Attack Of The Clones!

Final Report

Conor Lewis – C00246763
Supervisor – Chris Staff

Contents

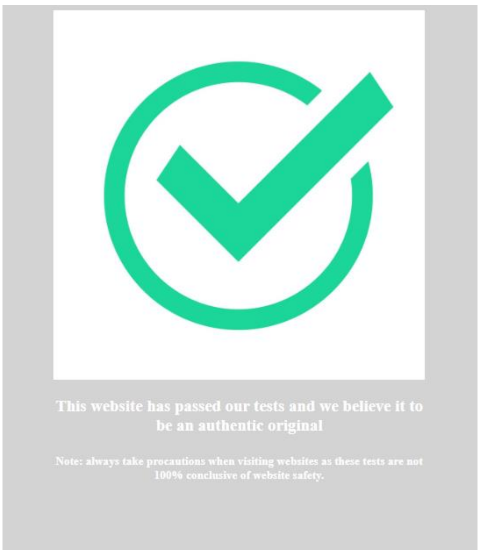
1. Introduction

Attack of the Clones is a fraudulent cloned website detection tool that can identify whether a website is an authentic original website or a clone of an already existing website. In this document I will be discussing the development process of the project. I will be discussing the tools used as part of this project in order to check legitimacy of the website and displaying this information to the users and I will also be discussing the skills I have developed during the creation of the project, the challenges faced during development and ways I overcame these challenges.

2. Description of Submitted Project

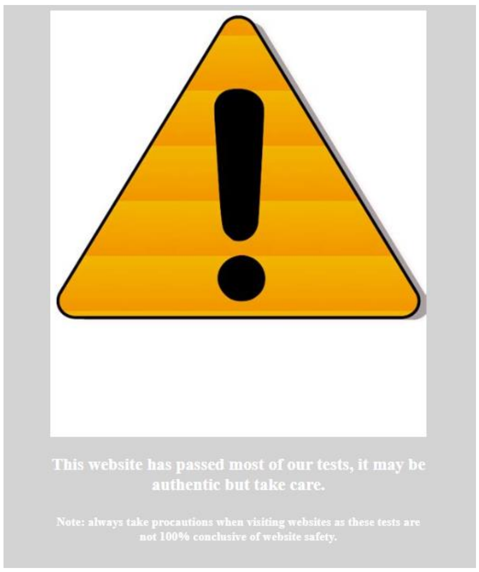
This cloned website detection tool detects fraudulent websites by running a number of scans on both the content on the website itself and the metadata of the website such as web certificates and WhoIs data of the website. The results of these tests are displayed to the user as a scale of safety, due to the large number of variables that are present on the wide spectrum of websites that are present on the internet.

In order for the program to run there are two separate applications that work together in order to operate. There is a browser extension developed for Google Chrome that the user can activate in order to check the legitimacy of the website the user is currently visiting. The browser extension then sends the URL of the website to the underlying python program and will display one of the pop-up windows (Fig. 2.1 – Fig. 2.4) depending on the finding of the underlying python application.


This website has passed our tests and we believe it to be an authentic original

Note: always take procautions when visiting websites as these tests are not 100% conclusive of website safety.

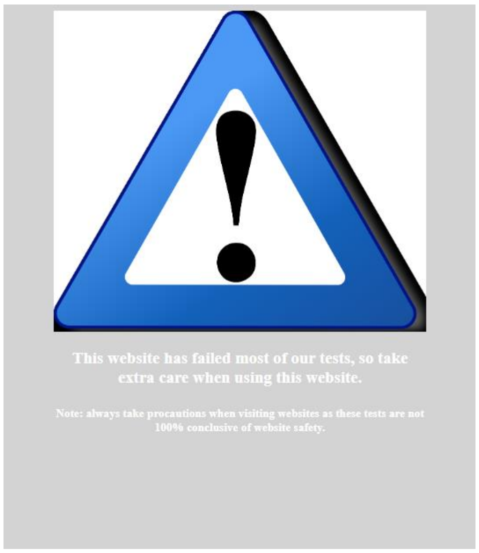
Figure 2.1: Popup window when website is deemed safe



This website has passed most of our tests, it may be authentic but take care.

Note: always take procautions when visiting websites as these tests are not 100% conclusive of website safety.

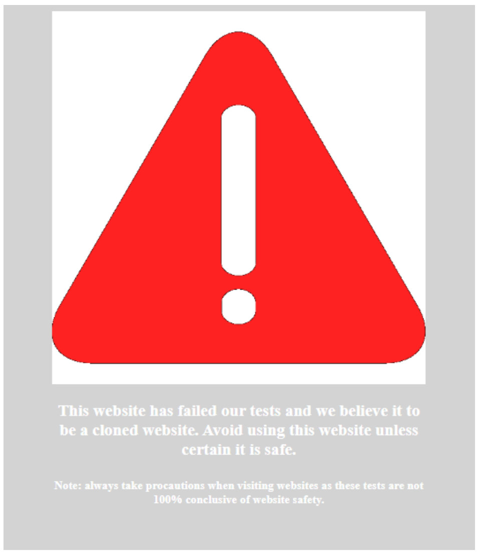
Figure 2.2: Popup window when website passes majority of tests



This website has failed most of our tests, so take extra care when using this website.

Note: always take procautions when visiting websites as these tests are not 100% conclusive of website safety.

Figure 2.3: Popup window when website only passes some tests


This website has failed our tests and we believe it to be a cloned website. Avoid using this website unless certain it is safe.

Note: always take precautions when visiting websites as these tests are not 100% conclusive of website safety.

Figure 2.4: Popup window when website is deemed unsafe

The underlying python application receives the website from the browser extension via a flask[1] server running from the python application and then perform runs three tests on the WhoIs[2] information. The tests are:

1. Check the status of the WhoIs information to ensure that the WhoIs information is valid.
2. Check that the domain on the WhoIs information is valid for the website that is associated with.
3. Check that the WhoIs information is from a list of accredited registrars.

The underlying python application then performs a further three tests on the SSL certificates for the website, the tests that are performed on the SSL certs are:

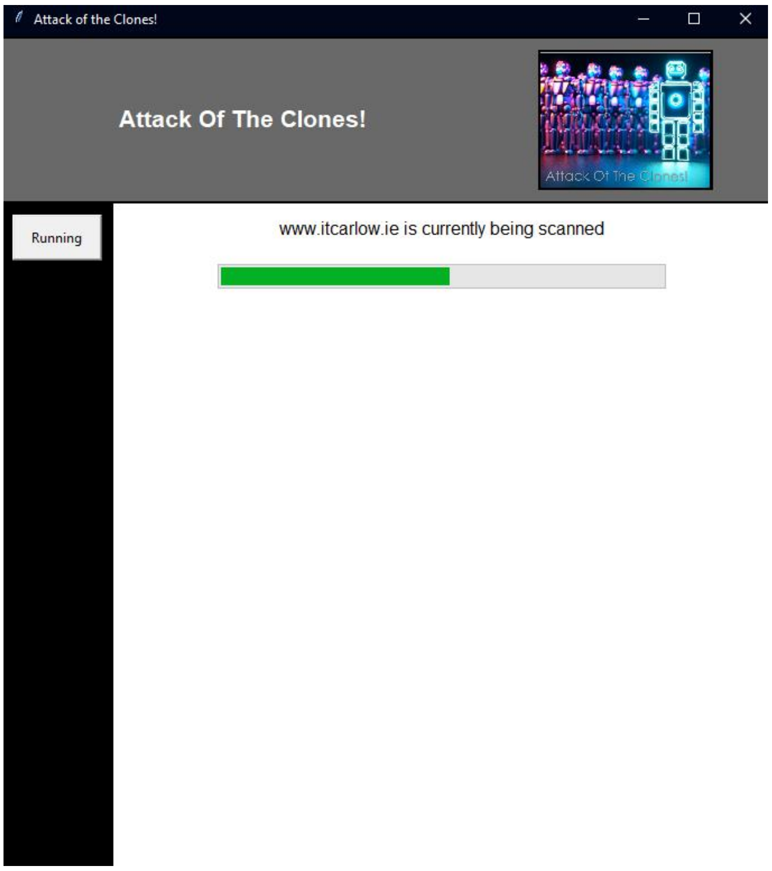
1. Check that the version is either TLS v1.3 or TLS v1.2 as other versions are since depreciated or considered no longer secure
2. Check that the SSL certificate is valid for the current time and not out of date.

[1] Flask.palletprojects.com. (n.d). *Welcome to Flask – Flask Documentation (2.2.x).* [online] Available at: https://www.flask.palletsprojects.com/en/2.2.x/.
[2] Whois.com. (2019). *Whois.com – Free Whois Lookup.* [online] Available at: https://www.whois.com/whois

3. Check that the SSL certificate is for the current website being visited by checking if the name of the website is present in the certificate information.

The underlying application then performs Named Entity Recognition in order to extract all organizations that are present on the website itself, then perform an internet search using duckduckgo[3] and seeing how similar the websites returned are to the searched website.

The python application notifies the user when it is running a search from the browser extension and also has the capability to show the progress of the scan through a progress bar that progresses as the search is being performed (Fig. 2.5).



www.itcarlow.ie is currently being scanned

Figure 2.5: Underlying python application's GUI

[3] DuckDuckGo (2019). *DuckDuckGo – Privacy, simplified.* [online] DuckDuckGo. Available at: https://duckduckgo.com/.

3. Description of Conformance to Specification and Design

3.1. Project Brief

Social engineering practices are used to get a scamming victim to visit a fraudulent website which is sometimes a clone of a legitimate website. Once the victim lands on the cloned website, they often disclose personal or private information, such as user ids, passwords, and bank account details. You are to determine whether the webpage the user is on is real or a scam, based upon the textual information and images on the webpage, and advise the user accordingly.

3.2. Mandatory Tasks

There were six mandatory tasks to be completed as part of this project, the first mandatory task was to create a web browser add-on. The second task was to extract phrases, etc., from the webpage source. The third mandatory task was to determine if the phrase is unique or if it appears on webpages of other organizations. The fourth perform an image search to see if they appear on other webpages and perform image search to see if they appear on other webpages. The fifth task was to determine if the website the user is on is legitimate or fake based on the search engine result. The final task would be to notify the user on the programs finding.

3.2.1. Web Browser add-on

For the web browser add-on, I developed a browser extension using HTML, CSS and JavaScript that would be available for Google Chrome, this extension would allow a user to open the extension when they have visited the website they wish to scan. The browser extension would provide the user a button to press a button in order to send the URL of the website to the underlying application via a request to the Flask server running as part of the underlying python application and be able to create a pop-up window to display findings to the user.

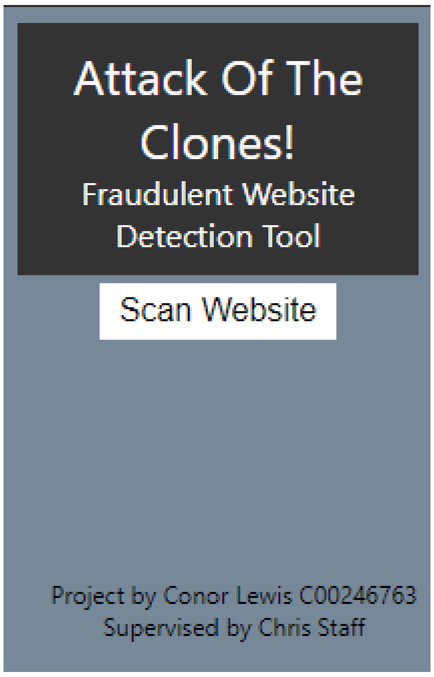
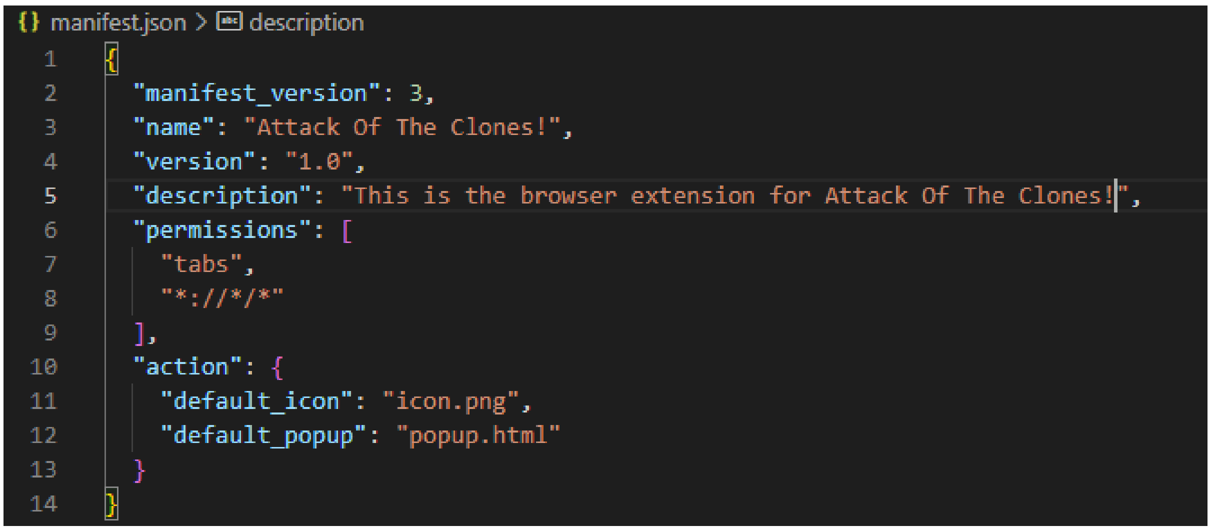

Attack Of The Clones!
Fraudulent Website Detection Tool
Scan Website
Project by Conor Lewis C00246763
Supervised by Chris Staff

Figure 3.1: Browser Extension that user can activate scan from

This section provided multiple challenges when creating this application as a crucial component of a browser extension is the manifest.json file of browser extension. This file provides Google Chrome with metadata regarding the browser extension. The manifest version of browser extensions is currently changing from manifest version 2 to manifest version 3 and manifest version 2 is soon to be deprecated. Manifest version 3 is a large change from manifest version 2 and is said to have great security benefits over the prior version, but this made creating this file much more difficult from a development perspective as a lot of the online resources regarding the creation of manifest.json files are for the soon to be depreciated manifest version and made the development of the browser extension much longer than initial estimated.

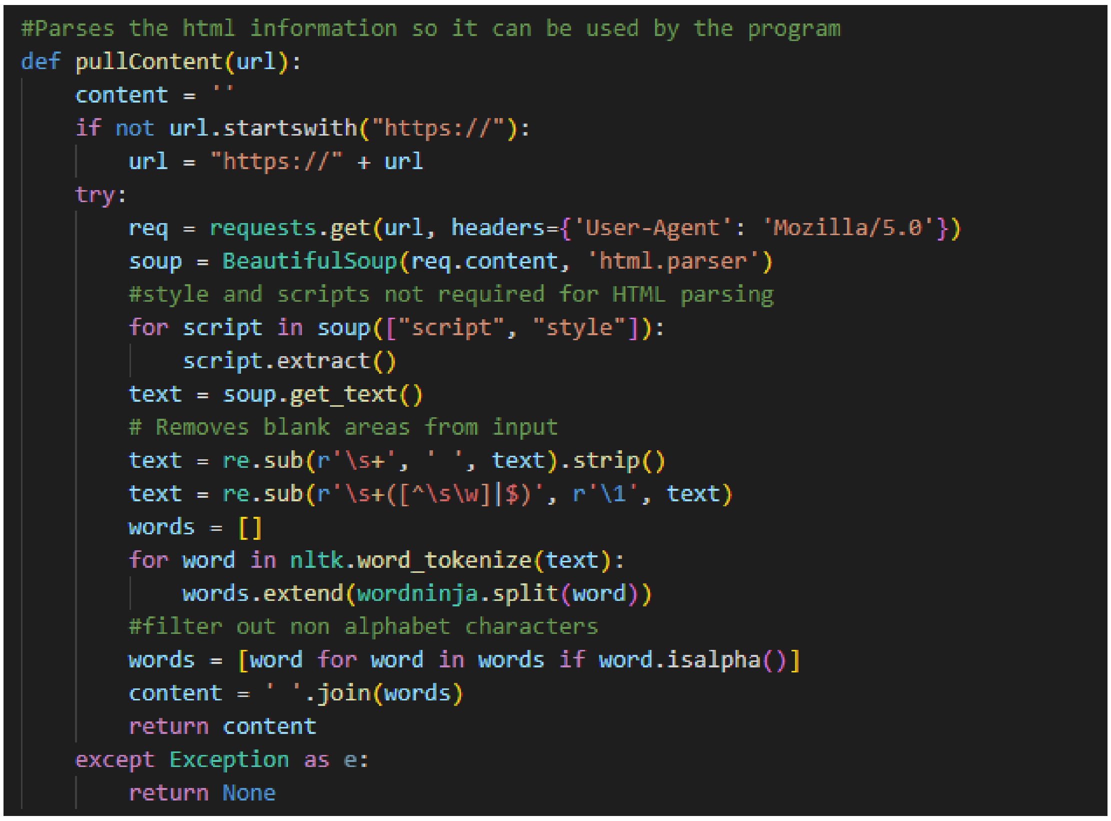


```
{} manifest.json > description
{
  "manifest_version": 3,
  "name": "Attack Of The Clones!",
  "version": "1.0",
  "description": "This is the browser extension for Attack Of The Clones!",
  "permissions": [
    "tabs",
    "*://*/*"
  ],
  "action": {
    "default_icon": "icon.png",
    "default_popup": "popup.html"
  }
}
```

Figure 3.2: Manifest.json file for browser extension

3.2.2. Text-processing system to extract phrases

In order to accomplish this task, I had to first parse the HTML information within the python application so that other areas of the program could operate on the information itself, for this I used a python library called BeautifulSoup[4], once the information was retrieved by the underlying python application I would then use natural language processing in the form of the Natural Language Toolkit [5](NLTK) to perform Named Entity Extraction and extract organizations that have been mentioned on a webpage.



```python
#Parses the html information so it can be used by the program
def pullContent(url):
    content = ''
    if not url.startswith("https://"):
        url = "https://" + url
    try:
        req = requests.get(url, headers={'User-Agent': 'Mozilla/5.0'})
        soup = BeautifulSoup(req.content, 'html.parser')
        #style and scripts not required for HTML parsing
        for script in soup(["script", "style"]):
            script.extract()
        text = soup.get_text()
        # Removes blank areas from input
        text = re.sub(r'\s+', ' ', text).strip()
        text = re.sub(r'\s+([^\s\w]|$)', r'\1', text)
        words = []
        for word in nltk.word_tokenize(text):
            words.extend(wordninja.split(word))
        #filter out non alphabet characters
        words = [word for word in words if word.isalpha()]
        content = ' '.join(words)
        return content
    except Exception as e:
        return None
```

Figure 3.4: Code used in order to pull content from a website

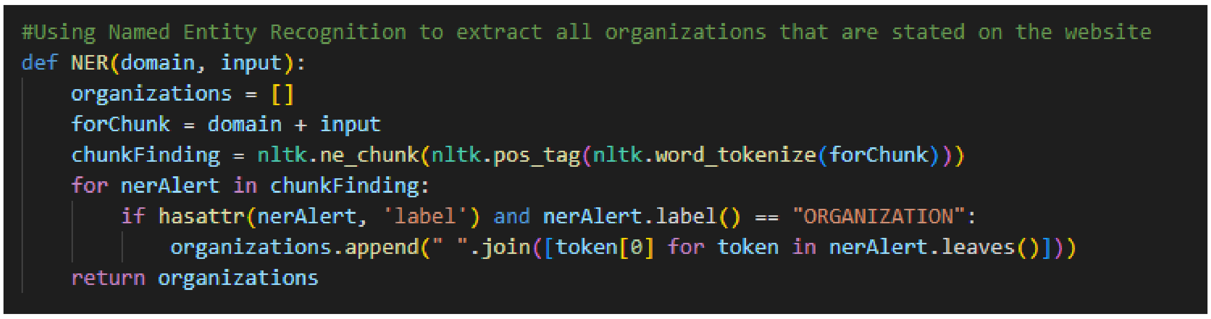
I had encountered some issues here with the data from the webpage displaying characters such as ‘\n’ or concatenating words together when the information was formatted in particular ways on the website being scanned and this would affect the natural language processing functions making incorrect results. I used additional libraries to better retrieve information from the webpages to

[4] Beautiful-soup-4.readthedocs.io. (n.d.). *Beautiful Soup Documentation – Beautiful Soup Documentation – Beautiful Soup 4.4.0 documentation.* [online] Available at: https://beautiful-soup-4.readthedocs.io/en/latest/.
[5] NLTK (2009). *Natural Language Toolkit – NLTK 3.4.4 documentation.* [online]. Nltk.org. Available at https://www.nltk.org

prevent these issues. The first library I used in addition to Natural Language Toolkit was wordninja[6], which would prevent concatenation of words were unintended. The second additional library I had to implement to better deal with HTML parsing of web contents was re, this library would allow the program to search for particular expressions in the content and replace it with another string that would not alter the results of natural language toolkit functions.

3.2.3. Determine if phrase is unique or appears on webpages of other organizations

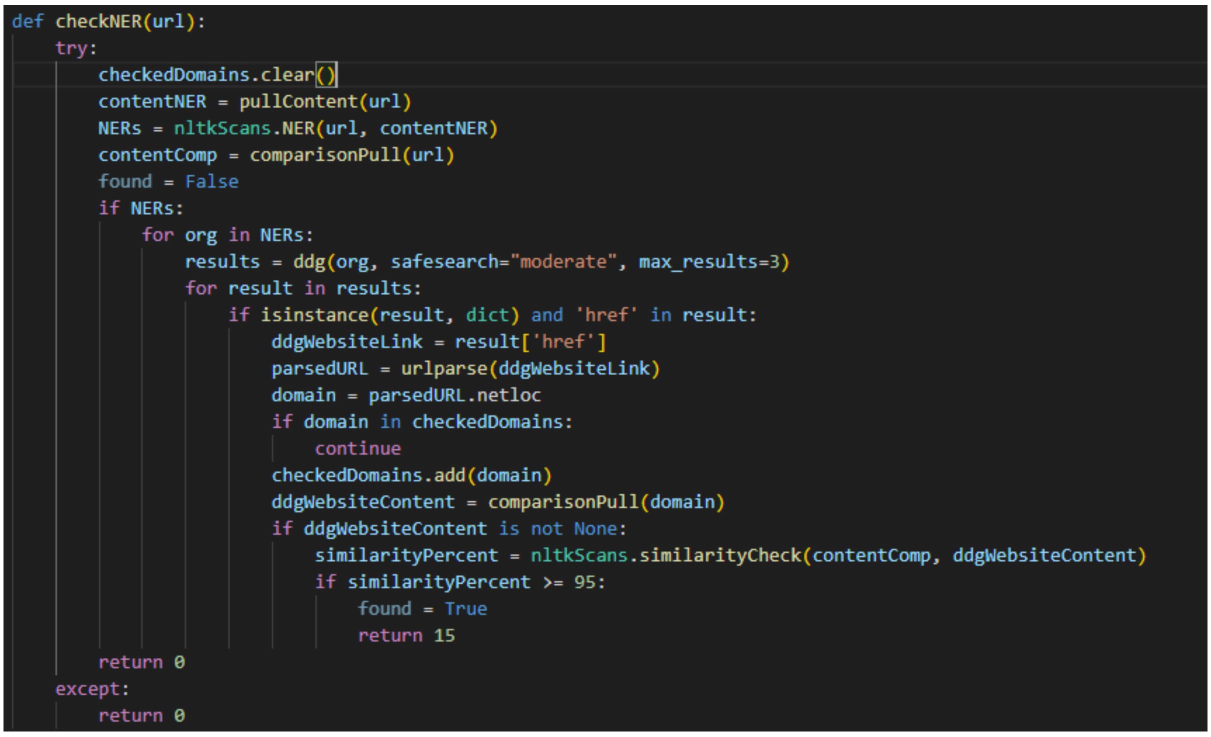
To accomplish the task of checking if the organizations extracted in the above step were unique or if they appeared in multiple webpages, I used the duckduckgo_search python library to perform a search of each webpage found similarity percentage would them be checked using cosine similarity to gain a similarity percentage. If this percentage was greater than 95% it would show that the two websites are the same, results between 90%-95% would be a likely indicator that the website may be a possible clone.



```python
#Using Named Entity Recognition to extract all organizations that are stated on the website
def NER(domain, input):
    organizations = []
    forChunk = domain + input
    chunkFinding = nltk.ne_chunk(nltk.pos_tag(nltk.word_tokenize(forChunk)))
    for nerAlert in chunkFinding:
        if hasattr(nerAlert, 'label') and nerAlert.label() == "ORGANIZATION":
            organizations.append(" ".join([token[0] for token in nerAlert.leaves()]))
    return organizations
```

Figure 3.5: Code to perform Named Entity Recognition on code entered

[6] Anderson, D. (2023). *Word Ninja.* [online] Github. Available at: https://github.com/keredson/wordninja.



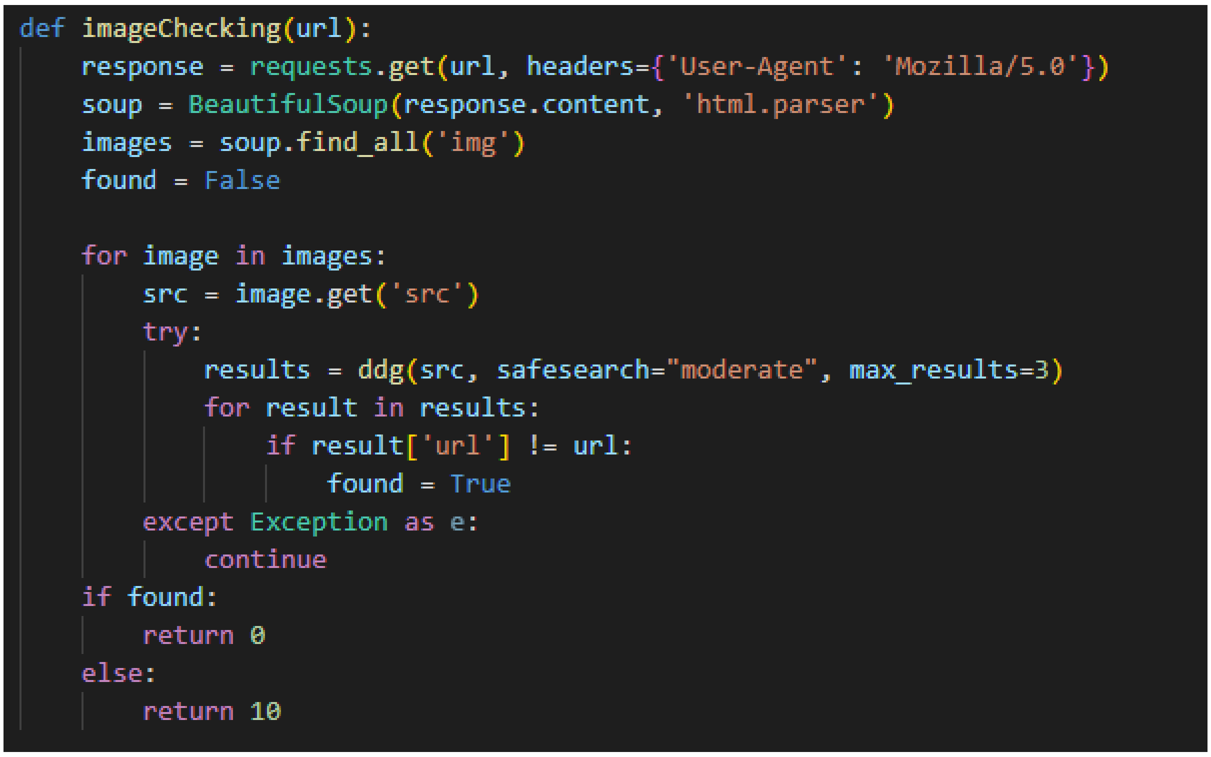
```python
def checkNER(url):
    try:
        checkedDomains.clear()
        contentNER = pullContent(url)
        NERs = nltkScans.NER(url, contentNER)
        contentComp = comparisonPull(url)
        found = False
        if NERs:
            for org in NERs:
                results = ddg(org, safesearch="moderate", max_results=3)
                for result in results:
                    if isinstance(result, dict) and 'href' in result:
                        ddgWebsiteLink = result['href']
                        parsedURL = urlparse(ddgWebsiteLink)
                        domain = parsedURL.netloc
                        if domain in checkedDomains:
                            continue
                        checkedDomains.add(domain)
                        ddgWebsiteContent = comparisonPull(domain)
                        if ddgWebsiteContent is not None:
                            similarityPercent = nltkScans.similarityCheck(contentComp, ddgWebsiteContent)
                            if similarityPercent >= 95:
                                found = True
                                return 15
        return 0
    except:
        return 0
```

Figure 3.6: Code that runs Named Entity Recognition and provides verdict

Initially I used a python library called googlesearch, in order to perform a google search on organizations found within the contents of a website, but this would constantly get hung on requests and lead to an infinite loop where the application could not finish the scan or return a number of errors. I found out this was due to google limiting the number of searches that can be made in a certain period of time. This would cause issues for webpages with a large number of organizations named on the website. To bypass this, I found that using duckduckgo_search to perform web searches allowed me to bypass these limitations, although the time it takes to perform this section of code is currently a weakness of the application, I tried optimising this function where possible and changed to using a more simplistic form of HTML parser to decrease the time it takes to check all organizations. Increasing the speed in which the application parses multiple websites could bring forth issues within the application such as leading to large network congestion.

3.2.4. Extract Images and perform image search to see if they appear on other webpages

Utilizing the BeautifulSoup library in python I am able to extract all images present within a webpage, I can then use duckduckgo_search library in order to perform a search to see if any website other then the website being scan has the image present on their website, if this is the case the overall legitimacy score is decreased as this would indicate that the website may not be an authentic original



```python
def imageChecking(url):
    response = requests.get(url, headers={'User-Agent': 'Mozilla/5.0'})
    soup = BeautifulSoup(response.content, 'html.parser')
    images = soup.find_all('img')
    found = False

    for image in images:
        src = image.get('src')
        try:
            results = ddg(src, safesearch="moderate", max_results=3)
            for result in results:
                if result['url'] != url:
                    found = True
        except Exception as e:
            continue
    if found:
        return 0
    else:
        return 10
```

Figure 3.7: Checks if image is present on webpages other than the original

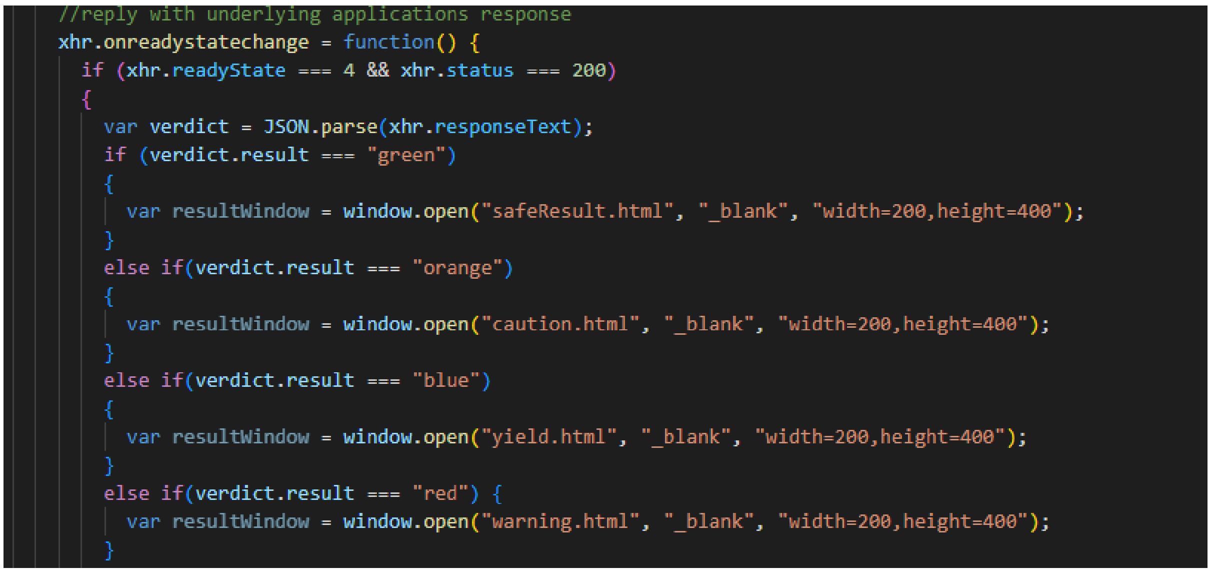
The scans for this task could be made much more in depth by do a reverse image search of an image file itself rather than just doing an image search of the image URL from the page by using Google's own python library but the issue with too many requests being made in a short period of time prevents me from using their library and using duckduckgo_search in its place.

3.2.5. Determine if the website the user is on is legitimate or fake based on search engine results

To conclude legitimacy of a website the user has entered to the program I created a scoring system out of one hundred and each test would add to the legitimacy score of a website. There would be four separate verdicts that could be provided as part of the program.

3.2.6. Warn the user as appropriate

This final mandatory task is performed by creating a popup window that the browser extension would display upon retrieving the results from the underlying application.



```
//reply with underlying applications response
xhr.onreadystatechange = function() {
  if (xhr.readyState === 4 && xhr.status === 200)
  {
    var verdict = JSON.parse(xhr.responseText);
    if (verdict.result === "green")
    {
      var resultWindow = window.open("safeResult.html", "_blank", "width=200,height=400");
    }
    else if(verdict.result === "orange")
    {
      var resultWindow = window.open("caution.html", "_blank", "width=200,height=400");
    }
    else if(verdict.result === "blue")
    {
      var resultWindow = window.open("yield.html", "_blank", "width=200,height=400");
    }
    else if(verdict.result === "red") {
      var resultWindow = window.open("warning.html", "_blank", "width=200,height=400");
    }
```

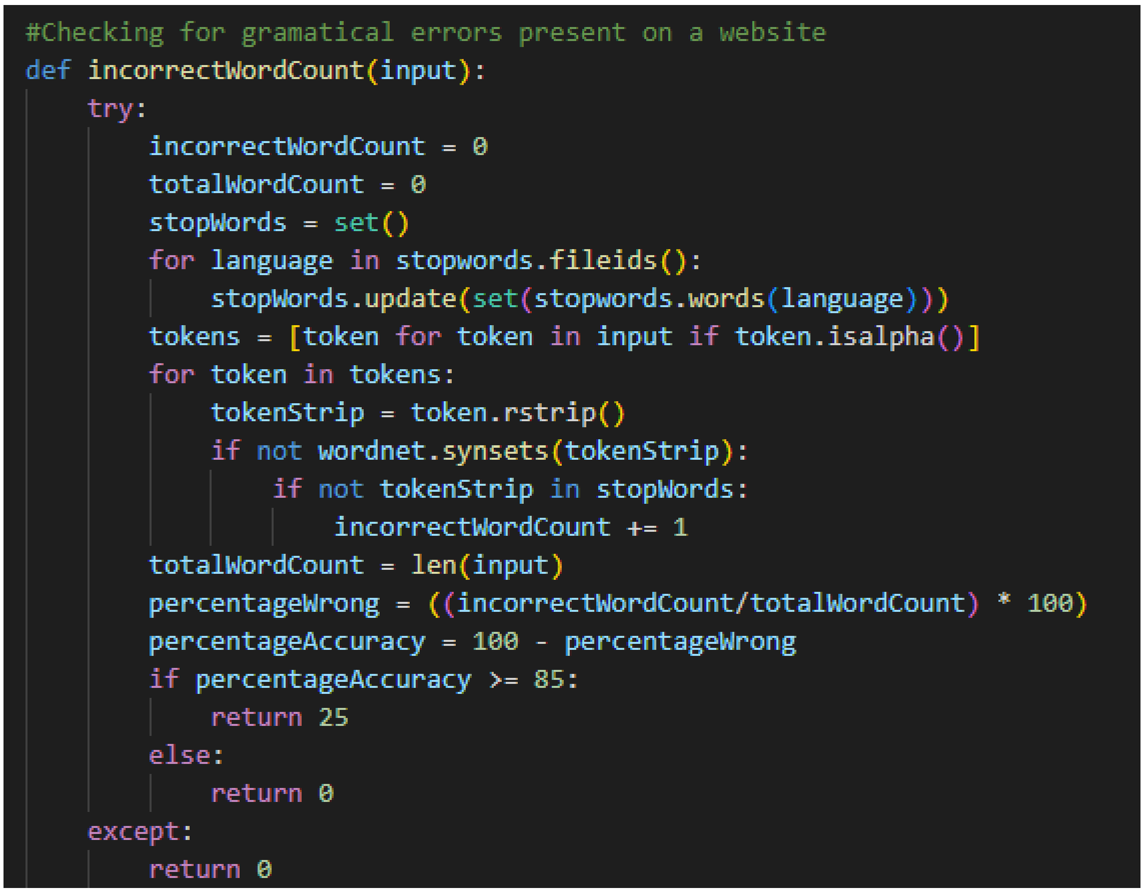
Figure 3.8: Code to popup window for user to see results of scans

3.3. Discretionary Tasks

I was able to accomplish three discretionary tasks. Check words in phrases for spelling mistakes and use the error ratio in the determination of legitimacy of the website, check that WhoIs information is valid for the website being checked and use this information in the determination of legitimacy and check that SSL certificate information is legitimate and suitable for the website the user is currently visiting.

3.3.1. Check words in phrases for spelling mistakes

I performed this task using natural language processing in particular I used Natural Language Toolkit where I checked the websites content against a list of words called stopwords. The stopwords are words commonly used in the language. I increased this list to include words from the languages Arabic, Azerbaijani, Basque, Bengali, Catalan, Chinese, Danish, Dutch, English, Finnish, French, German, Greek, Hebrew, Hinglish, Hungarian, Indonesian, Italian, Kazakh, Nepali, Norwegian, Portuguese, Romanian, Russian, Slovene, Spanish, Swedish, Tajik, and Turkish. I then tokenize the contents of the webpage using the NLTK tokenize function and then check to ensure that each token is present in the list of words in stopwords. I then calculate a percentage of incorrect words from the words present on the website.



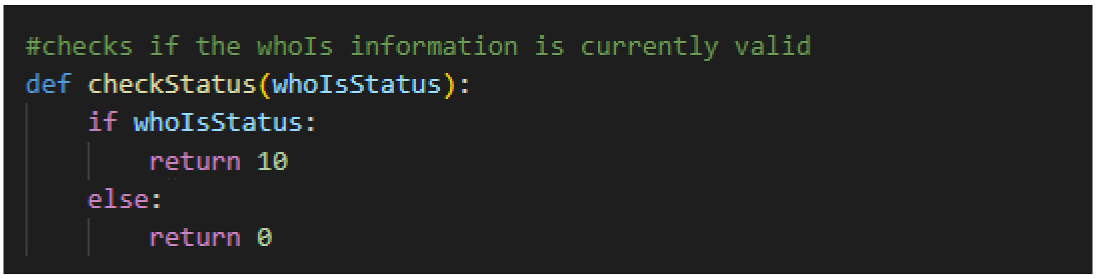
```python
#Checking for gramatical errors present on a website
def incorrectWordCount(input):
    try:
        incorrectWordCount = 0
        totalWordCount = 0
        stopWords = set()
        for language in stopwords.fileids():
            stopWords.update(set(stopwords.words(language)))
        tokens = [token for token in input if token.isalpha()]
        for token in tokens:
            tokenStrip = token.rstrip()
            if not wordnet.synsets(tokenStrip):
                if not tokenStrip in stopWords:
                    incorrectWordCount += 1
        totalWordCount = len(input)
        percentageWrong = ((incorrectWordCount/totalWordCount) * 100)
        percentageAccuracy = 100 - percentageWrong
        if percentageAccuracy >= 85:
            return 25
        else:
            return 0
    except:
        return 0
```

Figure 3.9: Code to perform spellcheck on website's contents

Since websites are capable of being used by individuals in many different countries a lot of websites contain text in multiple languages, which lead to many of the incorrect words being found on websites being a result of multilingual sections of webpages, to fix this issue I added support for multiple languages so words in other languages would no longer lead to a misleading result from the natural language toolkit function.

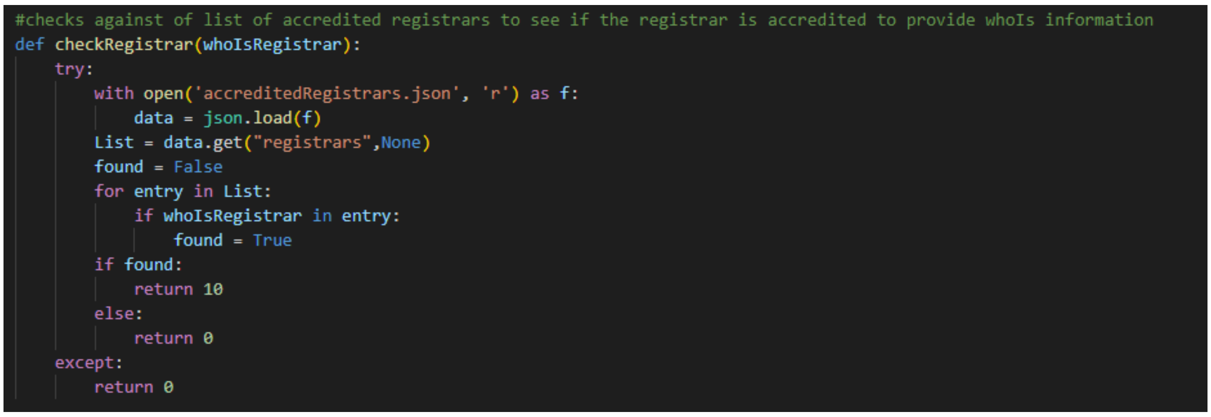
3.3.2. Check the WhoIs Information of the Website

In order to detect whether WhoIs information is legitimate and representative of the website it claims to be for I have enabled the underlying python application to perform three separate tests on the information gathered from the python library whois. There are three tests that are performed on the WhoIs information of a website. The first is to check that the WhoIs status of a website if there is a valid WhoIs status the website. The second test to run on the WhoIs information is to check if the WhoIs information is for the website that we are scanning and ensure that this website is not using the WhoIs information of another website. The third and final test ran on the WhoIs information is to ensure that the registrar associated with the WhoIs information is a reliable registrar and not being provided by an untrusted third party.



```python
#checks if the whoIs information is currently valid
def checkStatus(whoIsStatus):
    if whoIsStatus:
        return 10
    else:
        return 0
```

Figure 3.10: Code to checks WhoIs status is valid



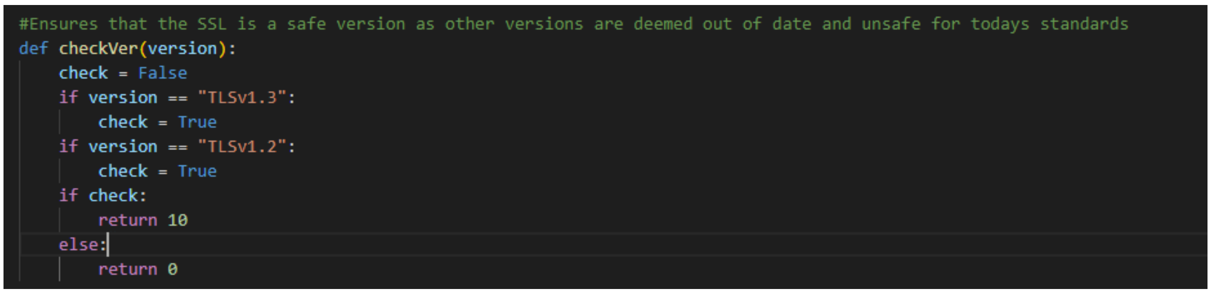
```python
#checks against of list of accredited registrars to see if the registrar is accredited to provide whoIs information
def checkRegistrar(whoIsRegistrar):
    try:
        with open('accreditedRegistrars.json', 'r') as f:
            data = json.load(f)
        List = data.get("registrars",None)
        found = False
        for entry in List:
            if whoIsRegistrar in entry:
                found = True
        if found:
            return 10
        else:
            return 0
    except:
        return 0
```

Figure 3.11: Code to check if WhoIs registrar is a json file containing list of accredited registrars

I had to conduct independent research to find a list of accredited registrars, I eventually decided on combining two separate lists of accredited registrars from https://www.weare.ie/full-list-of-ie-registrars/ and https://www.icann.org/en/accredited-registrars combining them into a json file where the application will compare the registrar found from the WhoIs search and against this list to see if it is present here.

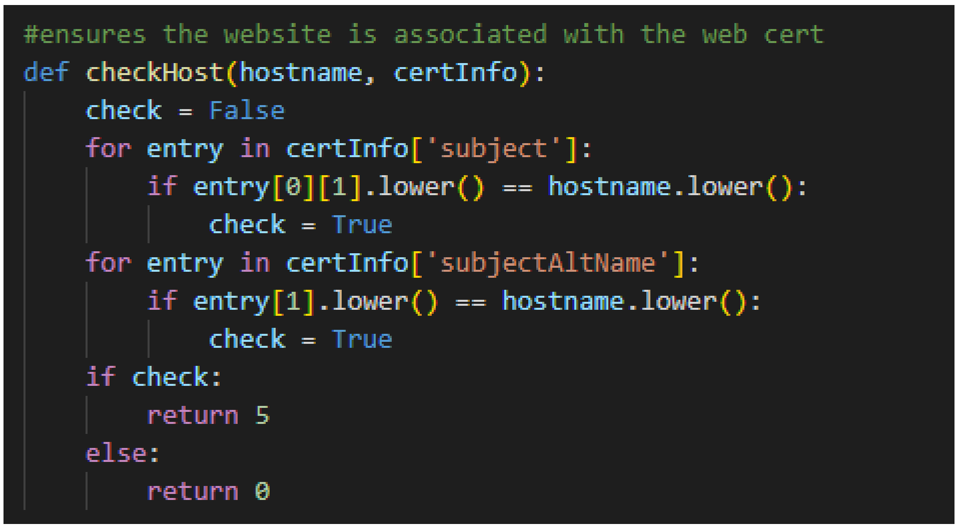
3.3.3. Check the SSL Certificates of the Website

To check for legitimacy of a website ensure that the website's certificates are valid and suitable could be used as another key indicator. In order to test that SSL certificates are legitimate for the website being visited I must create a connection to the website using SSL/TLS to ensure that there are a SSL certs present for the website being visited, if this fails that means that there is no SSL cert for the website meaning it's got a higher probability of being illegitimate. I then perform three tests on the information within the cert to check for legitimacy of the certs. The first of the test is to check what the version of the website cert is as there are only two versions that are currently safe to use for SSL certs as other versions have since been found insecure. The second test performed is to ensure that the SSL test is valid for the current time, to do this it must be checked that the version falls after the notBefore time present on the cert but also that cert is before the notAfter time present on the cert. The third and final tests is to check that the hostname of the website is assigned to the site being checked.



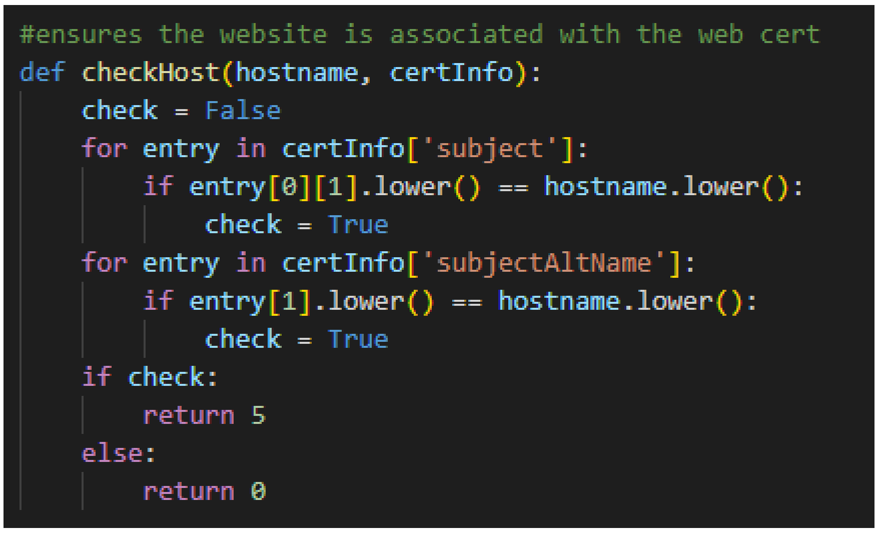
```python
#Ensures that the SSL is a safe version as other versions are deemed out of date and unsafe for todays standards
def checkVer(version):
    check = False
    if version == "TLSv1.3":
        check = True
    if version == "TLSv1.2":
        check = True
    if check:
        return 10
    else:
        return 0
```

Figure 3.12: Code to check SSL version is acceptable



```python
#ensures the website is associated with the web cert
def checkHost(hostname, certInfo):
    check = False
    for entry in certInfo['subject']:
        if entry[0][1].lower() == hostname.lower():
            check = True
    for entry in certInfo['subjectAltName']:
        if entry[1].lower() == hostname.lower():
            check = True
    if check:
        return 5
    else:
        return 0
```

Figure 3.13: Code to check that SSL certificate is valid for current time



```python
#ensures the website is associated with the web cert
def checkHost(hostname, certInfo):
    check = False
    for entry in certInfo['subject']:
        if entry[0][1].lower() == hostname.lower():
            check = True
    for entry in certInfo['subjectAltName']:
        if entry[1].lower() == hostname.lower():
            check = True
    if check:
        return 5
    else:
        return 0
```

Figure 3.14: Code to check hostname appears on SSL certificate

4. Description of Learning

During the development of the project, I was required to learn many new cyber security and application development skills throughout this project. I was also able to advance my knowledge on many subjects that have been touched on during my time in this course.

4.1. Technical Learning

At the beginning of this project, I had zero knowledge of python programming, so I had to learn how to develop code for python application, so I was required to establish an understanding of the capabilities and limitations of programming in python. With the aid of online resources, I was able to build a solid foundation of understanding and was able to create an underlying application that is able to perform multiple functions and utilizes many different python libraries.

I was also required to perform Natural Language Processing as part of this project. I performed research into how Natural Language Processing works on language. With the research I had performed through the duration of this project I was capable to use a python library called Natural Language Toolkit to perform multiple Natural Language Processing Functions such as a grammar check and Named Entity Recognition on pieces of text. I was also able to establish on the grammar check in order to perform a grammar check on texts consisting of multiple languages. This was a large achievement for me as there is much documentation regarding Natural Language Processing, I had to read through in order to gain the understanding required to perform the functions that I have.

In order for Inter-Process communication between the browser extension and the underlying python application I had to learn how to run Flask as part of the python application. This aided in my proficiency with the python language but also gain an understanding of how to develop higher quality web applications than just using languages I was more aware of such as php and JavaScript to create web applications.

To scan for the similarity of two pieces of text I used a system called cosine similarity, this is an advanced mathematical way to compare the similarity of two strings by measuring the cosine of the angle between vectors in multi-dimensional space. Doing research of the mathematics that this function is built on allowed me to advance my mathematical knowledge and gave me an increased understanding of how these algorithms can be put into practice in computer applications.

I had to learn how to create a browser extension during the development of this project and I learned early that a browser extension is built using only three programming languages which are CSS, HTML and JavaScript which meant I had to also advance my limited knowledge of JavaScript in order to perform functions such as sending information to an underlying Flask server.

A large component of running tests on the websites content was to learn how to parse HTML information in order to use the information present on the website within the python application. I learned there were multiple libraries that were capable of performing HTML parsing, but some libraries were better capable than others in certain aspects of HTML parsing. In this project I use HTML parsing to gather the text from webpages. Doing this gave me a better understanding of how sending and receiving information from a web server operates such as a web server uses particular headers to check the legitimacy of a sender.

In order to make the applications developed as part of this project usable for users, I had to develop Graphical User Interfaces for the users to interact with, and clear for users to see what is going on in the application where possible. To do this for the browser extension this was accomplishable using HTML and CSS which allowed me to develop upon my web design skills and I was able to create a GUI for the python application using Tkinter as part of this underlying application I was able to make the status of tests as visible as possible for users by implementing a progress bar that would slowly expand as the underlying systems performed the required tests.

4.2. Personal Learning

For this project I was required to do a great deal of independent learning outside of the classroom in order to gain the knowledge and skills to perform certain tasks as part of the project, I had to sieve through many different educational resources in order to narrow down the material to what was beneficial for my own understanding of specific tools or processes. I was aware when undertaking this project that there was a large amount of research I would have to partake in as part of the development of the application, but quantity of learning that was required did take me by surprise and lead to a large amount of my time being taking in order to perform my independent research.

I was able to gain a deeper understanding of how project management can be put into place in order to make conducting a project a smoother experience especially since the time frame of this project took place over a long time, which is unfamiliar to us as our usual duration for producing a project is much shorter than the timeframe given for this large project. Since learning from this project the importance of time management, I can better understand the importance of planning out the steps required and given times for external factors that would lead to certain aspects of the project taking longer than anticipated for example at the start of this project I found it difficult to balance exams and assignments with the independent work required whilst completing this project, but as time went on I found finding the balance much more achievable.

Throughout this project I was required to learn a large amount of new information in order to perform the tasks required in this project, which allowed the opportunity to advance upon my problem-solving skills when implementing these skills into my project as I had to do much troubleshooting during development with these new tools.

5. Review of Project

During the development process of this project, it was clear to see that creating a tool that would work universally across all websites and be completely accurate was an impossible task, the amount of variation between each website and the sophisticated ways that individuals are able to create unique websites is ever changing. This has brought forth a number of challenges when trying to make a tool that can work with all websites. Reliably testing this tool was itself a difficult challenge, with results occasionally being incorrect due to issues of internet connection and accessing confirmed websites to be illegitimate comes with great security risks due to malware being able to escape virtual machines when connected to the internet hence there are limited tests on actual fraudulent websites present.

I would have liked to increase the speed of which the program is able to conduct its tests especially the section where it's comparing the content of multiple websites, since this section in particular takes a long time to complete. I also wished to add some additional security measures into the application such as encrypting traffic between the browser extension and the underlying python application, but this was difficult due to my limited knowledge of JavaScript, so implementing the algorithms to do so were not possible for me in the time frame. I would have liked to have this browser extension to run automatically unless stated not to when the user visited a new site and the user be notified automatically rather than it being a manual function that the user has to do on each site. I was unable to do this due to the manifest version updating removing the ability to set browser extensions to be persistent.

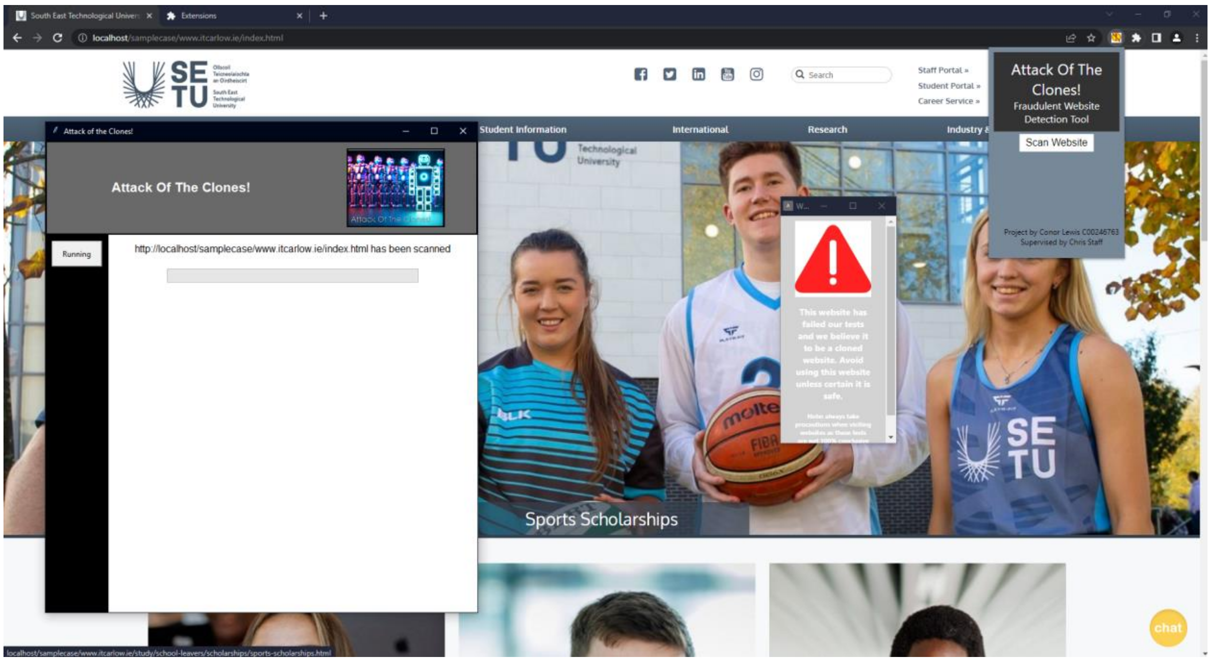
I believe that if this project would have benefited greatly on focusing on creating a scanning tool for a single security scan of a single website rather than making comparisons to other websites instead. This would have enabled greater in dept testing for specific cases and constantly comparing against other websites brings both network congestion and high workload for users especially with the limited capabilities of browser extensions leading to the requirement of both a browser extension and underlying python application which still takes a long time to conduct its tests.

If I was to start this project again from the beginning, I would do more research on an array of tools before finding one that suits my needs and sticking to that one throughout the project. Doing further research on alternative tools earlier would have prevented me from being led astray and trying to mould tools not as well catered to a specific job as another similar library that has better built in tools for particular functions. I would also have made the entire project a standalone application where the user only has to enter a website's URL into the python application and all tests are performed from there, as with the current application flow the user must already make a visit to the site in order to perform test, which is not ideal if the website contains malware on it.

I believe that overall, this project has been a success, although there are areas of the application that I do wish were improved in particular the time it takes in order for the program to provide its verdict, and that I was able to provide more concrete test cases for testing the application against known cloned websites. Here I will provide some tests cases to show that the application is functioning and can identify the difference between illegitimate clone and authentic original website.

The first test case was to involved creating a clone of the website https://www.itcarlow.ie I created this clone using HTTrack[7] and I hosted the cloned version on an xampp server on my host machine, I then performed the tests using the browser extension and underlying python application to check if the program could spot which was the authentic original and which one was an illegitimate clone. We can see below (Fig. 5.1) that the browser extension and the underlying python application were able to identify which website was the authentic and which was a clone as it can be seen in (Fig. 5.2) that the website passed the majority of tests.

[7] Httrack.com. (2017). *HTTrack Website Copier - Free Software Offline Browser (GNU GPL)*. [online] Available at: https://www.httrack.com/.

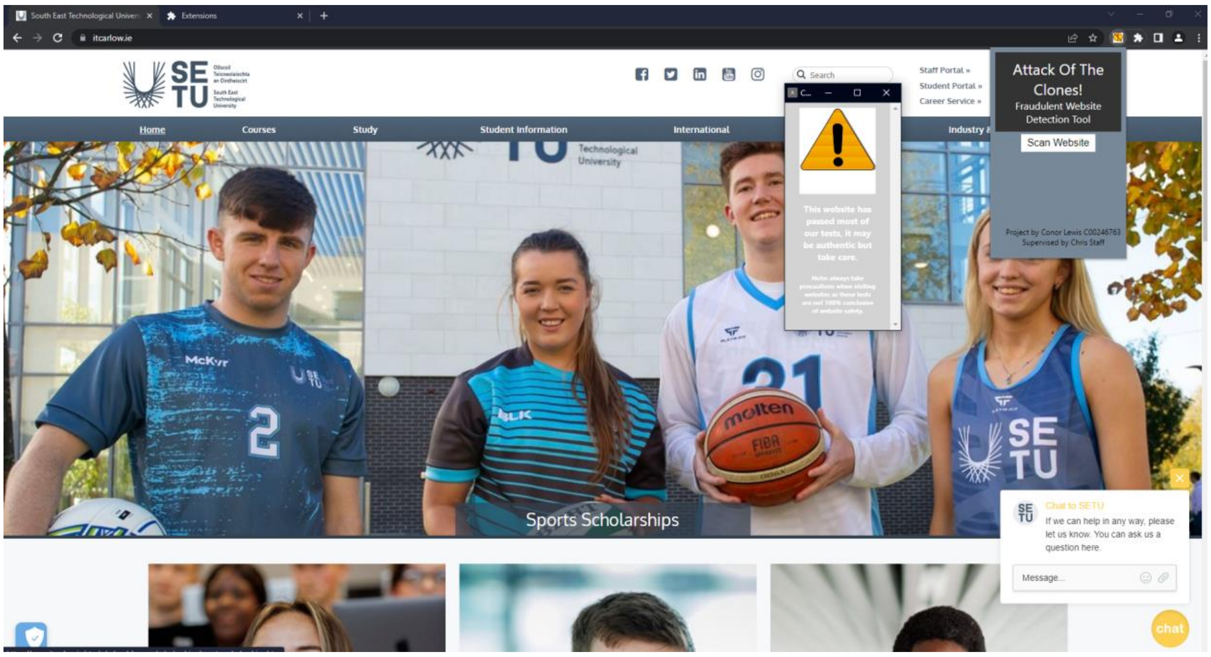

Attack Of The Clones!
Running
Attack Of The Clones! Fraudulent Website Detection Tool
Scan Website
Project by Conor Lewis C00246763 Supervised by Chris Staff

http://localhost/samplecase/www.itcarlow.ie/index.html has been scanned

This website has failed our tests and we believe it to be a cloned website. Avoid using this website unless certain it is safe.

Note: always take precautions when visiting websites as these tests are not 100% conclusive

Figure 5.1: Code to check hostname appears on SSL certificate

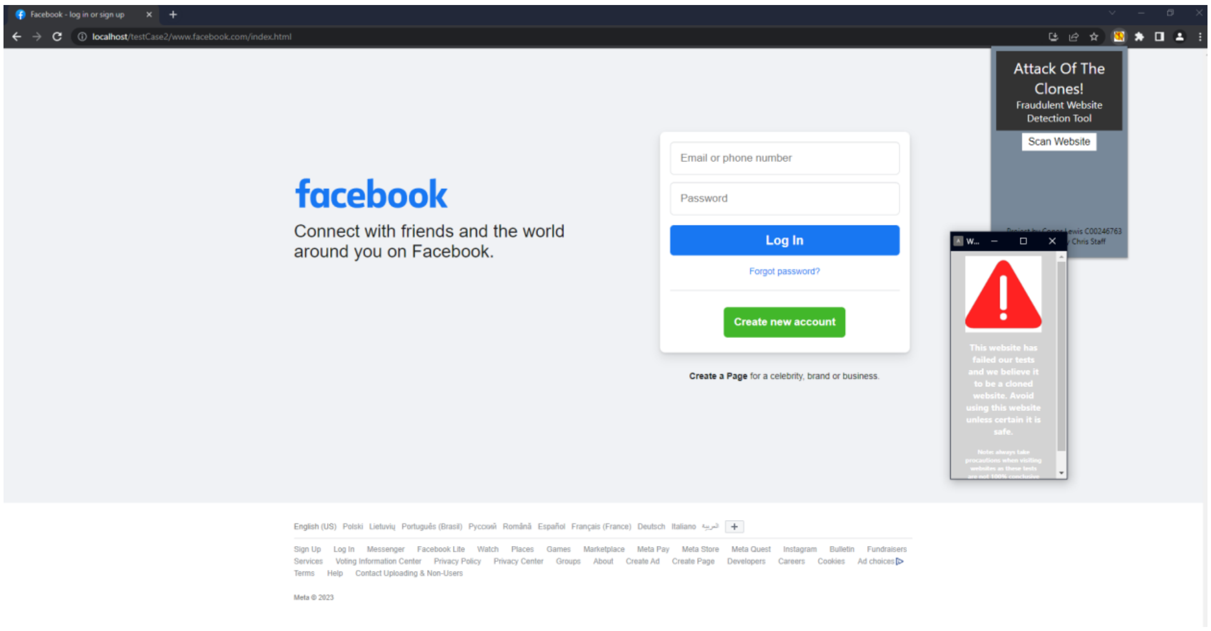

Attack Of The Clones! Fraudulent Website Detection Tool
Scan Website
Project by Conor Lewis C00246763 Supervised by Chris Staff

This website has passed most of our tests, it may be authentic but take care.

Note: always take precautions when visiting websites as these tests are not 100% conclusive of website safety.

Figure 5.2: Code to check hostname appears on SSL certificate

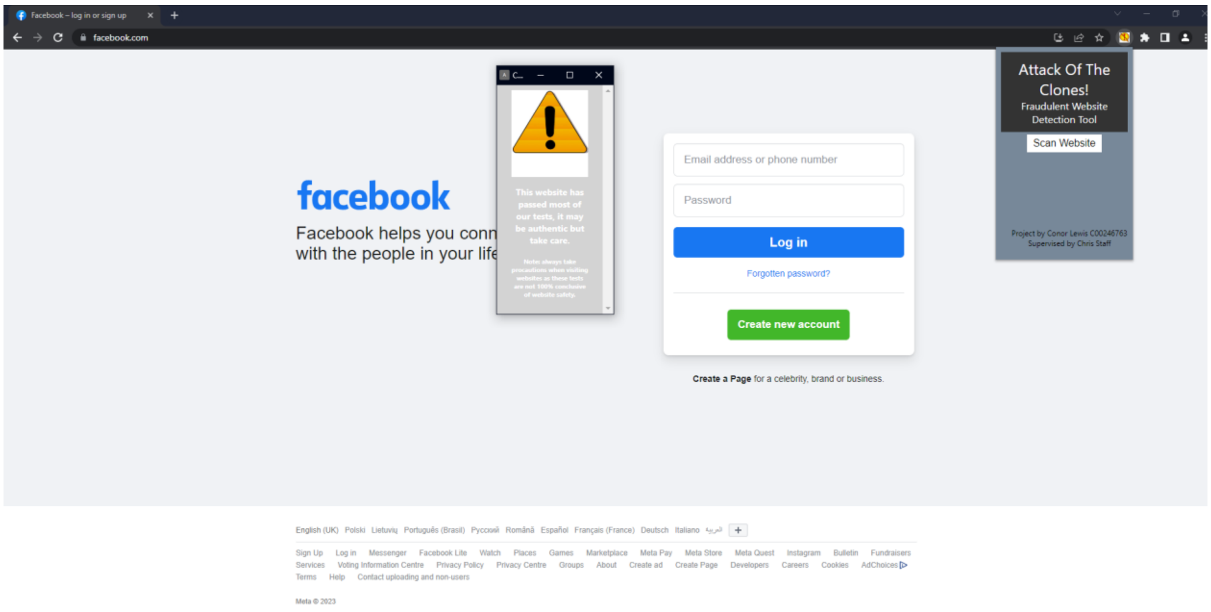
The second test case I performed to verify that the program has the capability to identify which websites are authentic originals and which was are cloned I used the same tools as the previous test in order to clone the website and host the website, in this test we can see that the application was able to alert which one was fraudulent and which was the authentic original by the windows that were displayed (Fig. 5.3, Fig. 5.4).


Facebook - log in or sign up
localhost/testCase2/www.facebook.com/index.html
facebook
Connect with friends and the world around you on Facebook.
Email or phone number
Password
Log In
Forgot password?
Create new account
Create a Page for a celebrity, brand or business.
Attack Of The Clones!
Fraudulent Website Detection Tool
Scan Website
Project by Conor Lewis C00246763
Supervised by Chris Staff
Meta © 2023

This website has failed our tests and we believe it to be a cloned website. Avoid using this website unless certain it is safe.

Note: always take procautions when visiting websites as these tests are not 100% conclusive

Figure 5.3: Code to check hostname appears on SSL certificate


Facebook – log in or sign up
facebook.com
facebook
Facebook helps you conn
with the people in your life
Email address or phone number
Password
Log in
Forgotten password?
Create new account
Create a Page for a celebrity, brand or business.
Attack Of The Clones!
Fraudulent Website Detection Tool
Scan Website
Project by Conor Lewis C00246763
Supervised by Chris Staff
Meta © 2023

This website has passed most of our tests, it may be authentic but take care.

Note: always take procautions when visiting websites as these tests are not 100% conclusive of website safety.

Figure 5.4: Code to check hostname appears on SSL certificate

6. Acknowledgements

I would like to thank my project supervisor Chris Staff for offering me guidance and support throughout the project. I would also like all other lecturers that also offered guidance during this project

7. Declaration of Plagiarism

I declare, this document in this submission in its entirety is my own work except for where duty acknowledged. I have cited the sources of all the quotations, paragraphs, summaries of information, tables, diagrams, or other material. This includes software and other electronic media that is integral property rights may reside. I have provided the complete biography at the end of my document detailing all the works and resources used in the presentation of this submission. I am aware that failure to comply with the Institute's regulations governing plagiarism constitutes a serious offense.

Conor Lewis	17/04/2023
______________________________	__________________
Signature	Date