**Thakur College of Science & Commerce**

Thakur Village, Kandivali East, Mumbai, Maharashtra 400101

# A

# PROJECT REPORT

# ON

# "URL Detection"

SUBMITTED BY

**Enoch Sudhakar Sawant**

**[T. Y. B.SC. CS 386]**

UNDER THE GUIDANCE OF

**PROF. GIRISH TERE**

Submitted in the partial fulfilment of the requirement for qualifying B.Sc. (C.S), Semester-V Examination.

**Mumbai University [2022-23]**

Thakur Educational Trust's (Regd.)

THAKUR COLLEGE OF SCIENCE & COMMERCE

AUTONOMOUS COLLEGE, PERMANENTLY AFFILIATED TO UNIVERSITY OF MUMBAI

NAAC Accredited Grade 'A' (3rd Cycle) & ISO 9001: 2015 (Certified)

Best College Award by University of Mumbai for the Year 2018-2019

CELEBRATING 25 YEARS OF GLORY

## DECLARATION BY LEARNER

I the undersigned, **Mr. Enoch Sawant** hereby, declare that the work embodied in this project work titled **"URL Detection"** forms my own contribution to the research work carried out under the guidance of Prof. **Girish Tere** is a result of my own research work and has not been previously submitted to any other University for any other Degree/Diploma to this or any other University.

Wherever reference has been made to previous works of others, it has been clearly indicated as such and included in the bibliography.

I, hereby further declare that all information of this document has been obtained and presented in accordance with academic rules and ethical conduct.

Certified By
Girish Tere

Learner
Enoch Sawant

Thakur Educational Trust's (Regd.)

THAKUR COLLEGE OF SCIENCE & COMMERCE

AUTONOMOUS COLLEGE, PERMANENTLY AFFILIATED TO UNIVERSITY OF MUMBAI

NAAC Accredited Grade 'A' (3rd Cycle) & ISO 9001: 2015 (Certified)

Best College Award by University of Mumbai for the Year 2018-2019

CELEBRATING 25 YEARS OF GLORY

# CERTIFICATE

This is to certify that **Mr. Enoch Sawant** has worked and duly completed his Project Work for the Degree of Bachelor under the Faculty of Science in Computer Science **(T.Y.B.Sc CS)**and the project is entitled, **URL Detection** towards the partial fulfilment of project work as prescribedby University of Mumbai for the year 2022-23.

I further certify that the entire work has been done by the learner and that no part of it has been submitted previously for any Degree or Diploma of any University.

| | |
|---|---|
| Prof. Ashish Trivedi | Prof. Girish Tere |
| Head of Department | Project Guide |

| | |
|---|---|
| Internal Examiner | External Examiner |

# ACKNOWLEDGEMENT

I would like to extend our heartiest thanks with a deep sense of gratitude and respect to all those who provides me immense help and guidance during my period.

I would like to thank my Project Guide **Prof.Girish Tere** for providing a vision about the system. I have been greatly benefited fromtheir regular critical reviews and inspiration throughout my work. I am grateful to them for their guidance, encouragement, understanding and insightful support in the development process.

I would also like to thank my college for giving required resources whenever I wanted and for giving the opportunity to develop the project.

I would like to express my sincere thanks to our Head of Department **Prof. Ashish Trivedi** for having facilitated us with the essential infrastructure & resources without which this project would not have seen light of the day.

I am also thankful to entire staff of **CS** for their constant encouragement, suggestions and moral support throughout the duration of my project.

Last but not the least I would like to mention here that I am greatly indebted to each and everybody my friends and who has been associated with my project at any stage but whose name does not find a place in this acknowledgement.

With sincere regards,
***Enoch Sudhakar Sawant***

# CONTENTS

## ABSTRACT

Malicious URLs host various unsolicited content and can pose a high threat to potential victims. Therefore, a fast and efficient detection technique is needed. In this thesis, we focus on the problem of detecting malicious URLs based on the information obtained from URLs using machine-learning technologies.

# CHAPTER 1

# INTRODUCTION

## 1.1 OBJECTIVE

Recently, with the increase in Internet usage, cybersecurity has been a significant challenge for computer systems. Different malicious URLs emit different malicious software and try to capture user information. Signature-based approaches have often been used to detect such websites and detected malicious URLs have been attempted to restrict access by using various security components.

## 1.2 INTRODUCTION

Nowadays the Internet has become a platform to communicate and share information. Nevertheless, it is a dangerous place that supports a wide range of criminal activities. Technological advancements have given rise to advanced techniques that attack and scam users. Although motivations behind these criminal activities may differ, most of them require users to perform some action, such as clicking, which redirects to a specific webpage or downloads some content.

Most of the attacking techniques are spread through sharing of compromised websites. Compromised URLs are websites that contain malicious content on a trusted website. URL, also known as Uniform Resource Locator, is the universal address for a resource on the Internet. It is used as a vector that makes Internet users susceptible to cybercrimes. Popular attacks by using these compromised URLs include phishing, spamming, social engineering, and drive-by-download. These attacks are performed by exploiting vulnerabilities of websites and tend to cause billions of dollars' worth of damage every year. According to Ponemon Institute, it costs around $5 million on average when an attack succeeds. This includes productivity loss, information theft, reputation damage, infrastructure damage, etc. There are different techniques to make malicious URLs look legitimate. The most commonly used technique is obfuscation, which is classified into four types: obfuscation of host with IP, with another domain, with large hostnames, and misspelling. A new method for obfuscating malicious URLs is through shortening services.

URL shortening services have emerged in the recent years. There are hundreds of shortening services available for everyone, usually for free. The shortening services obfuscate the actual link that make it difficult for users to identify if the website is malicious or benign. To avoid this, many shortening services use blacklists to filter the malicious URLs even before shortening them. Few shortening service provides a preview or statistics of the webpage. For example, google and bitly shortening services use '+' symbol at the end of the short URL; consequently, tinyurl shortening service use 'preview' before the website name.

For example, http://preview.tinyurl.com/abcXYZ. Many users are becoming victims by not calculating the risk when clicking a suspicious URL. If there is any way to inform users beforehand that the website is malicious, many people can be saved. This project aims to provide users with an effective solution by detecting those malicious URLs using machine learning approaches.

## 1.3 SCOPE

The purpose of our project is to increase the accuracy of predicting whether a particular hosted website is malicious or benign, generally antivirus companies such as (Norton, Kaspersky etc.) use blacklisting method in order to detect whether the website is malicious or benign. In this project we implement different machine learning model. The focus will be mainly on techniques using machine learning, considering they are achieving the best results.

# CHAPTER 2

## HARDWARE & SOFTWARE REQUIREMENT

# SYSTEM REQUIREMENT

## 2.1 HARDWARE REQUIREMENT

- Processor: Any Processor above 500 MHz
- RAM: 512Mb
- Hard Disk: 10 GB
- Input device: Standard Keyboard and Mouse
- Output device: High Resolution Monitor

## 2.2 SOFTWARE REQUIREMENT

- OS-Windows 7/8/10
- Python
- Machine Learning
- Python IDLE
- Jupyter Notebook
- Python Libraries

# CHAPTER 3

# TECHNICAL DESCRIPTION

## 3.1 FUNCTIONAL REQUIREMENTS

In software engineering, a functional requirement defines a function of a software system or its component. A function is described as a set of inputs, the behavior, and outputs. Functional requirements may be calculations, technical details, data manipulation and processing and other specific functionality that define what a system is supposed to accomplish. Behavioral requirements describing all the cases where the system uses the functional requirements are captured in use cases.

Here, the system has to perform the following tasks:

a. User enters a URL within the square brackets, he/she doesn’t know that the entered URL is benign or malicious.

b. According to the dataset provided and all the URL features, query features of an URL, it is trained and tested.

c. The training and testing part is done by logistic regression and Support vector machine algorithms. Tfid vectorizer is also used for vector transformation in the first phase.

d. The entered URL is then predicted as good or bad i.e. benign or malicious.

## 3.2 NON-FUNCTIONAL REQUIREMENTS

In systems engineering and requirements engineering, a non-functional requirement is a requirement that specifies criteria that can be used to judge the operation of a system, rather than specific behaviors. This should be contrasted with functional requirements that define specific behavior or functions. The plan for implementing functional requirements is detailed in the system design. The plan for implementing non-functional requirements is detailed in the system architecture. Other terms for non-functional requirements are "constraints", "quality attributes", "quality goals", "quality of service requirements" and "non-behavioral requirements".

# CHAPTER 4

## SOFTWARE ANALYSIS & DESIGN

## 4.1 Software Analysis

## Design Goals:

**HIGH ACCURACY: -**

This is one of the important goals to be achieved for any malicious URL detection. We want to maximize the detection of all the threats of by minimizing the detection of classifying benign URLs into malicious. Since no system is capable of perfect detection accuracy it has to differentiate between the ratios of benign and malicious by setting different levels of detection thresholds.

**FAST DETECTION: -**

An ideal system should be able to detect the malicious URL immediately and then block the URL to prevent any threats and harms to the public. The requirement of detection speed could be more severe, which sometimes needs the detection to be done in milliseconds such that a malicious URL request can be blocked immediately in real time whenever a user clicks on any URLs.

**HIGH SCALABILITY: -**

Increasing huge number of URLs, a real-world malicious URL detection system must scale up for training the models of training data (millions or billions). To achieve the high scalability desire, first explore more efficient and scalable algorithms and then the second is to build scalable learning systems in distributed computing environments.

**STRONG ADAPTATION: -**

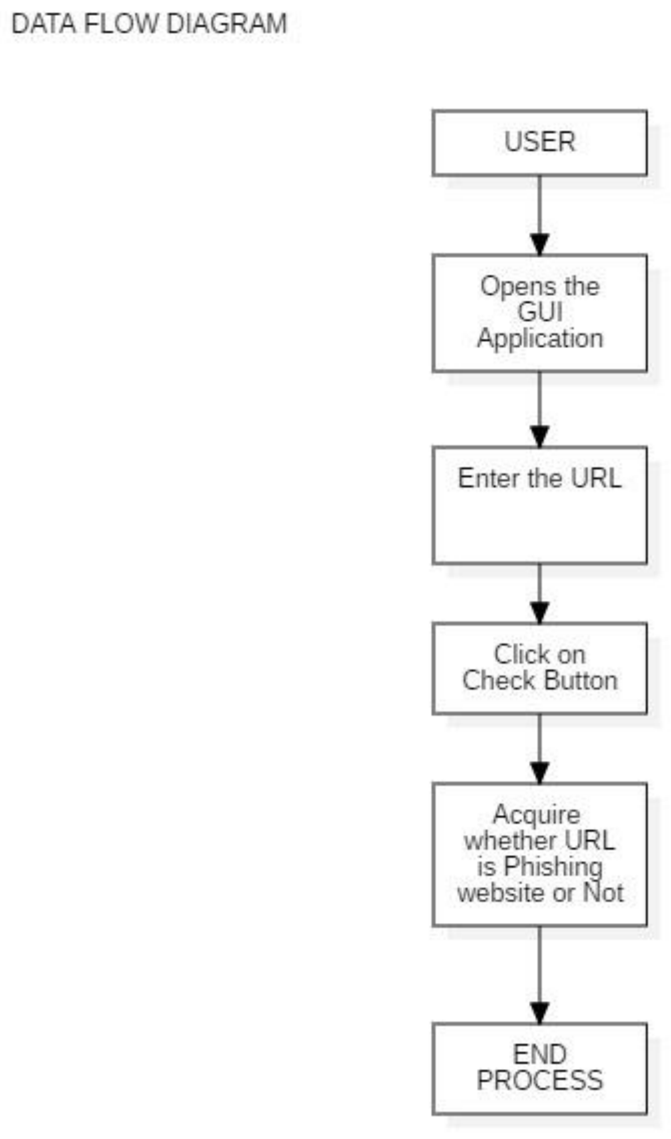
A real-world malicious URL detection system has to deal with a variety of practical complexity, including adversarial patterns such as concept drifting where the distribution of malicious URLs may change over time or even change in adversarial way to bypass the detection system, missing values, increasing number of new features, etc. A real- world malicious URL detection system must have a strong adaptation ability to work Effectively and robustly under most circumstances.

**GOOD FLEXIBILTY: -**

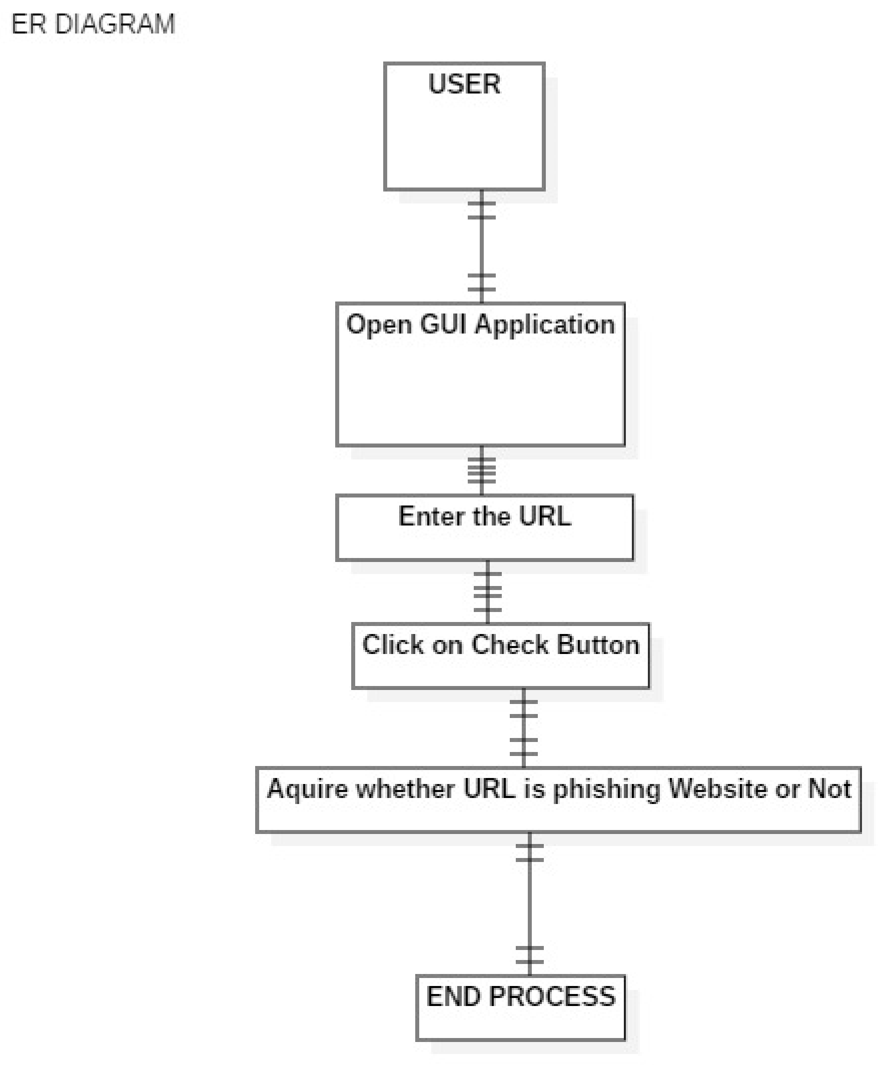
Given the high complexity of malicious URL detection of malicious URL detection, a real-world malicious URL detection system with machine learning should be designed with good flexibility for easy improvements and extensions. These include the quick update of the predictive models with respect to new training data, being easy to replace the training algorithm and models whenever there is a need, being flexible to be extended for training algorithm and the models whenever there is a need, being flexible to be extended for training models to deal with variety of new attacks and threats, and finally being able to interact with human beings whenever there is a need.

## 4.2 SOFTWARE DESIGN

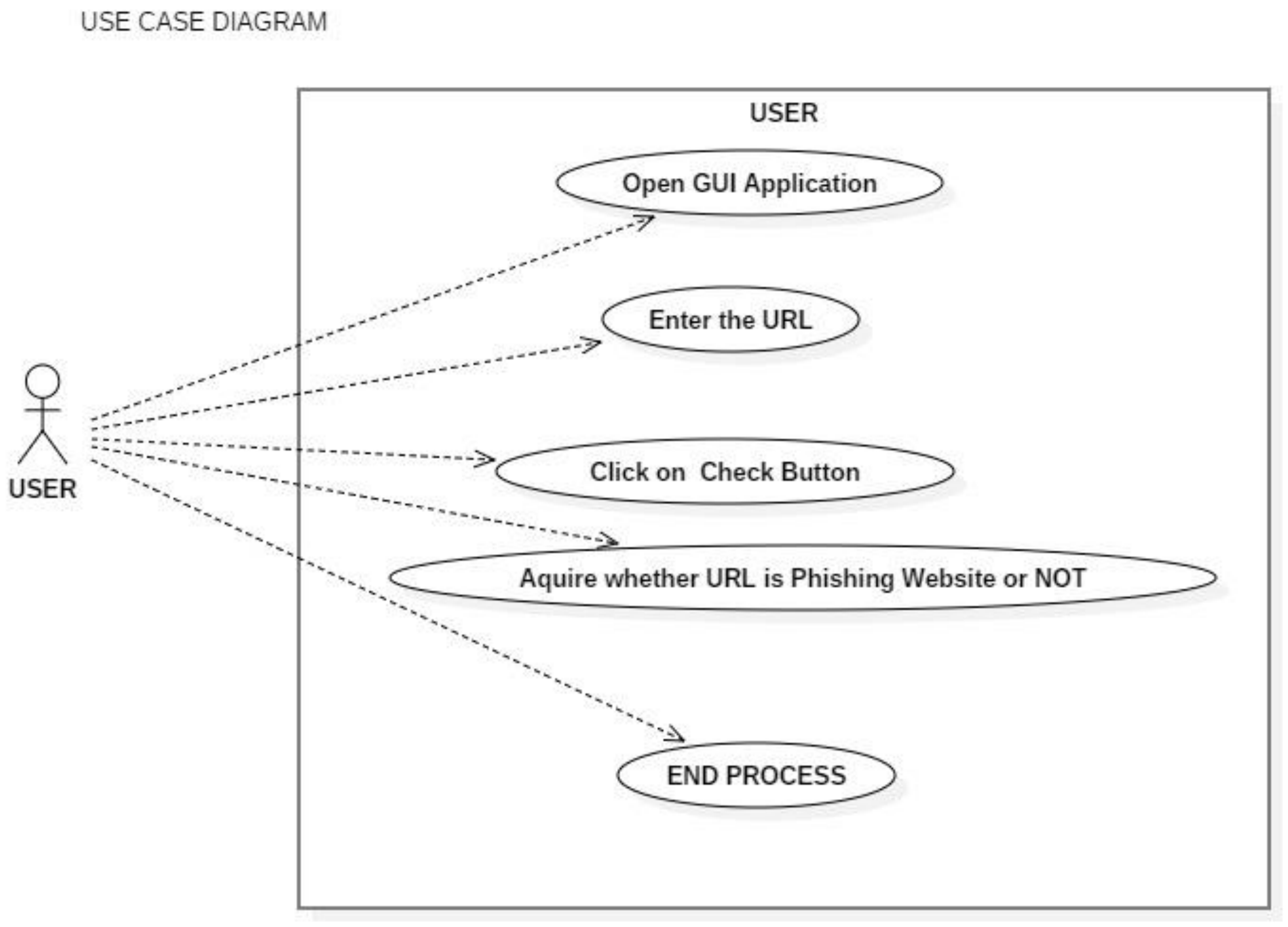
### 4.2.1 Data Flow Diagram


DATA FLOW DIAGRAM
USER
Opens the GUI Application
Enter the URL
Click on Check Button
Acquire whether URL is Phishing website or Not
END PROCESS

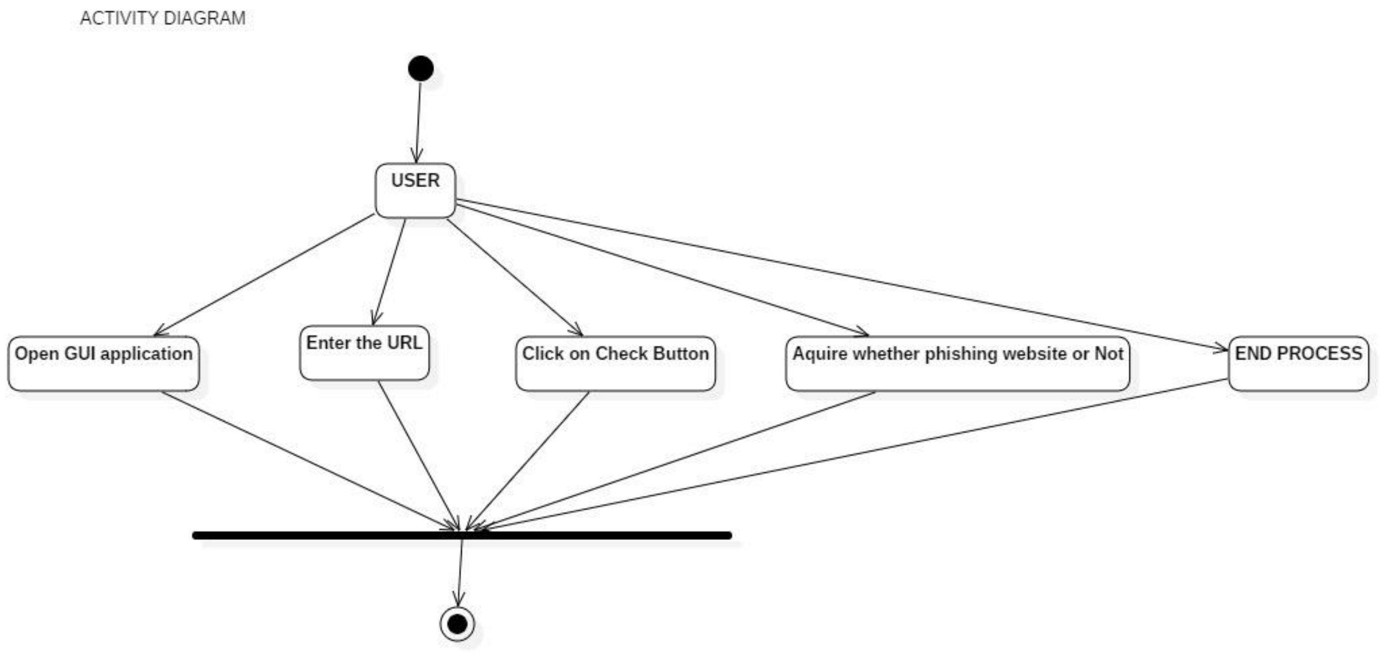
## 4.2.2 ER DIAGRAM



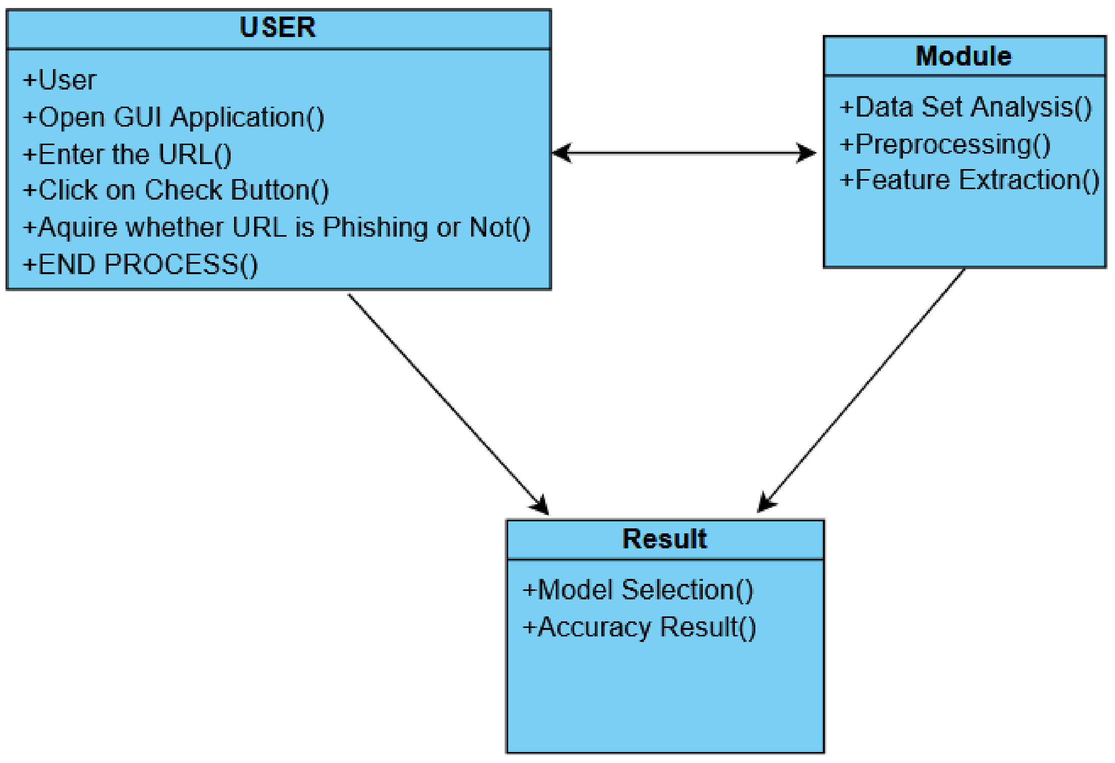
## 4.2.3 USE CASE DIAGRAM



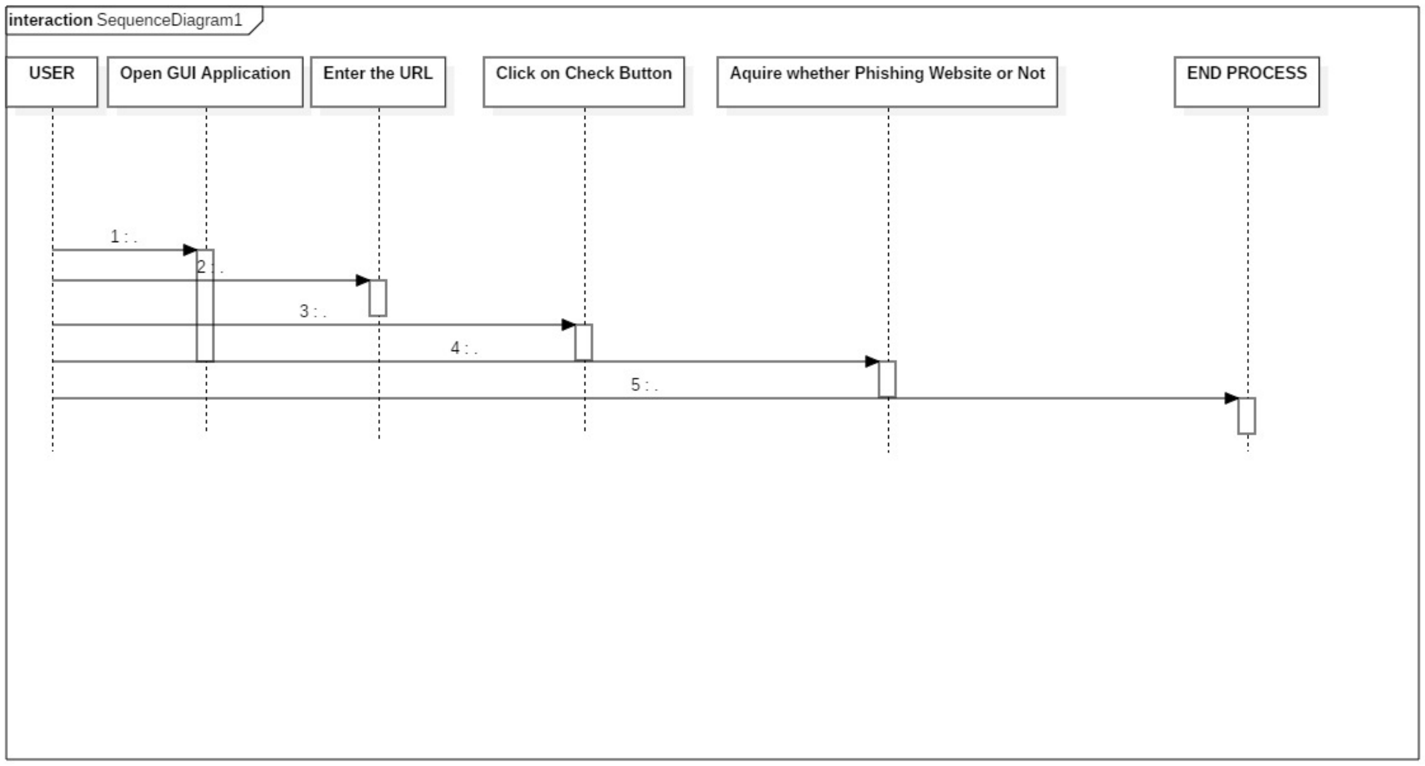
## 4.2.4 ACTIVITY DIAGRAM



## 4.2.5 CLASS DIAGRAM



## 4.2.6 SEQUENCE DIAGRAM



# CHAPTER 5

## DEVELOPMENT

## 5.1 SOURCE CODE

## SOURCE CODE

### gui.py: -

```
from PyQt5 import QtCore, QtGui, QtWidgets #works for pyqt5
import feature_extractor

class Ui_Spam_detector(object):
    def setupUi(self, Spam_detector):
        Spam_detector.setObjectName("Spam_detector")
        Spam_detector.resize(521, 389)
        self.centralwidget = QtWidgets.QWidget(Spam_detector)
        self.centralwidget.setObjectName("centralwidget")

"""check button code and its connectivity to button_click function"""
        self.check_button = QtWidgets.QPushButton(self.centralwidget)
        self.check_button.setGeometry(QtCore.QRect(210, 170, 93, 28))
        self.check_button.setObjectName("check_button")
        self.check_button.clicked.connect(self.button_click)

        """url input section"""
        self.url_input = QtWidgets.QLineEdit(self.centralwidget)
        self.url_input.setGeometry(QtCore.QRect(70, 111, 431, 31))
        self.url_input.setObjectName("url_input")

        self.label = QtWidgets.QLabel(self.centralwidget)
        self.label.setGeometry(QtCore.QRect(20, 110, 81, 31))
        self.label.setObjectName("label")

        """output message"""
        self.output_text = QtWidgets.QTextEdit(self.centralwidget)
        self.output_text.setGeometry(QtCore.QRect(30, 241, 461, 121))
        self.output_text.setObjectName("output_text")

        self.label_2 = QtWidgets.QLabel(self.centralwidget)
        self.label_2.setGeometry(QtCore.QRect(110, 10, 311, 41))
        self.label_2.setObjectName("label_2")

        Spam_detector.setCentralWidget(self.centralwidget)
        self.statusbar = QtWidgets.QStatusBar(Spam_detector)
        self.statusbar.setObjectName("statusbar")
        Spam_detector.setStatusBar(self.statusbar)
        self.retranslateUi(Spam_detector)
        QtCore.QMetaObject.connectSlotsByName(Spam_detector)

    def retranslateUi(self, Spam_detector):
        _translate = QtCore.QCoreApplication.translate
        Spam_detector.setWindowTitle(_translate("Spam_detector",
"MainWindow"))
        self.check_button.setText(_translate("Spam_detector", "Check
"))
        self.label.setText(_translate("Spam_detector",
"<html><head/><body><p><span style=\" font-size:10pt;\">URL
:</span></p></body></html>"))
        self.label_2.setText(_translate("Spam_detector",
"<html><head/><body><p align=\"center\"><span style=\" font-
size:16pt;\">Spam URL Detector</span></p></body></html>"))

    def button_click(self):
        text = self.url_input.text()
        #print(text)
        obj = feature_extractor.feature_extractor(text)
        str1,str2 = obj.extract()

        self.output_text.append("{} \n{}\n\n".format(str1,str2))


    #def show_output():

if __name__ == "__main__":
    import sys
    app = QtWidgets.QApplication(sys.argv)
    Spam_detector = QtWidgets.QMainWindow()
    ui = Ui_Spam_detector()
    ui.setupUi(Spam_detector)
    Spam_detector.show()
    sys.exit(app.exec_())
```

**feature_extractor.py: -**

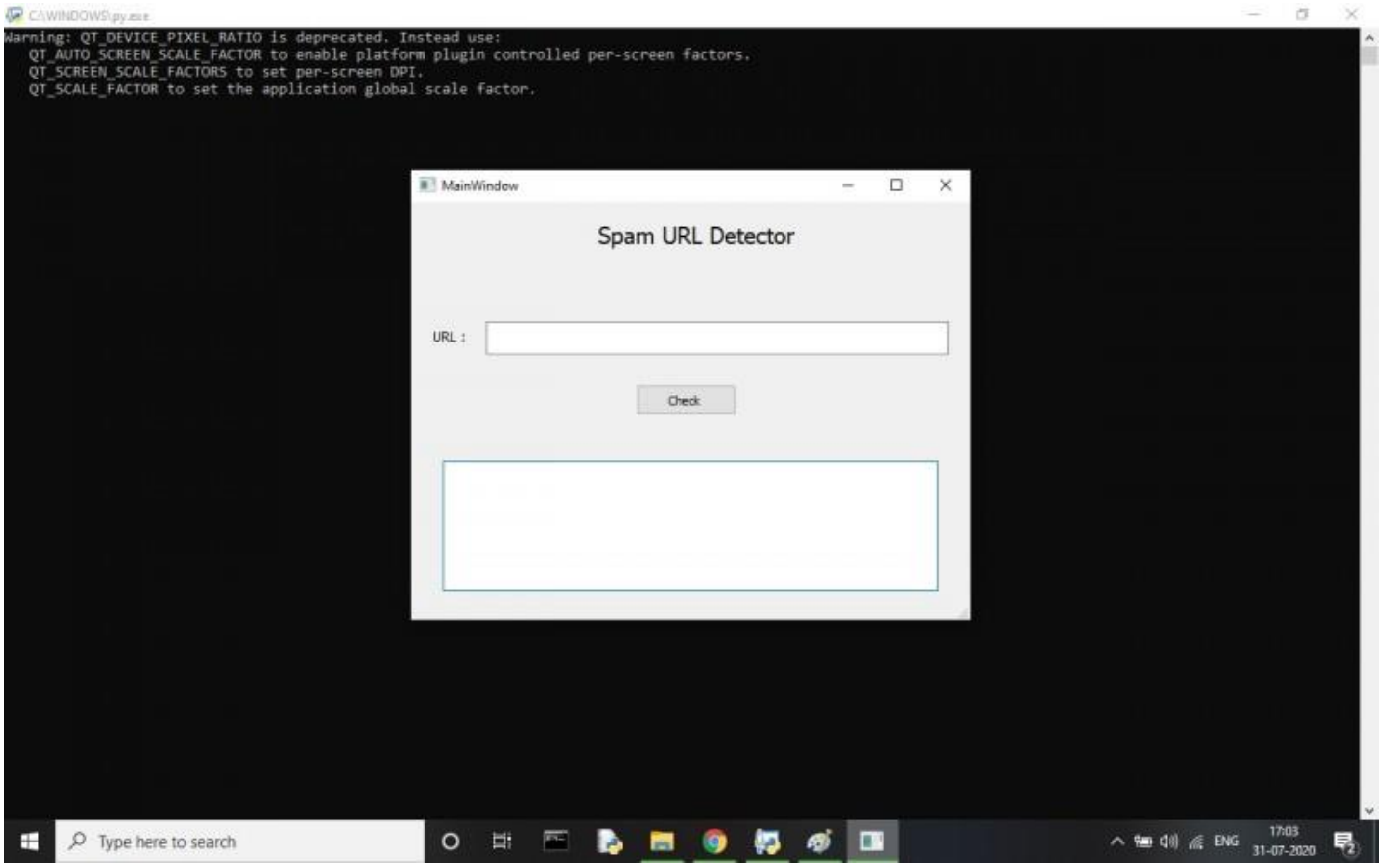
```
import pandas as pd
import numpy as np

import rf_model


class feature_extractor:

    def __init__(self,url:str):
        self.input_url = url

    def long_url(self,l):
        """This function is defined in order to differntiate website
based on the length of the URL"""
        l= str(l)
        if len(l) < 54:
            return 0
        elif len(l) >= 54 and len(l) <= 75:
            return 2
        return 1

    def have_at_symbol(self,l):
        """This function is used to check whether the URL contains @
symbol or not"""
        if "@" in str(l):
            return 1
        return 0

    def redirection(self,l):
        """If the url has symbol(//) after protocol then such URL is
to be classified as phishing """
        if "//" in str(l):
            return 1
        return 0

    def prefix_suffix_seperation(self,l):
        """seprate prefix and suffix"""
        if '-' in str(l):
            return 1
        return 0
```

```
 def sub_domains(self,l):
        """check the subdomains"""
        l= str(l)
        if l.count('.') < 3:
            return 0
        elif l.count('.') == 3:
            return 2
        return 1


    def extract(self):
        print("in script 2")
        input_data = [{"URL":self.input_url}]
        print('input taken')
        temp_df = pd.DataFrame(input_data)
        print("dataframe created")
        #expand argument in the split method will give you a new
column
        seperation_of_protocol =
temp_df['URL'].str.split("://",expand = True)
        print("step 1 done")
        #split(seperator,no of splits according to
seperator(delimiter),expand)
        seperation_domain_name =
seperation_of_protocol[1].str.split("/",1,expand = True)
        print("step 2 done")
        #renaming columns of data frame
        seperation_domain_name.columns=["domain_name","address"]
        print("step 3 done")
        #Concatenation of data frames
        splitted_data =
pd.concat([seperation_of_protocol[0],seperation_domain_name],axis=1)
        print("step 4 done")

        splitted_data.columns = ['protocol','domain_name','address']
        print("step 5 done")

        #splitted_data['is_phished'] = pd.Series(temp_df['Target'],
index=splitted_data.index)
        #print("step 6 done")
 """feature extraction starts here"""
        #Applying the above defined function in order to divide the
websites into 3 categories
        splitted_data['long_url'] =
temp_df['URL'].apply(self.long_url)
        print("feature extra 1")
        splitted_data['having_@_symbol'] =
temp_df['URL'].apply(self.have_at_symbol)
        print("feature extra 2")
        splitted_data['redirection_//_symbol'] =
seperation_of_protocol[1].apply(self.redirection)
        print("feature extra 3")
        splitted_data['prefix_suffix_seperation'] =
seperation_domain_name['domain_name'].apply(self.prefix_suffix_sepera
tion)
        print("feature extra 4")
        splitted_data['sub_domains'] =
splitted_data['domain_name'].apply(self.sub_domains)
        print("feature extra 5")
        #splitted_data.to_csv(r'dataset3.csv',header= True)


        return rf_model.predictor(splitted_data)
```

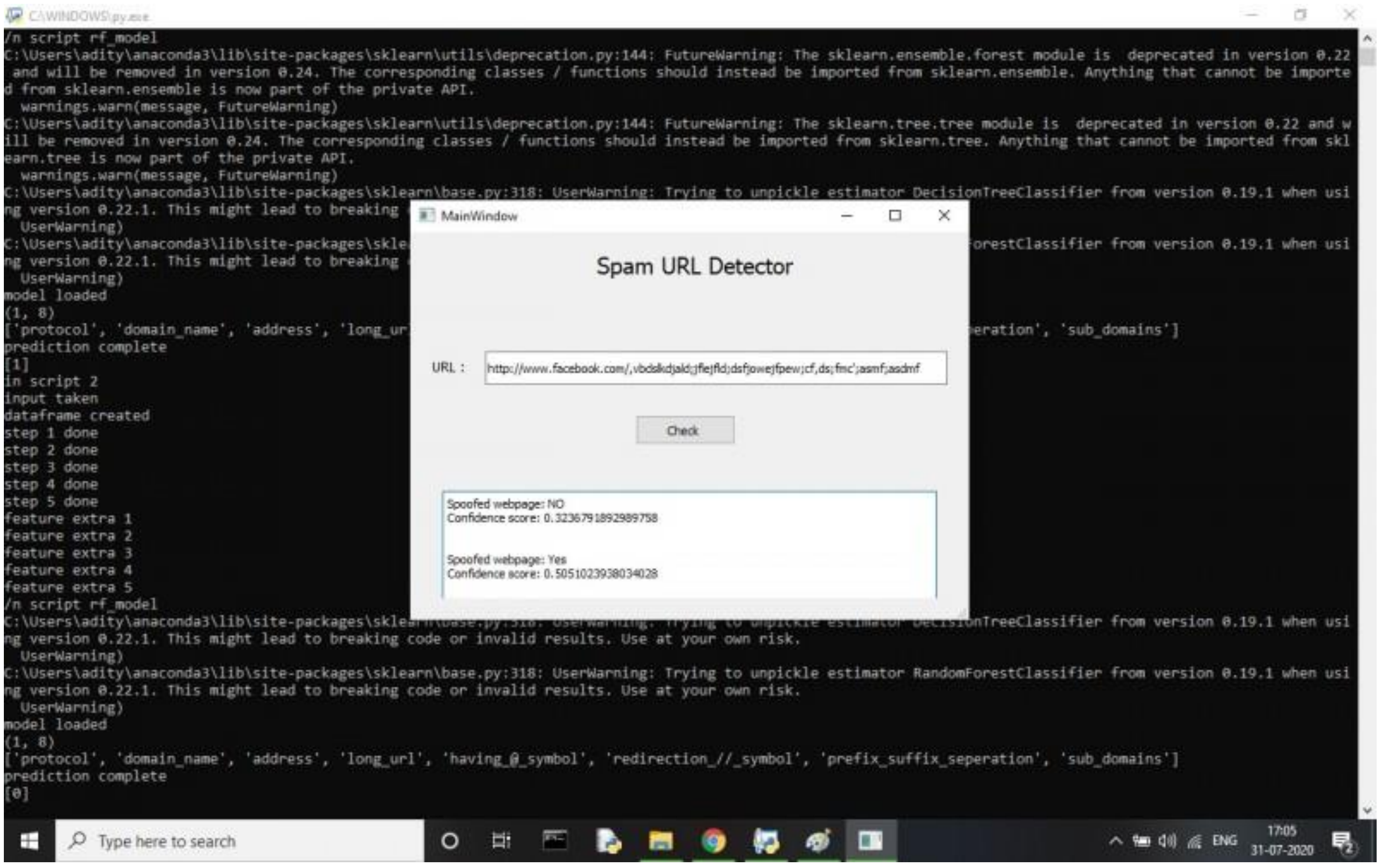
**rf_model.py: -**

```
import pickle
import numpy,sklearn,pandas

"""# save the model to disk
filename = 'finalized_model.sav'
pickle.dump(clf, open(filename, 'wb'))
"""

def predictor(splitted_data):
    print("/n script rf_model")
    # load the model from disk
    filename = 'finalized_model.sav'
    loaded_model = pickle.load(open(filename, 'rb'))
    print("model loaded")
    print(splitted_data.shape)
    print(list(splitted_data))
    x = splitted_data.columns[3:9]
    preds = loaded_model.predict(splitted_data[x])
    print("prediction complete")
    print(preds)
    if preds == 0:
        str1 = "Spoofed webpage: Yes"
    else: str1 = "Spoofed webpage: NO"

    score = loaded_model.predict_proba(splitted_data[x])
    str2 = "Confidence score: "+ str(score[0][1])

    return str1,str2
```

## 5.2 <u>SNAPSHOTS</u>



```
Warning: QT_DEVICE_PIXEL_RATIO is deprecated. Instead use:
   QT_AUTO_SCREEN_SCALE_FACTOR to enable platform plugin controlled per-screen factors.
   QT_SCREEN_SCALE_FACTORS to set per-screen DPI.
   QT_SCALE_FACTOR to set the application global scale factor.
```



C:\WINDOWS\py.exe

```
/n script rf_model
C:\Users\adity\anaconda3\lib\site-packages\sklearn\utils\deprecation.py:144: FutureWarning: The sklearn.ensemble.forest module is  deprecated in version 0.22
 and will be removed in version 0.24. The corresponding classes / functions should instead be imported from sklearn.ensemble. Anything that cannot be importe
d from sklearn.ensemble is now part of the private API.
  warnings.warn(message, FutureWarning)
C:\Users\adity\anaconda3\lib\site-packages\sklearn\utils\deprecation.py:144: FutureWarning: The sklearn.tree.tree module is  deprecated in version 0.22 and w
ill be removed in version 0.24. The corresponding classes / functions should instead be imported from sklearn.tree. Anything that cannot be imported from skl
earn.tree is now part of the private API.
  warnings.warn(message, FutureWarning)
C:\Users\adity\anaconda3\lib\site-packages\sklearn\base.py:318: UserWarning: Trying to unpickle estimator DecisionTreeClassifier from version 0.19.1 when usi
ng version 0.22.1. This might lead to breaking
  UserWarning)
C:\Users\adity\anaconda3\lib\site-packages\skle                                                  orestClassifier from version 0.19.1 when usi
ng version 0.22.1. This might lead to breaking
  UserWarning)
model loaded
(1, 8)
['protocol', 'domain_name', 'address', 'long_ur                                                  eration', 'sub_domains']
prediction complete
[1]
in script 2
input taken
dataframe created
step 1 done
step 2 done
step 3 done
step 4 done
step 5 done
feature extra 1
feature extra 2
feature extra 3
feature extra 4
feature extra 5
/n script rf_model
C:\Users\adity\anaconda3\lib\site-packages\skle                                                  onTreeClassifier from version 0.19.1 when usi
ng version 0.22.1. This might lead to breaking code or invalid results. Use at your own risk.
  UserWarning)
C:\Users\adity\anaconda3\lib\site-packages\sklearn\base.py:318: UserWarning: Trying to unpickle estimator RandomForestClassifier from version 0.19.1 when usi
ng version 0.22.1. This might lead to breaking code or invalid results. Use at your own risk.
  UserWarning)
model loaded
(1, 8)
['protocol', 'domain_name', 'address', 'long_url', 'having_@_symbol', 'redirection_//_symbol', 'prefix_suffix_seperation', 'sub_domains']
prediction complete
[0]
```

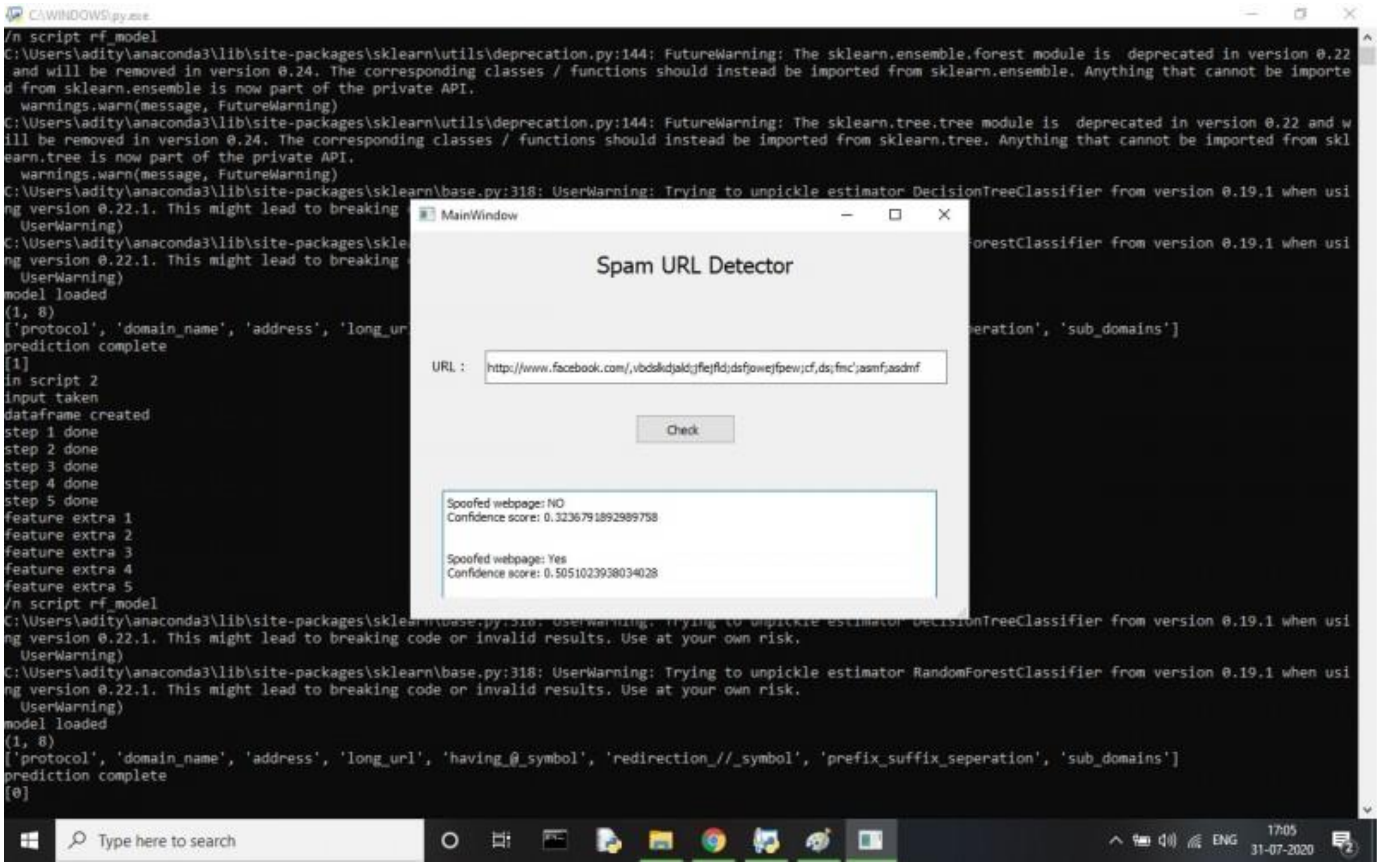
MainWindow

Spam URL Detector

URL : http://www.facebook.com/,vbdslkdjald;jflejfld;dsfjowejfpew;cf,ds;fmc';asmf;asdmf

Check

```
Spoofed webpage: NO
Confidence score: 0.3236791892989758


Spoofed webpage: Yes
Confidence score: 0.5051023938034028
```



C:\WINDOWS\py.exe

```
/n script rf_model
C:\Users\adity\anaconda3\lib\site-packages\sklearn\utils\deprecation.py:144: FutureWarning: The sklearn.ensemble.forest module is  deprecated in version 0.22
 and will be removed in version 0.24. The corresponding classes / functions should instead be imported from sklearn.ensemble. Anything that cannot be importe
d from sklearn.ensemble is now part of the private API.
  warnings.warn(message, FutureWarning)
C:\Users\adity\anaconda3\lib\site-packages\sklearn\utils\deprecation.py:144: FutureWarning: The sklearn.tree.tree module is  deprecated in version 0.22 and w
ill be removed in version 0.24. The corresponding classes / functions should instead be imported from sklearn.tree. Anything that cannot be imported from skl
earn.tree is now part of the private API.
  warnings.warn(message, FutureWarning)
C:\Users\adity\anaconda3\lib\site-packages\sklearn\base.py:318: UserWarning: Trying to unpickle estimator DecisionTreeClassifier from version 0.19.1 when usi
ng version 0.22.1. This might lead to breaking
  UserWarning)
C:\Users\adity\anaconda3\lib\site-packages\skle                                                  orestClassifier from version 0.19.1 when usi
ng version 0.22.1. This might lead to breaking
  UserWarning)
model loaded
(1, 8)
['protocol', 'domain_name', 'address', 'long_ur                                                  eration', 'sub_domains']
prediction complete
[1]
in script 2
input taken
dataframe created
step 1 done
step 2 done
step 3 done
step 4 done
step 5 done
feature extra 1
feature extra 2
feature extra 3
feature extra 4
feature extra 5
/n script rf_model
C:\Users\adity\anaconda3\lib\site-packages\skle                                                  onTreeClassifier from version 0.19.1 when usi
ng version 0.22.1. This might lead to breaking code or invalid results. Use at your own risk.
  UserWarning)
C:\Users\adity\anaconda3\lib\site-packages\sklearn\base.py:318: UserWarning: Trying to unpickle estimator RandomForestClassifier from version 0.19.1 when usi
ng version 0.22.1. This might lead to breaking code or invalid results. Use at your own risk.
  UserWarning)
model loaded
(1, 8)
['protocol', 'domain_name', 'address', 'long_url', 'having_@_symbol', 'redirection_//_symbol', 'prefix_suffix_seperation', 'sub_domains']
prediction complete
[0]
```

MainWindow

Spam URL Detector

URL : http://www.facebook.com/,vbdslkdjald;jflejfld;dsfjowejfpew;cf,ds;fmc';asmf;asdmf

Check

```
Spoofed webpage: NO
Confidence score: 0.3236791892989758


Spoofed webpage: Yes
Confidence score: 0.5051023938034028
```

# CHAPTER 6

# TESTING

## 6.1 TESTING

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

### TYPES OF TESTS

#### UNIT TESTING

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

#### INTEGRATION TESTING

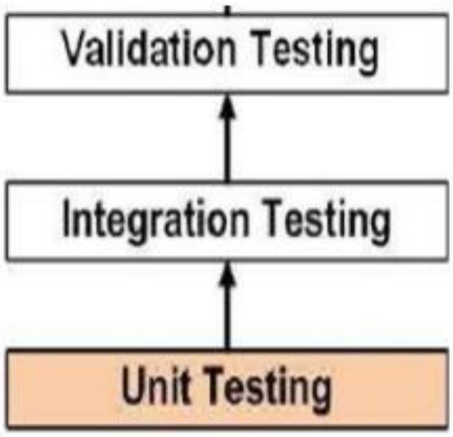
Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfaction, as shown by successfully unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

## VALIDATION TESTING

An engineering validation test (EVT) is performed on first engineering prototypes, to ensure that the basic unit performs to design goals and specifications. It is important in identifying design problems, and solving them as early in the design cycle as possible, is the key to keeping projects on time and within budget. Too often, product design and performance problems are not detected until late in the product development cycle — when the product is ready to be shipped. The old adage holds true: It costs a penny to make a change in engineering, a dime in production and a dollar after a product is in the field.

Verification is a Quality control process that is used to evaluate whether or not a product, service, or system complies with regulations, specifications, or conditions imposed at the start of a development phase. Verification can be in development, scale-up, or production. This is often an internal process. Validation is a Quality assurance process of establishing evidence that provides a high degree of assurance that a product, service, or system accomplishes its intended requirements. This often involves acceptance of fitness for purpose with end users and other product stakeholders.

**The testing process overview is as follows:**



## SYSTEM TESTING

System testing of software or hardware is testing conducted on a complete, integrated system to evaluate the system's compliance with its specified requirements. System testing falls within the scope of black box testing, and as such, should require no knowledge of the inner design of the code or logic.

As a rule, system testing takes, as its input, all of the "integrated" software components that have successfully passed integration testing and also the software system itself integrated with any applicable hardware system(s). System testing is a more limited type of testing; it seeks to detect defects both within the "inter-assemblages" and also within the system as a whole.

System testing is performed on the entire system in the context of a Functional Requirement Specification(s) (FRS) and/or a System Requirement Specification (SRS). System testing tests not only the design, but also the behavior and even the believed expectations of the customer. It is also intended to test up to and beyond the bounds defined in the software/hardware requirements specification(s).\

### Black-box testing

The black-box approach is a testing method in which test data are derived from the specified functional requirements without regard to the final program structure. It is also termed data-driven, input/output driven or requirements-based testing. A testing method emphasized on executing the functions and examination of their input and output data.

### White-box testing

Contrary to black-box testing, software is viewed as a white-box, or glass-box in white-box testing, as the structure and flow of the software under test are visible to the tester. This testing is based on knowledge of the internal logic of an application’s code. Testing plans are made according to the details of the software implementation, such as programming language, logic, and styles. Test cases are derived from the program structure. White-box testing is also called glass-box testing, logic-driven testing or design-based testing.

### End-to-end testing

Similar to system testing, involves testing of a complete application environment in a situation that mimics real-world use, such as interacting with a database, using network communications, or interacting with other hardware, applications, or systems if appropriate.

### Usability testing

User-friendliness check. Application flow is tested, Can new user understand the application easily, Proper help documented whenever user stuck at any point. Basically system navigation is checked in this testing.

### Install/uninstall testing

Tested for full, partial, or upgrade install/uninstall processes on different operating systems under different hardware, software environment.

### Recovery testing

Testing how well a system recovers from crashes, hardware failures, or other catastrophic problems.

### Security testing

Can system be penetrated by any hacking way. Testing how well the system protects against unauthorized internal or external access. Checked if system, database is safe from external attacks.

### Compatibility testing

Testing how well software performs in a particular hardware/software/operating system/network environment and different combination s of above.

### Comparison testing

Comparison of product strengths and weaknesses with previous versions or other similar products.

### Alpha testing

In house virtual user environment can be created for this type of testing. Testing is done at the end of development. Still minor design changes may be made as a result of such testing.

### Beta testing

Testing typically done by end-users or others. Final testing before releasing application for commercial purpose.

# CHAPTER 7

## BENEFITS

## 7.1 BENEFITS

- This will help to reduce the numbers of frauds being done by the hackers.
- People will be aware of the link without visiting such links.
- It will reduce the cybercrime happening around us.
- Easy to use and to understand the work (System).
- It will help the people to make them aware about such crimes.

# CHAPTER 8

## LIMITATIONS

## 8.1 <u>LIMITATIONS</u>

- Software is limited to Desktop only.
- System requires python interpreter/Jupyter installed on the system.
- Can Work on Accuracy to make it more precise.
- Time Consuming.
- GUI is in English only.

# CHAPTER 9

# FUTURE ENHANCEMENT

## 9.1 FUTURE ENHANCEMENT

- More effective feature extraction and representing learning like deep learning approaches.
- More effective machine learning algorithm for training the predictive models particularly with concept themes like more effective online learning.
- Other emerging challenges like domain adaption when applying a model to a new domain.
- A smart design of closed loop system of gaining labelled data and user feedback like integrating on online active learning approach in a real system.

# CHAPTER 10

# CONCLUSION

## 10.1 CONCLUSION

In this we showed a broad and organized study on malicious URL detection using machine learning techniques. We also presented an efficient design of malicious URL detection from machine learning perspective and then later we analyze the existing system for malicious URL detection particularly in the form of developing new feature representation and designing the new learning algorithms for determining malicious URL detection task. We also identify the requirements and challenges for developing malicious URL detection as a service of real-world cybersecurity applications.

# CHAPTER 11

## REFERENCES

## 11.1 REFERENCES

- Alshboul, Y., Nepali, R. K., & Wang, Y. (2015). Detecting malicious short URLs on Twitter. Twenty-first Americas Conference on Information Systems, 1-7.

- GOOGLE. Google Safe Browsing - Blacklist service provided by Google. Available also from: https://safebrowsing.google.com/.

- https://www. phishtank.com, 2016.

- http://searchengineland.Com/google-search-press-129925, 2012

- https://ritcsec.wordpress.com/2017/12/07/detecting-malicious-urls.

- D. Sahoo, C. Liu, and S. Hoi, “Malicious URL Detection using Machine Learning: A Survey”, CoRR, abs/1701.07179, pp. 1-17, 2017.

- https://arxiv.org/pdf/1701.07179

Thank You