University of New South Wales
SENG3011_Coders
Design Details

The API is based on scraping data from the web and presenting it on a web service.

Describe how you intend to develop the API module and provide the ability to run it in Web service mode.

Design : The API will be developed using the REST architecture and will utilise HTTP verbs such as GET, POST, PUT and DELETE as inputs and use the JSON format as output. REST focuses on resource-based operations which makes it easy for both developers and web-browsers to consume it.

Implementation : For implementing our API, the first step is to scrape data from the resource(CDC) provided and then extract the information and format it to the user. For scraping data from the website that has our information and have our server scrape the HTML from it.

The API will be called and passing parameters to it and with the help of some python libraries like BeautifulSoup. The developer will then use the BeautifulSoup library to parse this document, and extract the text from the p tag.The extraction process includes different pieces of information like time,location and how many people were affected by the outbreaks.The output received will be stored and then displayed via a web based GUI.

Testing: The API will be tested on Behavioral, Contractual and Result API testing grounds. The API will be examined whether it delivers expected behavior and handles unexpected behavior properly or not. It will be tested to see if tasks defined in specifications have been achieved by code with given right inputs and date format.
As a whole it supports the intended cases that it was designed to solve. Since we are going to use Python Flask in which constantly testing the api is fairly simple we can easily make test cases and then run the code through pytest and other test files.

Extensions: The team is wishing to further enhance the user experience with additional features like a map with affected areas for the period selected. This provides an easy

understanding the scope of outbreak. These features can help to make an application for mass welfare in future.

Optimization: The team will work on the optimisation of the API to make it efficient and speed up the scraping process. Excessive API calls can cause inefficient code. Team will optimize API and handle under fetching and over fetching problems. We will always aim to reduce redundancy in code and unnecessary requests made.

An API acts as an interface between two different applications so that they can communicate with each other. Web service facilitates interaction between two machines over a network. With RESTFul web services, there is a natural mapping between the HTTP methods. In our case we will be using the GET request. GET is used to retrieve data or perform a query on a resource. The data returned from the web service is a representation of the requested resource.

GET /http://www.cdc.gov/outbreaks/simple.htm HTTP/1.0

Accept-Language: application/json

When the client submits a request, it can include a header that specifies the data formats that it can handle. For example, a client that issues a GET request that retrieves an image can specify an When the web API returns the result, it should format the data and specify the JSON format.

API depends on response(data) from scraper which will be shown to user and Scraper takes command from the client to see which data it needs to fetch.

Discuss your current thinking about how parameters can be passed to your module and how results are collected. Show an example of a possible interaction. (e.g.- sample HTTP calls with URL and parameters).

Passing parameters to the API module

Query string parameters appear after a question mark (?) in the endpoint. The question mark followed by the parameters and their values is referred to as the “query string.” In the query string, each parameter is listed one right after the other with an ampersand (&) separating them.The parameters can be passed as,

GET http://api.website.com/1.1/search/specificWebpage.json?q=Dengue

url = http://api.website.com/1.1/search/specificWebpage.json?q=Dengue

There can be some other parameters as well specifying the language,

url = http://api.website.com/1.1/search/specificWebpage.json?q=Dengue&lang=en

of the response or the number of results,

url = http://api.website.com/1.1/search/specificWebpage.json?q=Dengue&count=15

that we want.

Collecting API results

To handle the API output we need to request to connect to the URL and then parse the JSON object received and extract the data that we need.

GET /http://www.cdc.gov/outbreaks/simple.html HTTP/1.0

Accept-Language: application/json

Example :

url = http://www.cdc.gov/outbreaks/simple.json?q=Dengue

response => requests.get(url)

Then, to read the data,

data = response.text

And then finally Parse JSON – convert the string to JSON by using,

parsed = json.loads(data)

In the above scenario, the request is searching for the word DENGUE in the provided specific Webpage.

The data will be extracted as :

```
import requests
import json

url = "http://www.cdc.gov/outbreaks/simple.json?q=Dengue"
response = requests.get(url)
data = response.text
parsed = json.loads(data)
date = parsed["date"]
```

After extracting the data from the web server provided, it will be combined with a Dataframe(Pandas) or any other in the below mentioned way:

```
import pandas as pd

outbreak = pd.dataframe({

        "location" : "Sydney"

        "headline" : "Outbreak in sydney"

        "time" : "2019-03-10"

        "reports" : "disease type and people affected"

})outbreak
```

The result can be either some data that we want or an explanation why it failed.For a failed request like,

http://api.website.com/1.1/search/specificWebpage.json?q=I_am_not_sick&lang=en, the expected output would be,

HTTP/1.1 Status_Code(e.g. 400) Status(Bad request)

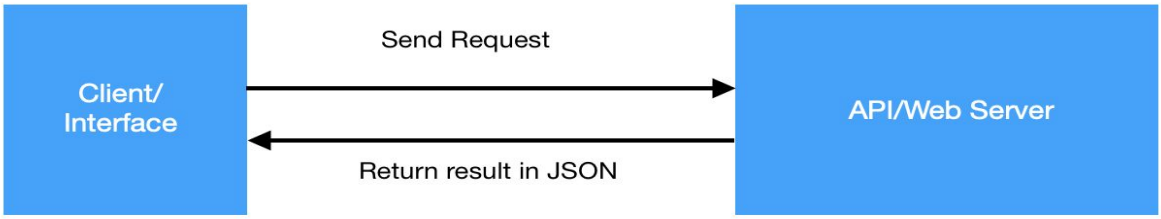
{

“error” : 400,

“message” : “invalid name”

}

Whereas, for a successful request, the response can be stored in a database in the same format as mentioned above using print(response.content).



As the above figure illustrates, the HTTP request will be sent to the Web Server using the API and then the result will be returned to the client in json object sent in in the body of the HTTP response.

Present and justify implementation language, development and deployment environment (e.g. Linux, Windows) and specific libraries that you plan to use.

In choosing which implementation language the team decided to use *Python*, we have considered both *NodeJS* and *Python* which are the two most popular languages in backend development, with the former being very popular with startups.

The languages were compared and the most suitable was selected.

Python	NodeJS
- Python allows tasks to be achieved with fewer lines of code	- Better performance overall: event-based programming makes its processes run faster
- Python provides more advanced web scraping frameworks such as Beautiful Soup which incorporates parsing trees and can parse less than well formed HTML	- Only Javascript is required to code the Frontend and Backend.
- It is better suited for larger projects as Python has clean coding standards	- NodeJS is better suited to real time web applications as it is based on Chrome's V8 engine which is very powerful and fast.
- Error handling means debugging takes less time and is very easy in Python in comparison to other languages.	

We narrowed down the list to two Python frameworks, Django and Flask and selected the most suitable one.

Flask	Django
- Flask is built on minimalism, enabling more flexibility for functionality.	- Django has inbuilt features and templates
- Flask has no browsable API option unlike Django so a less inconvenient alternative is to use Swagger in front of the API.	- Django's framework creates HTML pages to execute all API endpoints and for browsing.
- As a result, its performance is generally faster.	
- Much more freely able to integrate Flask with NoSQL databases such as MongoDB which stores data in JSON.	

For this project, the team wanted greater versatility in implementation as we planned to use a wide range of libraries with BeautifulSoup being one of them.In addition to that, the decided deployment environment were Windows as well as MacOS , not only because they are the most popular, but because we will be developing on those environments and testing will be needed to ensure that they operate the same way on both platforms despite Python being portable.