Internship Report for April - May 2018

Yash Saboo

2015-2019 Batch

T.Y.C.S.

15070121170

**MORE DETAILS THE PROJECT**

**Problem Statement:**

To develop a Google Chrome Web Extension which can detect phishing/imitation websites and alert users about the same based on previously browsed websites.

**Technology Used and Technical Details:**

In the month of April, many changes were done in the project. The “Layer 2” which was described in the previous report was named as “Layer 3”, because “Layer 1” was distributed into two different layers, that is “Layer 1” and “Layer 2”. I worked only on the client-side programming this month. The client-side approach was advanced to Machine Learning domain which led to many technical changes.

Following technologies were used (Updated Layer Names used):

**Client – Server Model:**

1. JavaScript for implementing the extension in Chrome.
2. Python to implement the algorithms
3. Flask, a web-framework, to host python scripts
4. REST APIs for connecting Client (Chrome Extension) to Server (Flask)
5. jQuery AJAX for using POST method of REST
6. Open\_SSL software to create certificates to enable HTTPS connections on server.

**From Layer 1 to Layer 8:**

1. Python to implement the algorithms
2. Flask, a web-framework, to host python scripts
3. Jupyter Notebooks for easy debugging of Python scripts.

These technologies were learnt and implemented, as when required, using various resources available on Internet. I enrolled in an online course on Cognitive Class called "Machine Learning with Python".

Certificate Link: <https://courses.cognitiveclass.ai/certificates/2eae01eeb0524b2c941dfe89817bc7c6>

**TASKS DONE**

**Task 1: Layer Formation**

**Layer 1 and Layer 2:** Image Processing Algorithms. (February - March)

**Layer 3:** Web Scraping (March - April)

*Note: Before the machine learning was introduced in the project, Layer 3’s (previously, Layer 2) implementation, testing and benchmarking was completed.*

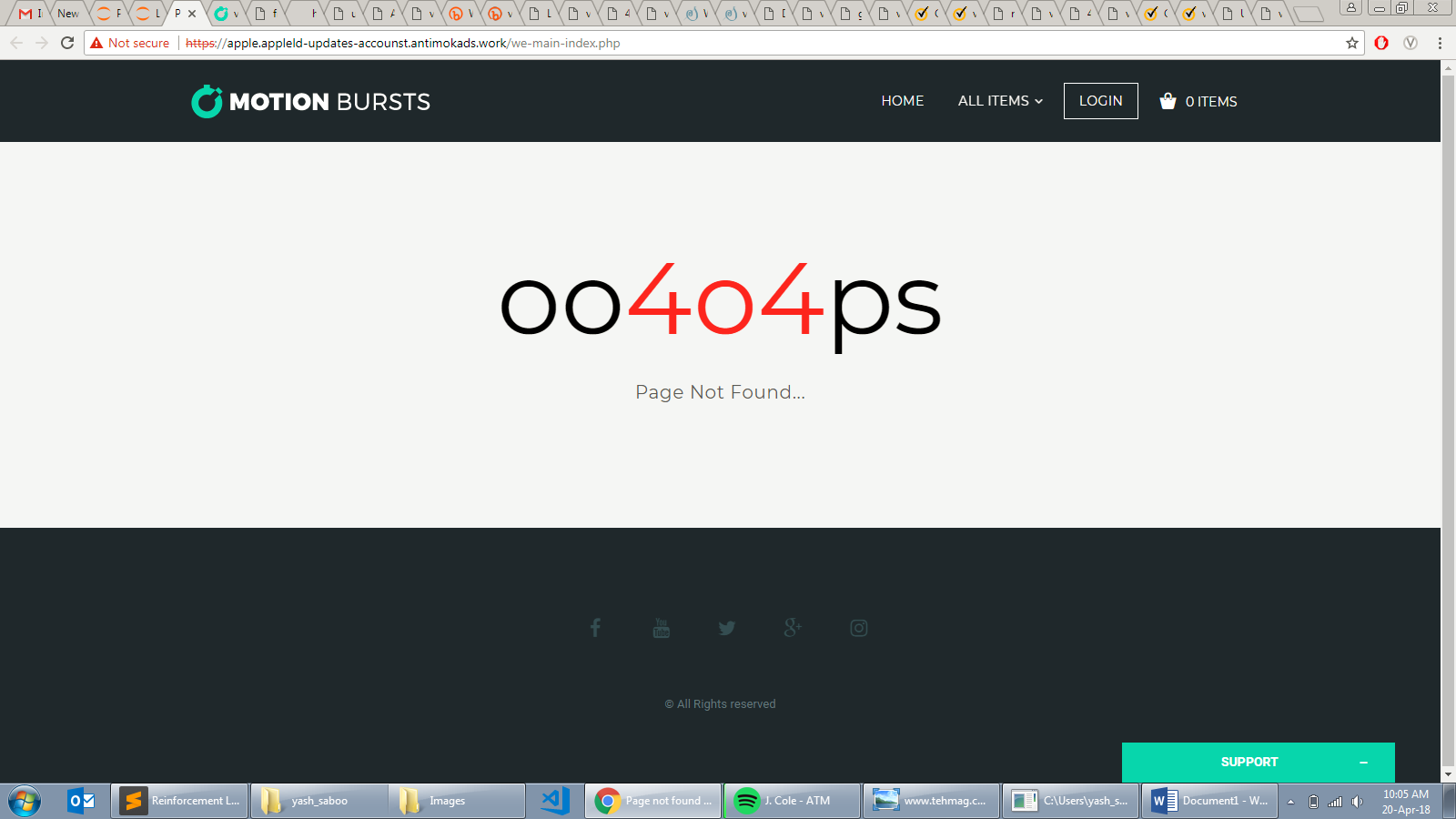
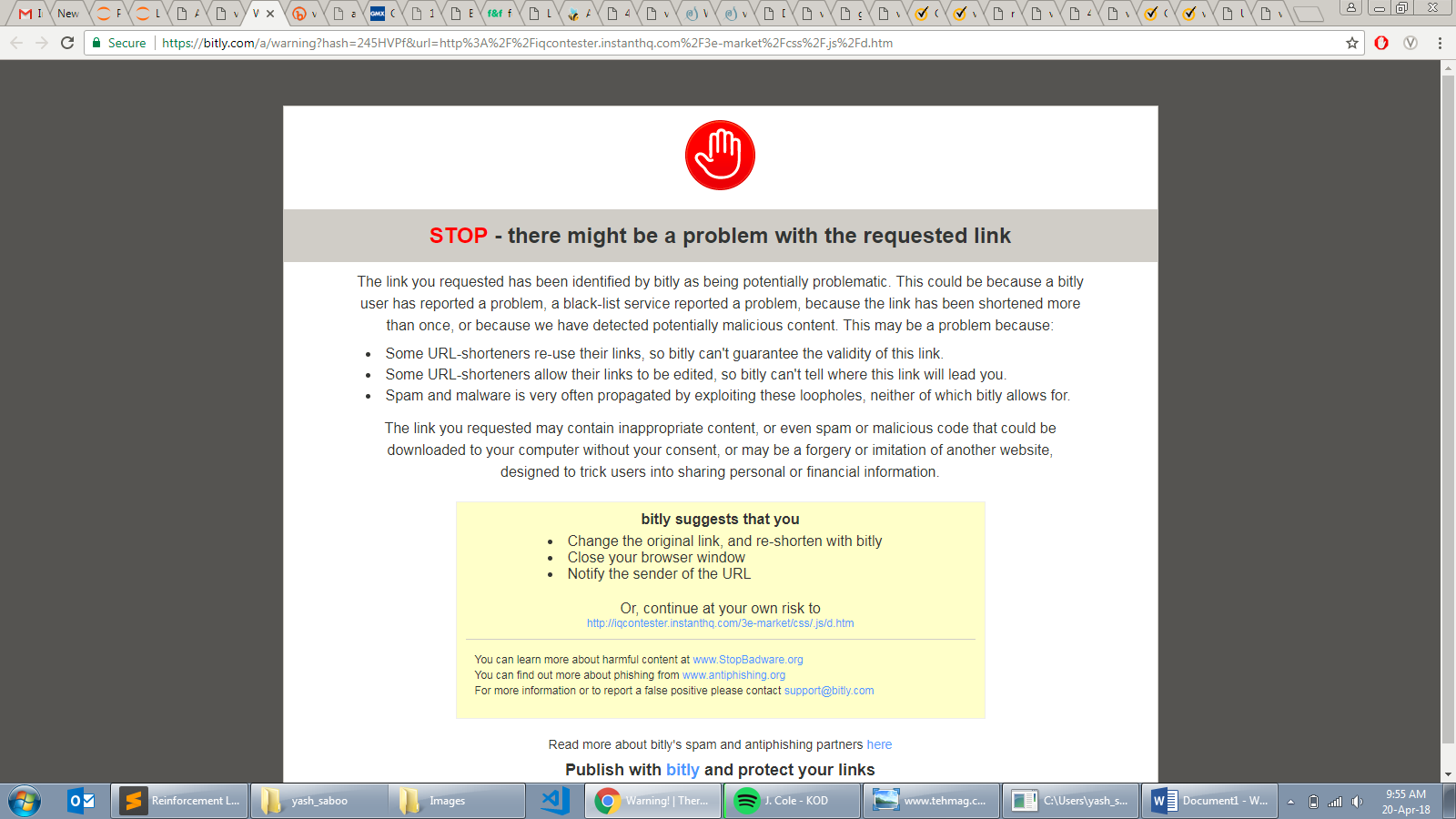
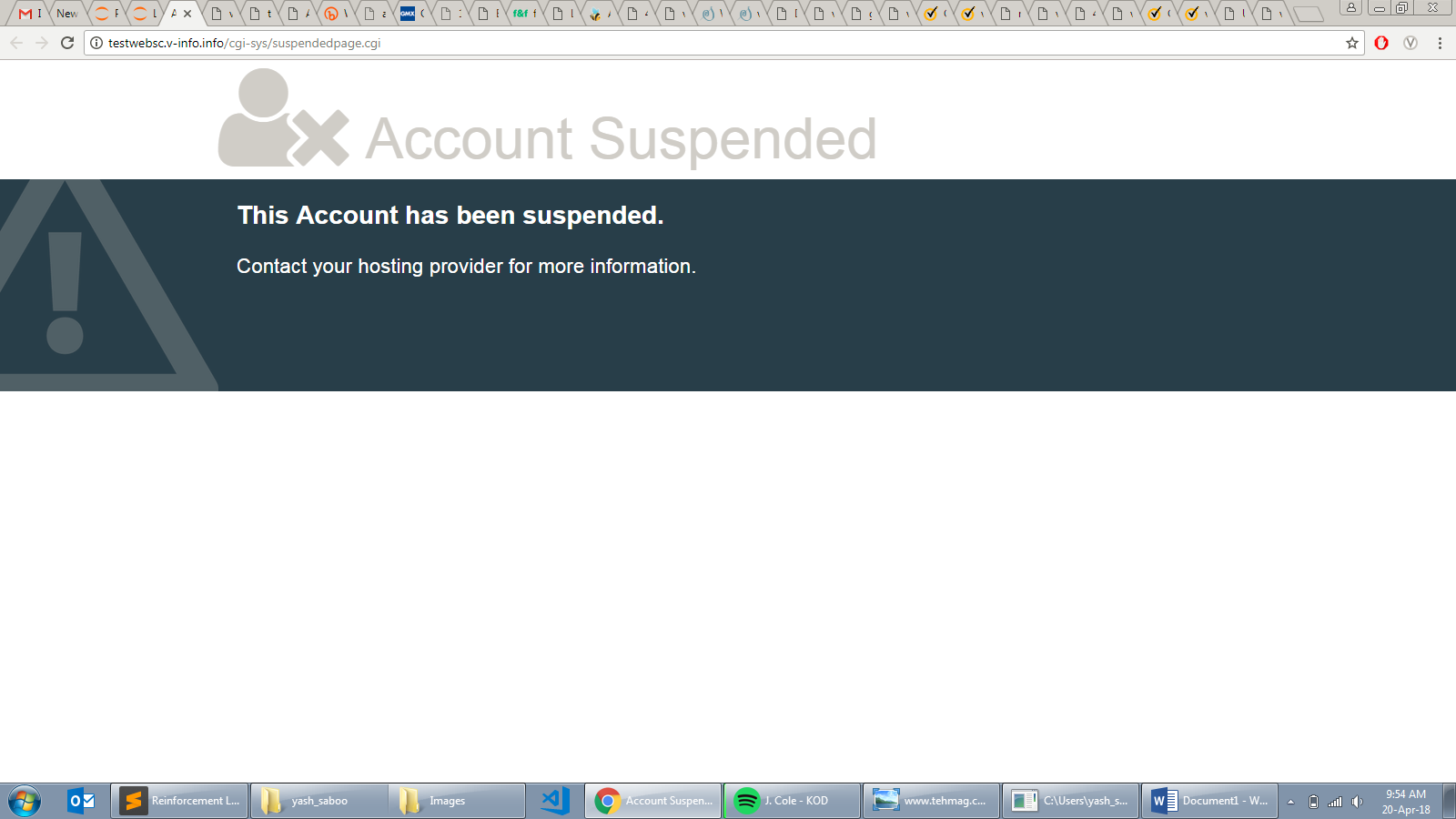
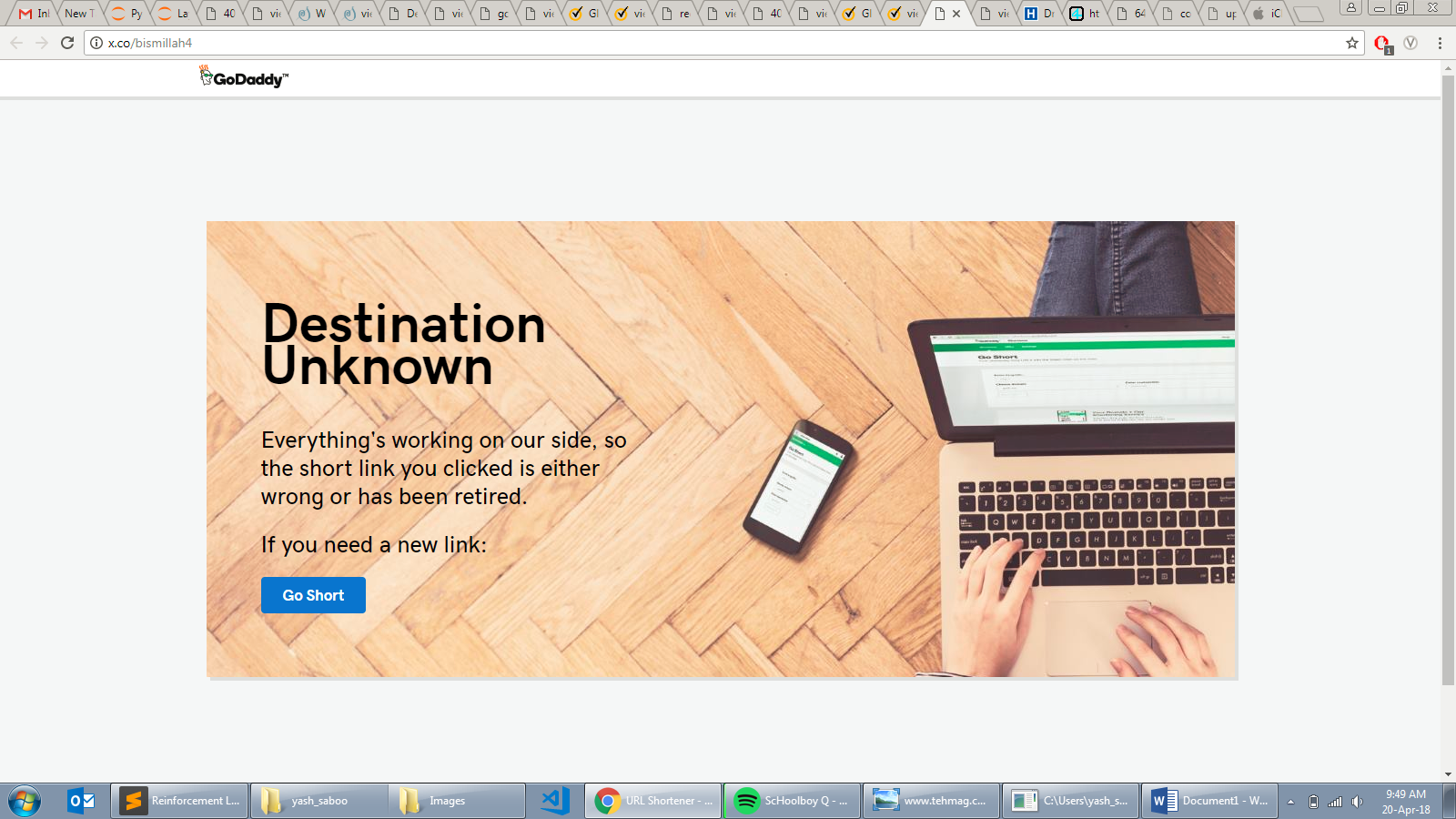
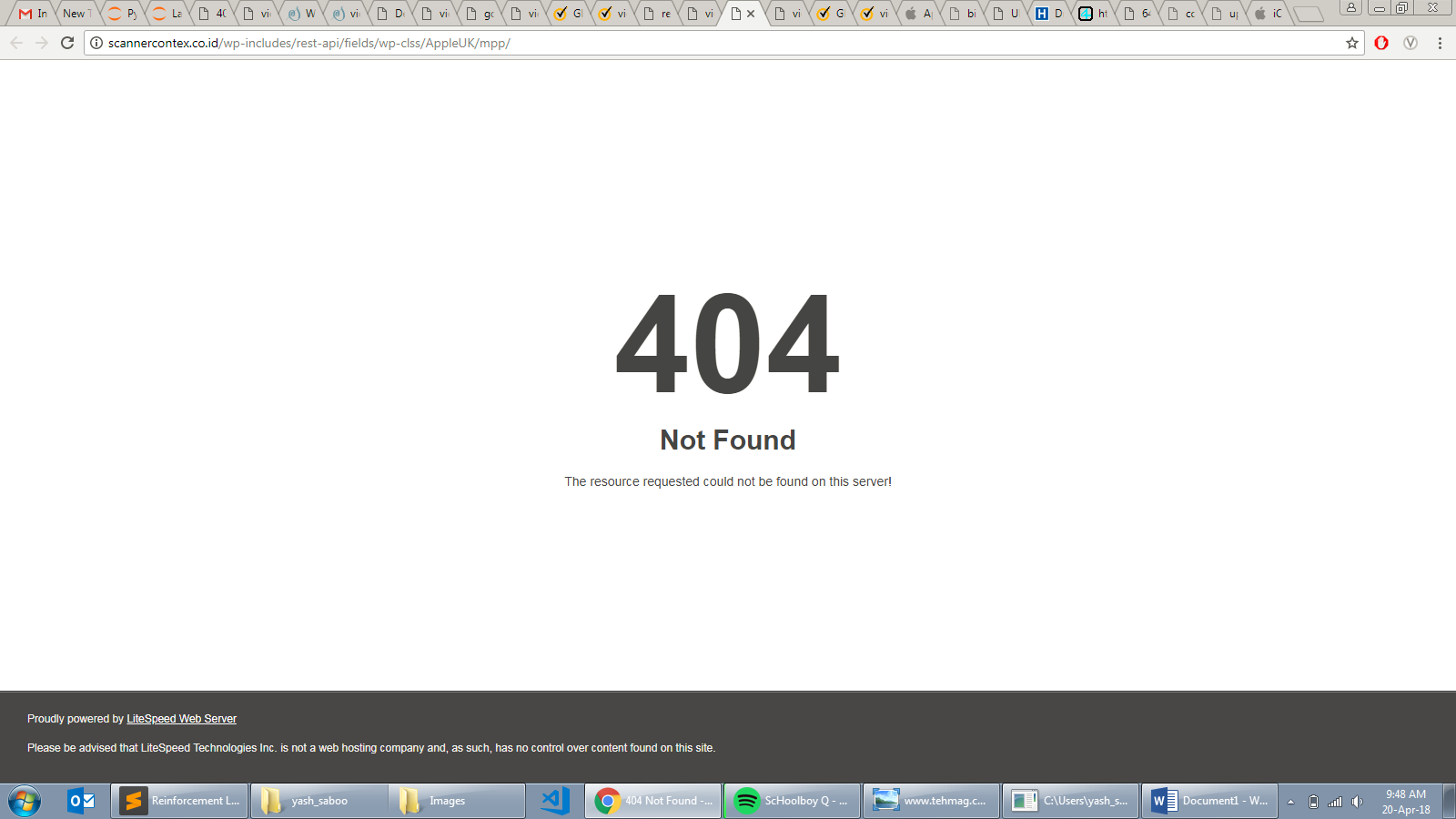
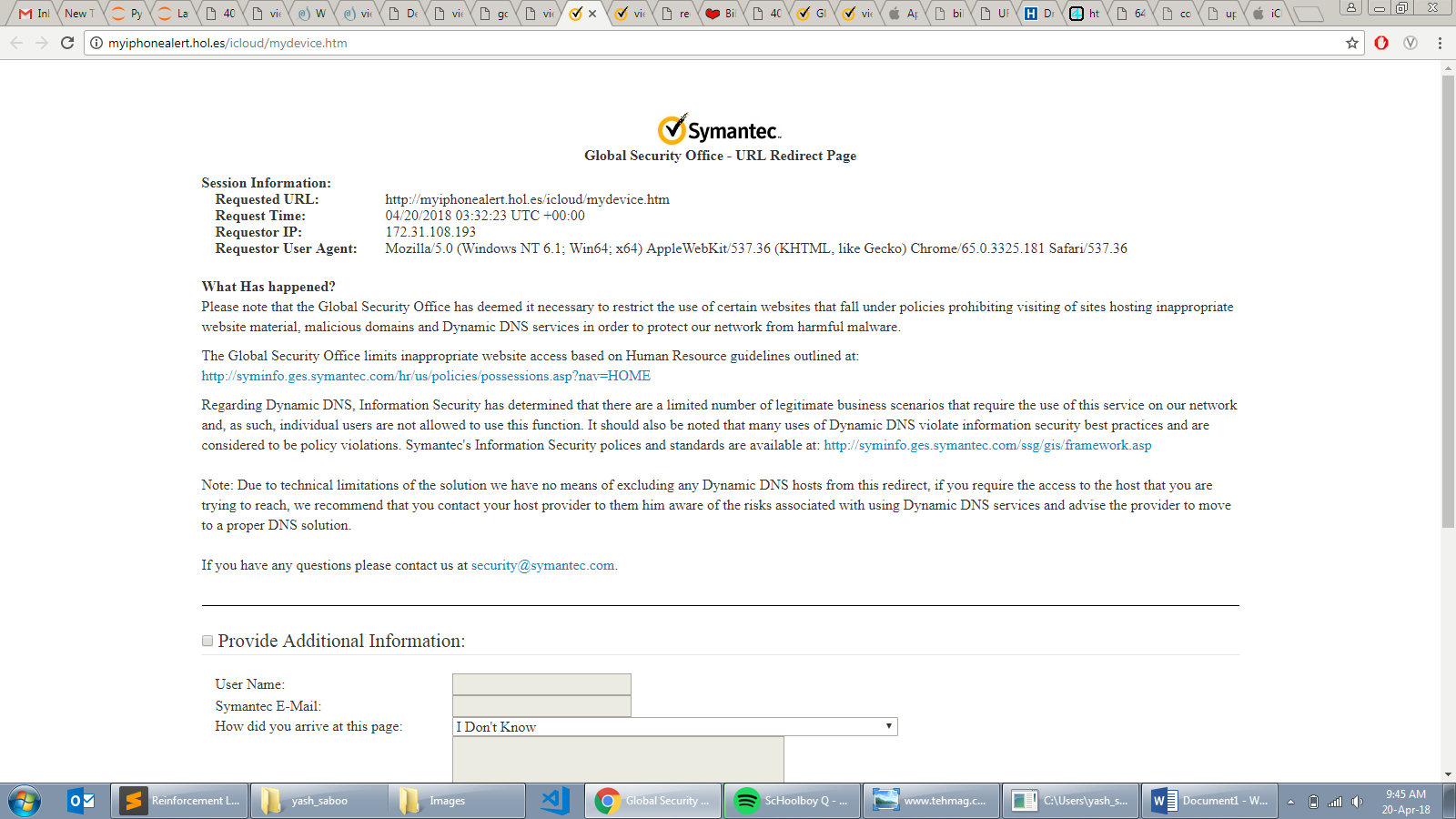
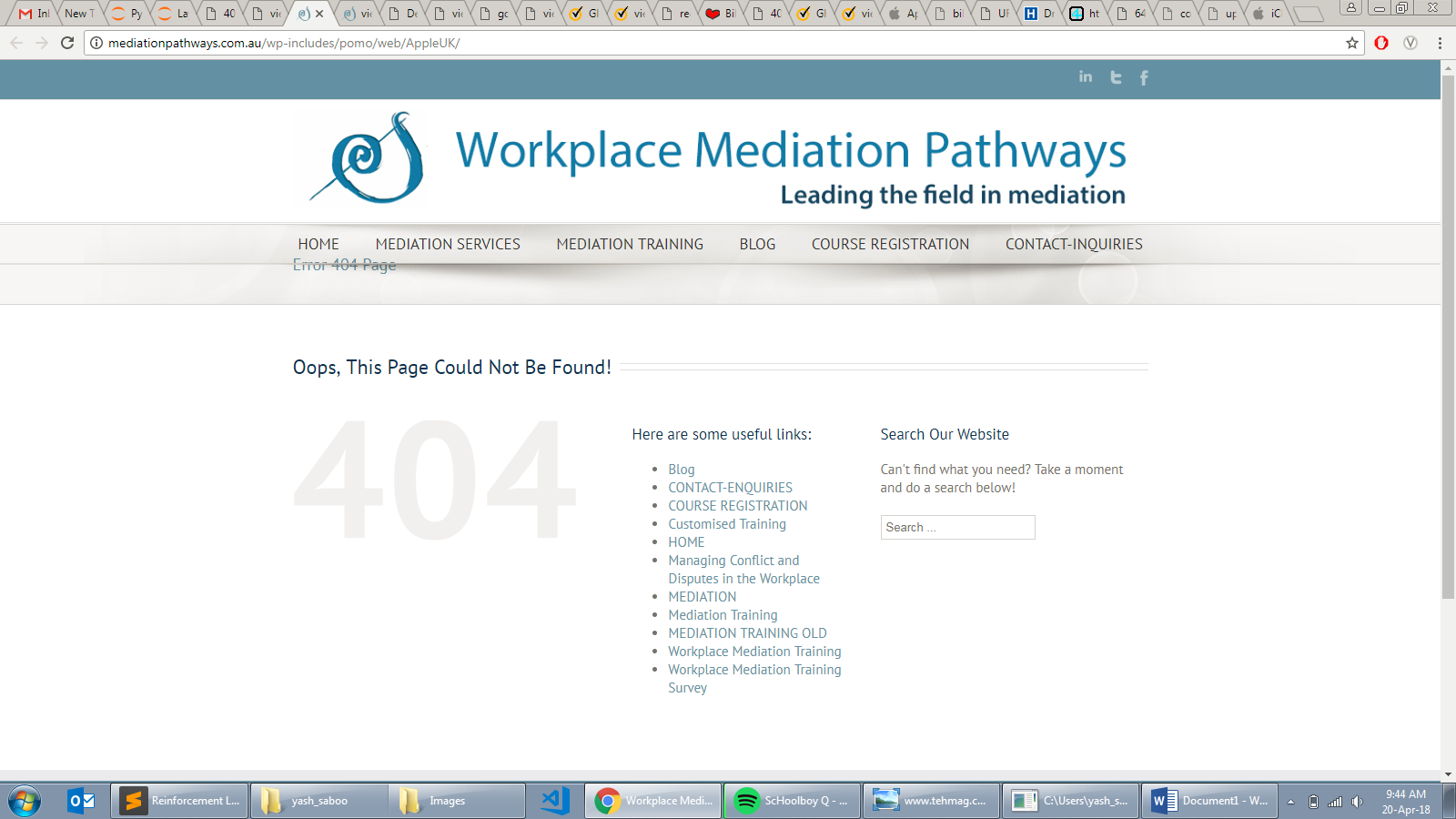
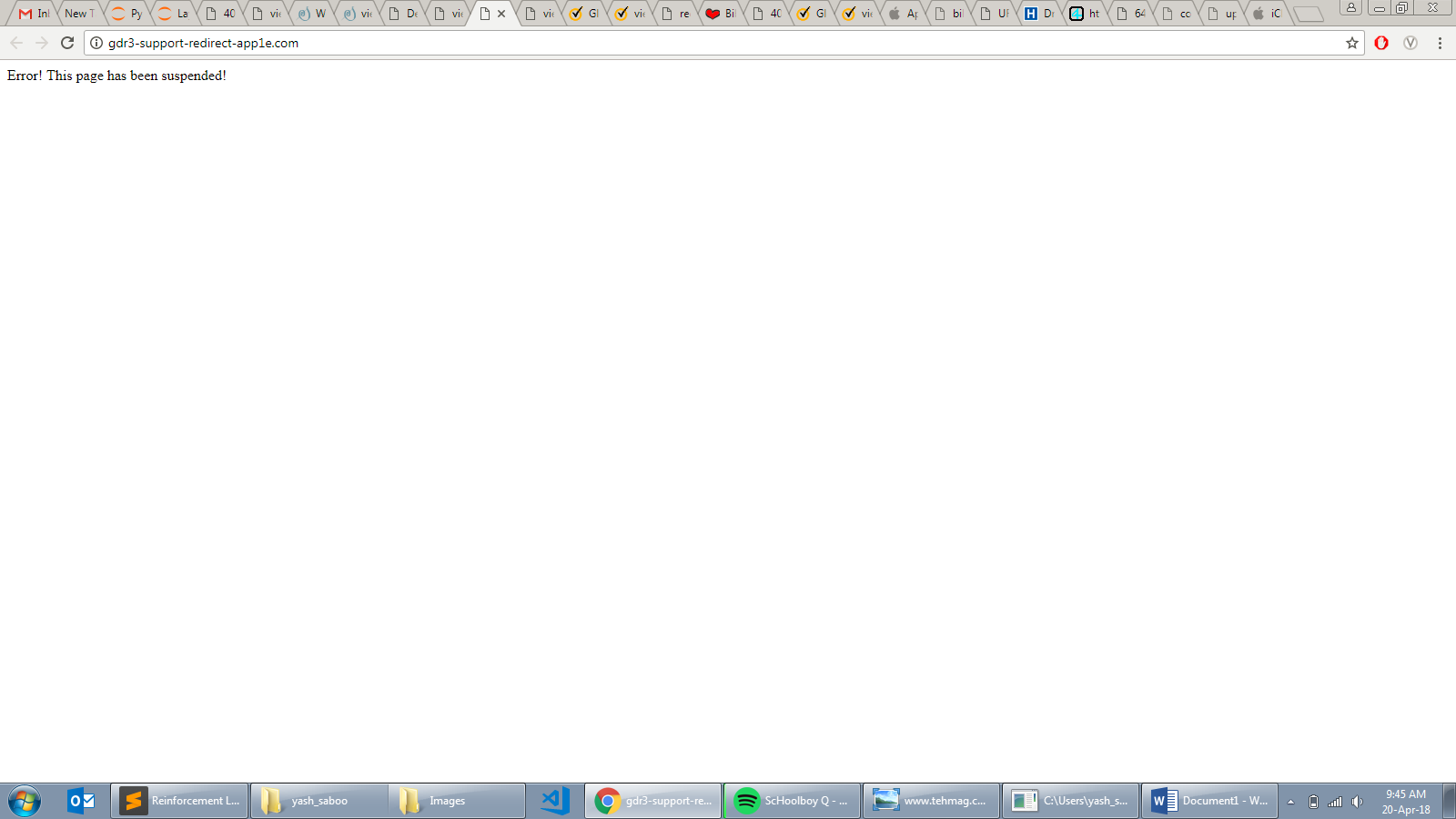
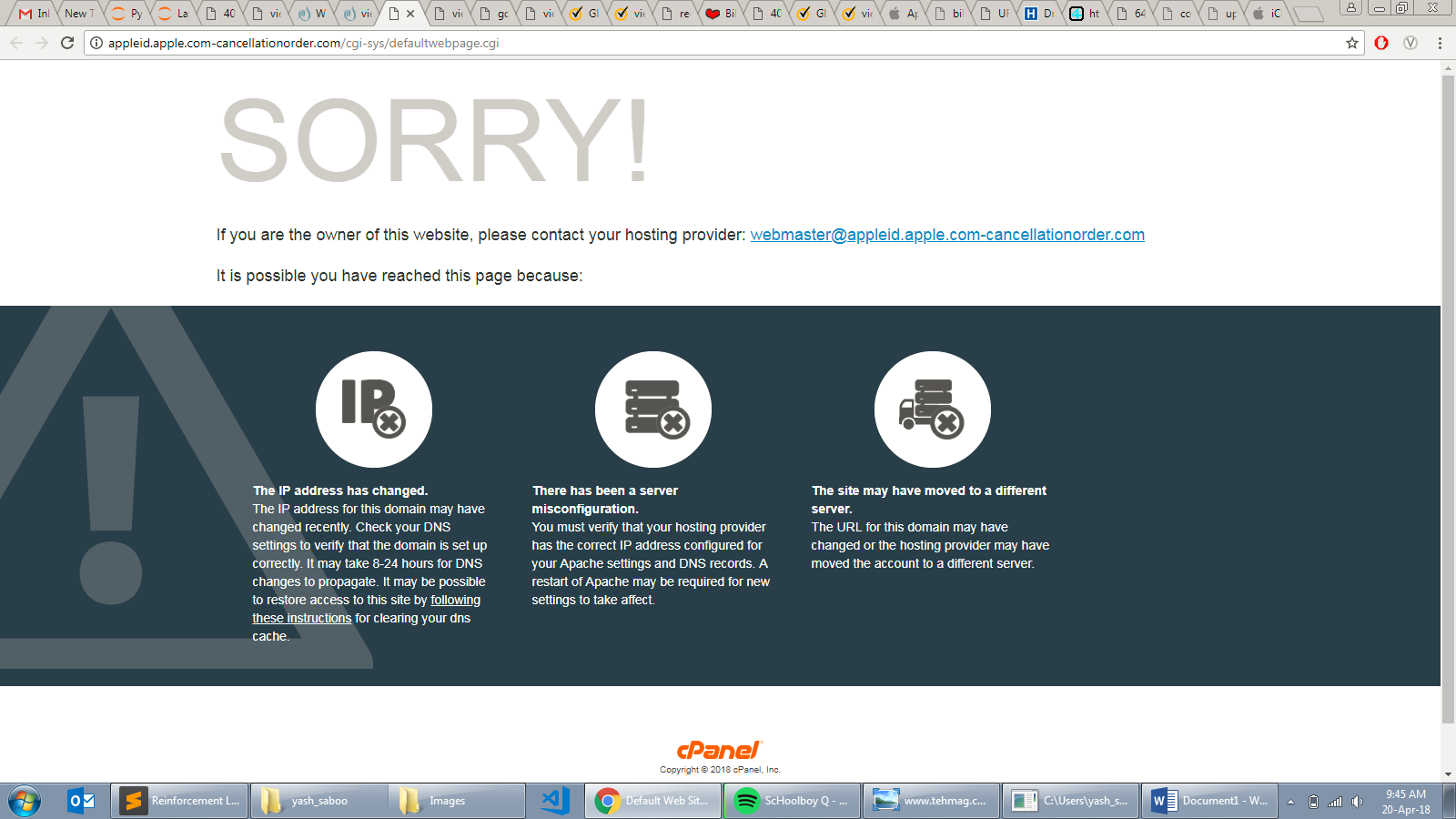
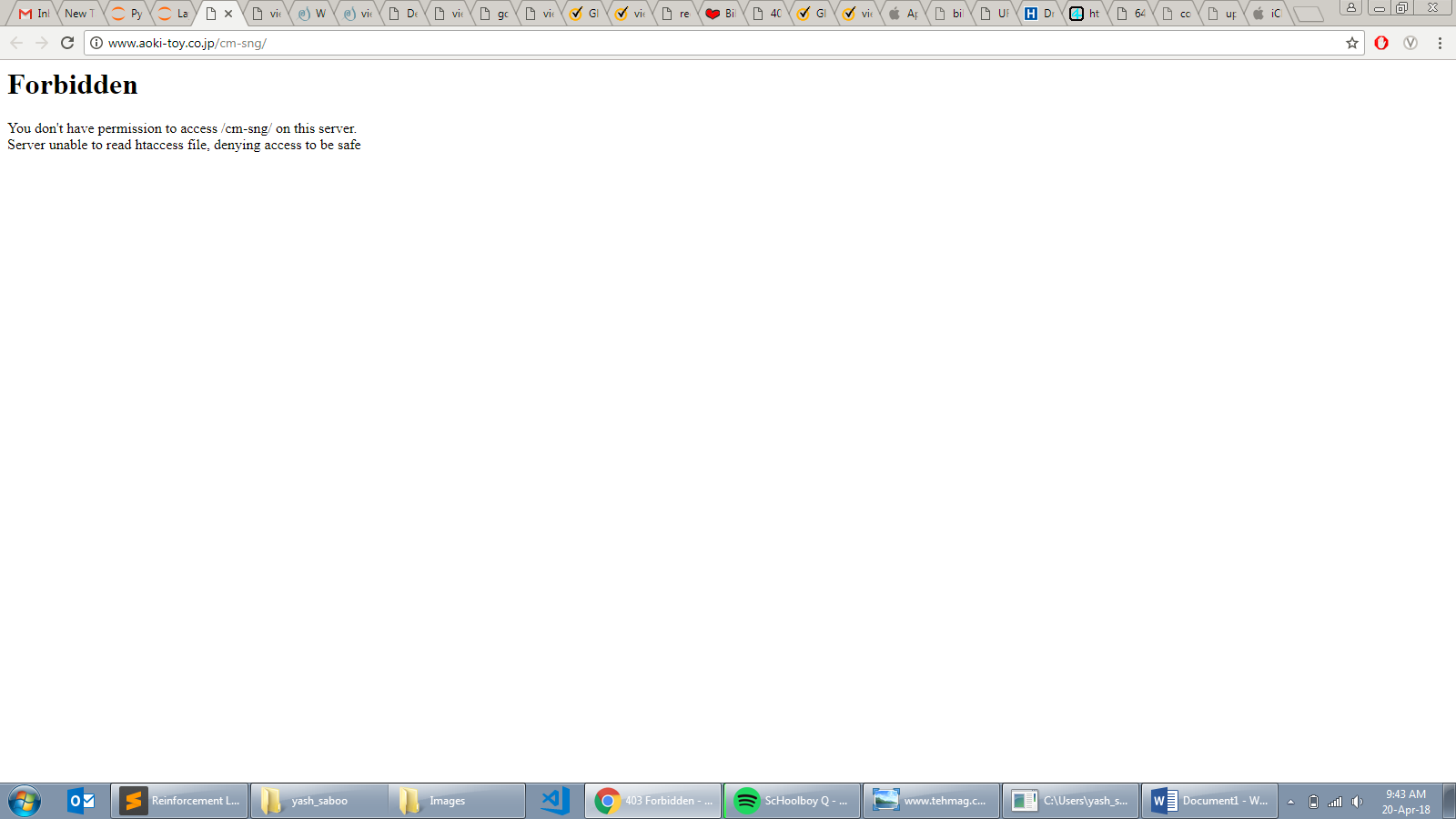
* **Choosing Algorithm and Technique** (March)
* **Data Selection** (March)
* **Data Preprocessing** (March)
* **Development** (March)
  + The library mentioned in the last report couldn’t produce satisfactory results so various other libraries were discovered to perform the same functionality, in the hope that they would give better results.
  + The main task for this layer was finding the correlation between two texts. We did find many libraries but none of the libraries were giving the expected result. There were few tweaks made here and there in the functionalities of the library usage, but in vain.
* **Testing** (April)
  + The algorithm was tested rigorously, both programmatically and manually.
  + As the web scraping depends upon the underlying code written for the legitimate and phishing website, the correlation between them was hard to figure out because of the following problems:
    - same website can be written in many different combination of scripting languages
    - rearrangement of the modules doesn’t change the website functionalities
    - naming problem, that is the same Image can be saved with a different name.
  + The above problems were known even before implementing it, but their extent was not. So optimistically, it was assumed that websites which looked similar would have at least some co-relation between their underlying coding pattern despite the above problems, but that was not the case.
* **Benchmarking** (April)
  + The process followed for benchmarking in Layer 1 was applied here as well.
  + We deduced that the assumptions that were made had some flaws and thus, couldn’t provide the expected results.

**Layer 4 – 8:** The overall details of the algorithms used are bound by a confidentiality clause but it basically involved different features of a website that can uniquely identify itself, and help in differentiating between a phishing and legitimate website. These features were found by reading various research papers and blogs and brainstorming with the team.

**Task 2: Feature Extraction for Machine Learning Model**

* There was no online Phishing database available which could serve our purpose for ML model, so I had to write a script which could extract the desired feature - data for both Phishing and Legitimate websites.
* The Phishing websites’ URL was extracted from Phishtank and the Legitimate websites’ URL was extracted from Alexa, web analytics service.
* As huge amount of data is required to train a Machine Learning model, the feature extraction script should be completely automated and have a capacity to run for days, without intervention, along with solving all the errors and exceptions that occur on the fly.
* The complete automation of feature extraction was one of the most challenging tasks I have performed. The following are few challenges I faced:
  + *Infinite Time Problem*: Sometimes a feature would be present for a website, but it would take infinite time to extract it. This leads to improper stoppage which doesn’t even raise an error explicitly. So, I created a decorator in Python called “timeout(numberOfseconds)” which would raise “TimeoutError” using SIGNAL function present in UNIX systems. (Thus, this script only works on UNIX – based systems)
  + *Data Corruption*: Some websites produce following errors which corrupts the data:
    - Throws 403 Error: Forbidden
    - Throws 404 Error: Website Not Found
    - Redirects to some other site which defeats the purpose of getting data of the original site altogether
    - As phishing sites are short-lived so the content for which it was categorized as Phishing has been replaced by some genuine content.
    - Account Suspended of the website Domain
    - When the URL link received is not available anymore, usually happens with shortened URL.
    - When the website is “Sleeping”, usually happens with the websites hosted on some hosting service websites.
    - The website blocked by Symantec Network

So, I research a lot about such sites and found a pattern in these sites which would repeat. There were many patterns present so writing a single function would not have worked. Thus, I had to program a complete new script which would identify such websites and filter them out. Few screenshots of such websites are as follows:



* + *System Overload*: Poor coding leads to system overloading. The following are the cases which contributed to the same:
    - The file opened was never closed which would create overloading of the system.
    - The “soup” built using Beautiful Soup library was not “decomposed”.
    - The “driver” used by Selenium was not “disposed”.
    - The “windows” opened by CV2 was not closed.

This really helped in improving my coding and inclined me towards reading about “Clean Code Technique”.

* + *Different Package, Similar Name*: In python there are two packages which have similar name but behave way differently. The default package that comes with Anaconda, “whois”, is not supposed to be used, rather package named “python-whois” is supposed to be used.

So, did the following every time the script is executed on a new console:

|  |
| --- |
| pip uninstall whois  pip install python-whois |

* + *Different Python Syntax for 2.x and 3.x*: File read using “open()” vs “with open()”

|  |
| --- |
| def foo():  f = open('a.txt','r')  for l in f:  pass  f.close()  def foo1():  with open('a.txt','r') as f:  for l in f:  pass |

Initially, I used foo() style to open files which lead to many exceptions and errors. Then, the foo1() style (introduced in Python 3.x) was used because the “with” statement automatically handles exceptions and errors by using the concept of context managers.

* + *Browser Version Problem*: When selenium is used with PhantomJS (Headless Browser), some sites don’t allow access to its website because it considers PhantomJS as a non-trustworthy browser. This led to many blank screenshots of the websites. Thus, I switched to Firefox browser” Option” with Selenium.

|  |
| --- |
| #Using PhantomJS  driver = webdriver.PhantomJS(executable\_path="C:/phantomjs-2.1.1-windows/bin/phantomjs")  #Using Firefox  from selenium.webdriver.firefox.options import Options  binary = FirefoxBinary(r'C:\Program Files (x86)\Mozilla Firefox\Firefox.exe')  fp = (r'C:\Users\username\AppData\Roaming\Mozilla\Firefox\Profiles\oqmqnsih.default')  opts = Options()  opts.profile = fp  firefox\_capabilities = DesiredCapabilities.FIREFOX  firefox\_capabilities['marionette'] = True  driver = webdriver.Firefox(capabilities=firefox\_capabilities,firefox\_binary=binary, firefox\_options = opts) |

* + *User Agent Problem*: When Beautiful Soup is used, one has to use Requests library to make a connection with the website. For the same purpose, a user agent is used. This user agent should be a verified one or else the connection is aborted. I used Mozilla as my legit agent.

|  |
| --- |
| hdr = {'User-Agent': 'Mozilla/5.0'} #Make the user agent verified, that is Mozilla  req = Request(URL,headers=hdr)  page = urlopen(req) #Get URL HTML contents  soup = BeautifulSoup(page, 'html.parser') |

* I successfully created a feature extraction automaton script by the end of the month and, as I write this, it is currently collecting running and collecting data.

**Task 3: Machine Learning Model**

* My current task is to explore Machine Learning domain, and read about the “Supervised Learning” and the models that can help in this project specifically.
* I am currently learning about Decision Trees, Random Forests and XGBoost models, which are supposedly going to be used in this project.
* Before I start implementing the models, I must write scripts to convert the raw data, collected by feature-extraction script, to processed data because the raw data cannot be directly fed or given to the model. Thus, I am also learning about preprocessing techniques used in Machine Learning.

All these files are uploaded regularly on the Bitbucket of Symantec using Git.

**FUTURE WORK**

Once the data has been extracted, it will be processed and models will be programmed on which the processed data will be trained and tested. Once the model gives a satisfactory accuracy, the focus will be on lowering the false positives given by the model. They will be described in the next report (May - June).

Thank You.