Anti-V Solutions

A PROJECT REPORT

Submitted by

Akshay Markhedkar(19BCE10355)

in partial fulfillment for the award of the degree of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE AND ENGINEERING

SCHOOL OF COMPUTING SCIENCE AND ENGINEERING

VIT BHOPAL UNIVERSITY

KOTHRIKALAN, SEHORE

MADHYA PRADESH - 466114

SEP 2022

VIT BHOPAL UNIVERSITY,KOTHRIKALAN, SEHORE MADHYA PRADESH – 466114

BONAFIDE CERTIFICATE

Certified that this project report titled **"Anti-V Solutions"** is the bonafide work of "**AKSHAY MARKHEDKAR (19BCE10355)".** who carried out the project work under my supervision. Certified further that to the best of my knowledge the work reported here does not form part of any other project / research work on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

PROGRAM CHAIR
Dr. J MANIKANDAN
School of Computer Science and Engineering
VIT BHOPAL UNIVERSITY

PROJECT GUIDE
Dr. Anand Motwani
School of Computer Science and Engineering
VIT BHOPAL UNIVERSITY

ACKNOWLEDGEMENT

First and foremost I would like to thank the Lord Almighty for His presence and immense blessings throughout the project work.

I wish to express my heartfelt gratitude to Dr. MANAS KUMAR MISHRA, Head of the Department, School of Computer Science for much of his valuable support encouragement in carrying out this work.

I would like to thank my internal guide Dr. Anand Motwani ,for continually guiding and actively participating in my project, giving valuable suggestions to complete the project work.

I would like to thank all the technical and teaching staff of the School of Computer Science, who extended directly or indirectly all support.

Last, but not the least, I am deeply indebted to my parents who have been the greatest support while I worked day and night for the project to make it a success.

LIST OF FIGURES

Figure 1- Use Case Diagram 09

Figure 2- File Upload Flow Diagram 10

Figure 3- Image File Upload Flow Diagram 10

Figure 4- Text File Upload Flow Diagram 11

Figure 5- PPT File Upload Flow Diagram 11

Figure 6- Executable File Upload Flow Diagram 11

Figure 7- How Django Works Diagram 13

Figure 8- Anti-V Application Home Page 18

Figure 9- Figure 9- Anti-V Application Result Page

- When there is a Malicious File 23

Figure 10- Figure 10- Anti-V Application Result Page

- When there is not Malicious File 23

ABSTRACT

Anti-V Solutions

Anti-V Solution Software is a very useful software for people who want to check and ensure that the file they had downloaded is malicious or not. Anti-V Solution Software provides user flexibility to search, check and find the file's intention from database. It provides the complete details of the virus type and version if get found. Files from any type can be scanned by this software. Simple Web API are used for scanning files from database. From the time of origin and still existence can be found by this software. With the database of thousand of hashes of viruses, users get a good assurity about the file. It also has a database of details about the viruses so after the detection it will give the output as the details of the viruses.

Key features of App:

- Check whether a file is malicious or not.
- Easy to use.
- Any type of file can be scanned (.pdf, .jpg, .ppt etc ...)

What are the different requirements to create this app?

This system helps us to check if the file is a virus or not , any type of the file can be checked by this software and the most advantage of this software is that we can easily change it's library or modify or update it with the present scenarios.

Software and languages used for creating this application:

- Softwares :
  - Sublime Text 3
  - Jupyter Notebook
  - Notepad
  - Anaconda
- Languages :
  - Django
  - Python
  - HTML / CSS

TABLE OF CONTENTS

CHAPTER NO.	TITLE	PAGE NO.
	List of Abbreviations	iii
	List of Figures	iv
	Abstract	v
	List of Tables	vi
1	**INTRODUCTION**	vii
	1.1 Introduction	vii
	1.2 Motivation for the work	vii
	1.3 Problem Statement and Objective of the work	vii
	1.5 Summary	vii
2	**LITERATURE SURVEY**	viii
	2.1 User Options	ix
	2.2 File Upload	ix
	2.2.1 File Upload System.	x
	2.2.2 Image (.jpg / .jpeg / .png / .bmp) File Upload	xi
	2.2.3 Text (.txt) File Upload	xi
	2.2.4 PPT (.ppt / .pptx) File Upload	xi
	2.2.5 EXE (.exe) File Upload	xi

3	**SYSTEM ANALYSIS**	xii
	3.0 Django Development Basics	xii
	3.1 What does Django code look like?	xii
	3.1.1 Views	xiii
	3.1.2 Urls	xiii
	3.1.3 Models	xiii
	3.1.4 Templates	xiii
	3.2 Sending the request to the right view (urls.py)	xiv
	3.3Handling the request (views.py)	xv
	3.4 Defining Data Models	xv
	3.5 Rendering Data (HTML Templates)	xvi
	3.5.1 Sample of Rendering	xvi
4	**SYSTEM DESIGN AND IMPLEMENTATION**	xvii
	4.1 Web Application Home Page	xvii
	4.2 Urls.py files	xix
	4.3 Forms.py file	xix
	4.4 Views.py file	xx
	4.5 Result Page	xxii
	4.6 Future Work	xxiv
5	**FUTURE ENHANCEMENT AND CONCLUSION**	xxv
	6.1 Limitation/Constraints of the System	
	6.2 Future Enhancements	
	6.3 Conclusion	

Chapter 1
INTRODUCTION

1.1 Introduction

Anti-V Solution Software is a very useful software for people who want to check and ensure that the file they had downloaded is malicious or not. Anti-V Solution Software provides user flexibility to search, check and find the file's intention from database. It provides the complete details of the virus type and version if get found.

My software "Anti-V Solutions" makes finding viruses easy. Files from any type can be scanned by this software. Simple Web API are used for scanning files from database. From the time of origin and still existence can be found by this software. With the database of thousand of hashes of viruses, users get a good assurity about the file. It also has a database of details about the viruses so after the detection it will give the output as the details of the viruses.

1.2 Motivation for the work

Everyday lakhs of hacking attacks are happening in the world but 50% + of them are caused due to viruses . Hacker penetrate into the system by these types of malicious scripts by using social engineering tools they uses same viruses many times and does think which virus will work if our fortune is worst a simple might get work ,so in-order to save from them whenever we download a file we must have to test it and for this "Anti-V Solutions" software is designed , user can check the file to know whether the file is malicious or not.

1.3 Problem Statement and Objective

The problem is how can we check that the file we had downloaded is malicious or not. To help in this context our "Anti-V Solutions" software comes into the picture , by uploading the file before it's opening or execution we can check it in our software to assure that file is not malicious one.

1.4 Summary

To stop the hackers get into their system. One have to check the file which they had downloaded so they not get affected by the malicious script. This "Anti-V Solutions" Software helps one to identify, the file and give the information to the user about the file whether it is malicious or not.

Chapter 2

LITERATURE SURVEY

2.1 User Options

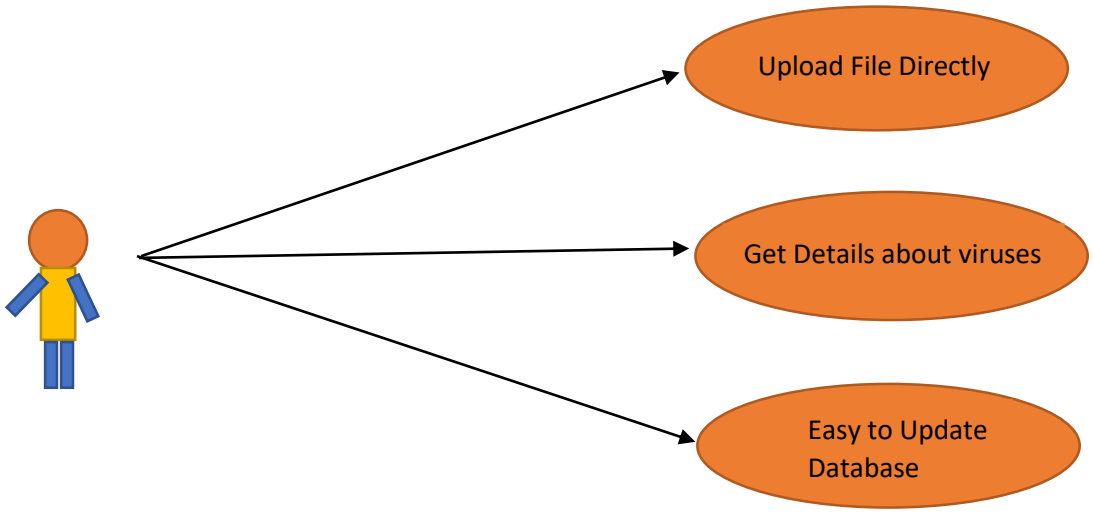


Figure 1- Use Case Diagram

Users of Anti-V Solutions software have following options as shown in Figure 1:

- User can upload file directly in one click.
- User can view details about the viruses
- User can easily update the database according to time very easily.

The following section describes each part in detail.

2.2 File Upload

We can upload file by click upload button. Any file of any type can be uploaded in this software, after the upload just press submit button. After the internal process completion we get the result about the file whether it is harmful or not.

2.2.1 File upload process in software

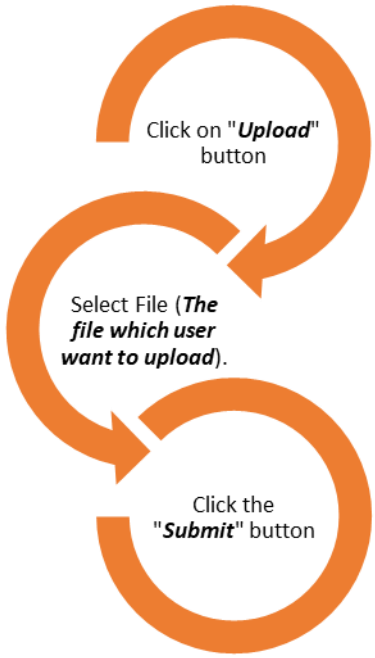


Figure 2- File Upload Flow Diagram

User can upload any file by click upload button. Then user have to select the file of any type , after the successful upload just press submit button. After all processes system will send the file to the internal system and user will get the result in the result page.

2.2.2 Image (.jpg / .jpeg / .png / .bmp) File Upload Process

Figure 3- Image File Upload Flow Diagram

User can upload image file by click upload button. Then user have to select the image file of any type , after the successful upload just press submit button. After all processes system will send the file to the internal system and user will get the result in the result page.

2.2.3 Text (.txt) File Upload

Figure 4- Text/Word File Upload Flow Diagram

User can upload text file by click upload button. Then user have to select the text file of any type , after the successful upload just press submit button. After all processes system will send the file to the internal system and user will get the result in the result page.

2.2.4 PPT (.ppt / .pptx) File Upload

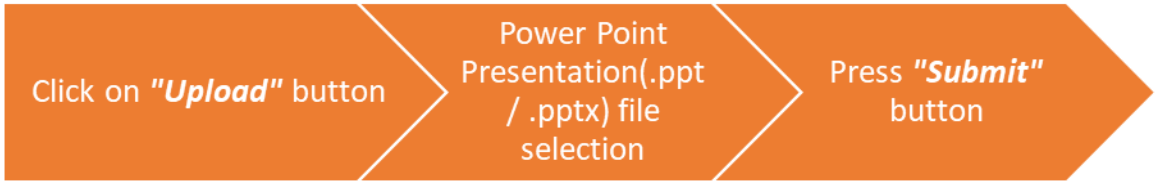


Figure 5- Power-point Presentation File Upload Flow Diagram

User can upload image file by click upload button. Then user have to select the image file of any type , after the successful upload just press submit button. After all processes system will send the file to the internal system and user will get the result in the result page.

2.2.5 EXE (.exe) File Upload

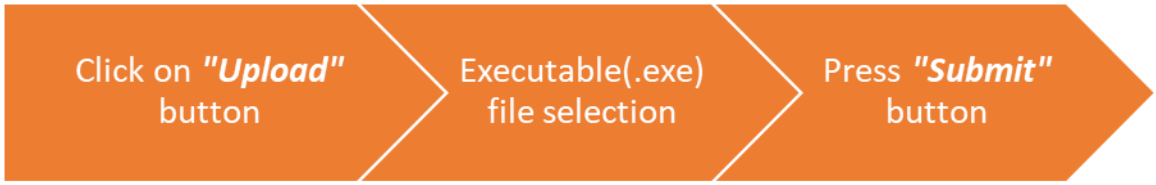


Figure 6- Executable File Upload Flow Diagram

User can upload executable file by click upload button. Then user have to select the executable file of any type , after the successful upload just press submit button. After all processes system will send the file to the internal system and user will get the result in the result page.

Chapter 3

Django Development Basics

3.0 Django Development Basics

Django is a high-level Python web framework that enables rapid development of secure and maintainable websites. Built by experienced developers, Django 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 is free and open source, has a thriving and active community, great documentation, and many options for free and paid-for support. [1].

Anti-V Solution Software is written in python programming language using SQLite database. The code of Anti-V Solution Software is compiled using Sublime Text 3 code editor into a Django Project in Anaconda environment, developed using HTML/CSS and Python Languages. These multiple files is considered in one application and run under the Django Web Server.

Django is written in Python, which runs on many platforms. That means that you are not tied to any particular server platform, and can run your applications on many flavors of Linux, Windows, and macOS. Furthermore, Django is well-supported by many web hosting providers, who often provide specific infrastructure and documentation for hosting Django sites.

3.1 What does Django code look like?

In a traditional data-driven website, a web application waits for HTTP requests from the web browser (or other client). When a request is received the application works out what is needed based on the URL and possibly information in POST data or GET data. Depending on what is required it may then read or write information from a database or perform other tasks required to satisfy the request. The application will then return a response to the web browser, often dynamically creating an HTML page for the browser to display by inserting the retrieved data into placeholders in an HTML template.

Django web applications typically group the code that handles each of these steps into separate file**s:**

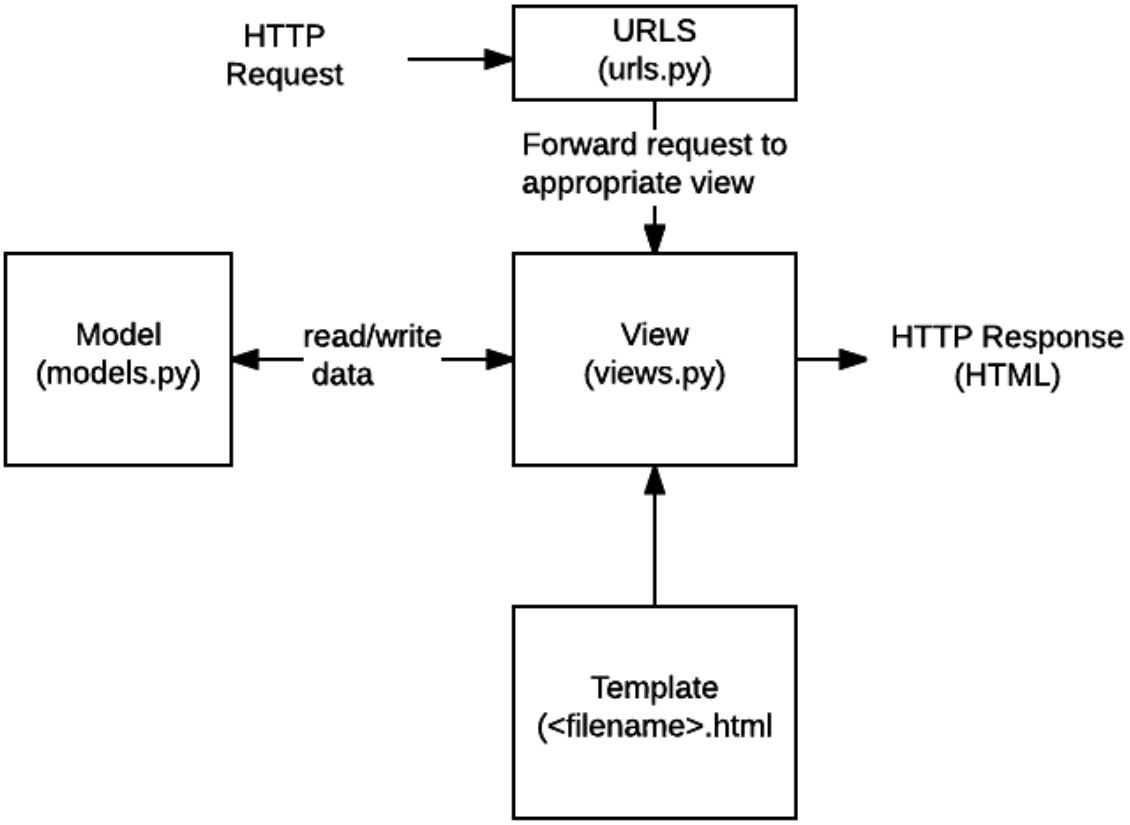


Figure-7 How Django Works

3.1.1 URLs:

While it is possible to process requests from every single URL via a single function, it is much more maintainable to write a separate view function to handle each resource. A URL mapper is used to redirect HTTP requests to the appropriate view based on the request URL. The URL mapper can also match particular patterns of strings or digits that appear in a URL and pass these to a view function as data.

3.1.2 View:

A view is a request handler function, which receives HTTP requests and returns HTTP responses. Views access the data needed to satisfy requests via *models*, and delegate the formatting of the response to *templates*.

3.1.3 Models:

Models are Python objects that define the structure of an application's data, and provide mechanisms to manage (add, modify, delete) and query records in the database.

3.1.4 Templates:

A template is a text file defining the structure or layout of a file (such as an HTML page), with placeholders used to represent actual content. A *view* can dynamically create an HTML page using an HTML template, populating it with

data from a *model*. A template can be used to define the structure of any type of file; it doesn't have to be HTML!

The sections below will give you an idea of what these main parts of a Django app look like (we'll go into more detail later on in the course, once we've set up a development environment).

3.2 Sending the request to the right view (urls.py)

A URL mapper is typically stored in a file named urls.py. In the example below, the mapper (urlpatterns) defines a list of mappings between *routes* (specific URL *patterns)* and corresponding view functions. If an HTTP Request is received that has a URL matching a specified pattern, then the associated view function will be called and passed the request.

```
"""myVirusChecker URL Configuration """

from django.conf.urls import include
from django.contrib import admin
from django.urls import path

urlpatterns = [
   path(r'fileupload', include('fileUpload.urls')),
   path('admin/', admin.site.urls),
]



"""fileUpload URL Configuration"""
from django.conf.urls import include
from django.contrib import admin
from django.urls import path
from .views import home
from . import views

urlpatterns = [

   path(r'/', views.home, name='home'),
   path(r'/fileprocess', views.fileprocess, name='fileprocess'),

]
```

The urlpatterns object is a list of path() and/or re_path() functions (Python lists are defined using square brackets, where items are separated by commas and may have an optional trailing comma. For example: [item1, item2, item3,]).

The "myVirusChecker URL Configuration" is a route (pattern) that will be matched.
The path() method uses angle brackets to define parts of a URL that will be captured and passed through include and it will attach the url from the first front url as well as it will call the url.py file from the fileUpload folder.

The "fileUpload URL Configuration" is the route that will be called when the pattern is matched. The notation views.fileprocess indicates that the function is called fileprocess() and can be found in a module called views (i.e. inside a file named views.py)

3.3 Handling the request (views.py)

Views are the heart of the web application, receiving HTTP requests from web clients and returning HTTP responses. In between, they marshal the other resources of the framework to access databases, render templates, etc.
The example below shows a minimal view function index(), which could have been called by our URL mapper in the previous section. Like all view functions it receives an HttpRequest object as a parameter (request) and returns an HttpResponse object. In this case we don't do anything with the request, and our response returns a hard-coded string. We'll show you a request that does something more interesting in a later section.

```
# filename: views.py (Django view functions)

from django.http import HttpResponse

def index(request):
    # Get an HttpRequest - the request parameter
    # perform operations using information from the request.
    # Return HttpResponse
    return HttpResponse('Hello from Django!')
```

Views are usually stored in a file called views.py.

3.4 Defining data models (models.py)

Django web applications manage and query data through Python objects referred to as models. Models define the structure of stored data, including the field *types* and possibly also their maximum size, default values, selection list options, help text for documentation, label text for forms, etc. The definition of the model is independent of the underlying database — you can choose one of several as part of your project settings. Once you've chosen what database you want to use, you don't need to talk to it directly at all — you just write your model structure and other code, and Django handles all the "dirty work" of communicating with the database for you.
The code snippet below shows a very simple Django model for a Team object. The Team class is derived from the Django class models.Model. It defines the team name and team level as character fields and specifies a maximum number of characters to be stored for each record.
The team_level can be one of several values, so we define it as a choice field and provide a mapping between choices to be displayed and data to be stored, along with a default value.

3.5 Rendering data (HTML templates)

Template systems allow you to specify the structure of an output document, using placeholders for data that will be filled in when a page is generated. Templates are often used to create HTML, but can also create other types of document. Django supports both its native templating system and another popular Python library called Jinja2 out of the box (it can also be made to support other systems if needed).

This function uses the render() function to create the HttpResponse that is sent back to the browser. This function is a *shortcut*; it creates an HTML file by combining a specified HTML template and some data to insert in the template.

3.5.1 Sample of Rendering:

```
## filename: best/templates/best/index.html

<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>Home page</title>
</head>
<body>
  {% if youngest_teams %}
    <ul>
      {% for team in youngest_teams %}
        <li>{{ team.team_name }}</li>
      {% endfor %}
    </ul>
  {% else %}
    <p>No teams are available.</p>
  {% endif %}
</body>
</html>
```

```
# filename: views.py

from django.shortcuts import render
from .models import Team

def index(request):
    list_teams =
Team.objects.filter(team_level__exact="U09")
    context = {'youngest_teams': list_teams}
    return render(request, '/best/index.html',
context)
```

Chapter 4

SYSTEM DESIGN AND IMPLEMENTATION

4.1 Web Application Home Page

When user opens Anti-V Solutions application, application home page as shown in Figure 14 is displayed. User can perform any available option from application home page. The user can upload files by clicking on upload button and submit the file by clicking on submit button.

```
<!DOCTYPE html>
<html>
<head>
	<title>My Home</title>
</head>
<body>
	<h1>Project Solution</h1>
	<h2><i>The Virus Detector</i></h2>
	<h2>Please upload a <i>"File"</i> for the <i>"Scan"</i></h2>

	<form action="fileprocess" method="POST" enctype="multipart/form-
data">
		{% csrf_token %}
		<input type="file" name="file"/>
		<input type="submit" name="Submit">
	</form>

	<h4>Discription of project :</h4>
	<p>The following project helps us to check whether a file is malicious or
not.</p>



</body>
</html>
```

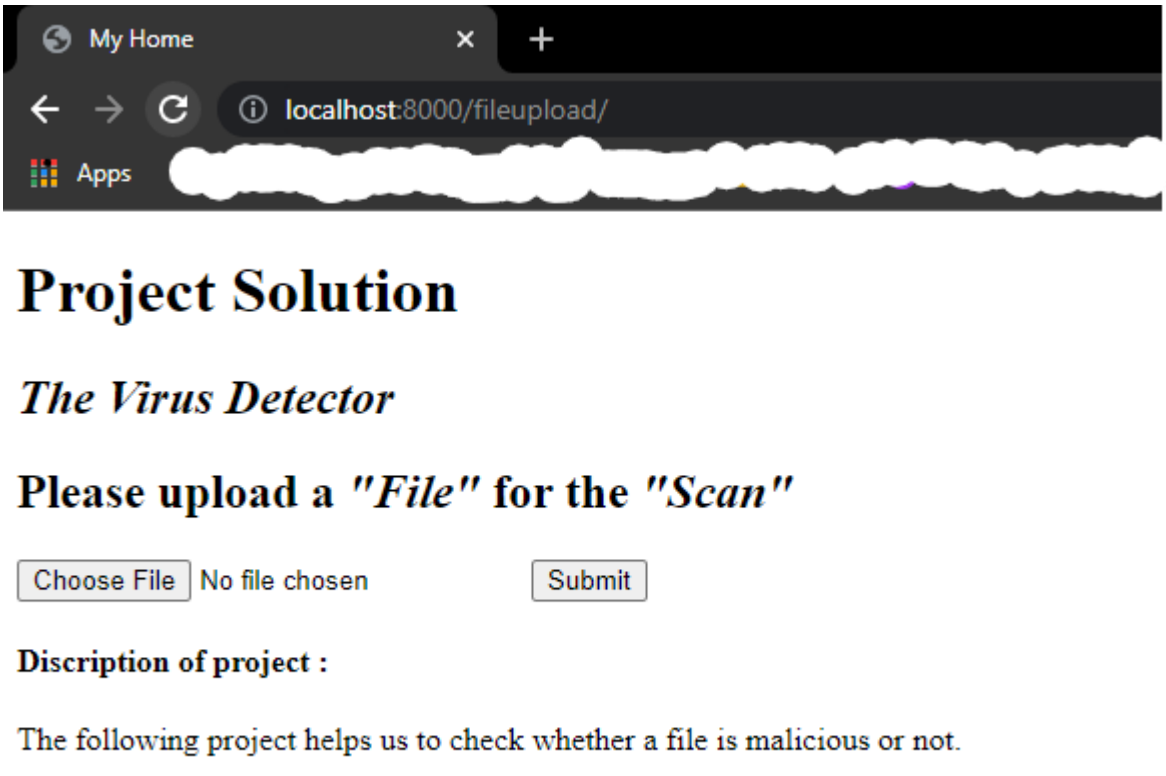


Figure 8- Anti-V Application Home Page

4.2 URLs

```
"""myVirusChecker URL Configuration """

from django.conf.urls import include
from django.contrib import admin
from django.urls import path

urlpatterns = [
    path(r'fileupload', include('fileUpload.urls')),
    path('admin/', admin.site.urls),
]
```

```
"""fileUpload URL Configuration"""
from django.conf.urls import include
from django.contrib import admin
from django.urls import path
from .views import home
from . import views

urlpatterns = [

    path(r'/', views.home, name='home'),
    path(r'/fileprocess', views.fileprocess,
name='fileprocess'),

]
```

4.3 Forms

```
from django import forms

class  FileUploadForm(forms.Form):
          """docstring for  """
          file = forms.FileField()
```

4.4 Views

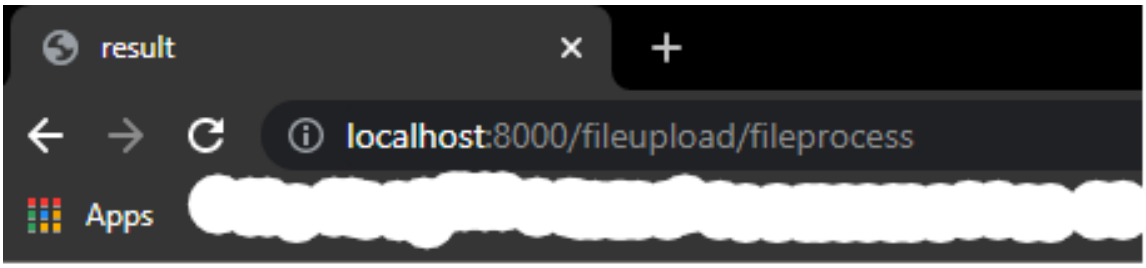
```
from django.shortcuts import render
from .forms import FileUploadForm

#now cyber part
import hashlib
#-----------


def handle_uploaded_file(f):
        with open('x','wb+') as destination:
                for chunk in f.chunks():
                        destination.write(chunk)

# Create your views here.

def home(request):
        return render(request,'home.html')

def fileprocess(request):
        form = FileUploadForm(request.POST, request.FILES)
        if form.is_valid():
                handle_uploaded_file(request.FILES['file'])

                #Cyber Security Begins
                #Malware Detection Project Starts Here
                #//-----------------------------------------------

                #Global Variable
                global malware_hashes #= list(open('vhash.txt','r').read().split('\n'))
                malware_hashes = list(open('vhash.txt','r').read().split('\n'))
                global virusInfo #= list(open('vrinfo.txt','r').read().split('\n'))
                virusInfo = list(open('vrinfo.txt','r').read().split('\n'))
```

```
        #Get Hash of File
        def sha256_hash(filename):
                with open(filename,'rb') as f:
                        bytes = f.read()
                        sha256hash = hashlib.sha256(bytes).hexdigest()
                        f.close()

                        print(sha256hash)
                return sha256hash

        #Malware Detection By Hash
        def malware_checker(pathOfFile):
                #global malware_hashes
                #global virusInfo

                hash_malware_check = sha256_hash(pathOfFile)
                counter = 0


                for i in malware_hashes:
                        if i == hash_malware_check:
                                return virusInfo[counter]

                        counter += 1

                return 0

        print(malware_checker('x'))


        res = malware_checker('x')




        #//-----------------------------------------------


return render(request,'result.html',{'res' : res })
```

4.5 Result

In this page user will get the result about the file is malicious or not

```
<!DOCTYPE html>
<html>
<head>
	<title>result</title>
	<style type="text/css">

	</style>
</head>
<body>
	<h1>This is Result of your Scanning</h1>
	<p>You will get information regarding your uploaded file below :</p>




	<I><b><h2 style="color: red" >{{res}}</h2></b></I>




</body>
</html>
```



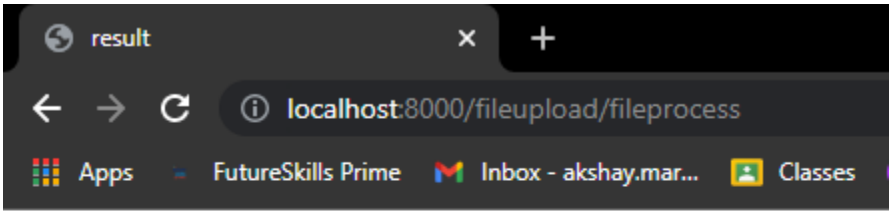
This is Result of your Scanning

You will get information regarding your uploaded file below :

Malicious Sample-test File as a Virus

Figure 9- Anti-V Application Result Page

When there is a Malicious File



This is Result of your Scanning

You will get information regarding your uploaded file below :

0

Figure 10- Anti-V Application Result Page

When there is not Malicious File

4.6 Future Work

This application provides easy interface for user to upload file and check it whether it is malicious or not. However in the future we will going to make a more comfortable GUI which helps user to use its web application. In the result we will add more information columns so one will get the full information in a sustainable way. Multiple Files checking function will also going to be added soon.

Chapter 5
CONCLUSION

This application is extremely handy and useful for checking a file to get the surety of the file its safe or not. There are various paid application on the internet but this is free and easy to use deployed on any machine and number of times can be used. Easy to modify its database of hashes. These types of small applications not only helps people to keep their system free of viruses but also give them a good amount of saving. Hackers will always try to attack number of times , sometimes with old malicious file and sometimes with new but this application modifying and updating system is so much simpler so most of the viruses will filter out.

But main aim of our project is not only to delete the virus but it also is to spread awareness about it. One must always have to check what he/she/o is downloading , its from anywhere the person like but must check otherwise it will cost him more than any paid-antivirus .It will help to make the lives of people simpler and stressfree.