

# SENG3011 - D1

## API-demic

Aaron Ly (z5208530)

Amelia Lau (z5209371)

Logann Guiney (z5294830)

Peter Tran (z5208980)

Kevin Well (z5214693)

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

The API module will be developed with the REST architecture. REST architectures typically run on a web service, hence we will be calling data and resources over HTTP calls and using keywords like “GET”, “POST”, “PUT”, “DELETE”,

Our service has three main components.

- A web scraper that scrapes data from our source (<https://crofsblogs.typepad.com/h5n1/>) periodically.
- A relational database that stores the data scraped by the web scraper.
- A web API to query the database and return JSON.

We intend to run the service on a cloud computing provider.

Since it is web based, we also need to host it online. The most cost efficient and simple hosting service was either Heroku or DigitalOcean. We have decided to use Heroku since it provides university students with a Student Tier.

**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).**

Parameters such as key\_terms, period of interest or location can be passed into the module to return filtered disease reports specified in the parameters criteria. So when the parameters are passed into the api module, we would then scrape our information source in order to locate reports relating to these parameters. This could include reports related to certain diseases or reports released during a certain period of time.

## Resource URL

GET /articles

## Query Parameters

Parameter	Type	Required	Description	Example
start_date	datetime	✓	The start date used to filter the articles	2015-10-01T08:45:10
end_date	datetime	✓	The end date used to filter the articles	2015-11-01T19:37:12

key_terms	string		A comma separated list of key terms used to filter the articles	Anthrax,Zika
location	string		Used to filter the articles by location	Sydney

## Example Request

- /articles?start\_date=2015-10-01T08:45:10&end\_date=2020-02-01T19:37:12&location=Sydney&key\_term=Coronavirus
  - Get all articles related to Coronavirus in China published between the first of October 2015 at 08:45:10 and the first of February 2020 at 19:37:12

## Example Response

```
{
  "url": "https://www.who.int/csr/don/17-january-2020-novel-coronavirus-japan-ex-china/en/",
  "date_of_publication": "2020-01-17 xx:xx:xx",
  "headline": "Novel Coronavirus -Japan (ex-China)",
  "main_text": "On 15 January 2020, the Ministry of Health, Labour and Welfare...",
  "reports": [
    {
      "event_date": "2020-01-03 xx:xx:xx to 2020-01-15",
      "locations": [
        {
          "country": "China",
          "location": "Wuhan, Hubei Province"
        },
        {
          "country": "Japan",
          "location": ""
        }
      ],
      "diseases": [
        "2019-nCoV"
      ],
      "syndromes": [
        "Fever of unknown Origin"
      ]
    }
  ]
}
```

Figure 1.1 Json Formatted Response

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

Implementation Language:

The API will be implemented using Python. We decided to use Python because we all have experience with it. Other languages such as Java, Javascript and C# were suggested but the lack of comfortability meant that some members would have to invest time and resources to learn those languages.

Python contains a vast range of libraries, which will be useful when creating a web API and a web scraper. We decided to use Flask as our web framework because it's lightweight and allows for flexibility in parsing parameters and organising routing for our API. It also supports JSON formatting which was important when returning a response for our clients. The other choice was Django, which has a ready to use admin framework, however it is not necessary for us and will require us to put additional resources into learning.

We are also using Python for the web scraping component of our service. A combination of the Requests and BeautifulSoup libraries will be used to scrape and parse our HTML source.

To store our data we have chosen to use a relational database system, PostgreSQL. We chose to use a relational database over a non-relational database like MongoDB so that we can form complex queries based on multiple properties of an object. We specifically chose PostgreSQL over other relational database systems because of its flexibility with custom data types, large number of native data types and because we all have experience with it.

Python libraries:

- Flask: Rest-API framework
- BeautifulSoup4: HTML Parser
- Requests: HTTP Library

Development environment:

- Windows
- Mac

Deployment environment:

- Ubuntu 18.04

**Describe final architecture of your API, justify the choice of implementation, challenges addressed and shortcomings, in your Design Details report in Github. You may update management information if necessary.**

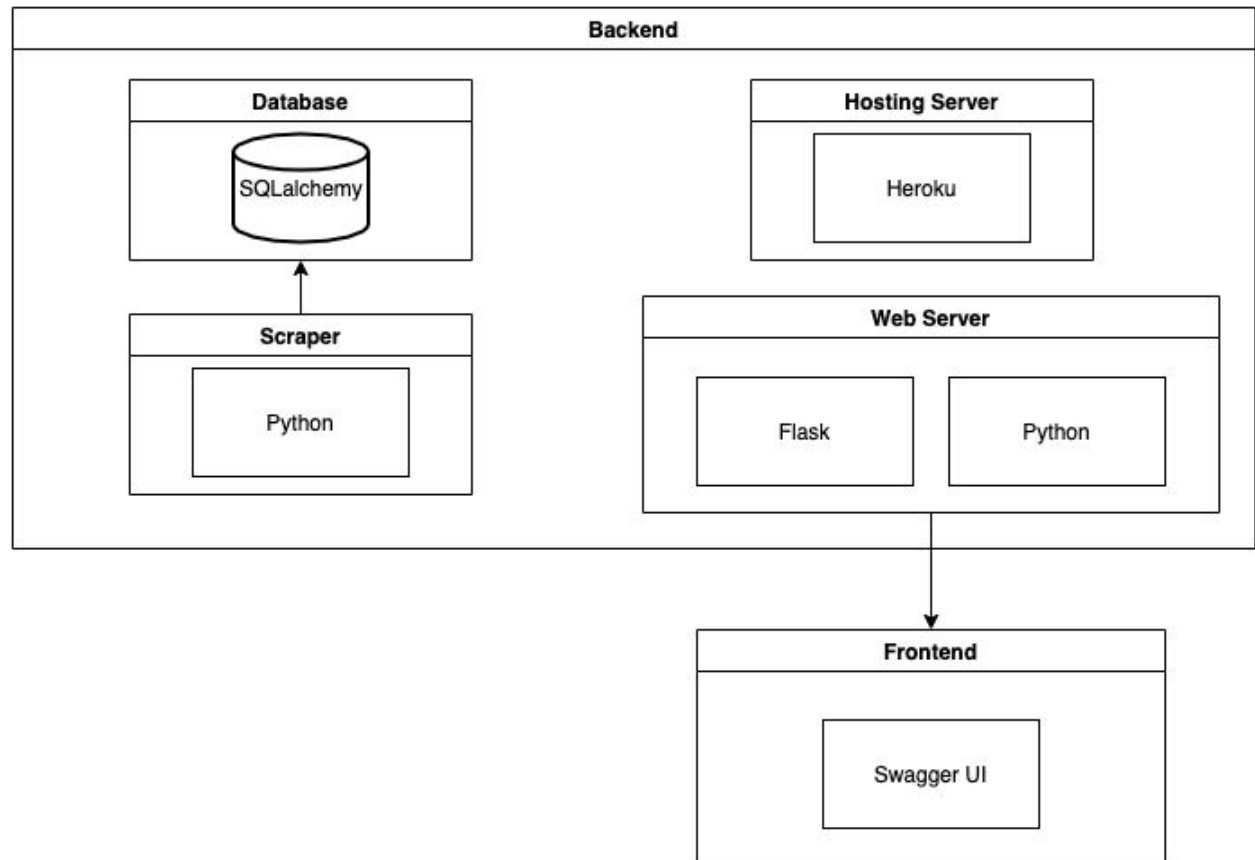


Figure 1.2 Software Architecture

Our final software architecture consists of mainly Python. As above the choice of Python allowed for many libraries, such as:

- Request, to easily parse raw HTML for our scraper
- BeautifulSoup4, library that allowed for easy scraping of the web
- Geotext, PyCountry, libraries that help detect locations
- Flask-sqlalchemy, framework for creating a ORM
- Psycopg2, Postgres Adapter for Python
- Flask-swagger-ui, library that assisted in generating a swagger documentation
- Re, Time, Datetime, Built in libraries to ease regex and date formatting

All these allowed us to scrape data, create flexible url params. An important factor was the ability to correctly identify the location, event dates and diseases in text, which

would be very hard in other languages such as JS, or Java. Python contains great libraries such as Geotext, to easier identify locations. Also if we have the time we could use Natural Languages to advance our filtering of the attributes.

For the database we decided to use PostgreSQL. The choice of PostgreSQL arises from the ability to host PostgreSQL databases on Heroku (choice of hosting server) and that Python has libraries such as SQLAlchemy to help us with connecting with Heroku and making simple queries.

Heruko was our final decision as a hosting server. The choice of Heroku over DigitalOcean was the pricing as we knew for Heruko there was a student Tier that allowed us to have perks for free. The other main reason was that a member used Heroku before and had experience as we were better off using it.

The challenges we faced were setting up the connection from databases (hosted by Heroku) to our app.py. Since an adapter was needed to connect to the Heroku database which ran on Postgres, it was required to download Postgres. Since barely anyone on our team had knowledge of setting up a server, this challenge was time consuming and tedious. However we managed to install Postgres's Python adapter psycopg2, which allows us to connect to the database and make queries. A minor problem that occurred during our project was the time for scraping data and found a method to periodically scrape data to feed our database with the latest information. The whole process of scraping around 4200+ articles took 2 hours.