ANALYTICS PLATFORM FOR PREDICTING EPIDEMICS

SENG3011 Software Engineering Workshops Specification

1 Background

The project specification has been developed in collaboration with the Integrated Systems for Epidemic Response (ISER) which is a NHMRC Centre for Research Excellence located at UNSW. ISER monitors global epidemics and provides critical analysis of outbreaks of public health significance. ISER also conducts applied systems research, enhance collaboration and build capacity in health systems research for epidemic control. It brings together experts in field epidemiology and epidemic response, military experts, international law and risk science experts, and government and non-government agencies involved in epidemic response.

The project specification is meant to contribute towards automating and epidemics detection system called EpiWATCH which uses openly available data sources to produce the following information:

- **Outbreak Alerts:** Rapid intelligence service providing outbreak alerts from publicly available sources.
- **Watching Briefs:** Critical analyses prepared on outbreaks that are serious, persistent, have unusual characteristics or high case fatality rates.
- **Digest:** A regular presentation/newsletter available to inform policy makers, government and other stakeholders about recent outbreaks.
- Refer to the article [5] for some background knowledge on using social media and news for epidemic detection.

2 Project Overview

Student teams will complete an outbreak surveillance system in two stages.

Stage 1: Develop an API to access disease reports following the provided specifications described in Section 3.

Stage 2: Develop a platform that provides one or more of the following functions for an end user interested in epidemics detection:

- Ability to integrate data from different sources and presented in a user-friendly way
- Ability to browse news related to a disease outbreak over a period of time/geographically, identify news about outbreak of interest
- Ability to examine social media related posts on disease outbreaks over a period of time, identify particular trends

- Ability to provide advanced analysis facilities such as analysing the impact of an outbreak on residents of the region over a period of time or predicting potential outbreaks based on historical patterns.

These functions should be provided through a user friendly and integrated web-based GUI. The requirements provided in this specification are very generic and teams have the flexibility to add new functionalities, after discussing them with their mentor. Furthermore, the teams have the flexibility to select the scope of their platform, what users they are targeting for, what business value they deliver. It is expected that teams will be delivering unique solutions at the end of the workshop.

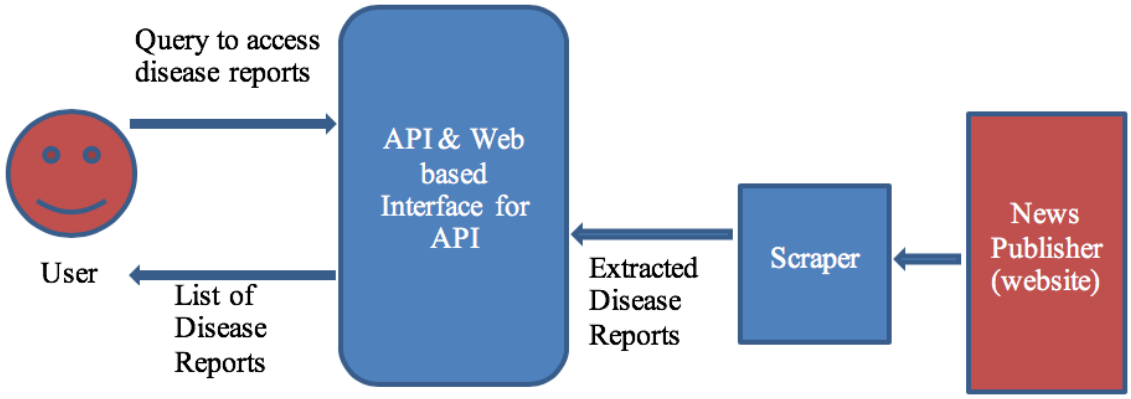
Details of stage 2 are described in section 4.

3 First stage

Each team will select a website from Appendix A as their main data source. Information on how the current EpiWatch team manually extracts data from these data sources is provided for you on WebCMS. The objective of this stage is to automate this data extraction and publication process.

Each team should:

- Develop a scraper to gather data from the main data source (Section 3.1).
- Develop an API to provide disease reports on demand, following the input and output formats defined (Section 3.2).
- Create a website with API documentation (Section 3.3)



The language in which the modules are to be written is not important as long as the API can be invoked publicly over the web (via Swagger).

3.1 Develop a scraper to frequently gather data from the main data source.

The objective of the scraper is to automatically access the news publishing web site, and fetch disease reports/news articles. You can decide how often scraper should run.

Based on the availability and reliability of the selected data source, you can design your scraper to access data source in a given frequency (e.g. daily) and accumulate data in a repository or to access data source when an API call is made and fetch the data on demand.

Note- We recommend processing raw news items, extracting metadata and saving them as separate attributes. Moreover, identifying key words, indexing and storing them in a database can be advantageous for efficient search.

You are free to use any third-party libraries freely available. An example is Python scrappy library (https://scrapy.org). [1] provides a tutorial on how to develop a scraper with scrappy.

3.2 Develop an API to accesses disease reports, following the input and output formats defined.

The objective is to build an API that would easily enable a client to find and access all the disease reports related to a given search terms including disease, and location for a given period of time.

The module aims to isolate some specific news according to the user's criteria.

The user has to input 3 main information: period of interest, key term and location

The module will then filter the disease reports first according to the period (start date + end date) entered by the user and then the key word and locations.

Note: It is easy to filter data, if the disease reports are annotated with relevant search terms and locations as additional attributes.

The API should deliver data in the specific format defined in Appendix C. It is acceptable to leave certain output fields empty, if disease report does not contain relevant information.

1. Input Parameters for API

The user has to input the 3 main information:

- period of interest
- key_terms
- location

Period of Time:

It's composed of two inputs:

- A start date
- An end date

Both dates have the respect the following format: **"yyyy-MM-ddTHH:mm:ss".**

These inputs can NOT be empty.

For example: if I want to retrieve all the news that were issued between the first of October 2015 at 08:45:10 (8:45am 10 seconds) and the first of November 2015 at 19:37:12 (7:37pm 12 seconds), my inputs will be:

- start_date = "2015-10-01T08:45:10"
- end_date = "2015-11-01T19:37:12"

Note: You can consider "timezone" as an optional parameter, specially if your datasouce provides time zones. If the user provides a value for it, then the service considers its value, otherwise it ignores the parameter (or uses a default value).

Key terms:

This input contains a comma separated list of all the key terms you want to get news about.

This input can be empty or omitted in the case where the user doesn't want to restrict his search.

Example: if you want to retrieve news related to Anthrax and Zika your input list should be:

- keyterms = "Anthrax,Zika"

API should not be case sensitive to the input terms. Set of frequently used key terms are provided in Appendix B.

Location:

User should be able to search disease reports by a location name (city/country/state etc.), which is a string to be matched with the content in the disease report.

Note: Disease reports published in data sources do not have a standard way of publishing locations. String based search is sufficient for your API. But we encourage teams to explore effective ways of extracting locations from the disease report and matching with user queries so that geographic locations can be matched at different levels. For example, is it possible to implement API in such a way that when a user is searching for New South Wales, all disease reports that mention "Sydney" are returned, detecting Sydney is located inside New South Wales. You may want to explore geo-location taxonomies for that.

3.3 Bonus Challenge: Turn publicly available datasets to APIs

Interested teams can also choose one/more publicly available datasets in the area of health (listed in Appendix A, section 5.2) and make them accessible as APIs to be used by other teams. The objective of this task is to utilize already pre-processed and acquired data in API design and development processes.

3.4 Documenting the API.

You are required to publish API documentation with parameter specifications, capability to execute and test the API online. This will be used by evaluators, and other teams in phase 2.

API documentation should be published in Swagger UI (https://swagger.io/tools/swagger-ui/). Swagger is an open-source software framework backed by a large ecosystem of tools that helps developers design, build, document, and consume RESTful Web services. Swagger allows you to describe the structure of your APIs so that machines can read them. [2] will provide basic introduction to Swagger platform.

Your choice of programming platform may have libraries that automatically generate swagger documents for your APIs. [3] is an example framework for Python.

Your SwaggerUI should be host in a public URL and evaluators should be able to try your API through that. An example Swagger API documentation can be found in [4].

Marking metrics for First Stage:

- API design should follow REST design principals (e.g. naming endpoints, versioning, giving examples, using HTTP methods such as POST, GET, PUT, etc. appropriately)
- API deployment and usage is documented in details (documentation should be sufficient for a non-technical person to understand your service, how to use it)
- API can pass the test without any exception (support different input values)

4 Second stage

Second stage is to develop a web application that integrates disease reports from the API you developed and other APIs. You are required to integrate with at least 1 of the other groups APIs; however, the more you leverage other groups' APIs the more functionalities your system supports. Third-party APIs including Google News API (https://newsapi.org/s/google-news-api), Twitter API, etc are allowed to use as well.

Using the interface you design and develop, users are able to search about disease, see visualized disease reports over time or based on locations, and get some predictions (based on news, trends, reports, etc).

Furthermore, each team will have the choice of adding new requirements in consultation with their mentor. Examples of new requirements are listed below:

- Adding analytics facilities such the ability to predict the outbreak of a particular disease based on a number of disease report appearing within a similar location and around the same time period. You are free to use any machine learning and deep learning techniques and models for this part.
- Enhancing the data visualization and the user experience in navigating through large number of disease reports over multiple locations and time periods. Appropriate usage of charts/maps/etc to visualize the output is important.
- Using the additional datasets provided in section 5.2 to provide value added information (no bonus marks).

The second stage gives a lot of freedom for the teams to choose an area they are interested in, they can refine their ideas during mentoring sessions. The main theme is looking at interrelations between disease reports and information extracted from social media to get an idea about possible outbreaks in multiple regions over specific time periods.

Note: Appendix F provides a list of web sites that may provide additional information useful for your platform.

5 Appendix A: Data Sources for outbreak reports

5.1 Data source - websites

The students need to choose one particular data source amongst those provided here:

- **WHO Website – Outbreaks News-**
  - http://www.who.int/emergencies/diseases/news/en/
- **ProMed-**
  - http://www.promedmail.org
- **CDC-**
  - https://www.cdc.gov/outbreaks/
- **Outbreak News Today-**
  - http://outbreaknewstoday.com
- **CIDRAP-**
  - http://www.cidrap.umn.edu
- **Global Incident Map-**
  - http://outbreaks.globalincidentmap.com

Note: additional details on how current users manually access data on these pages are published in WebCMS for your reference.

5.2 Additional datasets (Only for Bonus Challenge)

The students can also choose one/more datasets amongst those provided here.

Optional: You can use these datasets at the project stage 2 to add value and improve quality of the web application as well. No bonus marks awarded.

- **Coronavirus dataset-**
  - https://www.kaggle.com/brendaso/2019-coronavirus-dataset-01212020-01262020
  - https://www.kaggle.com/sudalairajkumar/novel-corona-virus-2019-dataset
  - https://github.com/CSSEGISandData/COVID-19
- **HIV/AIDS Living Cases-**
  - https://data.world/health/hivaids-living-cases-2012
  - https://wonder.cdc.gov/AIDSPublic.html
- **West Nile Virus Cases, 2006-present-**
  - https://data.world/chhs/3205b420-3f62-4a02-8d2e-9a9ed34c49f4
- **Zika Virus Epidemic-**
  - https://data.world/data-society/zika-virus-epidemic
- **CT Medical Images-**
  - https://www.kaggle.com/kmader/siim-medical-images
  - https://nihcc.app.box.com/v/DeepLesion
  - https://openfmri.org/dataset/

- **Flu tracker-**
    - https://flutrackers.com/forum/
    - Latest news page: - https://flutrackers.com/forum/search?searchJSON={%22last%22%3A{%22from%22%3A%22lastDay%22}%2C%22view%22%3A%22topic%22%2C%22starter_only%22%3A+1%2C%22sort%22%3A{%22lastcontent%22%3A%22desc%22}%2C%22exclude_type%22%3A[%22vBForum_PrivateMessage%22]}
- **H5N1 (Blog by Crawford Kilian)**
    - https://crofsblogs.typepad.com/h5n1/

6 Appendix B: Key Search Terms

General Search Term	Specific Search Term
Outbreak	Zika
Infection	MERS
Fever	Salmonella
Virus	Legionnaire
Epidemic	Measles
Infectious	
Illness	Category A Agents:
Bacteria	Anthrax
Emerging	Botulism
Unknown virus	Plague
Myster(ious)y disease	Smallpox and other related pox viruses
	Tularemia
	Junin Fever
	Machupo Fever
	Guanarito Fever
	Chapare Fever
	Lassa Fever
	Lujo Fever

	Hantavirus Rift Valley Fever Crimean Congo Hemorrhagic Fever Dengue Ebola Marburg

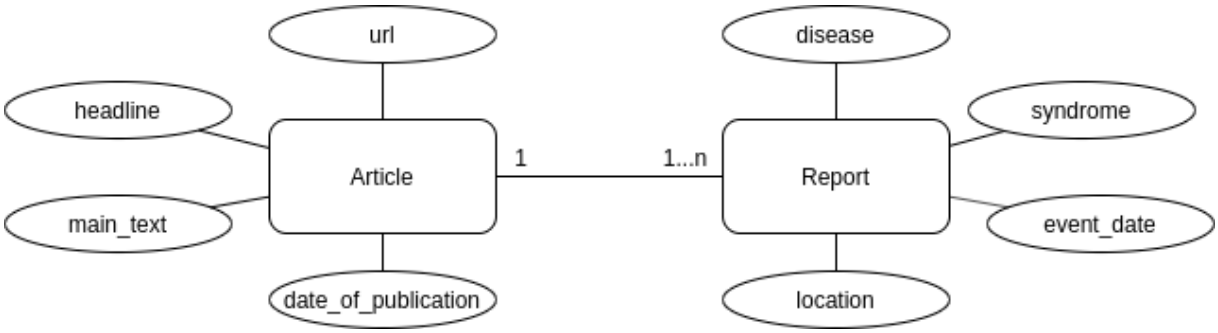
7 Appendix C: Disease Report Standard Format

7.1 Background

Throughout the report definition, we will refer to *articles* and *reports*. They are two different concepts. A single *article* is a url on a web site containing information about one or more cases of a disease and / or syndrome. A single *report* refers to one of these cases in an article.

In the simplest case, a single article only mentions a single case, e.g. “An individual with Coronavirus confirmed in Wuhan on 10 January 2020”. That would be modelled as a single article containing a single report.
In a more complicated case, a single article mentions two different cases, e.g. “An individual with Coronavirus confirmed in Wuhan on 10 January 2020 and someone with MERS confirmed in UAE on 15 January 2020”. This would be modelled as a single article containing two reports.

An article has data associated with it such as its headline, its url, its publication date and the body-text of the article. A report contains data such as diseases, syndromes, location and event_date (e.g.: date that the case was reported). Given below is an example schema for data associated with *articles* and *reports.*



However, as you may imagine, it is easier to find all disease and syndromes mentioned in an article, than it is to detect two entirely different reports in a single article. Therefore, in your initial prototype, you may wish to simplify the task by just having one article link to only one report - and adding all the diseases, syndromes and locations mentioned in the article to that single report. A more advanced system should be able to detect that an article mentions two separate reports, and create two report objects two represent the individual reports.

7.2 The article json

When a specific article is scanned (i.e. a single article such as a single WHO report, or a single news article on a news web site, etc.), the text within it should be processed for relevant disease information. That information should be stored as per the json schema described in this document. Your API should also transmit article data using this json schema - so that other groups know what to expect from your API.

A guide to interpret the schema description in this document:

- Any text within angled brackets should be replaced by some other json, as follows:
  - If the text within the angled brackets mentions a json type such as 'string', or 'number', it should be replaced with the appropriate type. For example, if in the schema, you encounter text such as '<string>' or '<number>', in your json file, these should be replaced by a valid JSON string or valid JSON number respectively.
  - If the text within the angled bracket mentions some json type with two colons and then a name, it refers to a subsection of this document that will describe the json that should replace the angled bracket. For example, if you encounter '<string::date>' within the schema, look for a section of this document called '<string::date>'. That section will describe what you should insert in those places.
- As usual, { } refers to a json object and [ ] refers to a json array.
- See https://www.json.org/ for more info on the json format.

You should store and transmit article data in the following json format:

```
{
    url: <string>,
    date_of_publication: <string::date>,
    headline: <string>,
    main_text: <string>,
    reports: [<object::report>]
}
```

- **url** => The url of the article
- **date_of_publication** => The date the article was published
- **headline** => The headline of the article
- **main_text** => The main body of the article text
- **reports** => In the simplest case, a single article will mention a single case, for example an individual with Coronavirus in Sydney. For such an article, the "reports" array will contain a single object::report object which will contain this information. In a more complicated article, perhaps two completely separate cases are mentioned in the same article. For example, individual with Coronavirus in Wuhan, and an individual with MERS in UAE. For such an article, the "reports" array may have two separate object::report objects - one that contains the information about the MERS case and the other containing the information about the Coronavirus case. This isn't straightforward at all. So at first - feel free to add everything to a single object::report object - and then later on try to add the ability to detect different case reports.

7.3 object::report

An object containing information about a case mentioned in an article.

```
{
    diseases: [<string::disease>],
    syndromes: [<string::syndrome>],
    event_date: <string::date>,
    locations: [<object::location>],
}
```

- **diseases** => The array of diseases represent a disjunction. That is, if an article states "the individual may have coronavirus or MERS - to be confirmed" - then both Corona and MERS are to be added to the diseases array. However, if the articles mentions a conjunction - that is e.g. two different disease cases are mentioned in the article "one individual was confirmed MERS another Corona" - then you'll want two separate object::report objects. But as mentioned earlier on, that is not straightforward to achieve in an automated fashion - so to begin with - your system may want to simply place all diseases it finds in an article into a single object::report. You can then add the more advanced functionality later.
- **syndromes** => The array of syndromes is a conjunction. That is, if three syndromes are listed for a particular case mentioned in the article, then all three are added to this object.
- **event_date** => The date on which the case occured (not the publication date of the article). E.g. if the text says "An individual began to show symptoms on 1 Jan 2020" in an article that was published on 10 January 2020, then the event_date is 1 Jan 2020.
- **locations** => An array of all the locations mentioned in the article referring to this particular case.

See the two example json files included in section 7.10 for clarification.

7.4 string::date

Contains either a single date or a range of dates:
Either `string::date-exact` or `string::date-range`

7.5 string::date-exact

yyyy-mm-dd hh:mm:ss format. Year is mandatory, every other segment is optional. Use 'x' character if missing. String must match the following regular expression: (can use https://regex101.com/ to test)

```
^(\d{4})-(\d\d|xx)-(\d\d|xx) (\d\d|xx):(\d\d|xx):(\d\d|xx)$
```

Examples of what is OK:

- 2018-xx-xx xx:xx:xx
- 2018-11-01 xx:xx:xx
- 2018-11-xx 17:00:xx

7.6 string::date-range

Let ***d1*** and ***d2*** both be of format string::date-exact (see section above). Then, string::date-range must follow format:

d1 to ***d2***

And furthermore d1 must be a date before d2.
The string must match the following regexp, and the first date must be before the second date:

```
^(\d{4})-(\d\d|xx)-(\d\d|xx)     (\d\d|xx):(\d\d|xx):(\d\d|xx)     to
(\d{4})-(\d\d|xx)-(\d\d|xx) (\d\d|xx):(\d\d|xx):(\d\d|xx)$
```

Some examples of what is OK:

- 2018-xx-xx xx:xx:xx to 2019-xx-xx xx:xx:xx
- 2018-11-01 17:xx:xx to 2018-12-xx xx:xx:xx

Some examples of what is NOT OK:

- 2018-xx-xx xx:xx:xx to 2017-xx-xx xx:xx:xx
  (*the first date is later than the second date*)
- 2018-11-03 xx:xx:xx to 2018-11-xx xx:xx:xx
  (it is ambiguous whether the second date is later than the first date)

7.7 string::disease

An identifier corresponding to the name of a disease listed in the supplementary file **disease_list.json.**

7.8 string::syndrome

An identifier corresponding to the name of a syndrome listed in the supplementary file **syndrome_list.json.**

7.9 object::location

The complications involved in storing locations can be a 3 month project in and of itself. So two versions will be allowed, a basic and an advanced version.

The basic version:

```
{
    country: <string>,
    location: <string>
}
```

Where *country* is the country name and *location* contains any other details about the location (e.g. province, city, etc.)

Advanced version using either the geonames database or the Google Places database:

```
{
    geonames_id: <number>
}
Or
{
    google_id: <number>
}
```

where the geonames_id field refers to a geonames ID from the geonames database (http://www.geonames.org/), or where google_id refers to a Google Places ID.

7.10 Two example disease reports and how they are published in JSON are shown below:

- Here is the article 1:
    - URL: https://www.who.int/csr/don/17-january-2020-novel-coronavirus-japan-ex-china/en/
    - Title: Novel Coronavirus – Japan (ex-China)
    - Date of publication: 17 January 2020

On 15 January 2020, the Ministry of Health, Labour and Welfare, Japan (MHLW) reported an imported case of laboratory-confirmed 2019-novel coronavirus (2019-nCoV) from Wuhan, Hubei Province, China.
The case-patient is male, between the age of 30-39 years, living in Japan.
The case-patient travelled to Wuhan, China in late December and developed fever on 3 January 2020 while staying in Wuhan. He did not visit the Huanan Seafood Wholesale Market or any other live animal markets in Wuhan. He has indicated that he was in close contact with a person with pneumonia.
On 6 January, he traveled back to Japan and tested negative for influenza when he visited a local clinic on the same day.

The JSON Example:

- Here is the article 2:
    - URL: www.who.int/lalala_fake_article
    - Title: Outbreaks in Southern Vietnam
    - Date of publication: 12 December 2018

Three people infected by what is thought to be H5N1 or H7N9 in Ho Chi Minh city. First infection occurred on 1 Dec 2018, and latest is report on 10 December. Two in hospital, one has recovered. Furthermore, two people with fever and rash infected by an unknown disease.

8 Appendix D - disease_list.json


```
[
{ "name": "unknown" },
{ "name": "other" },
{ "name": "anthrax cutaneous" },
{ "name": "anthrax gastrointestinous" },
{ "name": "anthrax inhalation" },
{ "name": "botulism" },
{ "name": "brucellosis" },
{ "name": "chikungunya" },
{ "name": "cholera" },
```

{ "name": "cryptococcosis" },
{ "name": "cryptosporidiosis" },
{ "name": "crimean-congo haemorrhagic fever" },
{ "name": "dengue" },
{ "name": "diphteria" },
{ "name": "ebola haemorrhagic fever" },
{ "name": "ehec (e.coli)" },
{ "name": "enterovirus 71 infection" },
{ "name": "influenza a/h5n1" },
{ "name": "influenza a/h7n9" },
{ "name": "influenza a/h9n2" },
{ "name": "influenza a/h1n1" },
{ "name": "influenza a/h1n2" },
{ "name": "influenza a/h3n5" },
{ "name": "influenza a/h3n2" },
{ "name": "influenza a/h2n2" },
{ "name": "hand, foot and mouth disease" },
{ "name": "hantavirus" },
{ "name": "hepatitis a" },
{ "name": "hepatitis b" },
{ "name": "hepatitis c" },
{ "name": "hepatitis d" },
{ "name": "hepatitis e" },
{ "name": "histoplasmosis" },
{ "name": "hiv/aids" },
{ "name": "lassa fever" },
{ "name": "malaria" },
{ "name": "marburg virus disease" },
{ "name": "measles" },
{ "name": "mers-cov" },
{ "name": "mumps" },
{ "name": "nipah virus" },
{ "name": "norovirus infection" },
{ "name": "pertussis" },
{ "name": "plague" },
{ "name": "pneumococcus pneumonia" },
{ "name": "poliomyelitis" },
{ "name": "q fever" },
{ "name": "rabies" },
{ "name": "rift valley fever" },
{ "name": "rotavirus infection" },
{ "name": "rubella" },
{ "name": "salmonellosis" },
{ "name": "sars" },
{ "name": "shigellosis" },
{ "name": "smallpox" },
{ "name": "staphylococcal enterotoxin b" },
{ "name": "thypoid fever" },
{ "name": "tuberculosis" },
{ "name": "tularemia" },
{ "name": "vaccinia and cowpox" },
{ "name": "varicella" },
{ "name": "west nile virus" },
{ "name": "yellow fever" },
{ "name": "yersiniosis" },
{ "name": "zika" },
{ "name": "legionares" },

{ "name": "listeriosis" },
{ "name": "monkeypox" },
{ "name": "COVID-19" }
]

9 Appendix E - syndrome_list.json

[
{ "name": "Haemorrhagic Fever" },
{ "name": "Acute Flacid Paralysis" },
{ "name": "Acute gastroenteritis" },
{ "name": "Acute respiratory syndrome" },
{ "name": "Influenza-like illness" },
{ "name": "Acute fever and rash" },
{ "name": "Fever of unknown Origin" },
{ "name": "Encephalitis" },
{ "name": "Meningitis" },
]

10 Appendix F: Websites for additional information

- Emerging Viruses, Virus Discovery and Virus Characterization (Blog by Ian M Mackay) http://www.scoop.it/t/virus-discovery-and-characterisation
- Training Programs in Epidemiology and Public Health Interventions Network (TEPHINET) – http://www.tephinet.org/
- Health map – http://www.healthmap.org/en/
- Health map daily diseases – http://www.healthmap.org/diseasedaily/category/outbreak
- Disease Outbreak News (DONs) – http://www.who.int/csr/don/en/
- MedScape Infectious Diseases – http://www.medscape.com/infectiousdiseases
- WHO ICD11 Disease Classification https://icd.who.int/
- WHO – The Weekly Epidemiological Record (WER) – http://www.who.int/wer/en/
- Nosocomial outbreak database – http://www.outbreak-database.com/Home.aspx
- WHO Dengue net – http://apps.who.int/globalatlas/default.asp
- Australia notifiable disease data (Provides downloadable tables, need to read and update frequently)– http://www9.health.gov.au/cda/source/cda-index.cfm
- NSW Infectious disease alerts - http://www.health.nsw.gov.au/infectious/alerts/Pages/default.aspx

11 References:

[1] https://www.scrapehero.com/tutorial-how-to-scrape-amazon-product-details-using-python/

[2] https://swagger.io/docs/specification/2-0/what-is-swagger/

[3] https://flask-restplus.readthedocs.io/en/stable/swagger.html#swagger-ui

[4] http://petstore.swagger.io

[5] https://theconversation.com/social-media-for-tracking-disease-outbreaks-fad-or-way-of-the-future-66401