

# SENG3011 DELIVERABLE 2

## API Design Details

Team Brickwalls

Mentor: Yi Zhuang

Z5205003 Allen Wu  
Z5207915 Monica He  
Z5207001 Vishnu Pillai  
Z5228933 Kshitiz Saini  
Z5190299 Sheina Edeline Tenggara

## Table of Contents

System Architecture	3
Final Architecture of API Design	4
Justification of Implementation	17
Complications and Shortcomings	18
Potential Future Changes	20

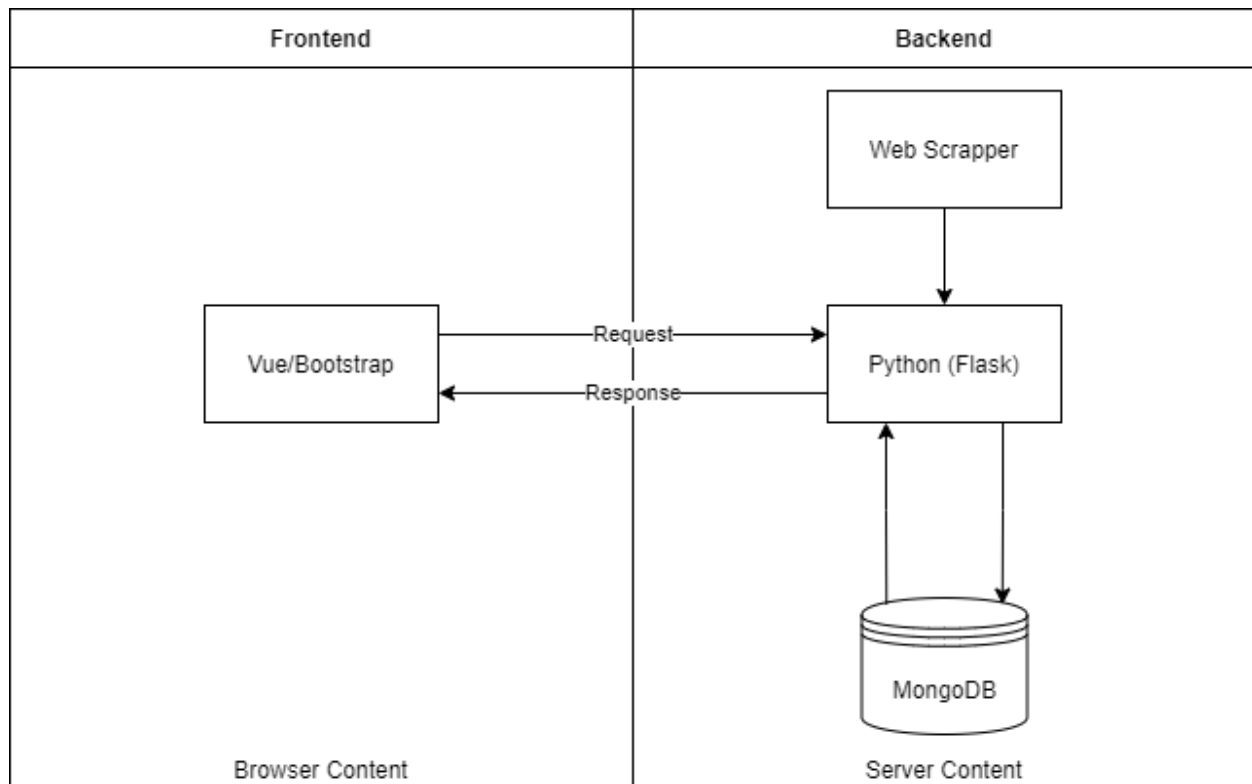
## System Architecture

System Architecture:

The system will have four main components:

1. Frontend
2. Database
3. API
4. Web scrapers

The final architecture of the API has been illustrated in the following flowchart:



The four main components will implemented through a variety of methods:

1. The frontend will utilise the Vue.js javascript framework
2. The backend database will utilise a MongoDB database
3. The API will be written in Python utilising Flask and will be deployed on Azure
4. The Web Scraper will be written in Python and is an integral part to the API

## Final Architecture of API Design

The API Design consists of 3 core components:

1. Python API
2. Web scraper
3. MongoDB database

### Python API

The API itself is implemented in Python and will utilise the lightweight web development framework Flask. Flask is currently one of the most popular web frameworks for Python and has a plethora of resources and documentation to assist in the development of the system. It also allows for the handling of API calls and can be integrated using a wide variety of database implementations, such as the MongoDB database that is a critical component of the project.

The API is hosted on Azure and can be found here: <https://diseasereportapi.azurewebsites.net/>

The API was designed on Stoplight.io and the JSON file of the API documentation can be viewed in the github repository, in PHASE\_1/API\_Documentation/Disease-Reports-API.v1.json.

Utilising Stoplight.io allows for calling the endpoints as shown:

GET ▼

Query [4]	Headers	Body	Path	Code Generation	Mocking
<input checked="" type="checkbox"/> start				20000101	×
<input checked="" type="checkbox"/> end				20211212	×
<input checked="" type="checkbox"/> keyterms				case	×
<input checked="" type="checkbox"/> location				United States	×
Key				Value	

Send ▼

Response Body	Response Headers [2]	Original Request	Status: 200 OK	Time: 7101ms
<div> <input type="radio"/> raw           <input checked="" type="radio"/> pretty           <input type="radio"/> rendered         </div> <pre> 1 { 2   "data": [ 3     { 4       "date_of_publication": "2021-03-13", 5       "headline": "Delaware reports 1st case of COVID-19 variant B.1.351", 6       "main_text": "The Delaware Division of Public Health (DPH) announced the first confirmed 7       "reports": [ 8         { 9           "diseases": [ 10            "coronavirus" 11          ], 12          "event_date": "2021-03-13", 13          "locations": [ 14            "Delaware" 15          ], 16          "syndromes": [ 17            "syndrometest1" 18          ] 19        } 20      ] 21    } 22  ] 23 } </pre>				

GET	https://diseasereportapi.azure	/restrictions?states=wa
-----	--------------------------------	-------------------------

Query [1]	Headers	Body	Path	Code Generation	Mocking
<input checked="" type="checkbox"/> states			wa		<input checked="" type="checkbox"/>
Key			Value		

Send

Response Body	Response Headers [2]	Original Request	Status: 200 OK	Time: 220ms
<input type="radio"/> raw <input checked="" type="radio"/> pretty <input type="radio"/> rendered				
<pre> 1 { 2   "data": [ 3     { 4       "rules": [ 5         { 6           "date effective": "Effective 12.01am, Monday, March 15:", 7           "link": "/government/announcements/victoria-transition-very-low-risk-march-15", 8           "rule 1": "travellers from Victoria are able to enter WA without self-quarantine" 9         }, 10        { 11          "date effective": "Effective 12.01am, Monday, March 15:", 12          "link": "/government/announcements/changes-capacity-venues", 13          "rule 2": "capacity limits expanded for venues." 14        } 15      ], 16      "state": "Western Australia" 17    } 18  ], 19  "endpoint": "/restrictions?states=wa", 20  "status": 200, 21  "team_name": "Team Brickwalls", </pre>				

## Iteration 1

Following feedback from the group's mentoring session, the API design was modified in order to meet the criteria specified in the specifications. Previously, disease reports were outputted in a JSON format having the following structure: id, date, country, cases and disease.

Disease reports are the outputs of the API and include the date, country, number of cases and the disease.

**Disease Report** {5} Disease reports are the outputs of the API and include the date, country, number of cases and the disease.

id **integer** Unique identifier for the given user.

date **string** <date> Date of the found case +1

country **string** Country with the disease cases

cases **integer** Number of cases of the disease +2

disease **string** Disease that has infected the country

The endpoints for the API which were originally created in Deliverable 1 include the following:

### GET

- /getReportID/{id}: retrieves disease report with the same corresponding id
- /getReportCountry/{country}: retrieves a list of disease reports with the same corresponding country
- /getReportDisease/{disease}: retrieves a list of disease reports with the same corresponding disease
- /getReportDate/{date1}/{date2}: retrieves a list of disease reports within the range of the passed dates
- /getReportDiseaseCountry/{disease}/{country}: retrieves a list of disease reports with the corresponding disease and country

### PUT

- /addReport: adds a disease report

The getReportID endpoint which was created in Deliverable 1 was removed as it was unnecessary to be able to search articles based on IDs as individuals looking for articles would not know specific IDs.

The getReportCountry endpoint was modified into a location endpoint which enables users to search disease reports using a location name (city/country/state), which is a string to be matched with the content in the disease report. This was changed in order to adhere to the project specifications and thus allowed us to cater for extra locations such as cities and states rather than just countries.

The getReportDisease endpoint was also modified and replaced with the key terms endpoint. This input contains a comma separated list of all the key terms you want to obtain news about. This input can be empty or omitted in the case where the user does not want to restrict their

search result. This modification is advantageous as it is dynamic and not hardcoded, allowing multiple key terms to be searched, rather than restricting it to diseases.

## Iteration 2

Following consultation with our team members and tutor, the endpoints for the API design was modified as shown below:

### GET

- `/articles?start={start}&end={end}&keyterms={keyterms}&location={location}`
- `/restrictions?states={states}`

### Article Endpoint: Parameters

The article endpoint is a GET request in which users can retrieve articles based on their query parameters.

These parameters are:

- Start:
- End
- Keyterms
- Location

The article endpoint has the start and end date query parameters. Both start and end date parameters are required in order for an article to be returned. The start and end date parameter is utilised in determining what articles are to be returned depending on the range of dates and must be a valid range. The format for both the start and end dates must be in the format %YYYY%MM%dd. E.g. 20210317 is equivalent to 17/03/2021. This format was used instead of the format specified in the project specification. The project specification outlined the date format to be “2015-10-01T08:45:10”, however after analysing the articles provided by the [datasource](#), it was determined that the time was unnecessary. It would inconvenience users to require them to input the date as well as the time and thus was removed. The current date format allows for an easily formattable date utilised in the API and more readable as special characters would be obscured in the URL endpoint.

The put request which provided the ability to add disease reports in our initial API design was removed completely. This was because it was unnecessary for users to add additional reports as it could potentially violate the validity and reliability of the resources. Due to the publicity of our API, allowing unknown users to add disease reports could interfere with the retrieval of accurate reports as information added by other users may not be relevant.

Below shows an example of the query using our new API design:

articles?start=20190701&end=20210319&location=United%20States&keyterms=case

```
{
  "data": [
    {
      "date_of_publication": "2021-03-13",
      "headline": "Delaware reports 1st case of COVID-19 variant B.1.351",
      "main_text": "The Delaware Division of Public Health (DPH) announced the first confirmed case of the COVID-19 variant, SARS-CoV-2 B.1.351, in the state.",
      "reports": [
        {
          "diseases": [
            "coronavirus"
          ],
          "event_date": "2021-03-13",
          "locations": [
            "Delaware"
          ],
          "syndromes": [
            "syndrometest1"
          ]
        }
      ],
      "url": "http://outbreaknewstoday.com/delaware-reports-1st-case-of-covid-19-variant-b-1-351/"
    }
  ],
  {

```

The format of the return object of the articles endpoint is:

articlesReturn {6}
endpoint string
status integer
team_name string
time string
time_taken string
▼ data article[] {5}
url string
date_of_publication string
headline string
main_text string
▼ reports array(object) {4}
diseases array(string)
syndrome array(string)
event_date string
locations array(string)

The data attribute is the list of articles that fit within the filter expressed in the endpoint. The other attributes exist as to generate logs of what endpoints are used, what the status code is, time the endpoint was used and the time it took to complete the request.



### Restrictions Endpoint: Parameters

An additional endpoint, `/restrictions?states={states}`, was added to our API design to enable users to retrieve relevant restrictions or rules for a given state. Users are able to provide an input to the API parameter in order to retrieve relevant information for that particular input. The current possible inputs are NSW, ACT, NT, QLD, SA, TAS, VIC, WA and ACT. Multiple inputs are possible and are done by separating each state with a comma and no space.

Below shows an example of the query utilising the restrictions endpoint:

`/restrictions?states=nt`

```
{
  "data": [
    {
      "rules": [
        {
          "rule 1": "fill in a Border Entry Form"
        },
        {
          "rule 2": "complete 14 days of mandatory supervised quarantine at your own expense*"
        }
      ],
      "state": "Northern Territory"
    }
  ],
  "endpoint": "/restrictions?states=nt",
  "status": 200,
  "team_name": "Team Brickwalls",
  "time": "2021-03-19 13:43:49.069023",
  "time_taken": "0:00:00.000266"
}
```

The format of the return object of the restrictions endpoint is:

restrictionsReturn {6}	
endpoint	string
status	string
team_name	string
time	string
time_taken	string
▼ data	state_restriction[] {2}
state	string
▼ rules	array[object] {4}
rule #	string
date effective	string
title	string
link	string

This is similar to the return object of the articles endpoint. The only difference is the data attributes which returns a list of restrictions. The restrictions is another object that displays the list of rules and restrictions in place for the specific state.

The format of the restriction objects is:

`state_restriction {2}`

`state string`

▼ `rules array[object] {4}`

`rule # string`

`date effective string`

`title string`

`link string`

However, the date effective, title and link attributes are optional and may not exist for some of the states. This is due to the different governmental websites for the states having different attributes and information displayed and thus the results were varied.

### Logging and the Log File

The API also keeps track of the endpoints that are called. These logs are generated in the return object of each GET request and also stored in a text file in PHASE\_1/API\_SourceCode/Azure\_Flask/log\_file.txt.

Using the API generates log output and stores the log information in a log file.

The log output has the 2 formats. One for errors and the other with the accompanying data.

`articlesReturn {6}`

`endpoint string`

`status integer`

`team_name string`

`time string`

`time_taken string`

► `data article[] {5}`

`error {6}`

`timestamp string`

`status integer`

`error string`

`message string`

`path string`

`client_request string`

The error log contains information of the timestamp, status code, error, error message, endpoint and client request. The return log contains information of the endpoint, status code, team name, time, time taken and the accompanying data, which may be a list of articles or restrictions and depends on which endpoint is called.

The log file contains a list of every request that was handled in the API.

Each log in the log file contains information on the endpoint, current time, time taken to handle the request and the status code. The format of each log is:

ENDPOINT: [<endpoint>] CURRENT TIME: [<time>] TIME TAKEN: [<time taken>] STATUS: [<status code>]

An example of the log file is shown below:

```
ENDPOINT: [/restrictions?] CURRENT TIME: [2021-03-18 22:45:54.715667] TIME TAKEN: [0:00:00.001547] STATUS: [200]
ENDPOINT: [/restrictions?states=sa] CURRENT TIME: [2021-03-18 22:45:54.723794] TIME TAKEN: [0:00:00.000327] STATUS: [200]
ENDPOINT: [/articles?start=20000401&end=20211202&location=United%20States&keyterms=case] CURRENT TIME: [2021-03-18 22:46:08.577973] TIME TAKEN: [0:00:00.001181] STATUS: [200]
ENDPOINT: [/articles?start=20100401&end=20001202] CURRENT TIME: [2021-03-18 22:46:08.577973] TIME TAKEN: [0:00:00.001181] STATUS: [200]
ENDPOINT: [/articles?start=201054001&end=200013142] CURRENT TIME: [2021-03-18 22:46:08.577973] TIME TAKEN: [0:00:00.001181] STATUS: [200]
```

## Testing

The API tests were conducted utilising the python unit tests package. There are 2 test files to test the API and they are both located in:

PHASE\_1/Test\_files

The articlesTest.py file tests the articles endpoint and the restrictionsTest.py file tests the restrictions endpoint.

The articles test involve testing the articles endpoint in different cases:

- No dates provided: API returns log with error code 400
- One of the dates provided: API returns log with error code 400
- Dates with wrong format is provided: API returns log with error code 400
- Dates with invalid range (start date is after the end date): API returns log with error code 400
- Dates are the same: API returns log with status code 200 and the corresponding articles
- Dates with valid range (start date is before the end date) API returns log with status code 200 and the corresponding articles
- Single keyterm: API returns log with status code 200 and the corresponding articles
- Multiple keyterms: API returns log with status code 200 and the corresponding articles
- Single location: API returns log with status code 200 and the corresponding articles
- Single keyterm and single location: API returns log with status code 200 and the corresponding articles

The restrictions test involve testing the restrictions endpoint in different cases:

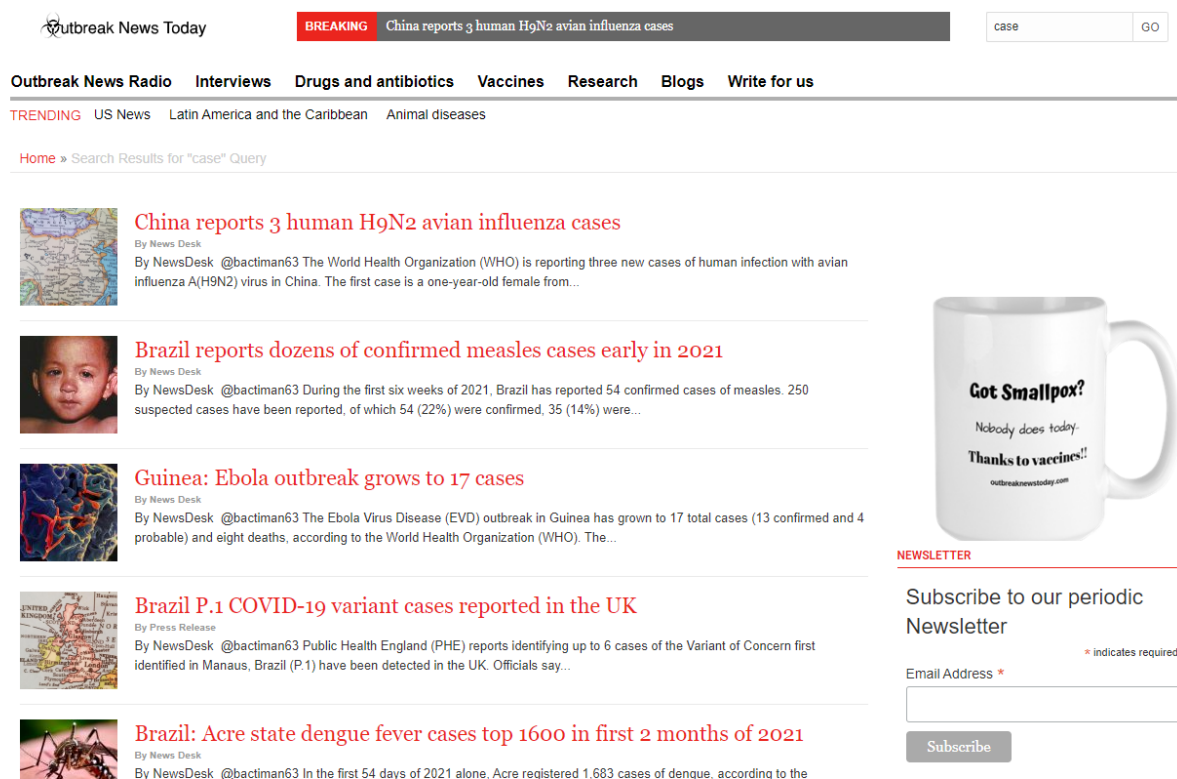
- No states parameter: API returns log with status code 200 and all restrictions of all states
- Empty states parameter: API returns log with error code 400
- Single states parameter: API returns log with status code 200 and corresponding restrictions of corresponding state
- Multiple states parameters: API returns log with status code 200 and corresponding restrictions of the corresponding states

## Web Scraper

Two web scrapers have been used in order to extract data from relevant data sources to create our application. The Web Scraper utilises Python and the corresponding BeautifulSoup library. The BeautifulSoup library allows for the parsing of structured data to easily access data within the HTML. As our chosen datasource, <http://outbreaknewstoday.com/>, contains a series of articles related to disease outbreaks, the web scraper will need to act as a crawler and dynamically traverse through multiple articles in order to retrieve relevant information.

Since the scraper is used to extract the data of specific articles within the datasource, it utilised the search functionality of the website to further filter the articles. which post within the website contains information on detected cases within the country. To ensure that the web scraper can access the relevant posts, it will start by searching "case" within the search field and then traverse all of the posts within the search field. This would increase the chance that the scraper encounters posts with relevant information. Therefore the starting page the web scraper will begin is <http://outbreaknewstoday.com/?s=case>.

The Web Scraper will then access these posts and extract the relevant information. These are the date of the article, headline, main text, URL of the article and the accompanying disease reports which have the information of the diseases, syndromes, event date and locations. With this information the disease report can then be added to the database as a simple JSON object.



The screenshot shows the Outbreak News Today website. At the top, there is a navigation bar with links: Outbreak News Today, BREAKING (China reports 3 human H9N2 avian influenza cases), CASE, and GO. Below this is a secondary navigation bar with links: Outbreak News Radio, Interviews, Drugs and antibiotics, Vaccines, Research, Blogs, and Write for us. A trending section lists: TRENDING US News, Latin America and the Caribbean, and Animal diseases. The main content area shows search results for "case" Query. The results include:

- China reports 3 human H9N2 avian influenza cases** (By News Desk, @bactiman63). The World Health Organization (WHO) is reporting three new cases of human infection with avian influenza A(H9N2) virus in China. The first case is a one-year-old female from...
- Brazil reports dozens of confirmed measles cases early in 2021** (By News Desk, @bactiman63). During the first six weeks of 2021, Brazil has reported 54 confirmed cases of measles. 250 suspected cases have been reported, of which 54 (22%) were confirmed, 35 (14%) were...
- Guinea: Ebola outbreak grows to 17 cases** (By News Desk, @bactiman63). The Ebola Virus Disease (EVD) outbreak in Guinea has grown to 17 total cases (13 confirmed and 4 probable) and eight deaths, according to the World Health Organization (WHO). The...
- Brazil P.1 COVID-19 variant cases reported in the UK** (By Press Release, @bactiman63). Public Health England (PHE) reports identifying up to 6 cases of the Variant of Concern first identified in Manaus, Brazil (P.1) have been detected in the UK. Officials say...
- Brazil: Acre state dengue fever cases top 1600 in first 2 months of 2021** (By News Desk, @bactiman63). In the first 54 days of 2021 alone, Acre registered 1.683 cases of dengue, according to the...

On the right side of the page, there is a mug with the text: "Got Smallpox? Nobody does today. Thanks to vaccines!! outbreaknewstoday.com". Below the mug is a newsletter subscription form with the heading "NEWSLETTER" and the text "Subscribe to our periodic Newsletter". The form includes an "Email Address \*" field and a "Subscribe" button. A small note indicates "\* indicates required".

Information will be extracted regularly every day at 1am to ensure that data is current and up to date. This data will be stored in the database for use for the frontend.

There will be different input parameters for the API which will extract relevant articles based on those parameters and output information about the article in the following manner:

article {5}
url string
date_of_publication string
headline string
main_text string
▼ reports array[object] {4}
diseases array[string]
syndrome array[string]
event_date string
locations array[string]

The web scraper to extract data from the above data source is currently not fully functional. In order to accommodate for this complication that was encountered, mock data was manually added to our MongoDB database in order to simulate the web scraper.

```

_id: ObjectId("60518a9a58fc9e21847d1551")
url: "http://outbreaknewstoday.com/delaware-reports-1st-case-of-covid-19-var..."
date_of_publication: "2021-03-13"
headline: "Delaware reports 1st case of COVID-19 variant B.1.351"
main_text: "The Delaware Division of Public Health (DPH) announced the first confi..."
reports: Array
  0: Object
    diseases: Array
      0: "coronavirus"
    syndromes: Array
      0: "syndrometest1"
    event_date: "2021-03-13"
    locations: Array
      0: "Delaware"

```

```

_id: ObjectId("60519a3658fc9e21847d1554")
url: "http://outbreaknewstoday.com/typhoid-taiwan-reports-1st-domestic-case-..."
date_of_publication: "2020-03-13"
headline: "Typhoid: Taiwan reports 1st domestic case of 2021"
main_text: "The Taiwan CDC reported Tuesday the first domestic case of typhoid fev..."
reports: Array
  0: Object
    diseases: Array
      0: "typhoid"
    syndromes: Array
      0: "syndrometest2"
    event_date: "2020-03-13"
    locations: Array
      0: "Taiwan"

```

```

_id: ObjectId("60519a8a58fc9e21847d1555")
url: "http://outbreaknewstoday.com/south-dakota-reports-1st-uk-coronavirus-v..."
date_of_publication: "2019-03-13"
headline: "South Dakota reports 1st UK Coronavirus Variant B.1.1.7 cases"
main_text: "The South Dakota Department of Health (SD-DOH) has confirmed that the ..."
reports: Array
  0: Object
    diseases: Array
      0: "coronavirus"
    syndromes: Array
      0: "syndrometest3"
    event_date: "2019-03-13"

```

An additional web scraper has been created in order to extract information from government restriction websites. This will be required as the planned application to be implemented focuses on travelling between states in Australia and thus will require information about travel restrictions and covid compliance rules within each state.

This data was extracted for each state from the following websites.

State	Data Source
NSW	<a href="https://www.nsw.gov.au/covid-19/what-you-can-and-cant-do-under-rules/border-restrictions/public-transport">https://www.nsw.gov.au/covid-19/what-you-can-and-cant-do-under-rules/border-restrictions/public-transport</a>
ACT	<a href="https://www.covid19.act.gov.au/community/travel">https://www.covid19.act.gov.au/community/travel</a>
NT	<a href="https://coronavirus.nt.gov.au/travel/quarantine">https://coronavirus.nt.gov.au/travel/quarantine</a>
QLD	<a href="https://www.qld.gov.au/health/conditions/health-alerts/coronavirus-covid-19/stay-informed/travel-advice">https://www.qld.gov.au/health/conditions/health-alerts/coronavirus-covid-19/stay-informed/travel-advice</a>
SA	<a href="https://www.covid-19.sa.gov.au/restrictions-and-responsibilities/travel-restrictions">https://www.covid-19.sa.gov.au/restrictions-and-responsibilities/travel-restrictions</a>
TAS	<a href="https://www.coronavirus.tas.gov.au/travellers-and-visitors">https://www.coronavirus.tas.gov.au/travellers-and-visitors</a>
VIC	<a href="https://www.coronavirus.vic.gov.au/covidsafe-travel-victoria">https://www.coronavirus.vic.gov.au/covidsafe-travel-victoria</a>
WA	<a href="https://www.wa.gov.au/organisation/department-of-the-premier-and-cabinet/covid-19-coronavirus-travel-and-quarantine#w-a-state-border-closure">https://www.wa.gov.au/organisation/department-of-the-premier-and-cabinet/covid-19-coronavirus-travel-and-quarantine#w-a-state-border-closure</a>

Similar to the first web scraper, it also will have different input parameters for the API. Depending on the input parameters passed to the API, relevant restrictions for states will be provided. A single state or multiple states can be provided in the parameters in order to retrieve the restrictions. If no states are passed to the parameters then all the restrictions for the states will be returned.

It was difficult to maintain the consistency of the json output files as most of the government sites had different restrictions and were structured in different ways. For example, some websites displayed the rules within the state whereas other websites consisted of links to additional websites in order to obtain the rules. In this case, the URL for those websites were outputted in the json file. In most cases, the json file was outputted in the following manner:

```
state_restriction {2}
  state string
  rules array[object] {4}
    rule # string
    date effective string
    title string
    link string
```

Some of the government websites did not contain any dates for which the restrictions would be effective from and assumed that the restrictions would become effective from the last date that the website was updated. In the case, where there was no date for which the website was last updated, then restrictions would be assumed to be effective from the current date.

An example of the web scraper for NSW restrictions:

```
{
  "state": "New South Wales",
  "rules": [
    {
      "rule 1": "Wear a face mask. Wearing a mask is mandatory on public transport in the Greater Sydney, Blue Mountains, Central Coast and Wollongong areas. Children 12 years and under are exempt but encouraged to wear a face mask. Mask must be worn on the concourse, at coach/bus stops, and while travelling. Penalties will apply.",
    },
    {
      "rule 2": "Plan ahead and avoid travelling in the busiest times if you can. If you are not already using public transport in the peak, please do not start now. Services are already close to capacity to allow for physical distancing at these times. Off peak times are between 10am and 3pm.",
    },
    {
      "rule 3": "Check the capacity of services before you travel with the Trip Planner, transport apps or social media to see which services have space available to maintain physical distancing.",
    },
    {
      "rule 4": "Practise good personal hygiene and maintain physical distancing.",
    },
    {
      "rule 5": "Look for the green dots. These have been placed to show you the safest places to sit and stand.",
    },
    {
      "rule 6": "Observe capacity limits: physical distancing capacity limits on services means you may be asked to wait for the next service.",
      "date effective": "2021-01-29T12:00:00Z"
    },
    {
      "rule 6": "Observe capacity limits: physical distancing capacity limits on services means you may be asked to wait for the next service.",
      "date effective": "2021-01-29T12:00:00Z"
    }
  ]
}
```

## MongoDB Database

The MongoDB database will be hosted on MongoDB Atlas, which is a dedicated cloud-based database service designed for MongoDB users. The backend API will link with the use of the PyMongo library which allows for ease of integration with the dedicated database.

The screenshot shows the MongoDB Atlas interface. The top navigation bar includes 'Disease Report Database', 'Atlas', 'Realm', and 'Charts'. The left sidebar lists 'DATA STORAGE' (Clusters, Triggers, Data Lake) and 'SECURITY' (Database Access, Network Access, Advanced). The main content area is titled 'Clusters' and shows details for 'Cluster0' (Version 4.4.4). It includes buttons for 'CONNECT', 'METRICS', and 'COLLECTIONS'. The cluster tier is 'M0 Sandbox (General)', the region is 'AWS / Sydney (ap-southeast-2)', and the type is 'Replica Set - 3 nodes'. The linked realm app is 'None Linked'. On the right, there are sections for 'Operations' (R: 0, W: 0) and 'Connections' (8), both with 'Last 6 Hours' history.

The screenshot shows the MongoDB Compass interface. The left sidebar shows the 'database' expanded, with 'articles' selected. The main content area displays the 'database.articles' collection with a size of 5.84KB, 10 documents, and a total index size of 36KB. The interface includes tabs for 'Find', 'Indexes', 'Schema', 'Anti-Patterns', 'Aggregation', and 'Search Inc'. A filter bar shows the filter '{ "filter": "example" }'. Below, the 'QUERY RESULTS 1-10 OF 10' are displayed as JSON documents. Each document contains fields for '\_id', 'url', 'date\_of\_publication', 'headline', 'main\_text', and 'reports' (an array).

```

{
  "_id": ObjectId("60518a9a58fc9e21847d1551"),
  "url": "http://outbreaknewstoday.com/delaware-reports-1st-case-of-covid-19-var...",
  "date_of_publication": "2021-03-13",
  "headline": "Delaware reports 1st case of COVID-19 variant B.1.351",
  "main_text": "The Delaware Division of Public Health (DPH) announced the first confi...",
  "reports": Array
}

{
  "_id": ObjectId("60519a3658fc9e21847d1554"),
  "url": "http://outbreaknewstoday.com/typhoid-taiwan-reports-1st-domestic-case-...",
  "date_of_publication": "2020-03-13",
  "headline": "Typhoid: Taiwan reports 1st domestic case of 2021",
  "main_text": "The Taiwan CDC reported Tuesday the first domestic case of typhoid fev...",
  "reports": Array
}

{
  "_id": ObjectId("60519a8a58fc9e21847d1555"),
  "url": "http://outbreaknewstoday.com/south-dakota-reports-1st-uk-coronavirus-v...",
  "date_of_publication": "2019-03-13",
  "headline": "South Dakota reports 1st UK Coronavirus Variant B.1.1.7 cases",
  "main_text": "The South Dakota Department of Health (SD-DH) has confirmed that the ...",
  "reports": Array
}

```



## Justification of Implementation

### Frontend

#### **Vue:**

Vue is a Javascript framework which utilises various tools for user interface development. Our main reason for choosing Vue for the frontend development was due to its flexibility, its Model View ViewModel (MVVM) structure and its easy integration and compatibility with pre-existing templates that can be found publicly. The flexibility and availability of the Vue templates allow us to present our software in a more captivating interactive platform which helps us maximise our software's features and services for the users. The MVVM structure of Vue provides us with two-way-bind data within the views which means that any changes made to the model will be reflected in the views and vice-versa. This results in a safer and easier abstraction from models.

#### **Bootstrap4:**

Bootstrap is a free CSS and HTML based frontend framework that is used for creating dynamic websites and web applications. The main reason for choosing Bootstrap4 was due to its adaptable structure, easy and fast responsiveness as well as its customizability. The availability of pre-built codes allows for reusability and reduces code complexity which is beneficial during this limited time frame. Bootstrap4 has a large library of functions which will provide us many elements to work with when producing a visually appealing frontend.

### Backend

#### **MongoDB:**

MongoDB is a free NoSQL document-oriented database platform that is compatible with many languages including Python. It stores data in JSON-like documents which eases linking of data to the backend for processing, storing, retrieving and updating information. Since data will be saved in JSON files, it is beneficial when working with data that is constantly being updated or new data. Furthermore, MongoDB is very easy to scale out and allows future possible expansion of the web application around the world.

MongoDB was chosen as it would provide an easily integrated database as there is a dedicated python library Pymongo which allows for easy manipulation of the database through python. MongoDB also has a cloud based server MongoDB Atlas which allows the database to be easily hosted and also manipulated. Some screenshots of the platform has been displayed in previous sections of the report

#### **Flask:**

Flask is a Python web development framework and is one of the most popular frameworks for Python. This means that there are abundant resources and documentation for us to work with and its capability to be used in conjunction with MongoDB. Python will be used for building the backend where it will handle API calls for storing and retrieval of data from and to the database.

## Complications and Shortcomings

### Platform of API

Initially, the API was planned to be deployed using AWS. However, during deployment the following errors were experienced below:

Recent events			Show all
			< 1 >
Time	Type	Details	
2021-03-18 18:04:08 UTC+1100	ERROR	Create environment operation is complete, but with errors. For more information, see troubleshooting documentation.	
2021-03-18 18:03:15 UTC+1100	INFO	Added instance [i-Ofd1079727004a312] to your environment.	
2021-03-18 18:03:05 UTC+1100	INFO	Command execution completed on all instances. Summary: [Successful: 0, Failed: 1].	
2021-03-18 18:03:05 UTC+1100	ERROR	[Instance: i-Ofd1079727004a312] Command failed on instance. Return code: 1 Output: Engine execution has encountered an error..	
2021-03-18 18:03:02 UTC+1100	ERROR	Instance deployment failed. For details, see 'eb-engine.log'.	

Due to the limited time constraints, the API was deployed using Azure which is a similar cloud computing platform to AWS. Azure retains a range of cloud services including computing, analytics, storage and networking that made AWS initially appealing, whilst also being easier to implement than AWS.

### Web Scraper

As of now, the web scraper that is critical to the operation and full functionality of the API is not available. This has occurred due to a series of miscommunications in the group, primarily between the lead developer of the web scraper and the rest of the group.

To rectify this mistake, the rest of the group has worked tirelessly to create a web scraper that more appropriately fits the task specifications whilst synchronising with the existing API. Due to the time constraints of this particular deliverable, the substitute web scraper has not been finished.

Currently, the web scraper for <http://outbreaknewstoday.com/>, returns information in a JSON file format as shown below.

```

{
  "url": "http://outbreaknewstoday.com/paraguay-dengue-cases-continue-to-increase-94771/",
  "date_of_publication": "March 17, 2021",
  "headline": "Paraguay: Dengue cases continue to increase",
  "main_text": "By NewsDesk\u00a0\u00a0@infectiousdiseasenewsDengue cases continue to increase in Paraguay in 2021. More than 6,000 suspected a",
  "reports": {
    "infected_numbers": "85",
    "disease": "",
    "syndromes": ", fever",
    "event_date": "",
    "locations": ""
  }
},
{
  "url": "http://outbreaknewstoday.com/non-viral-hepatitis-cases-possibly-linked-to-bottled-alkaline-water-81170/",
  "date_of_publication": "March 16, 2021",
  "headline": "Non-Viral hepatitis cases possibly linked to Bottled Alkaline Water",
  "main_text": "On March 13, the U.S. Food and Drug Administration was alerted to five cases of acute non-viral hepatitis (resulting in acute l",
  "reports": {
    "infected_numbers": "9",
    "disease": "unknown",
    "syndromes": ", fever, vomiting, nausea, fatigue",
    "event_date": "",
    "locations": ""
  }
},
{
  "url": "http://outbreaknewstoday.com/brazil-additional-monkeys-die-from-yellow-fever-in-santa-catarina-two-human-cases-in-2021/",
  "date_of_publication": "March 13, 2021",
  "headline": "Brazil: Additional monkeys die from yellow fever in Santa Catarina, Two human cases in 2021",
  "main_text": "By NewsDesk\u00a0\u00a0@bactiman63Santa Catarina already accounts for the death of 64 monkeys due to yellow fever.\u00a0The lat",
  "reports": {
    "infected_numbers": "9",
    "disease": "yellow fever",
    "syndromes": ", fever, vomiting, nausea, back and body pain, chills",
    "event_date": "",
    "locations": ""
  }
}

```

This web scraper has not been integrated with the API due to time constraints.

Despite this difficulty, the API itself can be partially used and hence tested. However, it requires test data, which has been provided.

## Endpoints

When the API was first implemented, there were a large number of endpoints. However, after receiving the appropriate feedback that the endpoints were unnecessary and overly numerous, many of them were either deleted or synthesised into one endpoint.

For the sake of redundancy, this section will not be elaborated further upon as it was previously extensively explained under the “Python API” section of this report.

## Potential Future Changes

Potential changes that could be made in the future include

- In order to increase scalability and usability, the API could be altered to allow users to search travel restrictions across all countries rather than just the Australian states (which is currently what has been implemented)
- In order to improve the accuracy of information for users, the google maps API could be integrated in order to search for locations of outbreaks or diseases more precisely
- A PUT articles endpoint could be implemented to allow users to add their articles into the database. Users would have to be authenticated through an encrypted authentication key to prevent irrelevant data.