Dissertation on

Web Page Classification for Safer Browsing

Submitted in partial fulfilment of the requirements for the award of degree of

Bachelor of Technology
in
Computer Science & Engineering

UE18CS390A – Capstone Project Phase - 1

Submitted by:

Manav Agarwal	**PES2201800025**
Rishab Kashyap	**PES2201800065**
Shreya Yuvraj Panale	**PES2201800117**
Shreya Venugopal	**PES2201800688**

Under the guidance of

Prof. U Ananthnagu
Professor of Department of Computer
Science and Engineering
PES University

January - May 2021

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
FACULTY OF ENGINEERING
PES UNIVERSITY
(Established under Karnataka Act No. 16 of 2013)
Electronic City, Hosur Road, Bengaluru – 560 100, Karnataka, India

PES UNIVERSITY

(Established under Karnataka Act No. 16 of 2013)
Electronic City, Hosur Road, Bengaluru – 560 100, Karnataka, India

FACULTY OF ENGINEERING

CERTIFICATE

This is to certify that the dissertation entitled

Web Page Classification for Safer Browsing

is a bonafide work carried out by

Manav Agarwal	**PES2201800025**
Rishab Kashyap	**PES2201800065**
Shreya Yuvraj Panale	**PES2201800117**
Shreya Venugopal	**PES2201800688**

In partial fulfilment for the completion of sixth semester Capstone Project Phase - 1 (UE18CS390A) in the Program of Study -Bachelor of Technology in Computer Science and Engineering under rules and regulations of PES University, Bengaluru during the period Jan. 2021 – May. 2021. It is certified that all corrections / suggestions indicated for internal assessment have been incorporated in the report. The dissertation has been approved as it satisfies the 6th semester academic requirements in respect of project work.

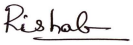
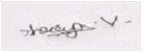
Signature Prof. U Ananthnagu Professor	Signature Dr. Sandesh B J Chairperson	Signature Dr. B K Keshavan Dean of Faculty

External Viva

Name of the Examiners	**Signature with Date**
1. _____	_____
2. _____	_____

DECLARATION

We hereby declare that the Capstone Project Phase - 1 entitled **"Web Page Classification for Safer Browsing"** has been carried out by us under the guidance of Prof. U Ananthnagu Professor of Department of Computer Science and Engineering PES University and submitted in partial fulfilment of the course requirements for the award of degree of **Bachelor of Technology** in **Computer Science and Engineering** of **PES University, Bengaluru** during the academic semester January – May 2021. The matter embodied in this report has not been submitted to any other university or institution for the award of any degree.

PES2201800025	**Manav Agarwal**	
PES2201800065	**Rishab Kashyap**	
PES2201800117	**Shreya Yuvraj Panale**	
PES2201800688	**Shreya Venugopal**	

ACKNOWLEDGEMENT

I would like to express my gratitude to Prof. U Ananthnagu, Department of Computer Science and Engineering, PES University, for her continuous guidance, assistance, and encouragement throughout the development of this UE18CS390A - Capstone Project Phase – 1.

I am grateful to the Capstone Project Coordinator, Dr.Sarasvathi V, Associate Professor, for organizing, managing, and helping with the entire process.

I take this opportunity to thank Dr. Sandesh B J, Chairperson, Professor, Department of Computer Science and Engineering, PES University, for all the knowledge and support I have received from the department. I would like to thank Dr. B.K. Keshavan, Dean of Faculty, PES University for his help.

I am deeply grateful to Dr. M. R. Doreswamy, Chancellor, PES University, Prof. Jawahar Doreswamy, Pro Chancellor – PES University, Dr. Suryaprasad J, Vice-Chancellor, PES University for providing to me various opportunities and enlightenment every step of the way. Finally, this Capstone Project could not have been completed without the continual support and encouragement I have received from my family and friends.

ABSTRACT

Browsing is one among the most common activities performed by people on a daily basis, ranging from simple facts to complex information retrieval and work. The internet is a vast chasm of infinite information from all over the world, providing resources from pictures to articles to computing resources. The advantage it provides proves to be a major disadvantage too, as malicious users tend to misuse this power to gain information from innocent users through various malicious methods, and hence our project aims to eliminate such URLs and links from the web searches of users and keep them safe from such threats. Through this literature, we aim to design a model that classifies such malicious URLs and block them before they can cause any harm using various methods of prevention.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER-1

INTRODUCTION

The internet today proves to be a most formidable place to retrieve or put up information and hence becomes a breeding ground for dangerous websites and malicious users who misuse the internet. The project focuses on the study of malicious URLs and their respective websites, and thereby fabricates a model that allows us to detect dangerous websites and URLs, and thus eliminate future recurrences of such incidents from occurring in our systems. Multiple approaches are used to establish the following method with the use of various concepts related to the browser search engines, machine learning, pattern analysis and so forth.

CHAPTER-2

PROBLEM DEFINITION

A large quantity of the scams and cyber-attack in the present day are caused because of the existence of malicious web pages. Malicious URLs reach naive users via text messages, email, advertisements or pop-up pages. Visiting such URLs can result in something as devastating as the user's email account being hacked, download of malware, spyware and ransomware, launching of phishing campaigns that may result in severe monetary losses. It may also lead to Denial of Service attacks.

Current solution:

The standard and fastest way to identify malicious URLs is by comparing URLs against blacklists. However, blacklists are never exhaustive and lack the ability to detect newly generated URLs

Our solution :

Identification of URLs posing a threat to browsing and web searches using a combination of various website detection schemes.

CHAPTER-3

LITERATURE SURVEY

3.1 Introduction

This chapter provides an insight into the literature that has dealt with similar problem definitions. It stands as a foundation for laying out the design and executing the project smoothly. In subsequent sections and subsections the various approaches used to solve the problem will be highlighted. Along with this the advantages and disadvantages of the approaches are also given.

The inferences will also include a brief insight into the machine learning models, algorithms and other methods used.

3.2 Approaches

Billions of links exist on the internet today and comprise a variety of topics and content within them that is browsed or added into by various users from around the world. All the data is present on the various pages created by users, each one accessible through a URL. The enormity of the internet also allows people to add in malicious pages and content and create fake URLs for the same purpose, and these are known as malicious or phishing URLs. The challenge is to look for the needle in the haystack and upon finding these URLs, block them and prevent similar types from spreading havoc on the internet. Numerous methods exist for the same and a variety of techniques to perform each of them are prevalent or in development. Machine Learning plays a vital role in the development of a majority of these methods.

The use of Page Ranking in Search Engine based methods prove to be among the safest methods to deploy while searching for such phishing links through the usage of page ranking graphs to establish how trustable a page really can be. Methods such as introduce us to the variety of phishing links created from AI as well as users, and provide solutions based on studying the patterns in the links. Gives us an insight into the locations of these URLs and safer methods to block them out of the servers that provide information to requesting users. Visual based similarity looks into the various uses of symbols, content as well as patterns within the URL to confirm the true nature of the page being accessed.

As mentioned before, Machine Learning is prevalent in almost all the above methods and plays a vital role in the classification of phishing URLs. Variety of machine learning models give us different results based on the method we deploy to extract the features as well as the source of the data we use to classify the same. We hence proceed to look into the various methods in detail in the following subsections.

3.2.1 Search Engine Based

This approach is based on the assumption that the reputation of a URL plays a major role in determining whether it is malicious or not. The advantage of this kind of solution is that it is easy to implement and does not require the content of the web page. Besides, it can be used for real-time analysis. In some cases, usually when combined with proactive phishing it is just used for feature extraction. However some web graph based approaches may be computationally complex.

This reputation is given by the rank given to the URL by a search engine such as Google, Bing, Yahoo etc or derived from these ranks.

It can also be obtained based on the search results obtained on querying the link or certain attributes of the link such as domain name on the search engine. In some cases, the reputation is obtained based on the connectivity of the URLs if their linking was represented in the form of a graph.

Paper Details	**Objective of paper, Techniques/Methods**	**Advantages**	**Limitations**
Web Phishing Detection Using a Deep Learning Framework (Yi et al. 2018)	Examine the IP flows from ISP and find a detecting technique based on two kinds of attributes- Original and	Does not require the content of the web page instead it requires only the properties and visiting behavior of	Mathematically complex

	Interactive	user and web page. Combines proactive phishing approaches also for better results.	

Table 3.1 Search Engine Based Literature Survey Paper Summary

3.2.2 URL-Based

Proactive Phishing is a method in which we visually inspect the URL itself for any suspicious traits. This is inclusive of the various methods used to parse through the URL string to identify certain patterns and symbols associated with the same. This can be done by the use of lexical analysis to scan the URL. The following methods as discussed are presented in various ways in the papers that follow.

Paper Details	Objective of paper, Techniques/Methods	Advantages	Limitations
A Machine Learning Approach for Detecting Malicious Websites using URL Features (Manjeri et al. 2019)	Compares multiple algorithms against combination of different attributes	Feature Selection Methods done were successful since the harmful URLs were correctly detected using only a very few features.	Class imbalance reduces the accuracy of these algorithms, hence removing the class imbalance will be a problem to look into

Table 3.2 URL Based Literature Survey Paper Summary

3.2.3 Visual Similarity Based

Visual Similarity is the process in which a URL's contents are scrutinized in order to classify it as a legitimate or a dangerous one. This can be done in plenty of ways such as checking the URL itself, looking into the basic site details such as the title and the images such as the logo, we could also look into the html content that has been added into the pages, and many more techniques. A variety of these techniques involve a certain level of risk taken by the users to analyze the page with respect to the content it delivers by directly accessing the link for retrieving this information.

Paper Details	Objective of paper, Techniques/Methods	Advantages	Limitations
Phish Haven—An Efficient Real-Time AI Phishing URLs Detection System(Sameen, Han, and Hwang 2020)	Make use of URL HTML and URL Hit along with parallelly computed ensemble methods for detecting various attacks. Summarizes the various processes to perform proactive phishing along with visual based similarity using multiple techniques.	Efficient and fast detection of URLs based on the types of symbols Can detect both human generated as well as AI generated malicious URLs Faster computation with Machine Learning models using parallel threads. Detection of malicious tiny URLs	Not the most efficient when it comes to analysis of zero-day attacks Real time third party dependencies

Table 3.3 Visual Similarity Based Literature Survey Paper Summary

3.2.4 DNS Based

Domain Name Servers are responsible for providing web content to all the requesting users at the local level. They are responsible for searching the correct pages as requested by the user across the entire internet by contacting various servers. This approach thus looks into the possibility and analysis of all the URLs stored at these servers and classifying them as legitimate or malicious.

Paper Details	Objective of paper, Techniques/Methods	Advantages	Limitations
Malware Detection using DNS Records and Domain Name features (Messabi et al. 2018)	A mix of Proactive Phishing Based and **DNS** to see which are the most corrupt TLDs based on previous datasets 10 fold Cross-Validation used	The feature selection is done in a good way since priorities have been assigned to the features based on effect they have on the accuracy	Low Accuracy. Attackers might change methods over time and new features need to be recognized.

Table 3.4 DNS Based Literature Survey Paper Summary

3.2.5 Heuristic Based

This approach involves obtaining the result based on certain rules that are inferred from elsewhere or whose application tends to give correct results. Such rules are called heuristics and this method usually involves the use of features or rules given in the previous method.

Paper	Objective of	Advantages	Limitations

Details	**paper, Techniques/Methods**		
A heuristic technique to detect phishing websites using TWSVM classifier (Rao, Pais, and Anand 2020)	A mix of attributes including basic URL-based ones, hyperlink-based and similarity-based attributes Obtains the level of dissimilarity between the login page of phishing site and homepage of compromised domain.	Can detect zero-day phishing attacks Recognizes phishing sites hosted on attacked web servers Does not get affected by search engine, page rank, and database Parallelized feature extraction and its classification to reduce time complexity	False positive rate of 2.23% because the homepage of the visited URL is not there Because of a high level of similarity between the free hosting domain and web site some legitimate web pages are classified wrongly Fails to detect low-content web-pages

Table 3.5 Heuristic Based Literature Survey Paper Summary

3.3 Inferences

It can be inferred that methods of combating this problem vary greatly. The DNS method will give the IP address which is easily traceable but it has low accuracy. Search Engine based methods are independent of the content of the webpage but are complex if one is trying to build their own ranking system. On the other hand, the heuristic based approach is accurate but feature extraction is costly. URL based approach is accurate and comparatively simple but it lacks in firm logic since it does not give any conclusive evidence on the cause of maliciousness. Visual based approaches are easy to comprehend as a solution but tedious in terms of feature extraction.

3.4 Conclusion

Thus as we have seen, each of the papers share the common trait of blocking out phishing or malicious URLs. They vary in their methodologies, each of which prove to be successful in their own way and have shortcomings of their own. Some methods have a lower accuracy while the others have a higher accuracy but only for certain types of URLs this can be tried and improved on to get better results by combining various features from these models.

In conclusion, we aspire to create a working application by using multiple features from some of the selected literature to improve the efficiency and build a model to perform malicious URL detection.

CHAPTER-4
SYSTEM REQUIREMENTS SPECIFICATION

4.1 Introduction

This chapter consists of the requirements and basic explanation of a system that will be used to detect malicious URLs. This will include a brief description of each requirement and its categorization based on its importance.

4.1.1 Project Scope

The project will not only identify malicious URLs but also be able to derive some behavioral similarities in these URLs. Improvement over the detection schemes existing currently. A possible shortcoming, may be dealing with a static set of web URLs where it is difficult to see if a website is still safe or malicious in real time. Also the product may be computationally expensive since it needs more resources to access the entire network of URLs on analyzing in real time.

4.2 Product Features

The set of functions performed by the product along with its sequence are given in the figure below:

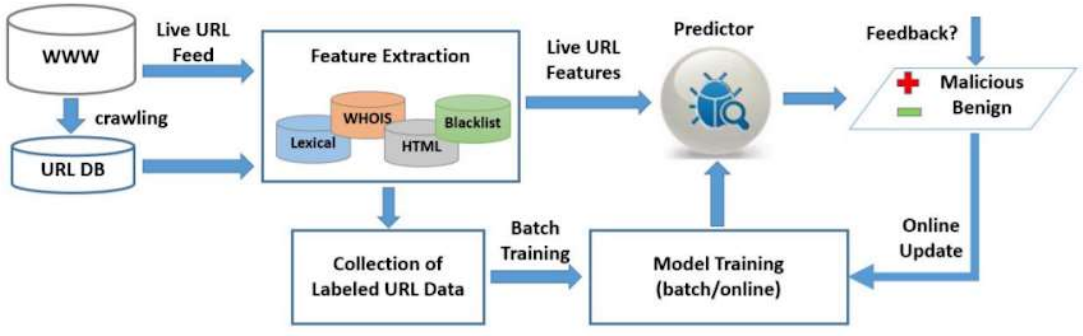


Fig. 4.1 Basic product features

- **Dataset Creation:**
  - Involves the URLs and the datasets from various sources
  - A database is created by combining all sources
  - A source may include URLs obtained by web-crawling also
  - From the dataset a sample is selected such that it has almost equal proportion of both training and testing data
  - Sample should also be diverse in terms of the domains it comprises of
- **Feature Extraction:**
  - All possible features are mined. These features maybe:
    - Search Engine Based: PageRank, Number of search results on querying key words, Rank of URL found when its key words are queried etc.
    - URL based: Lexical analysis counting the frequency of special characters, number of domain names, extracting keywords, http/https, presence of IP address, length etc.
    - DNS based: IP address, TLD, file system, information given by the Google API etc
    - HTML contents: For tags such as <p>, <h1>,<description>, etc
    - CSS properties: For page layout properties like height, width, padding, etc
    - Other general properties such as whether the home page for the web page exists and whether it leads to a login page that leads back to the same homepage
  - Some of the above features such as the search engine based, HTML and CSS related ones may not be available since the URL may be blocked by the browser
  - Hence these mined features must be consolidated and valid preprocessing must be done
  - Using a machine intelligence algorithm such as Decision trees with or without a wrapper or some form of attribute selection, statistical significance etc.
- Dataset Splitting into testing and training
- **Training Module:**

- Will apply multiple models and check which gives best accuracy
- Models may be:
  - Classification models such as SVM
  - Random Forests
  - Neural Networks and its variants
  - Genetic Algorithms such as Harmony Search
- These may be coupled with similarity metrics like the Jaccard Similarity
- They may include bagging and boosting techniques for better accuracy
- Combinations of multi-level or various combination of models may be experimented
- All these will be evaluated based on some performance metrics and the one which is evaluated as best will be finalized

- **Predictor:**
  - Passing the testing data into the training model
  - Classifying live URLs
  - Propagate the classification to the URLs that are closely associated with the predicted URLs thereby improving the performance

User Classes and Characteristics

The system can be used mainly by two kinds of users about which the details are given below:

	Naive Web User	**Network Security Engineer**
Definition	Person who is using the internet to access a web page and needs to be protected from malicious URLs	Person trying to research in this field or using this tool for commercial purposes or trying to enhance it. Also includes organizations that keep BlackList records

Frequency of Use	Each time a web-page is encountered tool is used to classify	Depends on the motive behind use. Example, a researcher will only look into the working once, a maintainer as and when required for enhancements etc.
Technical Capability	No technical knowledge	Specialized technical knowledge in computer science
Security Levels	Should only have the UI visible	Back-end must not be entirely visible as it can fall in the wrong hands but information regarding structure of product is known

Table 4.1 User Classes and Characteristics

Operating Environment

- Hardware Platform
  - Any system that supports browsing
  - System that allows the usage of a keyboard for data or URL entry
  - Monitor or device must allow all forms of notifications to be seen so as to receive warnings from the model
- Software Platform
  - Must allow browsing platform of HTTP/1.1
  - Should be allowed to parallely allow the model to take in input data
  - Permissions granted to access suspicious html pages if necessary
  - Respond to requests sent by the model to block a URL
- Operating System
  - Any Operating System that supports browsing

General Constraints, Assumptions and Dependencies

- **Access to some attributes are limited**

Phishing URLs that will be used for building the model mainly may not be available for search queries as it has been blocked by the browser. Hence the following features will not be accessible:

- Contents of the web-page
- Page-rank and web-graph connectivity related features: This is mainly because may of the blocked URLs do not appear on giving a search query
- DOM characteristics
- CSS characteristics

- **Cost is specific to client system for some attributes**

  For attributes such as those given below the time taken to obtain information regarding it depends on the speed of the machine and the internet connection. These may cause unwanted delays in some cases.

  - DNS related attributes
  - Search Query Results

- **Feature Extraction methods are time consuming**

  Some processes for feature extraction are tedious and since feature selection methods will be applied all the features extracted may not even be used. Some such processes are:

  - Lexical Analysis: Every character of the element must be traversed and sometimes re-traversed to obtain significant information.
  - CSS characteristics: Not only do the properties need to be extracted, those that are useful for the product must also be selected
  - Web-crawling to look into search engine related features: Each link has several paths that leads from it and each of them must be traversed
  - HTML tag data extraction: Traversing the DOM tree of a webpage is very tedious and time consuming

- **Effectiveness depends on the best kind of sampling done on the dataset to ensure no bias**
  - Proper proportion of benign and malicious URLs must be taken
  - Data must not be related to web pages of only a certain type. There should be diversity

    - Must not result in overfitting
- **It is assumed that:**
    - URLs given in the BlackList or WhiteList have been classified correctly
    - Malicious URLs have some sort of similarity i.e. follow a certain set of rules unintentionally
    - Reputed Pages are not malicious

Risks

- Too much time taken for classification leading to a crash
- If the dataset is not sampled properly the results may be inaccurate or biased
- May have more time and space complexity than existing methods
- Complete feature extraction may lead to a sparse dataset leading to inaccuracies
- Heuristics or rules generated may not be consistent with the current malicious URLs
- The product may be hacked or face some form of cyber security threat such as Denial of Service attacks

4.3 Use Case Diagram

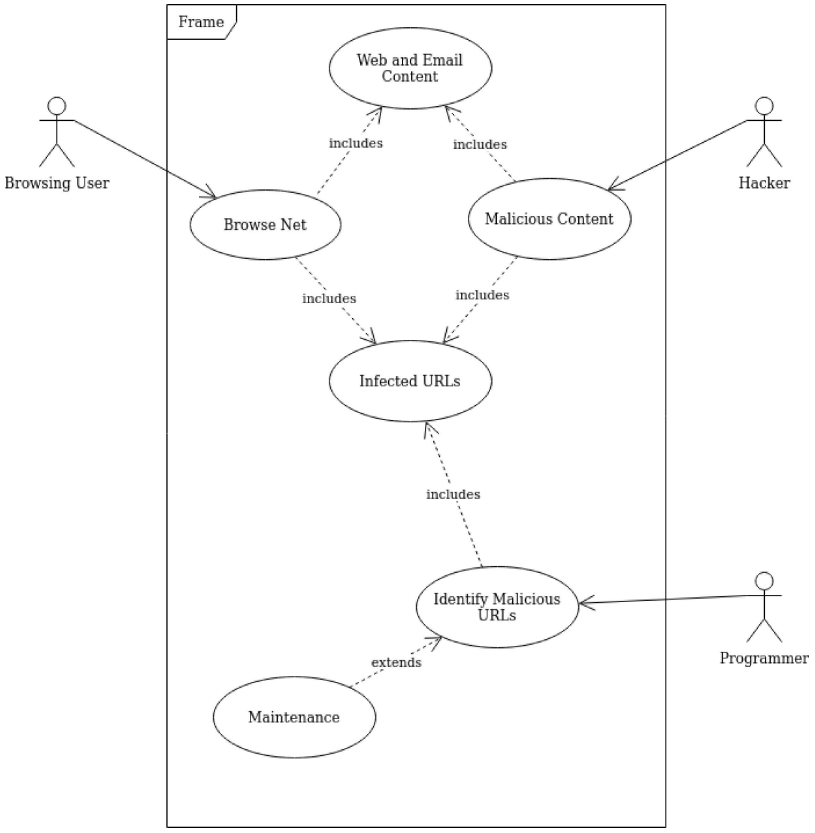


Fig. 4.2 Use-Case Diagram

4.4 Functional Requirements

Req1: Each Input must be a URL

Req2: There should be almost equal proportions of benign and malicious URLs

Req3: The occurrence of special characters must be extracted from the URLs

Req4: For all inputs the proportion of URLs that are not blocked by the browser must be calculated

Req5: For all inputs the proportion of those that give results on querying in a search engine must be calculated

Req6: If the proportion of inputs that satisfy Rule5 are greater than 50% Page Rank related attributes must be taken into account

Req7: Keywords from the URL must be extracted

Req8: Additional domains if present in the URL must be extracted

Req9: URL must be checked for HTTP or HTTPS

Req10: Number of outlinks from the URL must be extracted

Req11: Outlinks of the URL must be extracted

Req12: PageRank of the URL must be found

Req13: Length of the URL should be extracted

Req14: Preprocessing of data must be such that null characters are handled if all attributes for a URL cannot be extracted

Req15: At least 3 URL feature selection methods must be applied

Req17: At Least 3 training and testing split methods should be applied

Req19: At Least 5 URL classification models must be applied

Req20: At least 10 possible model combination performance must be explored

Req22: A genuine accuracy level must be achieved before approving any model combination

Req23: Each Phase must have multiple unit tests (Stub-Drive approach) before implementation

Req24: Must satisfy the Hardware and Software requirements before deploying the model as given in the below section

4.5 External Interface Requirements

User Interfaces

- Requires a browsing tab or a textbox to enter the URL to be analyzed
- Option for description of the working of the model
- Messages for malicious URL will be displayed to the user before it can be accessed.

Hardware Requirements

- Basic system supporting a simple browsing system
- Any screen or device that is compatible to display searches and handle the model
- Model Requirements:
- Any working computer system with any OS

- Keyboard to enter the URL required
- Mouse to operate and select the required entities

Software Requirements

- Python 3.6 and above
- Jupyter Notebooks (Colab)
- Access to network
- Access to any browser such as Google or Firefox
- Data Storage Services for URL analysis

Communication Interfaces

- Must have a decent internet speed to allow enough power for the model to identify the type of URL
- Browser must allow access rights to examine the user's entered URL
- Permission to access html in web pages for analysis.

4.6 Non-Functional Requirements

Performance Requirement

- Req16: Highest Accuracy URL feature selection method for the dataset must be applied
- Req18: Highest Accuracy train-test split method for the dataset must be applied
- Req21: For a new URL the output must be given in 0.2s
- Req25: Should function smoothly without any hitches
- Req26: Must display only the model combinations that have the highest accuracy.

Safety Requirements

- Req27: The model should not disrupt the browsing mechanism
- Req28: Users must not be blocked from the browsing system if the model faces any form of failures

- Req35: Maintainers should check for the health and performance of the system constantly on deployment
- Req36: Changes made to the system must be made offline and new modifications must be tested before allowing users to make use of it.

Security Requirements

- Req29: Only maintainers or developers of this project must have access to the underlying code.
- Req32: Users must have the proper system requirements to access the model to allow web detection to prevent any issues inflicted on the application in this process.
- Req33: The system must be prepared to defend itself against any corrupt files holding viruses.
- Req34: Any issues to the system must immediately alert the maintenance crew to check for bug fixes and eradicate issues.

4.7 Other Requirements

Req30: Product should not have more than 0.5% difference in accuracy from the existing accuracy

Req31: Product accuracy should not have more than 0.5% for different browsers

Appendix A: Definitions, Acronyms and Abbreviations

URL: Uniform Resource Locator

HTTP: HyperText Transfer Protocol

HTTPS: Secure HTTP

BlackList: Set of Malicious URLs

WhiteList: Set of Non-Malicious URLs

CHAPTER-5

HIGH LEVEL DESIGN

5.1 Introduction

This chapter contains the architectural and high level design details of a method to detect malicious URLs. It describes that the problem statement will be solved by applying machine learning and web-graph analysis approaches supported by feature selection and mining of features from a URL.

5.2 Current System

In the current system that is commercially prevalent, each browser keeps two lists one containing the identified malicious URLs and one containing the reputed URLs. The list containing the harmful URLs is called a Blacklist and any URL that is part of this list is blocked by the browser. The other list is called the Whitelist.

The main drawback in this is that new URLs are not a part of the Blacklist and usually the phishing URLs are new and short living. Hence a very small proportion of malicious URLs are actually captured in the Blacklist.

5.3 Design Considerations

5.3.1 Design Goals

- The current system of checking blacklists is a slightly slower process, and it's efficiency reaches a limit when it is unable to find a URL in the list even if it is malicious.
- Furthermore, as mentioned above, the URLs have a short lifespan and won't exist for too long and hence the prediction must be done quickly to block it before any damage is done

- Our goal is to minimize the dependency of such lists and create a more flexible and robust system to be placed in the environment for smoother and better functioning.

5.3.2 Architecture Choices

- A total of six basic methods were introduced as an outcome of the literature survey and consisted of a dozen methods to implement them all
- The use of Machine learning algorithms and systems prevailed in a majority of them, only being surpassed by the concept of lexical analysis used in almost all of the sources
- Some popular methods include the analysis of the URLs themselves to find unidentified or suspicious symbols within them. Another approach involved the study of each webpage resulting from the link to identify fraudulent data

We aim to exploit the URL in different ways to achieve better results:

- Lexical Analysis of all URLs
- Reference URLs – backlinks
- Rank of link in the results
- Page Rank
- Popularity based on searched keywords
- Domain name in URL
- Presence of IP address
- SSL final state
- Number of words in the URL (cardinality)

- Pros:
    - Cleaner method of studying each URL, thus making a thorough checkup before the actual classification
    - Usage of high efficiency models such as neural networks and page ranking algorithms for better performance
    - Levels of testing and clarification for accurate classification
- Cons:

- Costly process timewise due to a number of methods being used
- Memory utilization is higher due to the use of large modules for the system

5.3.3 Constraints, Assumptions and Dependencies

Include the limitations constraints that have a significant impact on the design of the system. Such constraints may be imposed by any of the following

- Interoperability requirements
  - System must be connected to a browsing system or must be connected to a source that gives URls as inputs.
  - Might have issues with the interface of the browsing system; Needs to be compliant with all of them and run smoothly or any of them.
- Interface/protocol requirements
  - Requires a basic computer system with any browser installed over it or a data source that allows it to retrieve URLs to classify then successfully.
  - Might lead to memory issues on smaller systems, thereby affecting the overall performance
- Data repository and distribution requirements
  - Must have a cluster of URLs fed into it as a testing set
  - Needs to be upgraded in order to allow single URLs at a time.
- Discuss the performance related issues as relevant.
  - Efficient in terms of classification and trustworthy in terms of safety
  - Needs to be configured to suit the system and reduce overall memory utilization to improve performance
- End-user environment.
  - Must have a browsing system and a system that supports enough RAM to accommodate the system.

    - Can also have an alternate data source to feed the system data for classification of URLs

- Availability of Resources.
    - We assume that the URLs and other inputs are available to the system whenever required
    - Initially in the form of batches or clusters, and later on in a dynamic format for real time classification
- Hardware or software environment
    - Any computer system with a good amount of RAM and memory capacity to store URLs and allow the system to freely classify the test data.
    - A browser system to generate these URLs while searching online
- Discuss issues related to deployment in target environment, maintainability, scalability, availability, etc.
    - Might have issues with compatibility of the system with the browser
- Any other requirements described in the Requirements Document.
    - None

5.4 High Level System Design

The system will consist of a URL given as input. Using this URL features will be extracted that are related to it. Then the features will be categorized and graph and non-graph based models and algorithms will be applied on them respectively. A mechanism of deriving a final conclusion from the results of these two approaches will give the output. The following diagram gives the basic logical data flow

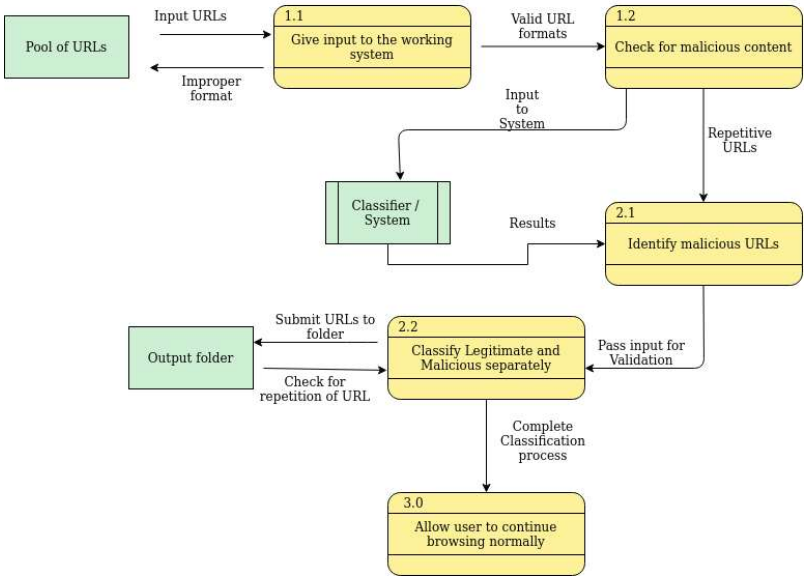


Fig 5.1 High Level System Design

5.4.1 Logical User Groups

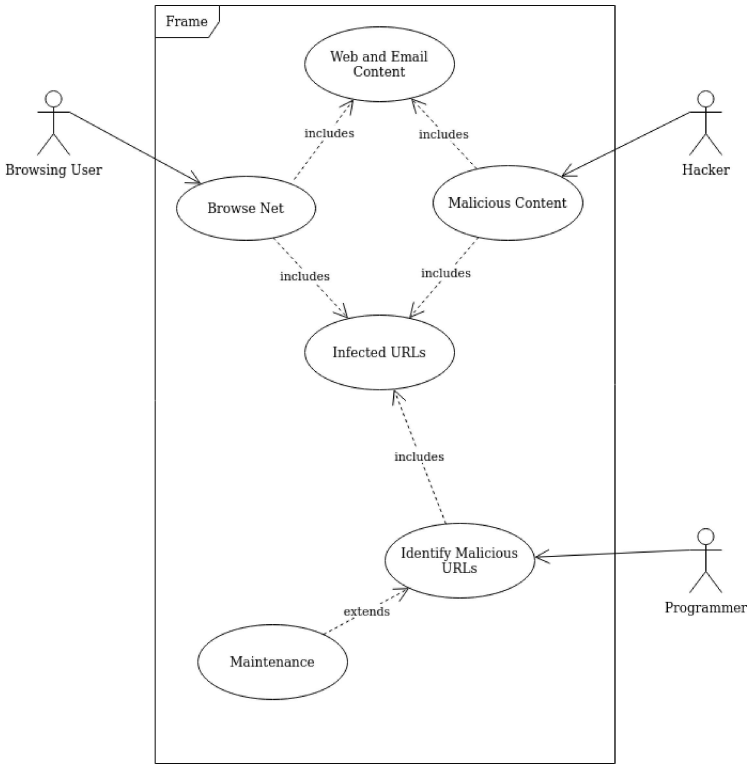


Fig 5.2 Logical User Groups

According to the Use Case Diagram given above the user groups that are associated with this product are:

- **Browsing User-** A naive user who is trying to view the contents in a URL without knowing if it is malicious or not. The system is made for protecting this user.
- **Hacker-** Person who is creating malicious URLs to create malicious content and harm the Browsing User.
- **Programmer-** Person creating and maintaining the malicious URL detection system.

5.4.2 Run Time View of the System

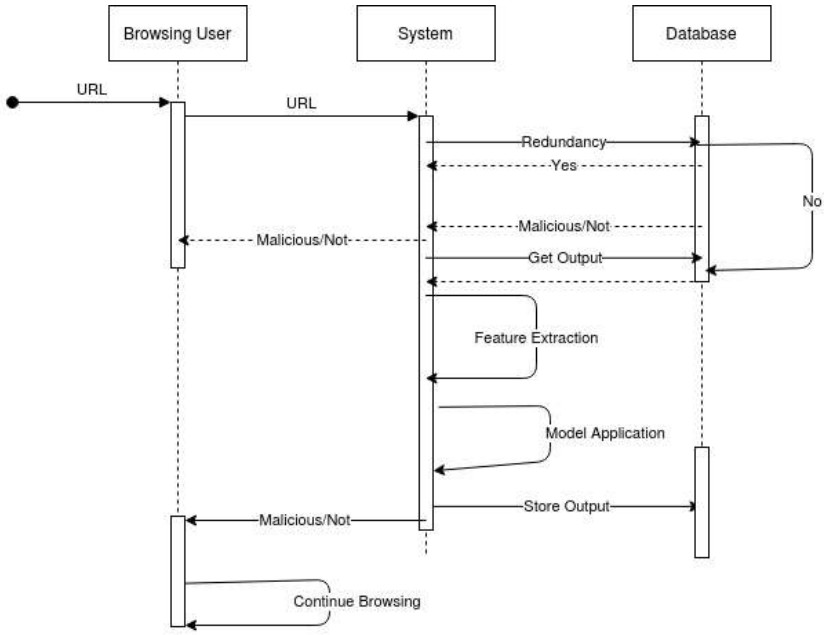


Fig 5.3 Run Time View of the System

The above diagram represents the system when in action after the model has been trained and its parameters have been finalized.

5.4.3 Project Management and Code Organization

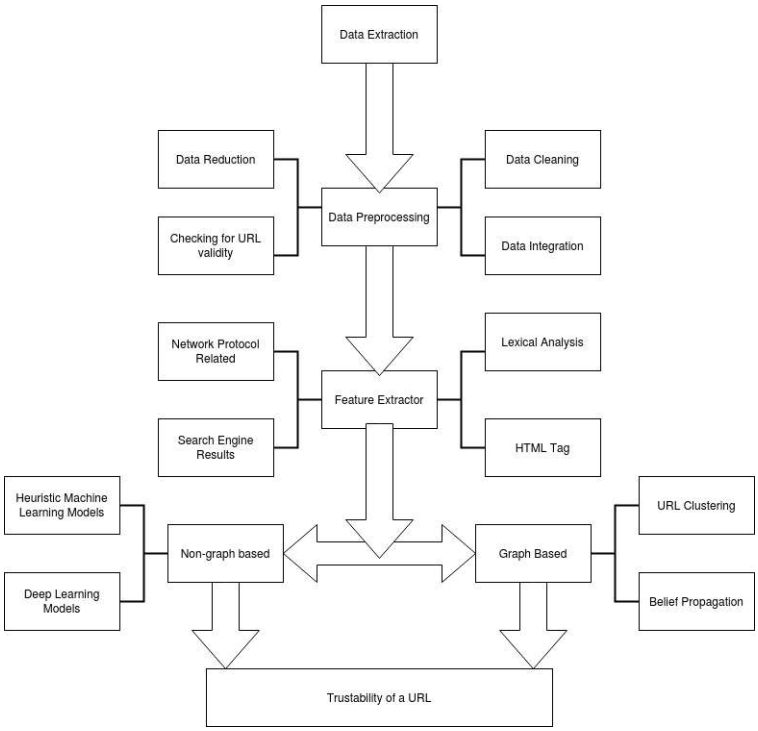


Fig 5.4 Project Management and Code Organization

5.4.4 Security

An important component includes keeping the data used for the classification secure so that it cannot be manipulated leading to wrong results. The criticality of ensuring this for this system is very high because if the data is manipulated and a malicious URL is classified as benign and set free or vice-versa the consequences can be terrible. Hence the following security measures or features can be applied:

- Restriction to transaction traffic
- Following the Principle of Least Privileges in terms of providing access permissions
- Auditing logins

- Have a constant visibility into its configuration state so that even the slightest anomaly can be noticed.

5.5 Design Description

5.5.1 Master Class Diagram

As we can see in the diagram above we have three different classes namely URL, Storage and Classification which each serve a basic functionality as stated below

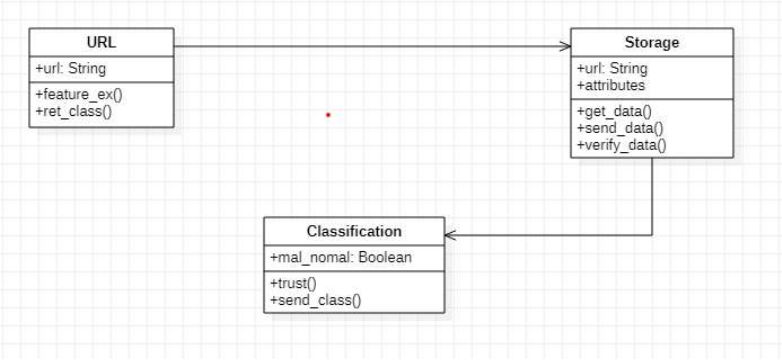


Fig 5.5 Master Class Diagram

- **URL:** The class URL will be responsible to take the user-input URL and analyze it to return if the URL has been classified as malicious or not.
  - **url:** This is an attribute of type String which stores the URL to be checked for classifying.
  - **feature_ex():** Is a function that is used to extract features from the given URL that will serve as inputs to the models that classify them as malicious or non-malicious.
  - **ret_class():** This function is used to return the result obtained (malicious or not) to the calling method.
- **Storage:** The storage class will be used to keep a track of all the URLs that have been checked and used to build a model based on which we perform our classification. It will also store the results of classification of various new websites as and when entered by the user to reduce the response time in case a URL is checked more than once.

- **attributes:** An array that contains a set of all the attributes that have been extracted from the URL to build the model/classifier.
- **get_data():** A function that gets the input from the URL class to check.
- **send_data():** A function that sends the result of classification of the model to the URL class.
- **verify_data():** A function that sends the attributes and the URL to the model built to classify it as one of the two classes.

➢ **Classification:** This class focuses on creation and updation of the model based on the training dataset and the newly received inputs from the user. It tests the entered URL with a pre-existing model and classifies it as malicious or non-malicious and returns the result to be stored into the database for the particular URL for further use.

- **mal_nomal:** Is a variable that stores boolean value for a given URL specifying if it's safe for browsing or not.
- **trust():** This function uses the built model to read the attributes required and classifies the URL as safe or not safe.
- **send_class():** This function sends the URL and the class it belongs to back to the storage/database to be stored and returned to the user.

5.5.2 Reusability Considerations

- **BeautifulSoup:** A Python library used to parse through HTML tags and get the values of attributes in them
- **Python SEO Analyzer:** For search engine related analysis and optimizations
- **Deep Learning Frameworks** such as TensorFlow

5.6 Diagrams

5.6.1 ER Diagram

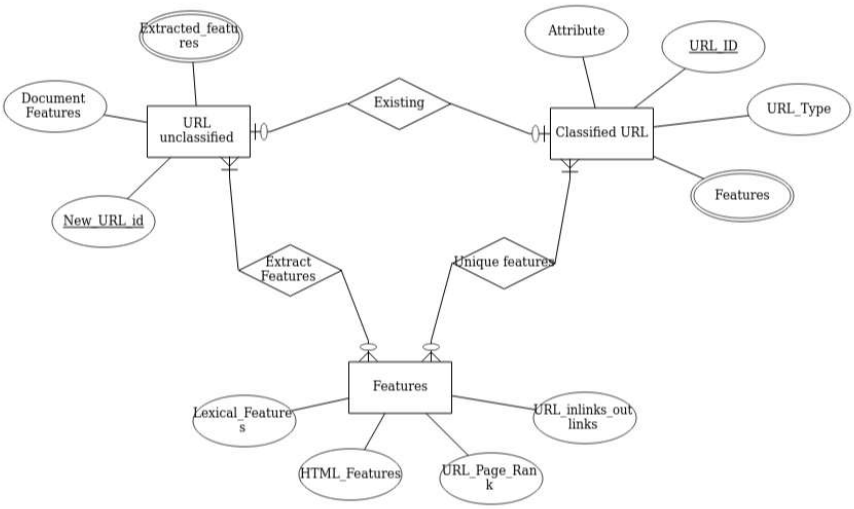


Fig 5.6 ER Diagram

The ER diagram shown above corresponds to the three main classes of data that will be stored here. The two main ones are the classified and unclassified URLs. The features provide an external reference to the types of features available from the URLs

5.6.2 Activity Diagram

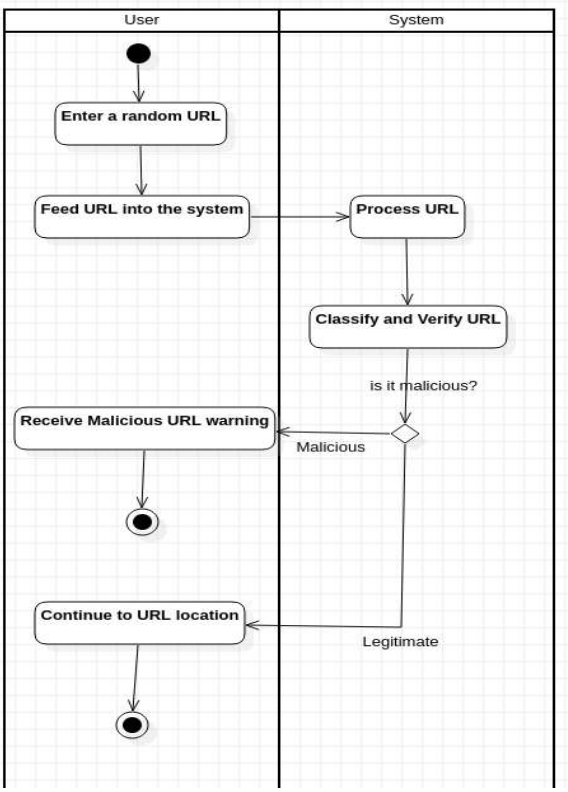


Fig 5.7 Activity Diagram

The above Activity diagram displays the step by step procedure of the entire process from the start to the end. It includes the two main classes participating in the process and the activity associated with each step of the process throughout.

The above diagram thus shows the relationship between the two classes and the flow of actions necessary for the system to work smoothly.

5.6.3 State Diagram

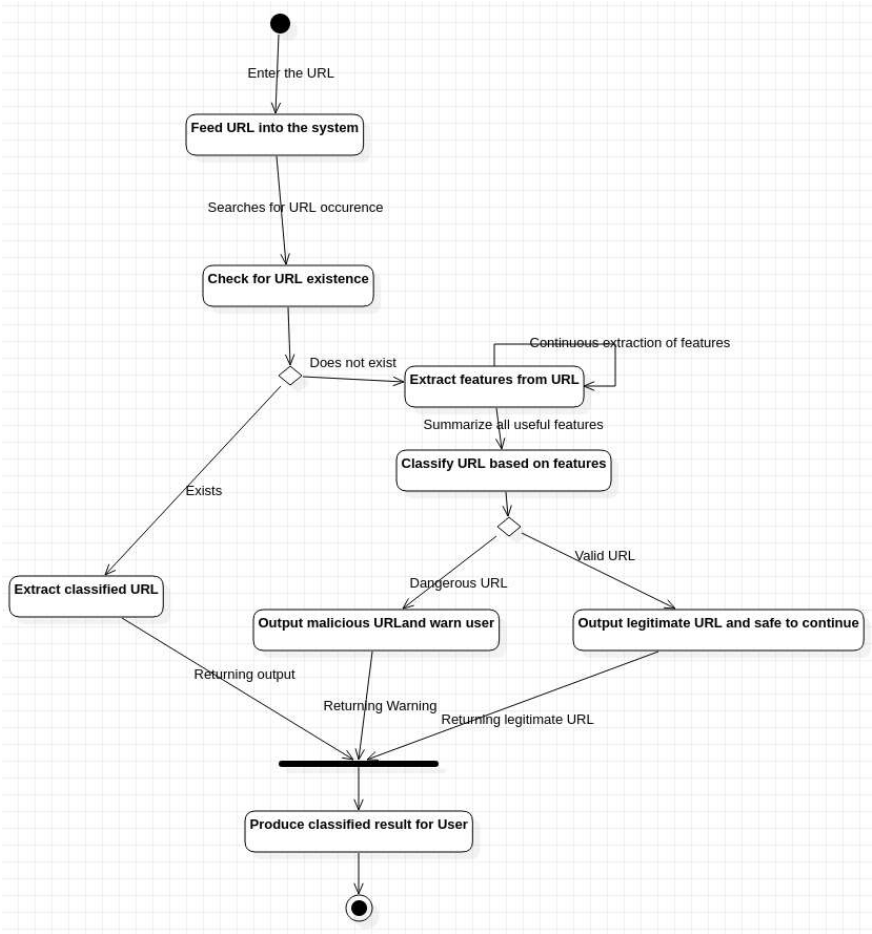


Fig 5.8 State Diagram

The state diagram is similar to the activity diagram, but instead represents the state in which the user is in as they go through each step of the process to obtain the final result.

5.6.4 User Interface Diagrams

The User Interface screen will contain a provision to enter a URL into the system and on entering it there will also be a provision of viewing the details regarding the trustability of the URL.

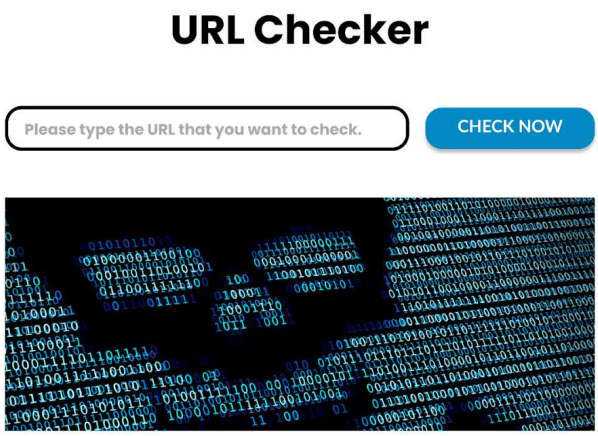


Fig 5.9 User Interface Diagrams

5.7 Report Layouts

A report that will cover the selection sorting and grouping criteria will contain the following details:

- Selection Criteria for the Features
- Selection Criteria for Data Extraction
- Grouping Criteria for validity of a Train and Testing Split
- Selection Criteria for the Models that should be used

This report is required to ensure that the method used to get the output is the best. It will make sure that the model is optimized to the best extent and isn't complex in terms of time and space. Also, it makes sure that the data that goes into the model is of good quality. This will lead to a system with little or no bias or overfitting. *Table layouts are given in Appendix E.*

5.8 External Interfaces

The diagram shown below shows the major interfaces of the product and how they will interact with one another. It highlights the fact that most of the implementation will be abstracted for the user. It also shows that the implementations should not affect the browser.

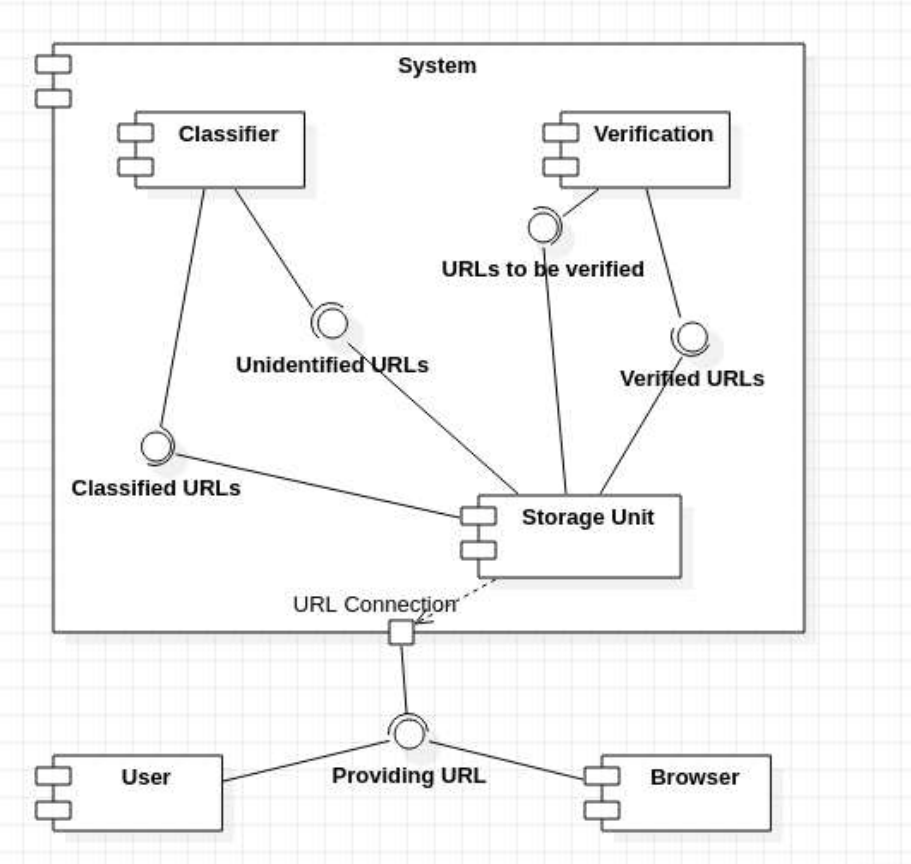


Fig 5.10 External Interfaces

5.9 Packaging and Deployment Diagram

The details of the physical implementation of each of the modules and their interactions are given below. This is how the system will work in a realistic sense.

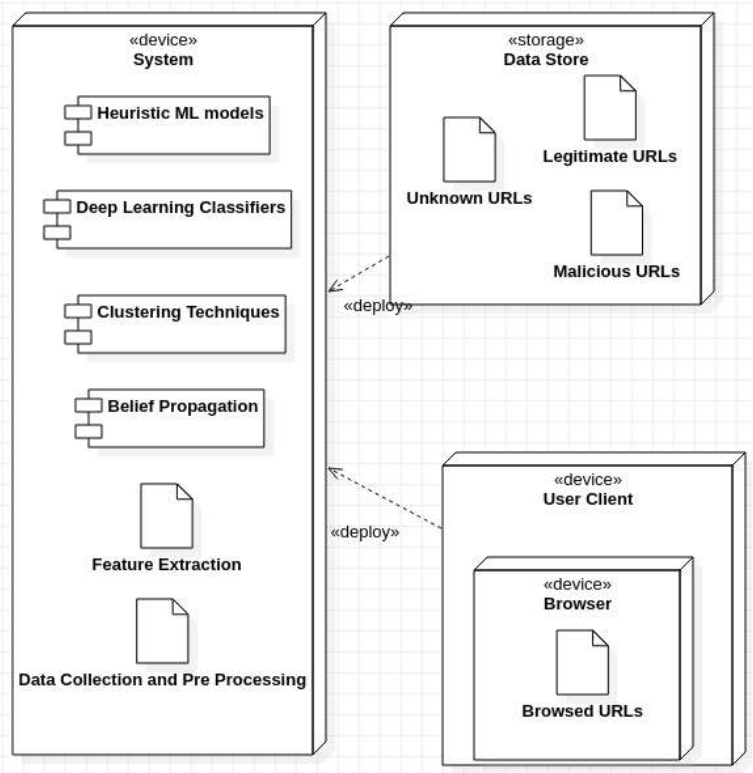


Fig 5.11 Packaging and Deploying Diagram

5.10 Help

This system may require some Python Libraries such as Beautiful Soup and search engine related libraries. The Python Documentation of these will be required.

5.11 Design Details

5.11.1 Novelty

- Unique combination of models used in the system for classification

5.11.2 Innovativeness

- Most techniques make use of only a single method for classification since it is a much faster process
- We combine methods to allow for a more flexible analysis of the URLs for improved efficiency, still maintaining feasibility of the system.

5.11.3 Interoperability

- The system should be able to execute over any operating system that we provide.
- Must be able to adapt to any browser for obtaining the URLs

5.11.4 Performance

- The system must be independent of the environment it is working in as described by the other factors and hence maintain a common performance ratio over all environments.
- Delivers the classified outputs with high accuracy without/irrespective of the influence of any external factors

5.11.5 Security

- Should not be affected by any external factors or from the URLs themselves if they are malicious
- Must be efficient in self defense techniques and provide warning for any form of system failure or threat prior to any form of ruination to it.

5.11.6 Reliability

- System must maintain a certain level of precision while classifying the URLs showing promising accuracy in any given environment.
- Must not falter in performance or produce false results of any sort or malfunction for any form of minor inconveniences.

5.11.7 Maintainability

- System must be able to produce warnings whenever it faces any issue with minimal need of major changes to it to get it working normally.
- Each module will be as independent of each other so as to prevent cascading of issues through the system.

5.11.8 Portability

- The system can function over any platform and environment that it is exposed to without faltering and maintaining performance

5.11.9 Legacy to modernization

- The system will serve as a big step towards the transformation from the legacy method i.e. BlackListing and WhiteListing to a much more automated and smart method
- Not only will the URLs that were previously classified as malicious be identified but other URLs that show similar behavior will also be identified

5.11.10 Reusability

- The lexical analysis related component of the model can be reused for any Natural Language Processing related application where inference must be obtained from text
- The web graph related component can be used for many search engine related applications such as Search Engine Optimizers, topic modelling, page rank algorithms etc.

5.11.11 Application compatibility

- The application will be browser independent
- However, it will not be language independent and can only be used for URLs with English language since some of the data is based on the contents of the URLs

5.11.12 Resource utilization, Etc.,

- The feature selection reduces the space complexity
- All the models used will be optimized thus making sure that there is no excessive resource utilization.

Appendix A: Definitions, Acronyms and Abbreviations

1. **URL :** Uniform Resource Locator
2. **IP :** Internet Protocol - Refers to the protocol address of the given link
3. **SSL :** Secure Socket Layers that enables encrypted communication between a web browser and a web server.
4. **BlackList :** Set of Malicious URLs
5. **WhiteList :** Set of Non-Malicious URLs
6. **Backlinks :** It is a link from some other website to that web resource.
7. **Page Rank :** It is an algorithm used by Google Search to rank web pages in their search engine results

Appendix B: References

- [1] https://dl.acm.org/doi/abs/10.1145/3231053.3231082
  Khulood Al Messabi, Monther Aldwairi, Ayesha Al Yousif, Anoud Thoban, and Fatna Belqasmi. 2018. Malware detection using DNS records and domain name features. In *Proceedings of the 2nd International Conference on Future Networks and Distributed Systems* (*ICFNDS '18*). Association for Computing Machinery, New York, NY, USA, Article 29, 1–7. DOI:https://doi.org/10.1145/3231053.3231082
- [2] https://www.hindawi.com/journals/wcmc/2018/4678746/
  Ping Yi, Yuxiang Guan, Futai Zou, Yao Yao, Wei Wang, Ting Zhu, "Web Phishing Detection Using a Deep Learning Framework", *Wireless Communications and Mobile Computing*, vol. 2018, Article ID 4678746, 9 pages, 2018. https://doi.org/10.1155/2018/4678746
- [3] https://link.springer.com/article/10.1007/s00521-020-05354-z
  Rao, R.S., Pais, A.R. & Anand, P. A heuristic technique to detect phishing websites using TWSVM classifier. *Neural Comput & Applic* (2020). https://doi.org/10.1007/s00521-020-05354-z
- [4] https://ieeexplore.ieee.org/document/8821879
  A. S. Manjeri, K. R., A. M.N.V. and P. C. Nair, "A Machine Learning

Approach for Detecting Malicious Websites using URL Features," *2019 3rd International conference on Electronics, Communication and Aerospace Technology (ICECA)*, 2019, pp. 555-561, doi: 10.1109/ICECA.2019.8821879.

- [5]https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9082616
  M. Sameen, K. Han and S. O. Hwang, "PhishHaven—An Efficient Real-Time AI Phishing URLs Detection System," in *IEEE Access*, vol. 8, pp. 83425-83443, 2020, doi: 10.1109/ACCESS.2020.2991403.
- [6]https://docs.google.com/document/d/1BAQRMT5p6TjQG2OHG7_4-q-JdHGFQ_Mmcz_jrXvcuFs/edit?usp=sharing

Appendix C: Record of Change History

#	Date	Document Version No.	Change Description	Reason for Change
1.	09/04/2021	01	Filled in all the subheadings as per requirements specified	Completion as well as definition of each level of the system's high level design.

Table 5.1 Record of Change History

Appendix D: Traceability Matrix

Project Requirement Specification Reference Section No. and Name.	DESIGN / HLD Reference Section No. and Name.
1. Introduction	1. Introduction
2. Literature Survey or Existing System	2. Current System
3.1. Product Features	4.3. Project Management and Code Organization
3.2.User Classes and Characteristics	4.1. Logical User Groups

3.3. Operating Environment	10. Packaging and Deployment Diagram
3.4.General Constraints, Assumptions and Dependencies	3.3. Constraints, Assumptions and Dependencies
4. Use Case Diagram	4.1. Logical User Groups
5. Functional Requirements	3.2. Architecture Choices 4. High-Level Design Diagram{4.2,4.3}
6. External Interface Requirements	7. User Interfaces Diagram 9. External Interfaces
7. Non-Functional Requirements	4.4. Security Features 12. Design Details

Table 5.2 Traceability Matrix

Appendix E: Report Layouts

1. Feature Selection

Correlation Coefficient	Number of Examples Available	Computational Resources Required	VIF	Can more features be derived from it?
Defines the level to which two features are correlated to each other. Features must have a strong correlation and must not result	Adequate number of examples with a good variation in them should be provided in order to get desirable	The features must be computationally feasible to use for a comparison and should not be	Variance inflation Factor is used to determine the covariance between the features. Lower the value, better	This is an advantage if we have many features to test against as it allows us to introduce uniqueness in the

in spurious values	results		the results	classification thus improving on the accuracy.

Table 5.3 Report Layout for Feature Selection

2. Data Extraction

Content of URL Accessible	Popularity of Dataset(if dataset)	Language of URL Content(Rejected if not English)
It is important to ensure that we have the URL as well as the contents available so as to perform feature extraction for the same.	Higher the popularity of the dataset, the more reliable it becomes to use as it has been considered a genuine source.	It is impossible to combine multiple languages as the models will not be able to understand them.

Table 5.4 Report Layout for Data Extraction

3. Grouping Criteria for validity of a Train and Testing Split

Cross-Validation	Data Variety	Data volume
Resampling technique that detects overfitting, ie, failing to generalize a pattern	Higher the variety in both the testing as well as training datasets, the better the outcome in terms of accuracy since we reduce biased values.	Volume plays an important role in deciding the ratio in which we split the data. Larger the size of the dataset, smaller the ratios

Table 5.5 Report Layout for Grouping Criteria for the Validity of Train and Testing Split

4. Models that should be used

For a neural network

Number of tunable parameters	Linear Separability of Data	Memory Requirements
Improves the precision and accuracy of the data	The kind of scatter plot obtained for data and whether it can be classified with a line	A neural network that performs similar with lesser memory usage will be preferred

Table 5.6 Report Layout for a Neural Network Model Evaluation

For a graph clustering algorithm

Fowlkes-M allows scores	Mutual information based scores	Completeness	Homog eneity	V-measure	Time Complexity
The Fowlkes-M allows Score is an evaluation metric to evaluate the similarity among clusterings obtained after applying different clustering algorithms.	A symmetric metric that is independent of the absolute value of the variables.	Describes a measure for analyzing whether all the data points have been classified into its respective cluster and each data point belongs to only one cluster	Homog eneity describe s the closene ss of the clusteri ng algorith m to this perfecti on.	Developed by the combinatio n of the previous 2 terms	Graph related algorithms take a lot of time as it has to explore every relevant node. Hence the time complexity involved is important

Table 5.7 Report Layout for a Graph Clustering Algorithm Evaluation

CHAPTER-6
CONCLUSION OF CAPSTONE PROJECT PHASE-1

So far the Project Requirements Specification and System Architecture has been prepared. A few datasets have been identified and they will be integrated. Also, some Python libraries have been tested to see if it can perform feature extraction for search engine related data and parsing HTML tags.

Through the course of designing the project, we have come across a variety of malpractices as well as methods to prevent them. They allow us to understand the issues we face in our everyday lives with the internet system. In the process, we have also understood plenty of concepts and new methods as well as technologies that allow us to improve on the security as well as the defense mechanism against such attacks.

Some of these include the advancement in programming languages such as Python, as well as the tools used for creating various architectures of models as well as systems that help in detecting, analyzing as well as preventing the rise of multiple threats such as this.

We have explored a majority of these, inclusive of data extraction methodologies using BeautifulSoup, selection of attributes from the data using data analytics through python and so on. The literature survey gives us a deep insight into the possibilities of choosing the best system to build, it's advantages and disadvantages and how to overcome any sort of difficulties through these.

To conclude as mentioned throughout the document, we have presented the requirements through the System Requirements Specifications which elucidate on the necessities in each field, as well as a high level design through the HLD document explaining the basic architecture and design of our system that we shall be building, allowing us to pinpoint areas we can improve upon, and thus aim at creating a smooth working and functioning model that serves its purpose.

CHAPTER-7

PLAN OF WORK FOR CAPSTONE PHASE-2

Capstone Phase-2 will consist mainly of the project implementation and testing. Additional literature survey will be done if required.

REFERENCES AND BIBLIOGRAPHY

- [1] Manjeri, A. S., K. R., A. M.n.v., and P. C. Nair. 2019. “A Machine Learning Approach for Detecting Malicious Websites Using URL Features.” In *2019 3rd International Conference on Electronics, Communication and Aerospace Technology (ICECA)*, 555–61.
- [2] Messabi, Khulood Al, Monther Aldwairi, Ayesha Al Yousif, Anoud Thoban, and Fatna Belqasmi. 2018. “Malware Detection Using DNS Records and Domain Name Features.” In *Proceedings of the 2nd International Conference on Future Networks and Distributed Systems*, 1–7. ICFNDS ’18 29. New York, NY, USA: Association for Computing Machinery.
- [3] Rao, Routhu Srinivasa, Alwyn Roshan Pais, and Pritam Anand. 2020. “A Heuristic Technique to Detect Phishing Websites Using TWSVM Classifier.” *Neural Computing & Applications*, 1–20.
- [4] Sameen, M., K. Han, and S. O. Hwang. 2020. “PhishHaven—An Efficient Real-Time AI Phishing URLs Detection System.” *IEEE Access* 8: 83425–43.
- [5] Yi, Ping, Yuxiang Guan, Futai Zou, Yao Yao, Wei Wang, and Ting Zhu. 2018. “Web Phishing Detection Using a Deep Learning Framework.” *Proceedings of the ... International Wireless Communications & Mobile Computing Conference / Association for Computing Machinery. International Wireless Communications & Mobile Computing Conference* 2018 (September). https://doi.org/10.1155/2018/4678746.