|  |  |
| --- | --- |
| **CONTENTS** |  |
| **TOPIC** | **PAGENO** |
| **ABSTRACT** | **I** |
| **LIST OF FIGURES** | **II** |
| **LIST OF TABLES** | **III** |
| **1. INTRODUCTION** | **1** |
| 1.1 Introduction and Objectives | 1 |
| 1.2 Project Objectives | 2 |
| 1.3 Purpose of the Project | 2 |
| 1.4 Existing system & Disadvantages | 2 |
| 1.5 Proposed system with Features | 3 |
| 1.6 Input and output Design | 3 |
| **2. LITERATURE SURVEY** | **5** |
| **3. SOFTWARE REQUIREMENT ANALYSIS** | **7** |
| 3.1 Modules and their Functionalities | 7 |
| 3.2 Functional Requirements | 7 |
| 3.3 Non-Functional Requirements | 8 |
| 3.4 Feasibility Study | 8 |
| 1. **SYSTEM SPECIFICATIONS**    1. Software requirements | 10 |
| 4.2 Hardware requirements | 10 |

1. [SOFTWARE DESIGN 11](#_TOC_250009)
   1. [System Architecture 11](#_TOC_250008)
   2. Dataflow Diagrams 13
   3. [UML Diagrams 14](#_TOC_250007)
2. CODING AND IMPLEMENTATION 19
   1. Source code 19
   2. [Implementation 47](#_TOC_250006)
3. [SYSTEM TESTING 48](#_TOC_250005)
   1. Types of System Testing 48
   2. [Testing Strategies 51](#_TOC_250004)
   3. [Sample Test cases 52](#_TOC_250003)
4. [OUTPUT SCREENS 53](#_TOC_250002)
5. [CONCLUSION 58](#_TOC_250001)
6. FUTURE ENHANCEMENTS 59
7. [REFERENCES 60](#_TOC_250000)

# ABSTRACT

Malicious social bots generate fake tweets and automate their social relationships either by pretending like a follower or by creating multiple fake accounts with malicious activities. Moreover, malicious social bots post shortened malicious URLs in the tweet in order to redirect the requests of online social networking participants to some malicious servers. Hence, distinguishing malicious social bots from legitimate users is one of the most important tasks in the Twitter network. To detect malicious social bots, extracting URL-based features (such as URL redirection, frequency of shared URLs, and spam content in URL) consumes less amount of time in comparison with social graph-based features (which rely on the social interactions of users).

Furthermore, malicious social bots cannot easily manipulate URL redirection chains. In this article, a learning automata-based malicious social bot detection (LA-MSBD) algorithm is proposed by integrating a trust computation model with URL-based features for identifying trustworthy participants (users) in the Twitter network. The proposed trust computation model contains two parameters, namely, direct trust and indirect trust. Moreover, the direct trust is derived from Bayes’ theorem, and the indirect trust is derived from the Dempster–Shafer Theory (DST) to determine the trustworthiness of each participant accurately. Experimentation has been performed on two Twitter data sets, and the results illustrate that the proposed algorithm achieves improvement in precision, recall, F-measure, and accuracy compared with existing approaches for MSBD.

**Keywords** : Malicious bots, Phishing, Twitter network, URL Features, Spam detection.

# LIST OF FIGURES

|  |  |  |  |
| --- | --- | --- | --- |
| **S.NO** | **FIGURE NO** | **DESCRIPTION** | **PAGENO** |
| 1 | 5.1 | System Architecture | 11 |
| 2 | 5.2.1 | Dataflow diagram | 13 |
| 3 | 5.3.1 | Sequence diagram | 15 |
| 4 | 5.3.2 | Usecase diagram | 16 |
| 5 | 5.3.3 | Activity diagram | 17 |
| 6 | 5.3.4 | Class Digram | 18 |
| 7 | 8.1 | Home Page | 53 |
| 8 | 8.2 | Inserting an URL | 54 |
| 9 | 8.3 | Malicious URL detected | 55 |
| 10 | 8.4 | Uploading an URL | 56 |
| 11 | 8.5 | Phishing Detected | 51 |

**LIST OF TABLES**

|  |  |  |  |
| --- | --- | --- | --- |
| **S.NO** | **TABLE NO** | **DESCRIPTION** | **PAGENO** |
| 1 | 7.3.1 | Sample TestCases | 52 |

# 1. INTRODUCTION

## Introduction and Objectives

Malicious social bots generate fake tweets and automate their social relationships either by pretending like a follower or by creating multiple fake accounts with malicious activities. Social media has played a more important role in our daily life. With billions of users producing and consuming information every day, it is a natural extension that people turn to this medium to read and disseminate news.

Social media bots are programs that vary in size depending on their function, capability, and design and can be used on social media platforms to do various useful and malicious tasks while stimulating human behaviour. Some social media bots provide useful services, such as weather updates and sports scores. These good social media bots are clearly identified as such and the people who interact with them know that they are bots.

However a large number of social media bots are malicious bots disguised as human users. These bots cause users to lose trust that social media platforms can deliver news honestly, as they become suspicious that the stories they see at the top of their feeds were “pushed” there by manipulative bots. With so many people turning to social media, malicious users like bots have begun to sway the conversations in whatever direction their creators want.

These malicious bots have been used for malicious tasks such as spreading false information about political candidates , inflating the perceived popularity of celebrities, deliberately pushing down the messages of protestors and activists, illicitly advertising by spamming the social web with links to commercial websites and influencing financial markets in an attempt to manipulate the direction of stock prices. Furthermore, these bots can change the results of common analyses performed on social media.

Some of the common attack methods of social media bots are: sleeper bots-they remain dormant for long periods of time, then wake up to launch their attack of thousands of posts in a short period of time(perhaps as a spam attack),and then return to a dormant state, trend jacking - use of top trending topics to focus on an intended audience for targeting purposes, watering hole attack-attacker guesses or observes which websites an organization often uses and infects one or more of them with malware, hashtag hijacking-use of hashtags to focus an attack(e.g. spam

,malicious links)on a specific audience using the same hashtag and click farming or like farming-inflate fame or popularity on a website through liking or reposting of content via click farms.

Bot detection is an important task in social media. Twitter, a popular social media platform, is plagued by automated accounts. Some studies estimated that around 15% of the accounts on Twitter Operates Automatically or Semi-automatically. One reason which might have stimulated the rise of the number of bots is the characteristics of Twitter. In this article, the malicious behavior of participants is analyzed by considering features extracted from the posted URLs (in the tweets), such as URL redirection, frequency of shared URLs, and spam content in URL, to distinguish between legitimate and malicious tweets.

To protect against the malicious social bot attacks, our proposed LA-based malicious social bot detection (LA-MSBD) algorithm integrates a trust computational model with a set of URL-based features for the detection of malicious social bots. However, our work is different from other existing works in the sense that we focus on detecting malicious social bots based on the LA model with the trust computational model. The LA has also been successfully applied in various areas, such as Internet of Things (IOT), cloud computing, social networks, wireless networks, and image processing.

## Project Objectives

By the end of this project you will understand

1. To predict the malicious URL in the twitter network .
2. To analyze the URL content in the given dataset.
3. It blocks the person who had shared the malicious URL.

## Purpose of the Project

Nowadays the persons are using the many social media platforms like twitter, instagram, facebook etc. As the usage of these platforms are gradually increasing many people are using it in a wrong way and misguide the people who are using these platforms through the wrong URL’s. The main motive of our project is to find the people who are sending these malicious URL’s and block the users.

## Existing Method

Several approaches have been proposed to detect spam in the Twitter network. These approaches are based on tweet-content features, social relationship features, and user profile

features. However, the malicious social bots can manipulate profile features, such as hashtag ratio, follower ratio, URL ratio, and the number of retweets. Most of the existing approaches are

based on conditional learning statements, where the model is trained with the model in order to detect malicious bots in OSNs. However, these approaches rely on statistical features instead of analyzing the social behavior of users.

### Disadvantages

Moreover, these approaches are not highly robust in detecting the temporal data patterns with noisy data

## Proposed Method

A learning automata-based malicious social bot detection (LA-MSBD) model is proposed by integrating a trust computation model with URL-based features for identifying trustworthy participants (users) in the Twitter network. In this we will be giving 13 parameters as input to the SVM algorithm which process these inputs and return a single digit either 0 or 1.The results illustrate that the proposed algorithm achieves improvement in precision, recall, F- measure, and accuracy compared with existing approaches for MSBD.

### Advantages

With the help of automata-based malicious social bot detection (LA-MSBD) we found the url’s which are malicious easily.

## Input and Output Design

### Input Design

The input design is the link between the information system and the user. It comprises the developing specification and procedures for data preparation and those steps are necessary to put transaction data in to a usable form for processing can be achieved by inspecting the computer to read data from a written or printed document or it can occur by having people keying the data directly into the system. The design of input focuses on controlling the amount of input required, controlling the errors, avoiding delay, avoiding extra steps and keeping the process simple. The input is designed in such a way so that it provides security and ease of use with retaining the privacy. Input Design considered the following things:

1. What data should be given as input?
2. How the data should be arranged or coded?
3. The dialog to guide the operating personnel in providing input.
4. Methods for preparing input validations and steps to follow when error occur.

### Objectives

1. Input Design is the process of converting a user-oriented description of the input into a computer-based system.
2. This design is important to avoid errors in the data input process and show the correct direction to the management for getting correct information from the computerized system.
3. It is achieved by creating user-friendly screens for the data entry to handle large volume of data. The goal of designing input is to make data entry easier and to be free from errors. The data entry screen is designed in such a way that all the data manipulates can be performed. It also provides record viewing facilities.
4. When the data is entered it will check for its validity. Data can be entered with the help of screens. Appropriate messages are provided as when needed so that the user will not be in maize of instant. Thus, the objective of input design is to create an input layout that is easy to follow.

## Output Design

A quality output is one, which meets the requirements of the end user and presents the information clearly. In any system results of processing are communicated to the users and to other system through outputs. In output design it is determined how the information is to be displaced for immediate need and also the hard copy output. It is the most important and direct source information to the user. Efficient and intelligent output design improves the system’s relationship to help user decision-making.

Designing computer output should proceed in an organized, well thought out manner; the right output must be developed while ensuring that each output element is designed so that people will find the system can use easily and effectively. When analysis design computer output, they should Identify the specific output that is needed to meet the requirements.

Select methods for presenting information. Create document, report, or other formats that contain information produced by the system. The output form of an information system should accomplish one or more of the following objectives.

* Convey information about past activities, current status or projections of the Future.
* Trigger an action.
* Confirm an action.
* Signal important events, opportunities, problems, or warnings.

# LITERATURE SURVEY

G. Lingam, R. R. Rout, and D. V. L. N. Somayajulu [1].Adaptive deep Q-learning model for detecting social bots and influential users in online social networks. In an online social network a botmaster establishes a social relationship among legitimate participants to reduce the probability of social bot detection. Social bots generate fake tweets and spread malicious information by manipulating the public opinion. Therefore, the detection of social bots in an online social network is an important task.

A.K. Jain and B. B. Gupta [2]. A machine learning based approach for phishing detection using hyperlinks information. The proposed approach incorporates various new outstanding hyperlink specific features to detect phishing attack. The proposed approach has divided the hyperlink specific features into 12 different categories and used these features to train the machine learning algorithms.

P. Shi, Z. Zhang, and K.-K.-R. Choo [3]. Detecting malicious social bots based on clickstream sequences. With the significant increase in the volume, velocity and variety of user data (e.g., user-generated data) in online social networks, there have been attempted to design new ways of collecting and analyzing such big data.

S. Madisetty and M. S. Desarkar [4]. A neural network-based ensemble approach for spam detection in Twitter. As the social networking sites get more popular, spammers target these sites to spread spam posts. Twitter is one of the most popular online social networking sites where users communicate and interact on various topics.

D. R. Patil and J. B. Patil [5]. Malicious URLs detection using decision tree classifiers and majority voting technique. Researchers all over the world have provided significant and effective solutions to detect malicious URLs. Still due to the ever changing nature of cyber-attacks, there are many open issues.

H. Gupta, M. S. Jamal, S. Madisetty, and M. S. Desarkar [6]. A framework for real-time spam detection in Twitter. With the increased popularity of online social networks, spammers find these platforms easily accessible to trap users in malicious activities by posting spam messages. In this work, we have taken Twitter platform and performed spam tweets detection.

G. Lingam, R. R. Rout, and D. V. L. N. Somayajulu [7]. Detection of social botnet using a trust model based on spam content in Twitter network. A social botnet is a collection of social bots in an online social network. The social bots are capable of performing malicious activities, such as spreading malware, phishing online social websites and posting spam content.

M. Al-Janabi, E. D. Quincey, and P. Andras [8]. Using supervised machine learning algorithms to detect suspicious URLs in online social networks. The increasing volume of malicious content in social networks requires automated methods to detect and eliminate such content. This paper describes a supervised machine learning classification model that has been built to detect the distribution of malicious content in online social networks

T. Wu, S. Liu, J. Zhang, and Y. Xiang [9]. Twitter spam detection based on deep learning. Twitter spam has long been a critical but difficult problem to be addressed. So far, researchers have developed a series of machine learning-based methods and blacklisting techniques to detect spamming activities on Twitter.

H. B. Kazemian and S. Ahmed [10]. Comparisons of machine learning techniques for detecting malicious webpages. The conventional method of detecting malicious webpages is going through the black list and checking whether the webpages are listed. Black list is a list of webpages which are classified as malicious from a user’s point of view.

C.-M. Chen, D. J. Guan, and Q.-K. Su [11].Feature set identification for detecting suspicious URLs using Bayesian classification in social networks. Social network services (SNSs) are increasing popular. Communicating with friends forms a social network that can be used to promptly share information with friends.

# SOFTWARE REQUIREMENTS ANALYSIS

## Modules and their Functionalities

### Data Analysis

After collecting datasets from various resources. Dataset must be pre-processing before training to the model. The data pre-processing can be done by various stages, begins with reading the collected dataset the process continues to data cleaning. In data cleaning the datasets contain some redundant attributes, those attributes are not considering for career prediction. So, we have to drop unwanted attributes and datasets containing some missing values we need to drop these missing values or fill with unwanted nan values in order to get better accuracy.

### Data Preprocessing

After collecting datasets from various resources. Dataset must be pre-processing before training to the model. The data pre-processing can be done by various stages, begins with reading the collected dataset the process continues to data cleaning. In data cleaning the datasets contain some redundant attributes, those attributes are not considering for career prediction. So, we have to drop unwanted attributes and datasets containing some missing Values we need to drop these missing values or fill with unwanted nan values in order to get better accuracy.

### Machine Learning Algorithm for Prediction

Machine learning predictive algorithms has highly optimized estimation has to be likely outcome based on trained data. Predictive analytics is the use of data, statistical algorithms and machine learning techniques to identify the likelihood of future outcomes based on historical data. The goal is to go beyond knowing what has happened to providing a best assessment of 9 what will happen in the future. In our system we used supervised machine learning algorithm having subcategories as classification and regression

## Functional Requirements

It provides the users a clear statement of the functions required for the system in order to solve the project information problem it contains a complete set of requirements for the applications. A requirement is condition that the application must meet for the customer to find the application satisfactory. A requirement has the following characteristics

1. It provides a benefit to the origination.
2. It describes the capabilities the application must provide in business terms.
3. It does not describe how the application provides that capability.
4. It is stated in unambiguous words. Its meaning is clear and understandable.
5. It is verifiable.

## Non-Functional Requirements

Career recommendation non-functional requirements, like interests he has, how hours he can work likewise, with today’s IT projects, to determine non-functional requirements, like availability, the approach requires that the designer 1st determine the scope: does the whole solution or only part of it need to be architected to meet minimum levels?

1. This is done through 4 steps:
2. Identify the critical areas of solutions
3. Identify the critical components within each critical area.
4. Determine each component's availability and risk.
5. Model worst-case failure scenarios.

## Feasibility Study

The feasibility of the project is analysed in this phase and business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the company. For feasibility analysis, some understanding of the major requirements for the system is essential.

1. Economical Feasibility
2. Technical Feasibility
3. Social Feasibility

### Economic Feasibility

This study is carried out to check the economic impact that the system will have on the organization. The amount of fund that the company can pour into the research and development

of the system is limited. The expenditures must be justified. Thus, the developed system as well within the budget and this was achieved because most of the technologies used are freely available. Only the customized products had to be purchased.

### Technical Feasibility

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

### Social Feasibility

The aspect of study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system, instead must accept it as a necessity. The level of acceptance by the users solely depends on the methods that are employed to educate the user about the system and to make him familiar with it. His level of confidence must be raised so that he is also able to make some constructive criticism, which is welcomed, as he is the final user of the system.

# SYSTEM SPECIFICATIONS

* 1. **Software Requirements** Operating system : Windows 7. Coding Language : Python.

Front-End : Python console.

## Hardware Requirements

System : Pentium IV 2.4 GHz.

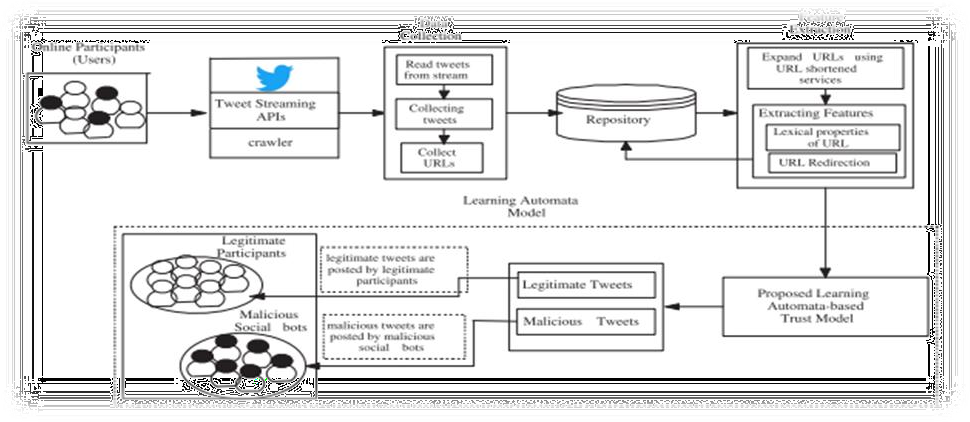
Hard Disk : 40 GB.

Monitor : 14’ Colour Monitor.

Ram : 512 Mb.

# SOFTWARE DESIGN

## System Architecture



**Fig 5.1** System Architecture

The methodology of review consists following steps

1. Data Collection
2. Feature Extraction
3. Classification
4. Testing
5. URL feature prediction

### Data Collection

The dataset was composed of URL’s of twitter data set along with their respective identity such as malicious or non malicious. Based upon these data the given input can be classified as either malicious or non-malicious. The dataset which we use in this project is the twitter data set collected.

### Feature Extraction

The feature extraction technique plays an important role. The features are the main parameter that are involved for classification of URL. Texture extraction is determined as the example of information or course of action of the structure with random interval.

### Classification

In a typical classification system image captured by a camera and then processed. In Supervised classification, most importantly preparing occurred through known gathering of pixels. The numbers of clusters decided by users. When trained pixels are not available, the supervised classification is used that is KNN.

### Testing

In the testing phase the URL’s are being tested.

### URL feature prediction

Finally, we get the URL is either malicious or non malicious.

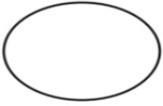
### Data Flow Diagrams

A Data Flow Diagram (DFD) is a traditional visual representation of the information flows within a system. A neat and clear DFD can depict the right amount of the system requirement graphically. It may be used as a communication tool between a system analyst and any person who plays a part in the order that acts as a starting point for redesigning a system. The DFD is also called as a data flow graph or bubble chart.

The Basic Notation used to create a DFD’s are as follows:

1. Dataflow : Data move in a specific direction from an origin to a destination.



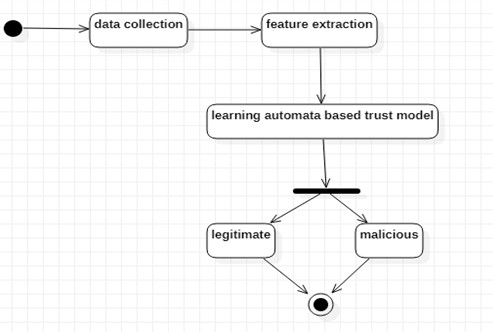
1. Process : People, procedures, or devices that use or produce (Transform) Data. The physical component is not identified.
2. Source : External sources or destination of data, which may be People, programs.



## DataFlow Diagram

The flow of execution of the current project is shown in the data flow diagram. The stages are present in the step wise process so that we can follow that and know about the complete details of the project. A Data Flow Diagram (DFD) is a traditional visual representation of the information flows within a system. A neat and clear DFD can depict the right amount of the system requirement graphically. It can be manual, automated, or a combination of both.

A data flow diagram (DFD) maps out the flow of information for any process or system. It uses defined symbols like rectangles, circles and arrows, plus short text labels, to show data inputs, outputs, storage points and the routes between each destination. A data flow diagram (DFD) is a graphical or visual representation using a standardized set of symbols and notations to describe a business's operations through data movement. They are often elements of a formal methodology such as Structured Systems Analysis and Design Method (SSADM).



**Fig 5.2.1** Dataflow Diagram

## UML Diagrams

UML is a standard language for specifying, visualizing, constructing, and documenting the artifacts of software systems.UML was created by the Object Management Group (OMG) and UML1.0 specification draft was proposed to the OMG in January1997.

There are several types of UML diagrams and each one of them serves a different purpose regardless of whether it is being designed before the implementation or after (as part of documentation). UML has a direct relation with object oriented analysis and design. After some standardization, UML has become an OMG standard.

The two broadest categories that compass all other types are:

1. Behavioral UML diagram
2. Structural UML diagram.

As then it suggests, some UML diagrams try to analyze and depict the structure of a system or process, whereas other describe the behavior of the system, its actors, and its building components.

The different types are broken down as follows:

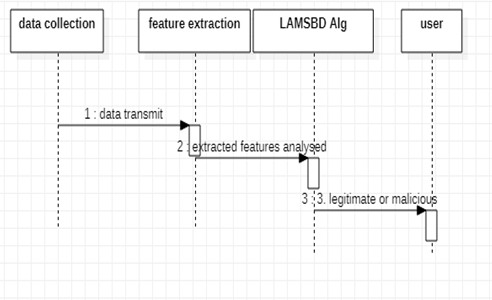
1. Sequence diagram
2. Usecase Diagram
3. Activity diagram
4. Class diagram

### Sequence Diagram

A sequence diagram simply depicts interaction between objects in a sequential order i.e. the order in which these interactions take place. We can also use the terms event diagrams or event scenarios to refer to a sequence diagram. Sequence diagrams describe how and in what order the objects in a system function. These diagrams are widely used by businessmen and software developers to document and understand requirements for and existing systems.

The sequence diagram is a good diagram to use to document a system's requirements and to flush out a system's design. The reason the sequence diagram is so useful is because it shows the interaction logic between the objects in the system in the time order that the interactions take place. In sequence diagrams, combined fragments are logical groupings, represented by a rectangle, which contain the conditional structures that affect the flow of messages. A combined fragment contains interaction operands and is defined by the interaction operator.

A lifeline represents an individual participant in a sequence diagram. A lifeline will usually have a rectangle containing its object name. If its name is "self", that indicates that the lifeline represents the classifier which owns the sequence diagram.

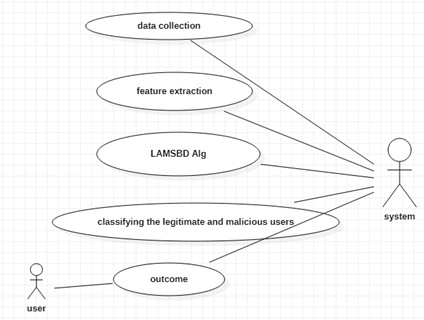


**Fig 5.3.1** Sequence Diagram

### Usecase Diagram

A use case diagram at its simplest is a representation of a user's interaction with the system that shows the relationship between the user and the different use case in which the user is involved. A use case diagram is used to structure of the behavior thing in a model. The use cases are represented by either circles or ellipses.

Use-case diagrams describe the high-level functions and scope of a system. These diagrams also identify the interactions between the system and its actors. The use cases and actors in use-case diagrams describe what the system does and how the actors use it, but not how the system operates internally. A use case is a written description of how users will perform tasks on your website. It outlines, from a user's point of view, a system's behavior as it responds to a request. Each use case is represented as a sequence of simple steps, beginning with a user's goal and ending when that goal is fulfilled.



**Fig 5.3.2** Usecase Diagram

### Usecase:

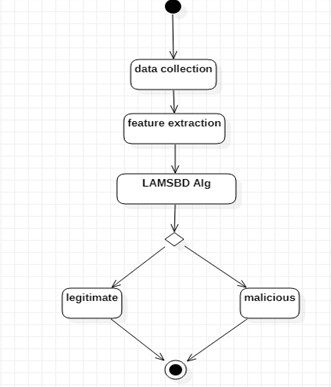
* User need to collect the data.
* The URL’S are stored in the dataset.
* User need to upload the url.
* Few algorithms are used to find the malicious URL’s.

### Activity Diagram

Activity diagram is another important diagram in UML to describe the dynamic aspects of the system. Activity diagram is basically a flowchart to represent the flow from one activity to another activity. This flow can be sequential, branched, or concurrent. Activity diagrams deal with all type of flow control by using different elements such as fork, join, etc.

An activity diagram is a behavioral diagram i.e. it depicts the behavior of a system. An activity diagram portrays the control flow from a start point to a finish point showing the various decision paths that exist while the activity is being executed. Activity diagram is basically a flowchart to represent the flow from one activity to another activity. The activity can be described as an operation of the system. The control flow is drawn from one operation to another. This flow can be sequential, branched, or concurrent.

An activity diagram shows business and software processes as a progression of actions. These actions can be carried out by people, software components or computers. Activity diagrams are used to describe business processes and use cases as well as to document the implementation of system processes.



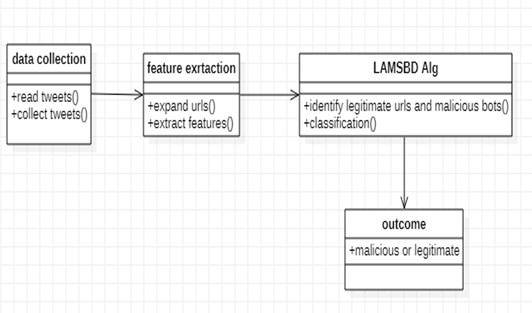
**Fig 5.3.3** Activity Diagram

### Class Diagram

A class diagram is an illustration of the relationships and source code dependencies among classes in the Unified Modeling Language (UML). In this context, a class defines the methods and variables in an object, which is a specific entity in a program or the unit of code representing that entity. Class diagram mainly consists of classes Class diagram describes the attributes and operations of a class and also the constraints imposed on the system. The class diagrams are widely used in the modeling of object oriented systems because they are the only UML diagrams, which can be mapped directly with object-oriented languages.

There are 3 main sections in class diagram

* + - * Upper section: Contains the name of the class. This section is always required, whether you are talking about the classifier or an object.
      * Middle section: Contains the attributes of the class.
      * Bottom section: Includes class operations (methods).



**Fig 5.3.4** Class Diagram

# CODE AND IMPLEMENTATION

#!/usr/bin/python

# -\*- coding: utf-8 -\*-

def warn(\*args, \*\*kwargs): pass

from django.shortcuts import render

from django.http import HttpResponse, JsonResponse from django.db.models import Q

from .models import \*

# Create your views here.

def error\_404\_view(request, exception): return render(request,'404.html')

def getdoc(request):

return render(request,'ppt.html') def index(request):

try:

return render(request, 'index.html') except:

return render(request, '404.html') def getuserfeedbackform(request):

try:

return render(request, 'userfeedbackform.html') except:

return render(request, '404.html')

def saveuserfeedbackform(request):

try:

obj = UserFeedBack()

obj.title = request.GET['usertitle'] obj.description = request.GET['userdescription'] obj.save()

mydict = {'feedback': True}

return render(request, 'userfeedbackform.html', context=mydict) except:

return render(request, '404.html') import warnings

warnings.warn = warn import warnings import joblib

from lxml import html

from json import dump, loads from requests import get import json

from re import sub

from dateutil import parser as dateparser from time import sleep

from django.http import HttpResponse from django.shortcuts import render import os

import pickle

import joblib import whois import datetime def result(request):

"""try:""" #nm=request.GET['url']

text=request.GET['url'] if not text.startswith('http'):

return render(request,"404.html")

if text.startswith('https://malicious-url-detectorv5.herokuapp.com/') : 'org':"The Legions",

'add':"New Delhi",

'city':"New Delhi",

'state':"New Delhi",

'ziip':"201301",

'country':"India",'emails':"[thelegions@gmail.com](mailto:thelegions@gmail.com)", 'org':"NA for google search",

'add':"NA for google search", 'city':"NA for google search", 'state':"NA for google search", 'ziip':"NA for google search",

if text.startswith('https://') or text.startswith('http://'): var13="Not Applicable"

varab="Not Applicable"

var11="Not Applicable" var10="Not Applicable" var5="Not Applicable" var4="Not Applicable" var3="Not Applicable" if len(text)<=9:

return render(request,'errorpage.html') aburl=-1

digits="0123456789" if text[8] in digits:

oneval=-1 else:

oneval=1

if len(text)>170: secval=-1 mal = True

else:

mal = False

#print (name,org,add,city,state,ziip,country,emails,dom) from json.encoder import JSONEncoder

final\_entity = { "predicted\_argument": [int(arg[0])]} # directly called encode method of JSON

#print (JSONEncoder().encode(final\_entity))

obj = Url()

obj.result = te #print (dom,rank)

tags = [name,org,state,add,city,ziip,country,emails,dom,rank] tags = list(filter(lambda x: x!="Not Found",tags)) tags.append(text)

obj.link = text obj.add = add obj.state = state obj.city = city

#obj.ziip = res['zip\_code'] obj.country = country obj.emails = emails obj.dom = dom

obj.org = org obj.rank = rank obj.registrar=registrar obj.save()

#print (add)

if add!=None:

if add and len (add)==1: add=add.replace(",","")

elif len(add)>1: add="".join(add)

#print (add)

name="".join(name) #print (name)

if emails!=None: emails="".join(emails)

if org!=None: org=org.replace(",","")

#print (org) dom="".join(dom) #print (dom)

if registrar: registrar=registrar.replace(",","")

#print (registrar) #print (emails) #print(city) import csv

with open ('static//dataset.csv','a') as res: writer=csv.writer(res)

s="{},{},{},{},{},{},{},{},{},{},{},{},{}\n".format(text,te,(name),

org, add, city, state, ziip,

country,emails,

str(dom),rank,str(registrar)) res.write(s)

return render(request,'result.html',{'result':'Real-time analysis successfull','f2':te,'mal': mal,'text':text,'name':name,

'org':org,

'add':add,

'city':city, 'state':state, 'ziip':ziip,

'country':country,'emails':emails,

else:

return render(request,'404.html') """except:

return render(request,'errorpage.html') """ def api(request):

try:

text=request.GET['query'] import datetime

if text.startswith('https://malicious-url-detectorv5.herokuapp.com/'): import datetime

mydict = { "query" : text,

"malware" : False,

"datetime" : str(datetime.datetime.now())

}

response = JsonResponse(mydict) return response

elif text.startswith('https:/[/www](http://www.youtube.com/%27)).[youtube.com/'):](http://www.youtube.com/%27)) import datetime

mydict = { "query" : text,

"malware" : False,

"datetime" : str(datetime.datetime.now())

}

response = JsonResponse(mydict) return response

elif text.startswith('https:/[/www](http://www.google.com/search?q=%27)).[google.com/search?q='):](http://www.google.com/search?q=%27)) import datetime

mydict = { "query" : text,

"malware" : False,

"datetime" : str(datetime.datetime.now())

}

response = JsonResponse(mydict) return response

if (text.startswith('https:/[/www](http://www.google.com/search?q=%27)%3D%3DFalse)).[google.com/search?q=')==False)](http://www.google.com/search?q=%27)%3D%3DFalse)) : else:

if text.startswith('https://') or text.startswith('http://'):

if len(text)<=9:

return render(request,'errorpage.html') aburl=-1

digits="0123456789" if text[8] in digits:

oneval=-1 else:

oneval=1

if len(text)>170:

except:

print("getaddrerrror DNE") d=0

name="Not found in database" org="Not found in database" add="Not found in database" city="Not found in database" state="Not found in database" ziip="Not found in database" country="Not found in database" emails="Not found in database" dom="Not Found"

if d!=0: try:

if len(res.creation\_date)>1:

a=res['creation\_date'][0] b=datetime.now()

c=b-a d=c.days

except:

a=res['creation\_date'] b=datetime.now() c=b-a

d=c.days """except:

print("getaddrerrror DNE") d=0"""

if d>365: eleventhval=1 aburl=1

elif d<=365: eleventhval=-1 aburl=-1

var11="Domain age working less than a year" if aburl==-1:

twelthval=-1 else:

twelthval=1

import urllib.request, sys, re

import xmltodict, json try:

xml =

urllib.request.urlopen(['h](http://data.alexa.com/data?cli=10&dat=s&url)t[tp://data.alexa.com/data?cli=10&dat=s&url=](http://data.alexa.com/data?cli=10&dat=s&url){}'.format(text)).read() result= xmltodict.parse(xml)

data = json.dumps(result).replace("@","") data\_tojson = json.loads(data)

url = data\_tojson["ALEXA"]["SD"][1]["POPULARITY"]["URL"]

rank= int(data\_tojson["ALEXA"]["SD"][1]["POPULARITY"]["TEXT"])

#print ("rank",rank) if rank<=100000:

thirt=1 else:

thirt=-1 #print (thirt)

except:

thirt=-1

rank="Not Indexed by Alexa" #print (rank)

filename = 'phish\_trainedv3.sav' loaded\_model = joblib.load(filename) #print (arg[0])

import whois

url=text

#print (res) if (d!=0):

name=res.domain\_name #print (res.domain\_name) org=res.org

#print (res.org) add=res.address #print (res.address) city=res.city

#print (res.city) state=res.state #print (res.state) ziip=res.zipcode #print (res.zipcode) country=res.country #print (res.country) emails=res.emails #print (res.emails)

dom=res.domain\_name #print (res.domain\_name)

else:

name="Not found in database"

org="Not found in database" add="Not found in database"

city="Not found in database" state="Not found in database" ziip="Not found in database" country="Not found in database" emails="Not found in database" dom="Not Found"

if dom=="Not Found" and rank==-1 : arg[0]=-1

#phishing if arg[0]==1:

te="Legitimate" else:

te="Malicious" if arg[0] == 1:

mal = True else:

mal = False

if dom=="Not Found" and rank=="Not Indexed by Alexa" : arg[0]=-1

#phishing if arg[0]==1:

te="Legitimate" else:

te="Malicious"

if arg[0] == 1: mal = True

else:

mal = False if arg[0] == 1:

malstatus = False else:

malstatus = True

from json.encoder import JSONEncoder

final\_entity = { "predicted\_argument": [int(arg[0])]} # directly called encode method of JSON

#print (JSONEncoder().encode(final\_entity)) #print (dom,rank)

"""res=whois.whois(url) obj = Url() obj.link=res["name"] print (res["name"]) obj.org=res['org']

print (res['org']) obj.add=res['address'] print (res['address']) obj.city=res['city'] print (res['city'])

obj.state=res['state']

print (res['state']) print (res['zipcode'])

obj.country=res['country'] print (res['country']) obj.emails=res["emails"][0] print (res["emails"][0]) obj.dom=res['domain\_name'] print (res['domain\_name']) obj.rank = rank

obj.save()

"""

'''return render(request, 'result.html',

{'result': 'Real-time analysis successfull', 'f2': te, 'mal': mal,'text':text})'''

import datetime mydict = {

"query" : url, "malware" : malstatus,

"datetime" : str(datetime.datetime.now())

}

response = JsonResponse(mydict) return response

except:

return render(request,'404.html'

def about(request):

#return HttpResponse("about") try:

return render(request, 'about.html') except:

return render(request, 'about.html') def geturlhistory(request):

try:

mydict = {

"urls" : Url.objects.all().order\_by('-created\_at')

}

return render(request,'list.html',context=mydict) except:

return render(request,'404.html') def discuss(request):

try:

mydict = {

"users" : UserFeedBack.objects.all()

}

return render(request,'discuss.html',context=mydict) except:

return render(request,'404.html') def search(request):

query = request.GET['search'] query = str(query).lower()

return render(request,'list.html',context=mydict) except:

return render(request,'404.html') def replyform(request,replyid):

try:

obj = UserFeedBack.objects.get(userid=replyid) mydict = {

"replyid" : obj.userid, "title" : obj.title,

"description" : obj.description

}

return render(request,'reply.html',context=mydict) except:

return render(request,'404.html') def savereply(request):

try:

print("debug start")

replyid = request.GET['replyid'] import sys

def main():

os.environ.setdefault('DJANGO\_SETTINGS\_MODULE', 'project.settings')

from django.core.management import execute\_from\_command\_line except ImportError as exc:

raise ImportError(

"Couldn't import Django. Are you sure it's installed and "

"available on your PYTHONPATH environment variable? Did you " "forget to activate a virtual environment?"

) from exc execute\_from\_command\_line(sys.argv)

if name == ' main ': main()

from django.db import models # Create your models here.

class UserFeedBack(models.Model):

userid = models.AutoField(primary\_key=True) title = models.CharField(max\_length=100) description = models.TextField()

reply = models.TextField()

replied = models.BooleanField(default=False)

created\_at = models.DateTimeField(auto\_now\_add=True) class Url(models.Model):

urlid = models.AutoField(primary\_key=True)

link = models.CharField(max\_length=1000,null=True,default="Not Found")

result = models.CharField(max\_length=100,null=True,default="Not Found") add = models.CharField(max\_length=1000,null=True,default="Not Found")

org = models.CharField(max\_length=100,null=True,default="Not Found") city = models.CharField(max\_length=100,null=True,default="Not Found") state = models.CharField(max\_length=100,null=True,default="Not Found") country = models.CharField(max\_length=100,null=True,default="Not Found") dom = models.CharField(max\_length=100,null=True,default="Not Found") emails = models.CharField(max\_length=100,null=True,default="Not Found") rank = models.IntegerField(null=True,default=0,blank=True)

registrar = models.CharField(max\_length=100,null=True,default="Not Found") #rank = models.CharField(max\_length=100,null=True,default="N A",blank=True) created\_at = models.DateTimeField(auto\_now\_add=True)

from django.urls import path from . import views urlpatterns = [

path('',views.index,name='index'), path('getuserfeedbackform',views.getuserfeedbackform,name="getuserfeedbackform"), ath('saveuserfeedbackform',views.saveuserfeedbackform,name="saveuserfeedbackform"), path('api',views.api,name='api'),

path('search',views.search,name="search"), path('result',views.result,name='result'), path('about',views.about,name='about'), path('geturlhistory',views.geturlhistory,name="geturlhistory"), path('discuss',views.discuss,name="discuss"), path('reply/<int:replyid>',views.replyform,name="reply"),

path('savereply',views.savereply,name="reply"),

path('searchdiscuss',views.searchdiscuss,name="searchdiscuss"), path('getdataset',views.getdataset,name='getdataset'), path('getdocs',views.getdoc,name='namedoc')

import os

import django\_heroku

from decouple import config import dj\_database\_url

BASE\_DIR = os.path.dirname(os.path.dirname(os.path.abspath( file ))) SECRET\_KEY = 'l=oc1ljo5a6j7h%j9s+^fvk)l94-2uyl8^ioj(#-(+=@n9%c-2' DEBUG = True

ALLOWED\_HOSTS = ['\*'] INSTALLED\_APPS = [

'django.contrib.admin', 'django.contrib.auth', 'django.contrib.contenttypes', 'django.contrib.sessions', 'django.contrib.messages', 'django.contrib.staticfiles', 'reset\_migrations',

'myapp'

]

MIDDLEWARE = [

'django.middleware.security.SecurityMiddleware', 'django.contrib.sessions.middleware.SessionMiddleware',

'django.middleware.common.CommonMiddleware', 'corsheaders.middleware.CorsMiddleware', 'django.middleware.common.BrokenLinkEmailsMiddleware', 'django.middleware.common.CommonMiddleware', 'django.middleware.csrf.CsrfViewMiddleware', 'django.contrib.auth.middleware.AuthenticationMiddleware', 'django.contrib.messages.middleware.MessageMiddleware', 'django.middleware.clickjacking.XFrameOptionsMiddleware', 'whitenoise.middleware.WhiteNoiseMiddleware'

]

ROOT\_URLCONF = 'project.urls' TEMPLATES = [

{

'BACKEND': 'django.template.backends.django.DjangoTemplates', 'DIRS': [os.path.join(BASE\_DIR, 'templates')],

'APP\_DIRS': True, 'OPTIONS': {

'context\_processors': [ 'django.template.context\_processors.debug', 'django.template.context\_processors.request', 'django.contrib.auth.context\_processors.auth', 'django.contrib.messages.context\_processors.messages',

],

},

},

]

WSGI\_APPLICATION = 'project.wsgi.application' DATABASES = {

'default': {

'ENGINE': 'django.db.backends.sqlite3', 'NAME': os.path.join(BASE\_DIR, 'db.sqlite3'),

}

}

AUTH\_PASSWORD\_VALIDATORS = [

{

'NAME': 'django.contrib.auth.password\_validation.UserAttributeSimilarityValidator',

},

{

'NAME': 'django.contrib.auth.password\_validation.MinimumLengthValidator',

},

{

'NAME': 'django.contrib.auth.password\_validation.CommonPasswordValidator',

},

{

'NAME': 'django.contrib.auth.password\_validation.NumericPasswordValidator',

},

]

# https://docs.djangoproject.com/en/3.0/topics/i18n/

LANGUAGE\_CODE = 'en-us'

TIME\_ZONE = 'Asia/Calcutta' USE\_I18N = True

USE\_L10N = True USE\_TZ = True

# Static files (CSS, JavaScript, Images)

# https://docs.djangoproject.com/en/3.0/howto/static-files/ STATIC\_URL = '/static/'

STATICFILES\_DIRS = [

os.path.join(BASE\_DIR, 'static'),

]

STATIC\_ROOT = os.path.join(BASE\_DIR, "static") django\_heroku.settings(locals())

CORS\_ORIGIN\_ALLOW\_ALL = True

### index.html

{% load static %}

<!doctype html>

<html lang="en">

<head>

<!-- Required meta tags -->

<meta charset="utf-8">

<meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no"> style="margin:0 auto;display: block;text-align: center;font-size: 6rem;color:red"></i>

<!--First row-->

<div class="row d-flex justify-content-center">

<div class="col-md-12 col-lg-12 mb-4">

<!-- Material outline input -->

<div class="container">

<input type="text" name="url" class="form-control form-control-lg" name="url" placeholder="Enter Suspected Url" required>

</div>

</div>

<div class="col-md-12 col-lg-12 mb-4">

<button type="submit" class="btn btn-danger" style="display: block;margin: 0 auto;">Real Time

Analysis</button>

</div>

</div>

<!--First row-->

</section>

<!-- Section -->

</div>

</form>

</div>

<div class="row">

<div class="col-md-6">

<img src="{%static 'img1.jpg' %}" class="img-fluid">

</div>

<div class="col-md-6">

<img src="{%static '4560.jpg' %}" class="img-fluid">

</div>

</div>

{% endblock %}

<div class="container-fluid">

</div>

<!-- Optional JavaScript -->

<!-- jQuery first, then Popper.js, then Bootstrap JS -->

<!-- JQuery -->

<!-- JQuery -->

<script type="text/javascript"

src="https://cdnjs.cloudflare.com/ajax/libs/jquery/3.4.1/jquery.min.js"></script>

<!-- Bootstrap tooltips -->

<script type="text/javascript"src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.14.4/umd/popper.m in.js"></script>

<!-- Bootstrap core JavaScript -->

<script type="text/javascript" src="https://cdnjs.cloudflare.com/ajax/libs/twitter-

bootstrap/4.3.1/js/bootstrap.min.js"></script>

<!-- MDB core JavaScript -->

<script type="text/javascript" src="https://cdnjs.cloudflare.com/ajax/libs/mdbootstrap/4.10.1/js/mdb.min.js"></script>

<script type="text/javascript">

$(document).ready(function () {

//Preloader preloaderFadeOutTime = 1200; function hidePreloader() {

var preloader1 = $('.spinner-border'); var preloader2 = $('#card');

preloader1.fadeOut(preloaderFadeOutTime); preloader2.fadeOut(preloaderFadeOutTime);

}

hidePreloader();

});

</script>

<footer class="iq-footer2 iq-over-black-80 iq-bg jarallax" style="background-image:

url('https://iqonicthemes.com/themes/qwilo/qwilo/images/bg/22.jpg'); background-position: left center;">

<div class="footer-bottom iq-ptb-20">

<div class="container">

<div class="row">

<div class="col-sm-6">

<div class="iq-copyright iq-mt-10 iq-font-white"> Copyright

<span id="copyright">

<script> document.getElementById('copyright').appendChild(document.createTextNode(new Date()

.getFullYear()))

</script>

</span>

<br> MbedTech

</div>

</div>

</div>

</div>

</div>

</footer>

</body>

</html>

### result.html

{% extends 'index.html' %}

{% block title %}Result Page{% endblock %}

<div class="card-body">

<!-- Text -->

<ul class="list-group">

<li class="list-group-item">Link : {{text}}</li>

<li class="list-group-item">Name : {{name}}</li>

<li class="list-group-item">Organisation : {{org}}</li>

<li class="list-group-item">Add : {{add}}</li>

<li class="list-group-item">City : {{city}}</li>

<li class="list-group-item">State : {{state}}</li>

<li class="list-group-item">Ziip : {{ziip}}</li>

<li class="list-group-item">Country : {{country}}</li>

<li class="list-group-item">Email : {{emails}}</li>

<li class="list-group-item">Dom : {{dom}}</li>

</div>

</div>

<!-- Card -->

</div>

</div>

<script>

Swal.fire(

'Phishing Detected !', 'You are safe :)',

'error'

)

</script>

{% endif %}

</div>

</div>

</div>

</div>

</div>

{% endblock %}

## Implementation

### Python

Python is a general-purpose interpreted, interactive, object-oriented, and high-level programming language. An [interpreted language,](https://en.wikipedia.org/wiki/Interpreted_language) Python has a design philosophy that emphasizes code [readability](https://en.wikipedia.org/wiki/Readability) (notably using [whitespace](https://en.wikipedia.org/wiki/Whitespace_character) indentation to delimit [code blocks](https://en.wikipedia.org/wiki/Code_block) rather than curly brackets or keywords), and a syntax that allows programmers to express concepts in fewer [lines of code](https://en.wikipedia.org/wiki/Source_lines_of_code) than might be used in languages such as [C++](https://en.wikipedia.org/wiki/C%2B%2B) or [Java.](https://en.wikipedia.org/wiki/Java_(programming_language)) It provides constructs that enable clear programming on both small and large scales.

Python interpreters are available for many [operating systems.](https://en.wikipedia.org/wiki/Operating_system) [C Python,](https://en.wikipedia.org/wiki/CPython) the [reference](https://en.wikipedia.org/wiki/Reference_implementation) [implementation](https://en.wikipedia.org/wiki/Reference_implementation) of Python, is [open source](https://en.wikipedia.org/wiki/Open_source) software and has a community-based development model, as do nearly all of its variant implementations. C Python is managed by the non- profit [Python Software Foundation.](https://en.wikipedia.org/wiki/Python_Software_Foundation) Python features a [dynamic type](https://en.wikipedia.org/wiki/Dynamic_type) system and automatic [memory management](https://en.wikipedia.org/wiki/Memory_management).

### Django

Django is a high-level Python Web framework that encourages rapid development and clean, pragmatic design. Built by experienced developers, it takes care of much of the hassle of Web development, so you can focus on writing your app without needing to reinvent the wheel. It’s free and open source.

Django's primary goal is to ease the creation of complex, database-driven websites. Django emphasizes [reusability](https://en.wikipedia.org/wiki/Reusability) and "pluggability" of components, rapid development, and the principle of [don't repeat yourself.](https://en.wikipedia.org/wiki/Don%27t_repeat_yourself) Python is used throughout, even for settings files and data models.

# SYSTEM 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 test. Each test type addresses a specific testing requirement.

## Types of Testing

### 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 other 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.

### Functional Test

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Functional testing is centered on the following items:

Valid Input : identified classes of valid input must be accepted. Invalid Input : identified classes of invalid input must be rejected. Functions : identified functions must be exercised.

Output : identified application outputs must be exercised. Systems/Procedures : interfacing systems or procedures must be invoked.

Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify Business process flows; data fields, predefined processes, and successive processes must be considered for testing. Before functional testing is complete, additional tests are identified and the effective value of current tests is determined.

### System Test

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

### White Box Testing

White Box Testing is a testing in which in which the software tester has knowledge of the inner workings, structure and language of the software, or at least its purpose. It is purpose. It is used to test areas that cannot be reached from a black box level.

### Black Box Testing

Black Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests, as most other kinds of tests, must be written from a definitive source document, such as specification or requirements document, such as specification or requirements document. It is a testing in which the software under test is treated, as a black box .you cannot “see” into it. The test provides inputs and responds to outputs without considering how the software works.

### Unit Testing

Unit testing is usually conducted as part of a combined code and unit test phase of the software lifecycle, although it is not uncommon for coding and unit testing to be conducted as two distinct phases.

### Test Strategy and Approach

Field testing will be performed manually and functional tests will be written in detail.

### Test Objectives

* All field entries must work properly.
* Pages must be activated from the identified link.
* The entry screen, messages and responses must not be delayed.

### Features to be Tested

* Verify that the entries are of the correct format
* No duplicate entries should be allowed
* All links should take the user to the correct page.

### Integration Testing

Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects. The task of the integration test is to check that components or software applications, e.g. components in a software system or – one step up – software applications at the company level – interact without error.

**Test Results:** All the test cases mentioned above passed successfully. No defects encountered.

### Acceptance Testing

User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.

**Test Results:** All the test cases mentioned above passed successfully. No defects encountered.

### Unit Testing

Unit testing is usually conducted as part of a combined code and unit test phase of the sdlc Although it is not uncommon for coding and unit testing to be conducted as two distinct phases.

## Testing Strategies

Field testing will be performed manually and functional tests will be written in detail.

### Test Objectives

* All field entries must work properly.
* Pages must be activated from the identified link.
* The entry screen, messages and responses must not be delayed.

### Features to be Tested

* Verify that the entries are of the correct format
* No duplicate entries should be allowed
* All links should take the user to the correct page.

### Integration Testing

Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interfaced effects. The task of the integration test is to check that components or software applications, e.g. components in a software system or – one step up – software applications at the company level interact without error.

**Test Results:** All the test cases mentioned above passed successfully. No defects encountered.

## Sample Test Cases

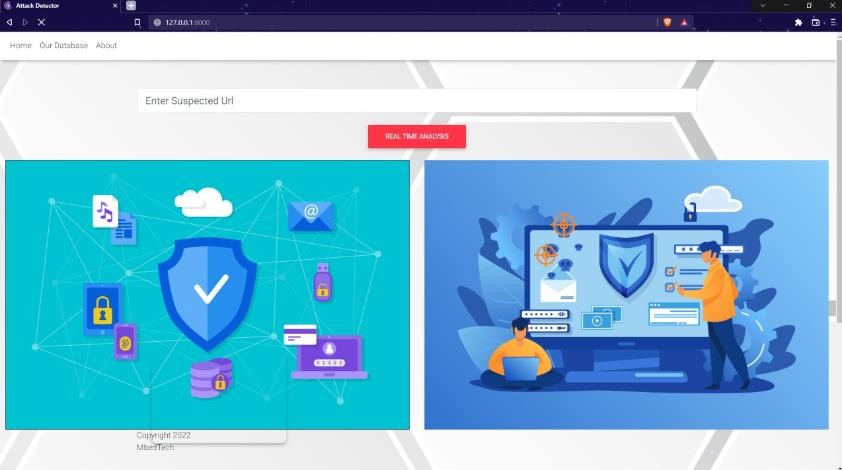
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Test Case | Expected Result | Result | Remarks (If Fails) |
| 1. | User login application | Application logged in | Pass | Provide login details |
| 2. | Provide input data | Check if the URL is affected or not | Pass | shows error the given input is not valid |

**Table 7.3.1** Sample Test Cases

### Test Results

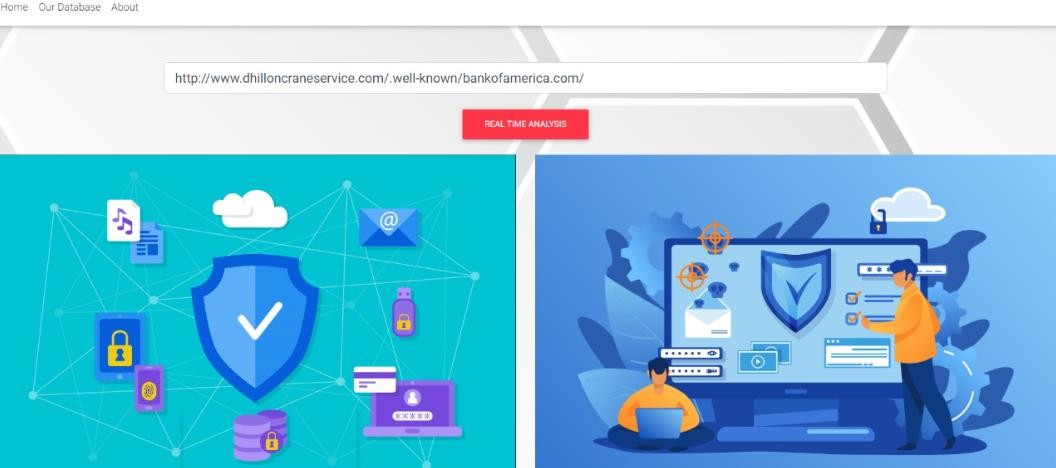
All the Test cases mentioned above passed Successfully. No defects are encountered.

# OUTPUT SCREENS



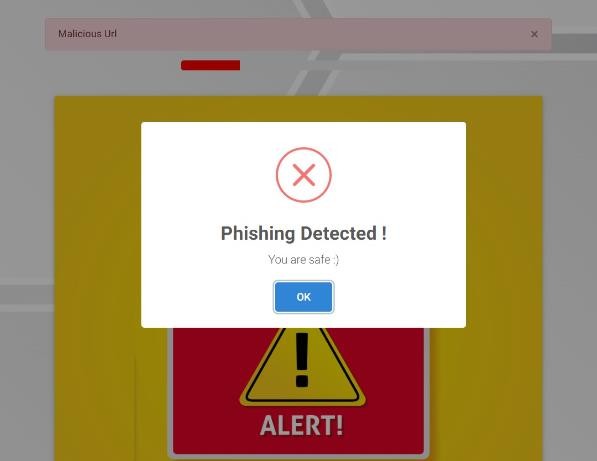
**Fig 8.1** Home page

This is the home page in this page we will upload the URL to detect whether the given URL is either malicious or not.



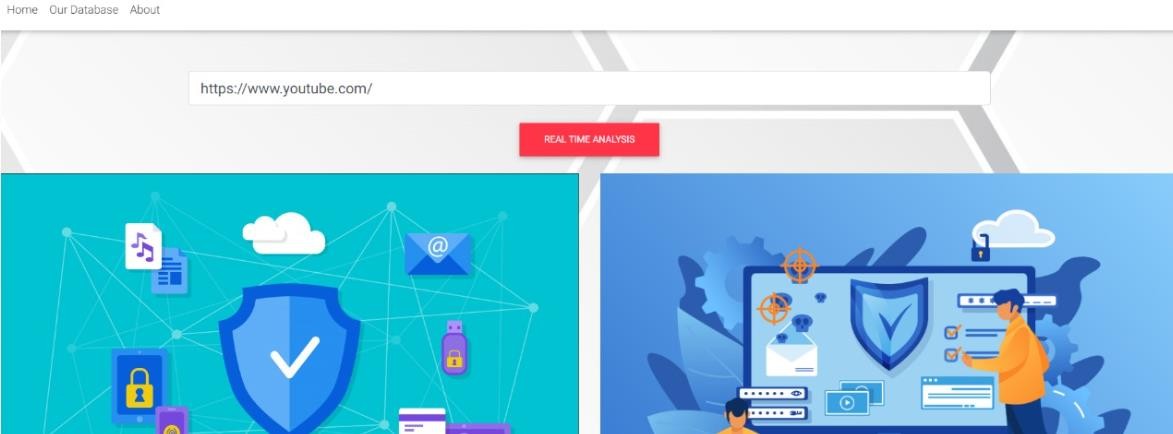
**Fig 8.2** Inserting an URL

Here we upload the URL took from different resourses.



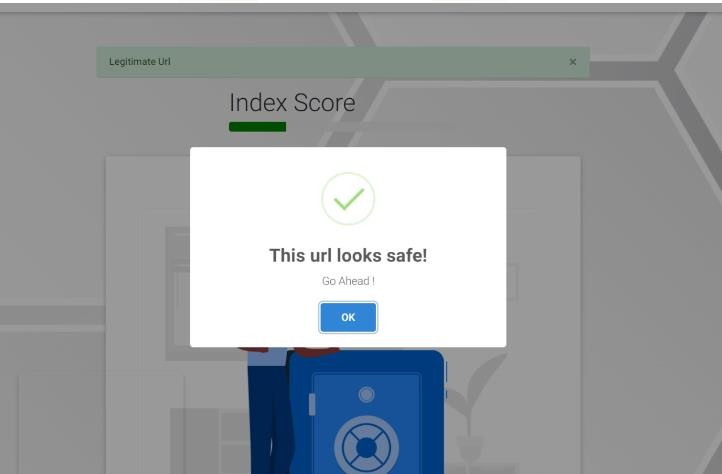
**Fig 8.3** Malicious URL Detected

In the figure the given URL is detected as malicious that may install a malicious bot in our device.



**Fig 8.4** Uploading an URL

Here we upload the URL took from different resourses.



**Fig 8.5** Phishing Detected

The given URL is non Malicious and the user can use the URL.

# CONCLUSION

This article presents an LA-MSBD algorithm by integrating a trust computational model with a set of URL-based features for MSBD. In addition, we evaluate the trustworthiness of tweets (posted by each participant) by using the Bayesian learning and DST. Moreover, the proposed LA- MSBD algorithm executes a fifinite set of learning actions to update action probability value (i.e., probability of a participant posting malicious URLs in the tweets).

The proposed LA-MSBD algorithm achieves the advantages of incremental learning. Two Twitter data sets are used to evaluate the performance of our proposed LA-MSBD algorithm. The experimental results show that the proposed LA-MSBD algorithm achieves upto 7% improvement of accuracy compared with other existing algorithms. For The Fake Project and Social Honeypot data sets, the proposed LA-MSBD algorithm has achieved precisions of 95.37% and 91.77% for MSBD, respectively. Furthermore, as a future research challenge, we would like to investigate the dependence among the features and its impact on MSBD.

# . FUTURE ENHANCEMENT

The Regression methods or the algorithms using the regression will be providing more accuracy than the classification algorithms. As we used SVM algorithm using classification in the proposed system we will be using SVM algorithm using Regression in the future so that the accuracy will be increased.

# REFERENCES

1. G. Lingam, R. R. Rout, and D. V. L. N. Somayajulu, “Adaptive deep Q-learning model for detecting social bots and influential users in online social networks,” Appl. Intell., vol. 49, no. 11, pp. 3947– 3964, Nov. 2019.
2. A.K. Jain and B. B. Gupta, “A machine learning based approach for phishing detection using hyperlinks information,” J. Ambient Intell. Hum. Comput., vol. 10, no. 5, pp. 2015–2028, May 2019.
3. P. Shi, Z. Zhang, and K.-K.-R. Choo, “Detecting malicious social bots based on clickstream sequences,” IEEE Access, vol. 7, pp. 28855–28862, 2019.
4. S. Madisetty and M. S. Desarkar, “A neural network-based ensemble approach for spam detection in Twitter,” IEEE Trans. Comput. Social Syst., vol. 5, no. 4, pp. 973–984, Dec. 2018.
5. D. R. Patil and J. B. Patil, “Malicious URLs detection using decision tree classifiers and majority voting technique,” Cybern. Inf. Technol., vol. 18, no. 1, pp. 11–29, Mar. 2018.
6. H. Gupta, M. S. Jamal, S. Madisetty, and M. S. Desarkar, “A framework for real-time spam detection in Twitter,” in Proc. 10th Int. Conf. Commun. Syst. Netw. (COMSNETS), Jan. 2018.
7. G. Lingam, R. R. Rout, and D. V. L. N. Somayajulu, “Detection of social botnet using a trust model based on spam content in Twitter network,” in Proc. IEEE 13th Int. Conf. Ind. Inf. Syst. (ICIIS), Dec. 2018.
8. M. Al-Janabi, E. D. Quincey, and P. Andras, “Using supervised machine learning algorithms to detect suspicious URLs in online social networks,” in Proc. IEEE/ACM Int. Conf. Adv. Social Netw. Anal. Mining, Jul. 2017.
9. T. Wu, S. Liu, J. Zhang, and Y. Xiang, “Twitter spam detection based on deep learning,” in Proc. Australas. Comput. Sci. Week Multiconf. (ACSW), 2017.
10. H. B. Kazemian and S. Ahmed, “Comparisons of machine learning techniques for detecting malicious webpages,” Expert Syst. Appl., vol. 42, no. 3, pp. 1166–1177, Feb. 2015.
11. C.-M. Chen, D. J. Guan, and Q.-K. Su, “Feature set identification for detecting suspicious URLs using Bayesian classification in social networks,” Inf. Sci., vol. 289, pp. 133–147, Dec. 2014.