D1 Design Details

After the COVID-19 global pandemic, it raises people's attention on the early detection of infectious disease. The Integrated Systems for Epidemic Response(ISER) at UNSW is a organization that working on this area. The EpiWATCH system they developed using open-source data detected many infectious diseases in very early stage and contribute towards global epidemic response.

In our project, we plan to develop a platform and an API towards our benchmark EpiWATCH system. The API we developed would be able to access by other teams in SENG3011 to find disease reports in CDC.

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

   Endpoint naming conventions:

   URLs as resources as nouns
   "RESTful URIs should refer to a resource that is a thing (noun) instead of referring to an action (verb) because nouns have properties which verbs do not have – similar to resources have attributes."
   – RESTfulAPI.net

   We will pass parameters to endpoints by using route and body, we put location parameters within the URL route, key terms and period of interest parameters in the HTTP request body. In this way, we make the endpoint URL intuitive and clean, keeping more complex parameters in the request body JSON object makes parameters more readable to humans.

   For versioning API, we put the version info in the URL of the request, this is the most straightforward way to resolve API versioning problem. It is transparent for user to see that which version of API they are using.

   For the response of our API, we want our API always returns a 200 as long as users pass parameters by following the documentation. To tell a request succeed or not, there are a "is_error" and a "message" fields with the response object to indicate the status of the request. By doing this, we have the ability to tell users more detailed message about the request instead of just an error code. If users try to access a nonexistent endpoint, an customized 404 response will be returned.

   Ability to run it in Web service mode:

   Run our API in Web service mode would mean our API is open for access to the internet and follows the RESTful protocol. It also needs to be fully documented so the people want to use it will know how and what they will get back.

   To deploy our API to public, we choose to use AWS API Gateway. It will provide a domain name to our API and everyone from anywhere can use this domain name to access our API. The reasons we choose AWS to deploy our API are free and easy to use, provide fast access from almost all region and gives us lots of options on the integration type for example AWS lambda, another AWS service and VPC link. This means we get to choose the type of endpoint and may use multiple types of endpoints. This could also mean slightly more to learn than others like DigitalOcean and Microsoft Azure, but AWS provides a great architecture and many more services we can use, like straight access to AWS DynamoDB from API. And nowadays AWS is many companies' choice, so learning to use AWS might benefit all our team members in later careers.

   For the documentation of our RESTful API, we plan to follow the OpenAPI Specification and use Swagger to help us design and document. By using Swagger, it makes sure we follow the RESTful protocol and build a documentation which both human can computer can understand. Beside that, it is also supported by AWS API Gateway, we can just import the API design and documentation from Swagger into AWS API Gateway.

   In order to collect data from the Centers for Disease Control and Prevention (CDC), developing our own Scraper is indispensable.

   The Scraper will have the basic function to simultaneously process three input parameters, which are period of interest, key terms and location. Based on the three parameters, the Scraper will be able to find related reports or articles published on the CDC website and store the collected data into a specific data format.

   In the design process, we are aiming to build a flexible Scraper. The flexible Scraper Can bring convenience in adding new functions and maintain processes. The Scraper can adapt to more conditions during the further. Such as taking even more parameters, working on different data source websites.

   By using website Scraper tools such as BeautifulSoup, we have the ability to extract data from object data sources and get to choose how we store the collected data.

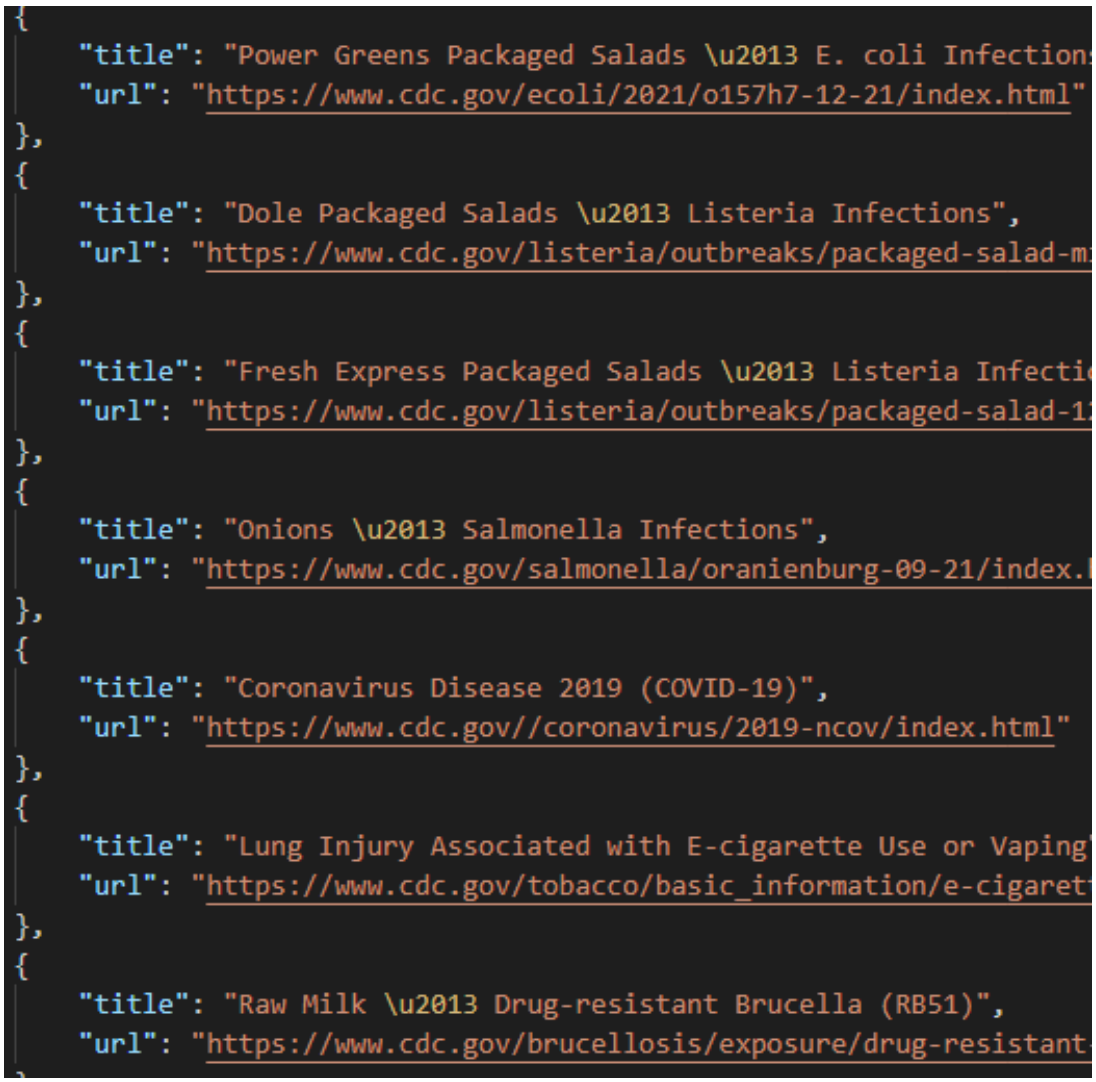
   Here is an example of us using the Scraper to get access to CDC and store the collected data into json format.

```
url = "https://www.cdc.gov/outbreaks/"
request = urllib.request.Request(url)
html = urllib.request.urlopen(request).read()

soup = BeautifulSoup(html,'html.parser')
main_table = soup.find("ul",attrs={'class':'list-bullet feed-item-list'})
links = main_table.find_all("a",class_="feed-item-title")

def is_absolute(url):
    return bool(urlparse(url).netloc)

extracted_records = []
for link in links:
    title = link.text

    url = link['href']
    if not is_absolute(url):
        # use urljoin to join url
        url = "https://www.cdc.gov/" + url

    record = {
    'title':title,
    'url':url
    }
    extracted_records.append(record)
```

U.S.-Based Outbreaks

Recent investigations reported on CDC.gov

- Power Greens Packaged Salads – *E. coli* Infections
  ANNOUNCED DECEMBER 2021
- Dole Packaged Salads – *Listeria* Infections
  ANNOUNCED DECEMBER 2021
- Fresh Express Packaged Salads – *Listeria* Infections
  ANNOUNCED DECEMBER 2021
- Onions – *Salmonella* Infections
  ANNOUNCED SEPTEMBER 2021
- Coronavirus Disease 2019 (COVID-19)
  ANNOUNCED JANUARY 2020
- Lung Injury Associated with E-cigarette Use or Vaping
  ANNOUNCED AUGUST 2019
- Raw Milk – Drug-resistant *Brucella* (RB51)
  ANNOUNCED FEBRUARY 2019



As you can see, The scraper collected the title name of the reports from CDC website, and got the corresponding URL of the report, The scraper also stored the collected data into json format.

After all, the Scraper is still facing a series of challenges such as processing speed and information accuracy. Currently using website Scraper tools such as BeautifulSoup, even with a sample requirement of scraping, it still takes 4-5 sec to collect the data and facing lack of accuracy.

In the modern day the user is seeking on the speed and accuracy, and the lack of speed and accuracy needs to be addressed urgently. In the future we could use artificial intelligence to help us sift through the information we need and could help the Scraper to increase the speed and accuracy.

2. 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)

   API stands for application programming interface, which is a set of definitions and protocols for building and integrating application software.

   When the back-end logic is implemented and deployed on a server, the back-end will have endpoints exposed to other applications to use. With restful API, to pass parameters to an API module, HTTP calls will be used, the parameters can be placed in route, body.

   For the initial design, we planed to use a database for the API. The scraper will fetch published date, reports and location from the CDC website on a daily bases, these data will be stored in the a database table, when parameters are passed to the API, the back-end will select data from the database instead of scrap the website, it will speed up the user query processing.

   Database example:

id	location	published_data	reports_url

1	Australia, Sydney	2022-01-01T00:00:00	<URL>
2	Australia, Brisbane	2022-01-02T07:01:12	<URL>

API parameter example:

Parameter location	**example**	**response**

Route and Body	GET http:/<Our Domain Name>/<API-Version>/Reports /Australia/Sydney
	{
	“key_terms“: [“Covid-19“, “mild“],
	“start date“: “2021-10-01T00:00:00“,
	“end date“: “2021-11-01T00:00:00”
	}
	“Australia“ and “Sydney“ are two route parameters, they are used to specify the location.
		success response
		{
		“status“: 200,
		“is_error“: false,
		“results“: [
		{
		“url“: <URL-String>,
		“publication date“: <Date-String>,
		“headline“: <String>,
		“main text“: <String>,
		“reports“: [<object: report>]
		}
		],
		“message“: ““
		}
		fail response
		{
		“status“: 200,
		“is_error“: true,
		“results“: [],
		“message“: “internal error“
		}
		{
		“status“: 200,
		“is_error“: true,
		“results“: [],
		“message“: “bad request“
		}

Body	GET http:/<Our Domain Name>/<API-Version>/Reports
	{
	“location“: “Sydney“,
	“key terms“: [“Covid-19“, “mild“],
	“period of interest“: {
	“start date“: “2021-10-01T00:00:00“,
	“end date“: “2021-11-01T00:00:00“
	}
	}
	“Period of interest” is used to specify the time requirement of reports.	

Responses components:

- “is_error“ is used to indicate whether the request is processed correctly.
- “results“ is a list of report object, each report object will contain information about one specific report. Theses report object are collected form our scraper, the scraper will find out CDC reports which are in the user specified period of interest and contains all the key terms.
- “message“ is used to tell users what goes wrong if an error occurred.

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

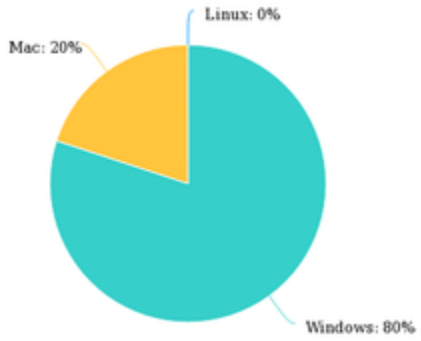
Chosen Technologies:

- **Development and Deployment Environment**: Linux

- **Primary Language:** Python3
- **Database:** PostgreSQL
- **Server and Serverless Architecture:** Serverless - AWS Lambda
- **Frontend Framework:** React
- **Test Framework:** Pytest
- **Specific Libraries:**
    - json
    - math
    - pandas
    - scrapy
    - pytorch
    - psycopg2
    - requests
    - datetime
    - selenium
    - beautifulsoup4
    - flask-restX

Result of Preference Questionnaire:
Development and Deployment Environment: Linux

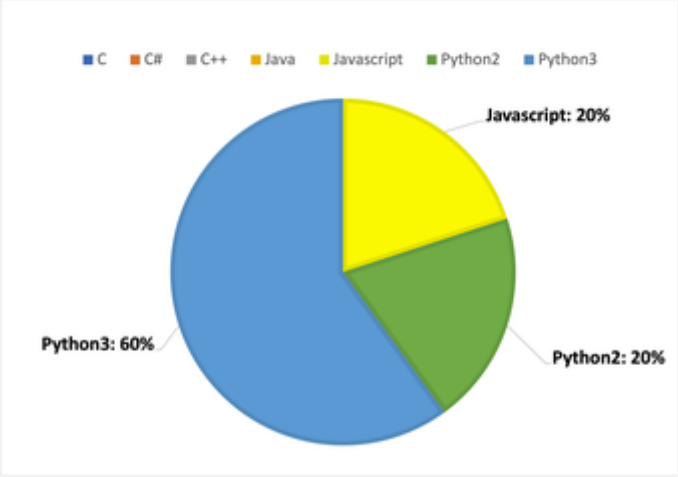
By analysing the result of questionnaires, it is easy to find that we have both Windows and MacOS users in our group. At the meantime, considering whether the software can run successfully on the CSE Server, Linux would be the best choice, avoiding development environment conflicts.



Shortcomings: Due to the free and open source nature of Linux, there is no specific support vendor for Linux, which means there is no after-sales service. Linux, as an older system, lags far behind Windows and MacOS in the graphical interface. However, in this project, we will not change or affect the system files, and rarely involve the use of graphical interfaces. So while Linux has its drawbacks, they don't have a negative impact on this project.

Primary Language: Python3

All members in the group has experience of Python3 from previous courses. Comparing with:



- **C**: Coding with Python3 is much faster and easier since Python is popular as an interpreted scripting language.
- **C#**: Code of Python3 is much cleaner and easier to write with Java and C/C++ extensions.
- **C++**: Python3 is relatively easy to learn, the standard library is powerful and extensive, and development is faster.
- **Java**: Python3 is cheaper to learn, the code is cleaner, the development time is shorter, and the variable types are dynamic, meaning that different types of data are stored at run time depending on the needs of the application.
- **Javascript**: Whereas ecosystem of Javascript is mostly focused on the front end, Python3's ecosystem is all over the computer, so Python is much more efficient.
- **Python2**: Except that Python2 is officially no longer maintained, Python3 uses Unicode characters as its native support to make data types simpler and string types uniform compared to Python2's use of ASCII as the default encoding.

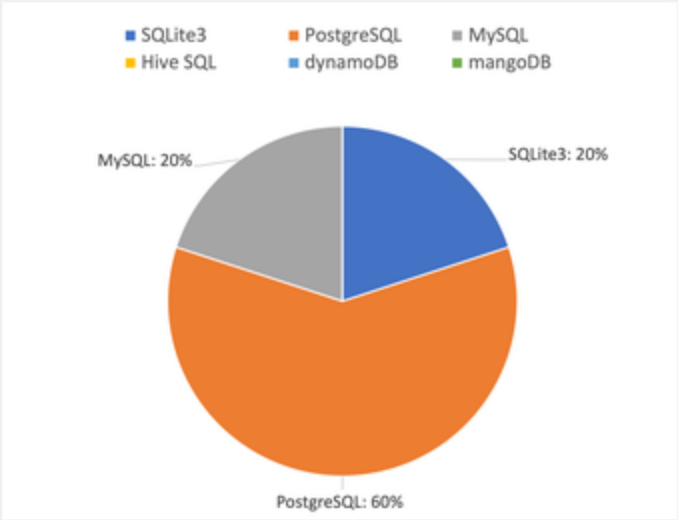
Shortcomings: The biggest obstacle to using Python3 in this project is that Python3 does not check variable types at compile time, which can easily lead to errors caused by variable type conflicts during programming. However, python3, as a primary language preference, allows team members to interpret and describe what they edit properly through **PyDoc**, avoiding variable type conflicts.

Database: PostgreSQL

All members has participated COMP3311, which is focused on database using PostgreSQL. Comparing with other SQL languages, PostgreSQL has the following advantages:

- PostgreSQL is an open source object-relational database management system,
- PostgreSQL focuses on extensibility and standards compliance,
- PostgreSQL is hailed as the "most advanced open source database" in the industry,
- PostgreSQL is completely free using the BSD protocol,
- PostgreSQL is multi-process and therefore significantly faster than threaded SQL languages such as MySQL.
- PostgreSQL supports a wide range of data types, including GIS collection types, JSON, data, and dictionaries.

Shortcomings: PostgreSQL is not as fast or popular as MySQL, but in this project, we focus on PostgreSQL's scalability, which means that the database can perform custom procedures. PostgreSQL is the best choice for possible future database migration.



Server and Serverless Architecture: Serverless - AWS Lambda

Both Server and Serverless Architecture have their advantages and disadvantages, but in this project, we pay more attention to Serverless Architecture's low cost, scalability and simplified backend programming. The security feature it provides is also important to us, so that we don't need to worry about security issue and put more time on develop our product. But since non of us learned about using serverless architecture, it definitely cost us more time to learn.

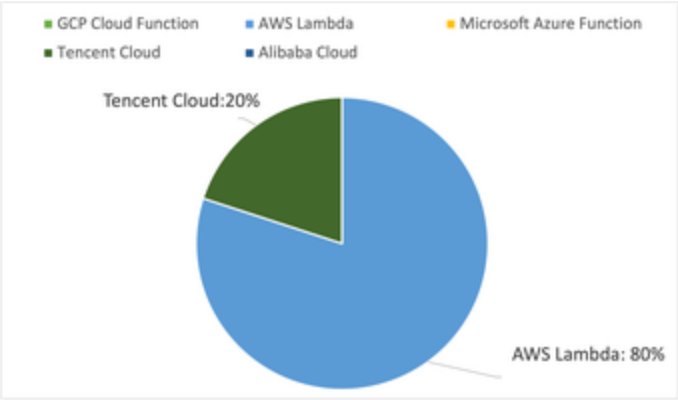
GCP Cloud Function, AWS Lambda and Microsoft Azure Function are the three famous Serverless Architectures globally. All of three Serverless Architectures are cheap to learn, scaled automatically and easy to trigger, AWS Lambda using API Gateway and GCP Cloud Function and Microsoft Azure Function are triggered by HTTP. However, the free trail of AWS Lambda is more suitable for this project as it provides one million free requests per month and 400,000 GB-seconds of compute time per month.

Shortcomings: When using AWSLambda, some cold start problems may be encountered, that is, the internal structure of AWSLambda must extract the entire source code to generate the execution path, and according to official sources, Python has the shortest cold start time, so the impact of cold start is relatively small to some extent. At the same time, reducing the code size when necessary is also the easiest way to reduce cold starts.

Frontend Framework: React

Chosen React as frontend framework not only because React has the largest number of users, but also because of its advantages. Comparing with:

- **Vue**: While React and Vue have many similarities, the React ecosystem is much richer, most notably when Vue uses templates, and React prefers JSX programming for rendering.
- **Angular**: React is relatively cheap to learn and has more freedom to choose from third-party libraries.
- **jQuery**: React manages project code more fully, avoids building complex program structures, and uses functional programming thinking to solve user interface rendering problems, greatly improving development efficiency.
- **Flask**: Flask's most prominent template, Jinja, is better suited for smaller projects, while React can be used for lightweight templates and, when necessary, for larger templates.
- **Django**: React is often used in combination with Django to separate the frontend from the backend. React is usually the frontend and Django is responsible for the backend. React is lighter than Django.



Shortcomings: React's minimum reuse unit is components and the target is UI components, so it is not suitable for a complete framework on its own. If the project is too large, a combination with other frameworks can be considered if necessary.

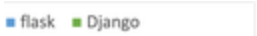
Test Framework: Pytest

Based on the previous COMP1531 experience, all members in the group are interested in using Pytest. Also, by comparing with the following test frameworks, advantages of Pytest can be found:

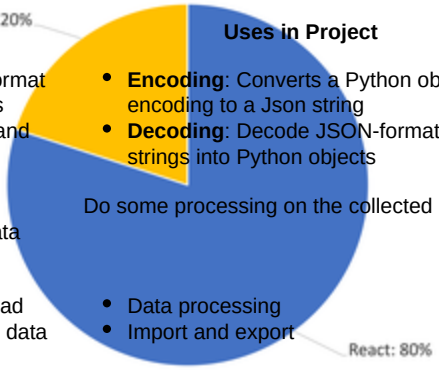
- **Unittest:** Pytest is a simple format that allows users to be confident in Unittest style test cases without modifying any of their code for better compatibility.
- **HtmlTestRunner:** Although HtmlTestRunner can generate test reports after testing, HtmlTestRunner itself is a copy of TextTestRunner, which belongs Unittest, and, therefore, is out of the consideration.

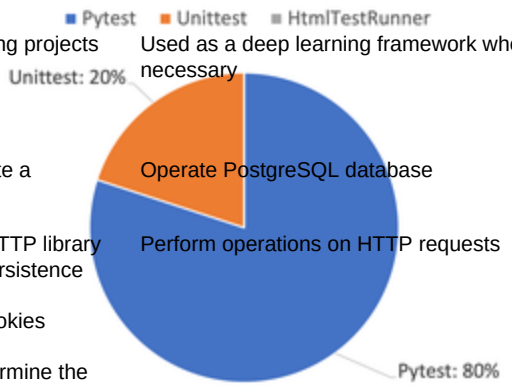
Shortcoming: Pytest is a third-party testing framework for Python. It is an extension framework based on UnitTest. Although UnitTest is more compatible, PyTest is much easier to use, has more plug-ins, and supports failure reruns. At the same time, as long as python3 and PyTest versions are consistent, plug-ins and function compatibility problems caused by version conflicts can be avoided.

Specific Libraries:



Python3 Libraries	Characteristics	Uses in Project
json	1. Lightweight text data interchange format 2. Easy to read and write with humans 3. It is also easy for machine parsing and generation	• **Encoding**: Converts a Python object encoding to a Json string • **Decoding**: Decode JSON-formatted strings into Python objects
math	1. Solve math operations effectively 2. It provides a certain basis for big data processing	Do some processing on the collected data
pandas	To a certain extent, it is convenient to read and write tables and analyse and merge data	• Data processing • Import and export
scrapy	1. Crawl website data 2. Extract structural data 3. An application framework	Crawl pages when necessary
pytorch	1. Build a library for deep learning projects 2. Clear grammar 3. Concise API 4. Easy to debug features	Used as a deep learning framework when necessary
psycopg2	A third-party library used to operate a PostgreSQL database	Operate PostgreSQL database
requests	1. The only Non-gmO Python HTTP library 2. Support HTTP connection persistence and connection pooling 3. Maintains a session using cookies 4. Upload files 5. Support to automatically determine the encoding of the response content 6. Support automatic encoding of internationalized URL and POST data	Perform operations on HTTP requests
datetime	1. It is a combination of the date and time modules 2. Contains functions and classes for date and time parsing, formatting, and arithmetic	Operate on the time set in the project





selenium	1. A Web application automation framework 2. Operate the browser through the browser driver, clicking the button 3. Use the browser driver to return the data after the operation, clicking on success, to the Selenium library	Easy to accurately understand each state of the web page
beautifulsoup4	1. Parse and process HTML and XML 2. Able to establish parsing tree according to HTML and XML syntax, and then efficiently parse the content 3. Encapsulate the professional web page format parsing part as a function 4. Several useful and quick processing functions are provided	Parsing HTML and XML, if necessary, results in documents
flask-restX	1. provides decorators for documenting APIs 2. provides Swagger API UI 3. provides easy access to multiple HTTP methods just by defining methods on resources	generate API documentation bases and provides resourceful routing

---

Questionnaire

https://aws.amazon.com/cn/blogs/china/4-solutions-to-reduce-the-cold-start-time-of-aws-lambda/?nc1=b_rp