HIGH LEVEL DESIGN DOCUMENT

Web Page Classification for Safer Browsing

UE18CS390A – Capstone Project Phase – 1

Submitted by:

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

Under the guidance of

Dr. N Mehala
Designation
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

TABLE OF CONTENTS

1. Introduction

This document 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.

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.

3. Design Considerations

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.

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

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

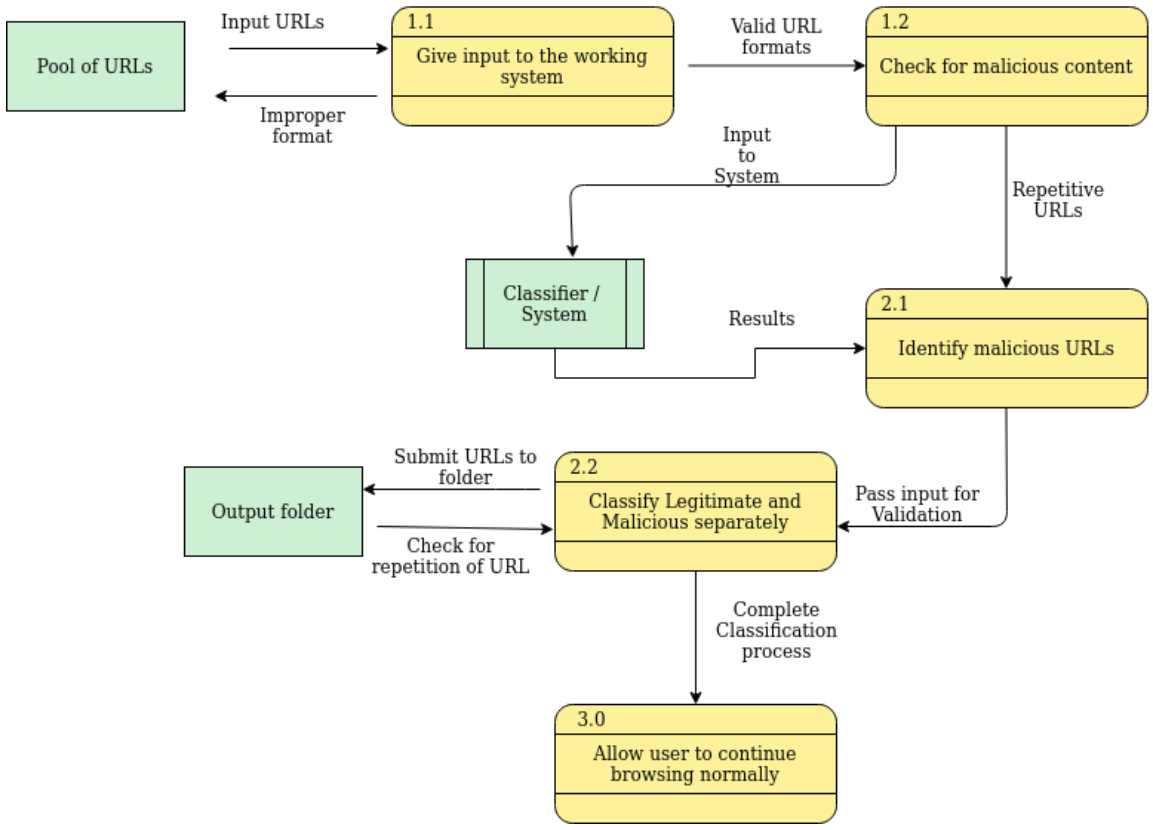
- Interoperability requirements
  - o System must be connected to a browsing system or must be connected to a source that gives URls as inputs.
  - o Might have issues with the interface of the browsing system; Needs to be compliant with all of them and run smoothly or any of them.
- Interface/protocol 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

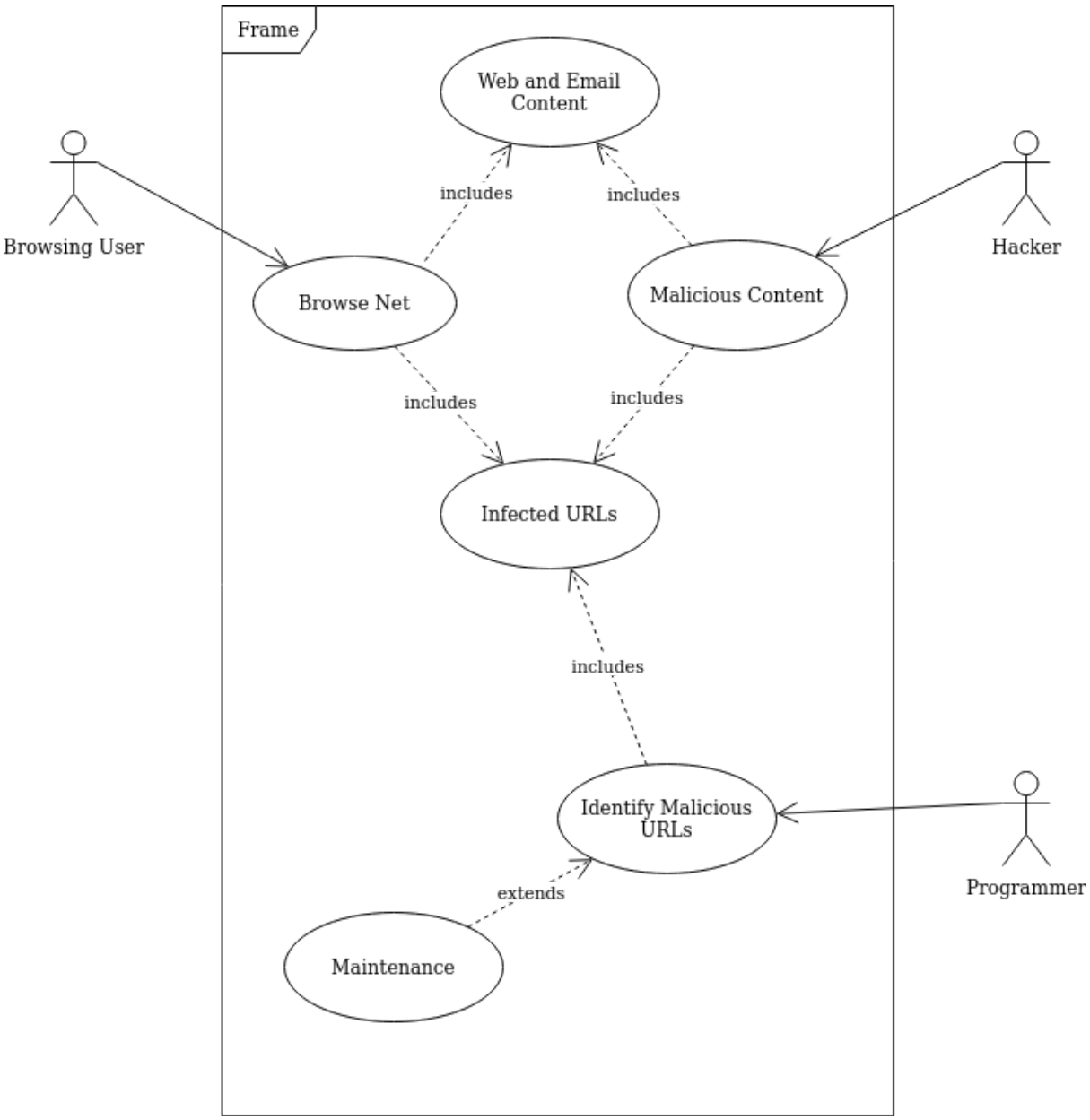
4. High Level System Design

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



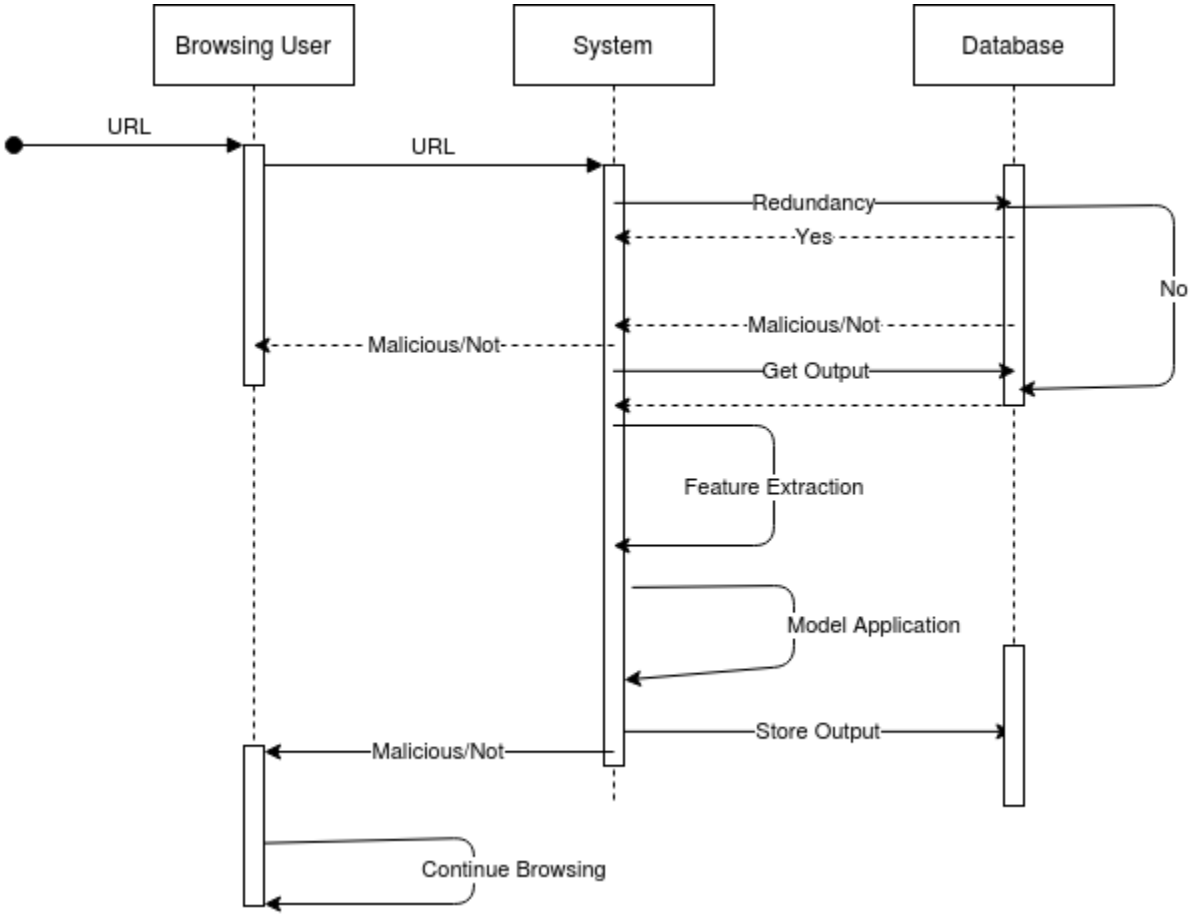
4.1. *Logical User Groups*



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

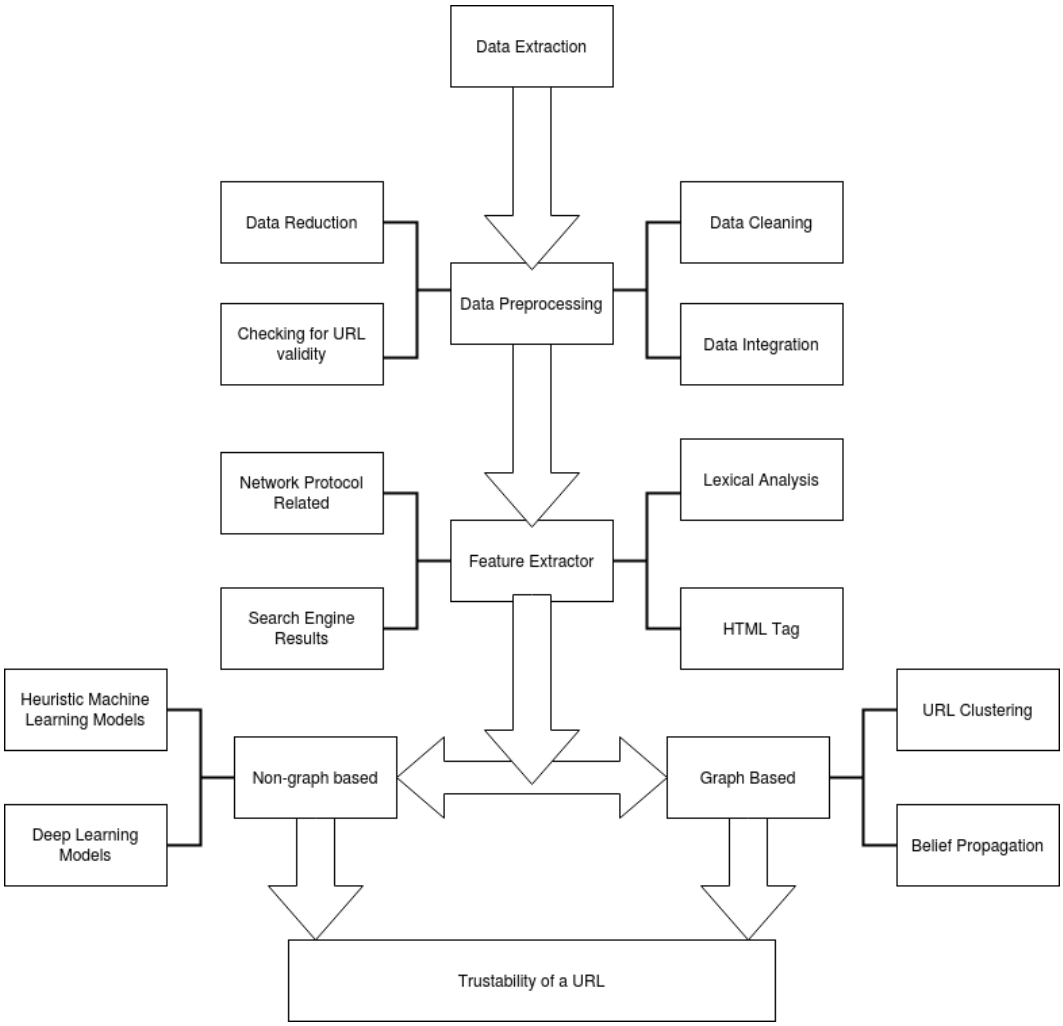
- **Browsing User**- A naive user who is trying to view the contents in a URL without knowing if it is malicious or not. The system is made for protecting this user.
- **Hacker**- Person who is creating malicious URLs to create malicious content and harm the Browsing User.
- **Programmer**- Person creating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

4.2. *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.

4.3. *Project Management and Code Organization*



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:

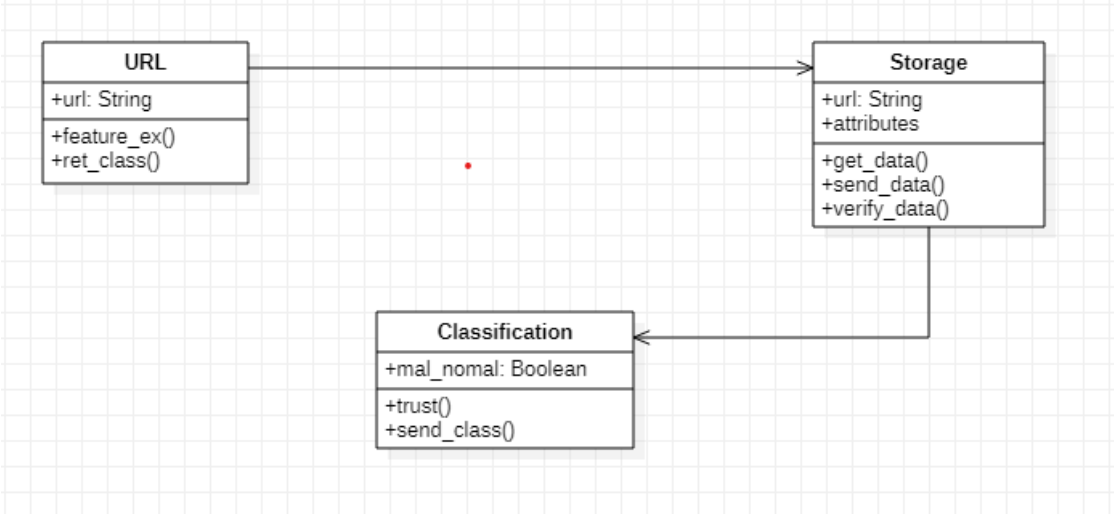
- Restriction to transaction traffic

- 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. Design Description

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



➢ **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.

5.2. *Reusability Considerations*

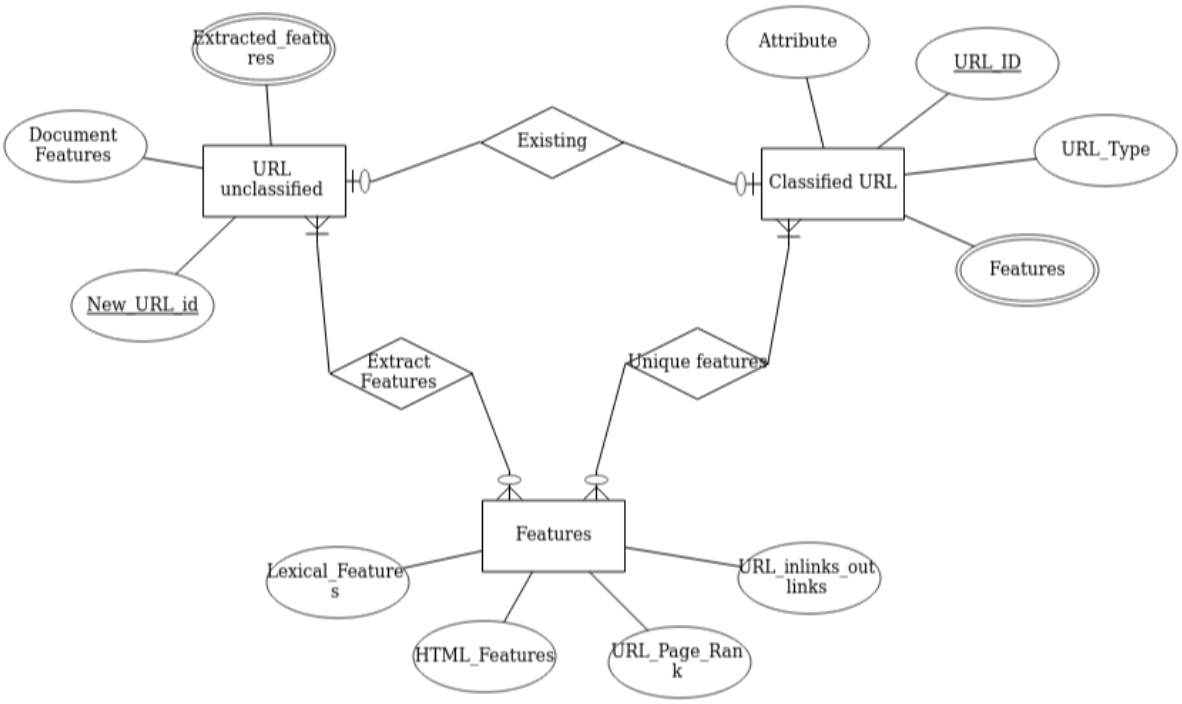
➢ **BeautifulSoup:** A Python library used to parse through HTML tags and get the values of attributes in them

➢ **Python SEO Analyzer:** For search engine related analysis and optimizations

➢ **Deep Learning Frameworks** such as TensorFlow

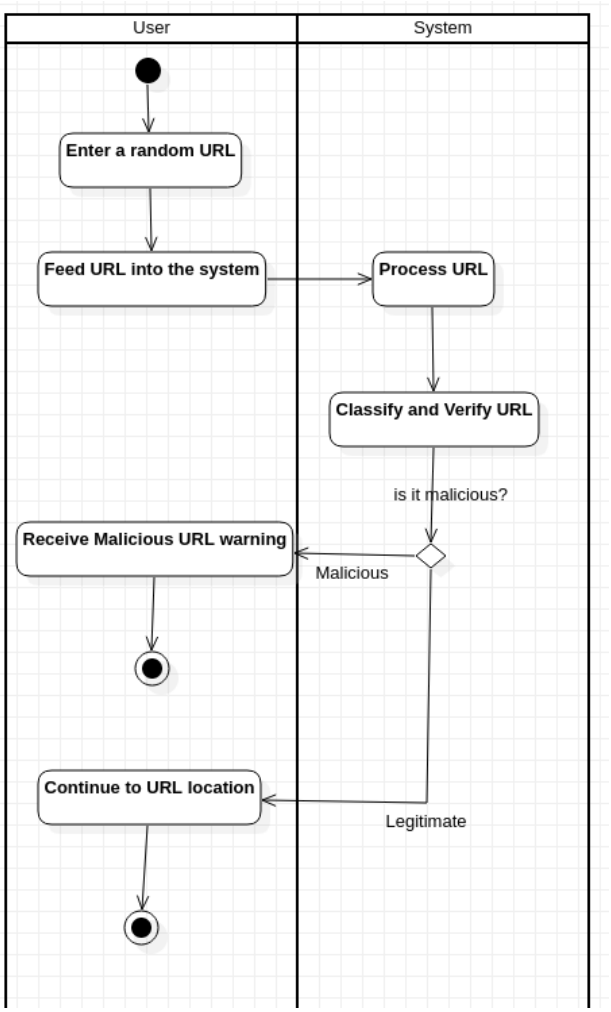
6. ER Diagram / Swimlane Diagram / State Diagram (include as appropriate)

6.1. 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

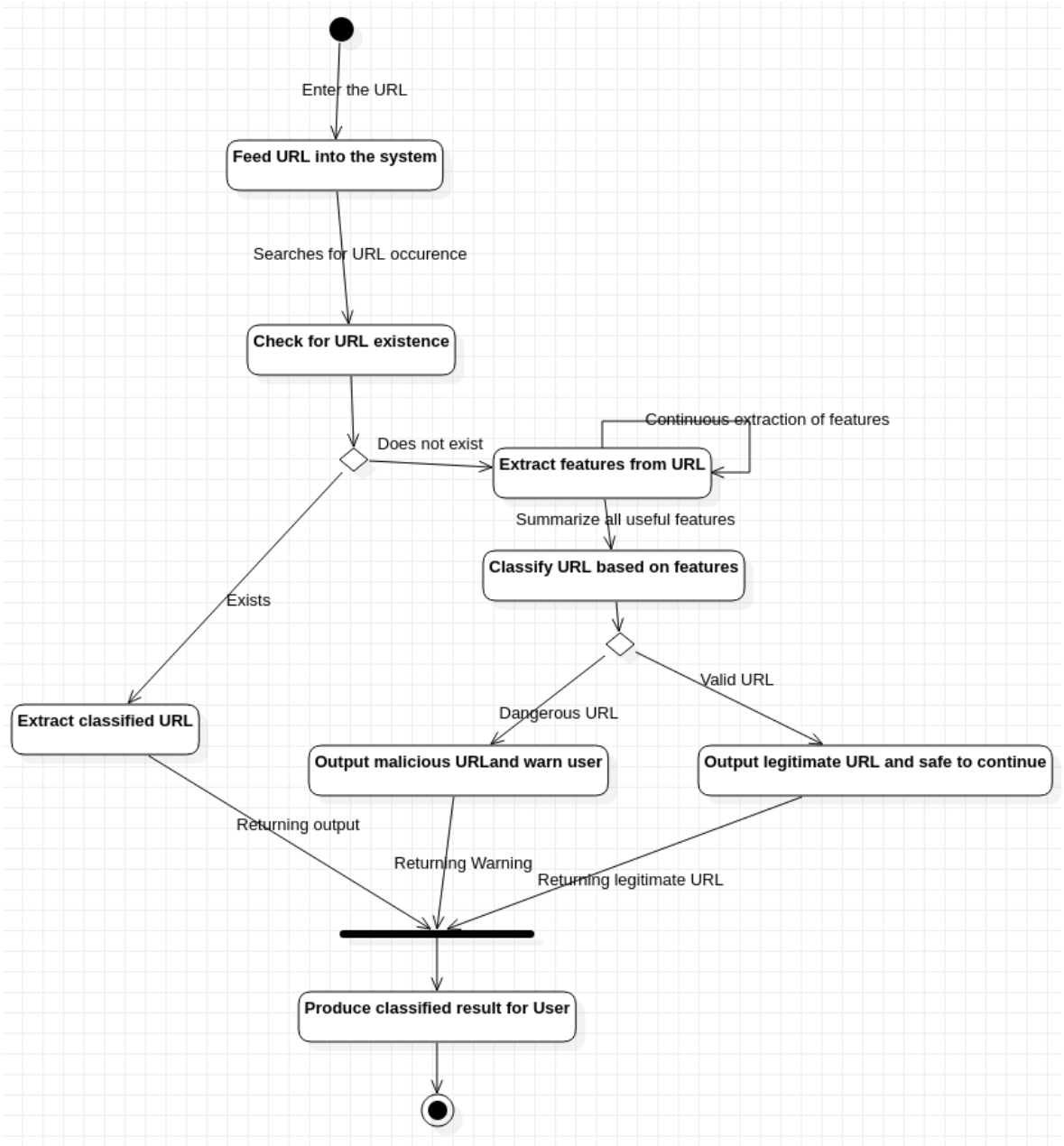
6.2. Activity Diagram



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.

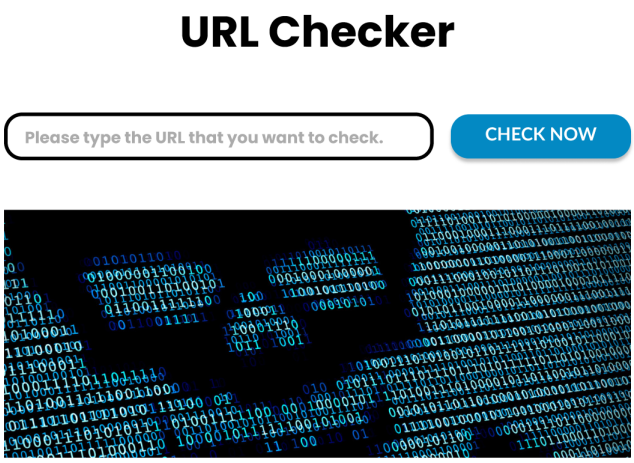
6.3. State Diagram



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.

7. 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.



8. Report Layouts

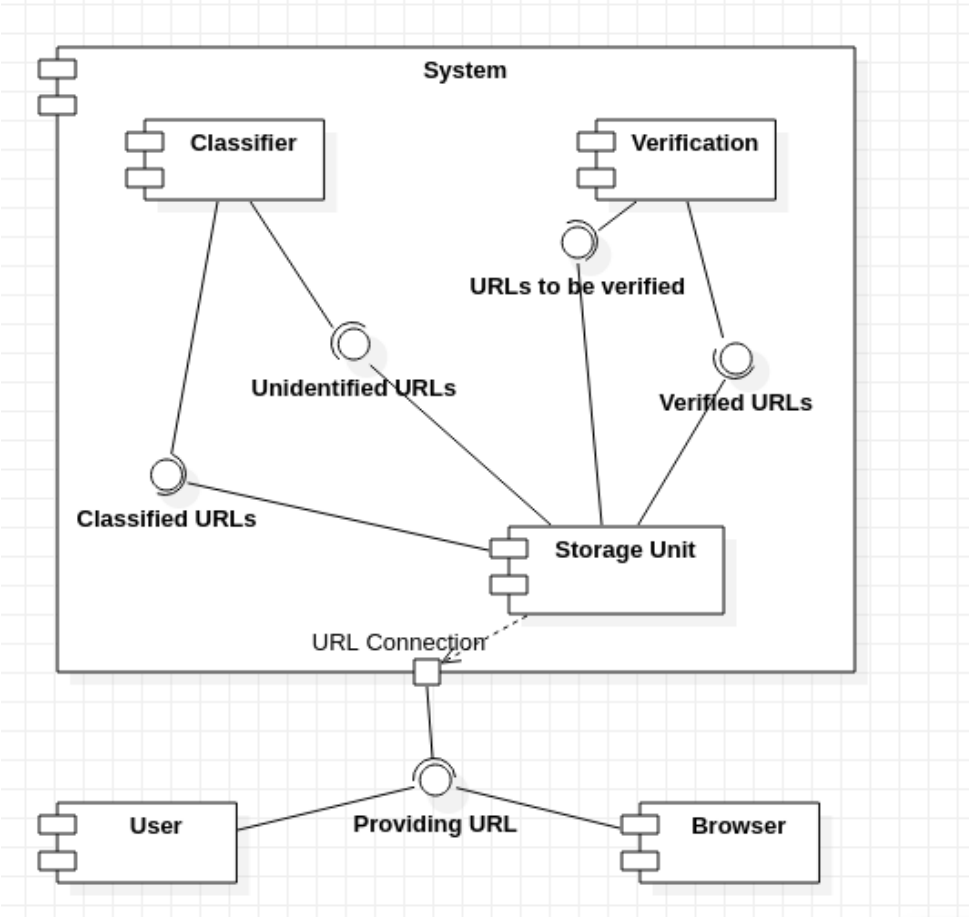
A report that will cover the selection sorting and grouping criteria will contain the following details:

- 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.*

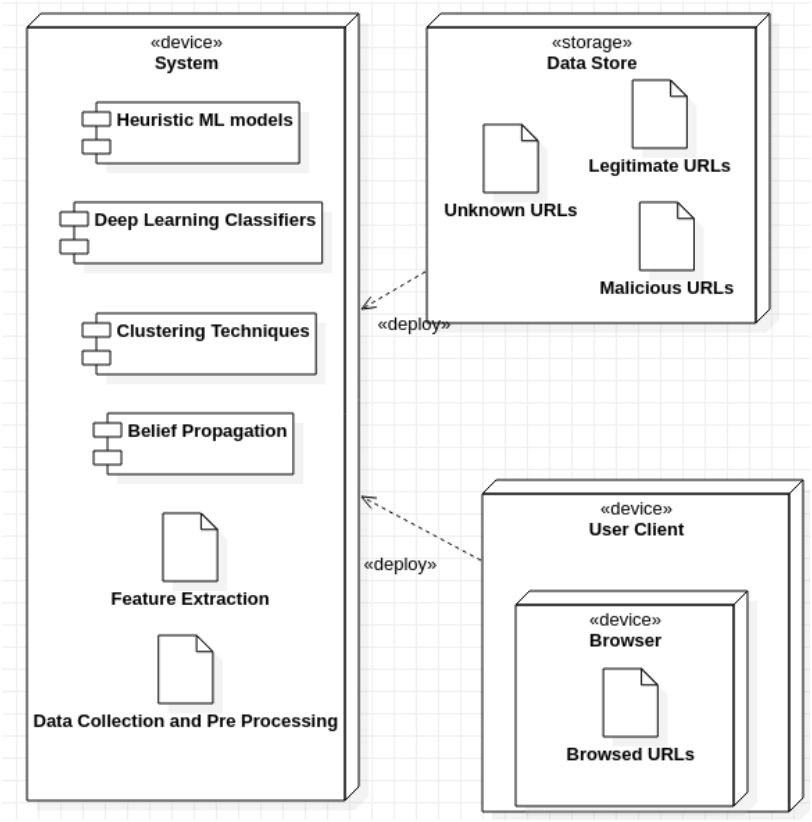
9. External Interfaces

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.



10. 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.



11. 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.

12. Design Details

12.1. *Novelty*

- Unique combination of models used in the system for classification

12.2. *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.

12.3. *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

12.4. *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

12.5. *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.

12.6. *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.

12.7. *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.

12.8. *Portability*

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

12.9. *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

12.10. *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.

12.11. *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

12.12. *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

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.
2.				
3.				

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

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 in spurious values	Adequate number of examples with a good variation in them should be provided in order to get desirable results	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 the results	This is an advantage if we have many features to test against as it allows us to introduce uniqueness in the classification thus improving on the accuracy.

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.

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

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

For a graph clustering algorithm

Fowlkes-Mallows scores	Mutual information based scores	Completeness	Homogeneity	V-measure	Time Complexity
The Fowlkes-Mallows Score is an evaluation metric to evaluate the similarity among clusterings obtained after applying different clustering algorithms.	This metric is independent of the absolute values of the labels: a permutation of the class or cluster label values won't change the **score** value in any way. This metric is furthermore symmetric	Completeness: A perfectly complete clustering is one where all data-points belonging to the same class are clustered into the same cluster. Completeness describes the closeness of the clustering algorithm to this perfection	Homogeneity describes the closeness of the clustering algorithm to this perfection.	Developed by the combination 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