed webpage, giving users a well-rounded understanding of potential risks. The dashboard also features an interactive map that pinpoints the geographical location of the hosting IP address, granting security analysts better situational awareness.

This report delves into various project components, such as client needs, system design, implementation, and testing, as well as addresses ethical, legal, and social aspects linked to phishing detection and data privacy. Lastly, we discuss the pros and cons of our proposed solution and outline future work aimed at enhancing system performance and keeping up with the constantly changing world of cybersecurity.

Keywords: Phishing URL Detection, Machine Learning, Visualization Dashboard, VirusTotal API, IP Geolocation, Cybersecurity

Table of Contents

1. Introduction.....	3
1.1 Project Description.....	3
1.2 Current Scenario.....	5
1.3 Problem Domain and Project as a Solution	6
1.3.1 Problem Domain.....	6
1.3.2 Project as a Solution	6
1.4 Aims and Objectives.....	7
1.4.1 Aim.....	7
1.4.2 Objectives.....	7
1.5 Structure of the Report	8
1.5.1 Background.....	8
1.5.2 Development	8
1.5.3 Testing and Analysis	8
1.5.4 Conclusion	8
2. Background and Literature Review	10
2.1 About the End Users.....	10
2.1.1 Client.....	10
2.2 Understanding the Solution	11
2.2.1 Overview of the System	11
2.3 Similar Projects.....	12
2.3.1 Project 1: Fishing for Phishers	12
2.3.2 Project 2: Phishing websites Detection by using Machine Learning Techniques.....	13
2.3.3 Project 3: Phishing Sites Prediction using Machine Learning	13
2.4 Comparisons.....	14
2.4.1 Comparison Table	14
2.4.2 Analysis and Conclusion of the Comparison.....	14
3. Design and Development.....	16
3.1 Considered Methodologies.....	16
3.2 Selected Methodology	16
3.2.1 Scrum Methodology	16
3.2.2 Justification for using Scrum Methodology	18
3.3 Phases of Methodology	19

3.4	Survey Results.....	20
3.4.1	Pre-Survey Results	20
3.4.2	Post-Survey Results.....	21
3.5	Requirement Analysis.....	23
3.5.1	Data Collection	23
3.5.2	Data Processing through Machine Learning Models	23
3.5.3	Feature Extraction	23
3.5.4	Real-Time Detection.....	23
3.5.5	Accuracy and Performance.....	23
3.5.6	Dashboard Visualization	24
3.5.7	User Interface	24
3.5.8	Storage	24
3.5.9	Deployment.....	24
3.6	Hardware and Software Requirements.....	25
3.8	Design	26
3.8.1	Use Case Diagram	26
3.8.2	Flowchart	27
3.8.3	Data Flow Diagram.....	29
3.8.4	Wireframes	30
3.8.5	Gantt Chart	30
3.9	Implementation	33
3.9.1	Data Collection	33
3.9.2	Feature Extraction	35
3.9.3	Model Training.....	39
3.9.4	Model Testing	40
3.9.5	Dashboard Development	41
3.9.6	Integration.....	45
3.9.7	Deployment.....	46
4.	Testing and Analysis.....	48
4.1	Test Plan	48
4.1.1	Unit Testing, Test Plan	48
4.1.2	System Testing, Test Plan	50
4.2	Unit Testing	51
4.3	System Testing	93

4.4	Critical Analysis.....	99
4.4.1	Strengths	99
4.4.2	Weaknesses	100
5.	Conclusion.....	102
5.1	Legal, Social, and Ethical Issues	103
5.1.1	Legal Issues	103
5.1.2	Social Issues	103
5.1.3	Ethical Issues	104
5.2	Advantages	105
5.3	Limitations	106
5.4	Future Work	107
6.	References and Bibliography.....	108
7.	Appendix.....	111
7.1	Client Approval Letter	111
7.2	Understanding the Solution	112
7.2.1	Project Elaboration	112
7.3	Considered Methodologies.....	121
7.3.1	Waterfall Methodology	121
7.3.2	Prototype Methodology	122
7.3.3	Agile Methodology.....	122
7.4	Pre-Survey	123
7.4.1	Pre-Survey Form	123
7.4.2	Sample of Pre-Survey Form	129
7.4.3	Pre-Survey Result	134
7.5	Post-Survey.....	141
7.5.1	Post-Survey Form	141
7.5.2	Sample of Filled Post-Survey Forms	148
7.5.3	Post-Survey Result.....	154
7.6	Hardware and Software Requirements.....	161
7.6.1	Hardware	161
7.6.2	Software.....	161
7.7	Designs	162
7.7.1	Gantt Chart	162
7.7.2	Work Breakdown Structure	164

7.7.3	Wireframe	165
7.8	Milestones	167
7.9	Progress Review Table	168
7.10	Expected Outcomes and Deliverables.....	169
7.10.1	Project Expected Outcomes	169
7.10.2	Project Deliverables	169
7.11	Project Risk, Threats, and Contingency plans.....	170
7.11.1	Risk and Threats	170
7.11.2	Contingency Plans	170
7.12	Sample Codes	171
7.12.1	Back-End Code	171
7.12.2	Front-End Code.....	182
7.13	Plagiarism Test Result.....	185

List of Figures

Figure 1: Phishing statistics as of 2022 (Rosenthal, 2022).	5
Figure 2: Screenshot of the Fishing for Phishers project (Sylvia, 2021).	12
Figure 3: Screenshot of the Phishing Sites Prediction using a Machine Learning Project (Tiwari, 2020).	13
Figure 4: Cycle of Scrum Methodology (BYDREC, 2022).	17
Figure 5: Use Case diagram of the system.	26
Figure 6: Flowchart for the model prediction.	27
Figure 7: Backend events after URL search in frontend.	28
Figure 8: Block Diagram.....	29
Figure 9: Updated Gantt Chart.....	30
Figure 10: Folder Containing a list of URLs.....	33
Figure 11: Dataset containing legit URLs.	34
Figure 12: Dataset containing Phishing URLs.....	34
Figure 13: Feature extraction code for search bar-based features.	35
Figure 14: Feature extraction code for domain-based features.....	36
Figure 15: Extracting features to phishing URLs (a).	36
Figure 16: Extracting features to phishing URLs (b).	37
Figure 17: Extracting features to legitimate URLs (a).	37
Figure 18: Extracting features to legitimate URLs (b).	38
Figure 19: Splitting the data for model training and testing.....	39
Figure 20: Training the model in bulk.	39
Figure 21: Testing the model in bulk.....	40
Figure 22: Dashboard Development Code (a).	41
Figure 23: Dashboard Development Code (b).	42
Figure 24: Dashboard Development Code (c).....	43
Figure 25: System home page.....	44
Figure 26: System dashboard page.....	44
Figure 27: Using fetch API for integration.....	45
Figure 28: Using flask to create a web server.	46
Figure 29: Extracting features to phishing URLs (a).	52

Figure 30: Extracting features to phishing URLs (b).	52
Figure 31: Feature extracted dataset for phishing URLs.	53
Figure 32: Extracting features to legitimate URLs (a).	54
Figure 33: Extracting features to legitimate URLs (b).	54
Figure 34: Feature extracted dataset for legitimate URLs.	55
Figure 35: Combining both CSV files.....	57
Figure 36: Saving the combined CSV file to "combined_data.csv".	57
Figure 37: Combined dataset CSV file.	58
Figure 38: Screenshot of bulk training.	59
Figure 39: Screenshot of bulk testing.	60
Figure 40: Code for saving the model into pickle file.	62
Figure 41: Screenshot of the classifier_model.pkl file.	63
Figure 42: Providing a phishing URL to scan.	65
Figure 43: System displaying the URL as phishing.....	65
Figure 44: Providing a legitimate URL to scan.	67
Figure 45: System displaying the URL as legitimate.....	67
Figure 46: A pop-up showing the URL doesn't exists.	68
Figure 47: Passing a URL to check the response of the virus total API.	70
Figure 48: Response from the API.....	70
Figure 49: Screenshot of the code converting the URL into IP address.	71
Figure 50: Providing an URL to scan for longitude and latitude.	73
Figure 51: Screenshot of longitude and latitude as response.....	73
Figure 52: Screenshot of Map in the dashboard.	74
Figure 53: Screenshot of the code for generating a screenshot.....	76
Figure 54: Screenshot of the webpage.....	76
Figure 55: Screenshot of the webpage getting displayed on the front end.	78
Figure 56: Screenshot of Domian being displayed on the front end.....	79
Figure 57: Screenshot of vendor results on the frontend.....	81
Figure 58: Screenshot of the counter displaying the number of vendors detecting the URL as phishing.	82
Figure 59: Screenshot of the detection date label displaying the correct date.....	83

Figure 60: Screenshot of reloading the page.	85
Figure 61: Screenshot of the page being reloaded.	85
Figure 62: Screenshot of the stored logs after reloading the page.....	86
Figure 63: Screenshot of the system running on a different device inside the same network (a).....	88
Figure 64: Screenshot of the system running on a different device inside the same network (b).....	88
Figure 65: shot while checking the functionality of the info button (a).	90
Figure 66: shot while checking the functionality of the info button (b).	90
Figure 67: Screenshot of the pop-up window inside the details button.	92
Figure 68: Providing a phishing URL to scan.	94
Figure 69: Dashboard displaying the details on the scan of the phishing URL.....	94
Figure 70: Additional details on the scan of phishing URL.	95
Figure 71: Providing a legitimate URL to scan.	97
Figure 72: Dashboard displaying the details on the scan of the legitimate URL.....	97
Figure 73: Additional details on the scan of legitimate URL.	98
Figure 74: Client Letter.....	111
Figure 75: Waterfall Methodology (Lucid Content, 2021).	121
Figure 76: Prototype Methodology (Martin, 2022).....	122
Figure 77: Agile Methodology (Martin, 2022).	122
Figure 78: Questions in the pre-survey form (a).....	123
Figure 79: Questions in the pre-survey form (b).....	124
Figure 80: Questions in the pre-survey form (c).....	125
Figure 81: Questions in the pre-survey form (d).....	126
Figure 82: Questions in the pre-survey form (e).....	127
Figure 83: Questions in the pre-survey form (f).....	128
Figure 84: Pre-Survey Result from an individual (a).	129
Figure 85: Pre-Survey Result from an individual (b).	130
Figure 86: Pre-Survey Result from an individual (c).	130
Figure 87: Pre-Survey Result from an individual (d).	131
Figure 88: Pre-Survey Result from an individual (e).	132

Figure 89: Pre-Survey Result from an individual (f).	132
Figure 90: Pre-Survey Result from an individual (g).	133
Figure 91: Pre-Survey Result from an individual (h).	133
Figure 92: Profession of individuals who filled out the pre-survey form.	134
Figure 93: Current college year of the student who filled out the form.	134
Figure 94: Faculty of the student who filled out the form.	135
Figure 95: Pre-Survey Question no.1	135
Figure 96: Pre-Survey Question no.2	136
Figure 97: Pre-Survey Question no.3	136
Figure 98: Pre-Survey Question no.4	137
Figure 99: Pre-Survey Question no.5	137
Figure 100: Pre-Survey Question no.6	138
Figure 101: Pre-Survey Question no.7	138
Figure 102: Pre-Survey Question no.8	139
Figure 103: Pre-Survey Question no.9	139
Figure 104: Pre-Survey Question no.10	140
Figure 105: Questions in the post-survey form (a).	141
Figure 106: Questions in the post-survey form (b).	142
Figure 107: Questions in the post-survey form (c).	143
Figure 108: Questions in the post-survey form (d).	144
Figure 109: Questions in the post-survey form (e).	145
Figure 110: Questions in the post-survey form (f).	146
Figure 111: Questions in the post-survey form (g).	147
Figure 112: Post-Survey Result from an individual (a).	148
Figure 113: Post-Survey Result from an individual (b).	149
Figure 114: Post-Survey Result from an individual (c).	150
Figure 115: Post-Survey Result from an individual (d).	150
Figure 116: Post-Survey Result from an individual (e).	151
Figure 117: Post-Survey Result from an individual (f).	151
Figure 118: Post-Survey Result from an individual (g).	152
Figure 119: Post-Survey Result from an individual (h).	152

Figure 120: Post-Survey Result from an individual (i).....	153
Figure 121: Profession of individuals who filled out the post-survey form.....	154
Figure 122: Post-Survey Question no.1.....	154
Figure 123: Post-Survey Question no.2.....	155
Figure 124: Post-Survey Question no.3.....	155
Figure 125: Post-Survey Question no.4.....	156
Figure 126: Post-Survey Question no.5.....	156
Figure 127: Post-Survey Question no.6.....	157
Figure 128: Post-Survey Question no.7.....	157
Figure 129: Post-Survey Question no.8.....	158
Figure 130: Post-Survey Question no.9.....	158
Figure 131: Post-Survey Question no.10.....	159
Figure 132: Post-Survey Question no.11.....	159
Figure 133: Post-Survey Question no.12.....	160
Figure 134: Post-Survey Question no.13.....	160
Figure 135: Old Gantt Chart.....	162
Figure 136: Work breakdown structure.....	164
Figure 137: Wireframe for the homepage of the system.....	165
Figure 138: Wireframe for the Dashboard page of the system.....	166
Figure 139: Wireframe for the Details page of the system.....	166
Figure 140: Feature extraction.ipynb code (a).	171
Figure 141: Feature extraction.ipynb code (b).	171
Figure 142: Feature extraction.ipynb code (c).....	172
Figure 143: Feature extraction.ipynb code (d).	172
Figure 144: Feature extraction.ipynb code (e).	173
Figure 145: Feature extraction.ipynb code (f).....	173
Figure 146: modeltrain.ipynb code (a).....	174
Figure 147: modeltrain.ipynb code (b).....	174
Figure 148: modeltrain.ipynb code (c).	175
Figure 149: modeltrain.ipynb code (d).....	175
Figure 150: modeltrain.ipynb code (e).....	176

Figure 151: main.py code (a).....	177
Figure 152: main.py code (b).....	178
Figure 153: main.py code (c).....	179
Figure 154: newlocation.py code	180
Figure 155: virustotalapi.py code.....	181
Figure 156: index.tsx code.....	182
Figure 157: dashboard.tsx code (a).....	183
Figure 158: dashboard.tsx (b).....	184
Figure 159: Originality Report (a).....	185
Figure 160: Originality Report (b).....	186
Figure 161: Originality Report (c).....	187
Figure 162: Originality Report (d).....	188
Figure 163: Originality Report (e).....	189

List of Tables

Table 1: Similar project comparison table.	14
Table 2: Tabular form of the dates in the Updated Gantt Chart.....	31
Table 3: Unit Testing, Test Plan.....	48
Table 4: System Testing, Test Plan.	50
Table 5: Test Case 1.	51
Table 6: Test Case 2.	56
Table 7: Test Case 3.	59
Table 8: Test Case 4.	60
Table 9: Test Case 5.	61
Table 10: Test Case 6.	64
Table 11: Test Case 7.	66
Table 12: Test Case 8.	68
Table 13: Test Case 9.	69
Table 14: Test Case 10.	71
Table 15: Test Case 11.	72
Table 16: Test Case 12.	74
Table 17: Test Case 13.	75
Table 18: Test Case 14.	77
Table 19: Test Case 15.	79
Table 20: Test Case 16.	80
Table 21: Test Case 17.	82
Table 22: Test Case 18.	83
Table 23: Test Case 19.	84
Table 24: Test Case 20.	87
Table 25: Test Case 21.	89
Table 26: Test Case 22.	91
Table 27: System Test Case 1.	93
Table 28: System Test Case 2.	96
Table 29: Tabular form of the dates in the old Gantt Chart.....	163
Table 30: Progress Table of the Project.	168

List of Abbreviation

PDML	Phishing Detection Using Machine Learning
ML	Machine Learning
URL	Uniform Resource Locator
API	Application Programming Interface
DNS	Domain Name System
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
ipynb	IPython Notebook
UI/UX	User Interface and User Experience
CLI	Command Line Interface
JS	JavaScript
JSON	JavaScript Object Notation

Phishing Detection Using Machine Learning (PDML)

CHAPTER 1: Introduction

1. Introduction

1.1 Project Description

Phishing is a type of social engineering cyberattack in which attackers obtain sensitive user information using malicious URLs, spam text messages, and voice communications. As the Internet has become a crucial part of people's lives, an increasing number of individuals are benefiting from the convenience it provides. On the other hand, there are several cyber-attacks emanating from the dark side of the Internet, including phishing. Based on some human flaws, hackers have created perplexing phishing websites to steal critical user information via harmful links, spam text, and voice communications.

To counter such attacks, methods such as URL black listing and URL detection utilizing ML have been implemented using data sets accessible on famous repositories such as Kaggle, UCI, and others. To overcome these drawbacks, a new approach, phishing detection using an ML technique integrated with a URL comparison technique, is introduced in this paper, where an ML-based system is created, that can instantly identify phishing URLs and stop people from accessing them. While making this project a test URL is passed through different features and fed to a ML model to detect whether the given test URL is a phishing URL or a legitimate URL.

The main goal of this project is to create an ML model that can effectively and accurately identify phishing URLs. We will first compile a sizable dataset of authentic URLs and phishing URLs in order to accomplish this purpose. From these URLs, we will extract numerous information such as the domain name, path, and query parameters. Additionally, we will examine the URLs using text-based criteria like the occurrence of dubious terms and URL length.

The data will next be cleaned up and transformed to make it ready for ML techniques. In order to determine the most crucial attributes for phishing URL detection, we will also experiment with other feature selection strategies. The dataset will then be used to train a variety of ML models, such as neural networks, decision trees, random

forests, and logistic regression. The effectiveness of each model will be assessed using metrics including test and train accuracy. In order to evaluate which model is the most successful at identifying phishing URLs, we will also compare the models' performance.

Last but not least, we'll create a web application that lets users enter a URL and determine whether or not it's a phishing URL which will also include various features which will be included in the dashboard of the system for visualization. The program will make the prediction using the ML model created in this project.

This project's overall goals are to show how ML can be used to identify phishing Websites and to offer a workable solution for enhancing internet security. A ML model that can be incorporated into current security systems to recognize and stop phishing assaults will be the project's output.

1.2 Current Scenario

With significant technological advancements in fields such as cloud computing, AI, and the internet, there is a significant shift from traditional information and data trading to e-trading. URL phishing is another type of cyber-attack that involves creating and distributing fraudulent websites that appear to be legitimate in order to trick people into disclosing personal information such as passwords, credit card numbers, and so on. According to reports, the monetary loss due to URL phishing was nearly 58 million USD in 2019, which increased even more in 2020 due to COVID-19, and the figures are still rising (Rosenthal, 2022). Although researchers have proposed various phishing detection techniques such as URL blacklisting and heuristic methods, advancements in Artificial Intelligence have proven to be a better filter of URLs to classify them as spam or legitimate.



Figure 1: Phishing statistics as of 2022 (Rosenthal, 2022).

1.3 Problem Domain and Project as a Solution

1.3.1 Problem Domain

In recent years, there has been an exponential increase in the number of phishing attacks worldwide, and many large corporations have been victims, suffering significant losses. According to an FBI report, phishing was the most common sort of cybercrime in 2020, with the number of phishing occurrences nearly doubling from 114,702 in 2019 to 241,324 in 2020 (Platten, 2021).

Such phishing attack causes business and individuals loss in various aspects such as:

- Loss of Valuable Data.
- Damaged Reputation.
- Direct Monetary Loss.
- Loss of Productivity.
- Loss of Customers.
- Financial Penalties.
- Intellectual Property Theft.
- Loss of Company Value.

1.3.2 Project as a Solution

This project will help to overcome the above-listed problems and will minimize the kinds of phishing attacks that occur. With the help of this system, which detects hundreds of thousands of phishing URLs with the help of ML, which can filter those URLs and provide accurate output to the users, can help individuals minimize the risk of falling into such kinds of attacks and becoming a victim of them.

1.4 Aims and Objectives

1.4.1 Aim

The main aim of this project is to mitigate the problem currently faced by internet users globally by providing them with a tool that can identify suspicious attacks and legitimate trusted URLs using ML and help individuals minimize the risk of being victims of such malware attacks.

1.4.2 Objectives

A set of measurable objectives has been set to accomplish the aim of the project:

- To carry out research, emphasize detailed understanding and learn ML and its applications.
- To study and analyze similar projects and systems.
- To have a comprehensive understanding of programming concepts by using Python Programming language and React Framework.
- To use various libraries and modules in Python to develop a prototype system to automate the network.
- To build a system using ML which detects Phishing URLs.
- To collect datasheets containing phishing and legitimate URL from the open-source platform.
- To study trends in URL that shows signs of Phishing.
- To deploy the system in a certain network.
- To determine the application areas of the project.
- To validate the output of the ML model by using it in a real-world application.
- To gather and study the client's requirements through surveys and feedback.
- To evaluate the project's application areas by testing and implementing the system in multiple scenarios.

1.5 Structure of the Report

1.5.1 Background

This chapter looks into the background material required to comprehend the project. It begins with a consideration of the end users' wants and requirements. It then goes into understanding the solution and how it fulfills those needs. The chapter also contains a review of related projects' literature, highlighting their important results and the influence they have on the current project along with a comparison table and comparative analysis are offered.

1.5.2 Development

This chapter describes the project's complete development process. It begins with a presentation of the project development approaches under consideration, followed by a description of the chosen methodology and its phases. The project's hardware and software needs are discussed in the requirement analysis portion, which is followed by the design phase, which includes use case, flowcharts, block diagram, and use case diagrams. The implementation part detail describes the development of the systems from the beginning to the end.

1.5.3 Testing and Analysis

This chapter focuses on the numerous testing methods used to evaluate the performance and functioning of the system. The test plan section explains the methodologies for both unit testing and system testing, followed by test execution and results display. There is also a critical analysis to analyze the project's merits, shortcomings, and opportunities for development.

1.5.4 Conclusion

This chapter reviews the project's outcomes, commenting on its goals and accomplishments. It examines the project's legal, social, and ethical difficulties, outlining potential hurdles and implementation considerations. The project's benefits and drawbacks are also discussed, providing a fair picture of its entire influence. Finally, this chapter suggests areas for further investigation and prospective system upgrades.

CHAPTER 2: Background

2. Background and Literature Review

2.1 About the End Users

2.1.1 Client

- **Name of the Client:** Compro International Pvt. Ltd.
- **Description of the Client:**

Compro International Pvt. Ltd. Is made up of a group of experts with exceptional technical and managerial skills who are dedicated to offering their clients unparalleled levels of service. It designs websites, offers scale-out NAS systems, and provides interactive solutions and data center infrastructure. It has been providing computing solutions in hardware and software to a wide range of financial institutions, travel agencies, commercial buildings, and non-governmental groups inside the country in addition to addressing personal computer demands.

- **Client Approval Letter:**

The client Approval Letter is placed in the appendix section of the report, [Click Here](#) to view it.

2.2 Understanding the Solution

2.2.1 Overview of the System

This PDML project is an innovative approach that combines ML with a user-friendly dashboard to detect phishing URLs and improve online safety. Data preparation, feature extraction, model training, and web interface integration are all part of the system. After cleaning and converting the raw data, the model is trained to spot phishing trends on a huge dataset of labeled URLs.

Through a simple frontend interface, users may submit URLs for scanning, and the backend server scans the URL, applies the ML model, and delivers the results. The dashboard displays these findings along with additional information such as domain, IP address, scanned date, antivirus vendor results, a screenshot of the webpage, and even a map with many more additional features. The project provides an excellent solution for spotting phishing URL-related dangers by combining the power of ML with an easy-to-use web interface, thereby increasing user security and online safety.

The rest of the contents of the Understanding the Solution are placed in the appendix section of the report, [Click Here](#) to view it.

2.3 Similar Projects

2.3.1 Project 1: Fishing for Phishers

With the addition of feature extraction, this project "Fishing for Phishers" is an ML-based phishing URL detection system created by a university student named "Kati Sylvia" that gives its user a field to input URLs and only displays whether it is a phishing or a legitimate URL. She trained her system for this project using two different datasets for both phishing and legitimate URLs using various algorithms, with the random forest classifier providing the highest testing score of around 94.5% among all.

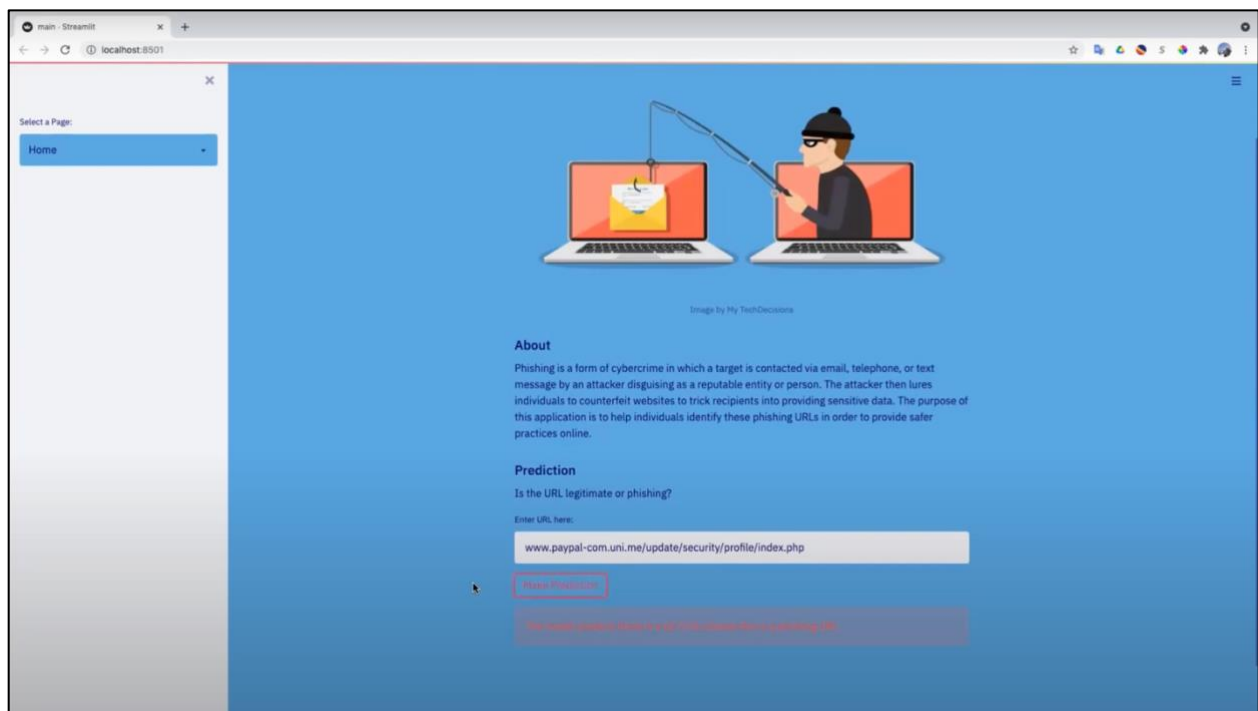


Figure 2: Screenshot of the Fishing for Phishers project (Sylvia, 2021).

2.3.2 Project 2: Phishing websites Detection by using Machine Learning Techniques

During a technical seminar at "The Maharaja Institute of Technology Mysore," he also developed a system that allows users to enter a URL and detects both phishing and legitimate URLs. This system was built with ensemble learning techniques like bagging and boosting, which help to improve ML results by combining multiple models for model deployment. XGBoost, the project's algorithm, has a 97.5% accuracy rate.

2.3.3 Project 3: Phishing Sites Prediction using Machine Learning

This project is also a phishing site prediction system made using ML developed by a student named "Tarun Tiwari" of "The Vellore Institute of Technology". The user can enter the desired URL into this system, and it will display whether it is phishing or legitimate. The proposed approach of this system is that it first collects raw data from a phish tank and divides it into train and test sets before starting the model build. Following these steps, the model is trained and the links are imported to prediction before being deployed in FastAPI.

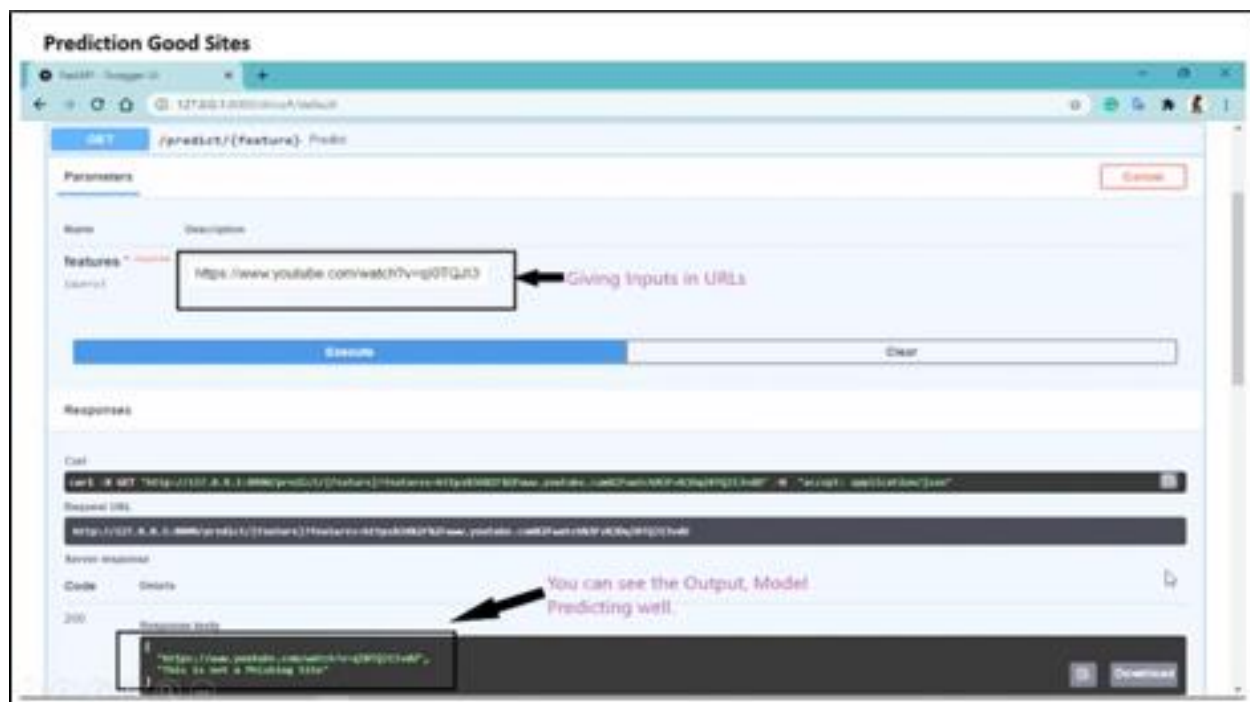


Figure 3: Screenshot of the Phishing Sites Prediction using a Machine Learning Project (Tiwari, 2020).

2.4 Comparisons

2.4.1 Comparison Table

Table 1: Similar project comparison table.

S.N.	Features	Project 1	Project 2	Project 3	This Project
1.	Programming Language	Python	Python	Python	Python, Next.js
2.	Store Scan Result	X	X	X	✓
3.	CLI	X	X	✓	X
4.	GUI	✓	✓	✓	✓
5.	Dashboard	X	X	X	✓
6.	Performs Reconnaissance	✓	✓	✓	✓
7.	Generate Report	X	X	X	X
8.	Feature Extraction	✓	✓	X	✓
9.	Generate Screenshot of webpage	X	X	X	✓
10.	Generate Map	X	X	X	✓

2.4.2 Analysis and Conclusion of the Comparison

Projects 1 and 2 have had a significant impact on my project, owing to the feature extraction component that can categorize spam URLs, which Project 3 lacked. In terms of algorithm, the random forest classifier provides the highest accuracy among others, and it is also used in projects 1 and 3 to train the model using ML techniques. In an overall comparison, my project stands out among these three projects because it has an interactive dashboard for the user where they can view the recently scanned URLs which also demonstrates the information on those URLs which is achieved through the integration of the APIs, it also generates a screenshot of the webpage and provides a map, and for the deployment part, the system is hosted on a specific IP address where any user can use this system inside a network. These are the distinguishing features that none of the other three projects have, making my project stand out from the crowd.

CHAPTER 3: Development

3. Design and Development

3.1 Considered Methodologies

The contents of the Considered Methodologies are placed in the appendix section of the report, [Click Here](#) to view it.

3.2 Selected Methodology

3.2.1 Scrum Methodology

The methodology which will be used for the development of this system is scrum methodology. It is an iterative, incremental approach to project management and product development (like websites and software). It is a non-linear and adaptable methodology that allows for scope adjustments mid-project with the purpose of continual improvement during those changes (Martin, 2022).

Since the scrum methodology's primary goal is to satisfy the client's objectives, it also embraces any changing requirements even in the late development cycle with continuous improvement, timely deliverables, and strong project visibility as well. This methodology consists of different sprint cycles where each cycle consists of phases like planning, development of the system, testing out the developed system, and reviewing it with the client and supervisors.

The project will be completed using this methodology over the course of several sprint cycles, with the first one consisting of creating wireframes for the system, the next being the development of the phishing detection, the following being the development of the dataset and feature extraction, the following being the development of the project's specific ML, and the last being the combination of all the modules created.

This methodology would be best suited for this project scenario because the first step is to develop a phishing detection system using ML, then the system is trained and tested using a datasheet containing phishing and legitimate URLs from the open source platform, and finally, it is reviewed. If any additional features are required on this project in the future, the next cycle of the agile methodology is carried out using the same methods.

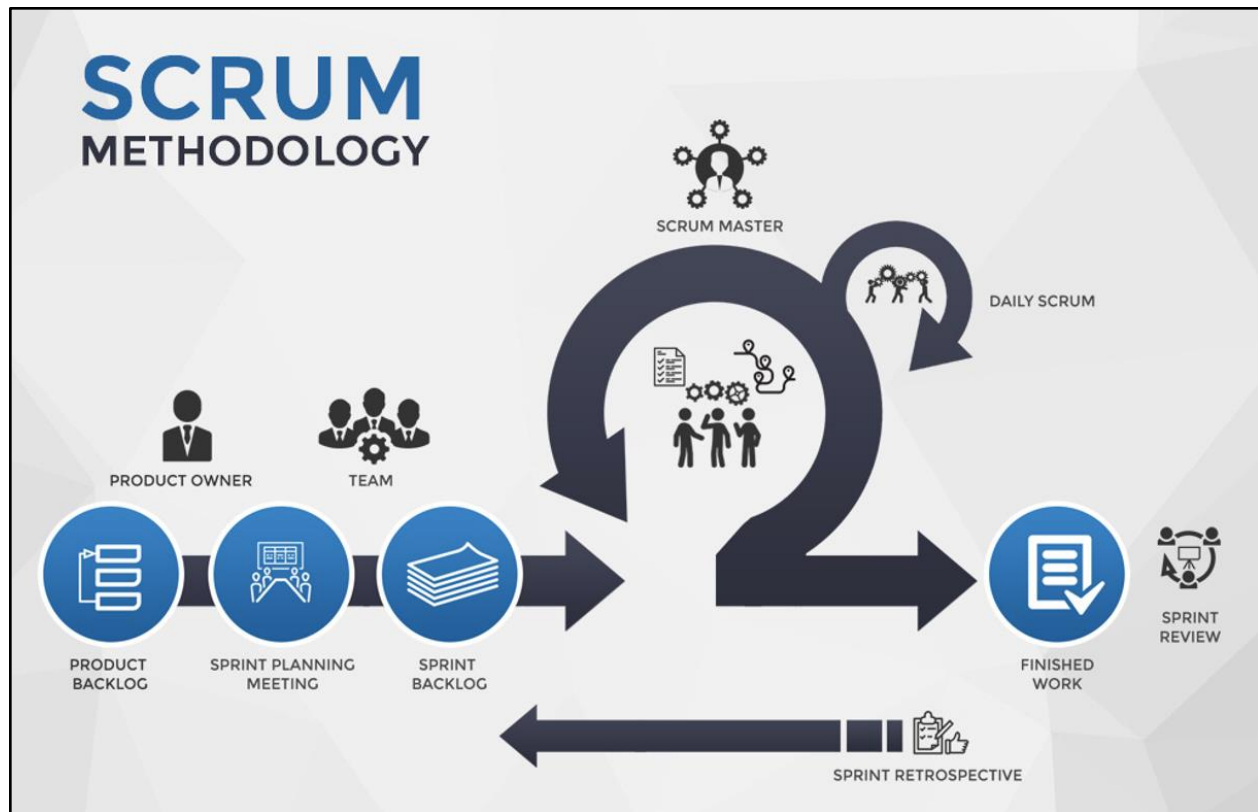


Figure 4: Cycle of Scrum Methodology (BYDREC, 2022).

3.2.2 Justification for using Scrum Methodology

The Scrum Methodology was chosen over the other methodologies considered. The following are the major benefits of Scrum Methodology that should be considered when choosing a methodology:

Improved flexibility and adaptability: Scrum enables teams to be more adaptable and flexible in the face of changing requirements and priorities. It enables teams to respond quickly to changing market conditions by allowing for continuous feedback and frequent iterations.

Improved quality and customer satisfaction: Scrum places an emphasis on delivering working software in short iterations, allowing for early feedback from customers and stakeholders. This feedback ensures that the final product meets the expectations and requirements of the customer, resulting in increased customer satisfaction.

Better risk management: Regular check-ins and reviews in Scrum allow teams to identify potential risks and issues early on. This assists teams in mitigating risks before they become major issues, as well as responding quickly to unexpected challenges.

Increased productivity and efficiency: Scrum allows teams to concentrate on providing value to customers through small, incremental improvements. As teams are constantly refining their work processes and improving their output, this approach can lead to increased productivity and efficiency.

Increased collaboration and teamwork: Scrum encourages teamwork and collaboration among team members, stakeholders, and customers. It encourages transparency, communication, and a shared sense of project ownership.

Overall, Scrum provides a structured and disciplined approach to software development that can help teams to deliver high-quality products on time and within budget.

3.3 Phases of Methodology

This project will carry out following the four different phases of this scrum methodology where the elaboration are as follows:

Step 1: Planning

During this phase, extensive research on the project topic will be conducted to determine how to carry out the project from the initiation, development, and deployment phases, including the necessary hardware and software requirements.

Step 2: Designing

During this phase, the proposed system's wireframes will be created using software such as Balsamiq wireframe for the system's front end.

Step 3: Development

During this phase, the project will be developed, including the development of the front end and back end of the system, as well as combining them and creating a working phishing detection tool.

Step 4: Testing

After development is completed, the system's functionality, performance, and testing are carried out using various testing methodologies such as white box testing, black box testing, and grey box testing, and those tests are included in the project documentation.

3.4 Survey Results

3.4.1 Pre-Survey Results

The pre-survey was conducted amongst various individuals from IT and computer engineering backgrounds for the pre-requirement collection procedure.

The findings from the survey are listed below:

- The survey results provide insights into participants' knowledge and experiences related to cyber security and phishing attacks.
- The majority of participants were IT students, with smaller percentages from security and computer engineering backgrounds.
- 95.5% of participants were familiar with the cyber security domain, and 86.4% knew what a phishing attack is and how it works.
- 40.9% of participants had already been a victim of a phishing attack.
- 68.2% of participants had received malicious emails that redirected them to another website.
- A significant proportion of participants had used threat intelligence platforms in the past, with Virus Total being the most commonly used platform.
- 63.3% of participants were aware of ML and how it works, but 36.4% were unaware.
- 72.7% of participants were aware that phishing URLs can be detected using ML algorithms.
- 68.2% of participants believed that implementing an ML-based system could help individuals avoid falling victim to phishing attacks.
- 95.5% of participants agreed that an organization should implement a system to protect them from phishing attacks.

Overall, the survey results suggest that there is a need for effective countermeasures against phishing attacks, and an ML-based system could be a valuable solution.

[Click Here](#) to view the Pre-Survey results.

3.4.2 Post-Survey Results

After the project methodology and development were completed, a post-survey was conducted to collect feedback on the system. This phase is critical because it allows for the assessment of user satisfaction, the identification of areas for improvement, and the gathering of insights for future system development. The project team can improve the system's functionality, usability, and performance by analyzing feedback and reviews. The post-survey is an important part of the project lifecycle because it ensures that the system meets the needs of the users and provides a high level of satisfaction.

The findings from the survey are listed below:

- 82.4% of survey participants were students, and 17.6% were security analysts or DevOps engineer.
- 94.1% of respondents understood what a phishing attack is and how it works.
- All participants found the system user-friendly and easy to navigate.
- System performance ratings: 35.3% scored it a 10, 35.3% scored it a 9, 23.5% scored it an 8, and 5.9% scored it a 5.
- All participants agreed the dashboard is useful for analyzing and visualizing both phishing and legitimate URLs.
- ML model accuracy ratings: 17.6% scored it a 10, 35.3% scored it a 9, 41.2% scored it an 8, and 5.9% scored it a 5.
- 88.2% of participants believed the system reduces successful phishing attacks.
- 94.1% of participants would recommend the system to others.
- 70.6% of participants found false positives or negatives in the PDML system.
- 76.5% of participants were satisfied with the system's features and functionalities.
- All participants agreed the system effectively identifies and blocks phishing attempts in real time.
- 70.6% of participants encountered bugs or issues while using the system.
- 64.7% of participants believed the system doesn't need significant changes or modifications.
- 94.1% of participants agreed to use the system as a long-term solution for detecting and preventing phishing URLs.

- The majority of the individuals requested to develop a browser extension for real-time phishing website detection.
- Individuals even demanded for enhancing the system with malware detection capabilities and also to include files, hashes, and folders.
- Mainly individuals also requested to improve the model's accuracy.

According to the feedback, most people found the system to be extremely simple to use and successful in detecting phishing attempts. The dashboard was useful for checking out both safe and dangerous URLs. Although our ML model received average grades overall, there is still space for improvement.

People offered several great suggestions for improving it, such as building a browser extension, adding a login system, and even identifying malware. They also proposed making it more efficient, increasing its scanning capabilities, and improving the user interface. While the majority of users would suggest our system and consider it a long-term solution, others encountered faults and concerns that we need to address.

[Click Here](#) to view the Post-Survey results.

3.5 Requirement Analysis

All requirements analysis is done based on research and inquiries with IT professionals, and features for this system are developed accordingly. The following requirements and features are elaborated on and specified:

3.5.1 Data Collection

To train ML models for phishing detection, the system should be able to collect a large and diverse dataset of URLs which is stored in a text file including both legitimate and phishing URLs in the dataset.

3.5.2 Data Processing through Machine Learning Models

For detecting phishing URLs, the system should employ appropriate ML algorithms such as decision trees, random forests, MLP classifiers, SVC, Logistic regression, Gradient boost classifier, and Ada boost Classifier as well.

3.5.3 Feature Extraction

The system should be able to extract and feed features from URLs into ML models. These features may include both search bar features and domain-based features such as domain names, IP addresses, character count, domain length, DNS record, web traffic, domain age, and other metadata.

3.5.4 Real-Time Detection

The system should be able to detect phishing URLs in real time and warn users of potential threats. It must also be capable of detecting new phishing patterns and adapting to new attack vectors.

3.5.5 Accuracy and Performance

In order to detect phishing URLs, the system must be accurate and efficient. It should have a low false-positive rate and be capable of handling large volumes of URLs with minimal latency.

3.5.6 Dashboard Visualization

Determine your visualization requirements and design a dashboard for end users. The dashboard, for example, could display the scanned URL's result as well as various additional URL details that may be useful to its end users.

3.5.7 User Interface

The system should include an easy-to-use dashboard that allows users to visualize detection results and monitor system performance. The dashboard should be simple to use and include interactive visualizations.

3.5.8 Storage

Browser local storage is a useful way for keeping scan logs in this project. This built-in function keeps data on the client side, which creates a customized scan history. Each log entry is saved as a JSON object and new scans will be added to local storage. The dashboard parses JSON objects to show logs, giving users instant access to their scanning history. Using local storage improves the user experience, decreases the amount of storage required on the server, and handles privacy issues.

3.5.9 Deployment

Determine the requirements for deployment, such as the platform and infrastructure needed to host the ML models and dashboard. For example, the project could use cloud-based platforms such as Amazon Web Services (AWS), and Microsoft Azure, or even host locally on a network.

3.6 Hardware and Software Requirements

The contents of the Hardware and Software Requirements are placed in the appendix section of the report, [Click Here](#) to view it.

3.8 Design

3.8.1 Use Case Diagram

A use case diagram is used to describe a system's dynamic behavior. It contains the functionality of the system by incorporating use cases, actors, and their interactions. It stimulates the jobs, services, and functionalities required by an application's system/subsystem. It represents a system's high-level functionality as well as how the user interacts with the system. (javatpoint, 2022)

Phishing detection using machine learning

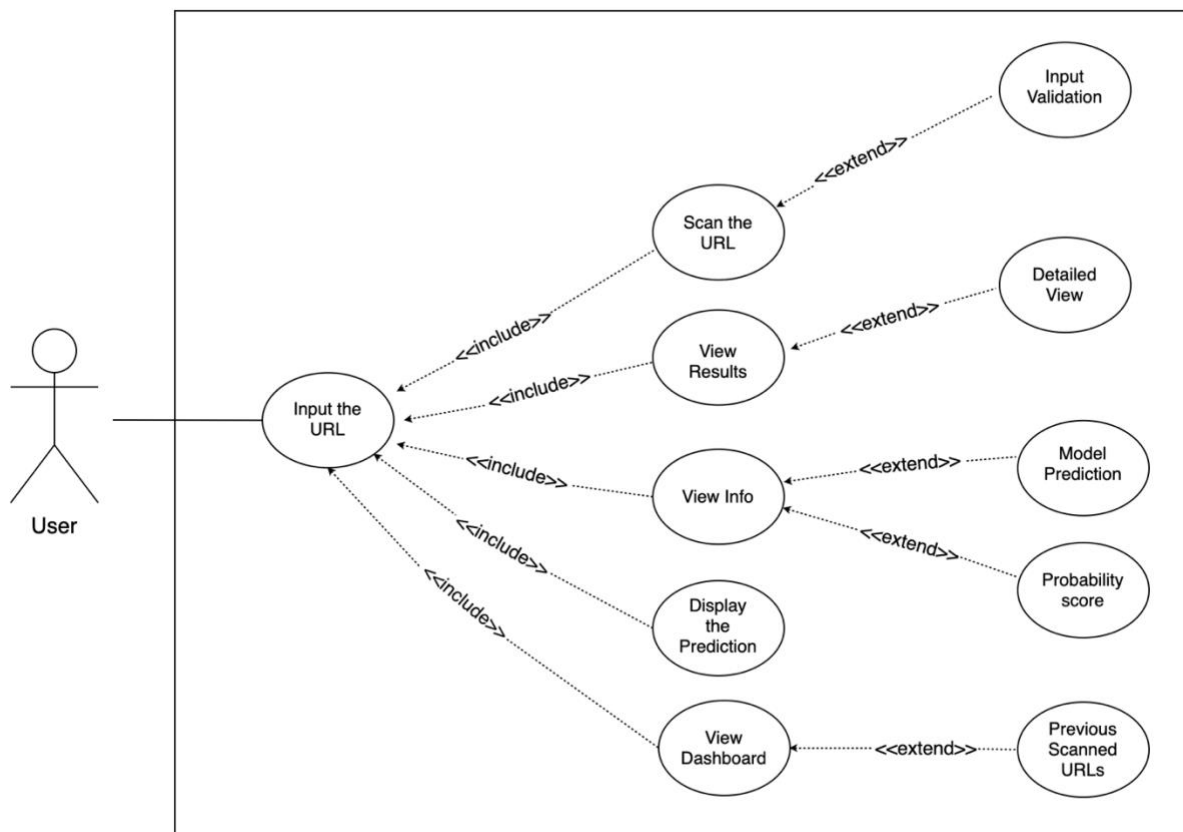


Figure 5: Use Case diagram of the system.

3.8.2 Flowchart

A flowchart is nothing more than a graphical depiction of steps. It displays steps in sequential order and is commonly used to depict the flow of algorithms, workflow, or processes. A flowchart typically depicts the processes as various types of boxes and their sequence by connecting them with arrows. (Visual Paradigm, 2023)

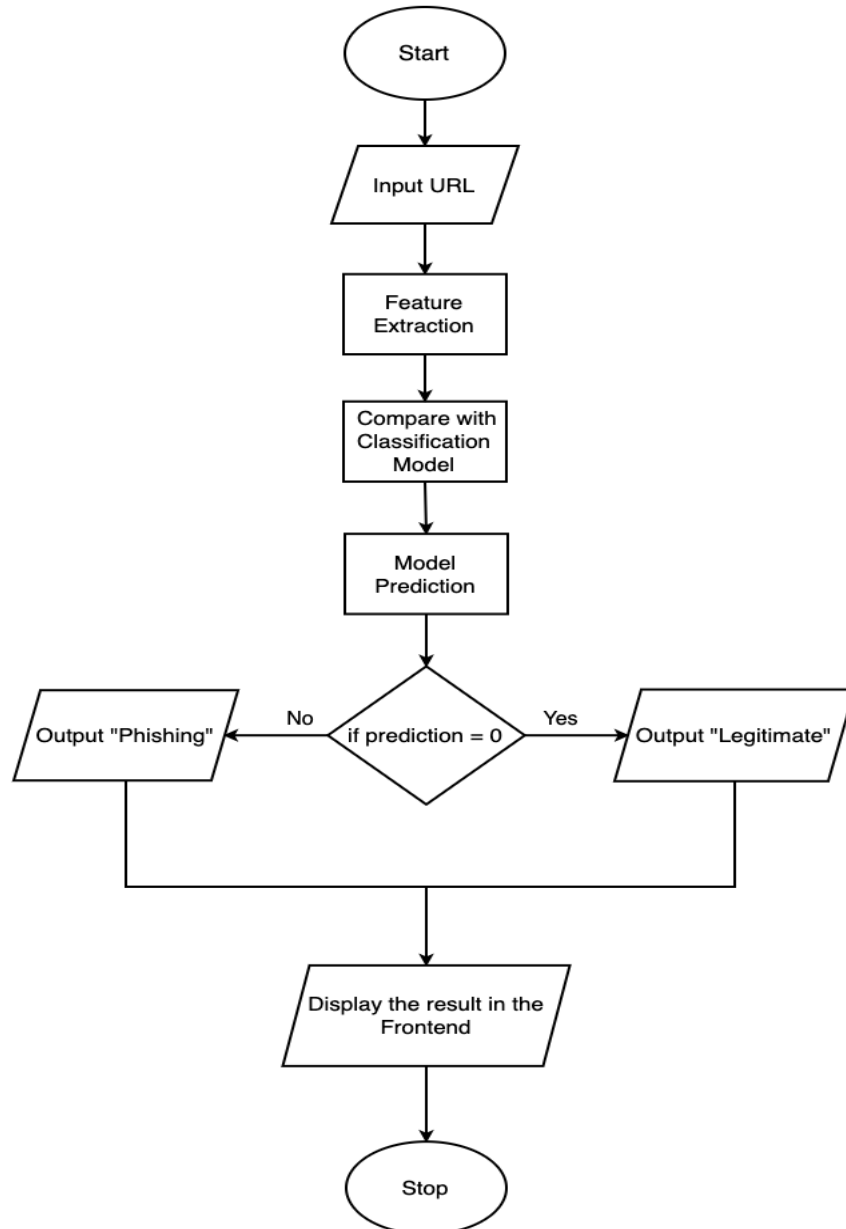


Figure 6: Flowchart for the model prediction.

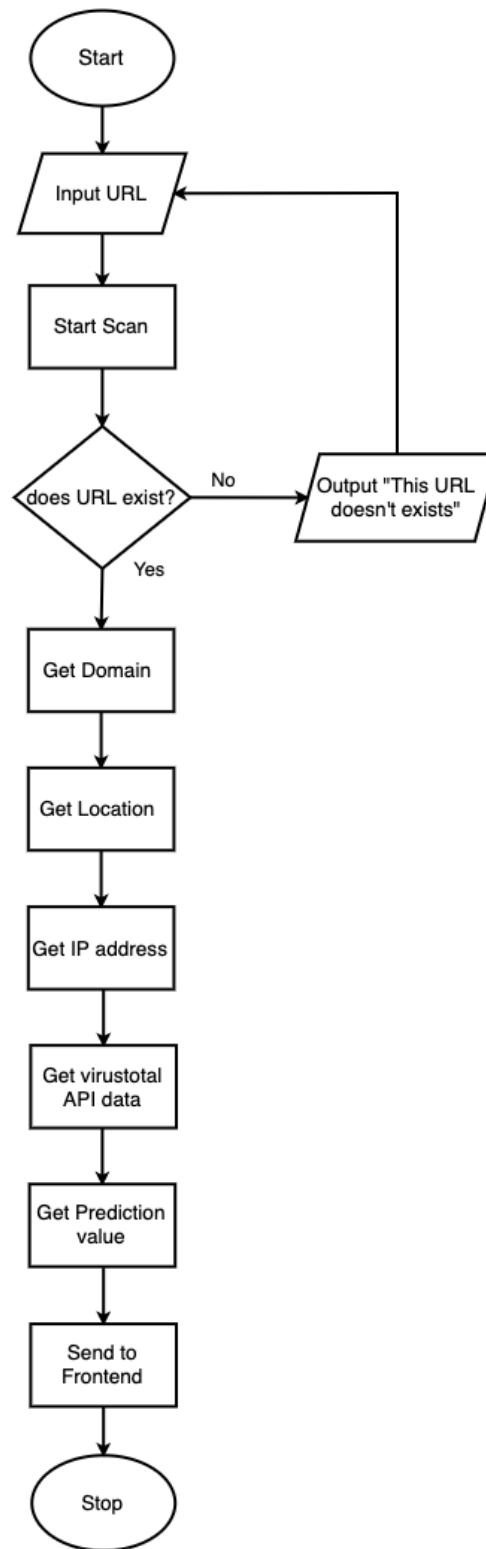


Figure 7: Backend events after URL search in frontend.

3.8.3 Data Flow Diagram

A data flow diagram (DFD) depicts the information flow for any process or system. It shows data inputs, outputs, storage sites, and the pathways between each destination using predetermined symbols such as rectangles, circles, and arrows, as well as brief text labels. (LucidChart, 2022)

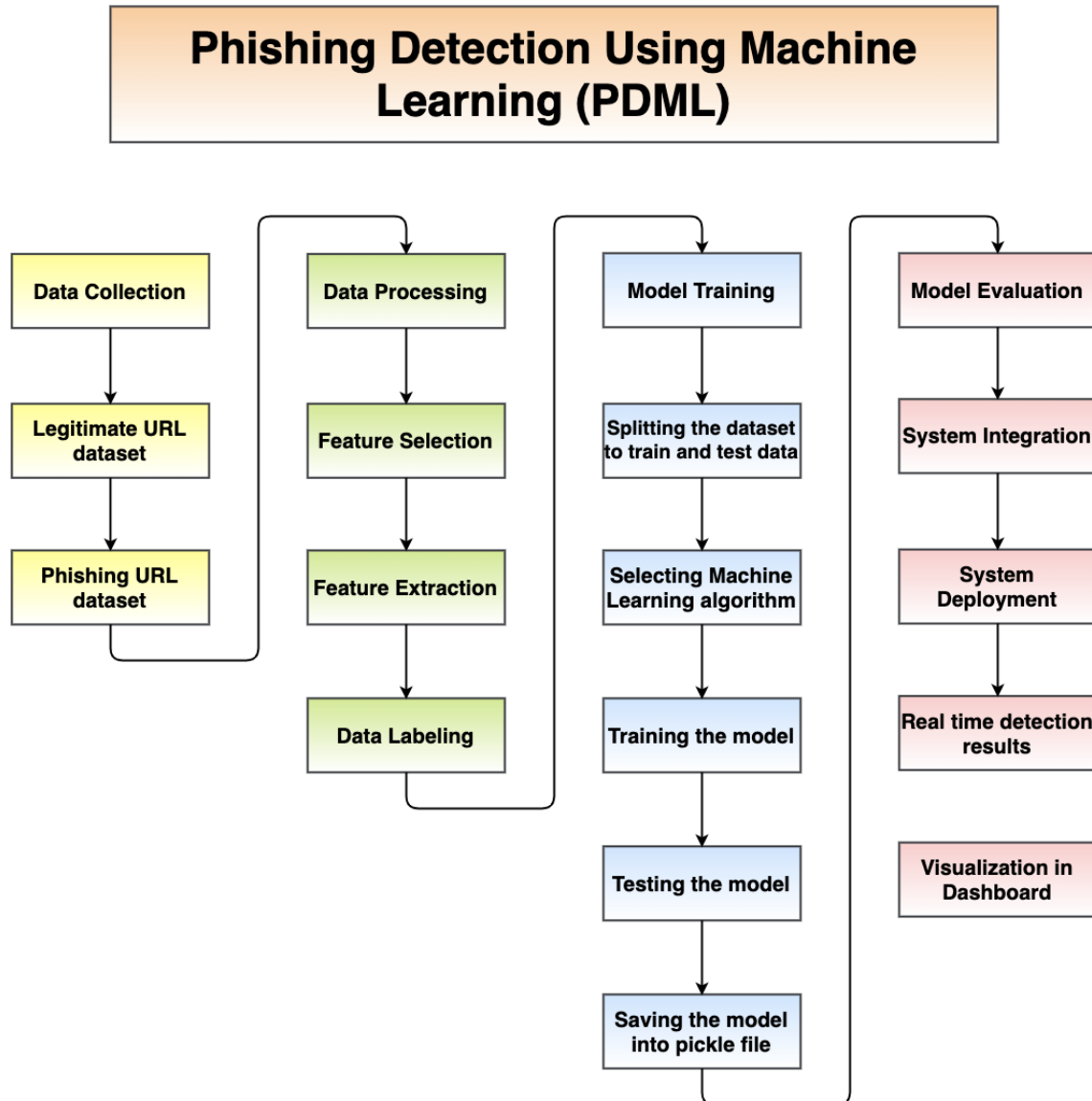


Figure 8: Block Diagram.

3.8.4 Wireframes

The Wireframes of the system are placed in the appendix section of the report, [Click Here](#) to view it.

3.8.5 Gantt Chart

A Gantt chart is a timeline representation of tasks that have been completed or are scheduled to be completed. It denotes when and for how long the processes take place and it is shown in a horizontal format.

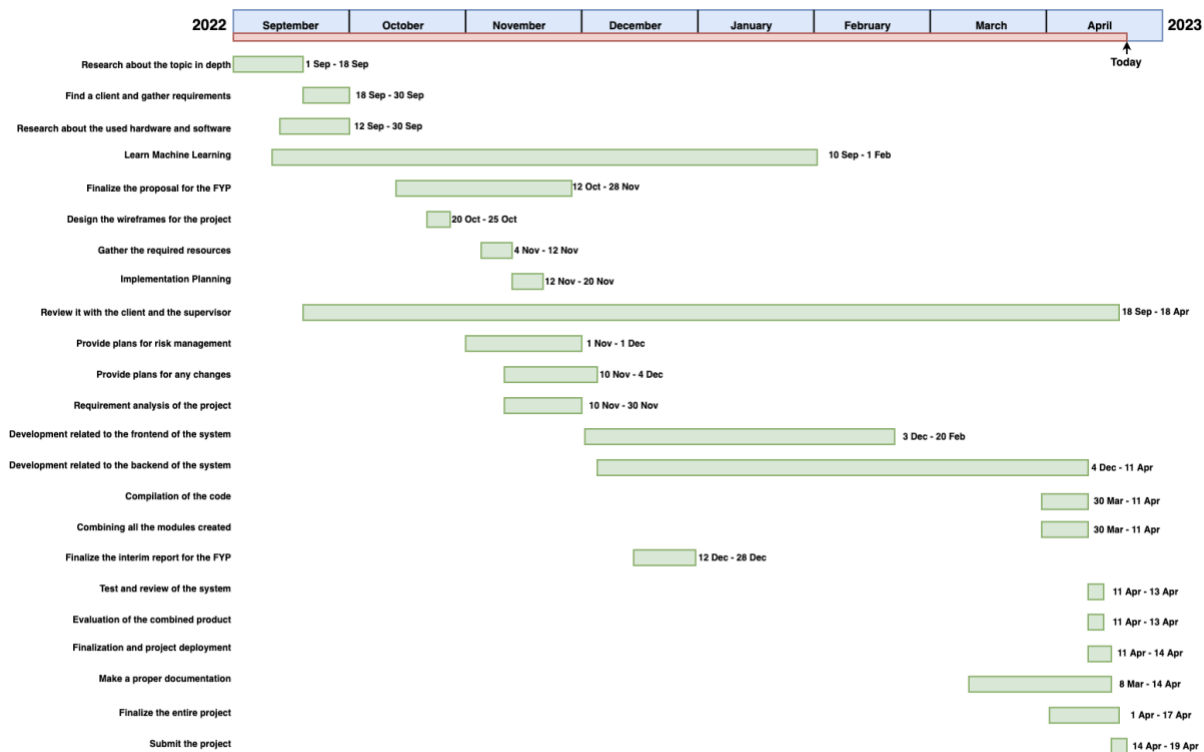


Figure 9: Updated Gantt Chart.

Table 2: Tabular form of the dates in the Updated Gantt Chart.

Tasks	Start Date	End Date	Duration
Research about the topic in depth	01/09/2022	18/09/2022	18 Days
Find a client and gather requirements	18/09/2022	30/09/2022	13 Days
Research about the used hardware and software	12/09/2022	30/09/2022	19 Days
Learn Machine Learning	10/09/2022	01/02/2023	143 Days
Finalize the proposal for the FYP	12/10/2022	28/11/2022	48 Days
Design the wireframes for the project	20/10/2022	25/10/2022	6 Days
Gather the required resources	04/11/2022	12/11/2022	9 Days
Implementation Planning	12/11/2022	20/11/2022	9 Days
Review it with the client and the supervisor	18/09/2022	18/04/2023	212 Days
Provide plans for risk management	01/11/2022	01/12/2022	31 Days
Provide plans for any changes	10/11/2022	04/12/2022	25 Days
Requirement analysis of the project	10/11/2022	30/11/2022	21 Days
Development related to the frontend of the system	03/12/2022	20/02/2023	80 Days
Development related to the backend of the system	04/11/2022	11/04/2023	158 Days
Compilation of the code	30/03/2023	11/04/2023	13 Days
Combining all the modules created	30/03/2023	11/04/2023	13 Days
Finalize the interim report for the FYP	12/12/2022	28/12/2022	16 Days
Test and review of the system	11/04/2023	13/04/2023	3 Days
Evaluation of the combined product	11/04/2023	13/04/2023	3 Days
Finalization and project deployment	11/04/2023	14/04/2023	4 Days
Make a proper documentation	08/03/2023	14/04/2023	38 Days
Finalize the entire project	01/04/2023	17/04/2023	17 Days
Submit the project	14/04/2023	19/04/2023	6 Days

The work breakdown structure is split and classed depending on the time in the Gantt chart above. Various milestones are also set over the time span, as mentioned in the milestone section above. The project will take roughly 7 months to complete from start to finish. The timeline begins with the date the topic was chosen and ends with the project's submission.

Comparison between Gantt Chart:

The dates of the components in the Gantt Chart for the interim and final report of the PDML project may have changed for a number of reasons that are justifiable. For example, due to schedule conflicts or changes in the project scope, discussing the project with the client and supervisor may have taken longer than intended. The development of the system's front-end and back-end that got completed before the assumed duration. Due to debugging needs or unanticipated complications, compiling the code and integrating all of the modules developed might have taken longer than planned.

Testing, evaluating the system, and project deployment got finished before time as the development of the system was completed early. Writing the final report and finalizing the project got ended early ensuring that everything was included and appropriately portrayed the project's outcomes as both of the components were already started ahead of time. These are all reasonable reasons why the dates in the Gantt chart may have moved during the course of the project.

The Old Gantt Chart of the system is placed in the appendix section of the report, [Click Here](#) to view it.

3.9 Implementation

This project generates the entire system using various tools and techniques. The tools and techniques employed throughout the project's completion perform admirably in enabling the system's practical implementation. This project was designed and developed using the Scrum Methodology, which divided the project into various cycles that included aspects of the chosen methodology. The project was completed by incorporating all of the following design elements:

3.9.1 Data Collection

In this period of the development of this system, we collected a large amount of data from a phish tank which is published by “The University of New Brunswick” and the rest from the platform such as Kaggle which consisted of both legitimate and phishing URL.

Where more than 10,000 URLs were taken from OpenPhish which is a repository of active sites and 35,300 benign URLs were gathered from top Alexa websites. The domains were crawled by a Heritrix web crawler to extract the URLs. Approximately 500,000 unique URLs are crawled initially and then passed through to remove duplicate and domain-only URLs. The extracted URLs were then run through Virustotal to filter out the benign URLs (University of New Brunswick, 2023).

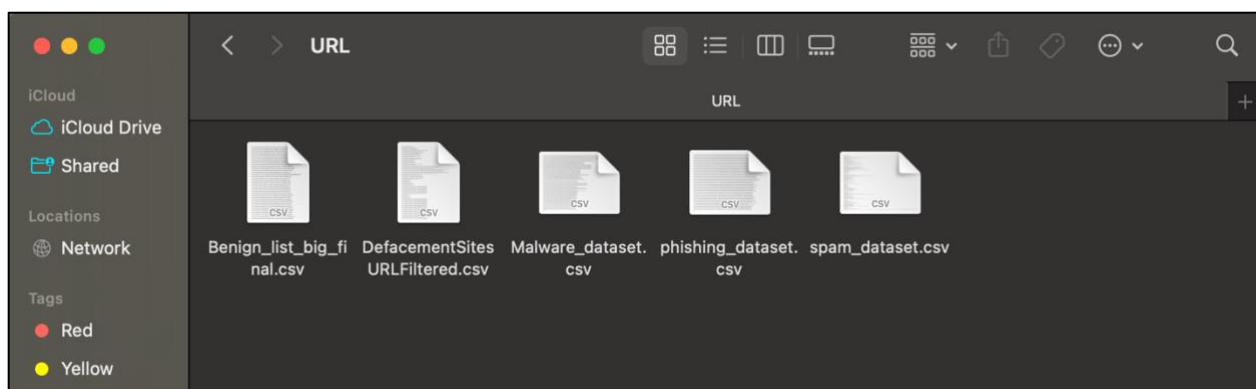


Figure 10: Folder Containing a list of URLs.



3.9.2 Feature Extraction

During this stage of the system's development, relevant features such as search bar-based features and domain-based features such as domain name, count of special character signs present in the URL, IP present in the URL, and many other features are extracted from the relevant URL and two new dataset files are created which contain both legitimate and phishing URL after feature extraction.

```
[193] # 8. Checking for Shortening Services in URL (Tiny_URL)
def tinyURL(url):
    ... match=re.search(shortening_services,url)
    ... if match:
    ...     ... return 1
    ... else:
    ...     ... return 0

[194] # 9. Checking for Prefix or Suffix Separated by (-) in the Domain (Prefix/Suffix)
def prefixSuffix(url):
    ... if '-' in urlparse(url).netloc:
    ...     ... return 1          # phishing
    ... else:
    ...     ... return 0          # legitimate

[195] def ippresent(url):
    domain = urlparse(url).netloc
    try:
    |     ip = socket.gethostbyname("domain")
    |     return 1
    except:
    |     return 0
```

Figure 13: Feature extraction code for search bar-based features.

2. Domain Based Feature

```
#The function dns_record takes a URL as input and returns a binary value indicating whether the domain has a DNS record or not.
def dns_record(url):
    domain_name = urlparse(url).netloc
    try:
        rec = whois.whois(domain_name)
        return 1
    except:
        return 0
```

[201]

Python

This feature measures the popularity of the website by determining the number of visitors and the number of pages they visit. However, since phishing websites live for a short period of time, they may not be recognized by the Alexa database (Alexa the Web Information Company., 1996). By reviewing our dataset, we find that in worst scenarios, legitimate websites ranked among the top 100,000. Furthermore, if the domain has no traffic or is not recognized by the Alexa database, it is classified as "Phishing".

```
def web_traffic(url):
    try:
        rank = BeautifulSoup(urllib.request.urlopen("http://data.alexa.com/data?cli=10&dat=s&url=" + url).read(), "xml").find("REACH")["RANK"]
        rank = int(rank)
        return rank
    except:
        return 0 #0 = Phishing
```

[202]

Python

Figure 14: Feature extraction code for domain-based features.

Extracting feature to phishing URL's

```
#Collecting 50 Phishing URLs randomly
phishUrl = data_P.copy()
phishUrl = phishUrl.reset_index(drop=True)
phishUrl.head()
```

[209]

Python

	phish_id	url	phish_detail_url	submission_time	verified	verification_time	online	ts
0	6557033	http://u1047531.cp.regruhosting.ru/acces-inges...	http://www.phishtank.com/phish_detail.php?phis...	2020-05-09T22:01:43+00:00	yes	2020-05-09T22:03:07+00:00	yes	C
1	6557032	http://hoysalacreations.com/wp-content/plugins...	http://www.phishtank.com/phish_detail.php?phis...	2020-05-09T22:01:37+00:00	yes	2020-05-09T22:03:07+00:00	yes	C
2	6557011	http://www.accsystemprblmhelp.site/checkpoint...	http://www.phishtank.com/phish_detail.php?phis...	2020-05-09T21:54:31+00:00	yes	2020-05-09T21:55:38+00:00	yes	Face
3	6557010	http://www.accsystemprblmhelp.site/login_atte...	http://www.phishtank.com/phish_detail.php?phis...	2020-05-09T21:53:48+00:00	yes	2020-05-09T21:54:34+00:00	yes	Face
4	6557009	https://firebasestorage.googleapis.com/v0/b/so...	http://www.phishtank.com/phish_detail.php?phis...	2020-05-09T21:49:27+00:00	yes	2020-05-09T21:51:24+00:00	yes	Micro

```
data_P.shape
```

[210]

Python

```
(14858, 8)
```

Figure 15: Extracting features to phishing URLs (a).


```

#Extracting the features and storing them in a list
phish_features = []
label = 1

for i in range(0, 14858):
    url = phishUrl['url'][i]
    phish_features.append(feature_extraction(url, label))

[212] Python

feature_names = ['Domain', 'IP', '@', 'URL Length', 'URL Depth', 'Redirection', 'https Domain', 'TinyURL',
                  'Prefix/Suffix', 'IP Present', 'HTTPS Token', 'Protocol Count', 'Protocol', 'Special Character Count',
                  'DNS Record', 'Web Traffic', 'Domain Age', 'Label']
phishing = pd.DataFrame(phish_features, columns= feature_names)
phishing.head()

[215] Python

...

```

	IP	@	URL Length	URL Depth	Redirection	https Domain	TinyURL	Prefix/Suffix	IP Present	HTTPS Token	Protocol Count	Protocol	Special Character Count	DNS Record	Web Traffic	Domain Age	Label
0	0	0	1	2	0	0	0	0	0	1	1	1	0	0	0	0	1
1	0	0	1	6	0	0	0	0	0	1	1	1	0	0	0	0	1
2	0	0	0	1	0	0	0	0	0	1	1	1	0	0	0	0	1
3	0	0	1	1	0	0	0	0	0	1	1	1	5	0	0	0	1
4	0	0	1	5	0	0	1	0	0	0	1	0	5	0	0	0	1

```

# Storing the extracted phishing URLs fatures to csv file
phishing.to_csv('data/phishing.csv', index= False)

[216] Python

```

Figure 16: Extracting features to phishing URLs (b).

Extracting feature to legitimate URL's

```

#Collecting 50 Legitimate URLs randomly
legitUrl = data_L.copy()
legitUrl = legitUrl.reset_index(drop=True)
legitUrl.head()

[217] Python

...

```

	URLs
0	http://1337x.to/torrent/1110018/Blackhat-2015-...
1	http://1337x.to/torrent/1122940/Blackhat-2015-...
2	http://1337x.to/torrent/1124395/Fast-and-Furio...
3	http://1337x.to/torrent/1145504/Avengers-Age-o...
4	http://1337x.to/torrent/1160078/Avengers-age-o...

```

data_L.shape

[218] Python

... (35377, 1)

#Extracting the features and storing them in a list
legit_features = []
label = 0

for i in range(0, 35377):
    url = legitUrl['URLs'][i]
    legit_features.append(feature_extraction(url, label))

[219] Python

```

Figure 17: Extracting features to legitimate URLs (a).


```

#Extracting the features and storing them in a list
legit_features = []
label = 0

for i in range(0, 35377):
    url = legitUrl['URLs'][i]
    legit_features.append(feature_extraction(url, label))

```

[219] Python

```

feature_names = ['Domain', 'IP', '@', 'URL Length', 'URL Depth', 'Redirection', 'https Domain', 'TinyURL',
                 'Prefix/Suffix', 'IP Present', 'HTTPS Token', 'Protocol Count', 'Protocol', 'Special Character Count',
                 'DNS Record', 'Web Traffic', 'Domain Age', 'Label']
legitimate = pd.DataFrame(legit_features, columns= feature_names)
legitimate.head()

```

[220] Python

	IP	@	URL Length	URL Depth	Redirection	https Domain	TinyURL	Prefix/Suffix	IP Present	HTTPS Token	Protocol Count	Protocol	Special Character Count	DNS Record	Web Traffic	Domain Age	Label
0	0	0	1	3	0	0	0	0	0	1	1	1	0	0	0	0	0
1	0	0	1	3	0	0	0	0	0	1	1	1	0	0	0	0	0
2	0	0	1	3	0	0	0	0	0	1	1	1	0	0	0	0	0
3	0	0	1	3	0	0	0	0	0	1	1	1	0	0	0	0	0
4	0	0	1	3	0	0	0	0	0	1	1	1	0	0	0	0	0

```

# Storing the extracted legitimate URLs fatures to csv file
legitimate.to_csv('data/legitimate.csv', index= False)

```

[221] Python

Figure 18: Extracting features to legitimate URLs (b).

3.9.3 Model Training

During this stage of the system's development, the ML model is trained using different algorithms such as Logistic Regression, Random Forest, or Deep Learning algorithms where we choose the algorithm that performs the best on the training data.

```

Splitting the data

# Separating & assigning features and target columns to X & y
y = data['Label']
x = data.drop('Label', axis=1)
x.shape, y.shape

[62]
... ((50235, 16), (50235,))

# Splitting the dataset into train and test sets: 70-30 split
from sklearn.model_selection import train_test_split

x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.3, random_state = 12)
x_train.shape, x_test.shape

[63]
... ((35164, 16), (15071, 16))

```

Figure 19: Splitting the data for model training and testing.

```

BULK TRAINING

models = [DecisionTreeClassifier(), RandomForestClassifier(), MLPClassifier(), SVC(), LogisticRegression(), GradientBoostingClassifier()]
for model in models:
    model.fit(x_train, y_train)
    print(str(model), ' -> train_accuracy: ', model.score(x_train, y_train))

[66]
... DecisionTreeClassifier() -> train_accuracy: 0.9274826527130019
RandomForestClassifier() -> train_accuracy: 0.9274826527130019
/Library/Frameworks/Python.framework/Versions/3.10/lib/python3.10/site-packages/sklearn/neural_network/_multilayer_perceptron.py:577: UserWarning:
warnings.warn(
MLPClassifier() -> train_accuracy: 0.9183824365828689
SVC() -> train_accuracy: 0.9070356045956092
/Library/Frameworks/Python.framework/Versions/3.10/lib/python3.10/site-packages/sklearn/linear_model/_logistic.py:458:
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
https://scikit-learn.org/stable/modules/linear\_model.html#logistic-regression
n_iter_i = _check_optimize_result(
LogisticRegression() -> train_accuracy: 0.8919064952792629
GradientBoostingClassifier() -> train_accuracy: 0.9124388579228757
AdaBoostClassifier() -> train_accuracy: 0.8946081219428962

```

Figure 20: Training the model in bulk.

3.9.4 Model Testing

During this stage of the system's development, the model's accuracy is tested by running it on a separate set of data that the model has not ever seen before.

```

BULK TESTING

models = [DecisionTreeClassifier(), RandomForestClassifier(), MLPClassifier(), SVC(), LogisticRegression(), GradientBoostingClassifier()],
for model in models:
    model.fit(x_train, y_train)
    print(str(model), ' -> test_accuracy: ', model.score(x_test, y_test))

[69]
... DecisionTreeClassifier() -> test_accuracy: 0.9258177957667043
RandomForestClassifier() -> test_accuracy: 0.9276093159047176
/Library/Frameworks/Python.framework/Versions/3.10/lib/python3.10/site-packages/sklearn/neural_network/multilayer_perceptron.py:684: ConvergenceWarning:
  warnings.warn(
MLPClassifier() -> test_accuracy: 0.9208413509388893
SVC() -> test_accuracy: 0.9125472762258643
/Library/Frameworks/Python.framework/Versions/3.10/lib/python3.10/site-packages/sklearn/linear_model/logistic.py:458: ConvergenceWarning:
  STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

  Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
  Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
    n_iter_i = _check_optimize_result(
LogisticRegression() -> test_accuracy: 0.896224537190631
GradientBoostingClassifier() -> test_accuracy: 0.9171919580651583
AdaBoostClassifier() -> test_accuracy: 0.8984805255125738

```

Figure 21: Testing the model in bulk.

3.9.5 Dashboard Development

During this stage of systems development, the development of the dashboard is coded on 'next.js' programming language and uses react framework for its development. The 'antd' library which is an open library of the react framework is also used to make pre-created tables, divisions, buttons, and many more.

```

1 import { Form, Input, Row, Button, Col } from 'antd'
2 import Head from 'next/head'
3 import { useRouter } from 'next/router';
4
5 export default function Home() {
6   const router = useRouter();
7   return (
8     <
9       <Head>
10        <title>Phishing Detection FYP | Sarthak Rana</title>
11        <meta name="description" content="Generated by create next app" />
12        <meta name="viewport" content="width=device-width, initial-scale=1" />
13        <link rel="icon" href="/favicon.ico" />
14      </Head>
15      <main className="h-screen flex items-center flex-col justify-center gap-4 bg-slate-100 overflow-hidden">
16        <h1 className="font-bold text-8xl text-black/70 -mt-16">Welcome</h1>
17        <h2 className="font-medium text-3xl text-black/50 mb-4">This site scans for phishing URLs!!</h2>
18        <Form className="w-full" onFinish={async (data) => { console.log(data);
19          try {
20            let result = await fetch(`http://localhost:8000/search?url=${encodeURIComponent(data.url)}`)
21            result = await result.json();
22            console.log(result)
23          } catch (error) {
24            console.log(error)
25          }
26        }
27        router.push('/dashboard')
28      >>
29      <Row className="flex justify-center" gutter={[24, 0]}>
30        <Col span={8}>
31          <Form.Item name="url" rules={[{
32            required: true, message: "Please Enter an URL!!!"
33          }]}>
34            <Input placeholder="Enter a URL:www.example.com" size="large" />
35          </Form.Item>
36        </Col>
37        <Col>
38          <Button size="large" type="primary" htmlType="submit"><span className="px-8 font-bold">Scan</span></Button>
39        </Col>
40      </Row>
41    </Form>
42  </main>
43 </>
44 )
45 }

```

Figure 22: Dashboard Development Code (a).

```
1  import Head from "next/head";
2  import { Table, Button, Modal, ConfigProvider } from "antd";
3  import { useRouter } from "next/router";
4  import Link from "next/link";
5  import { useState } from "react";
6
7  export default function Dashboard() {
8      const router = useRouter();
9      const [activeDetails, setActiveDetails] = useState({
10         url: "https://www.virustotal.com",
11         ip: "192.168.1.1",
12         hostingProvider: "Google LLC",
13         originalDeposition: "Phishing",
14         detectionDate: "2022-04-23",
15         modalPredection: 40,
16         probability: 80,
17         communityScore: 22,
18     });
19     const [showModal, setShowModal] = useState(false)
20     const [dataSource, setDataSource] = useState([
21         {
22             url: "https://www.a.com",
23             ip: "192.168.1.1",
24             hostingProvider: "Google LLC",
25             originalDeposition: "Phishing",
26             detectionDate: "2022-04-23",
27             modalPredection: 10,
28             probability: 20,
29             communityScore: 56
30         },
```

Figure 23: Dashboard Development Code (b).


```

73     <Head>
74         <title>DashBoard | Sarthak Rana</title>
75     </Head>
76     <main className="bg-slate-100 min-h-screen pt-12 px-24">
77         <Modal open={showModal} footer={null} width="750" centered onCancel={() => { setShowModal(false) }}>
78             <main className="m-6 grid grid-cols-2 gap-4">
79                 <div className="w-96 border flex flex-col gap-6 p-4">
80                     <div className="">
81                         <div className="text-black/60 font-bold">Source URL:</div>
82                         <a href={activeDetails.url} className="">{activeDetails.url}</a>
83                     </div>
84                     <div className="grid grid-cols-2">
85                         <div className="">
86                             <div className="text-black/60 font-bold">IP Address:</div>
87                             <a href={activeDetails.ip} className="">{activeDetails.ip}</a>
88                         </div>
89                         <div className="">
90                             <div className="text-black/60 font-bold">Status:</div>
91                             <div className="text-black/60 font-medium">{activeDetails.originalDeposition}</div>
92                         </div>
93                     </div>
94                     <div className="grid grid-cols-2">
95                         <div className="">
96                             <div className="text-black/60 font-bold">Location:</div>
97                             <div className="text-black/60 font-medium">{activeDetails.detectionDate}</div>
98                         </div>
99                         <div className="">
100                             <div className="text-black/60 font-bold">Domain:</div>
101                             <div className="text-black/60 font-medium">{activeDetails.detectionDate}</div>
102                         </div>
103                     </div>
104                     <div className="grid grid-cols-2">
105                         <div className="">
106                             <div className="text-black/60 font-bold">Title:</div>
107                             <div className="text-black/60 font-medium">{activeDetails.detectionDate}</div>
108                         </div>
109                         <div className="">
110                             <div className="text-black/60 font-bold">First Submission:</div>
111                             <div className="text-black/60 font-medium">{activeDetails.detectionDate}</div>
112                         </div>
113                     </div>
114                     <div className="grid grid-cols-2">
115                         <div className="">
116                             <div className="text-black/60 font-bold">Last Submission:</div>
117                             <div className="text-black/60 font-medium">{activeDetails.detectionDate}</div>
118                         </div>
119                         <div className="">
120                             <div className="text-black/60 font-bold">Scan Date:</div>
121                             <div className="text-black/60 font-medium">{activeDetails.detectionDate}</div>
122                         </div>

```

Figure 24: Dashboard Development Code (c).

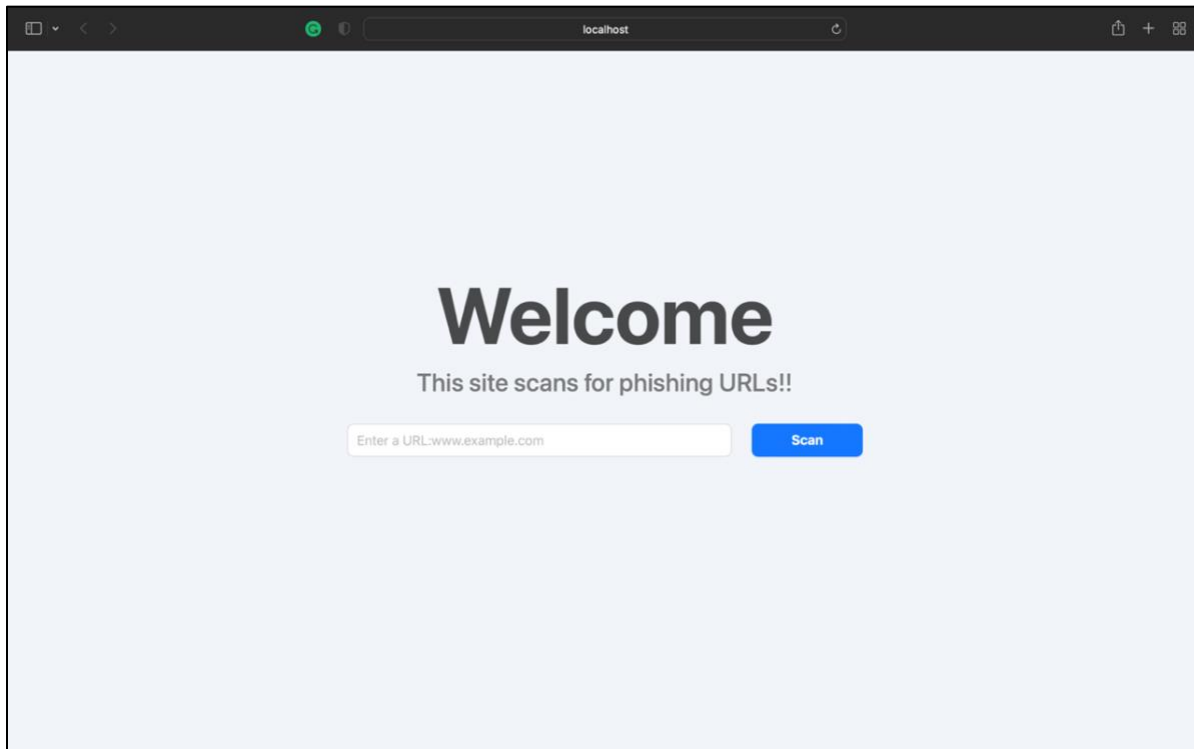


Figure 25: System home page.

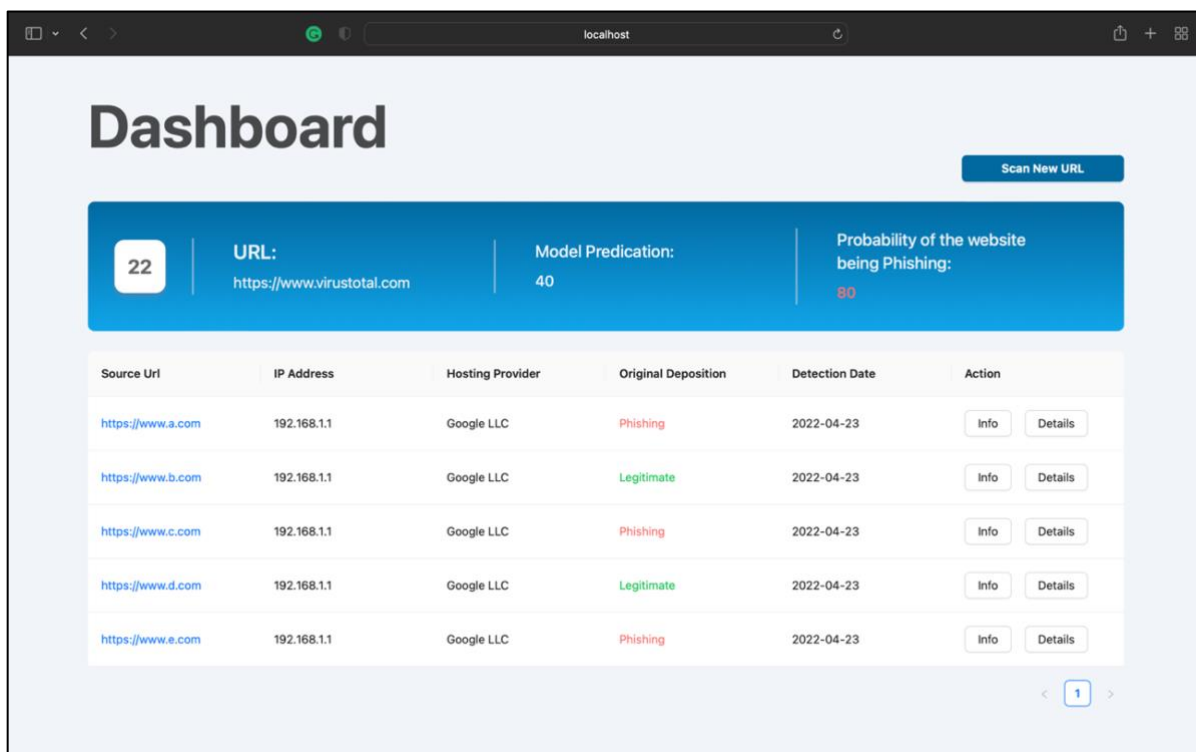


Figure 26: System dashboard page.

3.9.6 Integration

Fetch API is being used at this point in the system's development to connect the backend and front end. A flexible web tool called the Fetch API optimizes HTTP queries to enable frictionless communication between frontend and backend parts of online applications. The user experience is eventually improved by its support for a range of request methods and advanced features, which also enhance error handling and code clarity (JavaScript Tutorial, 2023). Overall, in this project, the web server, which is hosted on Flask, is called or retrieved by the Fetch API in order to gather data from the backend and display it on the front end.

```
setSearchLoading(true);
let result;
try {
  result = await fetch(`http://100.64.250.170:9696/search?url=${encodeURIComponent(data.url)}`);
  result = await result.json();
  store.set("searchScan", result);
  router.push({
    pathname: `/dashboard`,
    // query: { data: JSON.stringify(result) }
  })
} catch (error) {
  console.log(error)
  message.error("This URL doesn't exists")
}
```

Figure 27: Using fetch API for integration.

3.9.7 Deployment

Flask is used in this project as an easy-to-use web framework for creating a web server focused on detecting phishing URLs as it is an important part of our project since it allows for the smooth integration of phishing detection capabilities into a user-friendly web server.

```
app = Flask(__name__)
CORS(app)

@app.route('/search', methods=['GET'])
def search_url():
    return detect_phish(request.args.get("url"))

if __name__ == '__main__':
    app.run(host='0.0.0.0', port=9696)
```

Figure 28: Using flask to create a web server.

CHAPTER 4: Testing and Analysis

4. Testing and Analysis

4.1 Test Plan

4.1.1 Unit Testing, Test Plan

Table 3: Unit Testing, Test Plan.

Test Case	Objectives
1	To test if the datasets go under the code of feature extraction.
2	To test if both the legitimate and phishing dataset is merged or not after the feature extraction.
3	To test the bulk training of the dataset.
4	To test the bulk testing of the dataset.
5	To test if the output of the random forest algorithm is saved in a pickle file.
6	To test if the system is detecting phishing URL or not.
7	To test if the system is detecting legitimate URL or not.
8	To test if the search bar has URL validation or not.
9	To test if the Virus total API is responding or not.
10	To test if the Python code is generating a URL into an IP address or not.
11	To test if the code is generating longitude and latitude or not.

12	To test if the google maps API is generating map or not.
13	To test if the Python code is generating screenshots of the web page or not.
14	To test if the generated screenshot is displaying in the frontend or not.
15	To test if the Python code is generating domain name of a URL or not.
16	To test if the dashboard is displaying the results of all the antivirus vendors or not.
17	To test if the dashboard is displaying the count of number of vendors detecting the URL as phishing.
18	To test if the detection date label in the dashboard is displaying correct date or not.
19	To test if the logs of the URL are stored or not after reloading the page.
20	To test if the system is running on different devices on the same network or not.
21	To check the functionality of the 'Info' button.
22	To check the functionality of the 'Details' button.

4.1.2 System Testing, Test Plan

Table 4: System Testing, Test Plan.

Test Case	Objectives
1	To test the entire system by entering a phishing URL.
2	To test the entire system by entering a legitimate URL.

4.2 Unit Testing

- **Test Case 1: To test if the datasets go under the code of feature extraction.**

Table 5: Test Case 1.

Test Case 1	
Objective	To test if the datasets go under the code of feature extraction.
Action	Pass both phishing and legitimate URL dataset through the feature list in the Python code and save it to different CSV format files.
Expected Result	The URLs should pass through the feature extraction code and get saved into two new different CSV format files.
Actual Result	The URLs get passed through the code and get saved into two new different named CSV format files.
Conclusion	Test Successful.

Evidence:

Extracting feature to phishing URL's

```
#Collecting 50 Phishing URLs randomly
phishUrl = data_P.copy()
phishUrl = phishUrl.reset_index(drop=True)
phishUrl.head()
```

[209] Python

	phish_id	url	phish_detail_url	submission_time	verified	verification_time	online	ti
0	6557033	http://u1047531.cp.regruhosting.ru/acces-inges...	http://www.phishtank.com/phish_detail.php?phis...	2020-05-09T22:01:43+00:00	yes	2020-05-09T22:03:07+00:00	yes	(
1	6557032	http://hoysalacreations.com/wp-content/plugins...	http://www.phishtank.com/phish_detail.php?phis...	2020-05-09T22:01:37+00:00	yes	2020-05-09T22:03:07+00:00	yes	(
2	6557011	http://www.accsystemprblemhelp.site/checkpoint...	http://www.phishtank.com/phish_detail.php?phis...	2020-05-09T21:54:31+00:00	yes	2020-05-09T21:55:38+00:00	yes	Face
3	6557010	http://www.accsystemprblemhelp.site/login_atte...	http://www.phishtank.com/phish_detail.php?phis...	2020-05-09T21:53:48+00:00	yes	2020-05-09T21:54:34+00:00	yes	Face
4	6557009	https://firebasestorage.googleapis.com/v0/b/so...	http://www.phishtank.com/phish_detail.php?phis...	2020-05-09T21:49:27+00:00	yes	2020-05-09T21:51:24+00:00	yes	Micr

```
data_P.shape
```

[210] Python

... (14858, 8)

Figure 29: Extracting features to phishing URLs (a).

```
#Extracting the features and storing them in a list
phish_features = []
label = 1

for i in range(0, 14858):
    url = phishUrl['url'][i]
    phish_features.append(feature_extraction(url,label))
```

[212] Python

```
feature_names = ['Domain', 'IP', '@', 'URL Length', 'URL Depth', 'Redirection', 'https Domain', 'TinyURL',
                  'Prefix/Suffix', 'IP Present', 'HTTPS Token', 'Protocol Count', 'Protocol', 'Special Character Count',
                  'DNS Record', 'Web Traffic', 'Domain Age', 'Label']
phishing = pd.DataFrame(phish_features, columns= feature_names)
phishing.head()
```

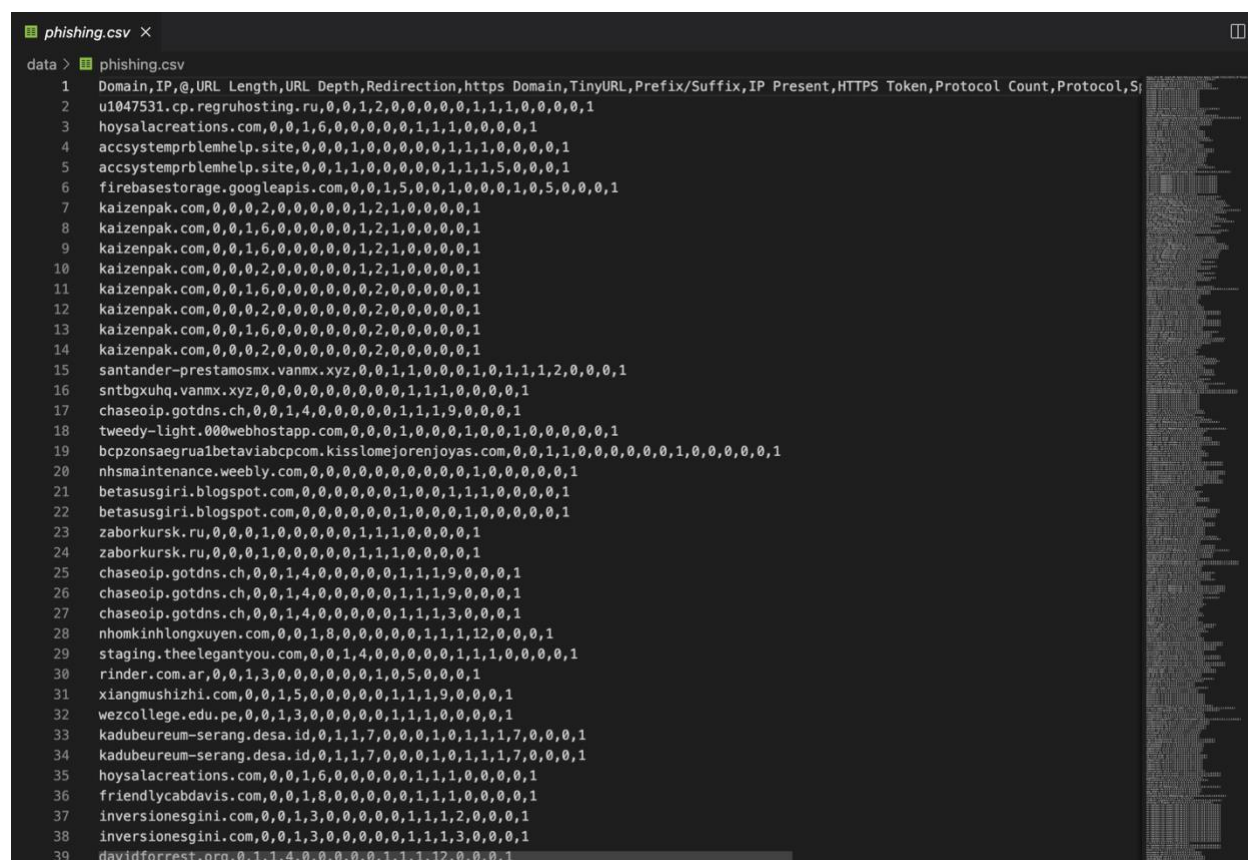
[215] Python

	IP	@	URL Length	URL Depth	Redirection	https Domain	TinyURL	Prefix/Suffix	IP Present	HTTPS Token	Protocol Count	Protocol	Special Character Count	DNS Record	Web Traffic	Domain Age	Label
0	0	0	1	2	0	0	0	0	0	1	1	1	0	0	0	0	1
1	0	0	1	6	0	0	0	0	0	1	1	1	0	0	0	0	1
2	0	0	0	1	0	0	0	0	0	1	1	1	0	0	0	0	1
3	0	0	1	1	0	0	0	0	0	1	1	1	5	0	0	0	1
4	0	0	1	5	0	0	1	0	0	0	1	0	5	0	0	0	1

```
# Storing the extracted phishing URLs fatures to csv file
phishing.to_csv('data/phishing.csv', index= False)
```

[216] Python

Figure 30: Extracting features to phishing URLs (b).



```

phishing.csv
data > phishing.csv
1 Domain,IP,@,URL Length,URL Depth,Redirection,https Domain,TinyURL,Prefix/Suffix,IP Present,HTTPS Token,Protocol Count,Protocol,S
2 ui047531.cp.regruhosting.ru,0,0,1,2,0,0,0,0,0,1,1,1,0,0,0,0,1
3 hoysalacreation.com,0,0,1,6,0,0,0,0,0,1,1,1,0,0,0,0,1
4 accsystemprblemhelp.site,0,0,0,1,0,0,0,0,0,1,1,1,0,0,0,0,1
5 accsystemprblemhelp.site,0,0,1,1,0,0,0,0,0,1,1,1,5,0,0,0,1
6 firebasestorage.googleapis.com,0,0,1,5,0,0,1,0,0,0,1,0,5,0,0,0,1
7 kaizenpak.com,0,0,0,2,0,0,0,0,0,1,2,1,0,0,0,0,1
8 kaizenpak.com,0,0,1,6,0,0,0,0,0,1,2,1,0,0,0,0,1
9 kaizenpak.com,0,0,1,6,0,0,0,0,0,1,2,1,0,0,0,0,1
10 kaizenpak.com,0,0,0,2,0,0,0,0,0,1,2,1,0,0,0,0,1
11 kaizenpak.com,0,0,1,6,0,0,0,0,0,0,2,0,0,0,0,0,1
12 kaizenpak.com,0,0,0,2,0,0,0,0,0,0,2,0,0,0,0,0,1
13 kaizenpak.com,0,0,1,6,0,0,0,0,0,0,2,0,0,0,0,0,1
14 kaizenpak.com,0,0,0,2,0,0,0,0,0,0,2,0,0,0,0,0,1
15 santander-prestamosx.vanmx.xyz,0,0,1,1,0,0,0,1,0,1,1,1,2,0,0,0,1
16 sntbgxuhq.vanmx.xyz,0,0,0,0,0,0,0,0,0,1,1,1,0,0,0,0,1
17 chaseip.gotdns.ch,0,0,1,4,0,0,0,0,0,1,1,1,9,0,0,0,1
18 tweedy-light.000webhostapp.com,0,0,0,1,0,0,0,1,0,0,1,0,0,0,0,0,1
19 bcpzonsaegruiabetaiabcpcom.kisslomejorenjoyas.com,0,0,1,1,0,0,0,0,0,1,0,0,0,0,0,1
20 nhsmaintenance.weebly.com,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,1
21 betasugiri.blogspot.com,0,0,0,0,0,0,1,0,0,1,1,1,0,0,0,0,1
22 betasugiri.blogspot.com,0,0,0,0,0,0,1,0,0,1,0,0,0,0,0,0,1
23 zaborkursk.ru,0,0,0,1,0,0,0,0,0,1,1,1,0,0,0,0,1
24 zaborkursk.ru,0,0,0,1,0,0,0,0,0,1,1,1,0,0,0,0,1
25 chaseip.gotdns.ch,0,0,1,4,0,0,0,0,0,1,1,1,9,0,0,0,1
26 chaseip.gotdns.ch,0,0,1,4,0,0,0,0,0,1,1,1,9,0,0,0,1
27 chaseip.gotdns.ch,0,0,1,4,0,0,0,0,0,1,1,1,3,0,0,0,1
28 nhomkinhlongxuyen.com,0,0,1,8,0,0,0,0,0,1,1,1,12,0,0,0,1
29 staging.theelegantlyou.com,0,0,1,4,0,0,0,0,0,1,1,1,0,0,0,0,1
30 rinder.com.ar,0,0,1,3,0,0,0,0,0,1,0,5,0,0,0,0,1
31 xiangmushizhi.com,0,0,1,5,0,0,0,0,0,1,1,1,9,0,0,0,1
32 wezcollege.edu.pe,0,0,1,3,0,0,0,0,0,1,1,1,0,0,0,0,1
33 kadubeureum-serang.desa.id,0,1,1,7,0,0,0,1,0,1,1,1,7,0,0,0,1
34 kadubeureum-serang.desa.id,0,1,1,7,0,0,0,1,0,1,1,1,7,0,0,0,1
35 hoysalacreation.com,0,0,1,6,0,0,0,0,0,1,1,1,0,0,0,0,1
36 friendlycabdavis.com,0,0,1,8,0,0,0,0,0,1,1,1,0,0,0,0,1
37 inversionesgini.com,0,0,1,3,0,0,0,0,0,1,1,1,2,0,0,0,1
38 inversionesgini.com,0,0,1,3,0,0,0,0,0,1,1,1,3,0,0,0,1
39 davidforrest.org,0,1,1,4,0,0,0,0,0,1,1,1,12,0,0,0,1

```

Figure 31: Feature extracted dataset for phishing URLs.

Extracting feature to legitimate URL's

```
#Collecting 50 Legitimate URLs randomly
legitUrl = data_L.copy()
legitUrl = legitUrl.reset_index(drop=True)
legitUrl.head()
```

[217] Python

```
...
URLs
0 http://1337x.to/torrent/1110018/Blackhat-2015-...
1 http://1337x.to/torrent/1122940/Blackhat-2015-...
2 http://1337x.to/torrent/1124395/Fast-and-Furio...
3 http://1337x.to/torrent/1145504/Avengers-Age-o-...
4 http://1337x.to/torrent/1160078/Avengers-age-o-...
```

```
data_L.shape
```

[218] Python

```
... (35377, 1)
```

```
#Extracting the features and storing them in a list
legit_features = []
label = 0

for i in range(0, 35377):
    url = legitUrl['URLs'][i]
    legit_features.append(feature_extraction(url, label))
```

[219] Python

Figure 32: Extracting features to legitimate URLs (a).

```
#Extracting the features and storing them in a list
legit_features = []
label = 0

for i in range(0, 35377):
    url = legitUrl['URLs'][i]
    legit_features.append(feature_extraction(url, label))
```

[219] Python

```
feature_names = ['Domain', 'IP', '@', 'URL Length', 'URL Depth', 'Redirection', 'https Domain', 'TinyURL',
                 'Prefix/Suffix', 'IP Present', 'HTTPS Token', 'Protocol Count', 'Protocol', 'Special Character Count',
                 'DNS Record', 'Web Traffic', 'Domain Age', 'Label']
legitimate = pd.DataFrame(legit_features, columns=feature_names)
legitimate.head()
```

[220] Python

	IP	@	URL Length	URL Depth	Redirection	https Domain	TinyURL	Prefix/Suffix	IP Present	HTTPS Token	Protocol Count	Protocol	Special Character Count	DNS Record	Web Traffic	Domain Age	Label
0	0	0	1	3	0	0	0	0	0	1	1	1	0	0	0	0	0
1	0	0	1	3	0	0	0	0	0	1	1	1	0	0	0	0	0
2	0	0	1	3	0	0	0	0	0	1	1	1	0	0	0	0	0
3	0	0	1	3	0	0	0	0	0	1	1	1	0	0	0	0	0
4	0	0	1	3	0	0	0	0	0	1	1	1	0	0	0	0	0

```
# Storing the extracted legitimate URLs features to csv file
legitimate.to_csv('data/legitimate.csv', index=False)
```

[221] Python

Figure 33: Extracting features to legitimate URLs (b).

```

legitimate.csv ×
data > legitimate.csv
1 Domain,IP,URL Length,URL Depth,Redirection,https Domain,TinyURL,Prefix/Suffix,IP Present,HTTPS Token,Protocol Count,Protocol,S
2 1337x.to,0,0,1,3,0,0,0,0,0,1,1,1,0,0,0,0,0
3 1337x.to,0,0,1,3,0,0,0,0,0,1,1,1,0,0,0,0,0
4 1337x.to,0,0,1,3,0,0,0,0,0,1,1,1,0,0,0,0,0
5 1337x.to,0,0,1,3,0,0,0,0,0,1,1,1,0,0,0,0,0
6 1337x.to,0,0,1,3,0,0,0,0,0,1,1,1,0,0,0,0,0
7 1337x.to,0,0,1,3,0,0,0,0,0,1,1,1,0,0,0,0,0
8 189.cn,0,0,1,3,0,0,0,0,0,1,1,1,5,0,0,0,0
9 2gis.ru,0,0,1,7,0,0,0,0,0,1,1,1,0,0,0,0,0
10 abc.go.com,0,0,1,5,0,0,0,0,0,1,1,1,0,0,0,0,0
11 abc.go.com,0,0,1,4,0,0,0,0,0,1,1,1,3,0,0,0,0
12 abcnews.go.com,0,0,1,3,0,0,0,0,0,1,1,1,0,0,0,0,0
13 adultfriendfinder.com,0,0,1,6,0,0,0,0,0,1,1,1,2,0,0,0,0
14 akhbarelyom.com,0,0,1,5,0,0,0,0,0,1,1,1,0,0,0,0,0
15 allegro.pl,0,0,1,1,0,0,0,0,0,1,1,1,0,0,0,0,0
16 allegro.pl,0,0,1,1,0,0,0,0,0,1,1,1,2,0,0,0,0
17 allegro.pl,0,0,1,1,0,0,0,0,0,1,1,1,0,0,0,0,0
18 allegro.pl,0,0,1,1,0,0,0,0,0,1,1,1,0,0,0,0,0
19 allegro.pl,0,0,1,1,0,0,0,0,0,1,1,1,2,0,0,0,0
20 allegro.pl,0,0,1,1,0,0,0,0,0,1,1,1,2,0,0,0,0
21 allegro.pl,0,0,1,1,0,0,0,0,0,1,1,1,0,0,0,0,0
22 allegro.pl,0,0,1,1,0,0,0,0,0,1,1,1,0,0,0,0,0
23 allegro.pl,0,0,1,1,0,0,0,0,0,1,1,1,0,0,0,0,0
24 allrecipes.com,0,0,1,2,0,0,0,0,0,1,1,1,4,0,0,0,0
25 allrecipes.com,0,0,1,3,0,0,0,0,0,1,1,1,4,0,0,0,0
26 allrecipes.com,0,0,1,3,0,0,0,0,0,1,1,1,4,0,0,0,0
27 allrecipes.com,0,0,1,3,0,0,0,0,0,1,1,1,4,0,0,0,0
28 allrecipes.com,0,0,1,3,0,0,0,0,0,1,1,1,4,0,0,0,0
29 allrecipes.com,0,0,1,3,0,0,0,0,0,1,1,1,4,0,0,0,0
30 allrecipes.com,0,0,1,3,0,0,0,0,0,1,1,1,4,0,0,0,0
31 allrecipes.com,0,0,1,4,0,0,0,0,0,1,1,1,3,0,0,0,0
32 allrecipes.com,0,0,1,4,0,0,0,0,0,1,1,1,3,0,0,0,0
33 allrecipes.com,0,0,1,4,0,0,0,0,0,1,1,1,3,0,0,0,0
34 anandtech.com,0,0,1,3,0,0,0,0,0,1,1,1,0,0,0,0,0
35 ap.org,0,0,1,4,0,0,0,0,0,1,1,1,0,0,0,0,0
36 ap.org,0,0,1,4,0,0,0,0,0,1,1,1,0,0,0,0,0
37 ap.org,0,0,1,4,0,0,0,0,0,1,1,1,0,0,0,0,0
38 ap.org,0,0,1,4,0,0,0,0,0,1,1,1,0,0,0,0,0
39 ap.org,0,0,1,4,0,0,0,0,0,1,1,1,0,0,0,0,0

```

Figure 34: Feature extracted dataset for legitimate URLs.

- **Test Case 2: To test if both the legitimate and phishing dataset is merged or not after the feature extraction.**

Table 6: Test Case 2.

Test Case 2	
Objective	To test if both the legitimate and phishing dataset is merged or not after the feature extraction.
Action	Merge both features extracted phishing and legitimate CSV file using python code into a single CSV file.
Expected Result	Both legitimate.csv and phishing.csv should get merged and make a new CSV file.
Actual Result	Both the CSV file gets merged and generates a new CSV file named combined_data.csv which contains both datasets.
Conclusion	Test Successful.

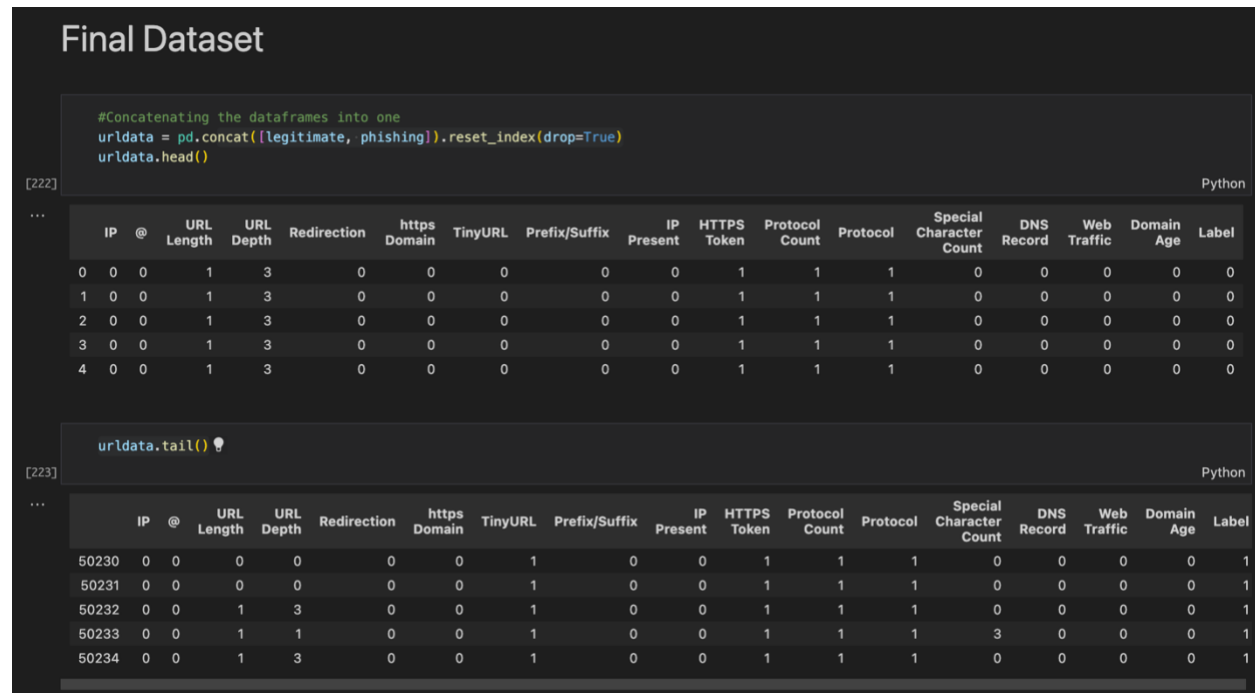
Evidence:

Figure 35: Combining both CSV files.

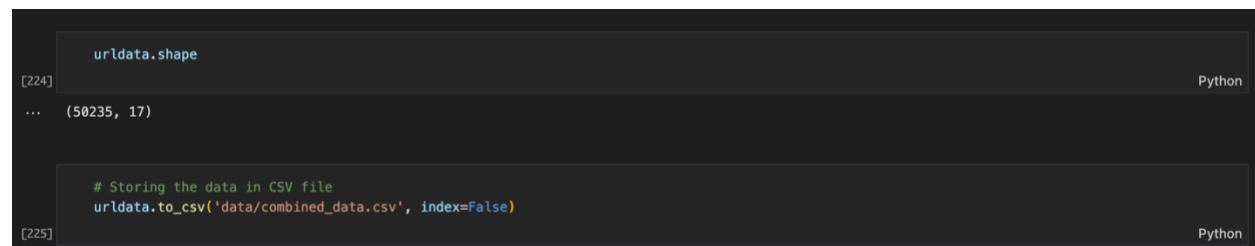


Figure 36: Saving the combined CSV file to "combined_data.csv".

```

combined_data.csv
data > combined_data.csv
1 Domain,IP,@,URL Length,URL Depth,Redirection,https,Domain,TinyURL,Prefix/Suffix,IP Present,HTTPS Token,Protocol Count,Protocol,S
2 1337x.to,0,0,1,3,0,0,0,0,0,1,1,1,0,0,0,0,0
3 1337x.to,0,0,1,3,0,0,0,0,0,1,1,1,0,0,0,0,0
4 1337x.to,0,0,1,3,0,0,0,0,0,1,1,1,0,0,0,0,0
5 1337x.to,0,0,1,3,0,0,0,0,0,1,1,1,0,0,0,0,0
6 1337x.to,0,0,1,3,0,0,0,0,0,1,1,1,0,0,0,0,0
7 1337x.to,0,0,1,3,0,0,0,0,0,1,1,1,0,0,0,0,0
8 189.cn,0,0,1,3,0,0,0,0,0,1,1,1,5,0,0,0,0
9 2gis.ru,0,0,1,7,0,0,0,0,0,1,1,1,0,0,0,0,0
10 abc.go.com,0,0,1,5,0,0,0,0,0,1,1,1,0,0,0,0,0
11 abc.go.com,0,0,1,4,0,0,0,0,0,1,1,1,3,0,0,0,0
12 abcnews.go.com,0,0,1,3,0,0,0,0,0,1,1,1,0,0,0,0,0
13 adultfriendfinder.com,0,0,1,6,0,0,0,0,0,1,1,1,2,0,0,0,0
14 akhbarelyom.com,0,0,1,5,0,0,0,0,0,1,1,1,0,0,0,0,0
15 allegro.pl,0,0,1,1,0,0,0,0,0,1,1,1,0,0,0,0,0
16 allegro.pl,0,0,1,1,0,0,0,0,0,1,1,1,2,0,0,0,0
17 allegro.pl,0,0,1,1,0,0,0,0,0,1,1,1,0,0,0,0,0
18 allegro.pl,0,0,1,1,0,0,0,0,0,1,1,1,0,0,0,0,0
19 allegro.pl,0,0,1,1,0,0,0,0,0,1,1,1,2,0,0,0,0
20 allegro.pl,0,0,1,1,0,0,0,0,0,1,1,1,2,0,0,0,0
21 allegro.pl,0,0,1,1,0,0,0,0,0,1,1,1,0,0,0,0,0
22 allegro.pl,0,0,1,1,0,0,0,0,0,1,1,1,0,0,0,0,0
23 allegro.pl,0,0,1,1,0,0,0,0,0,1,1,1,0,0,0,0,0
24 allrecipes.com,0,0,1,2,0,0,0,0,0,1,1,1,4,0,0,0,0
25 allrecipes.com,0,0,1,3,0,0,0,0,0,1,1,1,4,0,0,0,0
26 allrecipes.com,0,0,1,3,0,0,0,0,0,1,1,1,4,0,0,0,0
27 allrecipes.com,0,0,1,3,0,0,0,0,0,1,1,1,4,0,0,0,0
28 allrecipes.com,0,0,1,3,0,0,0,0,0,1,1,1,4,0,0,0,0
29 allrecipes.com,0,0,1,3,0,0,0,0,0,1,1,1,4,0,0,0,0
30 allrecipes.com,0,0,1,3,0,0,0,0,0,1,1,1,4,0,0,0,0
31 allrecipes.com,0,0,1,4,0,0,0,0,0,1,1,1,3,0,0,0,0
32 allrecipes.com,0,0,1,4,0,0,0,0,0,1,1,1,3,0,0,0,0
33 allrecipes.com,0,0,1,4,0,0,0,0,0,1,1,1,3,0,0,0,0
34 anandtech.com,0,0,1,3,0,0,0,0,0,1,1,1,0,0,0,0,0
35 ap.org,0,0,1,4,0,0,0,0,0,1,1,1,0,0,0,0,0
36 ap.org,0,0,1,4,0,0,0,0,0,1,1,1,0,0,0,0,0
37 ap.org,0,0,1,4,0,0,0,0,0,1,1,1,0,0,0,0,0
38 ap.org,0,0,1,4,0,0,0,0,0,1,1,1,0,0,0,0,0
39 ap.org,0,0,1,4,0,0,0,0,0,1,1,1,0,0,0,0,0

```

Figure 37: Combined dataset CSV file.

- **Test Case 3: To test the bulk training of the dataset.**

Table 7: Test Case 3.

Test Case 3	
Objective	To test the bulk training of the dataset.
Action	Train the “combined_data.csv” file through different ML algorithms.
Expected Result	The dataset file should get trained through different ML algorithms and display its accuracy.
Actual Result	The dataset file got trained through different ML algorithms and displayed accuracy of it.
Conclusion	Test Successful.

Evidence:

```

BULK TRAINING

models = [DecisionTreeClassifier(), RandomForestClassifier(), MLPClassifier(), SVC(), LogisticRegression(), GradientBoostingClassifier(), AdaBoostClassifier()]
for model in models:
    model.fit(x_train, y_train)
    print(str(model), ' -> train_accuracy: ', model.score(x_train, y_train))

[66] Python

... DecisionTreeClassifier() -> train_accuracy: 0.9274826527130019
RandomForestClassifier() -> train_accuracy: 0.9274826527130019
/Library/Frameworks/Python.framework/Versions/3.10/lib/python3.10/site-packages/sklearn/neural_network/_multilayer_perceptron.py:684: ConvergenceWarning:
  warnings.warn(
MLPClassifier() -> train_accuracy: 0.9183824365828689
SVC() -> train_accuracy: 0.9070356045956092
/Library/Frameworks/Python.framework/Versions/3.10/lib/python3.10/site-packages/sklearn/linear_model/_logistic.py:458: ConvergenceWarning: lbfgs fail
  STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
  https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
  https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
n_iter_i = _check_optimize_result(
LogisticRegression() -> train_accuracy: 0.8919064952792629
GradientBoostingClassifier() -> train_accuracy: 0.9124388579228757
AdaBoostClassifier() -> train_accuracy: 0.8946081219428962
  
```

Figure 38: Screenshot of bulk training.

- **Test Case 4: To test the bulk testing of the dataset.**

Table 8: Test Case 4.

Test Case 4	
Objective	To test the bulk testing of the dataset.
Action	Test the “combined_data.csv” file through different ML algorithms.
Expected Result	The dataset file should get tested through different ML algorithms and display its accuracy.
Actual Result	The dataset file got tested through different ML algorithms and displayed accuracy of it.
Conclusion	Test Successful.

Evidence:

```

BULK TESTING

models = [DecisionTreeClassifier(), RandomForestClassifier(), MLPClassifier(), SVC(), LogisticRegression(), GradientBoostingClassifier(), AdaBoostClassifier()]
for model in models:
    model.fit(x_train, y_train)
    print(str(model), ' -> test_accuracy: ', model.score(x_test, y_test))

[69] Python

... DecisionTreeClassifier() -> test_accuracy: 0.9258177957667043
RandomForestClassifier() -> test_accuracy: 0.9276093159047176
/Library/Frameworks/Python.framework/Versions/3.10/lib/python3.10/site-packages/sklearn/neural_network/multilayer_perceptron.py:684: ConvergenceWarning:
  warnings.warn(
MLPClassifier() -> test_accuracy: 0.9208413509388893
SVC() -> test_accuracy: 0.9125472762258643
/Library/Frameworks/Python.framework/Versions/3.10/lib/python3.10/site-packages/sklearn/linear_model/logistic.py:458: ConvergenceWarning: lbfgs fail
  STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
https://scikit-learn.org/stable/modules/linear\_model.html#logistic-regression
n_iter_i = _check_optimize_result(
LogisticRegression() -> test_accuracy: 0.896224537190631
GradientBoostingClassifier() -> test_accuracy: 0.9171919580651583
AdaBoostClassifier() -> test_accuracy: 0.8984805255125738

```

Figure 39: Screenshot of bulk testing.

- **Test Case 5: To test if the output of the random forest algorithm is saved in a pickle file.**

Table 9: Test Case 5.

Test Case 5	
Objective	To test if the output of the random forest algorithm is saved in a pickle file.
Action	After observing the training result saving the output from the “Random Forest Classifier” which has the highest accuracy amongst all in the pickle format.
Expected Result	The file should get saved in the “.pkl” format.
Actual Result	The file gets saved in the “.pkl” format.
Conclusion	Test Successful.

Evidence:

```
[91] forest = RandomForestClassifier()
    forest.fit(x_train, y_train)
Python

...
▼ RandomForestClassifier
RandomForestClassifier()

# save Random Forest model to file
import pickle
pickle.dump(forest, open("RandomForestClassifier.pickle.dat", "wb"))
[92]
Python

# load model from file
loaded_model = pickle.load(open("RandomForestClassifier.pickle.dat", "rb"))
loaded_model
[93]
Python

...
▼ RandomForestClassifier
RandomForestClassifier()

with open('classifier_model.pkl', 'wb') as f:
    pickle.dump(forest, f)
[94]
Python
```

Figure 40: Code for saving the model into pickle file.

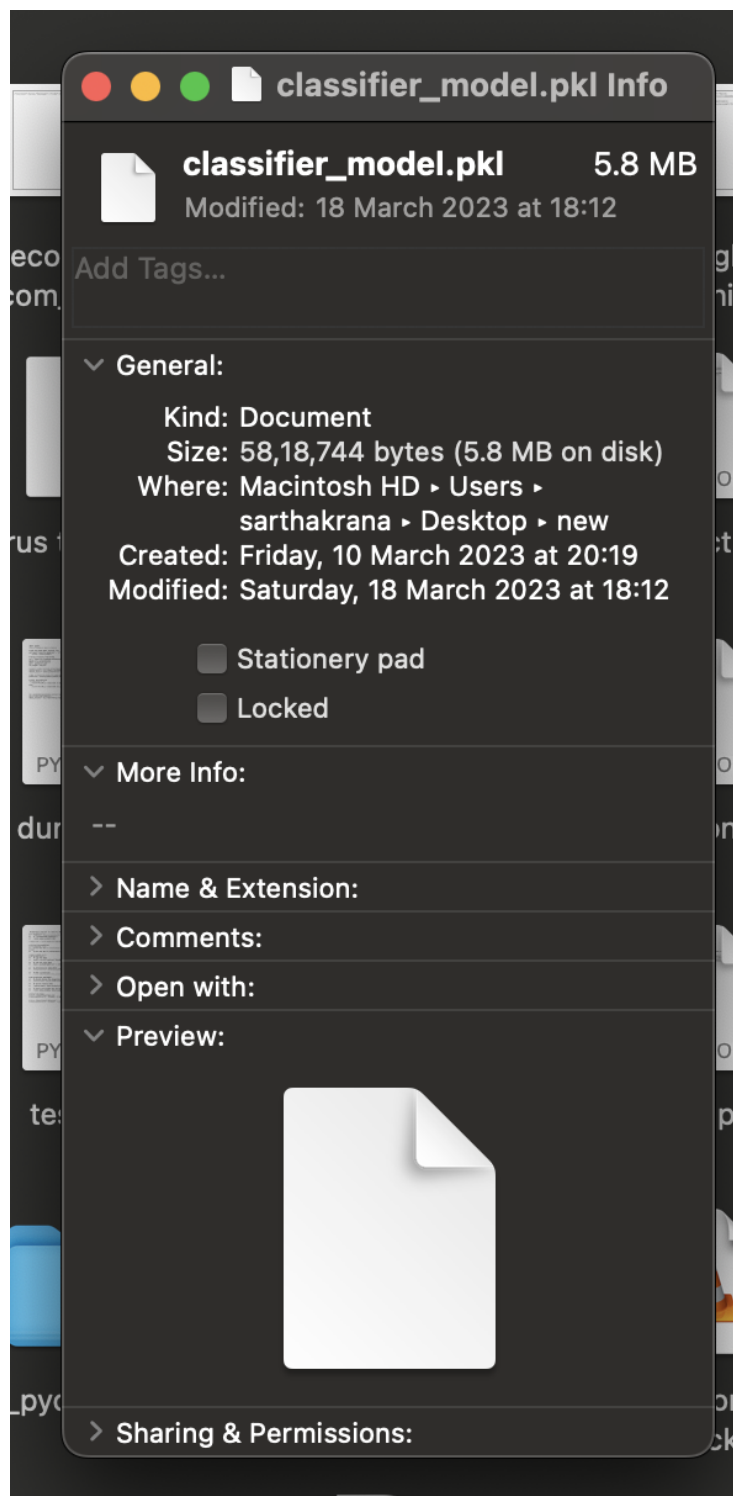


Figure 41: Screenshot of the classifier_model.pkl file.

- **Test Case 6: To test if the system is detecting phishing URL.**

Table 10: Test Case 6.

Test Case 6	
Objective	To test if the system is detecting phishing URL.
Action	Input a phishing URL into the system.
Expected Result	The model should predict the URL as phishing.
Actual Result	The model predicted the URL as phishing.
Conclusion	Test Successful.

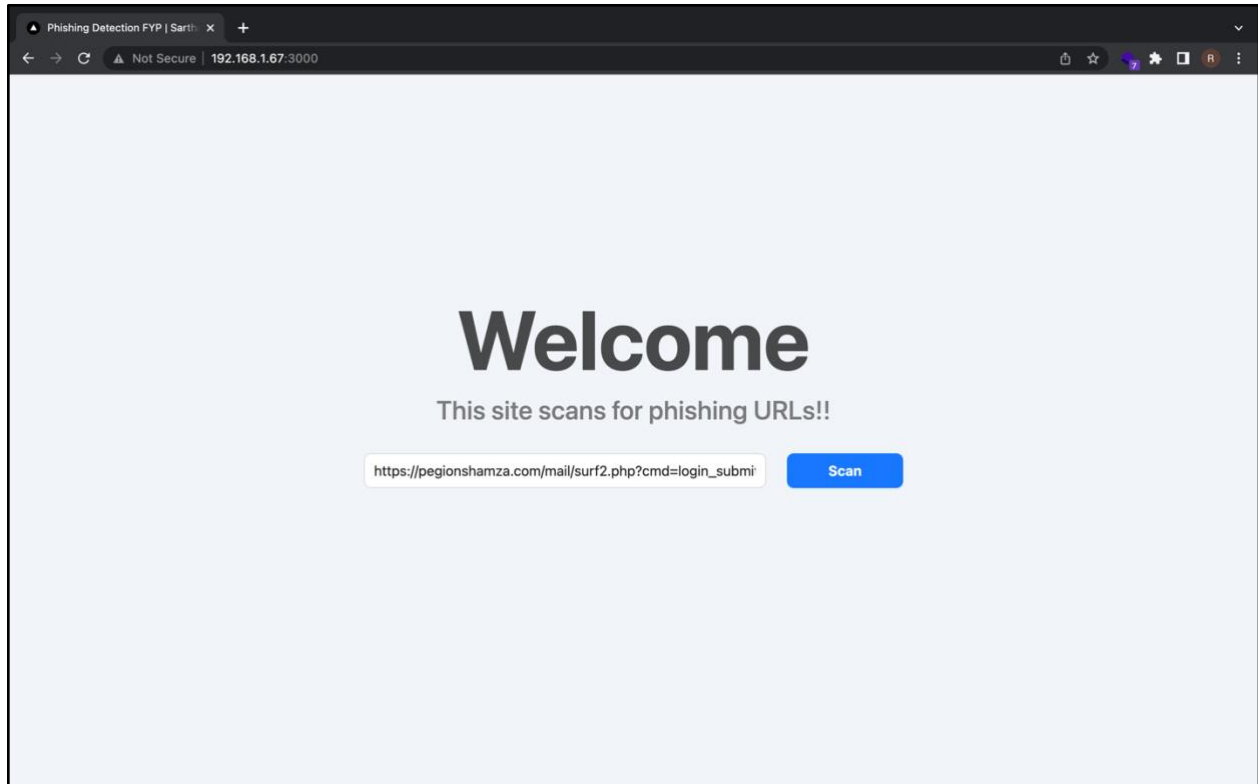
Evidence:

Figure 42: Providing a phishing URL to scan.

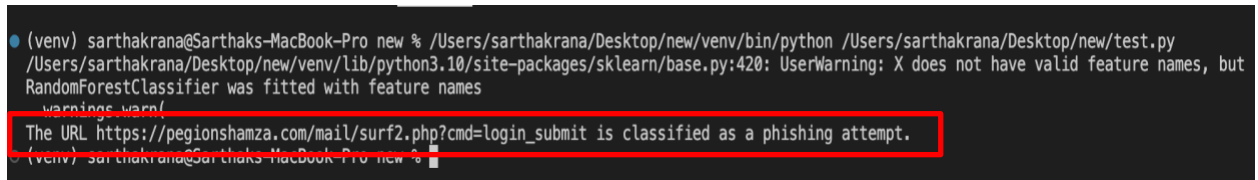


Figure 43: System displaying the URL as phishing.

- **Test Case 7: To test if the system is detecting legitimate URL.**

Table 11: Test Case 7.

Test Case 7	
Objective	To test if the system is detecting legitimate URL.
Action	Input a legitimate URL into the system.
Expected Result	The model should predict the URL as phishing.
Actual Result	The model predicted the URL as phishing.
Conclusion	Test Successful.

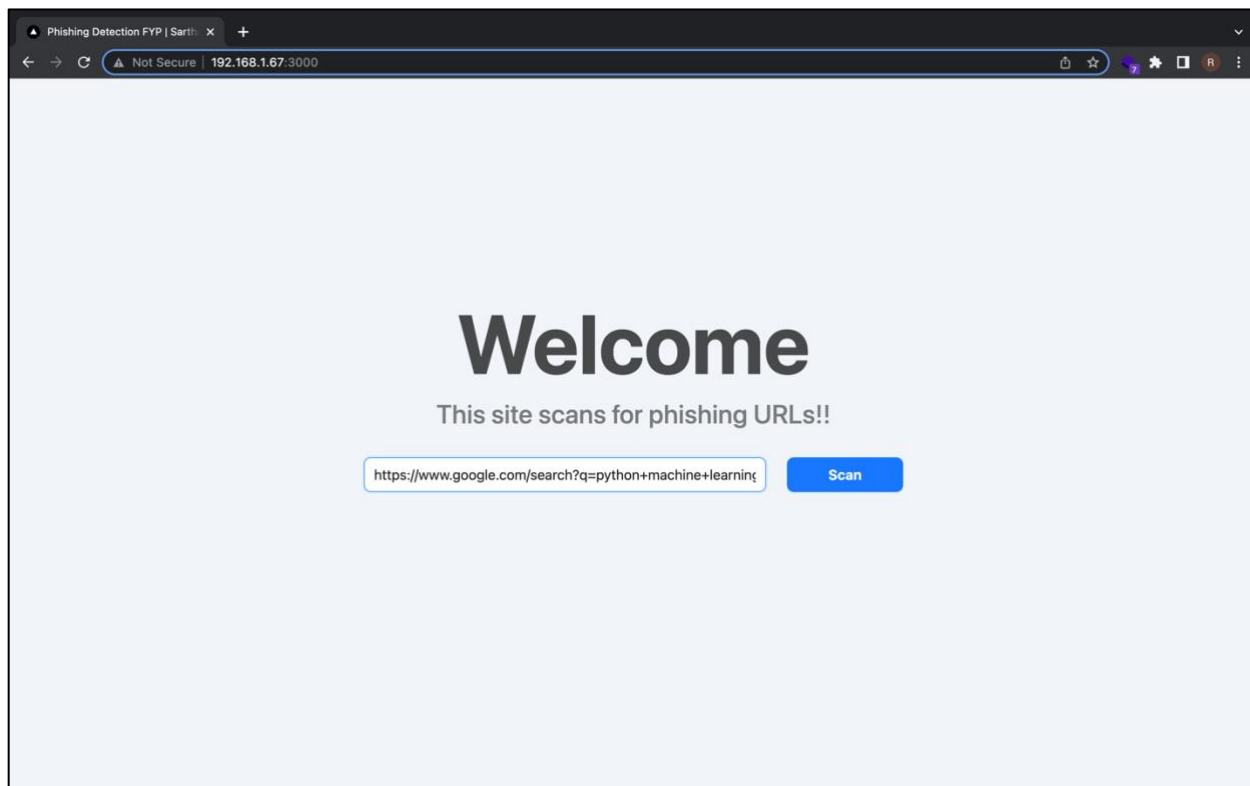
Evidence:

Figure 44: Providing a legitimate URL to scan.

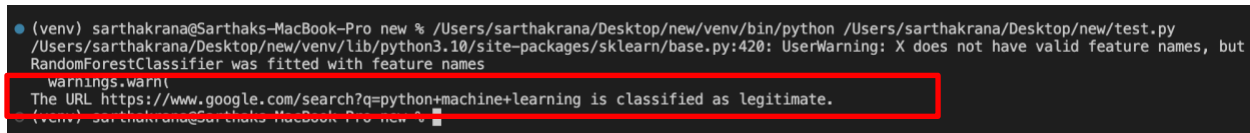


Figure 45: System displaying the URL as legitimate.

- **Test Case 8: To test if the search bar has URL validation or not.**

Table 12: Test Case 8.

Test Case 8	
Objective	To test if the search bar has URL validation or not.
Action	Input a word except for an URL in the search bar.
Expected Result	The search bar should display a pop-up as “The URL doesn’t exists”.
Actual Result	The search bar displayed a pop-up as “The URL doesn’t exists”.
Conclusion	Test Successful.

Evidence:

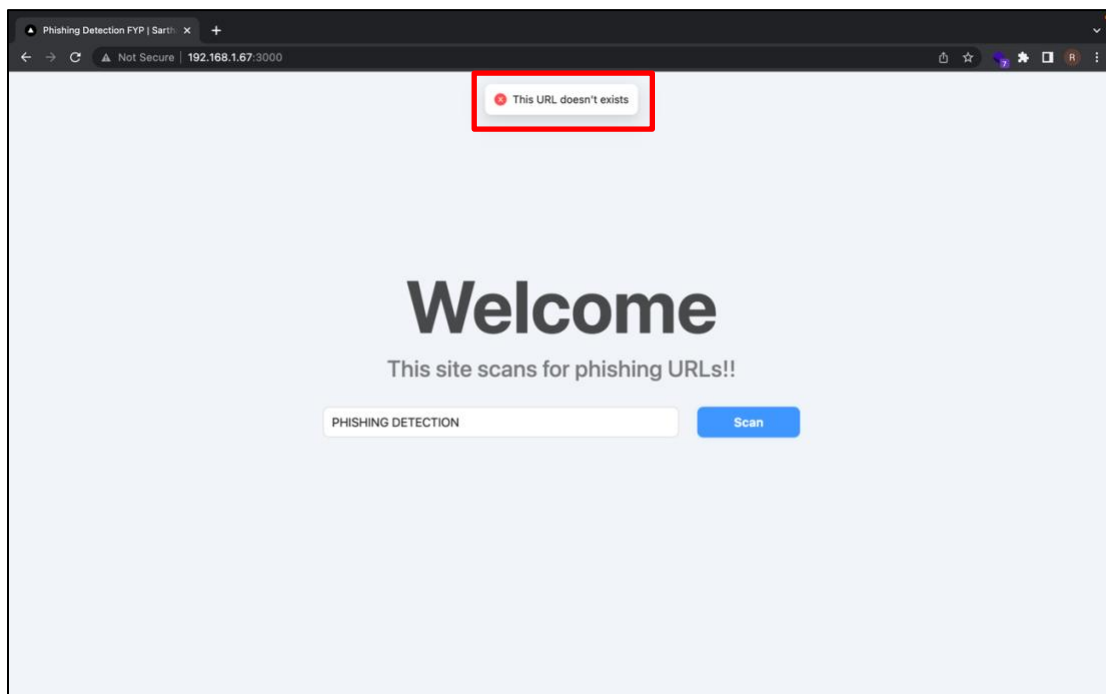


Figure 46: A pop-up showing the URL doesn't exist.

- **Test Case 9: To test if the Virus total API is responding or not.**

Table 13: Test Case 9.

Test Case 9	
Objective	To test if the Virus total API is responding or not.
Action	Pass an URL to test the response of the API.
Expected Result	Both legitimate.csv and phishing.csv should get merged and make a new CSV file.
Actual Result	Both the CSV file gets merged and generates a new CSV file named combined_data.csv which contains both datasets.
Conclusion	Test Successful.

Evidence:

```

API.py > ...
1  import requests
2  import base64
3
4  api = "https://www.virustotal.com/api/v3/urls/"
5  enter_URL = "http://facebook.com"
6  url_id = base64.urlsafe_b64encode(enter_URL.encode()).decode().strip("=")
7  print (url_id)
8
9  url = api + url_id
10
11  headers = {
12      "accept": "application/json",
13      "x-apikey": "6da45fde1f9d2d396f1572e545ee8b8dbcdf6020ab1622d3a986c6a77c75a1b2"
14  }
15
16  response = requests.get(url, headers=headers)
17
18  print(response.text)

```

Figure 47: Passing a URL to check the response of the virus total API.

```

• (venv) sarthakrana@Sarthaks-MacBook-Pro new % /Users/sarthakrana/Desktop/new/venv/bin/python /Users/sarthakrana/Desktop/new/API.py
aHR0cDovL2ZhY2Vib29rLmNvbQ
{
  "data": {
    "attributes": {
      "last_modification_date": 1681231167,
      "times_submitted": 62116,
      "total_votes": {
        "harmless": 621,
        "malicious": 584
      },
      "threat_names": [],
      "redirection_chain": [
        "http://facebook.com/",
        "https://facebook.com/"
      ],
      "last_submission_date": 1681230801,
      "last_http_response_content_length": 34534,
      "last_http_response_headers": {
        "X-XSS-Protection": "0",
        "Transfer-Encoding": "chunked",
        "Strict-Transport-Security": "max-age=15552000; preload; includeSubDomains",
        "Connection": "keep-alive",
        "report-to": "{\n\"max_age\":86400,\n\"endpoints\": [{\n\"url\":\n\"https://\\www.facebook.com\\browser_reporting\\?minimize=0\\\"}
      ],\n\"group\":\n\"coep_report\\\", {\n\"max_age\":259200,\n\"endpoints\": [{\n\"url\":\n\"https://\\m.facebook.com\\ajax\\mtouch_error_reports\\\"}]}
    }
  }
}

```

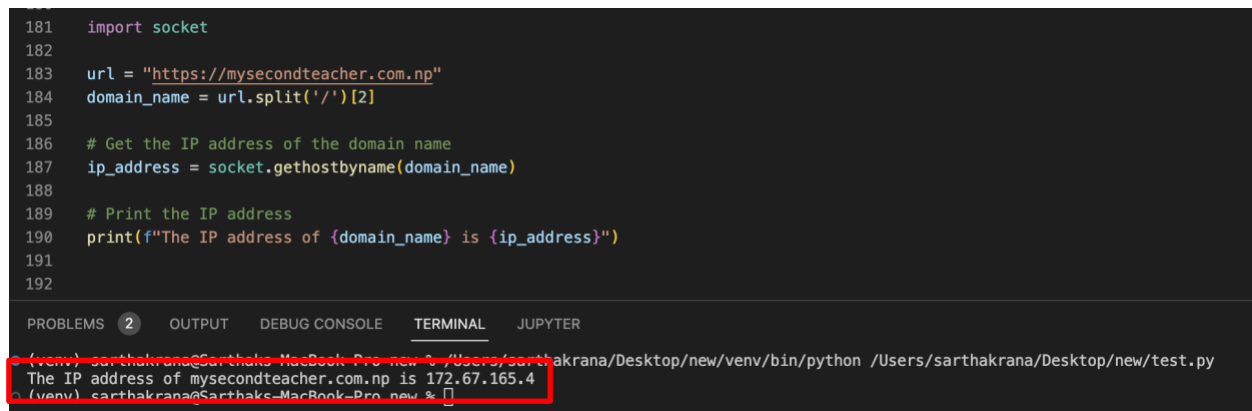
Figure 48: Response from the API.

- **Test Case 10: To test if the Python code is generating a URL into an IP address or not.**

Table 14: Test Case 10.

Test Case 10	
Objective	To test if the Python code is generating a URL into an IP address or not.
Action	Pass an URL to the Python code to check if it is converting it into an IP address or not.
Expected Result	The code should convert the URL into an IP address.
Actual Result	The code converted the URL into an IP address.
Conclusion	Test Successful.

Evidence:



```

181 import socket
182
183 url = "https://mysecondteacher.com.np"
184 domain_name = url.split('/')[2]
185
186 # Get the IP address of the domain name
187 ip_address = socket.gethostbyname(domain_name)
188
189 # Print the IP address
190 print(f"The IP address of {domain_name} is {ip_address}")
191
192

```

PROBLEMS 2 OUTPUT DEBUG CONSOLE TERMINAL JUPYTER

(venv) sarthakrana@Sarthaks-MacBook-Pro: new % /Users/sarthakrana/Desktop/new/venv/bin/python /Users/sarthakrana/Desktop/new/test.py

The IP address of mysecondteacher.com.np is 172.67.165.4

(venv) sarthakrana@Sarthaks-MacBook-Pro: new %

Figure 49: Screenshot of the code converting the URL into IP address.

- **Test Case 11: To test if the code is generating longitude and latitude or not.**

Table 15: Test Case 11.

Test Case 11	
Objective	To test if the code is generating longitude and latitude or not.
Action	Pass an URL to check if it is converting it into longitude and latitude or not.
Expected Result	The code should convert the URL into longitude and latitude.
Actual Result	The code converted the URL into longitude and latitude.
Conclusion	Test Successful.

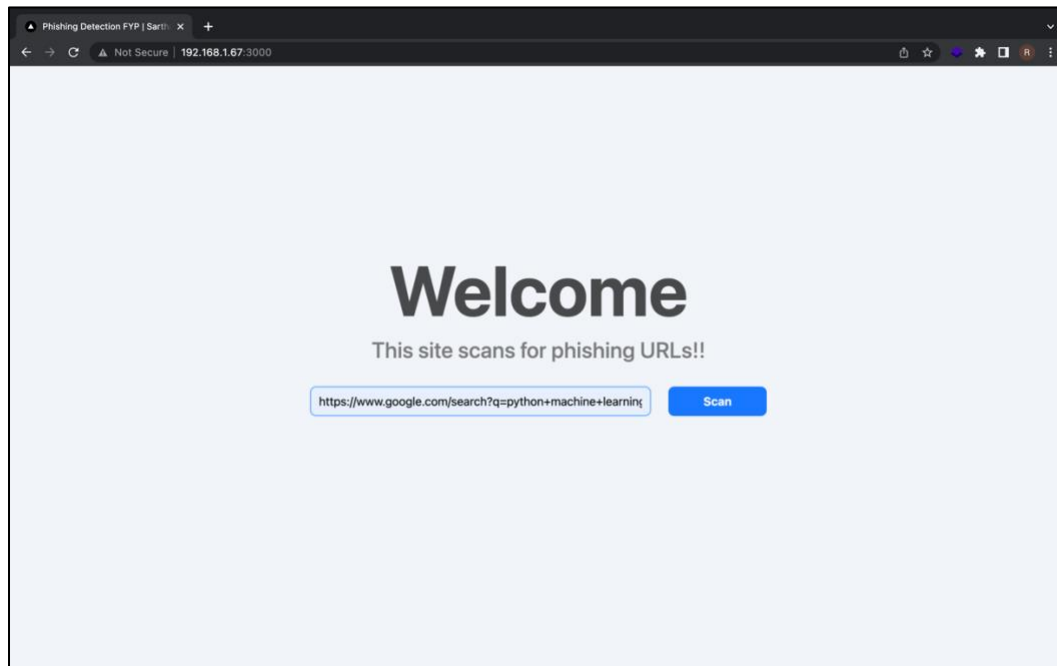
Evidence:

Figure 50: Providing an URL to scan for longitude and latitude.

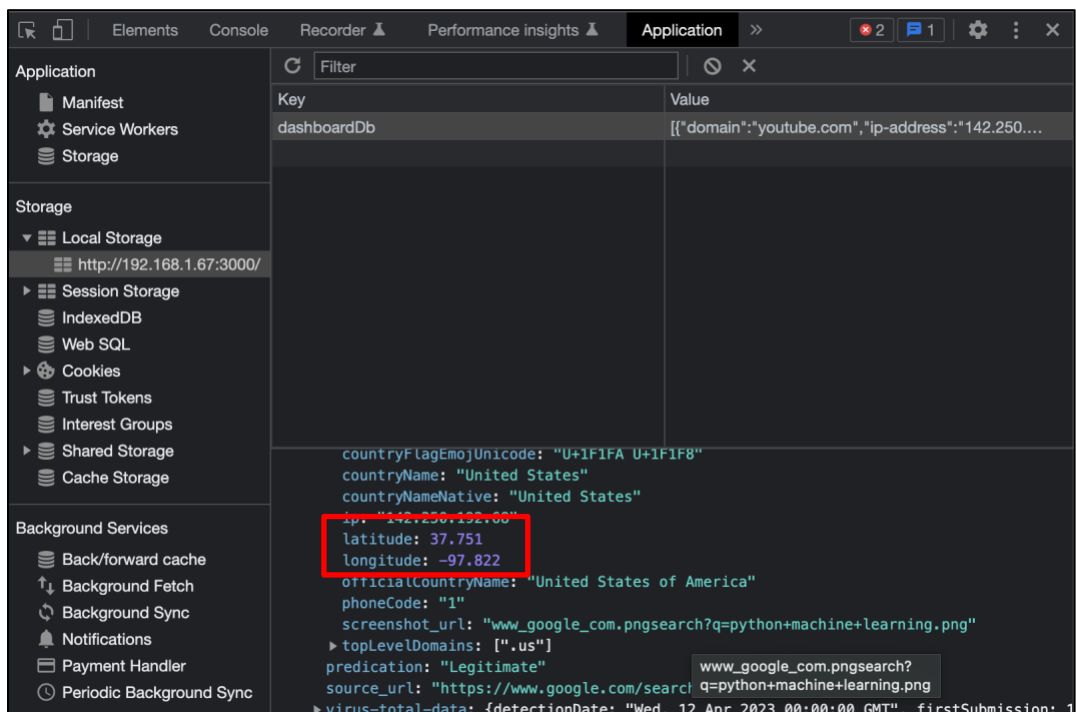


Figure 51: Screenshot of longitude and latitude as response.

- **Test Case 12: To test if the google maps API is generating maps or not.**

Table 16: Test Case 12.

Test Case 12	
Objective	To test if the google maps API is generating map or not.
Action	Pass an URL to check if the API generates a map.
Expected Result	The code should generate a map based on the hosting IP address.
Actual Result	The code generates a map based on the hosting IP address.
Conclusion	Test Successful.

Evidence:

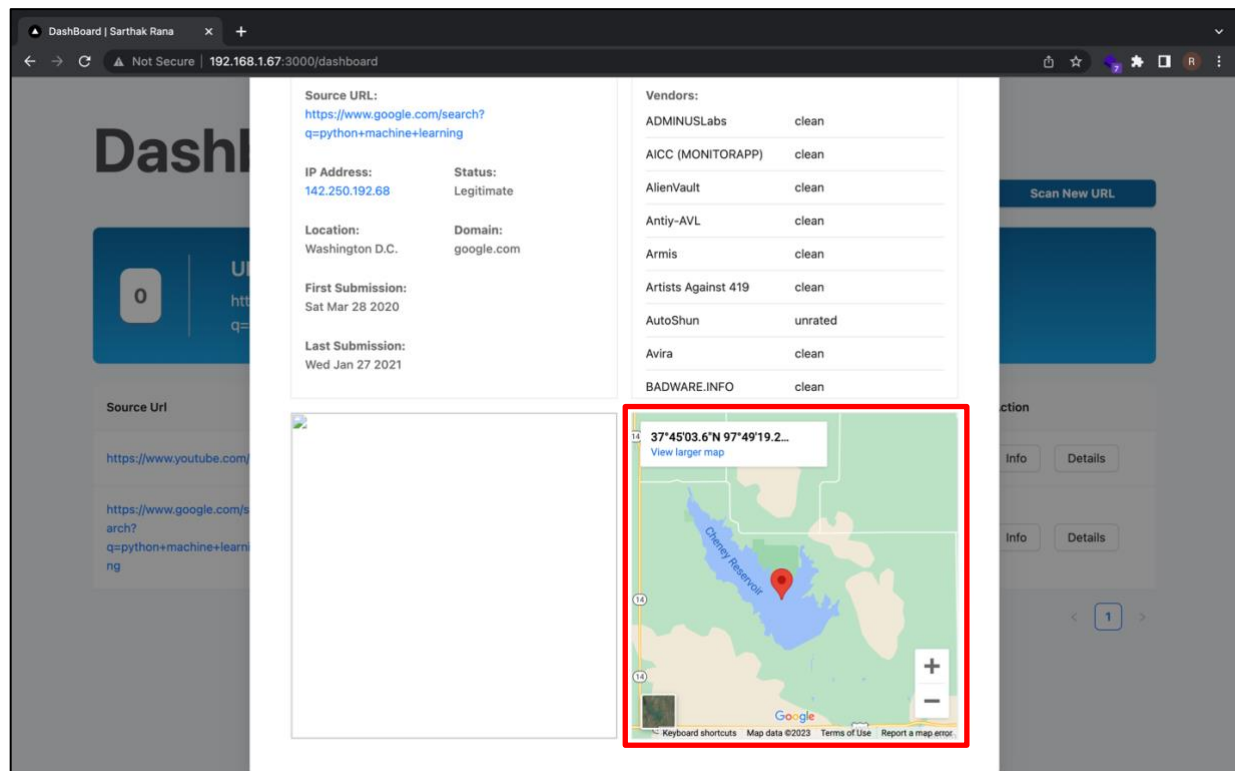


Figure 52: Screenshot of Map in the dashboard.

- **Test Case 13: To test if the Python code is generating screenshots of the web page or not.**

Table 17: Test Case 13.

Test Case 13	
Objective	To test if the Python code is generating screenshots of the web page or not.
Action	Pass an URL to check if the code generates a screenshot.
Expected Result	The code should generate a screenshot and save it in a folder under the name of the URL.
Actual Result	The code generates a screenshot and saves it in a folder under the name of the URL.
Conclusion	Test Successful.

Evidence:

```
from selenium import webdriver
from selenium.webdriver.chrome.options import Options

#Remove protocol and path from URL to get domain name
url = "https://mysecondteacher.com.np"
url = url + "/"

url_for_screenshot = url.replace("https://", "")
url_for_screenshot = url_for_screenshot.replace(".", "_")
url_for_screenshot = url_for_screenshot.replace("/", ".png")
print(url_for_screenshot)
# Configure options for the webdriver
options = Options()
options.headless = True # Run the browser in headless mode

# Create a new instance of the webdriver
driver = webdriver.Chrome(options=options)
# Load the URL
driver.get(url)
driver.save_screenshot(url_for_screenshot)

# Close the webdriver
driver.quit()
```

Figure 53: Screenshot of the code for generating a screenshot.

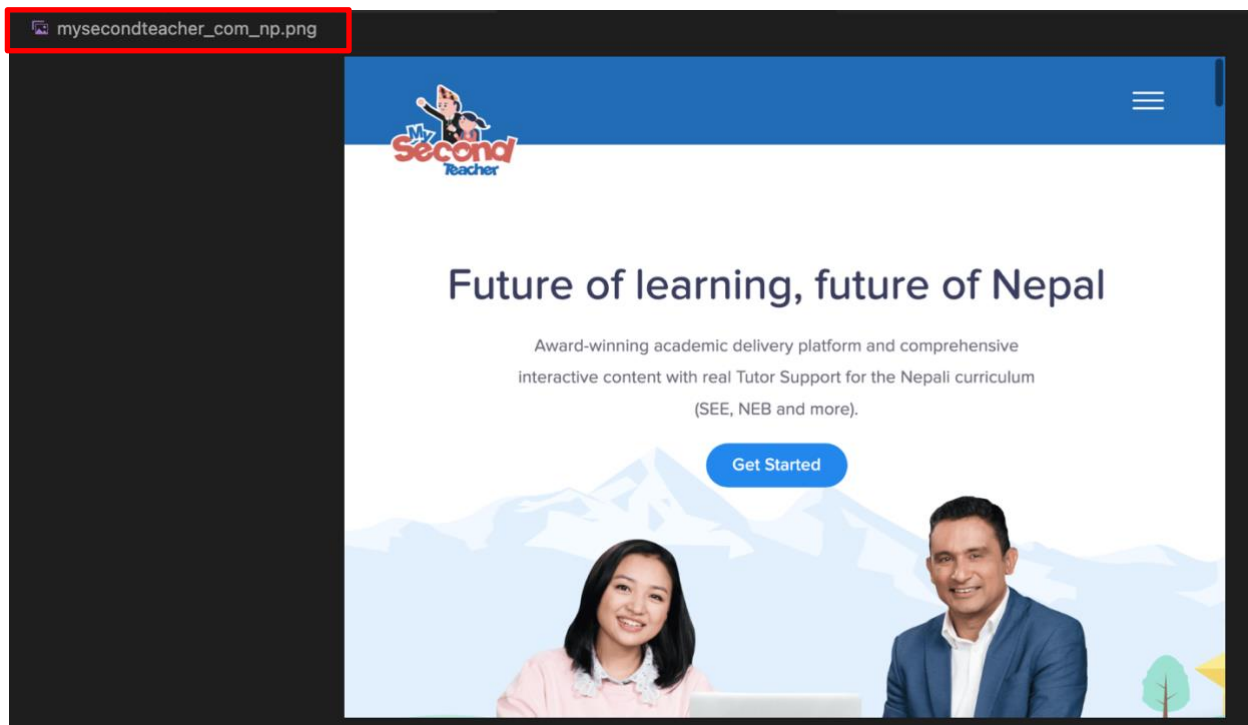


Figure 54: Screenshot of the webpage.

- **Test Case 14: To test if the generated screenshot is displaying in the front end or not.**

Table 18: Test Case 14.

Test Case 14	
Objective	To test if the generated screenshot is displaying in the front end or not.
Action	Check whether the screenshot of the webpage is displayed on the front end.
Expected Result	The screenshot should be displayed on one of the division on the front end.
Actual Result	The screenshot gets displayed on one of the divisions on the front end.
Conclusion	Test Successful.

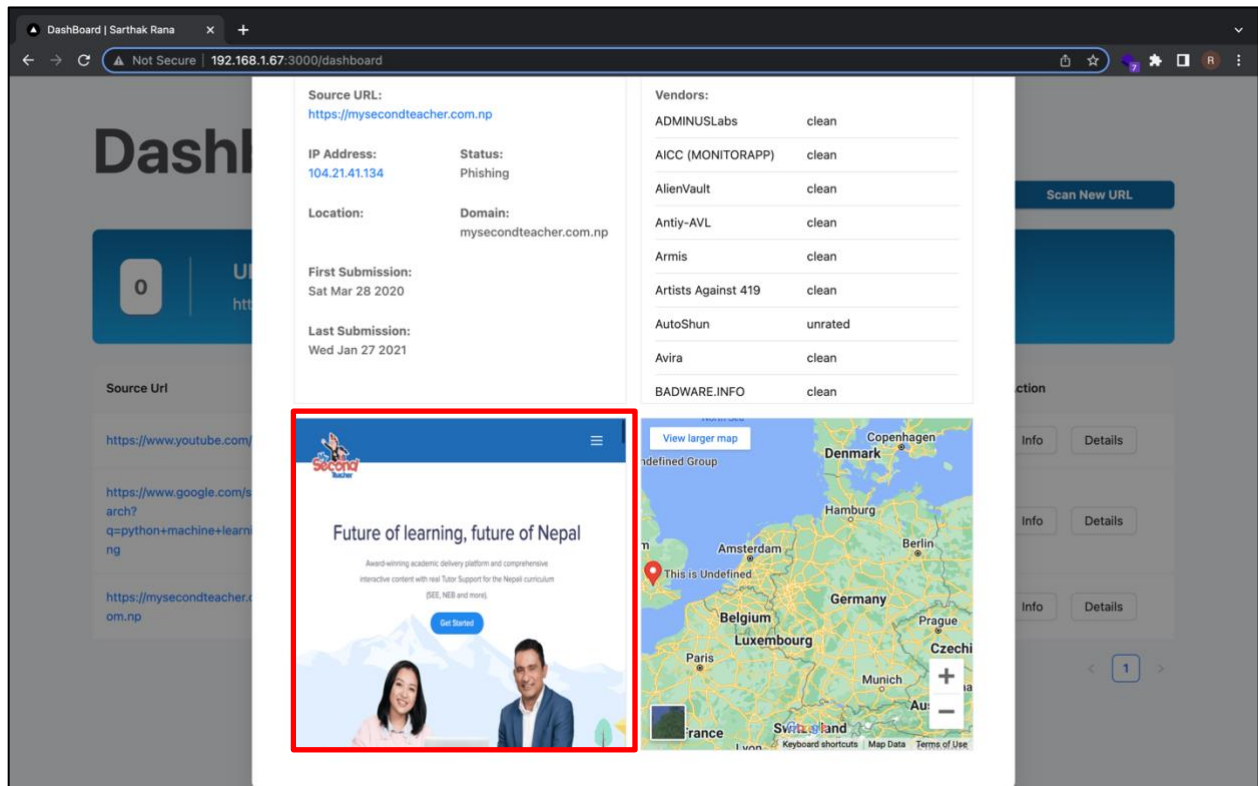
Evidence:

Figure 55: Screenshot of the webpage getting displayed on the front end.

- **Test Case 15: To test if the Python code is generating domain name of a URL.**

Table 19: Test Case 15.

Test Case 15	
Objective	To test if the Python code is generating domain name of a URL or not.
Action	Provide an URL as input to the system to check the domain name of it.
Expected Result	The code should generate the domain name of the URL and display it on the frontend of the dashboard.
Actual Result	The code generates the domain name of the URL and displays it on the frontend of the dashboard.
Conclusion	Test Successful.

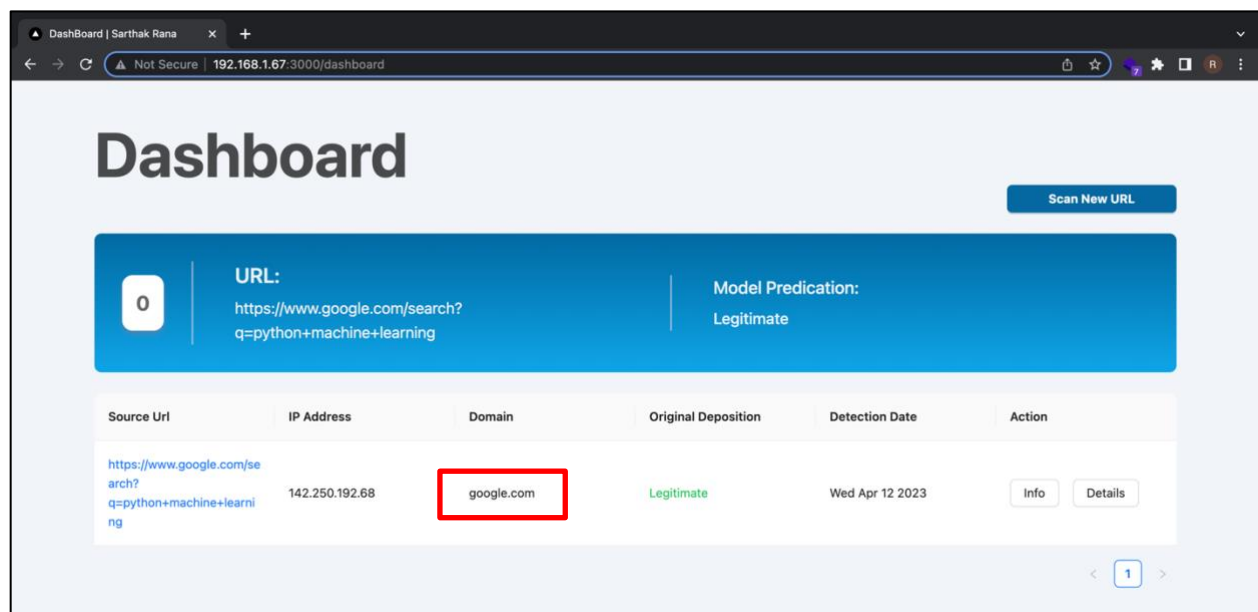
Evidence:

Figure 56: Screenshot of Domian being displayed on the front end.

- **Test Case 16: To test if the dashboard is displaying the results of all the antivirus vendors or not.**

Table 20: Test Case 16.

Test Case 16	
Objective	To test if the dashboard is displaying the results of all the antivirus vendors or not.
Action	Click on the details button to see the results of the antivirus vendors.
Expected Result	The vendor results should be displayed on one of the divisions on the front end.
Actual Result	The vendor results got displayed on one of the divisions on the front end.
Conclusion	Test Successful.

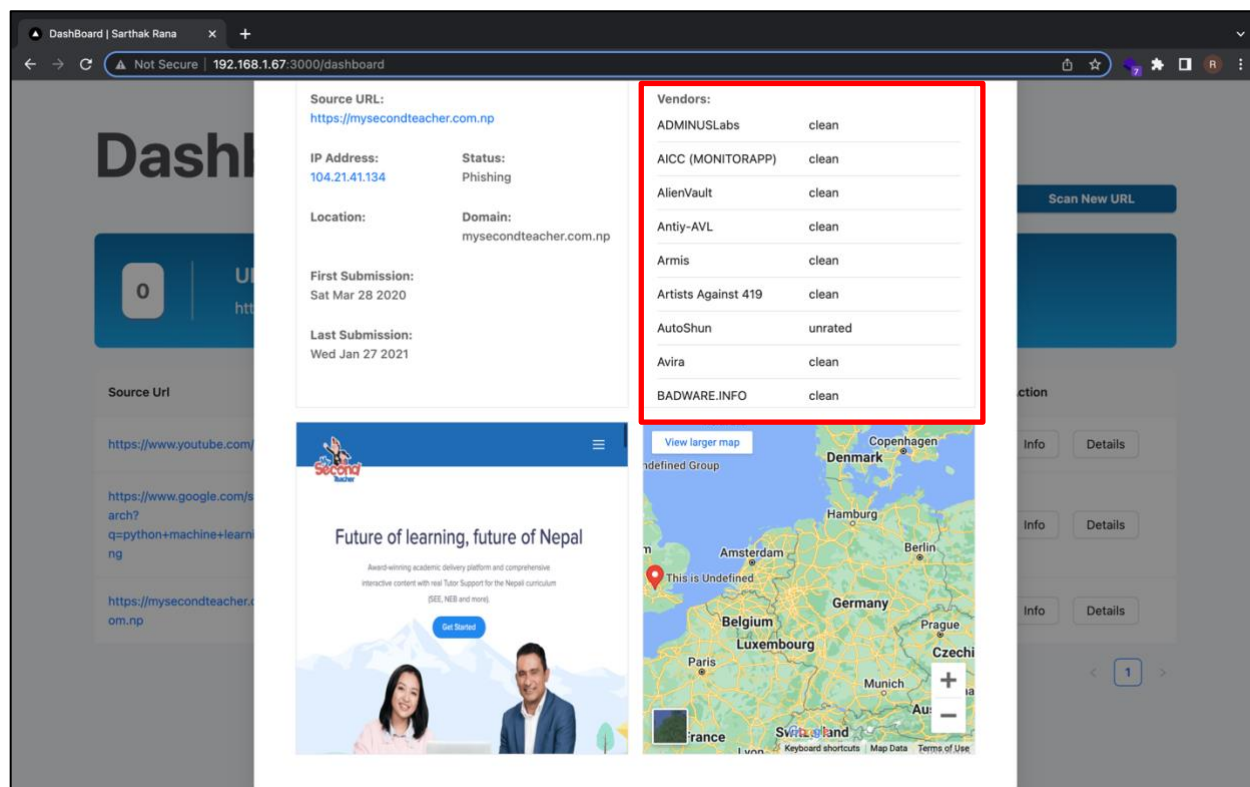
Evidence:

Figure 57: Screenshot of vendor results on the frontend.

- **Test Case 17:** To test if the dashboard is displaying the count of the number of vendors detecting the URL as phishing.

Table 21: Test Case 17.

Test Case 17	
Objective	To test if the dashboard is displaying the count of the number of vendors detecting the URL as phishing.
Action	Enter a phishing URL into the system and check the counter on the dashboard.
Expected Result	The counter should display the number of vendors who detected the URL as phishing.
Actual Result	The counter displayed the number of vendors who detected the URL as phishing.
Conclusion	Test Successful.

Evidence:

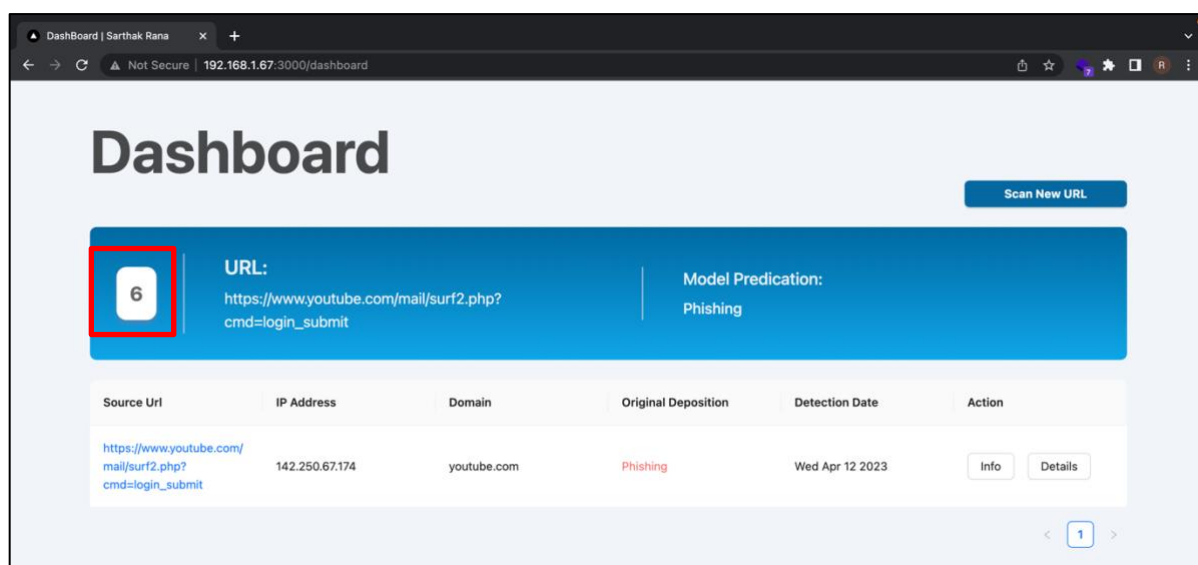


Figure 58: Screenshot of the counter displaying the number of vendors detecting the URL as phishing.

- **Test Case 18:** To test if the detection date label in the dashboard is displaying the correct date or not.

Table 22: Test Case 18.

Test Case 18	
Objective	To test if the detection date label in the dashboard is displaying the correct date or not.
Action	Enter an URL to scan and check whether the detection date matches the current date.
Expected Result	The detection date should match the date when the URL was scanned.
Actual Result	The detection date matches the date when the URL was scanned.
Conclusion	Test Successful.

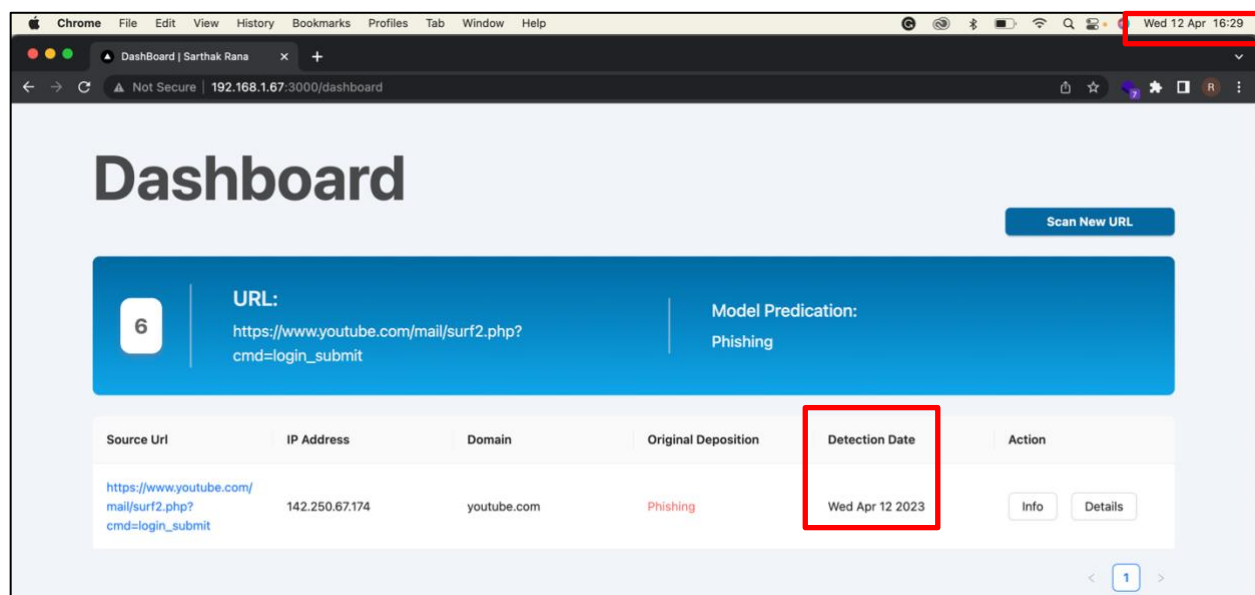
Evidence:

Figure 59: Screenshot of the detection date label displaying the correct date.

- **Test Case 19:** To test if the logs of the URL are stored or not after reloading the page.

Table 23: Test Case 19.

Test Case 19	
Objective	To test if the logs of the URL are stored or not after reloading the page.
Action	Reload the page to check the stored logs.
Expected Result	The previously scanned URL logs should get stored on the local storage of the browser.
Actual Result	The previously scanned URL logs got stored on the local storage of the browser.
Conclusion	Test Successful.

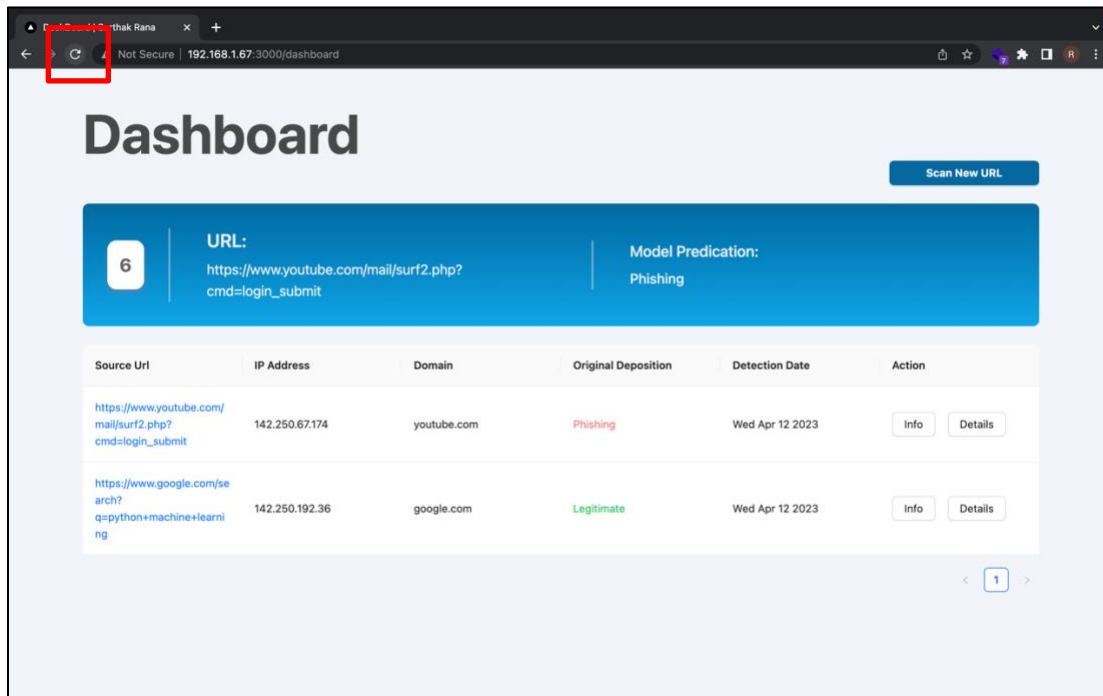
Evidence:

Figure 60: Screenshot of reloading the page.

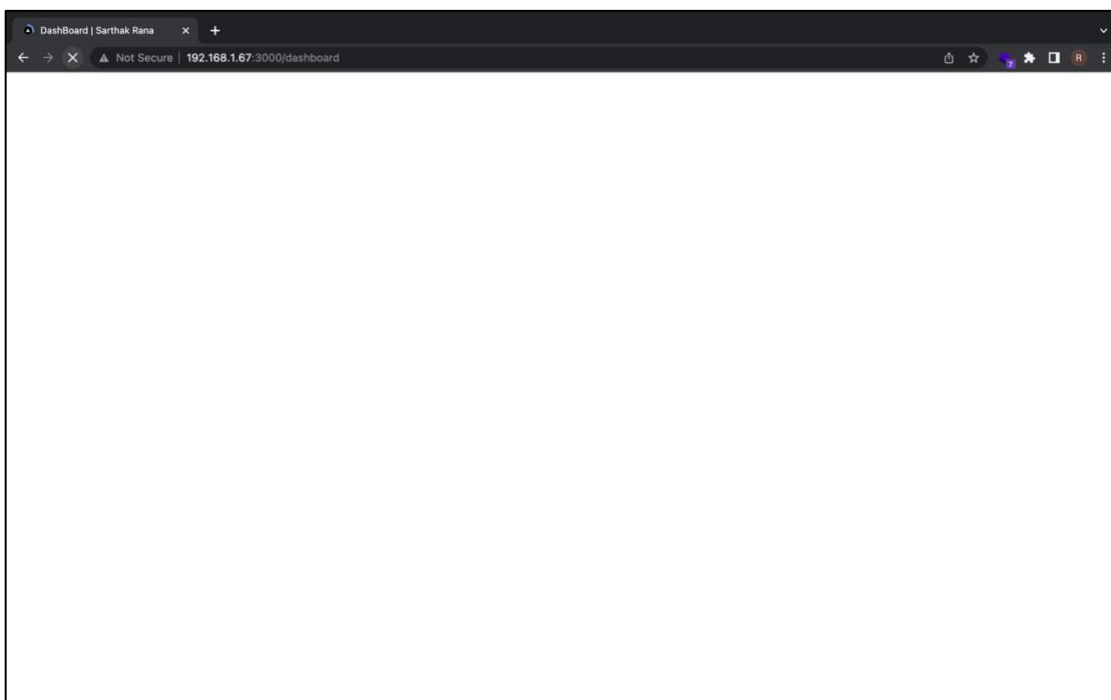


Figure 61: Screenshot of the page being reloaded.

Dashboard

Scan New URL

6

URL:
https://www.youtube.com/mail/surf2.php?cmd=login_submit

Model Predication:
Phishing

Source Url	IP Address	Domain	Original Deposition	Detection Date	Action
https://www.youtube.com/mail/surf2.php?cmd=login_submit	142.250.67.174	youtube.com	Phishing	Wed Apr 12 2023	Info Details
https://www.google.com/search?q=python+machine+learning	142.250.192.36	google.com	Legitimate	Wed Apr 12 2023	Info Details

< 1 >

Figure 62: Screenshot of the stored logs after reloading the page.

- **Test Case 20: To test if the system is running on different devices on the same network or not.**

Table 24: Test Case 20.

Test Case 20	
Objective	To test if the system is running on different devices on the same network or not.
Action	Type the hosted IP address on the different device inside the same network.
Expected Result	The system should run on the different device as well.
Actual Result	The system runs on the different device.
Conclusion	Test Successful.

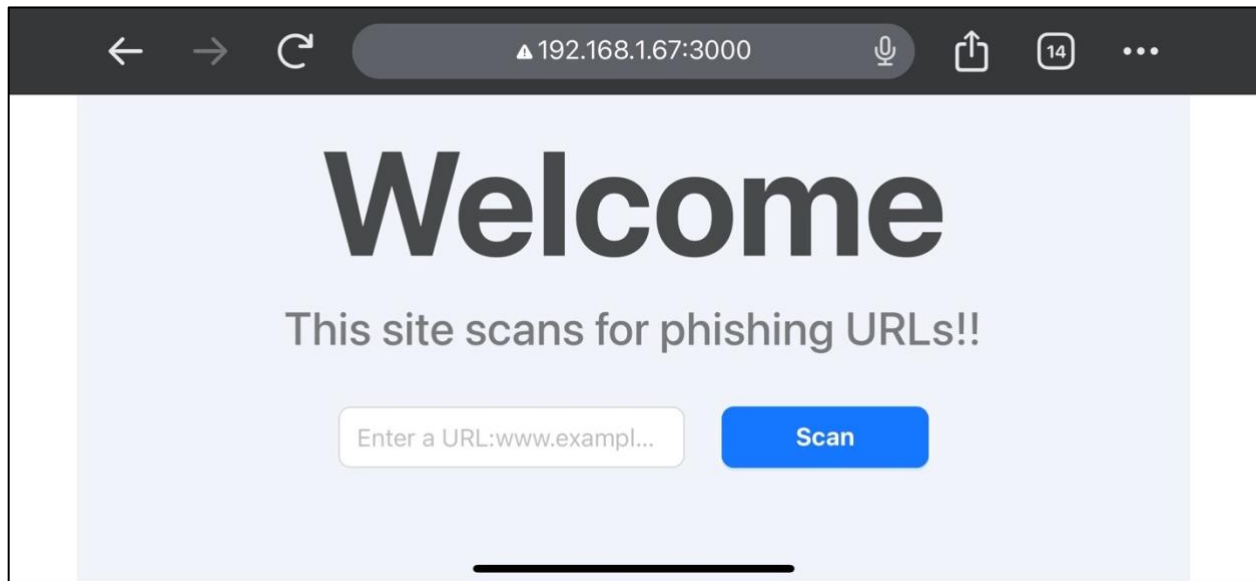
Evidence:

Figure 63: Screenshot of the system running on a different device inside the same network (a).

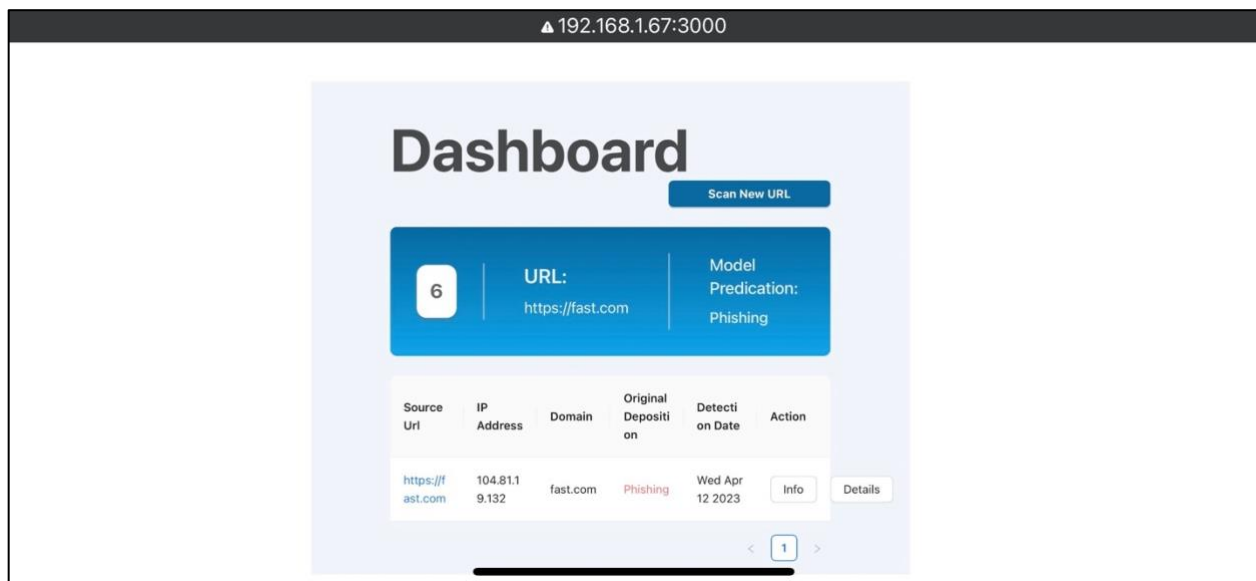


Figure 64: Screenshot of the system running on a different device inside the same network (b).

- **Test Case 21: To check the functionality of the 'Info' button.**

Table 25: Test Case 21.

Test Case 21	
Objective	To check the functionality of the 'Info' button.
Action	Click on the 'Info' button.
Expected Result	The values of the phishing counter, URL, and Model Prediction should change corresponding to the log of the scanned results.
Actual Result	The values of the phishing counter, URL, and Model Prediction changed corresponding to the log of the scanned results.
Conclusion	Test Successful.

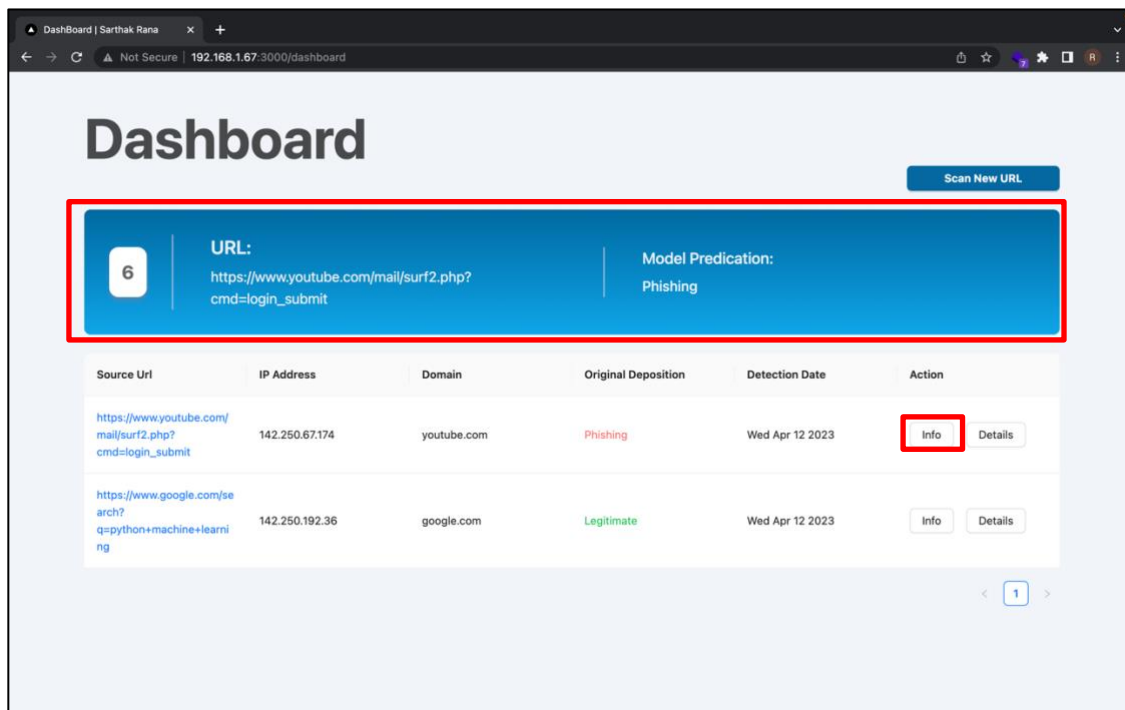
Evidence:

Figure 65: shot while checking the functionality of the info button (a).

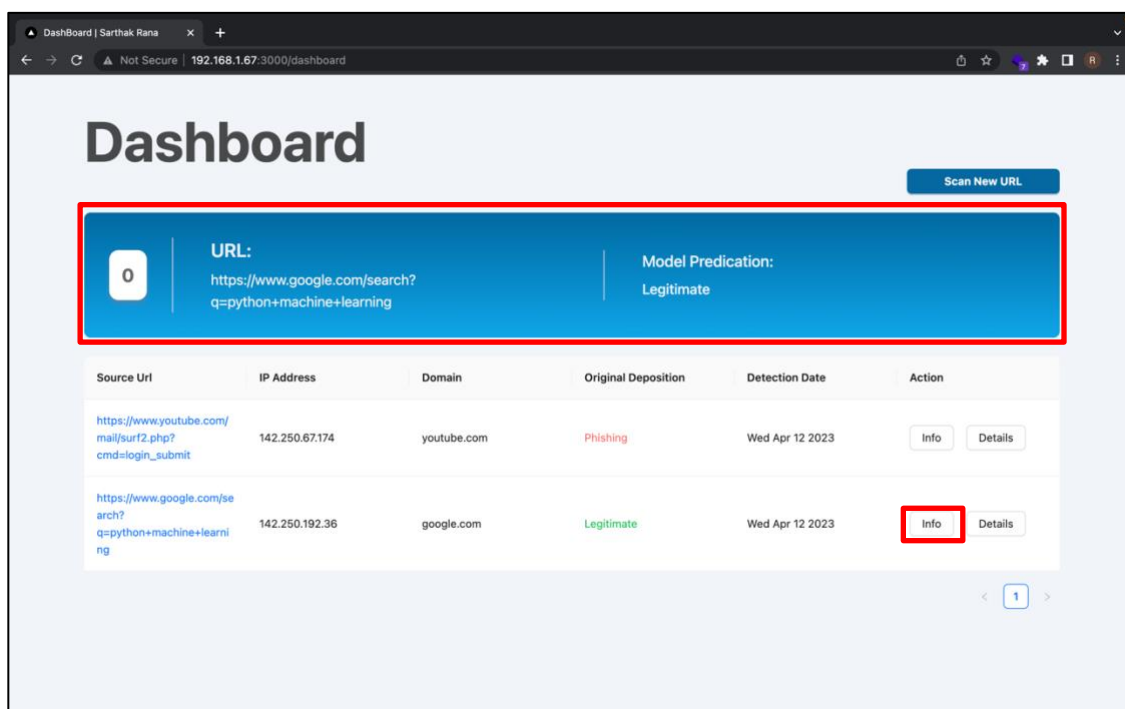


Figure 66: shot while checking the functionality of the info button (b).

- **Test Case 22: To check the functionality of the 'Details' button.**

Table 26: Test Case 22.

Test Case 22	
Objective	To check the functionality of the 'Details' button.
Action	Click on the 'Details' button.
Expected Result	A pop-up window should appear displaying additional information about the scan, vendor results, a screenshot of the page, and a map.
Actual Result	A pop-up window appears displaying additional information about the scan, vendor results, a screenshot of the page, and a map.
Conclusion	Test Successful.

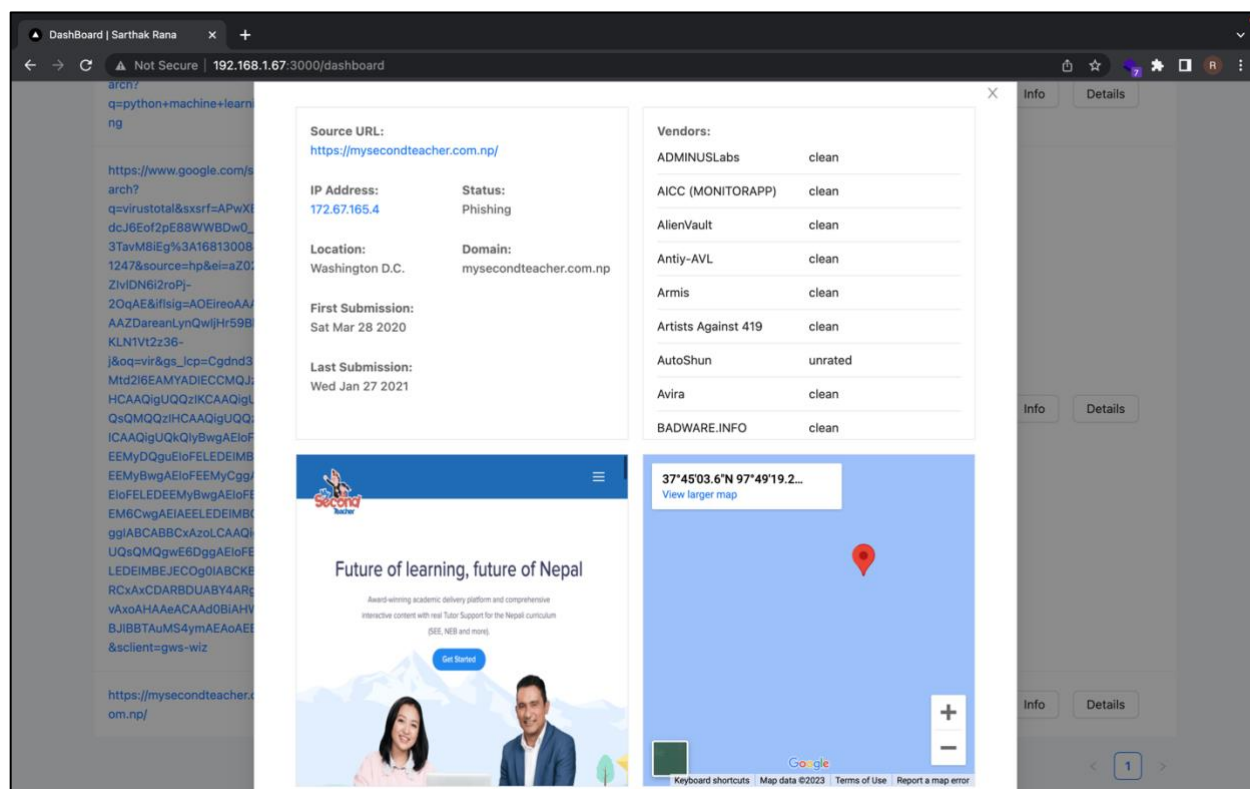
Evidence:

Figure 67: Screenshot of the pop-up window inside the details button.

4.3 System Testing

Table 27: System Test Case 1.

System Test Case 1	
Objective	To test the entire system.
Action	Enter a phishing URL to scan.
Expected Result	The system should scan the URL and display the detailed result on the dashboard.
Actual Result	The system successfully scans the URL and displays the information on the dashboard.
Conclusion	Test Successful.

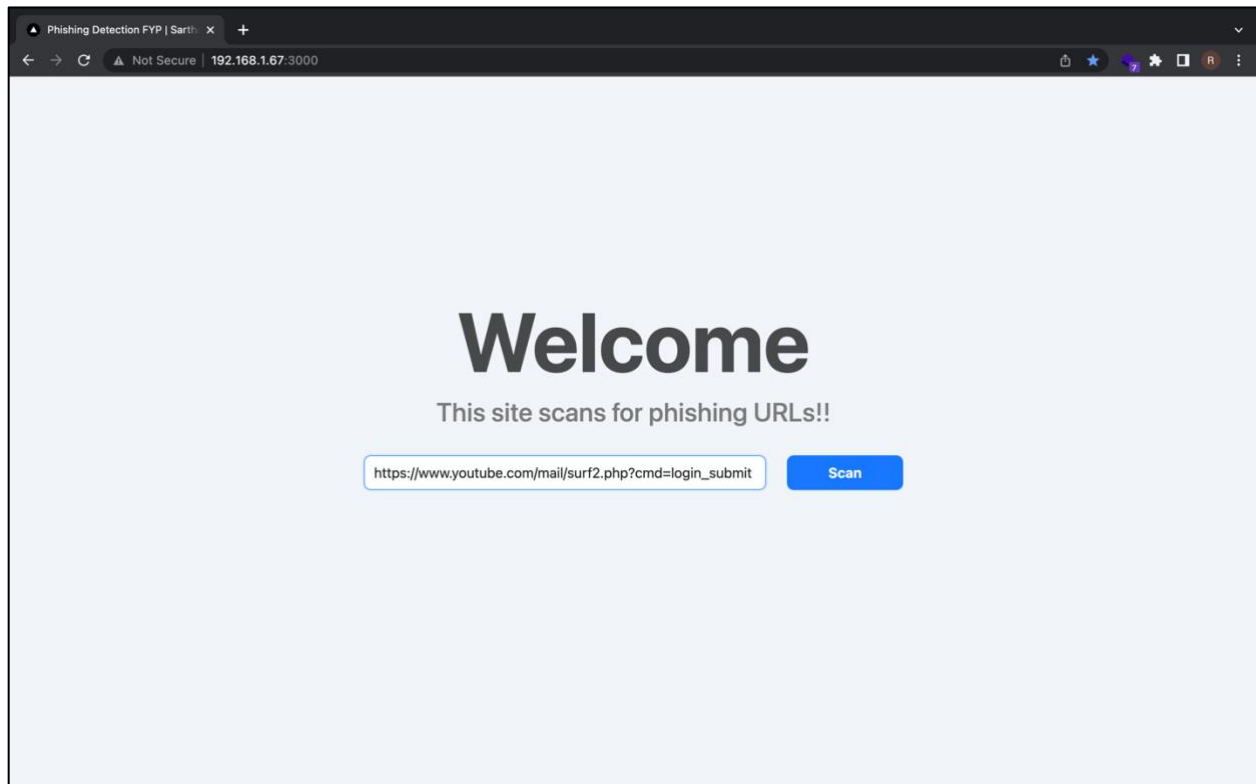
Evidence:

Figure 68: Providing a phishing URL to scan.

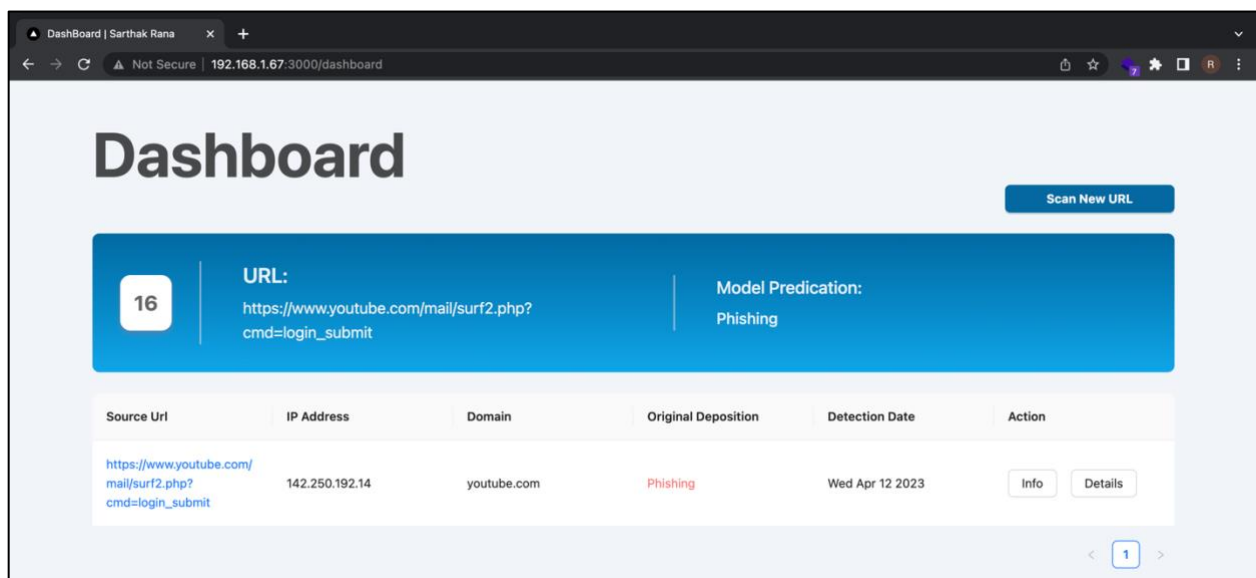


Figure 69: Dashboard displaying the details on the scan of the phishing URL.

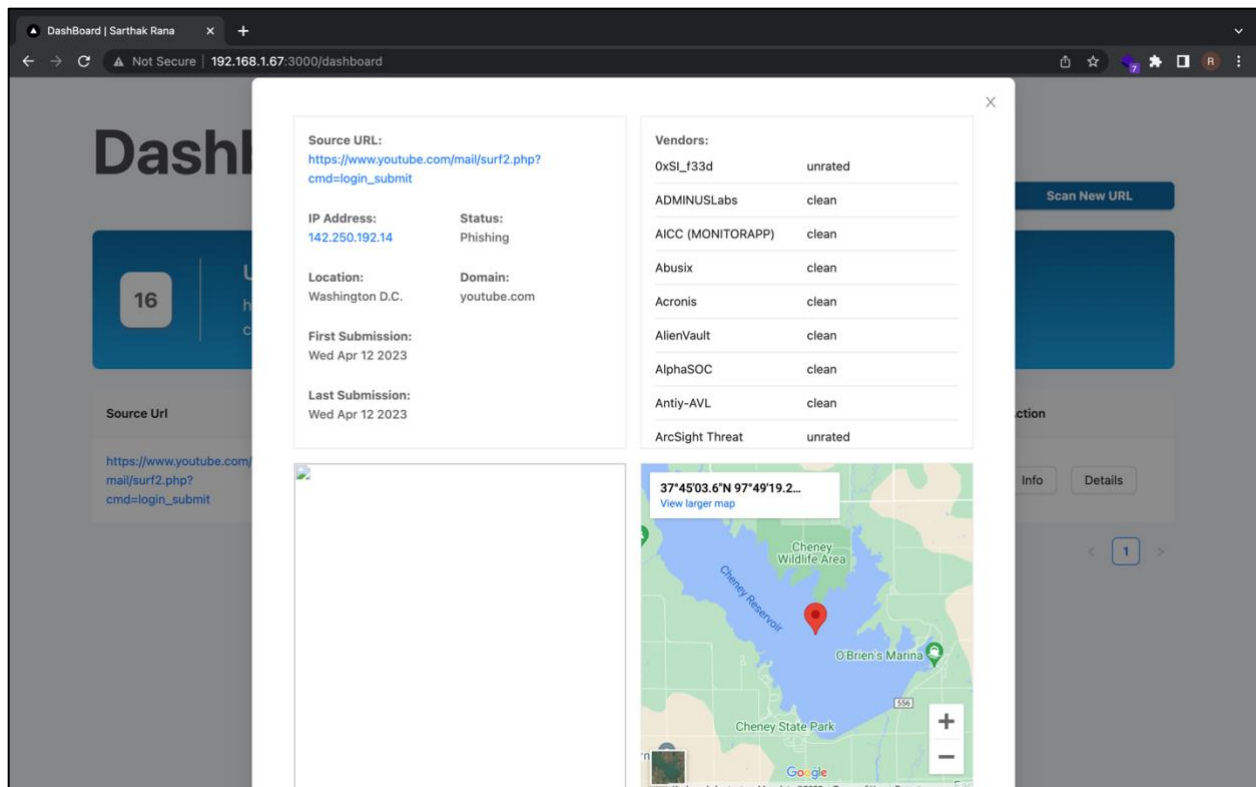


Figure 70: Additional details on the scan of phishing URL.

Table 28: System Test Case 2.

System Test Case 2	
Objective	To test the entire system.
Action	Enter a legitimate URL to scan.
Expected Result	The system should scan the URL and display the detailed result on the dashboard.
Actual Result	The system successfully scans the URL and displays the information on the dashboard.
Conclusion	Test Successful.

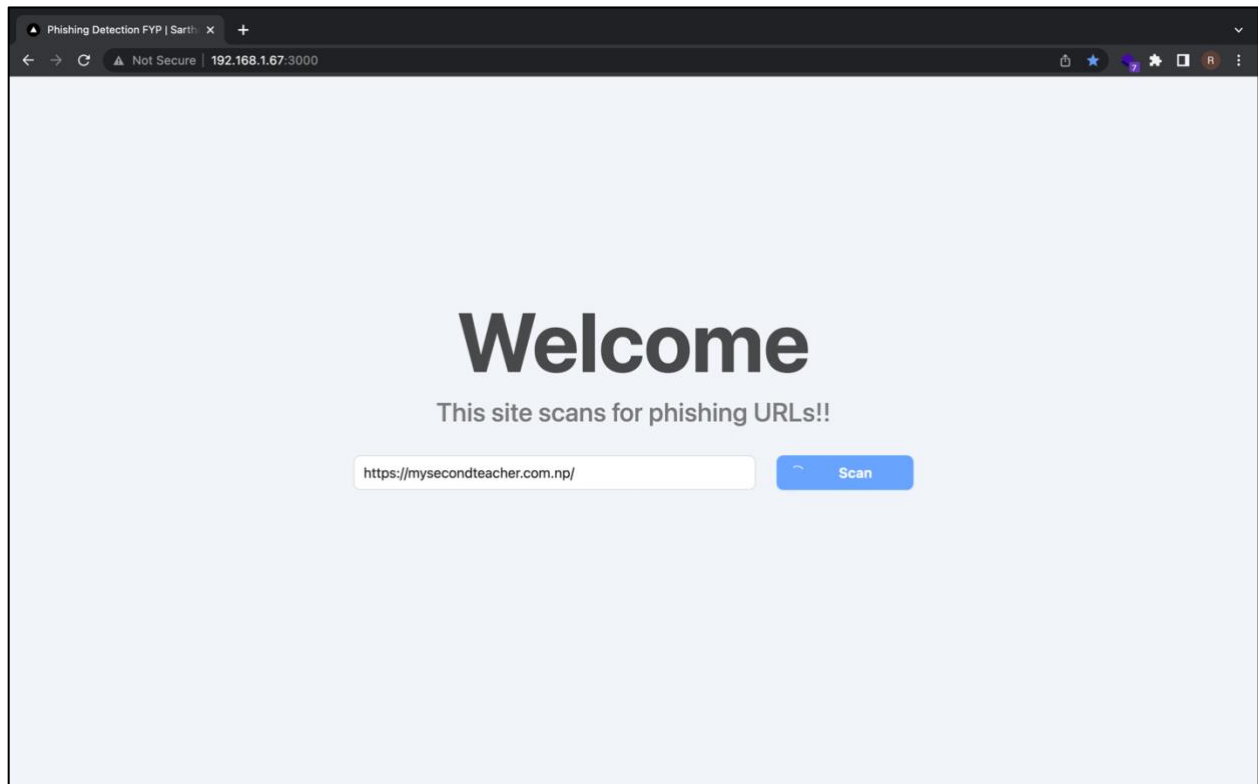
Evidence:

Figure 71: Providing a legitimate URL to scan.

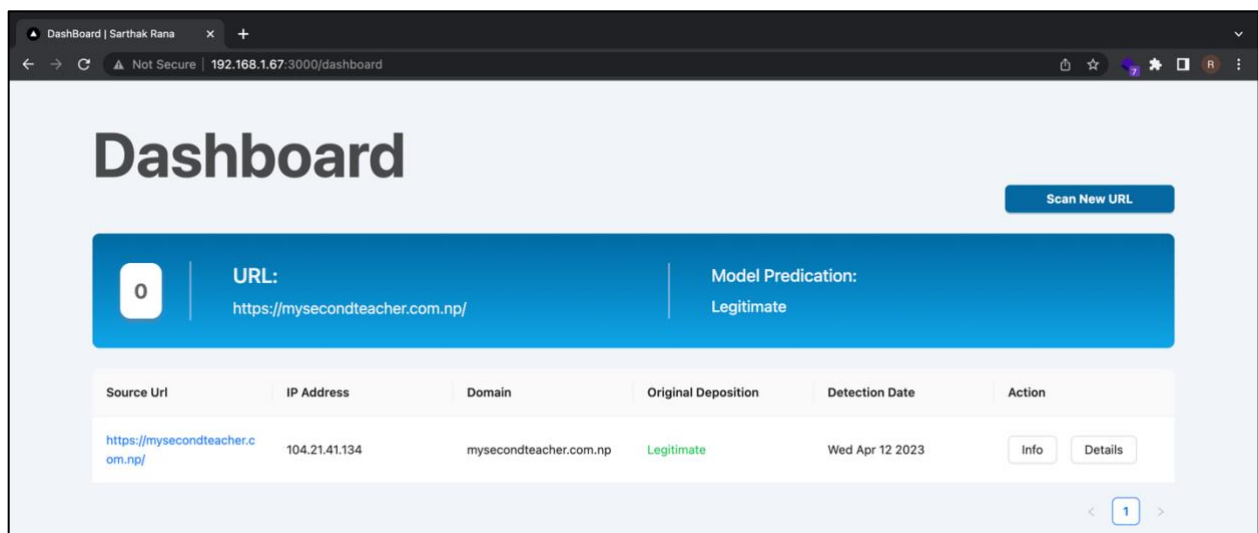


Figure 72: Dashboard displaying the details on the scan of the legitimate URL.

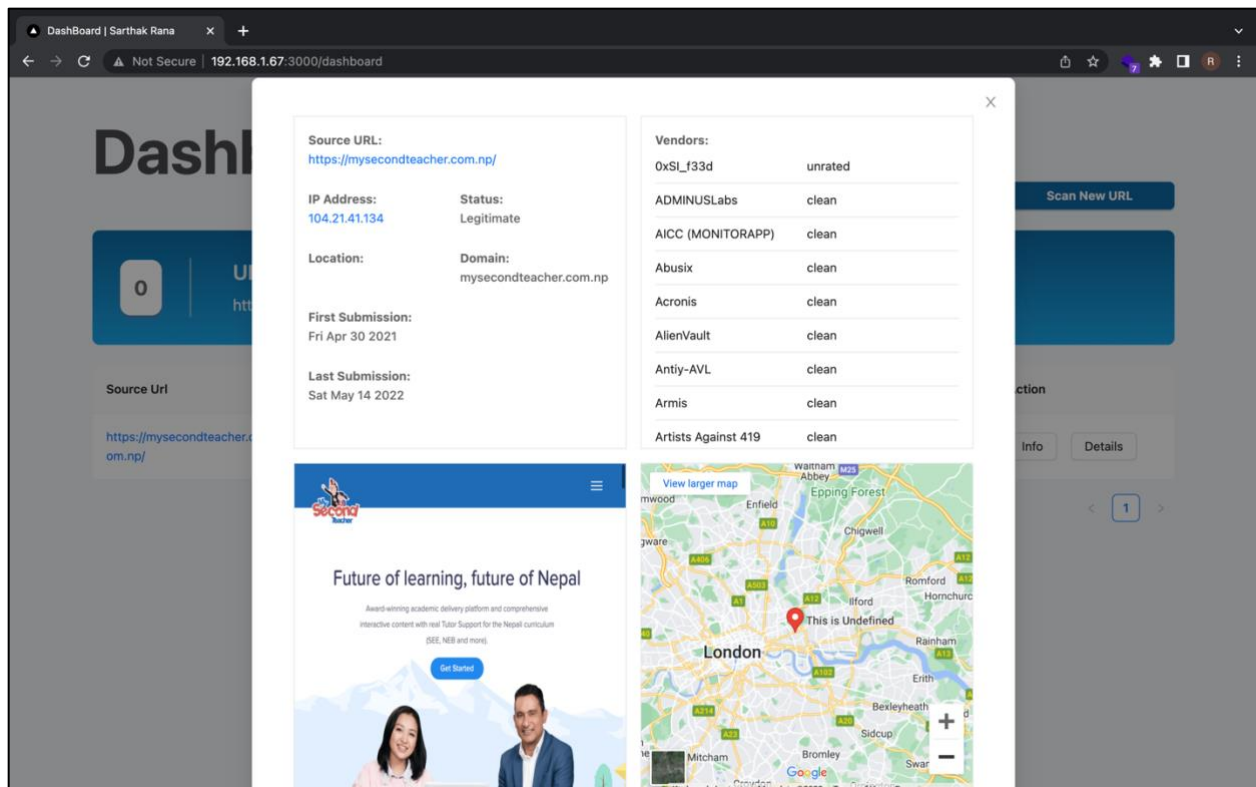


Figure 73: Additional details on the scan of legitimate URL.

4.4 Critical Analysis

The project was successfully completed using the Scrum Methodology, a key component of the agile approach. This project's main goal was to give users a system that can evaluate URLs to see if they are phishing attempts and provide supplemental information that may be shown on a dashboard. Through a combination of unit and system testing, the project's dependability, effectiveness, performance, and security were evaluated.

This testing strategy worked well for improving the system by fixing any faults or bugs that had not yet been fixed or patched. Unit testing was used to evaluate the system's individual components first, then system testing was used to perform a thorough analysis of the complete system. Despite the fact that all of the tests were successful, the testing run also helped identify the system's strengths and weaknesses.

4.4.1 Strengths

- All unit and system tests were successfully completed.
- The client's requirements were fully incorporated into the system.
- The project's anticipated outcomes and deliverables were achieved.
- The system's functionality, including data pre-processing, feature extraction, and model training, was confirmed.
- Enhanced performance was verified, ensuring the ML model and dashboard efficiently respond to user inputs.
- Users can enjoy a smooth experience while using the system.
- The system enables users to scan any URL to determine if it is a phishing attempt.
- The system saves the logs of scanned URLs in the browser's local storage.
- The system offers additional information for scanned URLs, such as domain, IP address, scan date, antivirus vendor results, a webpage screenshot, and even a map showing the hosting location of the IP address.
- The system effectively identifies and differentiates between legitimate and phishing URLs.

- The intuitive dashboard design allows users to easily access and interpret the results of URL scans.
- Cross-platform compatibility ensures that the system works seamlessly across various devices and browsers.
- Scalability is confirmed, allowing the system to handle increasing loads and accommodate future growth.

4.4.2 Weaknesses

- At present, the project depends on a limited dataset, which may negatively impact the ML model's accuracy and compromise its ability to effectively detect new phishing URLs.
- If the model overfits the training data, it could perform inadequately on previously unseen phishing URLs, leading to a decline in reliability and effectiveness.
- Although the system has been extensively tested, it may still generate occasional false positives or negatives, potentially undermining users' trust.
- Adapting the ML model to new phishing attack methods or techniques might prove challenging, necessitating frequent updates and maintenance to remain effective.
- The system might pose privacy concerns if it gathers or processes user data without properly implementing data protection and security measures.

CHAPTER 5: Conclusion

5. Conclusion

In conclusion, phishing attacks are more prevalent than ever before, and it is critical to detect them early to protect users. Although phishing detection cannot be completely eliminated, it can be significantly reduced by using ML. The deliverable of this project is to provide its users with a system that can identify both legitimate and phishing URLs using ML techniques. Other possible goals include identifying websites that are vulnerable and require further investigation and work and examining the current website environment, and predicting whether the network's website as well as the servers have maximum security.

This report investigated how ML approaches could detect phishing URLs with high accuracy. The developed model analyzed several URL features, including both search bar-based and domain-based features, to identify phishing URLs. The developed system also included a visualization dashboard that allows users to view the detected URLs and track their status. This dashboard is simple and easy to use, allowing users to identify and respond to phishing threats quickly.

Overall, the ML-based phishing URL detection system detected phishing URLs with high accuracy and effectiveness. The visualization dashboard improves usability by allowing users to monitor phishing threats in real time and take proactive measures to avoid harm. This system can be used in various organizations to protect their users from phishing attacks and thus improve cybersecurity.

5.1 Legal, Social, and Ethical Issues

5.1.1 Legal Issues

The creation and implementation of an ML-based phishing URL detection system with a visualization dashboard may raise legal concerns under the Electronic Transactions Act and the National Criminal Code. The system ensures that it complies with the Act's requirements, such as ensuring the security and reliability of electronic communications, and that it is designed to prevent fraud and other illegal activities. The system also follows cybersecurity, data protection, and privacy regulations that the National Criminal Code may include as it follows privacy and data protection, copyright and intellectual property, and mainly API usage and licensing.

5.1.2 Social Issues

The development and deployment of an ML-based phishing URL detection system with a visualization dashboard may raise social issues related to the Electronic Transactions Act and the National Criminal Code. One of these issues is the potential for false positives and false negatives, which could lead to users being denied access to legitimate websites or being directed to malicious websites as well as raising awareness and educating the public. Additionally, the system gathers and stores sensitive user data, such as IP addresses and domain info, which sparked social concerns about user privacy. Striking a balance between efficient phishing detection and respecting users' privacy expectations was a critical aspect of the project as it implements measures to protect user data from unauthorized access or disclosure.

5.1.3 Ethical Issues

During the development of this project and the final report, the provision of relevant references and citations, which is ensured by review and verification on multiple occasions by myself and supervisors to avoid any ethical difficulties, has taken plagiarism into account and avoided. When creating the reports and project, the LMU's laws and regulations were strictly followed and used. The system respects the customer's data, and the client owns no intellectual property. As a result, the system and report do not violate any ethical standards, and thus there is no violation.

Finally, addressing legal, social, and ethical problems was a critical component in developing and implementing this project. This project was built with these considerations in mind, providing a responsible and successful solution that complies with acceptable legislation, develops confidence, and contributes to a more secure and accessible digital world. Following these standards improved the tool's overall effectiveness, which will aid in its sustained adoption and usage by individuals and groups.

5.2 Advantages

There are several advantages to developing a project for phishing URL detection using ML with a dashboard for visualization:

- **Increased Detection Accuracy:** ML algorithms can analyze large amounts of data and identify patterns that humans may miss. This can help improve phishing detection accuracy, lowering the risk of successful attacks.
- **Real-Time Monitoring:** Users can monitor phishing attempts in real-time by using a dashboard for visualization. This can assist organizations in responding quickly to new threats and preventing additional attacks.
- **Efficient Resource Allocation:** By focusing on high-risk areas, ML-based phishing detection can help organizations allocate their resources more efficiently. This can help reduce the cost and time required to respond to phishing attacks.
- **Compliance with Regulations:** Organizations can demonstrate compliance with regulations such as the Electronic Transaction Act and the National Criminal Code by using ML-based phishing detection. This can help to lower the likelihood of fines and legal action.
- **Scalability:** ML-based phishing detection can be scaled to handle large volumes of data and traffic, so it is appropriate for organizations of all sizes.
- **Easy Integration:** Organizations can easily adopt and use ML-based phishing detection because it can be easily integrated with the existing network.
- **Future Proofing:** ML algorithms can adapt and learn from new threats and data, increasing their effectiveness over time. This can help to protect the system from emerging threats and keep organizations ahead of the competition.

5.3 Limitations

There are several limitations that should be considered when developing this project such as:

- **Limited Data Availability:** To be effective, ML algorithms require a large amount of high-quality data. The algorithm's performance may suffer as a result of the limited availability of such data.
- **Limited Explainability:** ML algorithms can be challenging to comprehend and explain. This could make it difficult to understand how the algorithm makes predictions and troubleshoot any problems that arise.
- **False Positives and Negatives:** The algorithm may produce false positives (identifying a legitimate website as a phishing website) or false negatives (failure to identify a phishing website), reducing the system's effectiveness.
- **Technological Limitations:** The computational power and resources available may have an impact on the performance of the ML algorithm. This could have an effect on the system's speed and accuracy.
- **Lack of Human Oversight:** Over-reliance on ML algorithms may result in a lack of human oversight and a failure to detect new and emerging threats on which the algorithm has not been trained.
- **Ethical Considerations:** The use of ML to detect phishing raises ethical concerns about algorithmic bias, data privacy, and transparency. It is critical that the system is designed and implemented in an ethical and responsible manner.

5.4 Future Work

There are several future works that can be considered for the project phishing URL detection using ML with a dashboard for visualization:

- Improving Accuracy: Continued research and development can improve the accuracy of the ML algorithms used for phishing URL detection. This can be achieved through improvements in data quality, feature selection, and model optimization.
- Integrating Multiple Data Sources: The integration of multiple data sources, such as email headers and network traffic, could improve the accuracy of phishing detection and reduce false positives.
- Enhancing User Experience: Enhancing the user experience of the dashboard could improve user adoption and make it easier for users to monitor and respond to phishing threats.
- Integrating with Third-Party Systems: Integrating the phishing URL detection system with third-party systems, such as threat intelligence feeds and security information and event management (SIEM) systems, could help improve the overall security posture of the organization.
- Making browser extensions through it: After making a browser extension of this system, the system can even push a pop-up notification through it where the users can be stopped before visiting such sites.
- Adding malware detection into it: ML techniques can be included in this project to also detect malware.

6. References and Bibliography

Rosenthal, M., 2022. *tessian*. [Online]

Available at: <https://www.tessian.com/blog/phishing-statistics-2020/>

[Accessed 17 December 2022].

Platten, M., 2021. *linekdin*. [Online]

Available at: <https://www.linkedin.com/pulse/current-state-phishing-how-digital-threat-protection-platten-cissp/>

[Accessed 10 October 2022].

Bharati, M. J., Kumar, T. P. & Reddy, B. P., 2019. *Detecting phishing website using machine learning*, Bangalore: New Horizon College of Engineering.

Sylvia, K., 2021. *Phishing URL Detection Presentation*. [Online]

Available at: <https://www.youtube.com/watch?v=sYiB2nnVOtU&t=769s>

[Accessed 25 December 2022].

Tiwari, T., 2020. *Phishing Sites Prediction Using Machine Learning*. [Online]

Available at: <https://www.youtube.com/watch?v=zKNXHluHneU>

[Accessed 25 December 2022].

Adobe Communication, 2022. *Adobe Experience Cloud Blog*. [Online]

Available at: <https://business.adobe.com/blog/basics/waterfall>

[Accessed 14 October 2022].

Lucid Content, 2021. *Lucidchart*. [Online]

Available at: <https://www.lucidchart.com/blog/waterfall-project-management-methodology>

[Accessed 14 October 2022].

Martin, M., 2022. *Guru99*. [Online]

Available at: <https://www.guru99.com/software-engineering-prototyping-model.html>

[Accessed 14 October 2022].

Martin, J., 2022. *coolbluweb*. [Online]

Available at: <https://coolbluweb.com/blog/guide-to-agile-methodology/>

[Accessed 14 October 2022].

BYDREC, 2022. *Full Comparison - Agile vs. Scrum vs. Waterfall Methodology*. [Online]

Available at: <https://blog.bydrec.com/a-comprehensive-comparison-between-the-agile-scrum-and-waterfall-methodology>

[Accessed 21 March 2023].

University of New Brunswick, 2023. *URL 2016 | Datasets | Research | Canadian Institute for Cybersecurity | UNB*. [Online]

Available at: <https://www.unb.ca/cic/datasets/url-2016.html>

[Accessed 3 September 2022].

javatpoint, 2022. *UML Use Case Diagram - Javatpoint*. [Online]

Available at: <https://www.javatpoint.com/uml-use-case-diagram>

[Accessed 2 April 2023].

D, M., H, H. & Y, K., 2008. An assesment of AI based techniques for recognition of phishing destinations. *Aust J Intell Ind Process Syst*, 10(2), pp. 54-63.

Visual Paradigm, 2023. *Flowchart Tutorial (with Symbols, Guide and Examples)*.

[Online]

Available at: <https://www.visual-paradigm.com/tutorials/flowchart-tutorial/>

[Accessed 3 April 2023].

Allabarton, Rosie, 2022. *How To Create Wireframes For A Website [With Examples]*.

[Online]

Available at: <https://careerfoundry.com/en/blog/ux-design/how-to-create-your-first-wireframe/>

[Accessed 3 April 2023].

JavaScript Tutorial, 2023. *JavaScript Fetch API Explained By Examples*. [Online]
Available at: <https://www.javascripttutorial.net/javascript-fetch-api/>
[Accessed 13 April 2023].

G, X., J, H., CP , R. & L , C., 2011. A include rich AI structure for identifying phishing sites. *ACM Trans Inf Syst Security*, 14(2), pp. 1-28.

Mahajan, R. & Siddavatam, I., 2018. Phishing website detection using machine learning algorithms. *International Journal of Computer Applications*, 181(23), pp. 45-47.

Guo, H. & Yang, X., 2007. A simple reliability block diagram method for safety integrity verification. *Reliability Engineering & System Safety*, September, 92(9), pp. 1267-1273.

Visual Paradigm, 2020. *What is Work Breakdown Structure?*. [Online]
Available at: <https://www.visual-paradigm.com/guide/project-management/what-is-work->
[Accessed 15 October 2022].

IH, W. & E, F., 2002. *Data mining: useful AI apparatuses and strategies with Java executions.*, New York: ACM.

Lucidchart, 2022. *What is a Data Flow Diagram | Lucidchart*. [Online]
Available at: <https://www.lucidchart.com/pages/data-flow-diagram>
[Accessed 18 April 2023].

Bajracharya, N., 2021. *Information system audit in Nepal: Everything you should know about - OnlineKhabar English News*. [Online]
Available at: <https://english.onlinekhabar.com/information-system-audit-in-nepal-explained.html>
[Accessed 24 December 2021].

Y, Z., J, H. & L, C., 2007. *CANTINA: A substance based way to deal with recognize phishing sites*, Cannada: ACM.

Salahdine, F., Marbet, Z. E. & Kaabouch, N., 2020. *Phishing Attacks Detection A Machine Learning-Based Approach*, Charlotte: University of North Carolina at Charlotte.

7. Appendix

7.1 Client Approval Letter

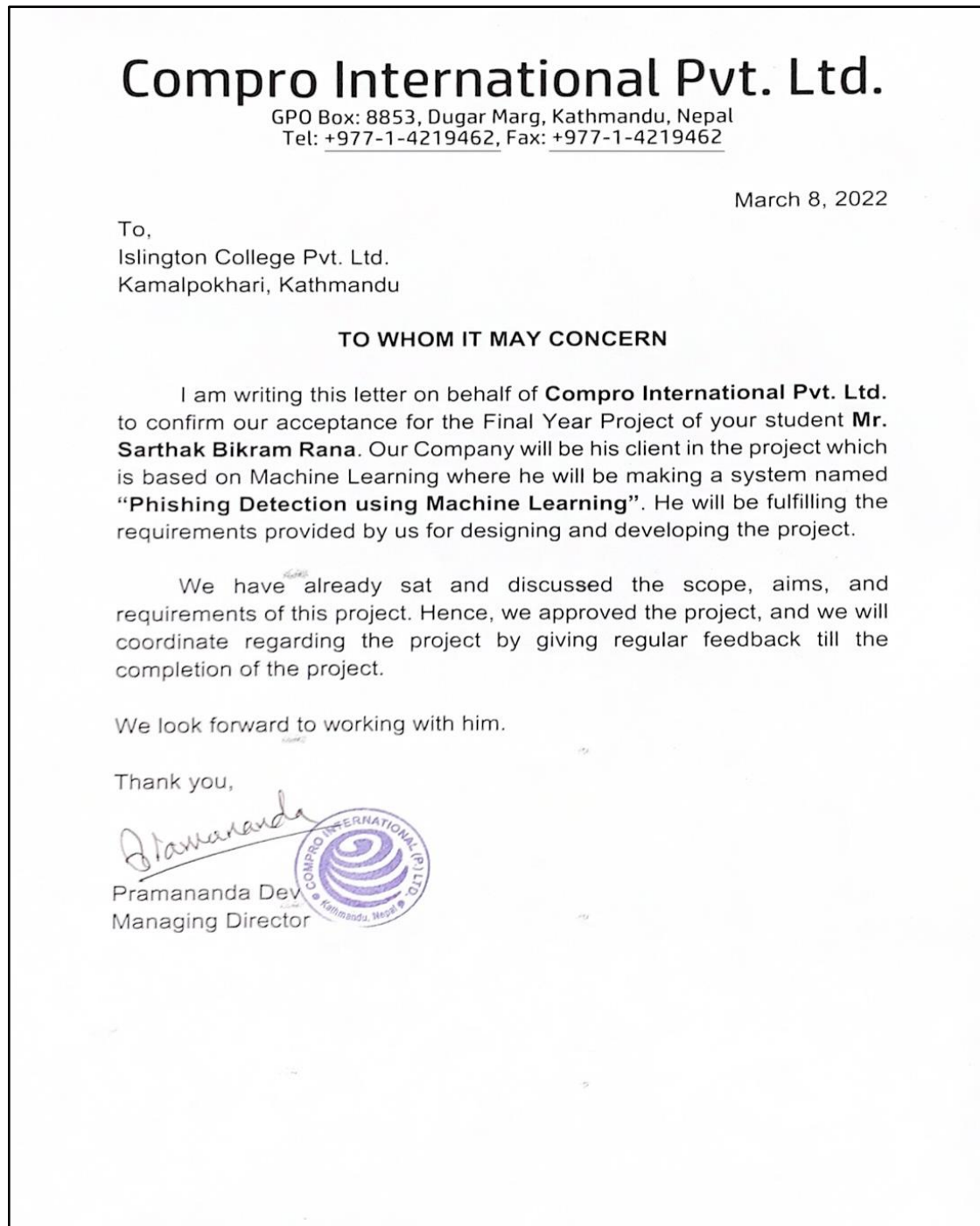


Figure 74: Client Letter.

Back To: [2.1 About the End Users](#)

7.2 Understanding the Solution

7.2.1 Project Elaboration

Attackers seek to acquire sensitive information by imitating trusted organizations in phishing attacks, which constitutes a significant risk to internet security. To address this problem, our project intends to create a phishing URL detection system that uses ML algorithms to identify potential phishing URLs. The project's development is separated into three major sections: back-end, front-end, and integration and deployment.

7.2.1.1 Back-End Development

The development of the system's back-end is fully coded in Python, with five different files, two of which are in 'ipynb' format and contain code for ML, and the other three are Python files, one of which contains the main code where the model predicts whether the URL is phishing or legit, and the other file contains the code for the virus total API and one contains code to retrieve the IP address of the domain.

- **feature extraction.ipynb**

First, the major purpose of the PDML project's backend development phase is to establish a stable and fast ML model capable of reliably detecting phishing URLs. To do this, the project begins with data collecting, accumulating a large dataset of both phishing and legal URLs from trustworthy web sources. The domain, presence of an IP address, '@' sign, special character count, URL length, number of '/', URL redirection, HTTPS token presence, shortening service usage, prefix and suffix, and counts of HTTP & HTTPS protocols in the URL are then extracted. Domain-specific factors such as DNS record verification, web traffic, and domain age are also taken into account. This comprehensive approach ensures a strong foundation for building a precise phishing URL detection system.

- **modeltrain.ipynb**

Following the completion of the feature extraction process, the revised dataset is shown using histograms using the 'matplotlib.pyplot' module. This approach provides a better knowledge of the value distribution of each feature in the dataset and aids in the identification of any potential outliers or inaccurate information. In addition, a heatmap is generated using the Seaborn package to investigate the correlations between the dataset's various attributes. This helps to comprehend multicollinearity and choose suitable features for the ML model. Finally, the dataset is separated into training and testing sets with a 70-30 split using the 'train_test_split' function from the 'sklearn.model_selection' module. This indicates that 70% of the data is utilized for training, while 30% is used for testing.

The dataset is trained and evaluated using multiple ML methods, such as the Decision Tree Classifier, which makes judgments based on input characteristics using a tree-like structure. A Random Forest Classifier is a group of decision trees that aggregate individual forecasts to improve accuracy and stability. A Multilayer Perceptron Classifier is an artificial neural network used to perform classification tasks. Support Vector Machine, a model that selects the optimum hyperplane in the feature space to split classes; Logistic Regression is a statistical model that models class probabilities using a logistic function. Gradient Boosting Classifier is an ensemble approach combining many weak learners to produce a strong learner. AdaBoost Classifier is another ensemble method that repeatedly modifies the weights of weak learners to produce a better-performing final model. Among these models, the Random Forest Classifier provides the highest train and test accuracy so the model is saved to a file in binary mode using Python's 'pickle' module which allows the model to be saved for later use without the need to retrain it.

- **main.py**

The model is imported into a new Python script containing all of the feature extraction logic once it has been trained and saved as a pickle file. The 'urlparse()' method parses the supplied URL and generates a feature vector from extracted features such as the existence of an IP address, the '@' sign, URL length, and others. The loaded model predicts if the URL is authentic or phishing once the feature vector is available. The outcome of the prediction is kept in a variable. If the prediction is zero, the URL is legit else, it is phishing.

In addition to these capabilities, the file contains Python code that displays the domain, location details, IP address, and other data from the VirusTotal API. Flask is also used to develop a web server since it allows for the smooth integration of phishing detection features into a user-friendly web server. The server is accessible from any IP address, providing users with a simple platform to use the phishing detection service.

- **virustotalapy.py**

In this Python code snippet, a method is defined which receives a URL and information about it from the VirusTotal API. First, the URL is base64 encoded and added to the base API URL. Following that, a 'headers' dictionary is constructed, which includes the necessary API key for authentication.

A 'get' request is sent to the API endpoint using the requests library, along with the authentication headers. After that, the response data is transformed into JSON format. A new dictionary named 'temp_response' is established to contain response-related information such as detection date, first submission date, last submission date, last analysis results, and last analysis stats. Finally, the method returns the 'temp_response' dictionary, which contains the data obtained from the VirusTotal API for the given URL.

- **newlocation.py**

A new function is created in this Python code sample to collect information about a given URL, such as its IP address and location. details. First, if a trailing slash is lacking from the URL, it adds one and produces a unique screenshot filename depending on the URL. Using the `socket.gethostbyname()` method, the domain name is extracted, and the IP address is retrieved. The APIIP.net service is then requested with the IP address and access key to obtain the necessary information. The method also takes a snapshot of the URL in headless mode using the Selenium WebDriver and saves it in the different folder. In the end, the function returns a dictionary containing the collected data, like the IP address, location details, and the screenshot URL of the website.

Various Python libraries with various proposals were utilized throughout the development of this system's backend, including:

'pickle' and **'joblib'**: These libraries are used for saving and loading the trained ML model.

'pandas' and **'numpy'**: These libraries provide data manipulation and numerical computation capabilities.

'urllib', **'socket'**, and **'ipaddress'**: These libraries help with URL parsing, IP address extraction, and other network-related tasks.

'whois' and **'datetime'**: These libraries are used for obtaining domain information and working with dates and times.

'bs4' (BeautifulSoup): This library is used for web scraping and parsing HTML and XML documents.

'flask' and **'flask_cors'**: These libraries help create a web server and handle cross-origin resource sharing for the project.

'sklearn': This library provides various ML algorithms, model evaluation metrics, and other tools for model development.

'seaborn' and **'matplotlib'**: These libraries are used for data visualization, such as creating histograms and heatmaps.

'selenium': This library is used for automating web browsers to capture screenshots of URLs.

'requests' and **'json'**: These libraries handle API requests and JSON data manipulation.

'folium' and **'webbrowser'**: These libraries are used for creating interactive maps and working with web browsers.

7.2.1.2 Frontend Development

The system's front-end is written in 'Next.js' and employs 'antd' as a UI/UX module to build forms, tables, buttons, divisions, and other elements. The front-end code is split into two files: one for the homepage, which includes a search bar, and another for the system dashboard. During this development of front-end 'Tailwind css' is used as the css processor and 'Typescript' is used for static typing which helps to catch errors during development rather than during runtime.

- **index.tsx**

This page is built on the react component for the PDML system's home page. It styles and handles form inputs using the Ant Design Library. The main page includes a welcome greeting, a brief description of the website's purpose, and a form where visitors may enter a URL to scan. The code sample is divided into sections in which relevant libraries and components from Ant Design, Next.js, and React are imported first.

The major exported component for this page is the 'Home' component. The 'useRouter' hook is used within the component to access the Next.js router, and 'useState' is used to handle the loading state of the search button. The page's primary content is then presented, including a welcome message, a brief description, and a form for visitors to enter a URL. When the user clicks on the 'Scan' button, the form's submit handler is invoked.

The form's submit handler is an asynchronous function that sends a request to a backend server that includes the input URL as a query parameter. It retrieves the result from the server, keeps it locally, and then navigates to the '/dashboard' route with the data it has retrieved. If an issue occurs during the retrieval process, a notice telling the user that the URL does not exist will be shown. When the request is completed or an error occurs, the search button's loading state is reset to false.

- **dashboard.tsx**

This code sample is a React component for the dashboard page of a URL-scanning application that displays and manages URL-related data. The component includes libraries like Next.js, Ant Design, and React. Ant Design is a library used particularly for UI components and style.

The DashBoard component begins with a number of hooks, one of which is 'useEffect,' which gets data from local storage when the component is attached, establishes the component's initial state, and ensures that any unnecessary data is cleaned up. When a new URL scan result becomes available, it is appended to the current data and set as the active detail. The 'useState' hooks are used to handle the component's many states, such as the active details, modal visibility, initial render status, and table data source.

The return statement contains the dashboard's primary content. It starts with the Head component, which sets the page title, and then moves on to a conditional rendering dependent on the 'firstRender' state. This guarantees that the main information is presented immediately once the first data gathering is finished. The dashboard has a title, a "Scan New URL" button that takes users back to the home page to scan a new URL, and a summary section that provides the contents of the presently selected URL. The summary section has a gradient background and visually shows the URL, model prediction, and other relevant details.

The Ant Design library's table component is used to display a list of URLs together with important information like as IP address, domain, and initial deposition. The 'dataSource' state is used to build the table, and the column rendering is adjusted to handle different data types, such as links and dates. Users may examine additional information about a certain URL by clicking the "Info" and "Details" buttons inside the table. The "Info" button refreshes the active details section, whereas the "Details" button opens a modal presenting further URL information. The modal has a grid structure that displays the source URL, IP address, status, location, domain, initial submission date,

final submission date, a list of vendors, a website screenshot, and a Google Maps 'iframe' with the URL's geolocation.

Finally, this React component creates a detailed dashboard page for a URL scanning program. It efficiently retrieves, displays, and organizes URL-related data in a user-friendly and aesthetically appealing manner, allowing users to engage with the program and get insights into the scanned URLs.

7.2.1.3 Integration & Deployment

The Fetch API is being used in this project for the integration of the front-end and the back-end. The Fetch API essentially contacts the Flask-hosted web server to retrieve data from the backend and display it on the front end.

The system is coded in such a way that it handles URL input from the user and sends a request to the backend server to detect phishing. When a user enters a URL, a function is called, indicating that the search is in progress, which is commonly shown by a loading animation or icon. The try block then makes a fetch request to the backend server at the hosting IP address, including the user-supplied URL as an encoded query parameter. If everything goes smoothly, the response is parsed as JSON and it is saved under a variable. The system then moves on to the dashboard route, most likely to showcase the phishing detection results.

The system's deployment is based on Flask, a lightweight web framework that facilitates deployment by creating a web server that can host the PDML service, providing a user-friendly interface to interact with the backend.

The Flask framework is imported, and an instance of the Flask app is created to manage the project's routes and requests. Then the CORS is enabled to allow cross-origin requests and seamless communication between the front-end and backend. It features a '/search' endpoint that accepts GET requests and searches the query parameter for a URL. When users access this endpoint, the 'search_url()' function is called, which then calls the 'detect_phish()' method. This function examines the supplied URL to see if it is associated with phishing.

Running the software on host '0.0.0.0' and port '9696' makes the web server available from any IP address, allowing users to use the phishing detection service from their preferred platform.

Back To: [2.2 Understanding the Solution](#)

7.3 Considered Methodologies

7.3.1 Waterfall Methodology

The Waterfall methodology, also known as the Waterfall model, is a sequential development process that falls like a waterfall through all phases of a project (for example, analysis, design, programming, and testing), with each phase entirely wrapping up before moving on to the next. (Adobe Communication , 2022)

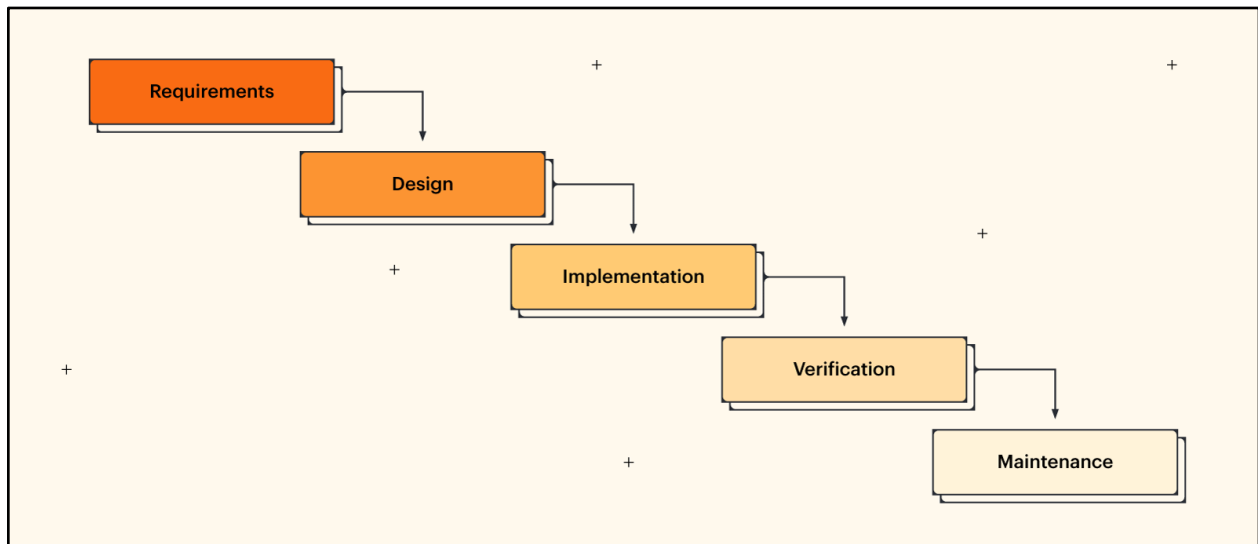


Figure 75: Waterfall Methodology (Lucid Content, 2021).

7.3.2 Prototype Methodology

The Prototype methodology is a software development model in which a prototype is developed, tested, and changed until it is acceptable. It also builds the foundation for the final system or software. It works best when the project's requirements are not fully understood. It is an iterative, trial-and-error process used by both the developer and the client. (Martin, 2022)

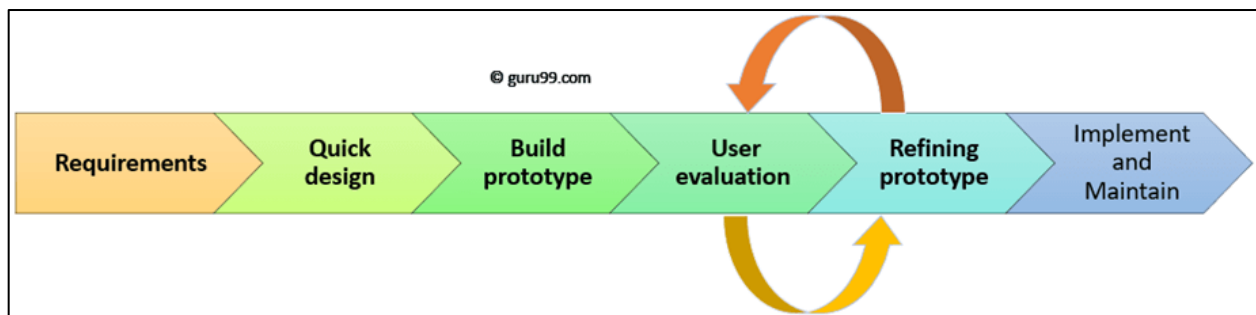


Figure 76: Prototype Methodology (Martin, 2022).

7.3.3 Agile Methodology

Agile Methodology is an iterative, incremental approach to project management and product development (like websites and software). It is a non-linear and adaptable methodology that allows for scope adjustments mid-project with the purpose of continual improvement during those changes. (Martin, 2022)

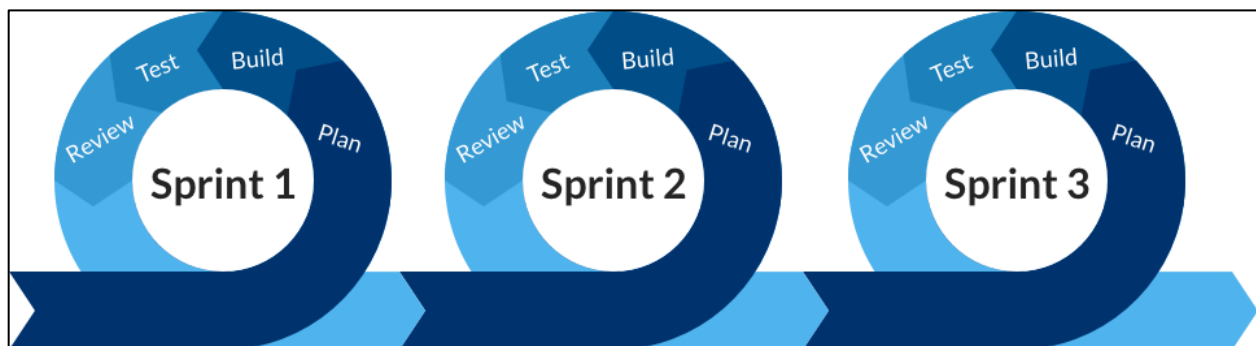


Figure 77: Agile Methodology (Martin, 2022).


Back To: [3.1 Considered Methodologies](#)

7.4 Pre-Survey

7.4.1 Pre-Survey Form

FYP Pre-survey Form

This is a pre-survey form for the FYP topic "Phishing detection using Machine Learning". The main aim of this project is to mitigate the global problem faced by internet users by providing them with a tool that can identify phishing URLs and legitimate trusted URLs using machine learning and help individuals minimize the risk of being victims of such phishing attacks.

sarthakbrana@gmail.com [Switch accounts](#) 

***Required**

Email *

Your email address

Full Name *

Your answer

Figure 78: Questions in the pre-survey form (a).

Phone Number

Your answer

Profession *

Your answer

Year

☐ Year 1

☐ Year 2

☐ Year 3

Figure 79: Questions in the pre-survey form (b).

Faculty

- ☐ Computing
- ☐ Computer Networking & IT Security
- ☐ Multimedia Technologies
- ☐ Mobile Application Development
- ☐ Artificial Intelligence
- ☐ International Business
- ☐ Digital Business Management
- ☐ Advertising & Marketing
- ☐ Events & Tourism Management
- ☐ Accounting & Finance

Are you familiar with the Cyber Security domain? *

- ☐ Yes
- ☐ No

Figure 80: Questions in the pre-survey form (c).

Do you know what a phishing attack is and how it works? *

☐ Yes

☐ No

☐ Maybe

Have you ever been the victim of such phishing attacks? *

☐ Yes

☐ No

☐ Maybe

Have you ever received any malicious emails redirecting you to a different webpage? *

☐ Yes

☐ No

☐ Maybe

Figure 81: Questions in the pre-survey form (d).

Have you ever used any threat intel platforms? If other, please enter below:

☐ Cisco Talos

☐ Virus Total

☐ Any Run

☐ Alien Vault

☐ Other: _____

Have you got any idea about Machine Learning and how it works? *

☐ Yes

☐ No

Did you know that this kind of phishing URL can be detected using Machine Learning algorithms? *

☐ Yes

☐ No

Figure 82: Questions in the pre-survey form (e).

The form is a vertical stack of three white question boxes with rounded corners, set against a light beige background. Each question box contains a question followed by three radio button options: 'Yes', 'No', and 'Maybe'. The first question asks if the system helps individuals avoid attacks. The second asks if keeping logs of scanned URLs is useful for future scenarios. The third asks if an organization or user should implement the system. At the bottom left is a brown 'Submit' button, and at the bottom right is a 'Clear form' link.

Does this kind of system help an individual from falling into such attacks? *

☐ Yes

☐ No

☐ Maybe

Do you think that keeping the logs of the previous scanned URL can be useful for future scenarios? *

☐ Yes

☐ No

Should an organization or a user implement this system? *

☐ Yes

☐ No

☐ Maybe

Submit

Clear form

Figure 83: Questions in the pre-survey form (f).

7.4.2 Sample of Pre-Survey Form

The screenshot shows a Google Forms interface for a pre-survey. At the top, it indicates '19 responses' with a green plus icon and a menu icon. Below this, there is a toggle switch for 'Accepting responses' which is turned on. The interface has three tabs: 'Summary', 'Question', and 'Individual', with 'Individual' being the active tab. Below the tabs, there is a dropdown menu showing 'sherpafura36@gmail.com' with a downward arrow. To the right of the dropdown is a navigation bar showing '< 18 of 19 >' with a small circular icon next to the number 18. Further right are icons for printing and deleting. Below the navigation bar, there is a yellow banner with the text 'Responses cannot be edited'. The main content area has the title 'FYP Pre-survey Form' in bold. Below the title is a paragraph: 'This is a pre-survey form for the FYP topic "Phishing detection using Machine Learning". The main aim of this project is to mitigate the global problem faced by internet users by providing them with a tool that can identify phishing URLs and legitimate trusted URLs using machine learning and help individuals minimize the risk of being victims of such phishing attacks.' At the bottom left of the content area, there is a red asterisk followed by the word 'Required'.

19 responses

Accepting responses

Summary Question Individual

sherpafura36@gmail.com

< 18 of 19 >

Responses cannot be edited

FYP Pre-survey Form

This is a pre-survey form for the FYP topic "Phishing detection using Machine Learning". The main aim of this project is to mitigate the global problem faced by internet users by providing them with a tool that can identify phishing URLs and legitimate trusted URLs using machine learning and help individuals minimize the risk of being victims of such phishing attacks.

***Required**

Figure 84: Pre-Survey Result from an individual (a).

Are you familiar with the Cyber Security domain? *

☒ Yes

☐ No

Do you know what a phishing attack is and how it works? *

☒ Yes

☐ No

☐ Maybe

This figure shows a pre-survey result for an individual. It consists of two sections. The first section asks 'Are you familiar with the Cyber Security domain? *' with two radio button options: 'Yes' (selected) and 'No'. The second section asks 'Do you know what a phishing attack is and how it works? *' with three radio button options: 'Yes' (selected), 'No', and 'Maybe'.

Figure 85: Pre-Survey Result from an individual (b).

Year

☐ Year 1

☐ Year 2

☒ Year 3

Faculty

☐ Computing

☒ Computer Networking & IT Security

☐ Multimedia Technologies

☐ Mobile Application Development

☐ Artificial Intelligence

This figure shows a pre-survey result for an individual. It consists of two sections. The first section is titled 'Year' and has three radio button options: 'Year 1', 'Year 2', and 'Year 3' (selected). The second section is titled 'Faculty' and has five radio button options: 'Computing', 'Computer Networking & IT Security' (selected), 'Multimedia Technologies', 'Mobile Application Development', and 'Artificial Intelligence'.

Figure 86: Pre-Survey Result from an individual (c).

Email *	sherpafura36@gmail.com
Full Name *	Fura Yanji Sherpa
Phone Number	9823001982
Profession *	Student

Figure 87: Pre-Survey Result from an individual (d).

Have you ever been the victim of such phishing attacks? *

☐ Yes

☒ No

☐ Maybe

Have you ever received any malicious emails redirecting you to a different webpage? *

☐ Yes

☐ No

☒ Maybe

Figure 88: Pre-Survey Result from an individual (e).

Have you ever used any threat intel platforms? If other, please enter below:

☐ Cisco Talos

☒ Virus Total

☐ Any Run

☐ Alien Vault

☐ Other: _____

Have you got any idea about Machine Learning and how it works? *

☐ Yes

☒ No

Figure 89: Pre-Survey Result from an individual (f).

Did you know that this kind of phishing URL can be detected using Machine Learning algorithms? *

☒ Yes

☐ No

Does this kind of system help an individual from falling into such attacks? *

☐ Yes

☐ No

☒ Maybe

Figure 90: Pre-Survey Result from an individual (g).

Do you think that keeping the logs of the previous scanned URL can be useful for future scenarios? *

☒ Yes

☐ No

Should an organization or a user implement this system? *

☒ Yes

☐ No

☐ Maybe

Submitted 17/12/2022, 16:55

Figure 91: Pre-Survey Result from an individual (h).

7.4.3 Pre-Survey Result

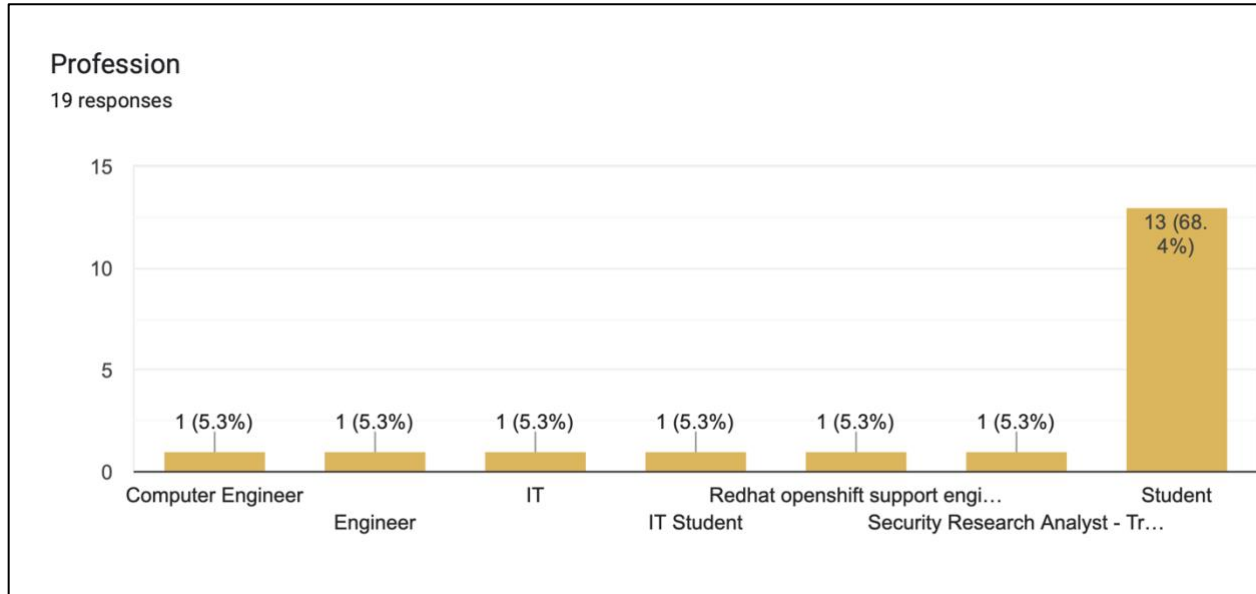


Figure 92: Profession of individuals who filled out the pre-survey form.

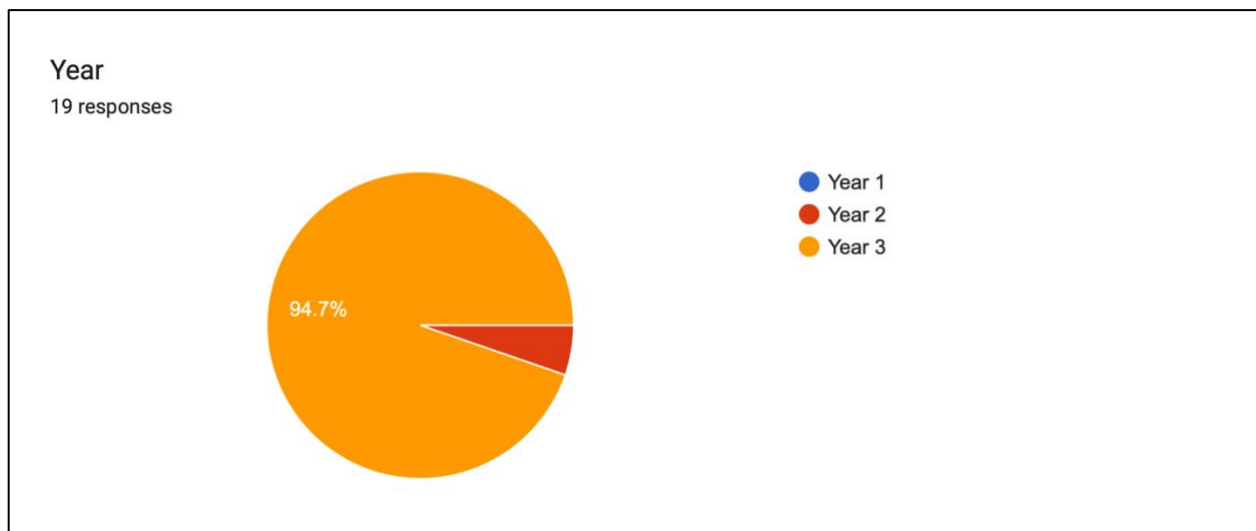


Figure 93: Current college year of the student who filled out the form.

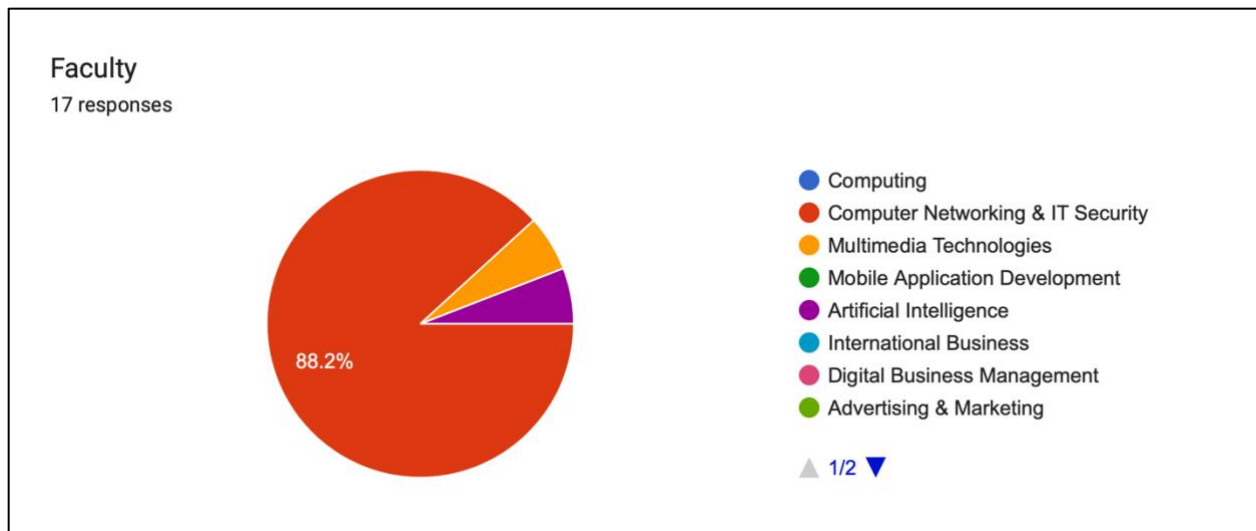


Figure 94: Faculty of the student who filled out the form.

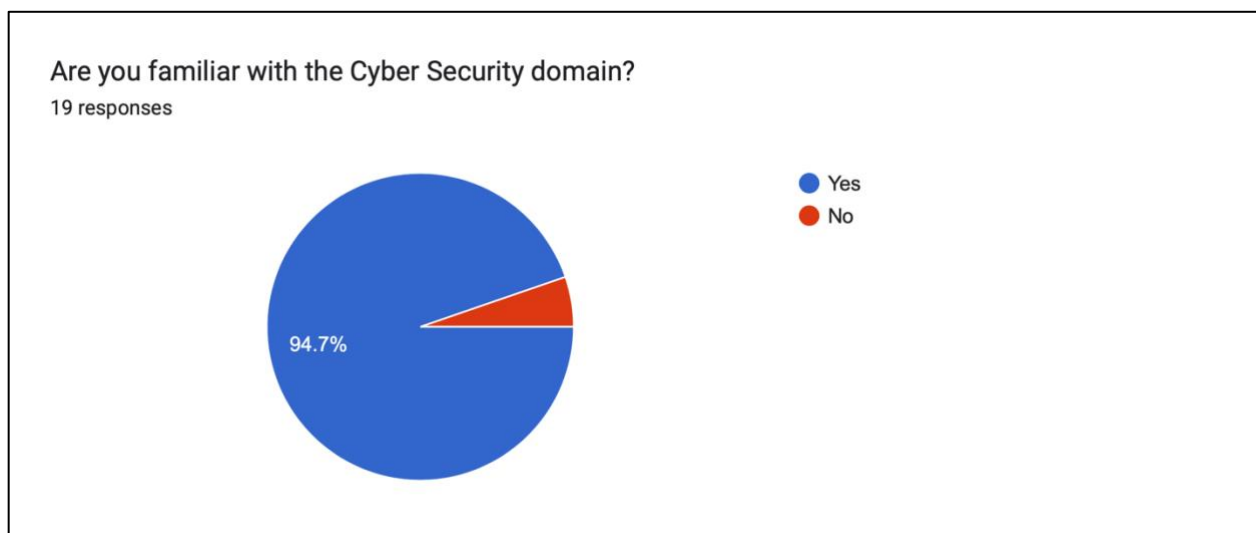


Figure 95: Pre-Survey Question no.1

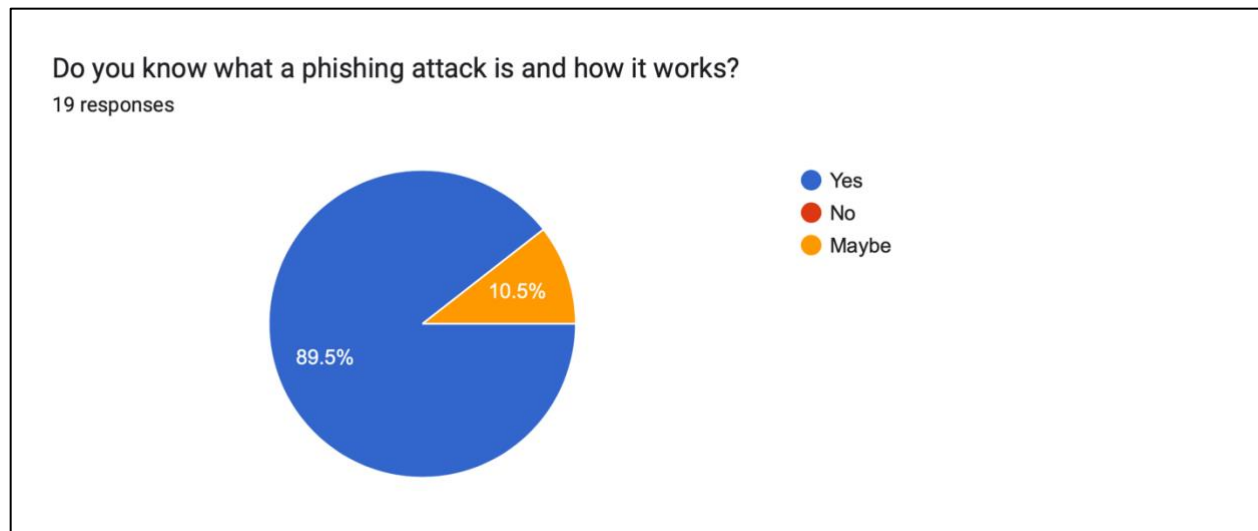


Figure 96: Pre-Survey Question no.2

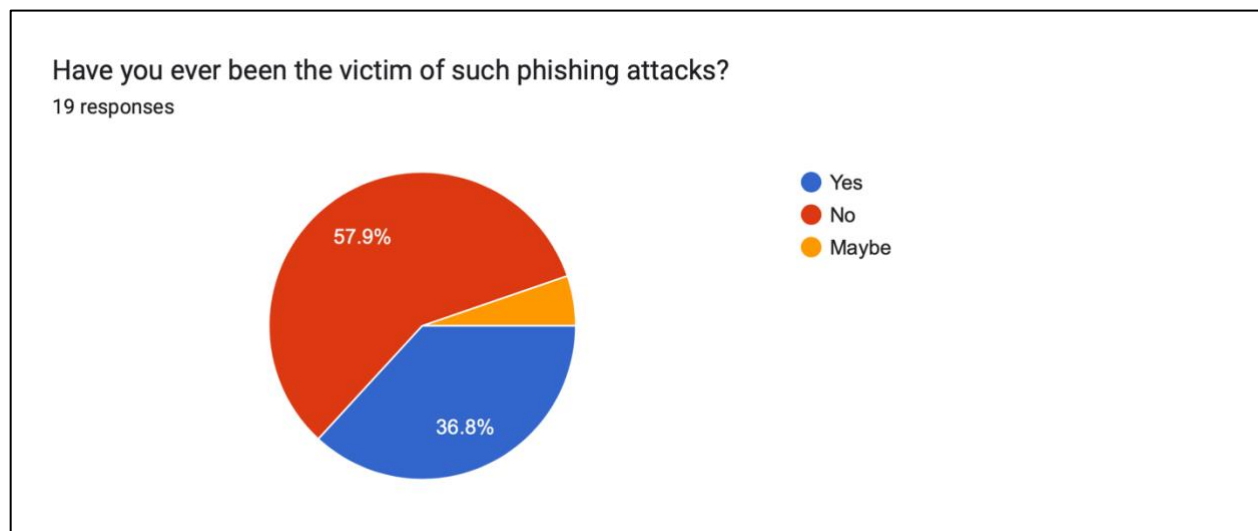


Figure 97: Pre-Survey Question no.3

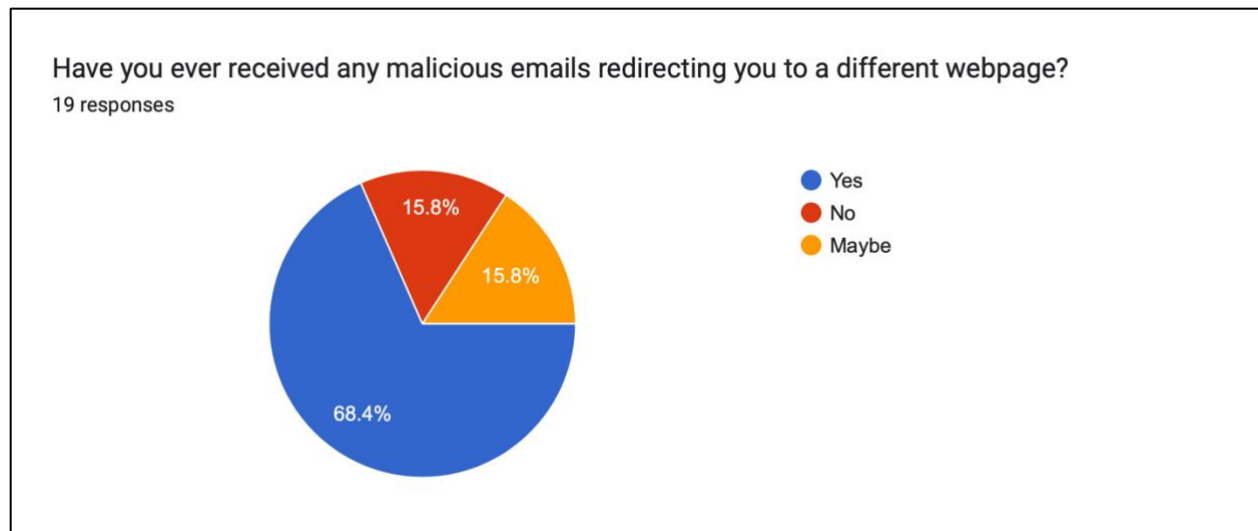


Figure 98: Pre-Survey Question no.4

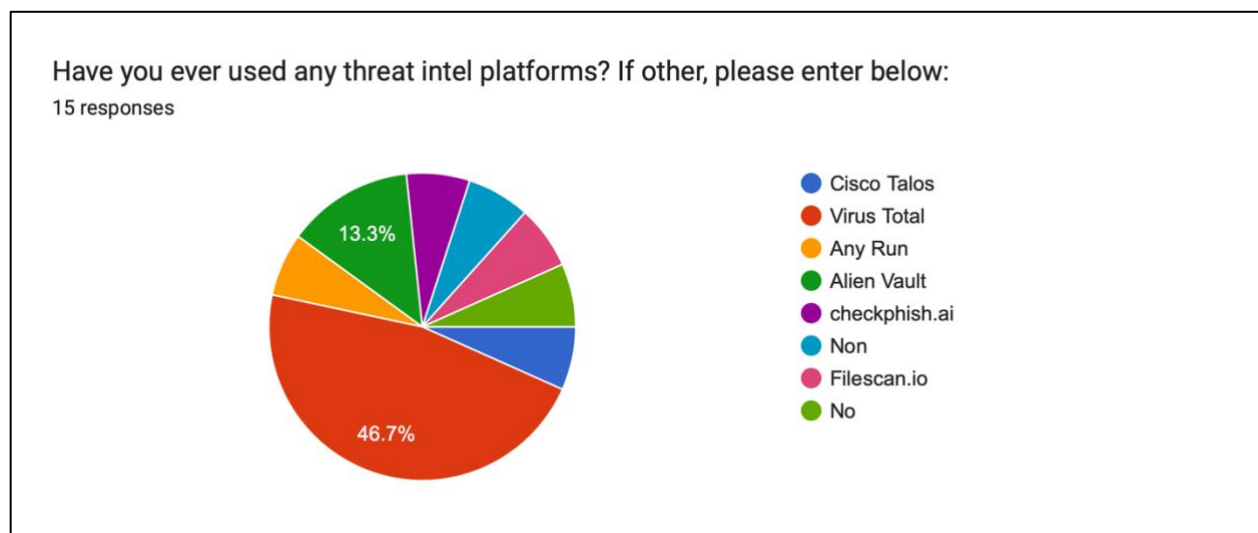


Figure 99: Pre-Survey Question no.5

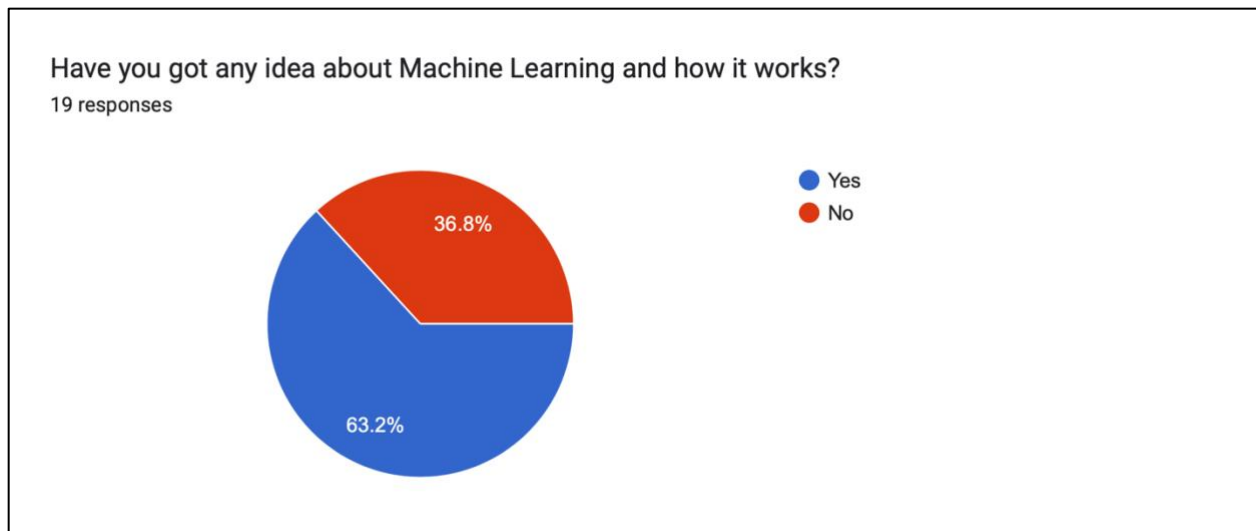


Figure 100: Pre-Survey Question no.6

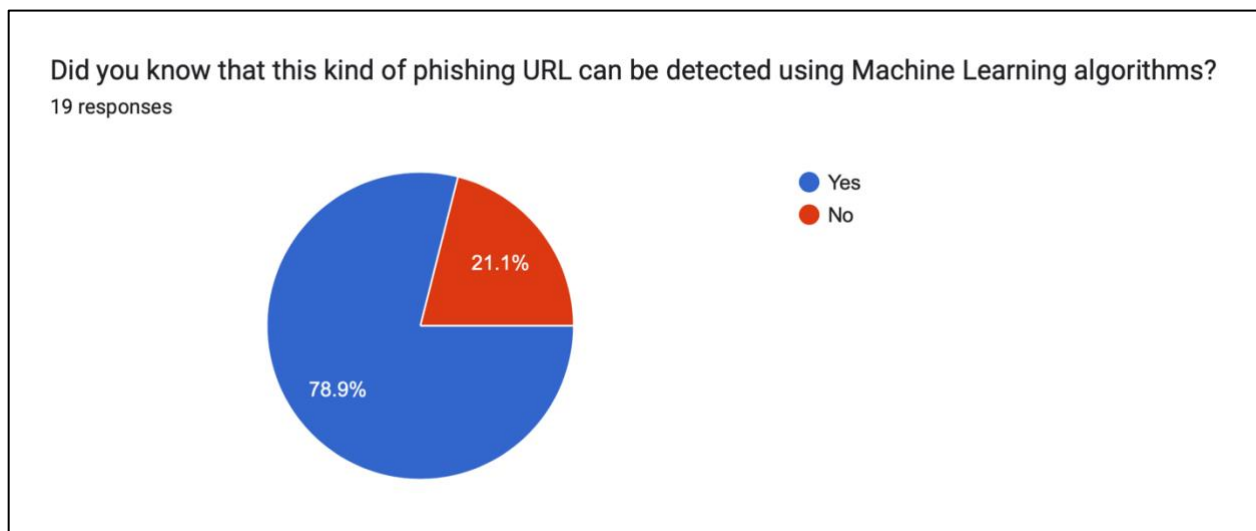


Figure 101: Pre-Survey Question no.7

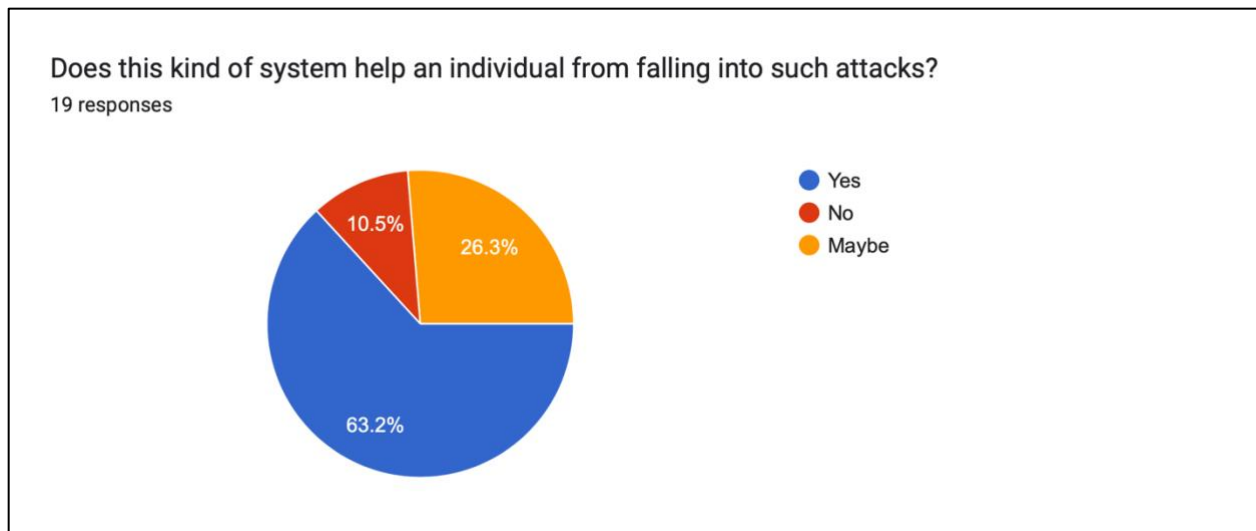


Figure 102: Pre-Survey Question no.8

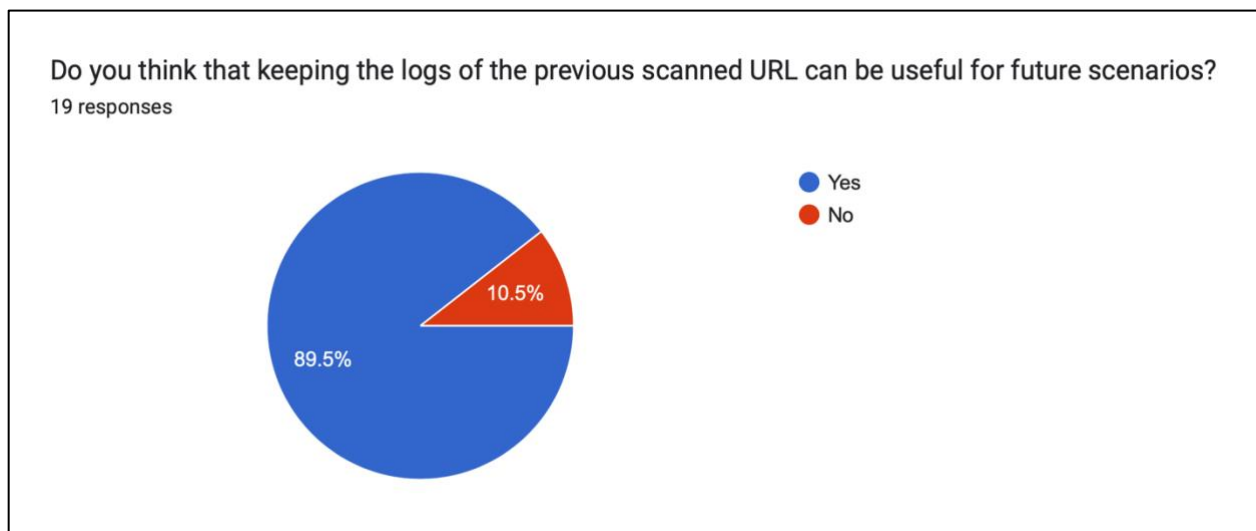


Figure 103: Pre-Survey Question no.9

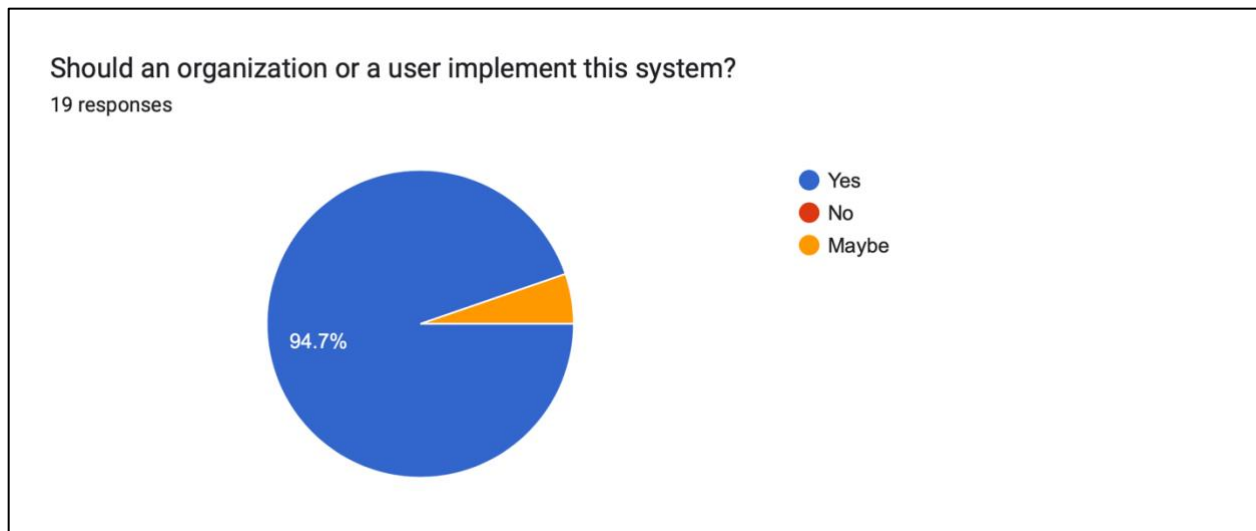


Figure 104: Pre-Survey Question no.10

Back To: [3.4.1 Pre-Survey Results](#)


7.5 Post-Survey

7.5.1 Post-Survey Form

FYP Post-Survey Form

This is a post-survey form for the FYP topic "Phishing detection using Machine Learning". The main aim of this project is to mitigate the global problem faced by internet users by providing them with a tool that can identify phishing URLs and legitimate trusted URLs using machine learning and help individuals minimize the risk of being victims of such phishing attacks.

NOTE: The collected data will be kept private and solely used for academic purposes only.

rollndope@gmail.com [Switch accounts](#) 

* Indicates required question

Email *

Your email address

Full Name *

Your answer

Figure 105: Questions in the post-survey form (a).

Phone Number

Your answer

Profession *

Your answer

Organization Name *

Your answer

Do you know what a phishing attack is and how it works? *

☐ Yes

☐ No

Figure 106: Questions in the post-survey form (b).

Was the system user-friendly and easy to navigate? *

☐ Yes

☐ No

On a scale of 1 to 10, how satisfied are you with the overall performance of the phishing detection system? *

1 2 3 4 5 6 7 8 9 10

Poor ☐ ☐ ☐ ☐ ☐ ☐ ☐ ☐ ☐ ☐ Excelent

Did you find the dashboard useful for analyzing and visualizing both phishing and legit URLs? *

☐ Yes

☐ No

Figure 107: Questions in the post-survey form (c).

How accurate do you think the machine learning model was in detecting phishing attempts? *

1 2 3 4 5 6 7 8 9 10

Poor ☐ ☐ ☐ ☐ ☐ ☐ ☐ ☐ ☐ ☐ Excelent

Did you find the system helps reduce the number of successful phishing attacks? *

☐ Yes

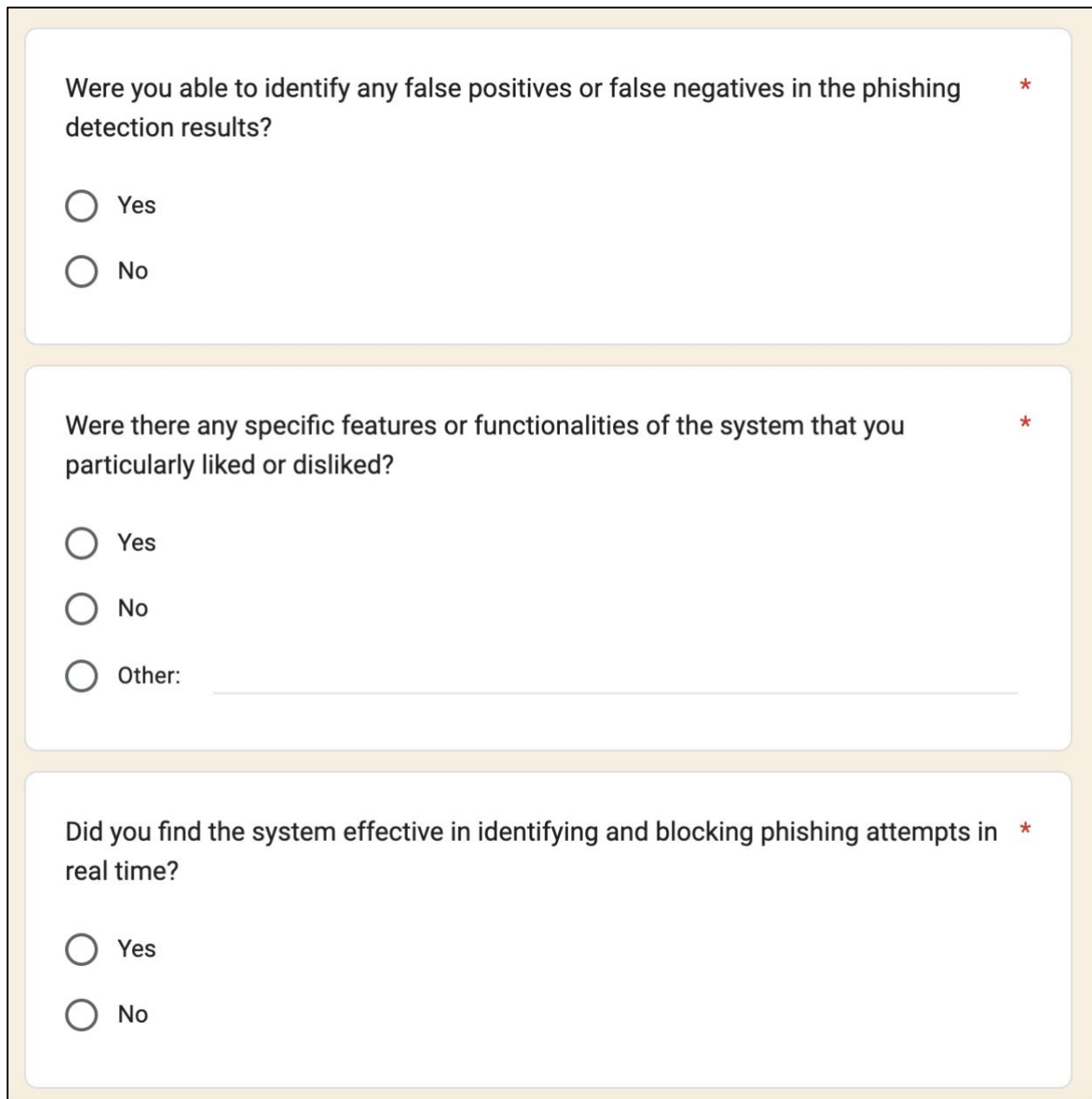
☐ No

Would you recommend this system to others for detecting phishing attempts? *

☐ Yes

☐ No

Figure 108: Questions in the post-survey form (d).



Were you able to identify any false positives or false negatives in the phishing detection results? *

☐ Yes

☐ No

Were there any specific features or functionalities of the system that you particularly liked or disliked? *

☐ Yes

☐ No

☐ Other: _____

Did you find the system effective in identifying and blocking phishing attempts in real time? *

☐ Yes

☐ No

Figure 109: Questions in the post-survey form (e).

Were there any technical issues or bugs you encountered while using the system? *

☐ Yes

☐ No

Did the system require any significant changes or modifications to fit the specific needs of your organization? *

☐ Yes

☐ No

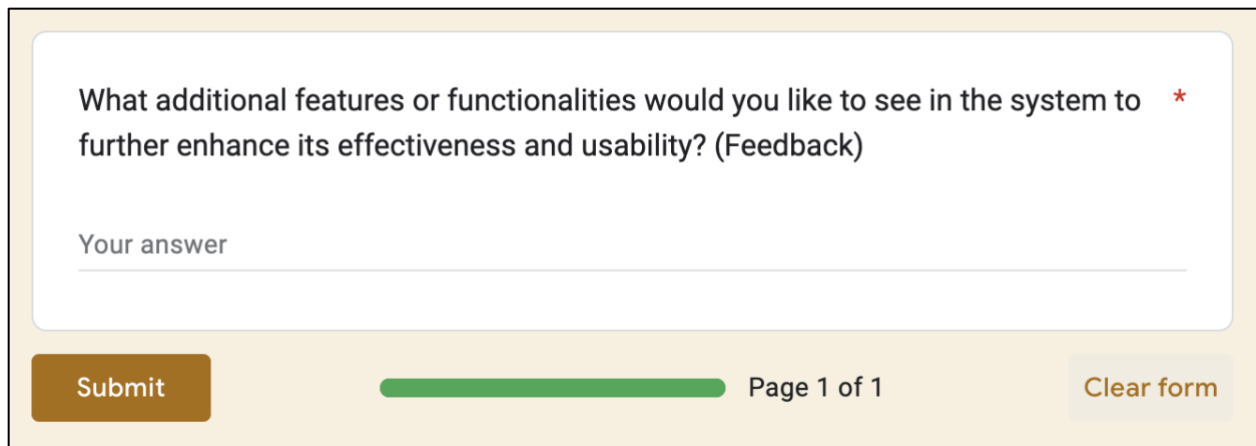
☐ Other: _____

Would you consider using this system as a long-term solution for detecting and preventing phishing attacks? *

☐ Yes

☐ No

Figure 110: Questions in the post-survey form (f).



What additional features or functionalities would you like to see in the system to further enhance its effectiveness and usability? (Feedback) *

Your answer

Submit

Page 1 of 1

Clear form

Figure 111: Questions in the post-survey form (g).

7.5.2 Sample of Filled Post-Survey Forms

The screenshot shows a Google Forms interface for a post-survey. At the top, it indicates '18 responses' and provides a 'Link to Sheets' button. A toggle switch for 'Accepting responses' is turned on. Below this, there are three tabs: 'Summary', 'Question', and 'Individual', with the 'Individual' tab currently selected. A dropdown menu shows the email address 'np01nt4s210101@islingtoncollege.edu.np'. Navigation arrows and a counter '10 of 18' are visible. Below the navigation bar, a message states 'Responses cannot be edited'. The main title of the form is 'FYP Post-Survey Form'. The body text describes the project's aim to mitigate phishing by using machine learning to identify phishing URLs and help users minimize risk. A note at the bottom states that the collected data will be kept private and used solely for academic purposes.

18 responses [Link to Sheets](#)

Accepting responses ☒

Summary Question **Individual**

np01nt4s210101@islingtoncollege.edu.np < 10 of 18 >

Responses cannot be edited

FYP Post-Survey Form

This is a post-survey form for the FYP topic "Phishing detection using Machine Learning". The main aim of this project is to mitigate the global problem faced by internet users by providing them with a tool that can identify phishing URLs and legitimate trusted URLs using machine learning and help individuals minimize the risk of being victims of such phishing attacks.

NOTE: The collected data will be kept private and solely used for academic purposes only.

Figure 112: Post-Survey Result from an individual (a).

Email *	np01nt4s210101@islingtoncollege.edu.np
Full Name *	Arya Amatya
Phone Number	9847694121
Profession *	Student

Figure 113: Post-Survey Result from an individual (b).

Organization Name *

Islington College

Do you know what a phishing attack is and how it works? *

☒ Yes

☐ No

Was the system user-friendly and easy to navigate? *

☒ Yes

☐ No

Figure 114: Post-Survey Result from an individual (c).

On a scale of 1 to 10, how satisfied are you with the overall performance of the phishing detection system? *

	1	2	3	4	5	6	7	8	9	10	
Poor	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input checked="" type="radio"/>	<input type="radio"/>	Excelent

Did you find the dashboard useful for analyzing and visualizing both phishing and legit URLs? *

☒ Yes

☐ No

Figure 115: Post-Survey Result from an individual (d).

How accurate do you think the machine learning model was in detecting phishing attempts? *

	1	2	3	4	5	6	7	8	9	10	
Poor	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input checked="" type="radio"/>	<input type="radio"/>	Excelent

Did you find the system helps reduce the number of successful phishing attacks? *

☒ Yes

☐ No

Figure 116: Post-Survey Result from an individual (e).

Would you recommend this system to others for detecting phishing attempts? *

☒ Yes

☐ No

Were you able to identify any false positives or false negatives in the phishing detection results? *

☒ Yes

☐ No

Figure 117: Post-Survey Result from an individual (f).

Were there any specific features or functionalities of the system that you particularly liked or disliked? *

☒ Yes

☐ No

☐ Other: _____

Did you find the system effective in identifying and blocking phishing attempts in real time? *

☒ Yes

☐ No

Figure 118: Post-Survey Result from an individual (g).

Were there any technical issues or bugs you encountered while using the system? *

☐ Yes

☒ No

Did the system require any significant changes or modifications to fit the specific needs of your organization? *

☒ Yes

☐ No

☐ Other: _____

Figure 119: Post-Survey Result from an individual (h).

Would you consider using this system as a long-term solution for detecting and preventing phishing attacks? *

☒ Yes

☐ No

What additional features or functionalities would you like to see in the system to further enhance its effectiveness and usability? (Feedback) *

With url other types such as files, hashes, folders must also be scanned.

Submitted 13/04/2023, 20:01

Figure 120: Post-Survey Result from an individual (i).

7.5.3 Post-Survey Result

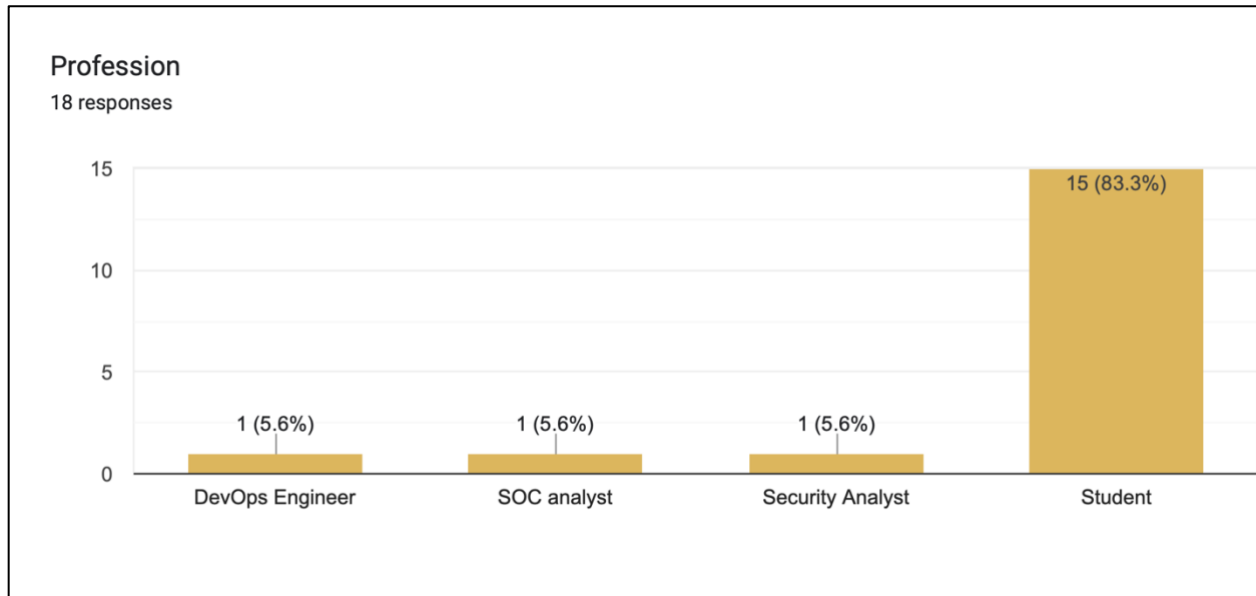


Figure 121: Profession of individuals who filled out the post-survey form.

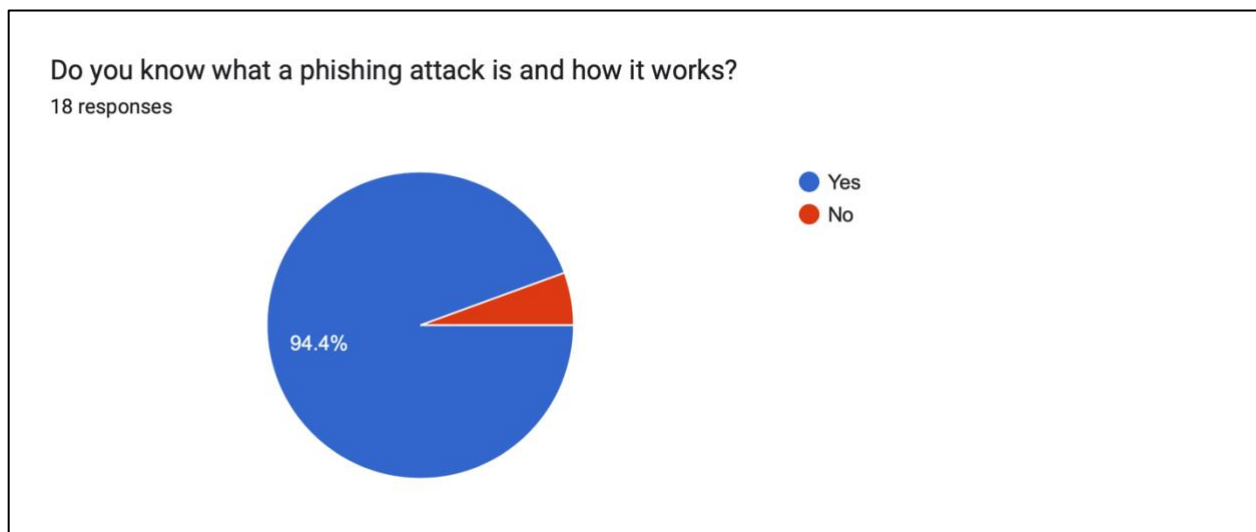
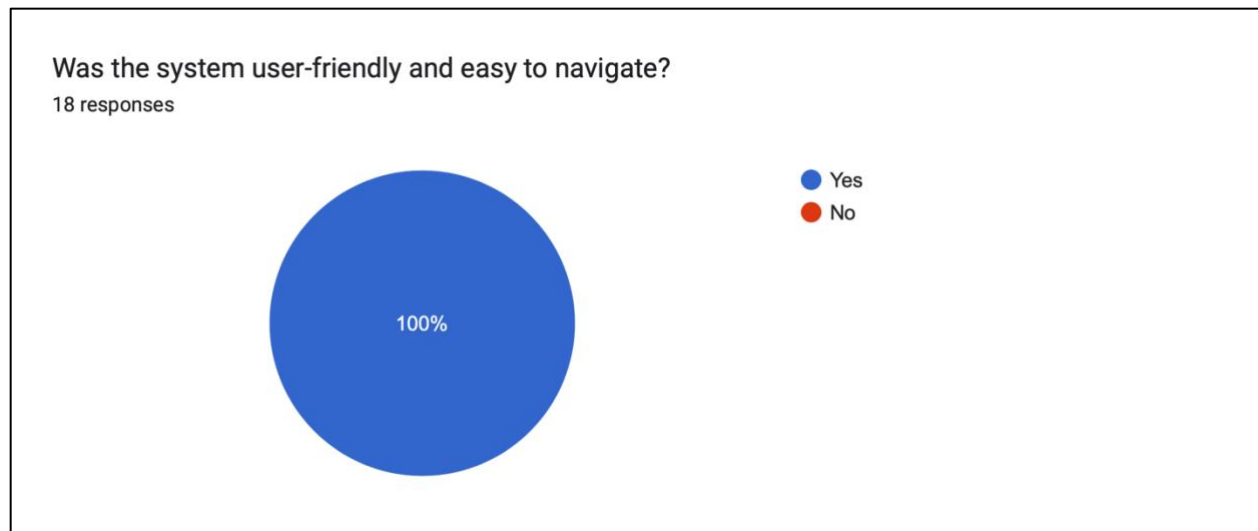
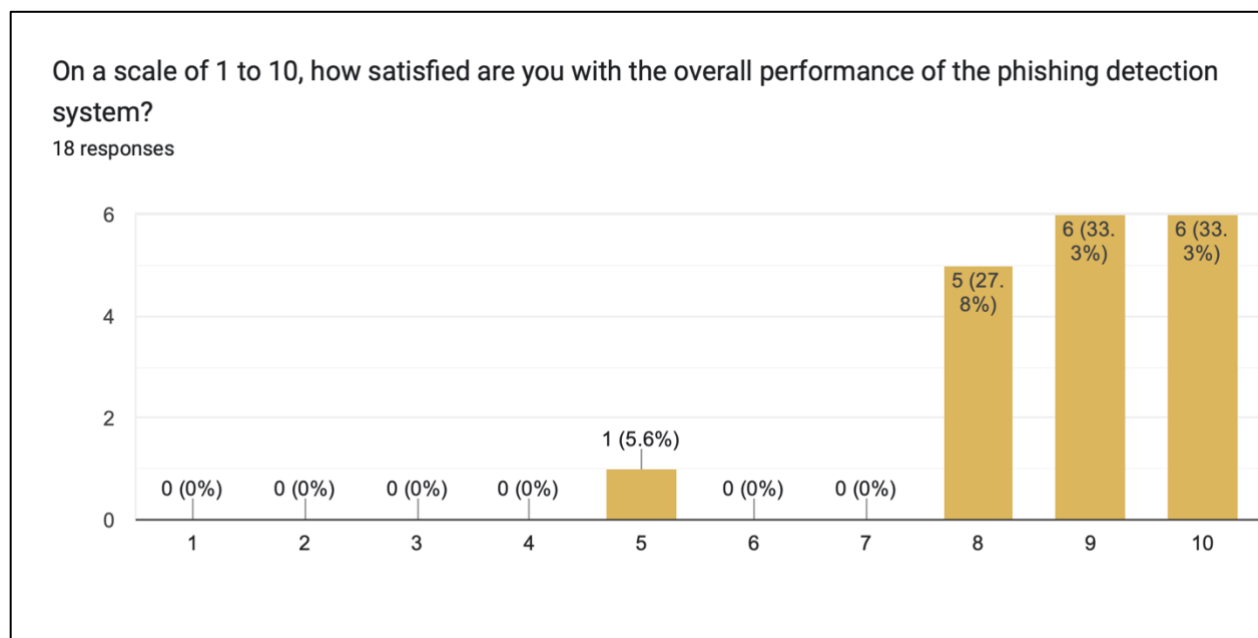


Figure 122: Post-Survey Question no.1

*Figure 123: Post-Survey Question no.2**Figure 124: Post-Survey Question no.3*

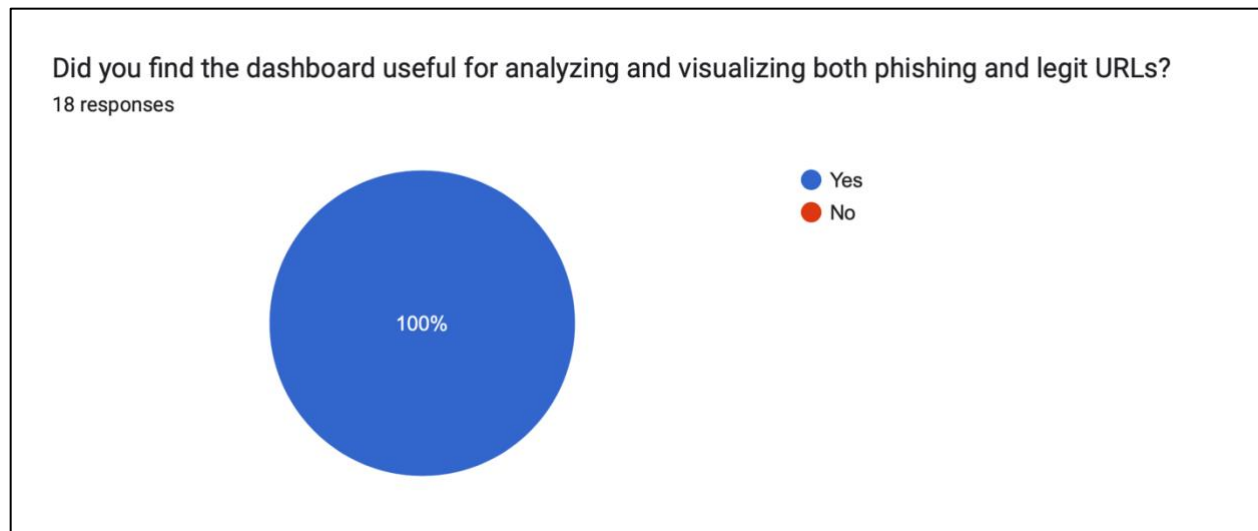


Figure 125: Post-Survey Question no.4

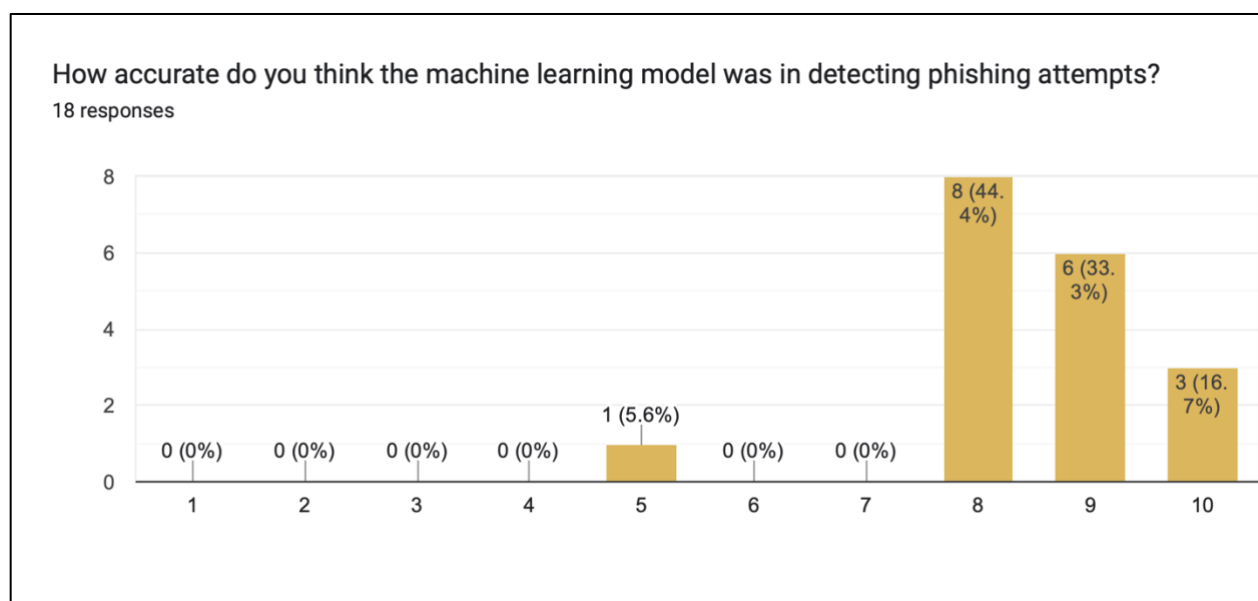


Figure 126: Post-Survey Question no.5

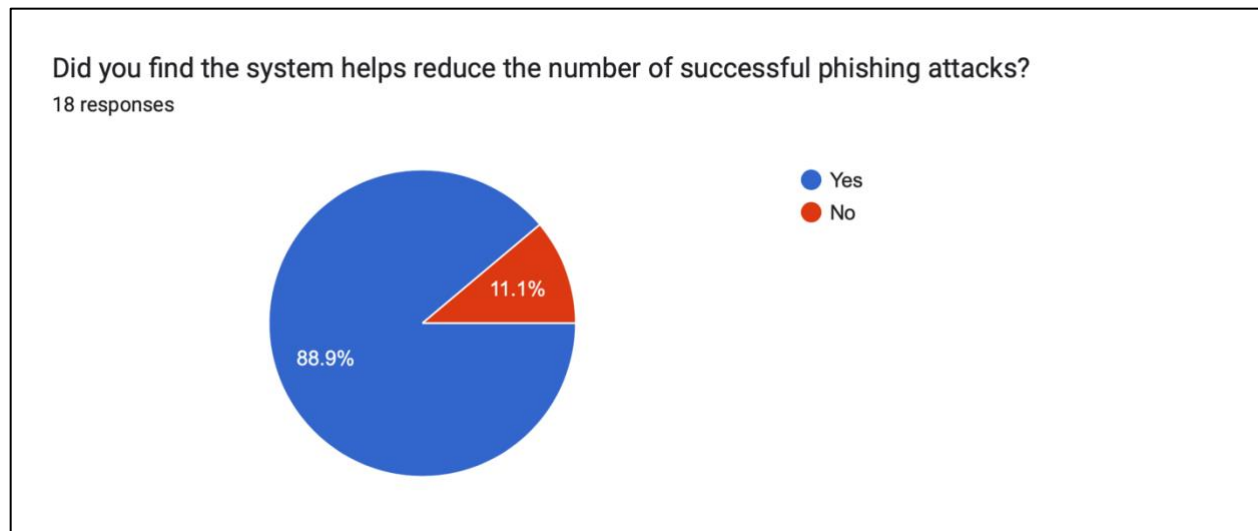


Figure 127: Post-Survey Question no.6

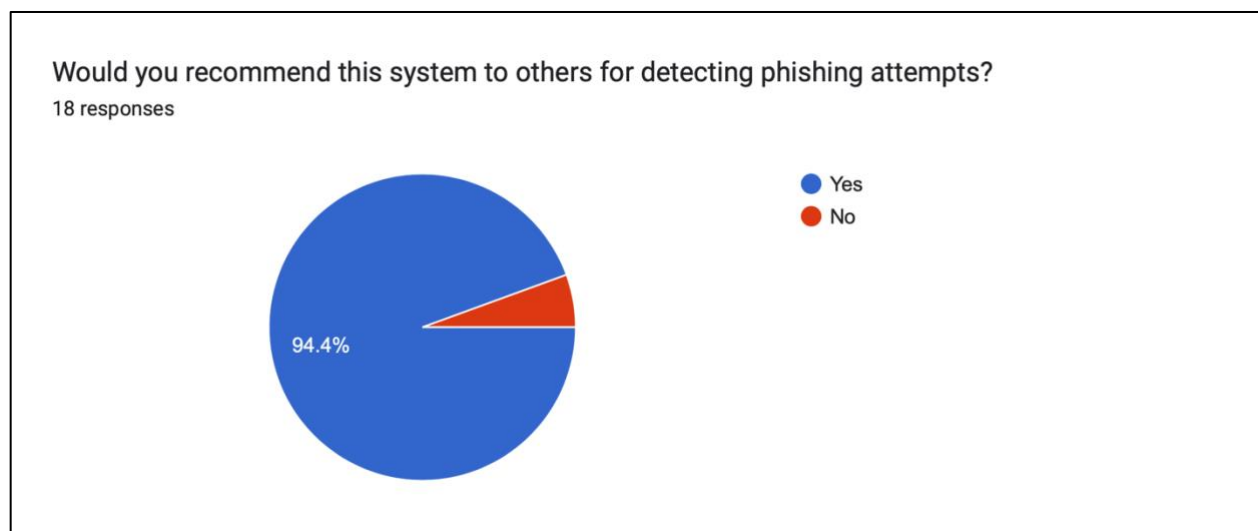


Figure 128: Post-Survey Question no.7

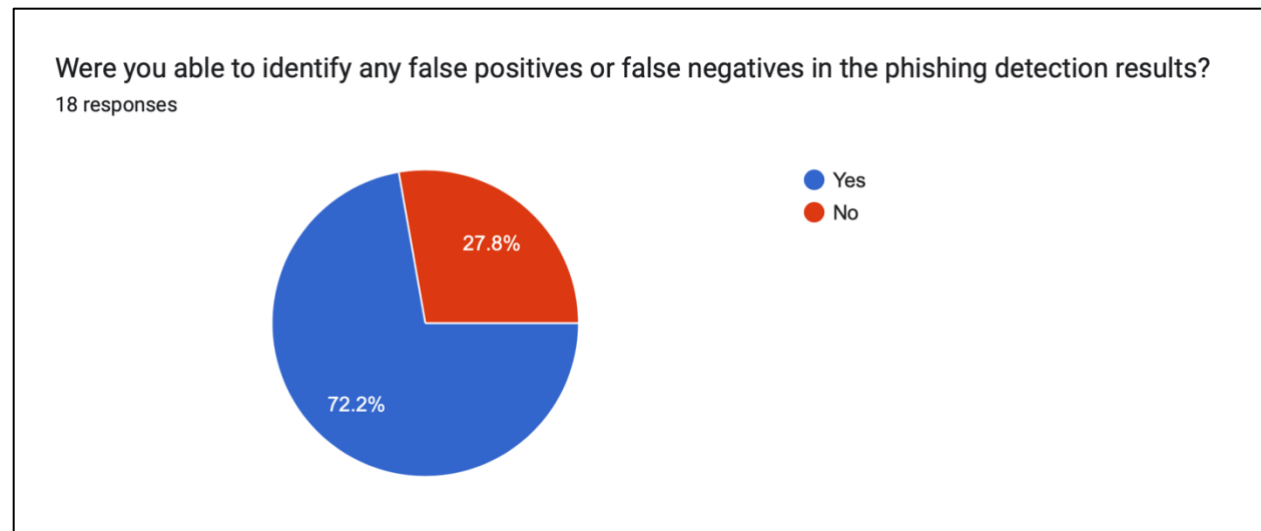


Figure 129: Post-Survey Question no.8

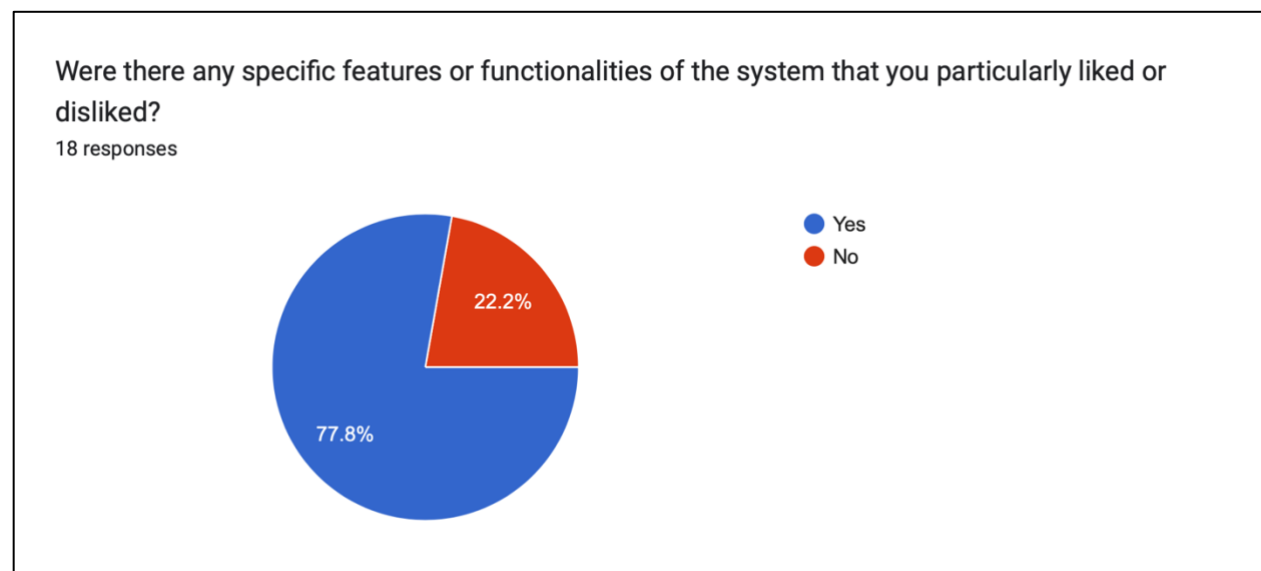


Figure 130: Post-Survey Question no.9

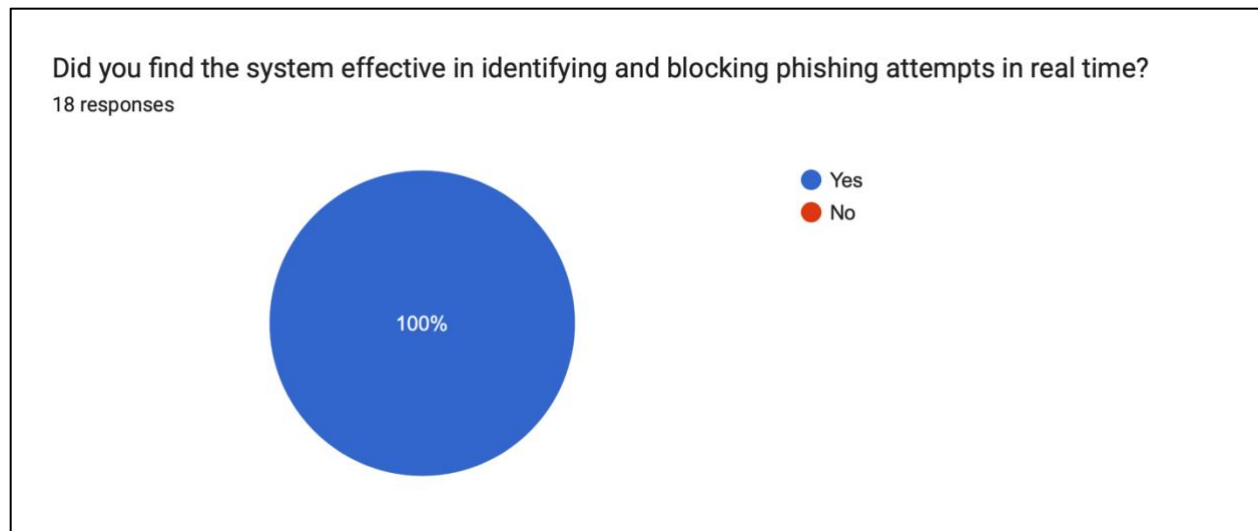


Figure 131: Post-Survey Question no.10

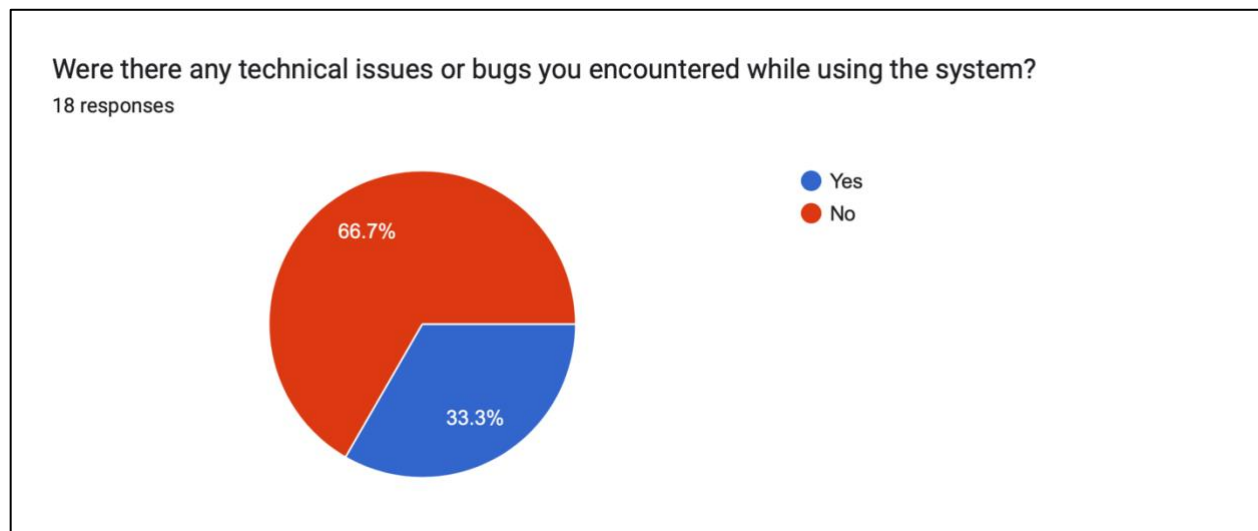


Figure 132: Post-Survey Question no.11

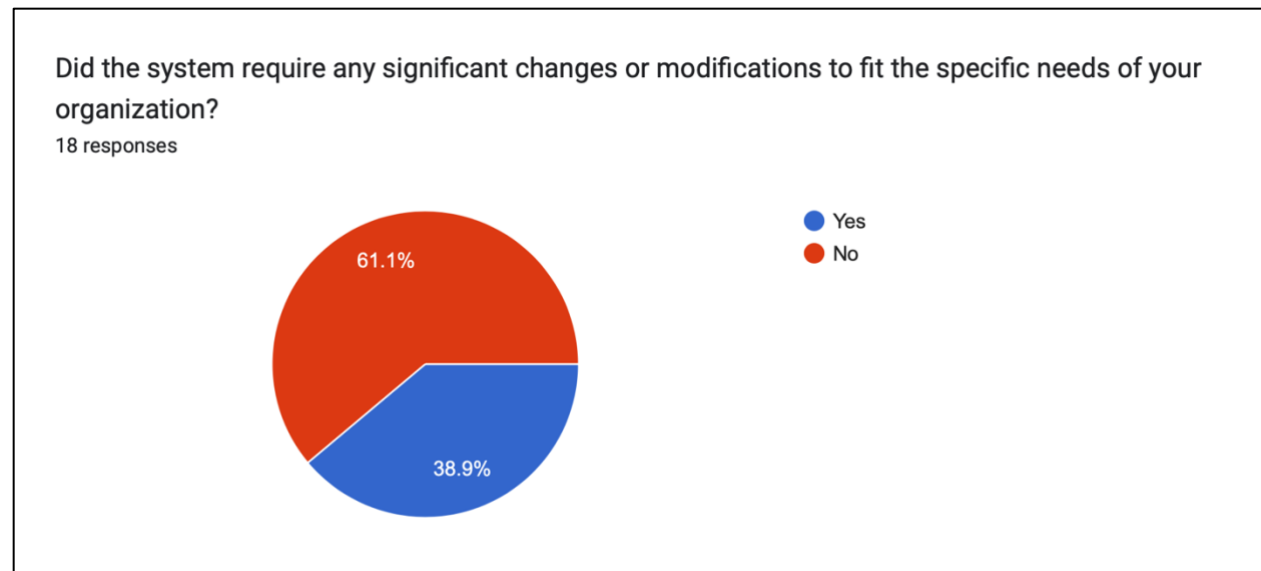


Figure 133: Post-Survey Question no.12

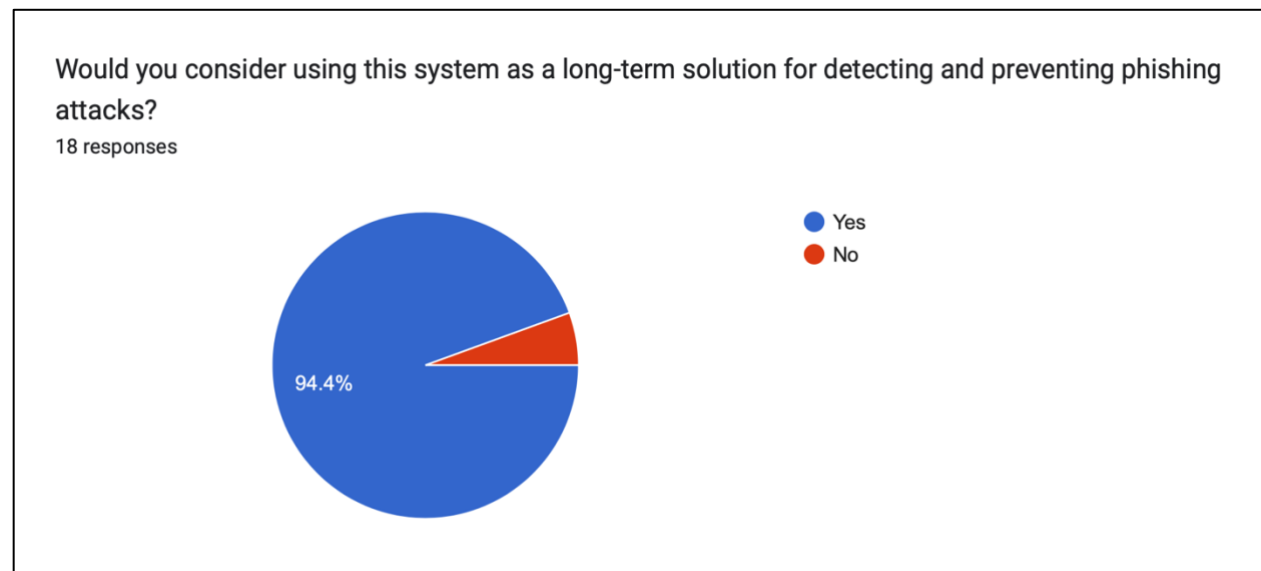


Figure 134: Post-Survey Question no.13

Back To: [3.4.2 Post-Survey Results](#)

7.6 Hardware and Software Requirements

7.6.1 Hardware

- A computer with a stable internet connection

7.6.2 Software

The software which is required for this project is:

- **React Framework** to develop the frontend GUI of the system.
- **GitHub** for version control.
- **VS-Code** for the text editor.
- **Python** is the main programming language for development.
- **Balsamiq** to create the wireframes.
- **Microsoft Office** which includes Word, Excel, and PowerPoint as documentation software.
- **draw.io** as designing tools.
- **API** such as Alexa API, VirusTotal API, google maps API, fetch API, APIIP.net API.

Back To: [3.6 Hardware and Software Requirements](#)

7.7 Designs

7.7.1 Gantt Chart

The below Gantt Chart represents the timeline of the project till the submission of the interim report.

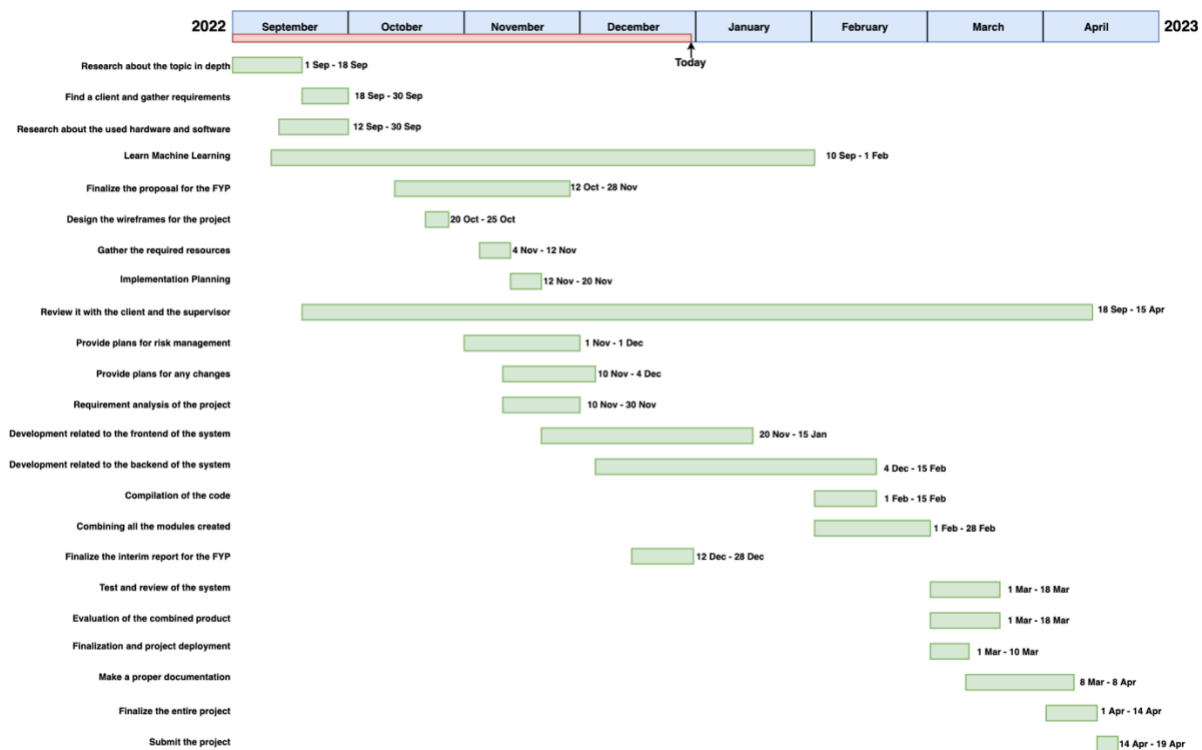


Figure 135: Old Gantt Chart.

Table 29: Tabular form of the dates in the old Gantt Chart.

Tasks	Start Date	End Date	Duration
Research about the topic in depth	01/09/2022	18/09/2022	18 Days
Find a client and gather requirements	18/09/2022	30/09/2022	13 Days
Research about the used hardware and software	12/09/2022	30/09/2022	19 Days
Learn Machine Learning	10/09/2022	01/02/2023	143 Days
Finalize the proposal for the FYP	12/10/2022	28/11/2022	48 Days
Design the wireframes for the project	20/10/2022	25/10/2022	6 Days
Gather the required resources	04/11/2022	12/11/2022	9 Days
Implementation Planning	12/11/2022	20/11/2022	9 Days
Review it with the client and the supervisor	18/09/2022	15/04/2023	209 Days
Provide plans for risk management	01/11/2022	01/12/2022	31 Days
Provide plans for any changes	10/11/2022	04/12/2022	25 Days
Requirement analysis of the project	10/11/2022	30/11/2022	21 Days
Development related to the frontend of the system	20/11/2022	15/01/2023	56 Days
Development related to the backend of the system	04/11/2022	15/02/2023	103 Days
Compilation of the code	01/02/2023	15/02/2023	15 Days
Combining all the modules created	01/02/2023	28/02/2023	28 Days
Finalize the interim report for the FYP	12/12/2022	28/12/2022	16 Days
Test and review of the system	01/04/2023	18/04/2023	18 Days
Evaluation of the combined product	01/04/2023	18/04/2023	18 Days
Finalization and project deployment	01/04/2023	10/04/2023	10 Days
Make a proper documentation	08/03/2023	08/04/2023	32 Days
Finalize the entire project	01/04/2023	14/04/2023	14 Days
Submit the project	14/04/2023	19/04/2023	6 Days

Back To: [3.8.5 Gantt Chart](#)

7.7.2 Work Breakdown Structure

A Work Breakdown Structure is a deliverable-oriented hierarchical decomposition of the work to be executed by the project team to accomplish the project objectives and create the required deliverables. The Work Breakdown Structure is developed to establish a common understanding of the project scope. It is a hierarchical description of the work that must be done to complete the deliverables of a project. (Visual Paradigm, 2020)

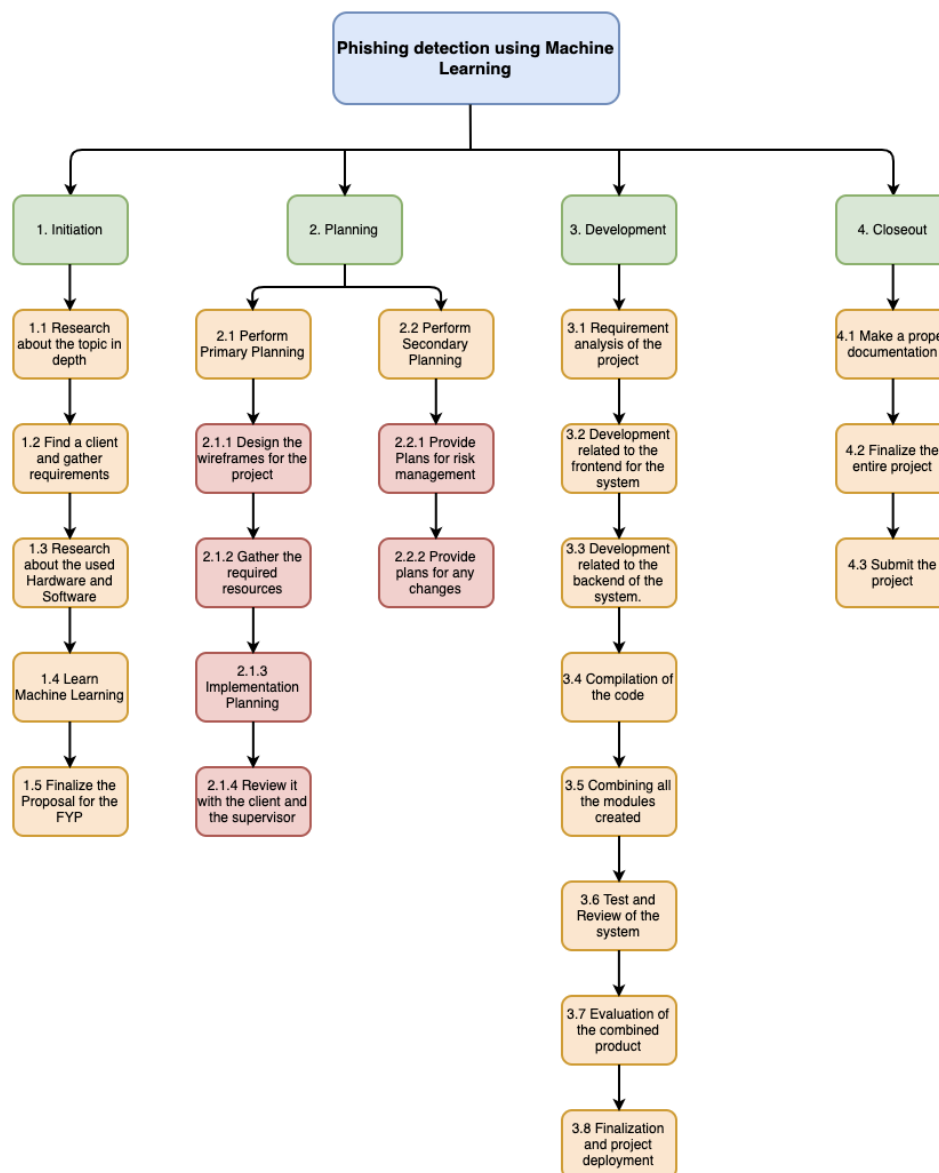


Figure 136: Work breakdown structure.

7.7.3 Wireframe

Wireframing is a UX design technique that allows UX designers to define and arrange the information hierarchy of their design for a website, app, or product. The concept of making a wireframe focuses on how the designer or client wants the user to take in information on a website. (Allabarton, Rosie, 2022)



Figure 137: Wireframe for the homepage of the system.

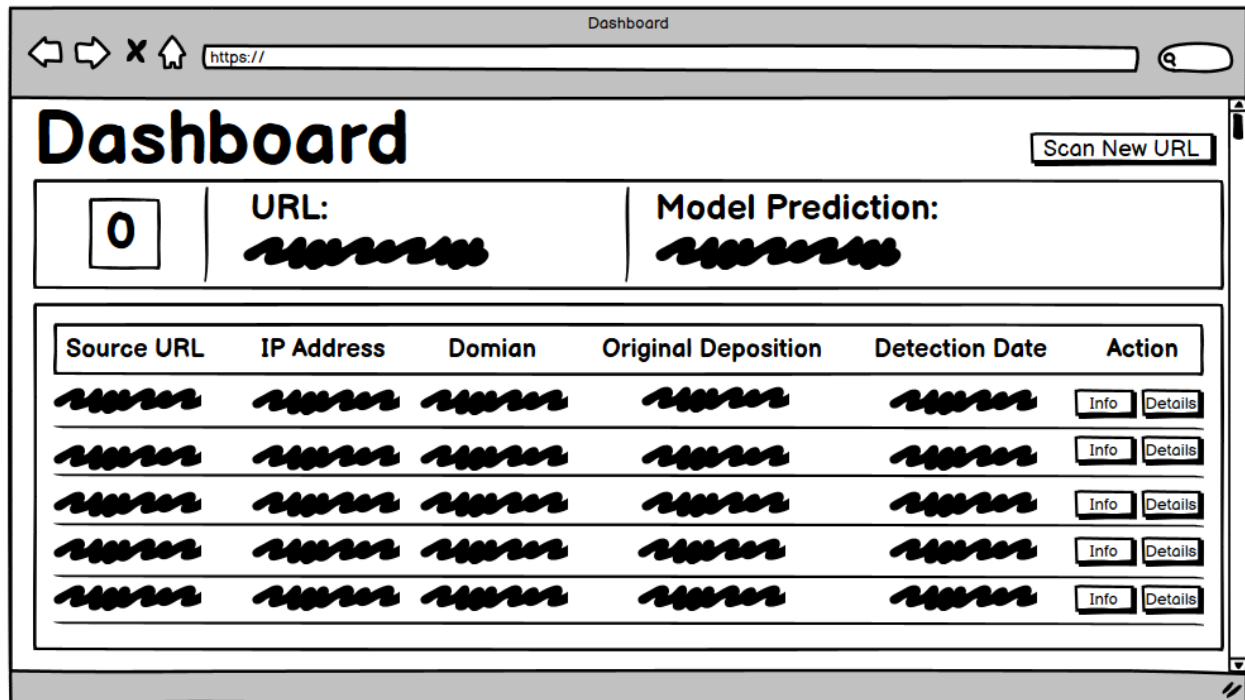


Figure 138: Wireframe for the Dashboard page of the system.

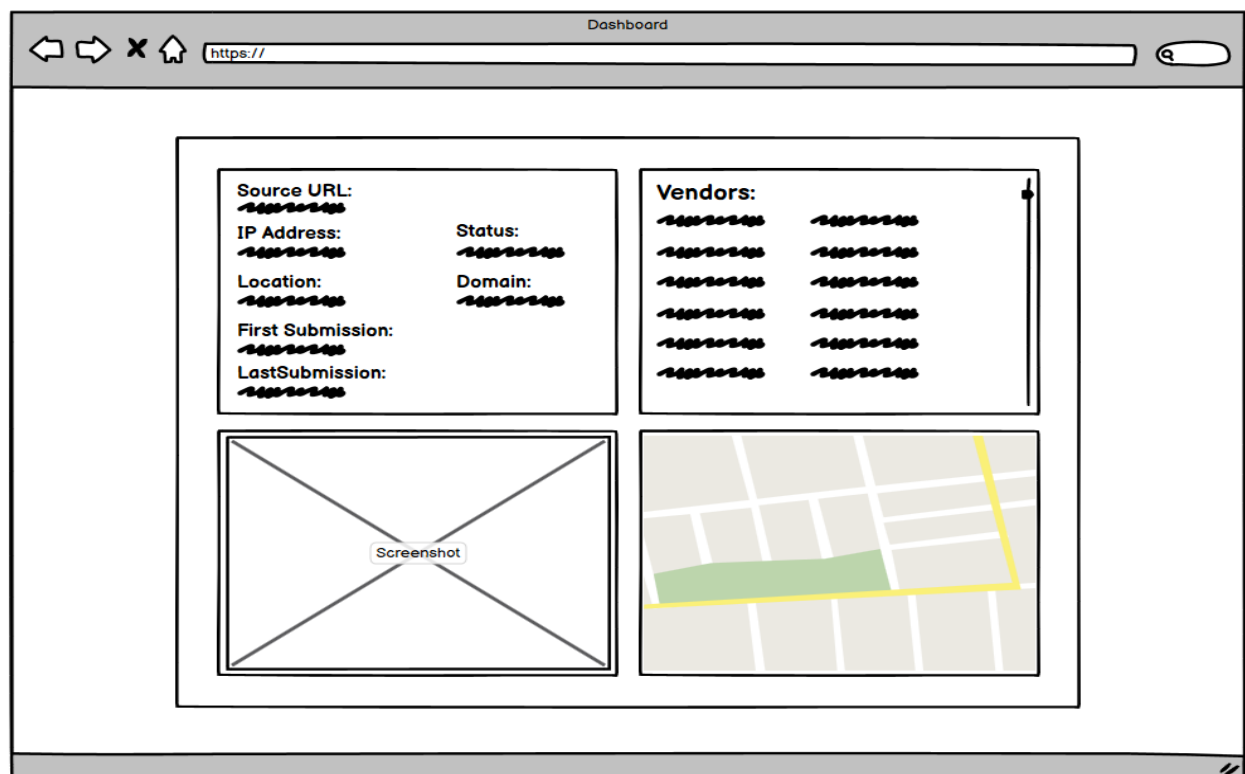


Figure 139: Wireframe for the Details page of the system.

Back To: [3.8.4 Wireframes](#)

7.8 Milestones

The milestones for this project are listed below:

- **Milestone 1:** Topic Finalization. (18th September)
- **Milestone 2:** Client Finalization and gathering requirements. (30th September)
- **Milestone 3:** Proposal Submission. (28th November)
- **Milestone 4:** Creating wireframes for the system. (25th October)
- **Milestone 5:** Completion of Interim Report. (28th December)
- **Milestone 6:** Complete development of the dataset and feature extraction. (10th December)
- **Milestone 7:** Complete development of the Machine Learning related to the project. (20th January)
- **Milestone 8:** Complete development of the GUI. (15th February)
- **Milestone 9:** Finalize Development. (5th March)
- **Milestone 10:** Complete development of the PDML system. (11th March)
- **Milestone 11:** Complete testing and review. (13th March)
- **Milestone 12:** Complete and finalize the documentation. (14th April)
- **Milestone 13:** Submit the Project. (19th April)

7.9 Progress Review Table

Table 30: Progress Table of the Project.

S.N.	Tasks	Status	Progress (%)	Completed Time
1	Research about the topic in depth	Completed	100%	Early
2	Find a client and gather requirements	Completed	100%	On-Time
3	Research about the used hardware and software	Completed	100%	On-Time
4	Learn Machine Learning	Completed	100%	On-Time
5	Finalize the proposal for the FYP	Completed	100%	Early
6	Design the wireframes for the project	Completed	100%	Early
7	Gather the required resources	Completed	100%	On-Time
8	Implementation Planning	Completed	100%	On-Time
9	Review it with the client and the supervisor	Completed	100%	On-Time
10	Provide plans for risk management	Completed	100%	On-Time
11	Provide plans for any changes	Completed	100%	On-Time
12	Provide plans for any changes	Completed	100%	On-Time
13	Requirement analysis of the project	Completed	100%	On-Time
14	Development related to the frontend of the system	Completed	100%	Late
15	Development related to the backend of the system	Completed	100%	Late
16	Compilation of the code	Completed	100%	On-Time
17	Combining all the modules created	Completed	100%	On-Time
18	Test and review of the system	Completed	100%	On-Time
19	Evaluation of the combined product	Completed	100%	On-Time
20	Finalization and project deployment	Completed	100%	On-Time
21	Make a proper interim report	Completed	100%	On-Time
22	Finalize the entire project	Completed	100%	On-Time
23	Submit the project	Completed	100%	On-Time

7.10 Expected Outcomes and Deliverables

7.10.1 Project Expected Outcomes

The goal of this project is to create an ML system capable of detecting phishing URLs with high accuracy avoiding false positives and negatives in real time. The approach will include a user-friendly dashboard for quick analysis and visualization of discovered phishing and legitimate URLs. The approach is predicted to effectively minimize the frequency of successful phishing attempts and to be beneficial to both people and enterprises. The accuracy, efficiency, and user-friendliness of the system and how well it operates the demands of its intended users will be used to determine the project's effectiveness.

7.10.2 Project Deliverables

This project's intended deliverables include a working ML system that can detect phishing URLs in real-time and also have a dashboard that allows users to evaluate and display the phishing and genuine URLs that have been discovered. The deliverables should also contain detailed documentation explaining how the system works and how to utilize the dashboard. Furthermore, the system should be rigorously evaluated to ensure its reliability, efficiency, and usability. Finally, a strategy for ongoing maintenance and support should be supplied to ensure the system's continuous efficacy.

7.11 Project Risk, Threats, and Contingency plans

7.11.1 Risk and Threats

The project risk and threats are the collections of events associated with it that obstruct the effective completion of the project. Some of the risks that I may face while completing my project, as well as mitigation methods, are stated below:

- One of the main drawbacks of URL blacklisting and the heuristic method is computational complexity and it will keep on becoming more complex with an increasing number of phishing websites.
- While running tests the system where the project is being developed might be on the verge of being a victim of phishing websites.
- The accuracy of the system detecting both phishing and legitimate might be low in the earlier stages of development.
- The list of phishing URLs used in the project is very sensitive information which if it fell on the wrong hand might be used to create a more advanced phishing system that goes undetected through any level of security measures.
- There may be random technical failures and errors throughout the project.
- One risk is the quality and quantity of data used to train the machine learning model. If the data is poor quality, it may negatively affect the accuracy of the model and the model may not perform well, leading to inaccurate detection of phishing URLs.

7.11.2 Contingency Plans

- Properly using and implementing devices to overcome technical failure.
- Experimenting with various strategies of machine learning to optimize the model.
- Discussing and taking advice from industry experts and supervisors on how the security level of the system can be optimized.
- The test should be run on a sandbox environment.

7.12 Sample Codes

7.12.1 Back-End Code

- feature extraction.ipynb

```
[179] import pandas as pd

Python

[180] #loading the phishing URLs data to dataframe
data_P = pd.read_csv('data/phishing_dataset.csv')
data_P.head()

Python
```

Figure 140: Feature extraction.ipynb code (a).

```
[181] #total number of datas
data_P.shape

... (14858, 8)

Python

[182] #loading the legit URLs data to dataframe
data_L = pd.read_csv('data/legit_dataset.csv')
data_L.columns = ['URLs']
data_L.head()

Python

[184] import requests
import urllib
from urllib.request import urlopen
import whois
import datetime
import time
import socket
import pandas as pd
import numpy as np
import ipaddress
import re
from urllib.parse import urlparse, urlencode
from bs4 import BeautifulSoup

Python

[185] #This function extracts the domain from the URL by parsing it using urlparse library and then returns the domain.
def getDomain(url):
    domain = urlparse(url).netloc
    if re.match(r"^\www.", domain):
        domain = domain.replace("www.", "")
    return domain

Python

[186] #This function checks if the given URL contains an IP address or not.
def havingIP(url):
    try:
        ipaddress.ip_address(url)
        ip = 1
    except:
        ip = 0
    return ip

Python
```

Figure 141: Feature extraction.ipynb code (b).

2. Domain Based Feature

```
#The function dns_record takes a URL as input and returns a binary value indicating whether the domain has a DNS record or not.
def dns_record(url):
    domain_name = urlparse(url).netloc
    try:
        rec = whois.whois(domain_name)
        return 1
    except:
        return 0
```

[201]

Python

This feature measures the popularity of the website by determining the number of visitors and the number of pages they visit. However, since phishing websites live for a short period of time, they may not be recognized by the Alexa database (Alexa the Web Information Company., 1996). By reviewing our dataset, we find that in worst scenarios, legitimate websites ranked among the top 100,000. Furthermore, if the domain has no traffic or is not recognized by the Alexa database, it is classified as "Phishing".

```
#This function takes in a URL as an input and returns the Alexa global rank of the website.
def web_traffic(url):
    try:
        rank = BeautifulSoup(urllib.request.urlopen("http://data.alexa.com/data?cli=10&dat=s&url=" + url).read(), "xml").find("REACH")['RANK']
        rank = int(rank)
        return rank
    except:
        return 0 #0 = Phishing
```

[202]

Python

Figure 142: Feature extraction.ipynb code (c).

```
def feature_extraction(url,label):

    feature = []

    #Search bar based features (14)
    feature.append(getDomain(url))
    feature.append(havingIP(url))
    feature.append(haveAtSign(url))
    feature.append(getLength(url))
    feature.append(getDepth(url))
    feature.append(redirection(url))
    feature.append(httpDomain(url))
    feature.append(tinyURL(url))
    feature.append(prefixSuffix(url))
    feature.append(ippresent(url))
    feature.append(https_token(url))
    feature.append(get_protocol_count(url))
    feature.append(get_protocol(url))
    feature.append(get_special_char_count(url))

    #Domain based features (3)
    dns = 0
    try:
        domain_name = whois.whois(urlparse(url).netloc)
    except:
        dns = 1

    feature.append(dns_record(url))
    feature.append(web_traffic(url))
    feature.append(domainAge(url))

    feature.append(label)

    return feature
```

[208]

Python

Figure 143: Feature extraction.ipynb code (d).

Extracting feature to phishing URL's

```
[209] #Collecting 50 Phishing URLs randomly
phishUrl = data_P.copy()
phishUrl = phishUrl.reset_index(drop=True)
phishUrl.head()

Python
```

```
[212] #Extracting the features and storing them in a list
phish_features = []
label = 1

for i in range(0, 14858):
    url = phishUrl['url'][i]
    phish_features.append(feature_extraction(url, label))

Python
```

```
[215] feature_names = ['Domain', 'IP', '@', 'URL Length', 'URL Depth', 'Redirection', 'https Domain', 'TinyURL',
    'Prefix/Suffix', 'IP Present', 'HTTPS Token', 'Protocol Count', 'Protocol', 'Special Character Count',
    'DNS Record', 'Web Traffic', 'Domain Age', 'Label']
phishing = pd.DataFrame (phish_features, columns= feature_names)
phishing.head()

Python
```

```
[216] # Storing the extracted phishing URLs fatures to csv file
phishing.to_csv('data/phishing.csv', index= False)

Python
```

Extracting feature to legitimate URL's

```
[217] #Collecting 50 Legitimate URLs randomly
legitUrl = data_L.copy()
legitUrl = legitUrl.reset_index(drop=True)
legitUrl.head()

Python
```

Figure 144: Feature extraction.ipynb code (e).

```
[219] #Extracting the features and storing them in a list
legit_features = []
label = 0

for i in range(0, 35377):
    url = legitUrl['URLs'][i]
    legit_features.append(feature_extraction(url, label))

Python
```

```
[220] feature_names = ['Domain', 'IP', '@', 'URL Length', 'URL Depth', 'Redirection', 'https Domain', 'TinyURL',
    'Prefix/Suffix', 'IP Present', 'HTTPS Token', 'Protocol Count', 'Protocol', 'Special Character Count',
    'DNS Record', 'Web Traffic', 'Domain Age', 'Label']
legitimate = pd.DataFrame (legit_features, columns= feature_names)
legitimate.head()

Python
```

```
[221] # Storing the extracted legitimate URLs fatures to csv file
legitimate.to_csv('data/legitimate.csv', index= False)

Python
```

```
[222] #Concatenating the dataframes into one
urldata = pd.concat([legitimate, phishing]).reset_index(drop=True)
urldata.head()

Python
```

```
[225] # Storing the data in CSV file
urldata.to_csv('data/combined_data.csv', index=False)

Python
```

Figure 145: Feature extraction.ipynb code (f).

- **modeltrain.ipynb**

```
[43] #importing basic packages
import sklearn
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt

Python
```

```
▷ [44] #Loading the data
data = pd.read_csv('data/combined_data.csv')
data.head()

Python
```

Figure 146: modeltrain.ipynb code (a).

Visualization of the data

+ Code + Markdown

```
[48] #Plotting the data distribution
data.hist(bins = 50,figsize = (30,30))
plt.show()

Python
```

```
▷ [49] #Correlation heatmap
plt.figure(figsize=(15,15))
sns.heatmap(data.corr())
plt.show()

Python
```

```
[51] #Dropping the Domain column
data = data.drop(['Domain'], axis = 1).copy()

Python
```

```
▷ [52] #checking the data for null or missing values
data.isnull().sum()

Python
```

```
[54] # shuffling the rows in the dataset so that when splitting the train and test set are equally distributed
data = data.sample(frac=1).reset_index(drop=True)
data.head()

Python
```

Figure 147: modeltrain.ipynb code (b).

Splitting the data

```
[62] # Separating & assigning features and target columns to X & y
y = data['Label']
x = data.drop('Label',axis=1)
x.shape, y.shape

... ((50235, 16), (50235,))

[63] # Splitting the dataset into train and test sets: 70-30 split
from sklearn.model_selection import train_test_split

x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.3, random_state = 12)
x_train.shape, x_test.shape

... ((35164, 16), (15071, 16))
```

Figure 148: modeltrain.ipynb code (c).

BULK TRAINING AND TESTING

```
[64] from sklearn.metrics import accuracy_score
from sklearn.tree import DecisionTreeClassifier #Decision Tree model
from sklearn.ensemble import RandomForestClassifier #Random Forest model
from sklearn.neural_network import MLPClassifier #Multilayer Perceptrons model
from sklearn.svm import SVC
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.ensemble import AdaBoostClassifier

Python

BULK TRAINING

[65] models = [DecisionTreeClassifier(),RandomForestClassifier(), MLPClassifier(), SVC(), LogisticRegression(),GradientBoostingClassifier(),AdaBoostClassifier()]
for model in models:
    model.fit(x_train,y_train)
    print(str(model) , ' -> train_accuracy: ' ,model.score(x_train,y_train))

Python

BULK TESTING

[66] models = [DecisionTreeClassifier(),RandomForestClassifier(), MLPClassifier(), SVC(), LogisticRegression(),GradientBoostingClassifier(),AdaBoostClassifier()]
for model in models:
    model.fit(x_train,y_train)
    print(str(model) , ' -> test_accuracy: ' ,model.score(x_test,y_test))

Python
```

Figure 149: modeltrain.ipynb code (d).

```
[91] forest = RandomForestClassifier()
    forest.fit(x_train, y_train)
Python

...
▼ RandomForestClassifier
RandomForestClassifier()

# save Random Forest model to file
import pickle
pickle.dump(forest, open("RandomForestClassifier.pickle.dat", "wb"))
[92]
Python

# load model from file
loaded_model = pickle.load(open("RandomForestClassifier.pickle.dat", "rb"))
loaded_model
[93]
Python

...
▼ RandomForestClassifier
RandomForestClassifier()

with open('classifier_model.pkl', 'wb') as f:
    pickle.dump(forest, f)
[94]
Python
```

Figure 150: modeltrain.ipynb code (e).

- main.py

```

1  import pickle
2  import pandas as pd
3  import numpy as np
4  import urllib
5  from urllib.request import urlopen
6  from urllib.parse import urlparse
7  from difflib import SequenceMatcher
8  import whois
9  import datetime
10 import socket
11 import ipaddress
12 import re
13 import joblib
14 from urllib.parse import urlparse
15 from urllib.parse import urlencode
16 from bs4 import BeautifulSoup
17 from flask import Flask, request
18 from flask_cors import CORS
19 from virustotalapi import get_virus_total_data
20 from newlocation import get_ip_details
21
22 # Load the pickled model
23 with open('classifier_model.pkl', 'rb') as f:
24     model = pickle.load(f)
25
26 # 1.This function extracts the domain from the URL by parsing it using urlparse library and then returns the domain.
27 def getDomain(url):
28     domain = urlparse(url).netloc
29     if re.match(r"^\www.", domain):
30         domain = domain.replace("www.", "")
31     return domain
32
33 # 2.This function checks if the given URL contains an IP address or not.
34 def havingIP(url):
35     try:
36         ipaddress.ip_address(url)
37         ip = 1
38     except:
39         ip = 0
40     return ip
41
42 # 3.This function checks if the given URL contains an '@' symbol, which is typically not present in a standard URL.
43 def haveAtSign(url):
44     if "@" in url:
45         at = 1
46     else:
47         at = 0
48     return at
49
50 # 4.This function takes a URL and returns a binary value indicating whether its length is greater than or equal to 54 or not.
51 def getLength(url):
52     if len(url) < 54:
53         length = 0
54     else:
55         length = 1
56     return length

```

Figure 151: main.py code (a).

```
169 # 17.The function domainAge takes a URL as input and returns the age of the domain in days.
170 def domainAge(url):
171     try:
172         domain_name = urlparse(url).netloc
173
174         whois_response = whois.whois(domain_name)
175         creation_date = whois_response.creation_date
176         creation_date = creation_date[0].date()
177         today = datetime.date.today()
178         days = (today - creation_date).days
179         if (days/30 < 6):
180             return 1
181         return 0
182     except:
183         return 0
184
185 # Define a function to extract features from a URL
186 def extract_features(url):
187     parsed_url = urlparse(url)
188     feature = []
189     feature.append(havingIP(url))
190     feature.append(haveAtSign(url))
191     feature.append(getLength(url))
192     feature.append(getDepth(url))
193     feature.append(redirection(url))
194     feature.append(httpDomain(url))
195     feature.append(tinyURL(url))
196     feature.append(prefixSuffix(url))
197     feature.append(ippresent(url))
198     feature.append(https_token(url))
199     feature.append(get_protocol_count(url))
200     feature.append(get_protocol(url))
201     feature.append(get_special_char_count(url))
202
203     dns = 0
204     try:
205         domain_name = whois.whois(urlparse(url).netloc)
206     except:
207         dns = 1
208
209     feature.append(dns_record(url))
210     feature.append(web_traffic(url))
211     feature.append(domainAge(url))
212
213     return feature
214
```

Figure 152: main.py code (b).

```

216 def detect_phish(url):
217     result = {}
218     result['domain'] = getDomain(url) #fetchs domain
219
220     if (url):
221         result['location_details'] = get_ip_details(url) #fetchs location details
222         result['ip-address'] = socket.gethostbyname(url.split('/')[2]) #fetchs IP address
223         result['virus-total-data'] = get_virus_total_data(url) #fetchs data from virustotal API
224
225         feature_vector = [havingIP(url), haveAtSign(url), getLength(url), getDepth(url), redirection(url), httpDomain(url), tinyURL(url), prefixSuffix(url), ippresent(url),
226                           https_token(url), get_protocol_count(url), get_protocol(url), get_special_char_count(url), dns_record(url), web_traffic(url), domainAge(url)]
227
228         prediction = model.predict([feature_vector])[0]
229
230         result["source_url"] = url
231         if prediction == 0:
232             result['predication'] = 'Legitimate'
233             return result
234         else:
235             result['predication'] = 'Phishing'
236             return result
237
238     return {"predication": "Failed: Url is not defined."}
239
240 # create a Flask application instance
241 app = Flask(__name__)
242
243 # enable Cross-Origin Resource Sharing (CORS) for the app
244 CORS(app)
245
246 # define a route for the '/search' endpoint and specify that it accepts only GET requests
247 @app.route('/search', methods=['GET'])
248 def search_url():
249
250     # call the detect_phish function with the URL passed as a query argument and return the result
251     return detect_phish(request.args.get("url"))
252
253 # start the Flask application server
254 if __name__ == '__main__':
255
256     # make the server available from any IP address and use port 9696
257     app.run(host='0.0.0.0', port=9696)

```

Figure 153: main.py code (c).

- newlocation.py

```
1  import socket
2  import requests
3  import json
4  import folium
5  import webbrowser
6  from selenium import webdriver
7  from selenium.webdriver.chrome.options import Options
8
9  def get_ip_details(url):
10
11     # Initialize an empty dictionary for storing the results
12     result = {}
13
14     # Append a trailing slash to the URL to avoid errors in the IP address lookup
15     url = url + "/"
16
17     # Create a string for use in the screenshot filename
18     url_for_screenshot = url.replace("https://", "")
19     url_for_screenshot = url_for_screenshot.replace(".", "_")
20     url_for_screenshot = url_for_screenshot.replace("/", ".png")
21
22     # Extract the domain name from the URL
23     domain_name = url.split('/')[2]
24
25     # Lookup the IP address associated with the domain name
26     ip_address = socket.gethostbyname(domain_name)
27
28     # Call the APIIP service to get additional IP details
29     access_key = '3b89c992-d5b9-4cd8-91ba-3d7ff961605f'
30     apiip = f'http://apiip.net/api/check?ip={ip_address}&accessKey={access_key}'
31
32     response = requests.get(apiip)
33
34     result = json.loads(response.text)
35
36     # Add the screenshot URL to the results dictionary
37     result['screenshot_url'] = url_for_screenshot
38
39     # Use Selenium and ChromeDriver to capture a screenshot of the website
40     options = Options()
41     options.headless = True
42
43     driver = webdriver.Chrome(options=options)
44     driver.get(url)
45     driver.save_screenshot("./phising-detection-frontend/public/"+url_for_screenshot)
46     driver.quit()
47
48     # Return the results dictionary
49     return result
```

Figure 154: newlocation.py code

- **virustotalapi.py**

```
1 import requests
2 import base64
3 from datetime import date
4
5 #loading the virustotal API
6 api = "https://www.virustotal.com/api/v3/urls/"
7
8 def get_virus_total_data(phish_detect_url):
9
10     #encoding the URL in Base64 format
11     url_id = base64.urlsafe_b64encode(phish_detect_url.encode()).decode().strip("=")
12     url = api + url_id
13     headers = {
14         "accept": "application/json",
15
16         #providing the API key
17         "x-apikey": "6da45fde1f9d2d396f1572e545ee8b8dbcdf6020ab1622d3a986c6a77c75a1b2"
18     }
19
20     response = requests.get(url, headers=headers)
21
22     #retrieving the relavant information from the json response
23     response_data = response.json()
24
25     #Storing the response in a dictionary
26     temp_response = {}
27     print(response_data)
28     temp_response['detectionDate'] = date.today()
29     temp_response['firstSubmission'] = response_data['data']['attributes']['first_submission_date']
30     temp_response['lastSubmission'] = response_data['data']['attributes']['last_submission_date']
31     temp_response['last_analysis_results'] = response_data['data']['attributes']['last_analysis_results']
32     temp_response['last_analysis_stats'] = response_data['data']['attributes']['last_analysis_stats']
33     return temp_response
```

Figure 155: *virustotalapi.py* code.

7.12.2 Front-End Code

- index.tsx

```

1  import { Form, Input, Row, Button, Col } from 'antd'
2  import Head from 'next/head'
3  import { useRouter } from 'next/router';
4
5  export default function Home() {
6    const router = useRouter();
7    return (
8      <
9        <Head>
10         <title>Phishing Detection FYP | Sarthak Rana</title>
11         <meta name="description" content="Generated by create next app" />
12         <meta name="viewport" content="width=device-width, initial-scale=1" />
13         <link rel="icon" href="/favicon.ico" />
14       </Head>
15       <main className='h-screen flex items-center flex-col justify-center gap-4 bg-slate-100 overflow-hidden'>
16         <h1 className='font-bold text-8xl text-black/70 -mt-16'>Welcome</h1>
17         <h2 className='font-medium text-3xl text-black/50 mb-4'>This site scans for phishing URLs!!</h2>
18         <Form className='w-full' onFinish={async (data) => { console.log(data);
19           try {
20             let result = await fetch(`http://localhost:8000/search?url=${encodeURIComponent(data.url)}`)
21             result = await result.json();
22             console.log(result)
23           } catch (error) {
24             console.log(error)
25           }
26         }
27         router.push('/dashboard')
28       >>
29       <Row className='flex justify-center' gutter={[24, 0]}>
30         <Col span={8}>
31           <Form.Item name="url" rules={[{
32             required: true, message: "Please Enter an URL!!!"
33           }]}>
34             <Input placeholder='Enter a URL:www.example.com' size='large' />
35           </Form.Item>
36         </Col>
37         <Col>
38           <Button size='large' type='primary' htmlType='submit'><span className='px-8 font-bold'>Scan</span></Button>
39         </Col>
40       </Row>
41     </Form>
42   </main>
43 </>
44 )
45 }

```

Figure 156: index.tsx code.

- **dashboard.tsx**

```
1  import Head from "next/head";
2  import { Table, Button, Modal, ConfigProvider } from "antd";
3  import { useRouter } from "next/router";
4  import { useEffect, useState } from "react";
5  import store from "store";
6
7  export default function Dashboard() {
8    const router = useRouter();
9    useEffect(() => {
10      const initialData = store.get("dashboardDb") ?? [];
11      if (store.get("searchScan")) {
12        initialData.push(store.get("searchScan"));
13        setActiveDetails(store.get("searchScan"));
14        store.set("searchScan", undefined);
15      }
16      setDataSource(initialData)
17      if ((Object.keys(activeDetails).length === 0)) {
18        setActiveDetails(initialData[0])
19      }
20      store.set("dashboardDb", initialData)
21      setFirstRender(false);
22    }, [])
23
24    const [activeDetails, setActiveDetails] = useState({});
25
26    const [showModal, setShowModal] = useState(false)
27    const [firstRender, setFirstRender] = useState(true);
28
29    const [dataSource, setDataSource] = useState([])
30    return <>
31      <Head>
32        <title>DashBoard | Sarthak Rana</title>
33      </Head>
```

Figure 157: dashboard.tsx code (a).

```

124     <Table
125       columns={[{
126         dataIndex: "source_url",
127         title: "Source Url",
128         render: (data: string) => {
129           return <a href={data} className="text-blue-400">{data}</a>
130         }
131       }],
132       { dataIndex: "ip-address", title: "IP Address" },
133       { dataIndex: "domain", title: "Domain" },
134       {
135         dataIndex: "predication",
136         title: "Original Deposition",
137         render: (data: string) => { return <article className={` ${data === "Legitimate" ? "text-green-500" : "text-red-400"} `}>{data}</article> }
138       },
139       {
140         dataIndex: "virus-total-data",
141         title: "Detection Date",
142         render: (data: any) => { return <article>{new Date(data["detectionDate"]).toDateString()}</article> }
143       },
144       {
145         dataIndex: "action",
146         title: "Action",
147         render: function (_, data) {
148           // console.log(data);
149           return <div className="flex gap-4">
150             <Button onClick={() => { setActiveDetails(data) }}>Info</Button>
151             <Button onClick={() => { setActiveDetails(data); setShowModal(true) }}>Details</Button>
152           </div>
153         }
154       }
155     ]}
156     dataSource={dataSource}
157     tableLayout="fixed"
158   ></Table>
159 </div>
160 </main>
161 }
162 </>
163 }
164 }

```

Figure 158: dashboard.tsx (b).

7.13 Plagiarism Test Result

17/04/2023, 07:20

Originality report

COURSE NAME
CS6P05 - FYP - Shishir Subedi and Suraj Neupane

STUDENT NAME
SARTHAK BIKRAM RANA

FILE NAME
Final Report

REPORT CREATED
17 Apr 2023

Summary

Flagged passages	11	0.9%
Cited/quoted passages	7	0.8%

Web matches

coursehero.com	1	0.3%
visual-paradigm.com	4	0.3%
analyticsteps.com	1	0.2%
careerfoundry.com	2	0.1%
ijcaonline.org	2	0.1%
stakeholdermap.com	1	0.1%
fraud.com	1	0.1%
javatpoint.com	1	0.1%
medium.com	1	0.1%
unb.ca	1	0.1%
researchgate.net	1	0.1%
markconnects.com	1	0.1%
guru99.com	1	0.1%

1 of 18 passages
Student passage **QUOTED**

I confirm that I understand my coursework needs to be submitted online via Google Classroom under the relevant module page before the deadline in order for my assignment to be accepted and marked. I...

Top web match

Unformatted text preview: Enter your Full Name Here Enter your Full Name Here Enter your Full Name Here Enter your Full Name Here
Enter your Full Name Here Assignment Due Date: Assignment Submission...

Coursework Cover Page London Met For all Multimedia Modules 1 ... <https://www.coursehero.com/file/77813162/Coursework-Cover-Page-London-Met-For-all-Multimedia-Modules-1docx/>

2 of 18 passages
Student passage **FLAGGED**

phishing is another type of cyber-attack that involves creating and distributing fraudulent websites that appear to be legitimate in order to trick people into disclosing personal information such as...

about:blank Page 1 of 5

Figure 159: Originality Report (a).

17/04/2023, 07:20

[Top web match](#)

Phishing is a type of fraud that involves cybercriminals sending emails or other forms of communication pretending **to be legitimate organisations in order to trick individuals into revealing personal...**

Phishing - Tips, tricks, and strategies to protect your business and ... <https://www.fraud.com/post/phishing>

3 of 18 passages
Student passage **FLAGGED**

A use case diagram is used to describe a system's dynamic behavior. It contains the functionality of the system by incorporating use cases, actors, and their

[Top web match](#)

Uml use case diagram **A use case diagram is used to represent the dynamic behavior** of a system. It encapsulates **the system's functionality by incorporating use cases, actors, and their relationships.**

UML Use Case Diagram - Javatpoint <https://www.javatpoint.com/uml-use-case-diagram>

4 of 18 passages
Student passage **FLAGGED**

...is nothing more than a graphical depiction of steps. It displays steps in sequential order and is commonly used to depict the flow of algorithms, workflow, or processes

[Top web match](#)

It shows steps in sequential order and is widely used in presenting the flow of algorithms, workflow or processes. Typically, a flowchart shows the steps as boxes of various kinds, and their order by...

Flowchart Tutorial (with Symbols, Guide and Examples) <https://www.visual-paradigm.com/tutorials/flowchart-tutorial/>

5 of 18 passages
Student passage **CITED**

...to depict the flow of algorithms, workflow, or processes. A flowchart typically depicts the processes as various types of boxes and their sequence by connecting them with arrows

[Top web match](#)

It shows steps in sequential order and is widely used in presenting the flow of algorithms, workflow or processes. Typically, **a flowchart shows the steps as boxes of various kinds, and their order by...**

Flowchart Tutorial (with Symbols, Guide and Examples) <https://www.visual-paradigm.com/tutorials/flowchart-tutorial/>

6 of 18 passages
Student passage **FLAGGED**

...Where more than 10,000 URLs were taken from OpenPhish which is a repository of active sites and 35,300 benign URLs were gathered from top Alexa...

[Top web match](#)

Phishing URLs: Around **10,000 phishing URLs were taken from OpenPhish which is a repository of active phishing sites.** Malware URLs: More than 11,500 URLs related to malware websites were obtained from...

URL dataset (ISCX-URL2016) - University of New Brunswick <https://www.unb.ca/cic/datasets/url-2016.html>

7 of 18 passages
Student passage **FLAGGED**

...Heritrix web crawler to extract the URLs. Approximately 500,000 unique URLs are crawled initially and then passed through to remove duplicate and domain-only URLs

about:blank

Page 2 of 5

Figure 160: Originality Report (b).

17/04/2023, 07:20

[Top web match](#)

The domains have been passed through a Heritrix web crawler to extract the URLs. Around half a million **unique URLs are crawled initially and then passed to remove duplicate and domain-only URLs.**

URL Feature Engineering and Classification | Nerd For Tech - Medium <https://medium.com/nerd-for-tech/url-feature-engineering-and-classification-66c0512fb34d>

8 of 18 passages
Student passage FLAGGED

Mahajan, R. & Siddavatam, I., 2018. **Phishing website detection using machine learning algorithms.** International Journal of Computer Applications, 181(23), pp. 45-47.

[Top web match](#)

Phishing Website Detection using Machine Learning Algorithms Rishikesh Mahajan

Phishing Website Detection using Machine Learning Algorithms <https://www.ijcaonline.org/archives/volume181/number23/mahajan-2018-ijca-918026.pdf>

9 of 18 passages
Student passage CITED

...I., 2018. Phishing website detection using machine learning algorithms. **International Journal of Computer Applications**, 181(23), pp. 45-47.

[Top web match](#)

Rishikesh Mahajan and Irfan Siddavatam. Phishing Website Detection using Machine Learning Algorithms. **International Journal of Computer Applications** 181(23):45-47, October 2018.

Phishing Website Detection using Machine Learning Algorithms <https://www.ijcaonline.org/archives/volume181/number23/30087-2018918026>

10 of 18 passages
Student passage FLAGGED

Guo, H. & Yang, X., 2007. **A simple reliability block diagram method for safety integrity verification.** Reliability Engineering & System Safety, September, 92(9), pp. 1267-1273.

[Top web match](#)

A simple reliability block diagram method for safety integrity verification. September 2007; Reliability Engineering [?] System Safety 92(9):1267-1273.

A simple reliability block diagram method for safety integrity verification https://www.researchgate.net/publication/223620904_A_simple_reliability_block_diagram_method_for_safety_integrity_verification

11 of 18 passages
Student passage CITED

Bajracharya, N., 2021. **Information system audit in Nepal: Everything you should know**

[Top web match](#)

His prior positions were Secretary at ... <https://oag.gov.np/page/m-46/en> **Information system audit in Nepal: Everything you should know** ...

Class of auditor in nepal <https://tyjo.markconnects.com/post/class-of-auditor-in-nepal/83731853>

12 of 18 passages
Student passage CITED

about:blank

Page 3 of 5

Figure 161: Originality Report (c).

17/04/2023, 07:20

The Waterfall methodology, also known as the Waterfall model, is a sequential development process that falls like a waterfall through all phases of a project (for example, analysis, design,...

[Top web match](#)

The Waterfall technique, also known as the Waterfall model, is a sequential development process that falls like a waterfall through all phases of a project (for example, analysis, design, programming,...

Waterfall Methodology: Working, Advantages & Disadvantages <https://www.analyticssteps.com/blogs/waterfall-methodology-working-advantages-disadvantages>

13 of 18 passages
Student passage FLAGGED

The Prototype methodology is a software development model in which a prototype is developed, tested, and changed until it is acceptable. It also builds the foundation for the final system...

[Top web match](#)

Summary In Software Engineering, **Prototype methodology is a software development model in which a prototype is built, test and then reworked when needed until an acceptable prototype is achieved.**

Prototype Model in Software Engineering - Guru99 <https://www.guru99.com/software-engineering-prototyping-model.html>

14 of 18 passages
Student passage FLAGGED

Work Breakdown Structure A Work Breakdown Structure is a deliverable-oriented hierarchical decomposition of the work to be executed by the project team to accomplish the project objectives and...

[Top web match](#)

Work Breakdown Structure A deliverable oriented hierarchical decomposition of the work to be executed by the project team to accomplish the project objectives and create the required deliverables.

What does Work Breakdown Structure mean in Project Management? <https://www.stakeholdermap.com/project-dictionary/work-breakdown-structure-meaning.html>

15 of 18 passages
Student passage FLAGGED

...accomplish the project objectives and create the required deliverables. **The Work Breakdown Structure is developed to establish a common understanding of the project scope**

[Top web match](#)

The Work Breakdown Structure (WBS) is developed to establish a common understanding of project scope. It is a hierarchical description of the work that must be ...

What is Work Breakdown Structure? - Visual Paradigm <https://www.visual-paradigm.com/guide/project-management/what-is-work-breakdown-structure/>

16 of 18 passages
Student passage CITED

...to establish a common understanding of the project scope. **It is a hierarchical description of the work that must be done to complete the deliverables of a project.** (Visual Paradigm, 2020)

[Top web match](#)

It is a hierarchical description of the work that must be done to complete the deliverables of a project.

What is Work Breakdown Structure? - Visual Paradigm <https://www.visual-paradigm.com/guide/project-management/what-is-work-breakdown-structure/>

about:blank Page 4 of 5

Figure 162: Originality Report (d).

17/04/2023, 07:20

17 of 18 passages

Student passage FLAGGED

Wireframing is a UX design technique that allows UX designers to define and arrange the information hierarchy of their design for a website, app, or product

Top web match

What is a wireframe? **Wireframing is a practice used by UX designers which allows them to define and plan the information hierarchy of their design for a website, app, or product.**

The Definitive Guide: How To Make Your First Wireframe <https://careerfoundry.com/en/blog/ux-design/how-to-create-your-first-wireframe/>

18 of 18 passages

Student passage CITED

...app, or product. The concept of making a wireframe **focuses on how the designer or client wants the user to take in information on a website.** (Allabarton, Rosie, 2022)

Top web match

This process **focuses on how the designer or client wants the user to process information on a site,** based on the user research already performed by the UX design team.

The Definitive Guide: How To Make Your First Wireframe <https://careerfoundry.com/en/blog/ux-design/how-to-create-your-first-wireframe/>

about:blank

Page 5 of 5

Figure 163: Originality Report (e).