COVID-19 Chatbot   
Project Report

NUS Master of Technology  
Intelligent Reasoning System (IRS)

**Group 6**

**Lim Wee Kiat A0213481M**

**Eu Jin Marcus Yatim A0124453M**

**Teoh Yee Seng A0213501B** 10-May-2020

# Table of Content

[Table of Content 2](#_Toc39005543)

[1 Executive Summary 4](#_Toc39005544)

[2 Business Justification 5](#_Toc39005545)

[3 Team Member & Contribution 6](#_Toc39005546)

[4 Project Solution 6](#_Toc39005547)

[4.1 Project Deliverables 6](#_Toc39005548)

[4.1.1 Chatbot Features 6](#_Toc39005549)

[4.1.2 Implementation Concepts 6](#_Toc39005550)

[4.1.3 Outcome 7](#_Toc39005551)

[4.2 Knowledge Representation 8](#_Toc39005552)

[4.2.1 Data Acquisition and Synchronization 8](#_Toc39005553)

[4.2.2 Decision table in Self-Assessment Tool 9](#_Toc39005554)

[4.2.3 Hospital Location Database Structure 11](#_Toc39005555)

[5 Project Architecture & Implementation 12](#_Toc39005556)

[5.1 Architecture Overview 12](#_Toc39005557)

[5.2 Process Flow 12](#_Toc39005558)

[5.3 Cognitive System 13](#_Toc39005559)

[5.3.1 Create an Agent 13](#_Toc39005560)

[5.3.2 Create Intents 13](#_Toc39005561)

[5.3.3 Entity Definition 14](#_Toc39005562)

[5.3.4 Webhook Integration & Fulfilment 15](#_Toc39005563)

[6 Project Performance & Validation 21](#_Toc39005564)

[7 Challenge & Recommendation 21](#_Toc39005565)

[7.1 Challenges 21](#_Toc39005566)

[7.1.1 Data Acquisition via Web Crawling 21](#_Toc39005567)

[7.1.2 Server Challenge 22](#_Toc39005568)

[7.1.3 Diagnosis Accuracy 22](#_Toc39005569)

[7.1.4 Reaching out to the Public 23](#_Toc39005570)

[7.2 Future Improvements 23](#_Toc39005571)

[7.2.1 Better Data Acquisition 23](#_Toc39005572)

[7.2.1 Better and More Reliable Servers 23](#_Toc39005573)

# 1 Executive Summary

The novel coronavirus pandemic, also known as COVID-19, has undoubtedly and most unfortunately grown to be one of the worst global pandemics in recent history. As of April 7, 2020, there were more than 1.4 million confirmed cases of COVID-19 around the world and more than 86,000 deaths, touching every continent except Antarctica. Singapore has also been hit relatively hard by the virus. The infographic shown in Figure 1 gives more details.

  
Fig. 1: Singapore’s COVID-19 cases (as of Apr 7, 2020)

Indeed, the threat of COVID-19 is very real and is only growing. The world has to comprehend this threat and it is by understanding the situation better that we may hopefully win the battle against the pandemic sooner rather than later. However, the need to understand how to tackle the virus is not necessarily exclusive to healthcare experts or servicemen. Everyone has to play their part in taking the right actions in order to help curb the spread of the virus. It may be as simple as sanitizing your hands frequently, to keeping up to date with the latest news on a country’s preventive measures. If any one individual fails to understand and carry out the right actions, the risk of spreading the virus is compounded and if one individual becomes many, said risk will no doubt grow exponentially.

Our project team is dedicated in equipping the general public with the relevant knowledge they should know in order to best protect themselves and those around them against the spread of the virus. As such, we have come up with a solution to help disseminate information on COVID-19 as simply and conveniently as possible. In fact, we have taken the term ‘knowledge’ quite literally by introducing artificial intelligence aided systems in our solution.

First and foremost, our solution will take advantage of the cognitive system capabilities found in a chatbot. We will be making use of the natural language processing provided by Google’s Dialogflow service and deploying it on Telegram, one of the most popular and widely used messaging platforms. These will help us to reach out to a vast majority of the general public. Secondly, we apply machine reasoning to acquire and prepare our knowledge base of information related to COVID-19. Further to that, we have made use of machine inference in the form of rules so that users can ask questions and be provided with the relevant answers and knowledge that they need. Last but certainly not least, our solution is also a reasoning system as we understand that each user’s needs are unique. As such, we are able to search for answers that best suit the user’s specific needs, such as providing the user with an optimal answer relative to his/her location.

Our project team hopes that with our solution, the general public will be able to receive updated knowledge through a method that is easy to use, accessible, understandable and personalized. It is only through the efforts of every single individual in doing their part in fighting COVID-19 that we may stop this pandemic. If our solution could play a small role in doing that, then it would already be a humble success in our books.

# 2 Business Justification

**“An investment in knowledge pays the best interest.”**

***Benjamin Franklin***

At the start of our team’s journey, there were not many notable applications or chatbots that were specifically catered to providing useful information and guidance on COVID-19. At that point in time, circa late January to early February 2020, the threat of the virus had not reached its global crisis level yet. However, it was nonetheless recognized to be growing and action had to be taken swiftly. The Singapore government was a few weeks away from rolling out its now popular applications in helping the citizens to combat the virus and there were also no relevant chatbots on Telegram platform that were related to it as well. Thus, we felt that there was a gap to be filled in the market and we set out in developing a chatbot that would fill this need for COVID-19 related information to be disseminated and accessed in a quick, convenient and simple method.

To quote Benjamin Franklin, “An investment in knowledge pays the best interest”. Our team acknowledged that this was reason enough to set forth on our solution and we sought to provide knowledge to the general public as a service. To help everyone around to be able to stay safe and stay healthy were adequate returns as well.

# 3 Project Team

## 3.1 Project Objective

??????????????????????????????

## 3.2 Team Members

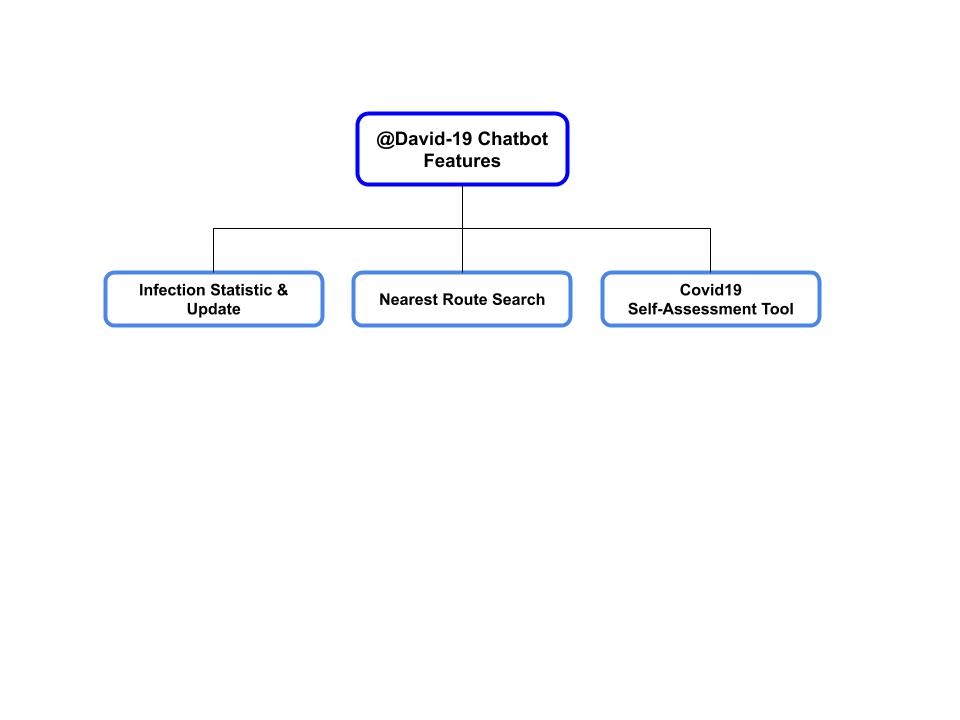
|  |  |
| --- | --- |
| Full Name | Work Items (Who Did What) |
| Lim Wee Kiat | * Project idea generation * Dialogflow – intents implementation & testing, entity creation, contexts & parameter design for feedback * Django webhook – Telegram integration, Django database design & setup, infection statistic, infection trend plots, A\* search for route & google API integration, feedback card, subscription & announcement * Heroku host & Cronjob Scheduler * Debug & troubleshooting * Graphic design & project report writing |
| Eu Jin Marcus Yatim | * Project idea generation |
| Teoh Yee Seng | * Project idea generation |

# 4 Project Solution

## 4.1 Project Deliverables

### 4.1.1 Chatbot Features

Our main goal of this group project is to integrate an intelligent chatbot system, which is capable of informing users about the current global situation of the COVID-19 pandemic, recommends appropriate solutions of COVID-19 related problems to the chatbot users, and help users to search for nearby hospitals. Below is the diagram which summarizes the deliverables of the chatbot.



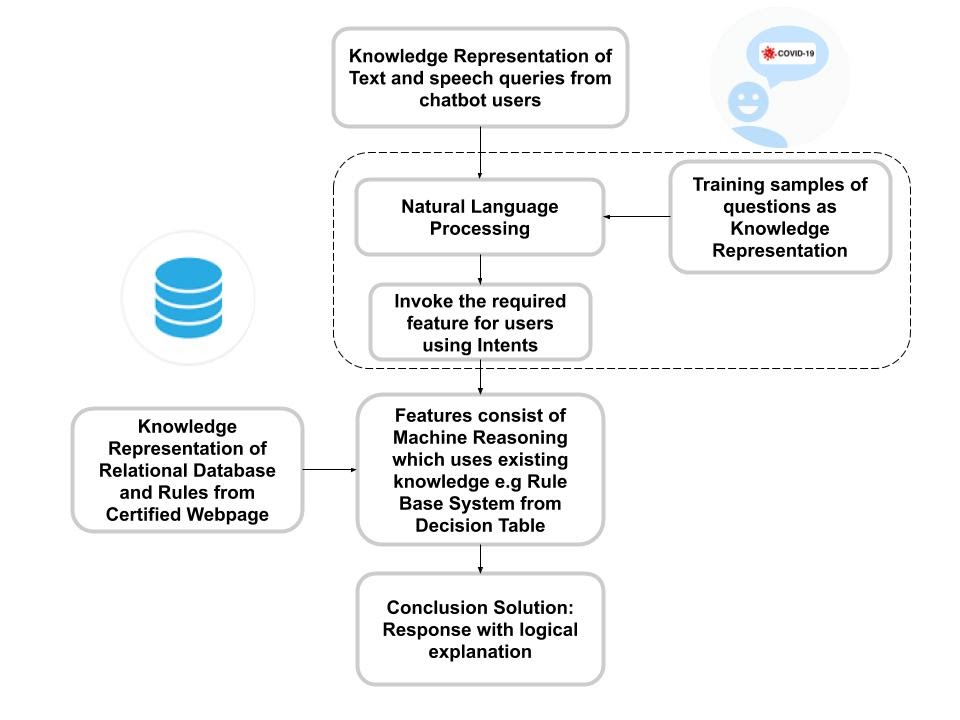
**Infection Statistic & Update:** Provide overall latest updates on infection status, infection trend and death rate of selected major countries. Plot and illustrate the data for user reference.

**Nearest Route Search**: Locate the nearest hospital or polyclinic from any specified location defined by the users.

**COVID-19 Self-Assessment Tool**: Ask users a series of questions such as travel history, recently contacted person, and symptom of users. Collect responses from users, and provide useful recommendations to users based on a set of rules.

### 4.1.2 Implementation Concepts

Before we go into the technical implementation of the project, designing a flowchart is a good way for us to understand the overall high-level structure of the intelligent chatbot system, as shown in the diagram below.



Our chatbot is basically a goal driven system, as the chatbot will focus on delivering accurate and reliable answers from users’ queries. When the users start to ask questions, text or speech will be transformed into useful data by NLP, which will be used by our backend server (webhook) to choose the right feature and process the output needed by users. Certain features such as acquiring the latest status and news of COVID-19, will need to extract information from official web pages. This can be done by applying a web scraping function to capture the important data into our database system, and the reasoning system will fully utilize the database to generate a required solution.

### 4.1.3 Outcome

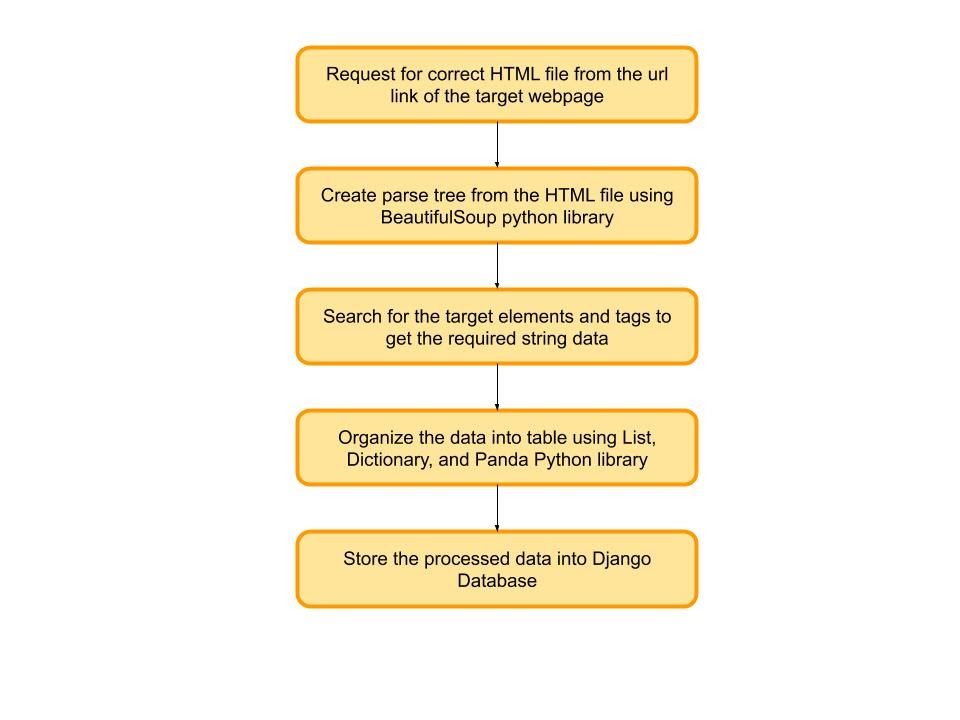
As for the software implementation, we have applied various open source softwares such as Dialogflow, Google API, Python Pip packages and so on. As depicted in the previous diagram, those components inside the dashed container will be executed by Dialogflow API. The reason behind the choice of using Dialogflow is that it is one of the best NLP platforms that can be installed in various chatbot applications with various languages and on multiple platforms. Various data extraction libraries and the machine reasoning method using the decision table will be explained in the next section below.

## 4.2 Knowledge Representation

### 4.2.1 Data Acquisition and Synchronization

#### Extraction Method

Below shows the flow chart on how the extraction of data is done in our chatbot system.



In order to extract resources and data from online web pages, our system will need to have a web scraping function to do the job. One of the best ways to create such a function is to make use of an open source python library such as Beautiful Soup. It consists of useful APIs which are able to pull data out from web page files such as HTML and XML files, by creating a parse tree of the file to allow the system to navigate, search, and modify the target string data. The BeautifulSoup library consists of “find” and “find\_all” search methods, which make use of elements and tags in the HTML file for the searching.

After extracting string data from an HTML file, the system will organize the data into tables using various data containers such as list, dictionary from standard Python Library, and DataFrame from Panda Python Library. The table will then be stored into Django Database as a model object instance, which can be accessed by other features of the chatbot system server.

#### Synchronization Method

Below depicted the list of active web pages where the data are to be extracted and synchronized by the chatbot.

|  |  |
| --- | --- |
| Website | Data to Crawl |
| *https://www.worldometers.info/coronavirus/* | Infection Status across all countries |
| *https://www.moh.gov.sg/covid-19* | Latest updates & advisories from Ministry of Health, Singapore |
| *https://github.com/CSSEGISandData/COVID-19* | Infection trend data from data repository maintained by John Hopkins University Center |

To synchronize the database of the chatbot server with the frequently updated online webpage, the data extraction of the system is required to be executed in certain periods of time, which is about 15 mins. This can be done with the help of Cronjob external time scheduler, keeping the information updated automatically. This allows users to get the most accurate answers at any point of time while using the chatbot.

### 4.2.2 Decision table in Self-Assessment Tool

Since one of our chatbot features is to allow users to self-assess for the COVID-19 symptoms, we have designed a decision table as a form of knowledge representation of the chatbot reasoning system. The decision table is formed by a series of questions, which will in turn output appropriate answers categorized from low to high risk. There are a total five follow up questions to be requested by the chatbot, such as previous travel history of the user, previous contact with the infected person, and current health status of the users. Users will need to either agree or disagree with the questions, using their own choices of word (agree with “yes”, “yup”, “of course”, disagree with “no”, “nope”, “not at all”).

Different combinations of answers from users will trigger different responses from the chatbot. The provided responses can be categorized from high risk, medium risk, low risk, and no risk. If the users agree that they have either traveled outside Singapore or contacted confirmed infected people, without showing any virus symptom, the chatbot will warn the users to follow the quarantine order or just stay at home and check for their own symptoms. Otherwise, if the users agree that they have exhibited virus symptoms such as cough, fever, or shortness of breath, the level of risk recommendation by chatbot will be based on the number of symptoms the users have. Showing three symptoms will lead to high risk recommendation by chatbot, two symptoms lead to medium risk recommendation, one symptom leads to low risk recommendation, and zero symptom leads to no risk recommendation.

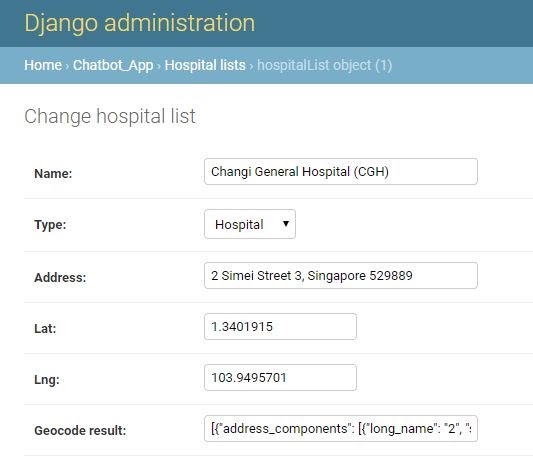
Should the users receive a high-risk recommendation from chatbot, the chatbot will proceed to ask the user if they want to find out the nearby hospital location. It will trigger the new conversation about the location of the hospital should the users agree. Below shows the decision table where the chatbot will output the recommendation based on the response from users.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Rule No.** | **Triggering combinations** | | | | | **Logical Condition** | **Descriptions** | **Recommendation** |
| **Other Questions (Q)** | | **Symptoms (S)** | | |
| **Travel Outside Singapore (Q1)** | **Contact with infected people(Q2)** | **Cough(S1)** | **Fever(S2)** | **Breathing(S3)** |
| R1 | 1 | 1 | 1 | 0 | 0 | ((Q1 v Q2) ^ (S1 v S2 v S3)) v ((Sum(Q)=0) ^ (Sum(S)=3)) | If sum of Q and sum of S are not zero, or if sum of Q is zero and sum of S is 3. | HR1 |
| 0 | 0 | 1 | 1 | 1 |
| R2 | 1 | 0 | 0 | 0 | 0 | Q1 ^ (Sum(S)=0) | Prioritize Q1 | HR2 |
| R3 | 0 | 1 | 0 | 0 | 0 | (!Q1 ^ Q2) ^ (Sum(S)=0) | Only consider not Q1 and Q2 conditions. | HR3 |
| R4 | 0 | 0 | 1 | 1 | 0 | (Sum(Q)=0) ^ (Sum(S)=2) | When the sum of Q is zero, check for the sum of S. | MR |
| R5 | 0 | 0 | 0 | 1 | 0 | (Sum(Q)=0) ^ (Sum(S)=1) | LR |
| R6 | 0 | 0 | 0 | 0 | 0 | (Sum(Q)=0) ^ (Sum(S)=0) | NR |
|  |  |  |  |  |  |  |  |  |
| Note: Q consists of Q1 and Q2, S consist of S1, S2 and S3. | | | | |  |  |  |  |

**Citation: “Singapore COVID-19 Symptom Checker.”, sgcovidcheck.gov.sg/.**

### 4.2.3 Hospital Location Database Structure

Other than using a decision table, the chatbot system also makes use of relational databases as a knowledge representation of the machine reasoning system of the chatbot. As stated in section 4.2.1, information from web scraping will be collected as a relational database and stored into the Django database. Other than that, the chatbot system is also able to directly upload data files into the Django database. A list of hospitals and polyclinics with their properties such as name, address, longitude, and latitude, can be converted into model object instances, where the nearest route search feature will apply the object instance for its own calculation and output results for users. The detailed explanation about the feature in chatbot will be covered in section 5 of this report. Below is an example of the model object instance created in Django Database.



# 5 Project Architecture & Implementation

## 5.1 Architecture Overview

**Dialogflow** has been selected to be the cognitive solution framework for this project due to its simple and intuitive interface, ease of deployment and webhook integration, completeness of documentation and abundant implementation examples on the internet.

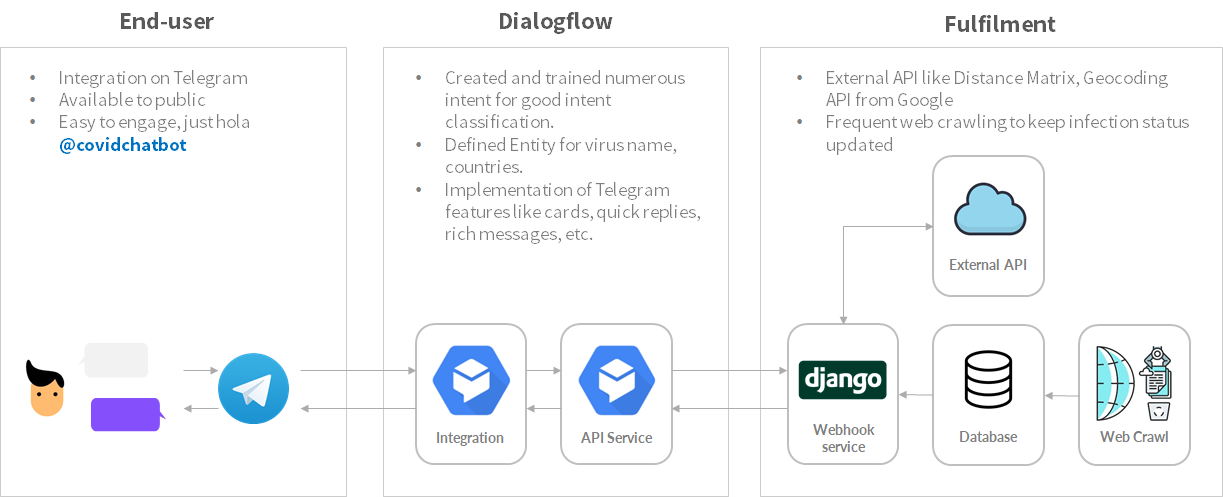
There are multiple deployable platforms for the chatbot, ranging from creating your own front-end, facebook messenger, slack, skype, Telegram and many more. The team has finally decided to use **Telegram** as the main chatbot deployment platform as it has access to the public masses, decent integration to Dialogflow features, and easy to implement. Gaining access to more public users is one of the key metrics of the team’s decision as the team hopes that this chatbot solution can help the public to gather latest, essential and accurate information. By gaining more users, the team can also get more training data from different variations of questions asked and get insights of what other essential information the people are looking for. These data will then be used to improve the accuracy of intent matching and improve the functionality by adding new intents from the gained insights.

To keep the chatbot running almost 24/7, the **webhook service** that is built on Django framework and responsible for all of the main functionalities are hosted on **Heroku**, alongside with a scheduled POST request to trigger web crawling scripts every 15 mins to keep the chatbot’s database updated. Our Django webhook service is also integrated with other external API like Google geolocation API to help users find the nearest route to hospital or polyclinic.

## 5.2 Process Flow

Illustration at the end of this section shows how the process flow from Telegram platform to Dialogflow, then to Django webhook service and back to the user.

1. User engages chatbot on Telegram platform. Any message sent to the bot will be forwarded to Dialogflow as API
2. Dialogflow classifies the message into user’s intent
3. Depending on the intent, if there is no fulfilment needed for that intent, Dialogflow will return the response as API back to Telegram. Telegram then presents the message in chat form back to the user.
4. If fulfilment is needed for the intent, Dialogflow will pass the intent, parameters detected, user’s messages to Django webhook service. Webhook service will call external API, infer response from rules, look-up statistical data from knowledge base etc, if required.
5. The response is then passed back to Dialogflow as API, then subsequently to Telegram as chat form to the user.



## 5.3 Cognitive System

To enable a proper functioning cognitive system, the following components discussed below need to be established.

### 5.3.1 Create an Agent

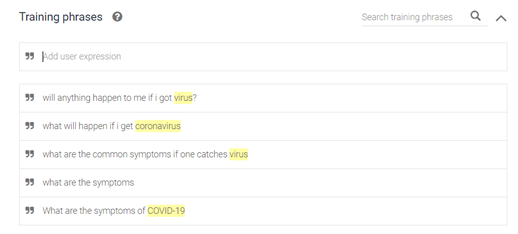
Firstly, a Dialogflow agent must be created on Dialogflow Cloud console. A Dialogflow agent is a virtual agent that can translate human language into actions through a Natural Language Understanding (NLU) module.

As stated in Section 4.1 Project Deliverables, the agent will try to understand what the user is enquiring about, based on the context of user’s question, agent will perform a classification and trigger an intent that the user is most likely looking for based on its confidence level, and return the respective response for that intent.

### 5.3.2 Create Intents

In order for the agent to have intents to match with the context of the user's question, we need to create intents for the agent. For each agent, many intents can be created where they can be sometimes combined to handle a complete conversation.

To create an intent, we will need training phrases. In this example, we want to create an intent to inform users about symptoms of COVID-19. Training phrases are example questions or phrases for what the users might ask, type or say.

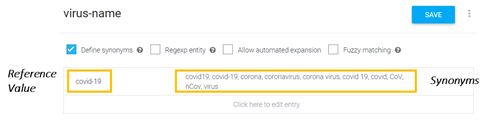


After the model is trained and when the users ask a question that resembles the training phrases, the agent will trigger this intent as it should get the highest confidence score in its intent classification.

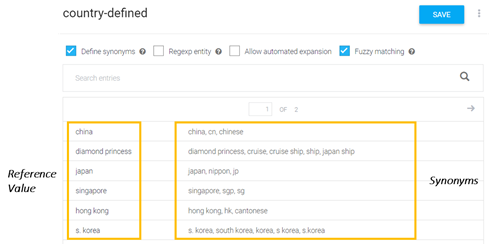


### 5.3.3 Entity Definition

Using the same example above, when user queries about “symptoms for coronavirus”, the Dialogflow agent will be able to associate “coronavirus” to COVID-19 virus, although they appear to be different. This is made possible by defining Entity and synonyms in Dialogflow console, which allow us to define multiple words or phrases that are considered to be of similar or equivalent. For example, “virus-name” entity was created with reference value “covid-19”, which has the synonyms of “corona”, “coronavirus”, “nCov”, “virus”, “covid 19”, etc. With that, any variations of COVID-19 listed in the synonym list will always be associated with “covid-19” as Entity parameter.



Entity definition is extremely important when a user queries about a situation report for a specific country in our chatbot application. When a country is detected in a sentence or question, Dialogflow will associate the country to its reference value, and return the reference value as parameter to our Django back-end. This parameter will then be used as a keyword to perform database query or look-up function from the database, hence it is extremely important that the keyword of user defined country must match with the keyword of country in database else an error will be returned. To avoid such look-up error, the team has diligently created over 60 countries’ reference value in defining Entity for country. Example shown below:



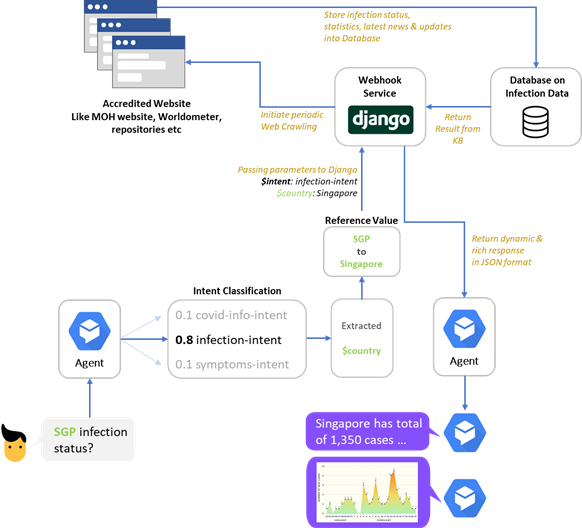
### 5.3.4 Webhook Integration & Fulfilment

With the help of Webhook integration, we can make our Dialogflow agent smarter by responding with dynamic and rich content back to the users. The team has created a Webhook service that is built on Django, where we gather and store extensive heterogeneous information and knowledge from various websites via web crawling, integrate to external APIs to complete certain fulfilment for the user, and build our rule-based inference machine.

#### Infection Statistic & Update

As one of the project’s deliverables, we want to provide updates on infection status, infection trend and death rate, for any country.

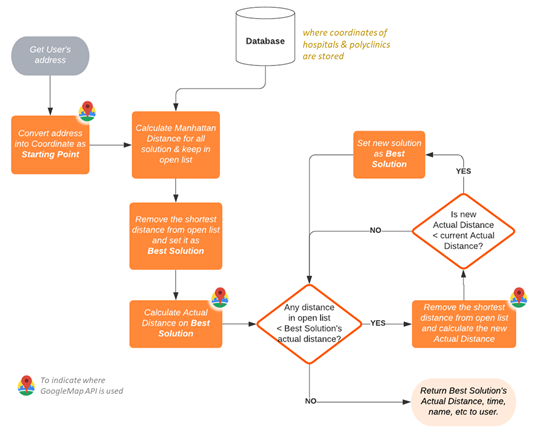
Other than web crawling script as described in Synchronization Method in Section 4.2.1, another python script to generate infection & death trends for certain countries are also being plotted periodically as well and published onto the internet to generate image urls. Those image urls are then being used to send as photos to the users on Telegram platform. An illustration to depict how users’ text are being parsed by Dialogflow and how the information is passed from Webhook service back to users, is shown below.



#### Nearest Route Search

The other main feature of our chat bot is to locate the nearest hospital or polyclinic from any specified location defined by the users. Search technique used is A\* search, with heuristic function defined as straight line distance between coordinate of all hospitals or polyclinics and coordinate of user’s defined location.

Name and address of all major hospitals and polyclinics are being manually extracted from the internet, converted into coordinates by using Geocoding API by Google manually, and stored in chatbot’s Database. When a user triggers the “find nearest hospital” intent, Dialogflow agent will get the user's address as parameter and pass it to our Webhook service. With the integration of Google API on our webhook service, the user’s address is then converted into coordinates as an initial point. We then calculate the Manhattan distance from the initial point to all destination points and select the first solution with the shortest distance. First solution is then removed from an open list while keeping the rest of the other possible solution in. The actual route distance is calculated for the first solution by using Distance Matrix API by Google. Search will continue to explore another potential solution in the open list if any of the solutions has lesser straight line (Manhattan) distance than actual route distance. Search will end and select the current solution as the best solution, if there is no shorter distance in the open list than the current one. The process of the search is illustrated in flow chart below:



#### Smart COVID-19 Self-Assessment Tool

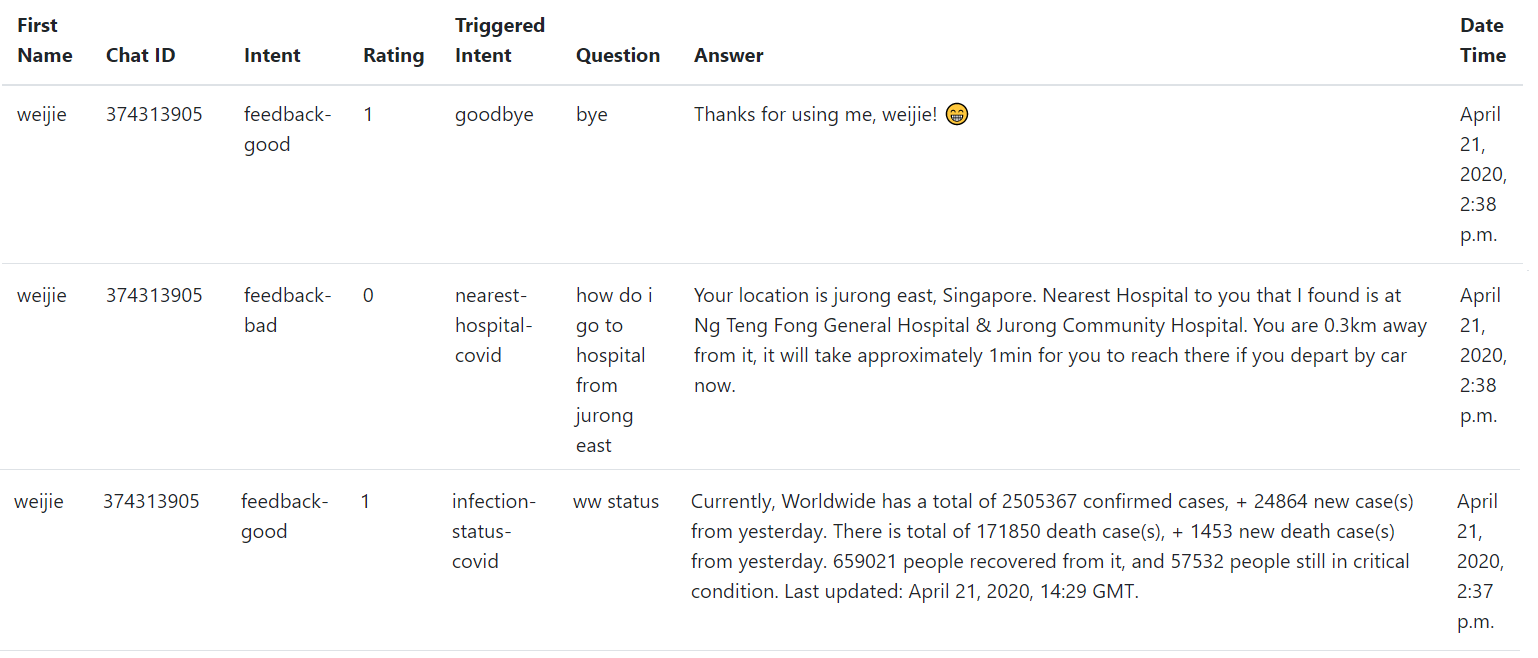
|  |  |
| --- | --- |
| Based on rules set derived from Section 4.2.2, whenever a user triggers the “Self-Assessment” intent, chatbot will ask a series of questions to the user. Dialogflow will pass the user’s responses to webhook services, where it gets inferred by our rules to classify the risk level of the user, and propose recommendations based on risk level.  Should the users receive a high-risk recommendation from chatbot, the chatbot will proceed to ask the user if they want to find out the nearby hospital location. It will trigger the new conversation about the location of the hospital should the users agree. | https://lh4.googleusercontent.com/2px6Ely5RTGSs3Gd7n7tpId9-7cjZL11GA3AJmTDXIcKco47wwc1mENLuEsiQnygfAVfhD_sTakxUieGAo0EiUtKmN4gc8eZ05YQOEnAKkaH3BQzqPkKoiEKauj54-h_b6GFGY4S |

#### User’s Feedback Feature

|  |  |
| --- | --- |
| https://lh6.googleusercontent.com/3mHirXNLnvrJFsg8tctVZn1SCi32nnMuo9Bb5KobaWONKtKLZh0LpUZf6KW9EwCFWewa50EBDDNilTBzM11gD4e--1fs7Eg8ZBktHKczx8SB-BUA61M0AfDnLOOZ1Sx8nMMmwkQe | Chatbot is to serve users or customers. A major metric for a chatbot is that users is able to utilize our chatbot and walk away with the information they seek for contentedly. We want to solicit feedback from the users in a very simple and frictionless way so that we can measure how good our chatbot is performing out in the field. These feedback data will be collected in the webhook service’s database and will be used to gauge accuracy of our data and relevance of the chatbot’s responses to user’s queries, so that the team can investigate what has gone wrong, or anything that the team can do to further improve and develop the chatbot in order for its services to be better. |

The team has created a feedback intent in Dialogflow with two follow-ups intent for good and bad rating. Aside from this, the team also made use of the webhook service with context to automatically ask users for feedback after completing certain intents. For example, a feedback card was appended to our “latest-news” intent, so users can rate “thumbs-up” or “thumbs-down” depending on the accuracy of information or relevance of response to their original intent.

Feedback card is only introduced to several major intents and customizable to be appended to any intent handled by webhook service by just setting *self.get\_fb = 1* at the end of intent’s code. By responding to the feedback, user’s rating (1 for good rating, 0 for bad rating, -1 for unrecognized rating), queries and chatbot’s response will also be passed back into the webhook service via Context, so that we can associate the rating to the conversation. Below is an example of feedback information collected:



#### Additional Features

During this pandemic period, new information is being released constantly and the team acknowledges that it is exhausting to keep on following the latest news, advisories, rules and regulations imposed by the local government to keep COVID-19 virus at bay. Hence, the team has created a **subscription and announcement** feature on chatbot, which it is able to broadcast announcements to every user that has subscribed. Whenever a user triggers an intent that involves webhook service, their name and chat ID will be stored in the database and they are auto-subscribed to the announcement list, unless they opt-out from the list. Announcement can be made by using the “announcement.py” file in the source code folder, where the administrator will have to enter their personalized broadcast message before run. For more details, refer to user guide to make personalized announcement.

Besides, the team has also made the chatbot to **address users by their name**, which they registered on their Telegram. This is to give the users a feeling of personal touch to enhance their user experience, as though the chatbot is their personalized COVID-19 expert.

# 6 Project Performance & Validation

|  |  |
| --- | --- |
| Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.  Sed ut perspiciatis unde omnis iste natus error sit voluptatem accusantium doloremque architecto beatae vitae dicta sunt explicabo. Nemo enim ipsam voluptatem quia voluptas sit aspernatur aut odit aut fugit, sed quia consequuntur magni dolores eos qui ratione voluptatem sequi nesciunt. |  |

# 7 Challenge & Recommendation

Our team was successful in deploying the Telegram chat bot app and it is now live **@covidchatbot**. Of course, this came with its own set of challenges and there are also many ways we can improve the implementation of the chat bot with the luxury of more time and resources. The sections below describe the challenges we faced, how we tackled them and lastly further improvements for future releases.

## 7.1 Challenges

### 7.1.1 Data Acquisition via Web Crawling

As described in **4.2.1 Extraction Method** and further detailed in **5.2.4.1 Infection Statistic & Update**, information and data are extracted from three different trusted websites via a web crawling script. These websites are constantly updated and change dynamically. This is a huge obstacle as even the slightest change in the website’s CSS script can throw off the web crawling script completely. Needless to say, data cannot be extracted in real time if the latter occurs and this will cause our chat bot to not perform as desired. As such, regular downtime tracking & maintenance have to be performed on the web crawling script to fit the website’s latest CSS script. However, there are also no website updates or maintenance notices that pre-empt a change and this makes it difficult for the team to perform timely maintenance of the web crawling script. All these means that there is a high maintenance cost of the chatbot while also still being at risk of it not performing optimally.

Currently, we have included exception handling in our code to throw an exception as soon as an error occurs. However, as mentioned above, we are not able to pre-empt the error and the chatbot will be vulnerable to outdated data nonetheless.

### 7.1.2 Server Challenge

As described in **5.1 Architecture Overview**, the team chose to go with Heroku. Heroku is likewise a PaaS that offers scalable server usage. The user friendliness, ease of deployment and fuss free package plans made Heroku a suitable choice for our needs. However, our team has opted to use Heroku’s free tier package and this no doubt still creates limitation of server sleeping within 30 mins of inactivity.

To work out the best countermeasure, we included a POST request to trigger the web crawling scripts every 15 mins. This was done to create a balance between keeping the servers active, keeping within our limited hours and last but not least, providing as up to date to real time information as possible.

### 7.1.3 Diagnosis Accuracy

As described in section **4.3.1 Diagnosis Decision table**, our team has formulated a diagnosis that follows a set of rules. This diagnosis and its rules are written based on extensive research on the novel coronavirus from the Singapore government’s own service at <https://sgcovidcheck.gov.sg/>. However, our diagnosis cannot be treated as definite answers but more of a rough guideline. We have included a caveat that proper healthcare should always be sought out if anything is amiss. Besides, new symptoms and diagnosis information are being discovered from time to time and it certainly is a challenge for the team to always keep our self-assessment tool up-to-date.

### 7.1.4 Reaching out to the Public

We believe that it takes everyone to play a role in protecting our greater well-being during a pandemic and this chatbot service is for the benefit of the public to be prepared to do just that. However, for the chatbot to serve its purpose, the first challenge would be for it to reach out to the public. This would mean for the chat bot to be promoted and made known of its existence. At this current stage in time, our team does not have the resources to market or promote the chatbot in order for it to gain traction among a wider audience. However, we believe that word of mouth and the prevalence of social media would help us to boost the awareness of the chatbot.

## 7.2 Future Improvements

### 7.2.1 Better Data Acquisition

As described in **7.1.1 Data Acquisition via Web Crawling**, web crawling of websites leaves one prone to being at the mercy of the website changing and the web crawler script failing. Not to mention having to do regular maintenance of the code as well. A more sustainable, efficient and effective approach would be to depend on a call to an API to provide information in a standard JSON format or even a csv file format. At the moment, there are not many of such APIs available, however, there is a promising one on GitHub that is regularly updated and curated (see Note for more information). Future release would see the code updated to remove the web crawling function and replace its functionality with an API call instead.

### 7.2.1 Better and More Reliable Servers

As described in **7.1.2 Server Challenge**, the chatbot is limited in more ways than one by the speed and power of its server. At the moment, our team has limited funds to acquire a better and more reliable server. It is with hope that in the near future, with more resources, we would be able to improve this aspect.

# APPENDIX OF REPORT A

Project Proposal

**GRADUATE CERTIFICATE: Intelligent Reasoning Systems (IRS)**

**PRACTICE MODULE: Project Proposal**

|  |
| --- |
| **Date of proposal:**  10 May 2020 |
| **Project Title:**  ISS Project – COVID-19 Chatbot Project |
| **Sponsor/Client:** *(Name, Address, Telephone No. and Contact Name)*  Institute of Systems Science (ISS) at 25 Heng Mui Keng Terrace, Singapore  NATIONAL UNIVERSITY OF SINGAPORE (NUS)  Contact: Mr. GU ZHAN / Lecturer & Consultant  Telephone No.: 65-6516 8021  Email: [zhan.gu@nus.edu.sg](mailto:zhan.gu@nus.edu.sg) |
| **Background/Aims/Objectives:**  The proposed intelligent eco-system will make use of various advanced machine reasoning techniques and components to foster generic intelligent system adoption and agile implementation for business. |
| **Requirements Overview:**   * Research ability * Programming ability * System integration ability |
| **Resource Requirements (please list Hardware, Software and any other resources)**  Hardware proposed for consideration:   * GPU, RaspberryPi, AlphaBot, NVidia Jetson Box, etc.   Software proposed for consideration:   * Reasoning systems, e.g. KIE jBPM, Drools, AppFormer, OptaPlanner, Fuzzy logic, Optimization, etc * Pertained machine learning models, e.g. Vision, Speech, NLP * Machine learning use cases, e.g. Orange3, R * Deep learning tools, e.g. Neural Network Console Sony, Python Keras * Chat-bots, e.g. ChatterBot, DBpedia Chat-bot * Cognitive systems, e.g. MyCroft * Robotic Process Automation, .e.g TagUI * Cloud computing/server, e.g. Amazon, Google, IBM, Azure, etc. * Application container, e.g. Docker |
| **Number of Learner Interns required: (Please specify their tasks if possible)**  a team of four to six project members (or individual work upon lecturer approval) |
| **Methods and Standards:**   |  |  |  | | --- | --- | --- | | **Procedures** | **Objective** | **Key Activities** | | | **Requirement Gathering and Analysis** | The team should meet with ISS to scope the details of project and ensure the achievement of business objectives. | 1.        Gather & Analyze Requirements | | 2.        Define internal and External Design | | 3.        Prioritize & Consolidate Requirements | | 4.        Establish Functional Baseline | | **Technical Construction** | ·         To develop the source code in accordance to the design. | 1.        Setup Development Environment | | ·         To perform unit testing to ensure the quality before the components are integrated as a whole project | 2.        Understand the System Context, Design | | 3.        Perform Coding | | 4.        Conduct Unit Testing | | **Integration Testing and acceptance testing** | To ensure interface compatibility and confirm that the integrated system hardware and system software meets requirements and is ready for acceptance testing. | 1.        Prepare System Test Specifications | | 2.        Prepare for Test Execution | | 3.        Conduct System Integration Testing | | 4.        Evaluate Testing | | 5.        Establish Product Baseline | |  | | **Acceptance Testing** | To obtain ISS user acceptance that the system meets the requirements. | 1.        Plan for Acceptance Testing | | 2.        Conduct Training for Acceptance Testing | | 3.        Prepare for Acceptance Test Execution | | 4.        ISS Evaluate Testing | | 5.        Obtain Customer Acceptance Sign-off | |  | | **Delivery** | To deploy the system into production (ISS standalone server) environment. | 1.        Software must be packed by following ISS’s standard | | 2.        Deployment guideline must be provided in ISS production (ISS standalone server) format | | 3.        Production (ISS standalone server) support and troubleshooting process must be defined. | |  | |

**Team Formation & Registration**

|  |
| --- |
| Team Name: |
| Project Title (repeated): |
| System Name (if decided): |
|  |
| Team Member 1 Name:  Lim Wee Kiat |
| Team Member 1 Matriculation Number:  A0213481M |
| Team Member 1 Contact (Mobile/Email):  +65-94575417 / e0508582@u.nus.edu |
|  |
| Team Member 2 Name: |
| Team Member 2 Matriculation Number: |
| Team Member 2 Contact (Mobile/Email): |
|  |
| Team Member 3 Name: |
| Team Member 3 Matriculation Number: |
| Team Member 3 Contact (Mobile/Email): |

|  |  |  |
| --- | --- | --- |
| **For ISS Use Only** | | |
| **Programme Name:** | **Project No:** | **Learner Batch:** |
| **Accepted/Rejected/KIV:** | | |
| **Learners Assigned:** | | |
| **Advisor Assigned:**  Contact: Mr. GU ZHAN / Lecturer & Consultant  Telephone No.: 65-6516 8021  Email: [zhan.gu@nus.edu.sg](mailto:zhan.gu@nus.edu.sg) | | |

# APPENDIX OF REPORT B

Mapped System Functionalities against knowledge, techniques and skills of modular courses

# APPENDIX OF REPORT C

Installation and User Guide

# APPENDIX OF REPORT D

List of Intents

# APPENDIX OF REPORT E

Individual Reports

# APPENDIX OF REPORT F

Abbreviations & References