



NATIONAL UNIVERSITY OF SINGAPORE

RPA & IPA PROJECT

COVID-19 Assistant

Group Members

DAI YIRUI

GUO FENG

LI DUO

LU JIAHAO

Student ID

A0195167U

A0195173Y

A0195364W

A0091835Y

Supervised by

GU Zhan (Sam)

NUS-ISS

Sunday 3rd May, 2020

Contents

1	Business Problem Background	5
2	Objectives & Success Measurements	6
2.1	Objectives	6
2.1.1	COVID-19 Trends Queries	6
2.1.2	News and Subscription	6
2.1.3	NUS Temperature Submission	7
2.2	Success Measurements	7
3	Solution	8
3.1	Assumptions	8
3.1.1	Target Audience	8
3.1.2	Websites	8
3.1.3	Data Source and Quality	8
3.1.4	Server Environment	8
3.2	Project Scope	9
3.3	Knowledge Model	9
3.3.1	Knowledge Identification	9
3.3.2	Knowledge Acquisition	10
3.3.3	Knowledge Refinement	11
3.4	System Architecture	11
4	Implementation	13
4.1	Dialogflow	13
4.2	System Server	15
4.3	Database	16
4.4	Web RPA	17
4.5	Scheduler	19
4.6	Slack	20
5	Performance & Validation	21
6	Conclusions	24
6.1	Findings	24
6.2	Recommendations	25

7	References	26
	Appendices	27
	Appendix A User's Manual	27
A.1	Prerequisite	27
A.1.1	Accounts	27
A.1.2	Codes & Packages	27
A.1.2.1	Download Project Code Files	27
A.1.2.2	Setup Environment	27
A.2	Server Setup	28
A.2.1	Crawl Data from Website	28
A.2.2	Start Up Main Server & Scheduler	29
A.2.3	Start 'ngrok' Service	31
A.3	Dialogflow Setup	32
A.3.1	Import Agent	32
A.3.2	Create Fulfilment	35
A.3.3	Integration	35
A.4	Test Demo	36
	Appendix B Individual Report	38
B.1	GUO FENG	38
B.1.1	Personal Contribution	38
B.1.2	What I have learnt	38
B.1.3	How I can apply the knowledge and skills in other situations . . .	39
B.2	DAI YIRUI	39
B.2.1	Personal Contribution	39
B.2.2	What I have learnt	40
B.2.3	How I can apply the knowledge and skills in other situations . . .	40
B.3	LU JIAHAO	40
B.3.1	Personal Contribution	40
B.3.2	What I have learnt	41
B.3.3	How I can apply the knowledge and skills in other situations . . .	41
B.4	LI DUO	42
B.4.1	Personal Contribution	42
B.4.2	What I have learnt	42
B.4.3	How I can apply the knowledge and skills in other situations . . .	42
	Appendix C Project Related Files	43

Executive Summary

The spread of coronavirus disease 2019 (COVID-19) has already taken on pandemic proportions, affecting over 100 countries since December 2019 [1].

Many countries have restricted residents from public social interaction. The Singapore government has extended Circuit Breaker [2] to bring down community cases. Most people will stay at home to avoid potential risks. At the same time, they still need to get the updated information about COVID-19.

Although news and dashboards about COVID-19 are flooded through different web pages, to quickly get a personalized summary from a reliable source can still be a challenge.

Our team designed a system which can retrieve COVID-19 data and provide users based on their preferences. NUS students and staff could also provide the body temperature to our system and it will automatically update to NUS Daily Temperature Declaration website. This greatly reduces users' time on login and body temperature input everyday.

1 Business Problem Background

The outbreak of Coronavirus Disease 2019 (COVID-19) has spreaded over the world. It caused thousands of deaths and tens of thousands of infections. It also paralyzes cities and villages across countries and has a disruptive impact on business, travel, and education around the world.

As of 1st May 2020, total confirmed cases have exceeded 3.3 million [3] and the numbers continue to grow. In the current environment, for most ordinary people, the number of confirmed cases around themselves and their loved ones is important daily information. Many websites provide abundant information about COVID-19 but not all the people want this overwhelming information.

In the same time, NUS has asked each student to declare the body temperature twice daily, in the morning and afternoon. A photograph of each temperature reading should be taken with a date and time stamp. The image should be retained for seven days.

2 Objectives & Success Measurements

2.1 Objectives

The primary target audience of the system are people who care about COVID-19 information.

There are 5 major objectives in this project:

- Provide timely COVID-19 information based on users' queries
- Provide COVID-19 subscription based on users' preference
- Provide translated news subscription in multiple languages
- Help NUS users update body temperature to "DAILY TEMPERATURE DECLARATION"

2.1.1 COVID-19 Trends Queries

To quickly and timely update information that users care about, our team developed a chatbot that users can interact with. It can quickly provide specific COVID-19 information based on users' subscription.

COVID-19 information includes global and country specific data. It covers "total cases", "new cases", "total deaths", "new deaths", "total recovered", "active cases" and "serious cases".

2.1.2 News and Subscription

News about COVID-19 can be summarized and shared with users. A daily news update about COVID-19 could be sent to users. Users are also able to view news from chatbot. News can be provided in multiple languages based on users' needs.

2.1.3 NUS Temperature Submission

Staff and Students from NUS can submit body temperatures through our system. This reduces the cumbersomeness of opening the website every day.

2.2 Success Measurements

There are 4 key measures of the system:

1. Whether the system was able to understand the user
2. Whether the system was able to present the specific COVID-19 information being requested by the user
3. Whether the system was able to provide stable personalized subscription to the user
4. Whether the system was able to successfully submit body temperature on behalf of the NUS user

3 Solution

3.1 Assumptions

3.1.1 Target Audience

The major target audience would be the NUS students and staff who want to get quick updates on COVID-19 information every day. They may have family members in different countries and use different languages. They are also required by NUS to submit a daily temperature declaration every day.

Although NUS students and staff are the main target audience. Any people who want to check COVID-19 information or make a daily subscription can also use the system.

3.1.2 Websites

The team will use specific websites to capture the data, and no customization to choose other sources of information. While in short time, the crawling function is able to work properly. However, information sources (websites) may undergo maintenance and update. Under such circumstances, interaction between the system and those information sources may be affected. System admin may be required to update the system, such as the xpath, when change happens.

3.1.3 Data Source and Quality

All information used to build COVID-19 is from Worldometer [3] and World Health Organization (WHO) [4] websites. It is assumed the data provided by these websites are correct and updated.

3.1.4 Server Environment

The server is hosted locally and the user needs to store the information locally. The user should have all required libraries and data in place before starting the system. In

addition, the user should have an email account and Google Cloud account to run the standalone scheduler.

3.2 Project Scope

The project comprises User front Chatbot, Covid-19 Web RPA, News and Subscription. The User front Chatbot is the conversational user interface that gets request from user to

- Upload temperature to NUS Temperature Declaring System
- Update subscription preference
- Check COVID-19 Data by Input Country
- Get latest News regarding COVID-19

It is built using Google Dialogflow and has been trained with intents.

Web Robotic process automation (RPA) tools are used to get daily COVID-19 data automatically from chosen websites. Temperature automated upload is also done by Web RPA. The user credentials are stored locally, and the RPA tool will retrieve a local json file and login into NUS temperature declaration system and help update . TagUI and selenium are used to achieve this.

The scheduler is used to send system generated emails to users. It is an independent system from RPA and the main system.

3.3 Knowledge Model

Knowledge modeling can be classified into three parts [5]:

- (i) Knowledge identification
- (ii) Knowledge specification
- (iii) Knowledge refinement

3.3.1 Knowledge Identification

Knowledge identification sets the groundwork for the next stage encompassing knowledge specification. Information sources that are deemed to be useful are identified in prepa-

ration of knowledge acquisitions. In the context of building a chatbot, two main sources have been identified and are documented in Table 3.1.

S/N	Source of Information	Insights from information sources	Knowledge acquisition technique
1	Worldometer website	It provides COVID-19 cases data in different countries and trend charts	TagUI and selenium
2	WHO website	It provides COVID-19 global news everyday	TagUI and goose3
3	NUS website	It provides past temperature readings	selenium

Table 3.1: Knowledge Source and Acquisition Technique

3.3.2 Knowledge Acquisition

Following from the identification of knowledge sources, knowledge acquisition is conducted to capture the domain knowledge. The techniques adopted to acquire the knowledge have been describe in Table 3.1 and the corresponding results are presented using a dependency diagram as shown in Figure 3.1.

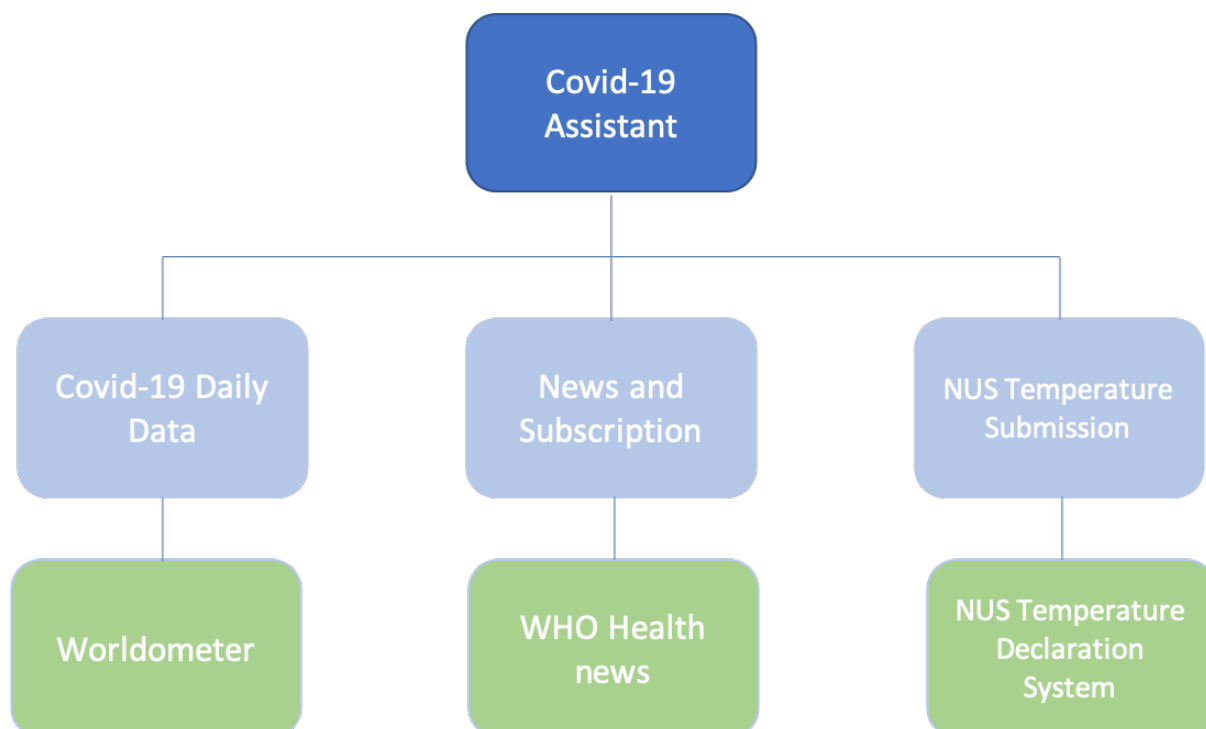


Figure 3.1: Domain Model

3.3.3 Knowledge Refinement

The knowledge refinement is an iterative process and it include model validation and model refinement. For model validation, different sets of test data will be used to run the simulation, and the result will be compared with sample data.

For model refinement, after getting different actual results from expected results, the team adjusted the training phrases to get the expected result.

3.4 System Architecture

Figure 3.2 illustrates the system architecture of how different components interact with each other. Users use Slack as an interface to communicate with the intelligent agent. Dialogflow receives the input and extracts the intents from the sentence, then feeds into the flask server.

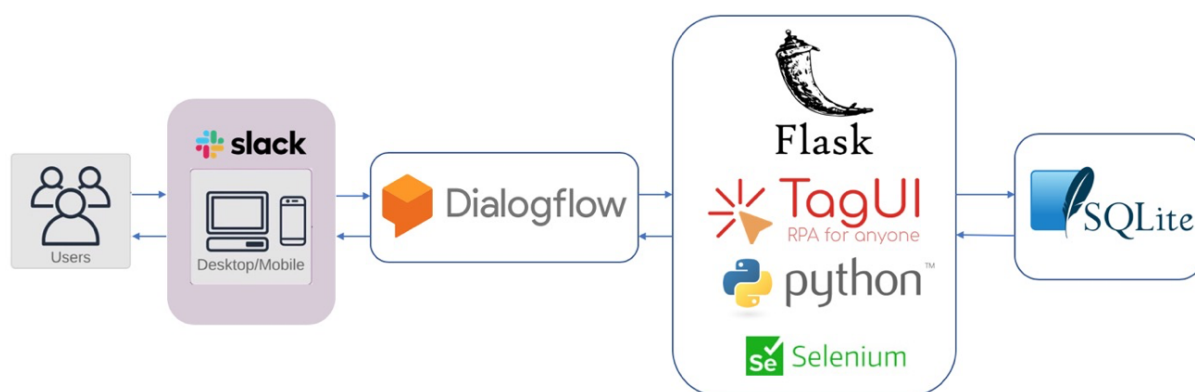


Figure 3.2: System Architecture

The Flask server will process the information with TagUI and selenium to capture the data automatically and store it into DB / return to the user.

Figure 3.3 shows more details regarding each component in the system and how information has been passed though inside the system. More description of each component will be covered under the System Implementation section.

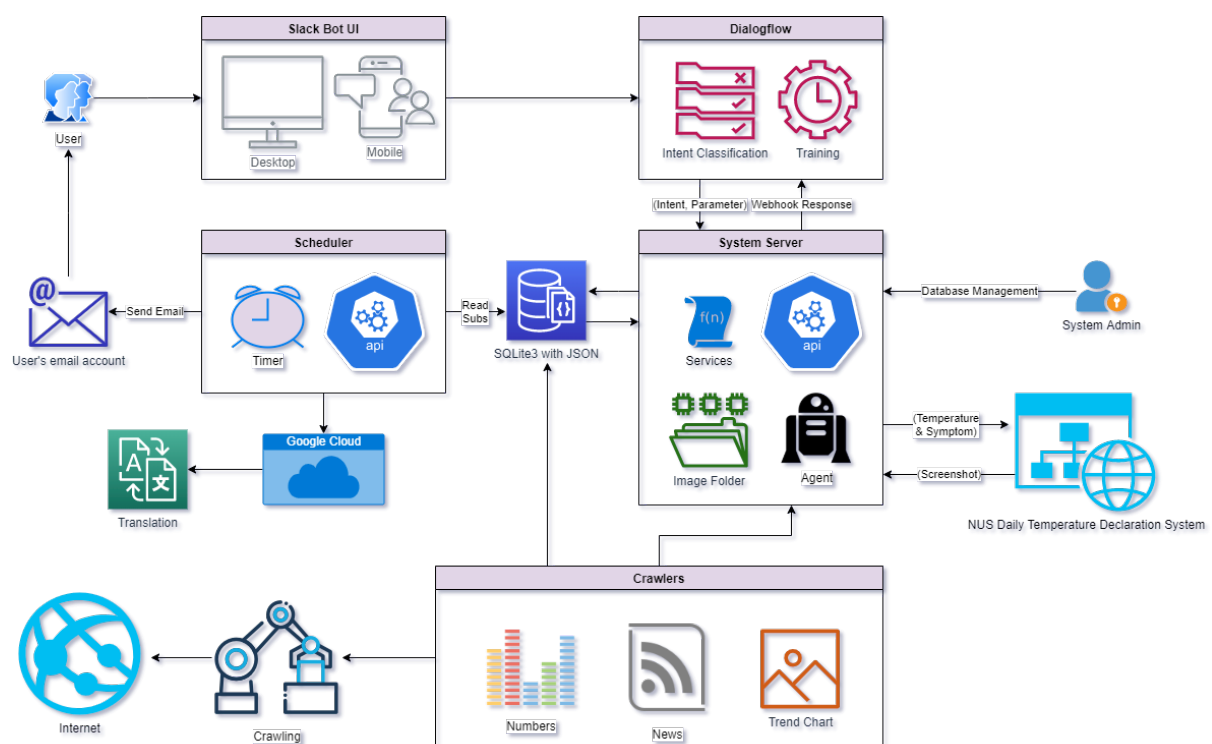


Figure 3.3: Detailed System Architecture

4 Implementation

4.1 Dialogflow

Dialogflow is used as the user input interceptor thanks to its effectiveness in the language processing. After several rounds of training, Dialogflow is able to detect the user's intention from his/her input text. In the scope of this project, the team defined the list of intents shown in Table 4.1.

Table 4.1: Dialogflow intents

Intent Name	Follow-up Intent	Remarks
Default Fallback Intent	-	Default system intent. Used to respond when encounters unknown phrases
Default Welcome Intent *	-	Default system intent. Used to respond when welcome users. It is customized so that it also reports the global COVID-19 status together with user input suggestions
CheckCovidStatus *	-	Check the COVID-19 status/data
CheckRecord *	-	Check the last screenshot after temperature declaring
CheckStatusByCountry	PopulateCountry *	Check the COVID-19 status/data
RetrieveNews *	-	Retrieve and respond the top 5 news from WHO
UpdateSubscription	PopulateEmailAddress * PopulateLanguagesList *	Allows user to update his/her subscription preference
UploadMyTemperature	PopulateTemperature * PopulateSymptom-yes * PopulateSymptom-no *	Allow user to declare his/her temperature and symptom on NUS Temperature Declaring Website

* Webhook enabled

The team has split all identified intents into 4 categories - Temperature Declaration, Subscription, News, and Country Data Query. There was a discussion within the team regarding the hierarchy of the intents. On one hand, a tree structure (Figure 4.1) - starting from the very first conversation, which allows the user to choose his/her intent by clicking among 4 buttons, provides more accuracy in detecting the user's intention. On the other hand, a flat design (Figure 4.2) assures the independence of different intent groups, and allows the user to directly express his/her intention from the first conversation. But this could potentially lead to a mis-classification of intent. To handle such a situation, the system treats all unidentified intents as default welcome intent, and provides suggestions in the response as guidance.

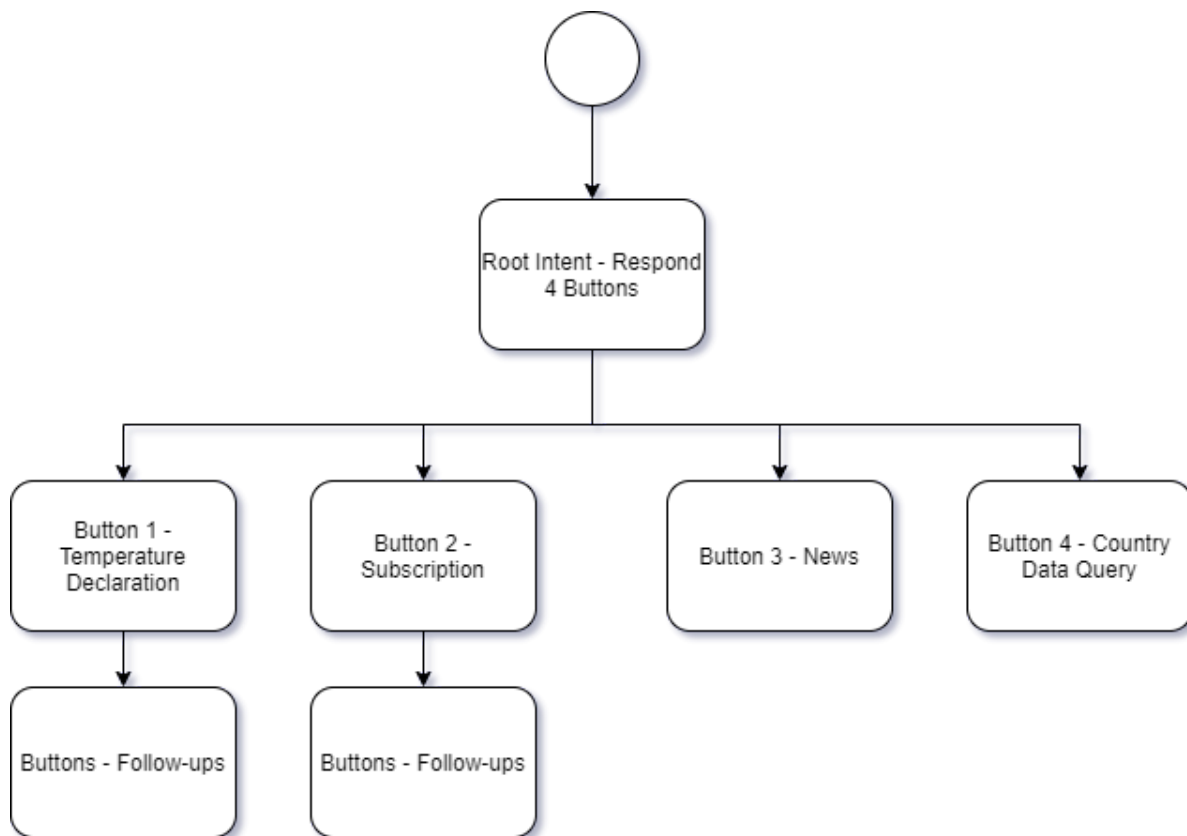


Figure 4.1: Tree hierarchy structure of intents

The team decided to use flat structure to utilize the language processing capability of Dialogflow. Starting from the first detected intent, follow-up intents are trained in the context to fulfill multi-step functions such as temperature uploading. Moreover, in order to fulfill the business logic, some of the intents need to be accomplished with a set of input parameters. Details are described in Table 4.2.

Webhook is enabled in order to complete most of the functions. Since Dialogflow requires a secure channel, ngrok is set up to tunnel main service HTTP port on localhost and forward messages to a global accessible HTTPS URL, which is then used by the Dialogflow Webhook component.

Generally, Dialogflow has shown its effectiveness in detecting user's intent in the project scope. However, there's also some limitations when use Dialogflow as the interceptor. Firstly, it is impractical to build triggers based on image input. Also, there's a strict 5 seconds timeout limit for Slack with Webhook response. There is some data that is crawled from different sources. It is impractical to get all of them in just 5 seconds. Therefore, the team implemented several functions following the asynchronous design pattern so that data can be captured and returned to the user in time.

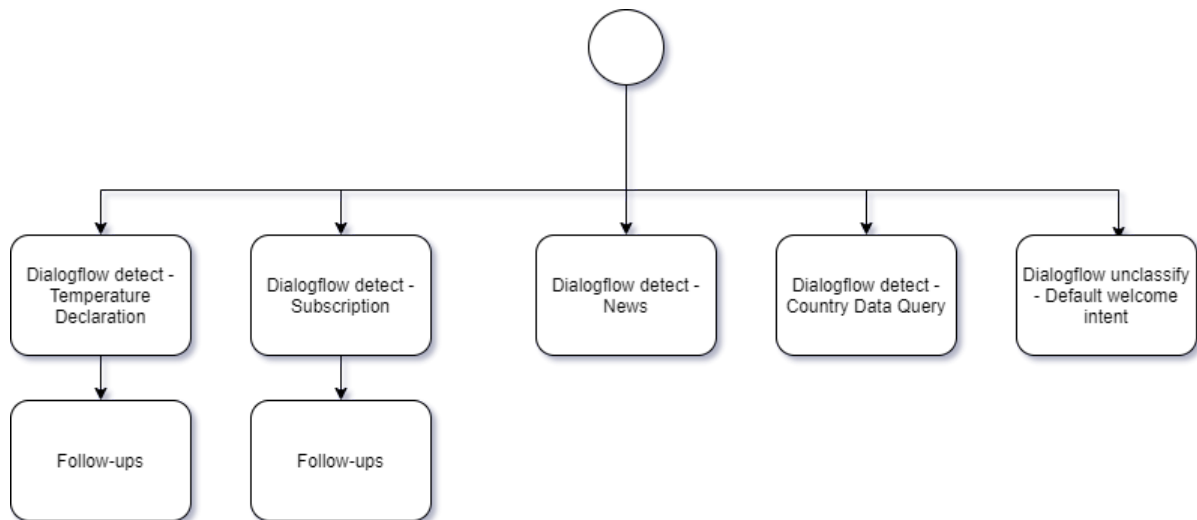


Figure 4.2: Flat hierarchy structure of intents

Table 4.2: Dialogflow intent parameters

Intent Name	Parameters	Remarks
CheckCovidStatus	country (M) and date (O)	If a date is not provided, the system will pick up the current date
CheckStatusByCountry	country (M)	System will pick up the current date
PopulateEmailAddress	email (M)	Populate the email address to subscribe news publication
PopulateLanguagesList	language list (M)	Populate the list of preferred languages to subscribe news publication
PopulateTemperature	temperature (M)	Populate temperature on NUS Temperature Declaring Website
PopulateSymptom-yes PopulateSymptom-no	symptom (M)	Populate temperature on NUS Temperature Declaring Website

M - Mandatory

O - Optional

4.2 System Server

Python was chosen as the programming language. Python is well known for its wide variety of libraries and ease of use. It also provides good integrability with other components like selenium and TagUI. The team used the Python Flask framework to set up the main service server. It contains several services that can be used by the frontend (Dialogflow) system. Once started, it listens to the service port: 5001 and processes incoming requests based on particular parameters within, such as the “intent” and “parameters” sent by Dialogflow. An “intent” is mapped to one or multiple services on the server. Each service fulfills a specific business logic.

The service server also initiates a selenium web driver whenever it starts. The web driver

subsequently reads the user’s credential of NUS DAILY TEMPERATURE DECLARATION from local. Ideally, the credential should be encrypted and stored in the database. Due to the project timeline limit, the team stored the user’s credential in a local json file. Once the web driver successfully gets the credential, it will attempt to login to the NUS DAILY TEMPERATURE DECLARATION and wait until the user triggers temperature upload intent. The team designed such an approach to facilitate the fulfillment of multi-steps service under the constraint of Webhook response timeout settings.

The team also used the Python Flask framework to set up the backend APIs (DB Operations).

Table 4.3: APIs

Components	API	Remarks
Frontend (Dialogflow)	/main	Communicate with Dialogflow agent
Backend (DB Operation)	/update_case	Insert crawled COVID-19 case statistics to DB
	/query_case	Query COVID-19 cases with date and country
	/init	Create table for crawled COVID-19 case by country
	/create_news	Create table for crawled COVID-19 news
	/create_subscriptions	Create table to record user’s subscription info
	/drop_table	Drop table

The “/main” API is used as the Dialogflow Webhook endpoint URL. Other APIs are meant to be used by the system administrator to maintain the database structure. Users are not expected to manipulate the backend database structure, though APIs are accessible via HTTP requests. Instead, users are expected to interact with the system through the frontend UI - Slack Bot and the scope is limited only to those defined “intent”s in Dialogflow.

4.3 Database

The team used SQLite to store all the application data. SQLite a small, fast, self-contained, high-reliability, full-featured, SQL database engine implemented by C-Language.

3 tables are created for User Subscription, COVID-19 news and COVID-19 statistics respectively. The data schema are summarised in Table 4.4–4.6 with primary key (PK).

To ensure the maximum flexibility for COVID statistics information (column ‘cov_info’), the team uses JSON string as the data structure. The advantage of using JSON is if attributes need to be updated, the underlying DB schema does not need to be altered.

Table 4.4: User Subscription

Column	Type	PK	Sample Content
to_email	text	Y	xxxx@gmail.com
date_stamp	text	N	2020-05-01 11:21:10
Language	text	N	Chinese, English

Table 4.5: COVID-19 news

Column	Type	PK	Sample content
date_stamp	text	Y	2020-05-01
news_title	text	Y	WHO Director-General's opening remarks at the media briefing on COVID-19 - 29 April 2020
news_link	text	N	https://www.who.int/dg/speeches/detail/who-director-general-s-opening-remarks-at-the-media-briefing-on-covid-19—29-april-2020
news_summary	text	N	The Emergency Committee was divided in its opinion, and did not advise that I declare a public health emergency of international concern.

Table 4.6: COVID-19 Statistics

Column	Type	PK	Sample Content
date_stamp	text	Y	2020-05-01
country_name	text	Y	Spain
cov_info	text	N	{ "total_cases": 1605712, " total_deaths": 95766, "total_recovered": 356977 }

4.4 Web RPA

Three different types of data need to be collected before answering users' queries.

It includes:

- Varies cases numbers
- Trend charts
- News

The first two are collected from Worldometer [3] and the last one is collected from WHO [4].

The team started with TagUI [6] to collect COVID-19 number data. There is a table in Worldometer which is structured for TagUI to save into a database. However, the charts were not successfully saved. The team then explored the raw html elements and found the charts showing trends are not static images. They are Scalable Vector Graphics (SVG), which is a container to define graphics [7].

The case trend data chart can also be generated using crawled data in the database. The system retrieves requested case data by country. Then by using the matplotlib, the system saves a plot chart in the local and exposes its path using Flask API, and this URL is populated in the Dialogflow fulfillment_message:image_uri in the response. Benefits & drawbacks of using data generated charts are compared with the selenium captured charts in Table 4.7.

Table 4.7: Comparison between data generated charts and selenium captured charts

	Selenium captured	Generated from data
Speed	Slow - initiated with crawling, web driver waits till the chart load up on website and internally parse by imgkit library	Fast - triggered by intent, retrieve from local database and parse by matplotlib library
Completeness	Good - data range from several months	Bad - data range from 5 consecutive days, and subject to data availability
Compatibility	Bad - imgkit library has dependency on the wkhtmltoimage package. Requires installation and configuration on Windows OS	Good - pip install matplotlib
Layout	Good - Refined and good presentation of data	Bad - system generated basic line chart with nearly zero customization
Size	Big - around 4 MB, and will not be rendered directly by Slack. User needs to click on “expand it anyway” to view the image.	Small - will be directly rendered by Slack

To get the charts, the team needs to get the svg element and render back to an image file. Since TagUI is not designed to crawl html elements, the team needs another method to get it. selenium is found to be able to get the svg elements. Once the svg element is crawled, imgkit is used to convert html to image.

To update NUS DAILY TEMPERATURE DECLARATION, TagUI was initially used. However, there is a special inbuilt function (detectWebdriver) in NUS website to detect whether the user is a robot or a human. Unfortunately, TagUI is not able to pass such a check. The team tested some other tools and found selenium is able to bypass it. Thus

selenium is chosen to update temperature.

The Webdriver component from selenium is able to identify the webelement by xpath, and get the user's credential and login into the system. After logging in, the temperature and symptom will be retrieved from user input with Dialogflow, selenium will key in the information and submit.

After temperature uploading, a screenshot of the record page is taken and saved to the local. The last screenshot taken will be loaded by the server so that it can be sent back if the user triggers a result check intent. The team noticed that an image file with the same name will be cached by the system so that whenever the user requests the result, only the first saved screenshot will be shown in the frontend even though it has been replaced by the latest one at the server side. To handle this problem, a time stamp is appended to the image name and the system will load the latest image based on the time sequence.

News are crawled by TagUI and goose3 is used to summarize news.

4.5 Scheduler

The scheduler is used to send system generated emails to users. It runs as a standalone thread and is independent from the main service server. Having been designed in such a way, the scheduler is able to perform well even if the main server is down.

The “send email” action is triggered by a time set by the system administrator and it is triggered on a daily basis. Once action is triggered, the scheduler will retrieve the latest news from the NEWS_INFO table. Then it will loop through the SUBSCRIPTION_INFO table and list out all subscribers' email addresses. For each address in the list, the scheduler will firstly find out what are the preferred languages set by the user. Then, for each language, the scheduler will invoke the Google Cloud API - translation function and translate the news content into the target subscription language if it is not English. Together with the news link, the content after translation will be sent to the subscriber's email address. Figure 4.3 shows the architecture of the scheduler.

Currently, the team has set only the subscription function under the scheduler. In the production environment, data crawling should ideally be grouped under the scheduler as well, so that the system can automatically refresh in every 24 hours. Considering that the project assessor needs to set up the system together with all data in place, the team sets aside all crawling functions, and requests the assessor to manually trigger the crawling functions for data setup.

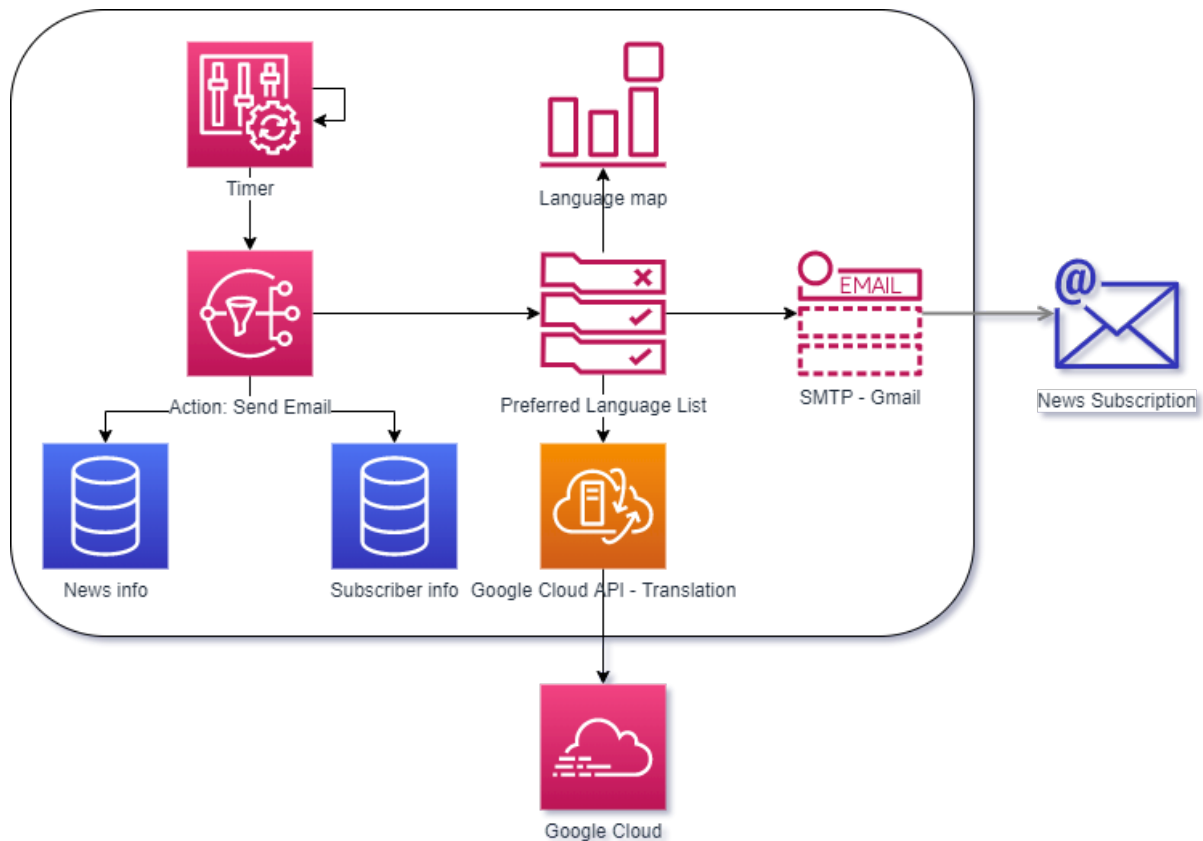


Figure 4.3: Scheduler Architecture

4.6 Slack

The team used Slack as UI for the system. There are two remarkable strengths while using Slack to integrate with Dialogflow. Firstly, Slack is available on both PC end and mobile end (Slack APP). Secondly, it is quite easy and straightforward to configure and test using Slack. Whereas for other APPs such as Facebook Messenger, it requires a complete setup of Page, call token and policy URL before hand.

Google Assistant provides even greater integrability with Dialogflow. More features such as voice input, and rich media response are supported by Google Assistant, but not Slack. Due to the time constraint of this project, the team decided to use Slack as the system UI.

5 Performance & Validation

We performed validation on 3 different scenarios to ensure that the system provides the correct expected output.

Scenario 1 was to test whether the system is able to answer the user's query regarding the COVID-19 status by country.

Scenario 2 was to test whether the system is able to register user's temperature on NUS's website.

Scenario 3 was on the extreme case where the user is not eligible for any Credit Cards.

It should be able to fulfil the below 3 features at the user's request:

1. Answer user's query regarding the COVID-19 status by country

The backend DB will be updated with all countries' COVID-19 data by Web RPA every morning. The data will be used to reply user's queries.

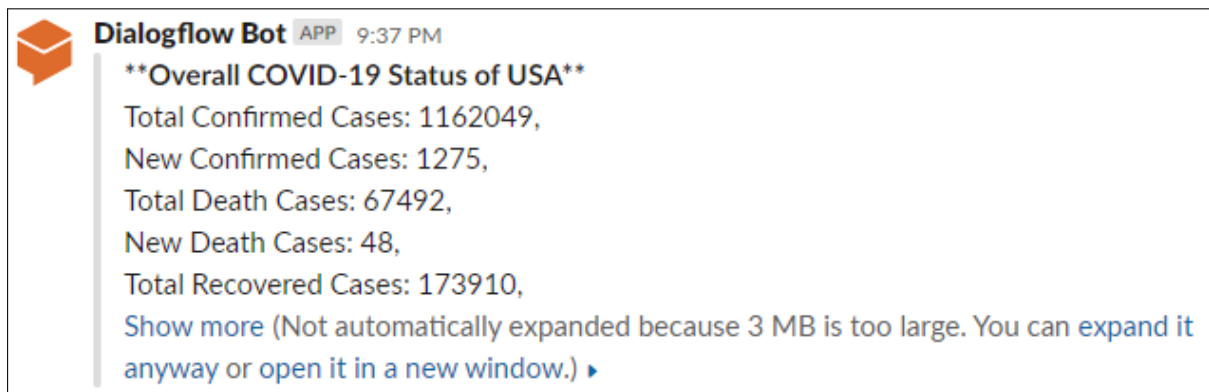


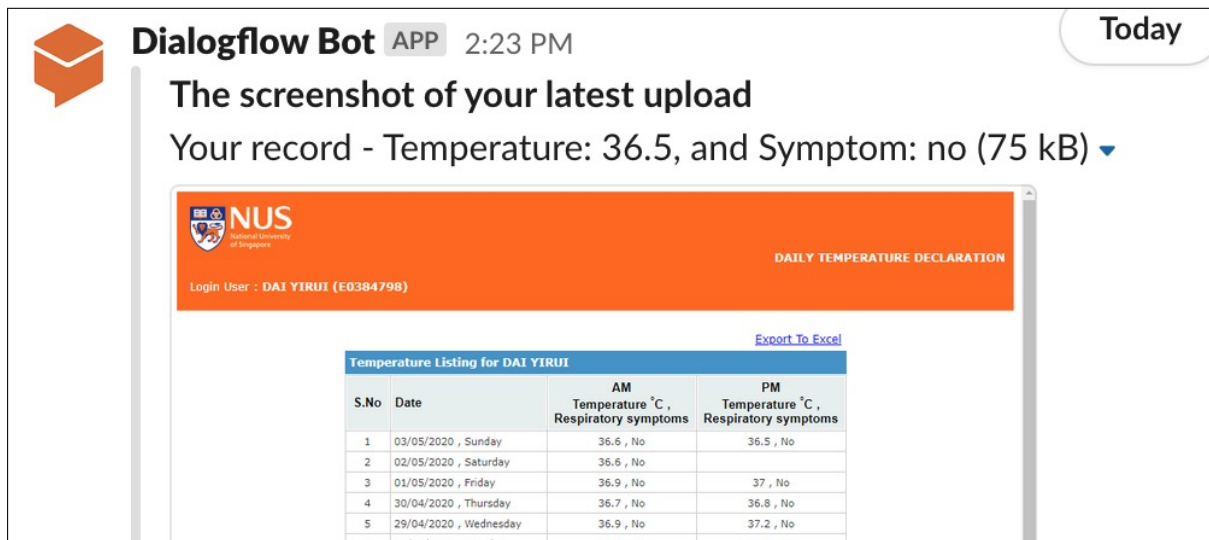
Figure 5.1: Bot response of COVID-19 related data of USA

2. Register user's temperature on NUS's website

It takes in the temperature value and helps enter it in the NUS website to finish the registration.

3. Periodically push COVID-19 breaking news to users.

4. Register user's subscription preference and push latest news to user's email account according to user's preference.



Dialogflow Bot APP 2:23 PM Today

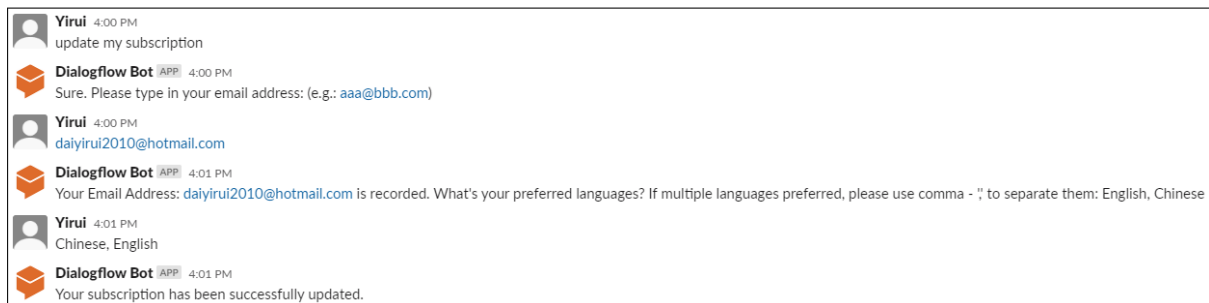
The screenshot of your latest upload
Your record - Temperature: 36.5, and Symptom: no (75 kB) ▾

NUS National University of Singapore
Login User : DAI YIRUI (E0384798)

[Export To Excel](#)

S.No	Date	AM Temperature °C, Respiratory symptoms	PM Temperature °C, Respiratory symptoms
1	03/05/2020, Sunday	36.6, No	36.5, No
2	02/05/2020, Saturday	36.6, No	
3	01/05/2020, Friday	36.9, No	37, No
4	30/04/2020, Thursday	36.7, No	36.8, No
5	29/04/2020, Wednesday	36.9, No	37.2, No

Figure 5.2: Bot response of temperature records of the user



Yirui 4:00 PM
update my subscription

Dialogflow Bot APP 4:00 PM
Sure. Please type in your email address: (e.g.: aaa@bbb.com)

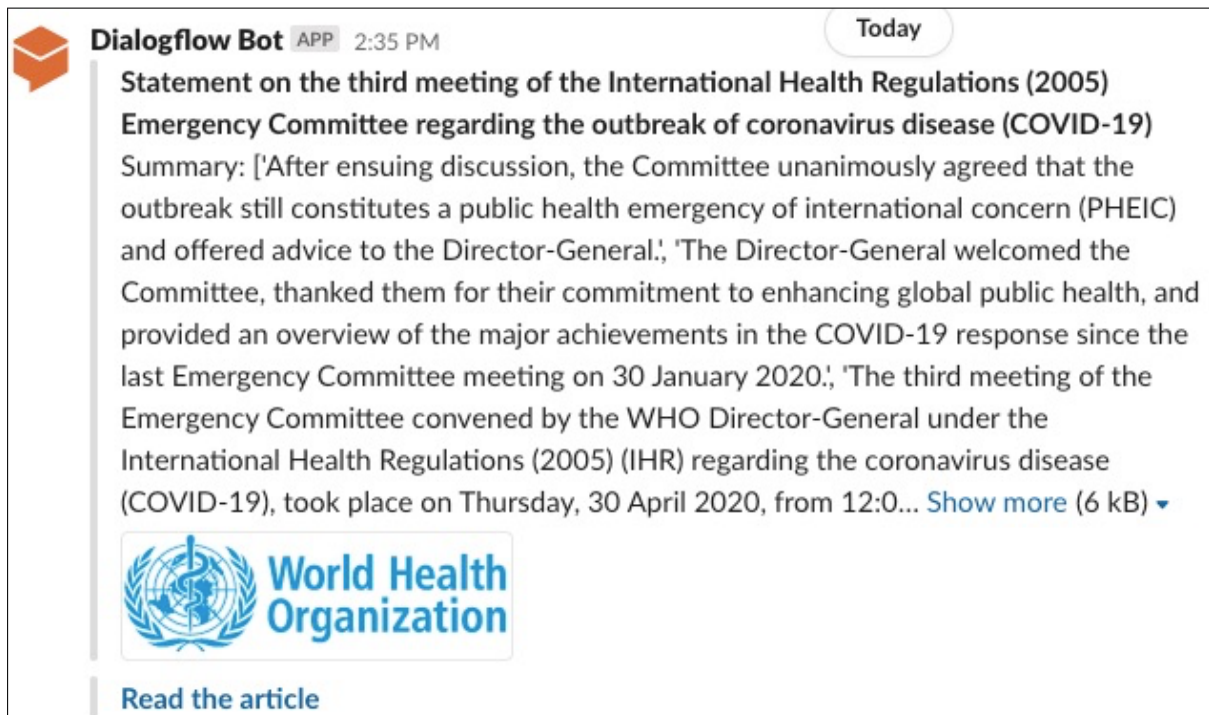
Yirui 4:00 PM
daiyirui2010@hotmail.com

Dialogflow Bot APP 4:01 PM
Your Email Address: daiyirui2010@hotmail.com is recorded. What's your preferred languages? If multiple languages preferred, please use comma - ',' to separate them: English, Chinese

Yirui 4:01 PM
Chinese, English


Dialogflow Bot APP 4:01 PM
Your subscription has been successfully updated.

Figure 5.3: Conversation between user and bot regarding the subscription updates



Dialogflow Bot APP 2:35 PM Today

Statement on the third meeting of the International Health Regulations (2005) Emergency Committee regarding the outbreak of coronavirus disease (COVID-19)
Summary: ['After ensuing discussion, the Committee unanimously agreed that the outbreak still constitutes a public health emergency of international concern (PHEIC) and offered advice to the Director-General.', 'The Director-General welcomed the Committee, thanked them for their commitment to enhancing global public health, and provided an overview of the major achievements in the COVID-19 response since the last Emergency Committee meeting on 30 January 2020.', 'The third meeting of the Emergency Committee convened by the WHO Director-General under the International Health Regulations (2005) (IHR) regarding the coronavirus disease (COVID-19), took place on Thursday, 30 April 2020, from 12:0... [Show more](#) (6 kB) ▾

 **World Health Organization**

[Read the article](#)

Figure 5.4: Bot response of COVID-19 related news

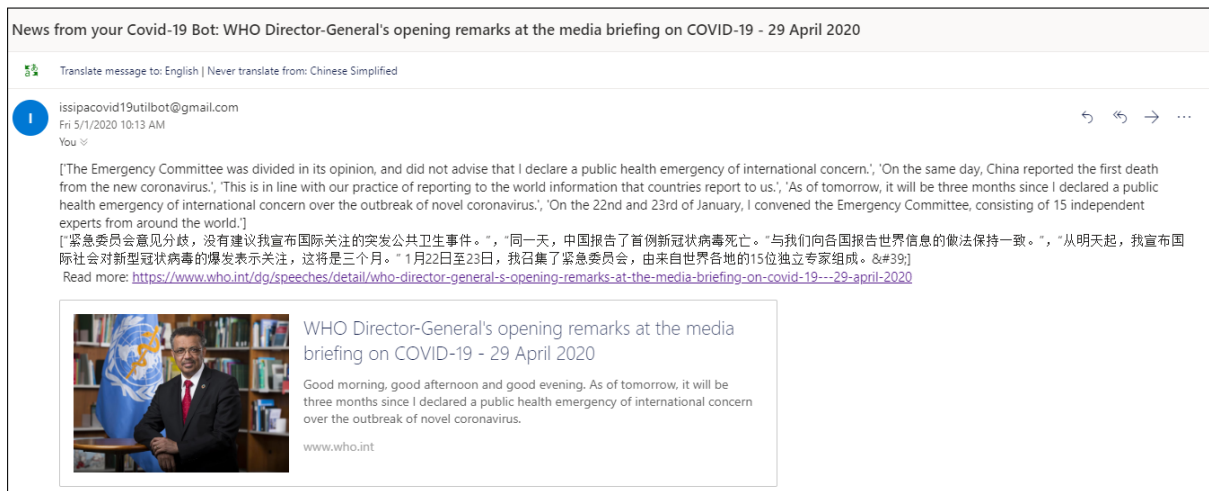


Figure 5.5: Email sent by the bot according to user's preference

6 Conclusions

Our system, COVID-19 Assistant, will collect COVID-19 information and provide users with the relevant information. It is friendly to users with a simple Slack chatbot UI. Users only need to type the queries then the system will answer. The information is personalized so users are able to get the information they need most, especially in this era of information explosion. To NUS users, it even provides the temperature update function so NUS students and staff no longer need to open web pages to fill temperature everyday.

6.1 Findings

- There are anti-crawling mechanisms implemented in some of the websites, where TagUI will be blocked, thus a higher level of “walk-around” approach needs to be taken in order to achieve the desired outcome.
- Selenium provides higher flexibility compared with TagUI, for example, it is able to create dummy header information to bypass some anti-crawling checks.
- In Dialogflow, it is impractical to build and train intent that is based on image input. Generally speaking, development of Dialogflow focus more on utilization of its NLP capability. In addition, there’s a 5 second timeout constraint on Webhook response between Dialogflow and Slack. Many functions in this project scope are thus designed and implemented in an asynchronous approach.
- The team has explored OCR functions using Cloud AI (Google Cloud API - vision), since reading temperature from an image of a thermometer is a good use case in this context. The performance was not ideal. Cloud AI tends to output all texts detected from the image, including the thermometer brand ect. Although it is possible to filter out the true temperature reading, it does not ensure 100% accuracy.
- Moreover, temperature reading by Cloud AI varies a lot from the true reading due to factors like lightness and shooting angle. The team considered OCR using Local AI also. But it will take tremendous time to train, which is not a good trade off in this project scope.

- The team has experimented with the use of a multi-thread process to handle asynchronous multi-steps service. The strategy works fine until the converge step. The team used some global variables to store persistent information such as temperature, so that the parameter can be used in later steps. But the team noticed that global variables do not converge and maintain well under the multi-thread context. Instead, each thread should hold its own variables. In addition, management of different thread lifespan within the server is tedious. It is also not the focus of this project. The team decided to delegate thread management to the Flask framework and continue using global variables to store persistent information.

6.2 Recommendations

In order to have a better experience as a virtual assistant, NLP models need to be brought in to have a better understanding of users' requests, for example, users may ask the questions in a casual manner, if the pre-built questions do not cover those, the agent may not be able to response with meaningful content, while with NLP models, it would capture the topics from the question, and response accordingly with a higher accuracy.

7 References

- [1] E. Callaway, “Time to use the p-word? coronavirus enter dangerous new phase”, *Nature*, vol. 579, no. 277, p. 12, 2020.
- [2] M. Mohan. (2020). Covid-19 circuit breaker extended until jun 1 as singapore aims to bring down community cases ‘decisively’: Pm lee, [Online]. Available: <https://www.channelnewsasia.com/news/singapore/covid-19-circuit-breaker-extended-june-pm-lee-speech-apr-21-12662054>.
- [3] Worldometer. (2020). Covid-19 virus pandemic - worldometer, [Online]. Available: <https://www.worldometers.info/coronavirus/>.
- [4] WHO. (2020). Coronavirus disease (covid-19) news, [Online]. Available: <https://www.who.int/emergencies/diseases/novel-coronavirus-2019/media-resources/news>.
- [5] G. Schreiber, *Knowledge Engineering And Management (mit Press)*. Universities Press, 2001.
- [6] kelaberetiv. (2020). Command-line tool for digital process automation (rpa), [Online]. Available: <https://github.com/kelaberetiv/TagUI>.
- [7] w3schools. (2020). Html svg, [Online]. Available: https://www.w3schools.com/html/html5_svg.asp.

A User's Manual

A.1 Prerequisite

A.1.1 Accounts

To use Dialogflow and Slack, the following accounts are required:

- Gmail Account
- Slack Account

If you do not have these two accounts, please go to Slack (<https://slack.com/signin>) or Google (<https://accounts.google.com/signUp>) to register.

A.1.2 Codes & Packages

A.1.2.1 Download Project Code Files

Download project from **GitHub**

(<https://github.com/ddgone2far/ISA-IPA-2020-03-11-IS1PT-GRP-High5.git>).

A.1.2.2 Setup Environment

1. Setup Anaconda Environment

```
$> conda create -n ipa python=3.7
$> conda activate ipa
(ipa) $> pip install flask pytz tagui goose3 schedule google-cloud-translate
selenium matplotlib imgkit
```

2. Setup imgkit Library

To run imgkit on Windows OS, user needs to install binary - wkhtmltopdf and configure its path in the file

```
~/ISA-IPA-2020-03-11-IS1PT-GRP-High5/dialogflow_python_backend/chart_crawler.py
```

At line 54

```
path_wkhtmltoimage = r'D:\wkhtmltopdf\bin\wkhtmltoimage.exe'
```

3. Setup Chromedriver

Get Chromedriver from <https://chromedriver.chromium.org/downloads>.

Install one that fits with your OS at

```
~/ISA-IPA-2020-03-11-IS1PT-GRP-High5/dialogflow_python_backend
```

4. Setup ngrok Download ngrok from <https://ngrok.com/download> and unzip to local

```
~/ngrok
```

A.2 Server Setup

A.2.1 Crawl Data from Website

Crawling is independent from running the main app. It is required to run by the system administrator once a day to scrap relevant information. In order to give more flexibility to the admin, this function is not set under scheduler, so that admin can run as many times as they want.

1. Change directory to the python project

```
(ipa) $> cd ~/ISA-IPA-2020-03-11-IS1PT-GRP-High5/dialogflow_python_backend
```

2. Crawl COVID-19 related data: Total confirmed/new/death case (Global/Countries)

```
(ipa) $> (ipa) $> python crawler.py
```

Note: a new window will popup. Some warning message: “Exception happened while trying to convert for integer, setting value to 0” will show in the conda console. You can ignore it as it happens when the system encounters empty data. The whole crawling process can last over 10 minutes. You may intervene at any time but it will lead to loss of some information in the database. “OK” will be shown in the console once the crawling process completes.

```
Exception happened while trying to convert for integer, setting value to 0.
```

```
Exception happened while trying to convert for to integer, setting value to 0.  
Exception happened while trying to convert for to integer, setting value to 0.  
OK
```

3. Crawl COVID-19 related news (Limited to 5 News)

```
(ipa) $> python news_crawler.py
```

Note: a new window will popup. The whole process will last around 20-30 seconds.

4. Crawl COVID-19 related trend charts

```
(ipa) $> python chart_crawler.py
```

Note: a new window will pop up. A TagUI agent will firstly attempt to get a list of countries and their corresponding link to the trend chart. Based on the list, another Selenium web driver will crawl trend charts for all countries and save them to local. These charts will be used once user queries about COVID-19 cases of a specific country. The whole crawling process can last over 10 minutes. You may intervene at any time but it will lead to loss of trend charts for some countries.

A.2.2 Start Up Main Server & Scheduler

1. Update the NUS Temperature Declaring System Credentials at local

Change the email/password in the JSON file to yours:

```
~/ISA-IPA-2020-03-11-ISIPT-GRP-High5/dialogflow_python_backend/email_info.json
```

Credential file content:

```
{  
  "1":{  
    "email":"nusstu\\NUSNETID",  
    "password":"NUSNETPASSWORD"  
  }  
}
```

Note: Credentials are only saved in the local environment. It is not accessible from the public. if you are nus staff, kindly use “nusstf” as prefix, and follow by double slashes.

2. Start the main server

```
(ipa) $> python app.py
```

Note: everytime when app.py runs, it will be 1-2 windows popup. At any time, please do not close or intervene with these windows. At the same time, any incoming messages will be printed in the Anaconda console. If credential is properly configured and authenticated, you will be able to see the temperature declaring page in the popup window(s)

- All students who are allowed to come to campus when circuit breaker measures are in place, and students staying in hostels need to continue to declare your temperature twice a day.
- Although staff will not be coming to campus until after 1 June because of circuit breaker measures, all staff are to continue to declare their temperature twice a day unless they have been exempted.

I declare that the following information is correct and accurate.

Date : 01/05/2020, Friday PM ▾

My temperature reading is : ° C

Do you have cough, runny nose or sore throat that you have recently just acquired and is/are not due to pre-existing conditions? : ☐ No ☐ Yes

- Please be reminded to take photographs of the thermometer reading with a date-time stamp, and to retain the images for seven days, as you may be asked to upload the images for verification.
- The camera images from your mobile phone will already contain information about your device, as well as details on the date and time that the image was taken.

3. Start the scheduler (Optional)

Configure your own Google Cloud Service, edit line 61 and 62. Put your own Google Cloud Credential file and Project ID.

```
def input_text_translation(text, target):
    try:
        client = translate.TranslationServiceClient(
            from_service_account_json(
                # Google Cloud API Credential File
                "D:/GoogleCloud/ipadaiyirui001-9cad5c929128.json"
            )
        )
        parent = client.location_path(
            # Google Cloud Project ID
            'ipadaiyirui001',
            'global')
    except:
```

Initiate a new Anaconda window

```
$> conda activate ipa
```

Change directory to the python project

```
(ipa) $> cd ~/ISA-IPA-2020-03-11-IS1PT-GRP-High5/dialogflow_python_backend
```

Run the scheduler

```
(ipa) $> python scheduler.py
```

Note: there is no log being printed out in the console, instead, it will keep running

A.2.3 Start 'ngrok' Service

1. Start a new command prompt, and change to your ngrok directory
2. Start ngrok http service on localhost:5001

For Mac or Unix/Linux:

```
~ngrok $> ./ngrok http 5001
```

For Windows:

```
~ngrok $> ngrok http 5001
```

Note down the https service URL:

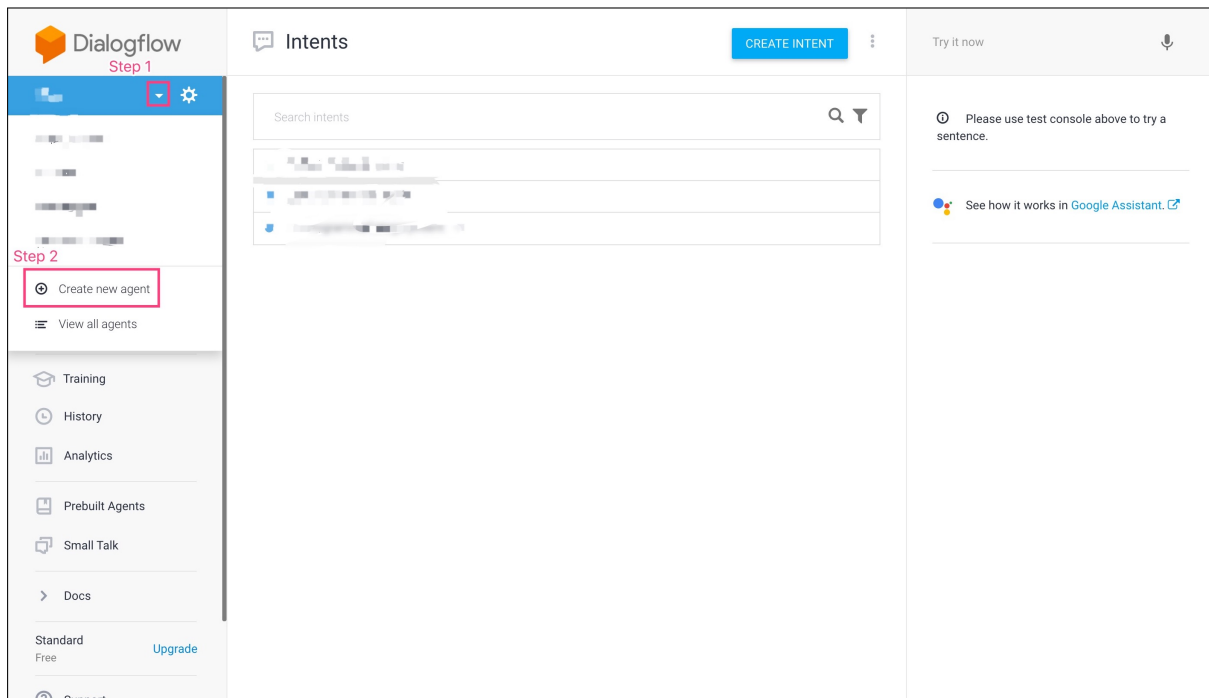
```
ngrok by @inconshreveable

Session Status      online
Session Expires     2 hours, 36 minutes
Version             2.3.35
Region              United States (us)
Web Interface        http://127.0.0.1:4040
Forwarding           http://fb6bdc76.ngrok.io -> http://localhost:5001
Forwarding           https://fb6bdc76.ngrok.io -> http://localhost:5001
```

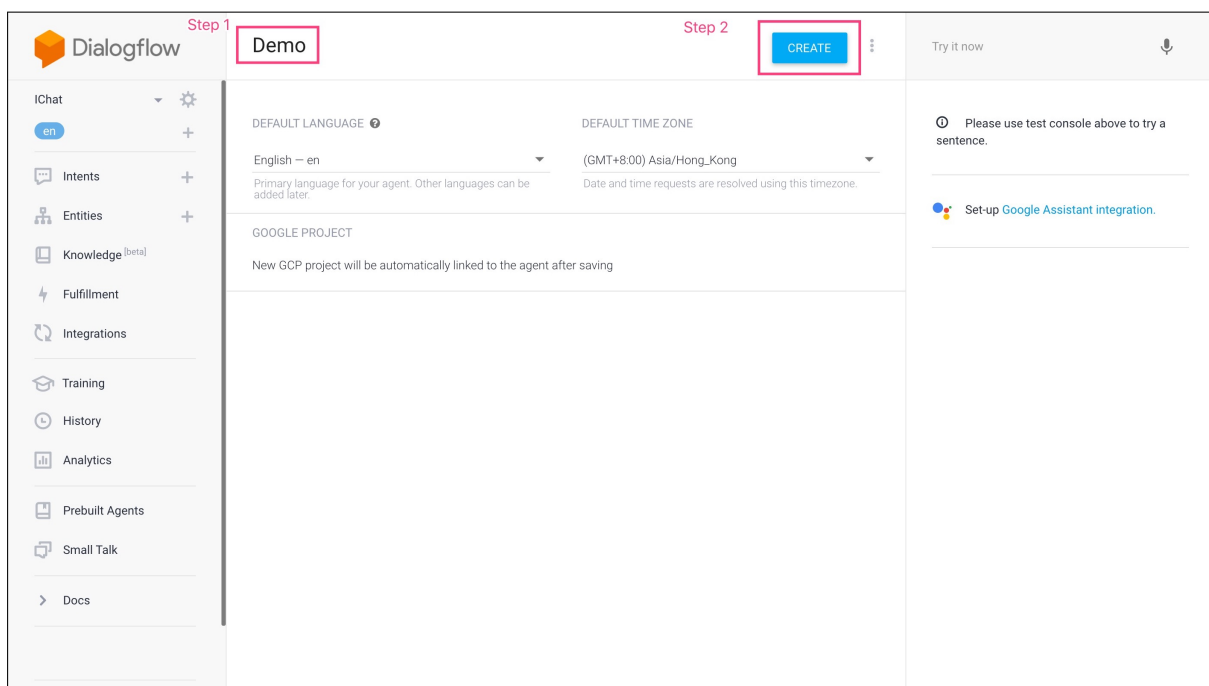

A.3 Dialogflow Setup

A.3.1 Import Agent

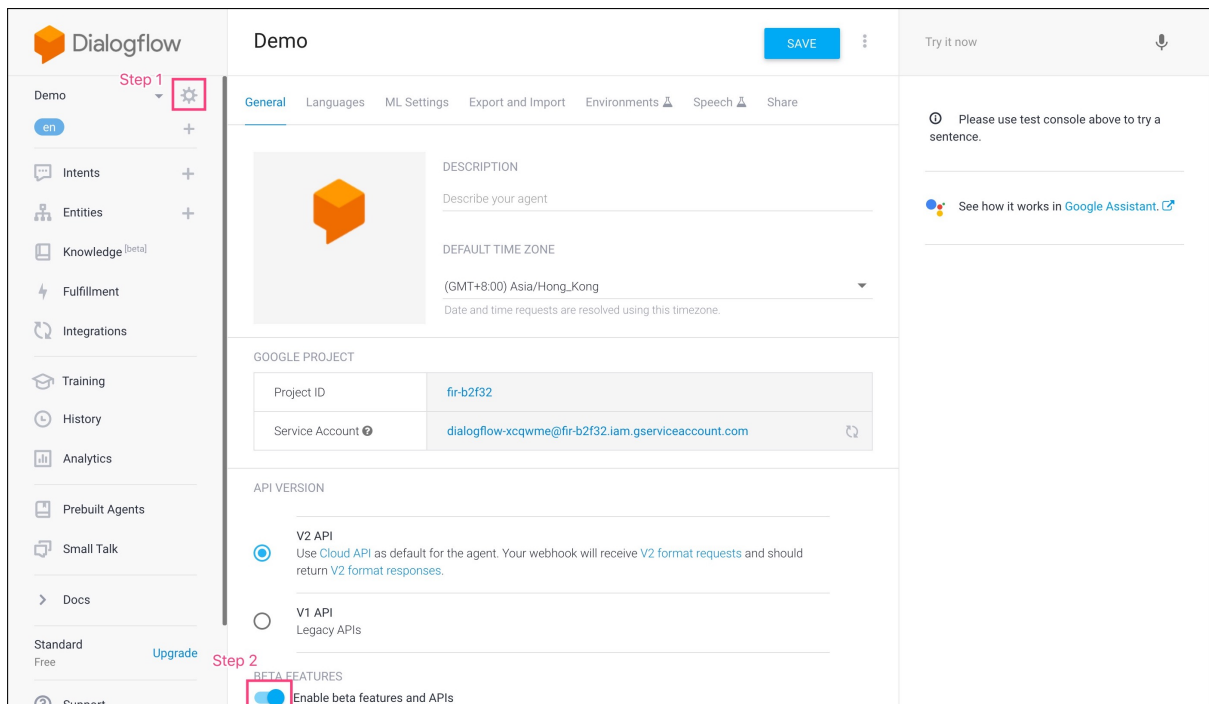
1. Go to Dialogflow Console (<https://console.dialogflow.com>), then click “Create new agent”



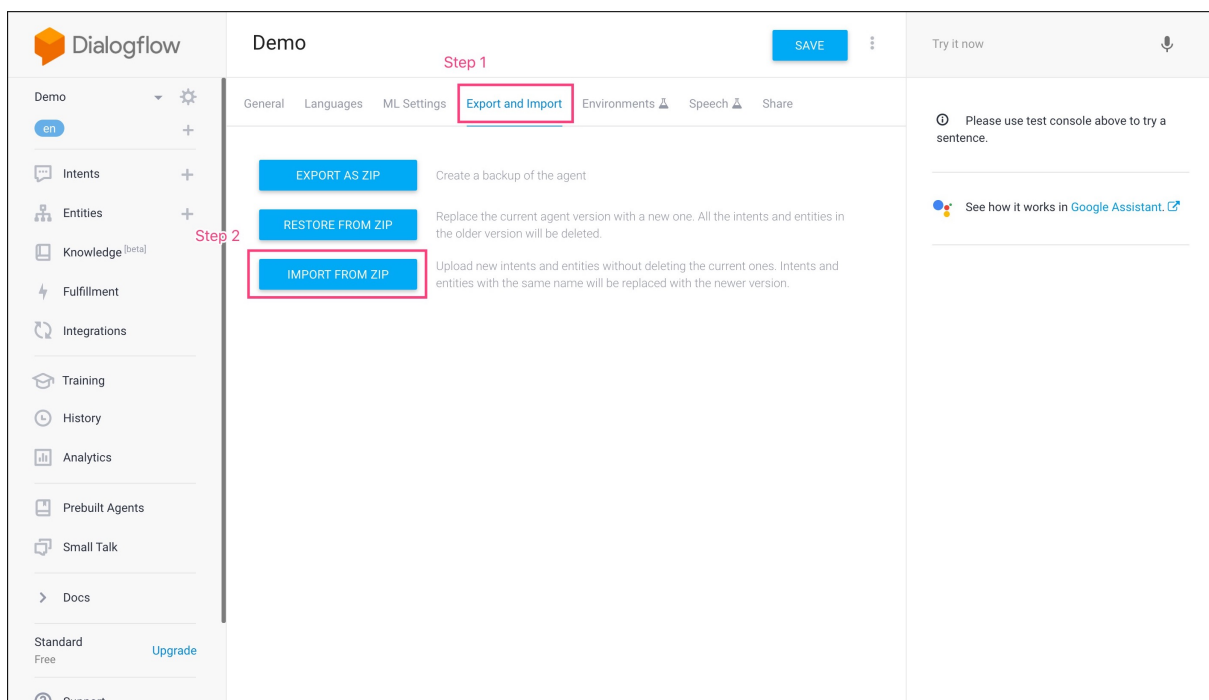
2. Put Agent name as you wish, then click “CREATE” button



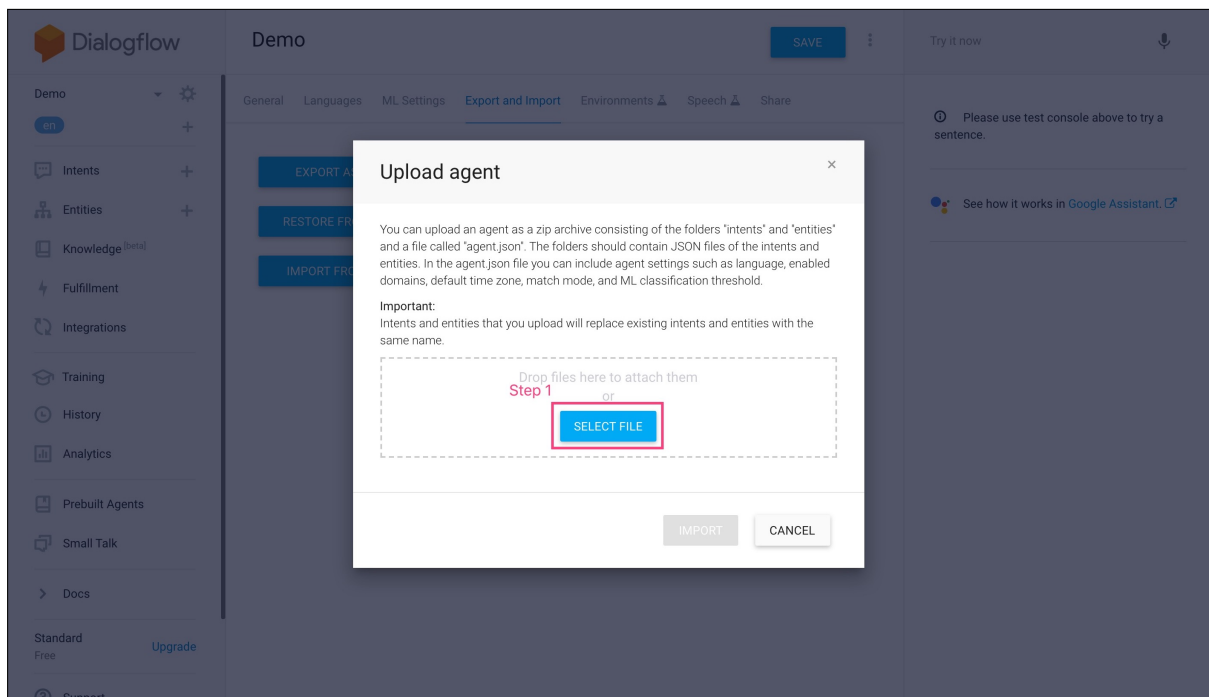
3. Go to Settings and enable beta features and APIs



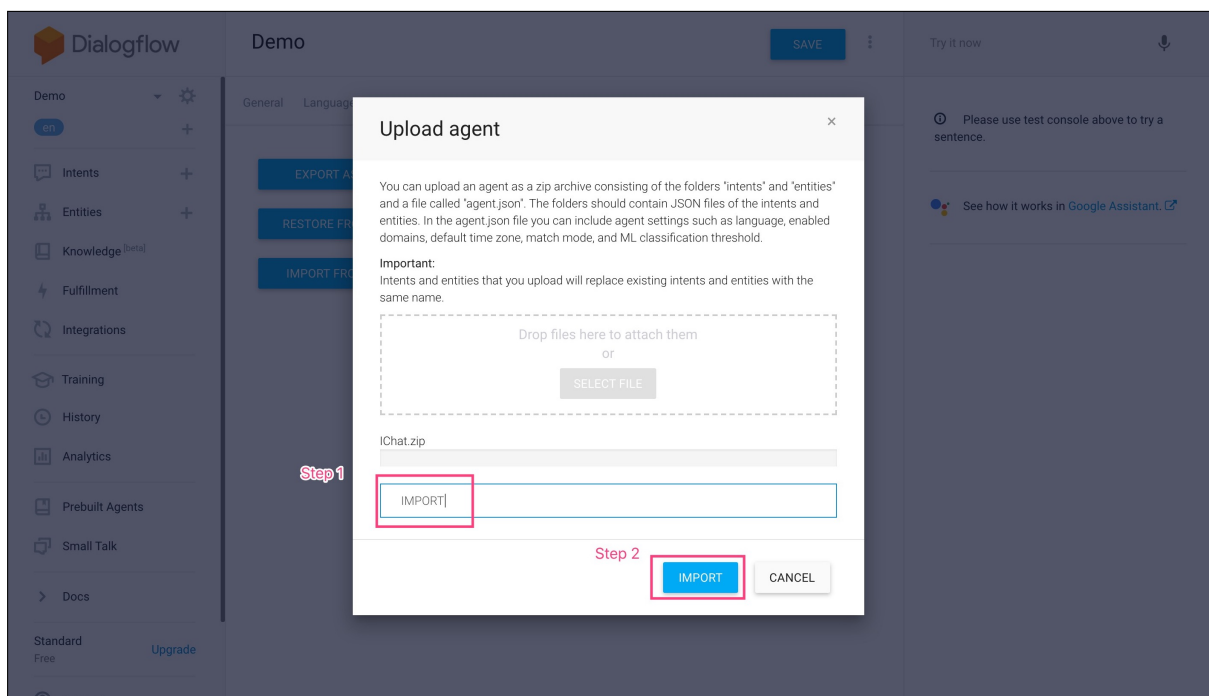
4. Go to “Export and Import” and click “IMPORT FROM ZIP”



5. Click “SELECT FILE”



6. Choose “COVID19UTIL_BOT.zip”, type IMPORT and click “IMPORT” button. Click “Done” button once imported.

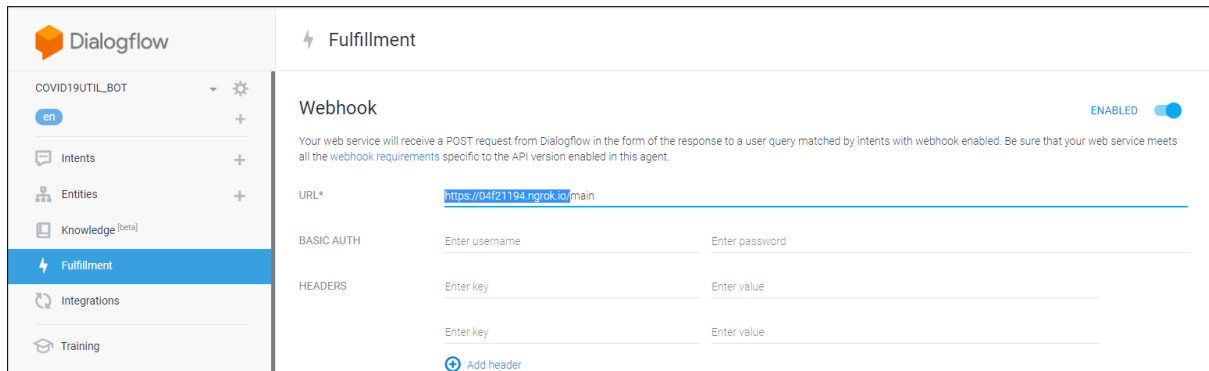


So far the “Intent” should be all imported into this agent. Let’s continue to set up Fulfillment by Webhook.

A.3.2 Create Fulfilment

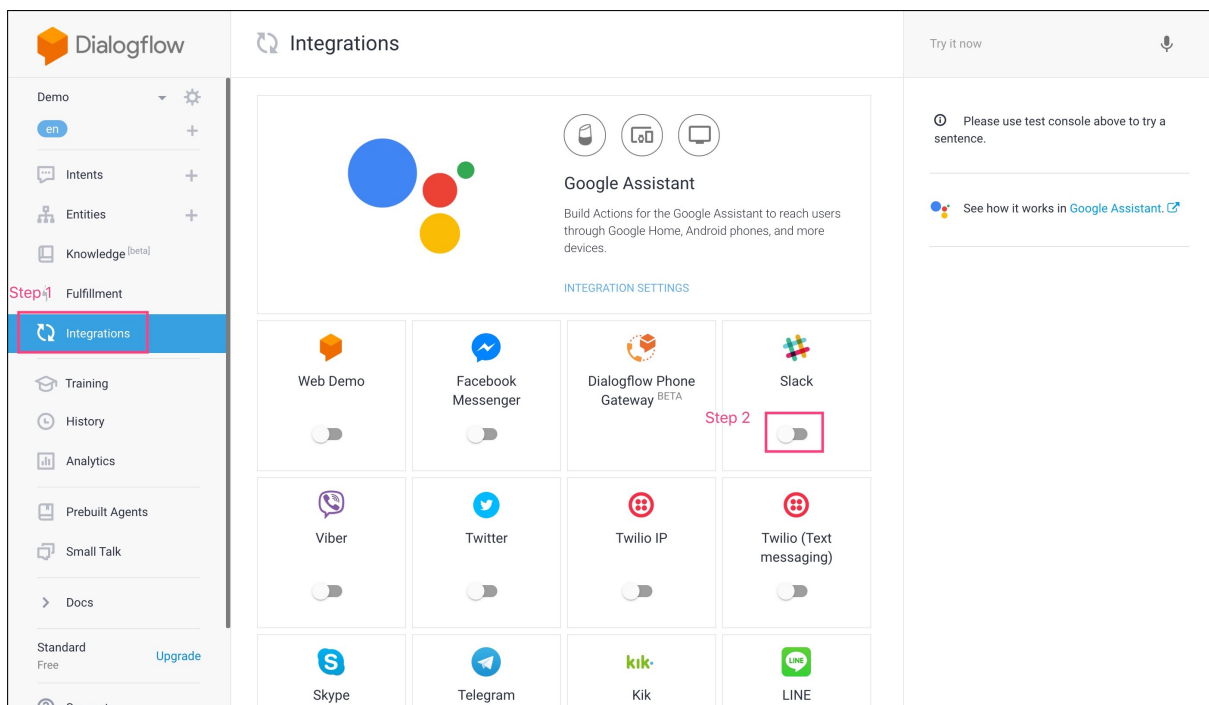
1. Click “Fulfillment” and update the Webhook URL with the one generated by ngrok service. Then click “Save”.

Note: there’s a “/main” after the ngrok service URL

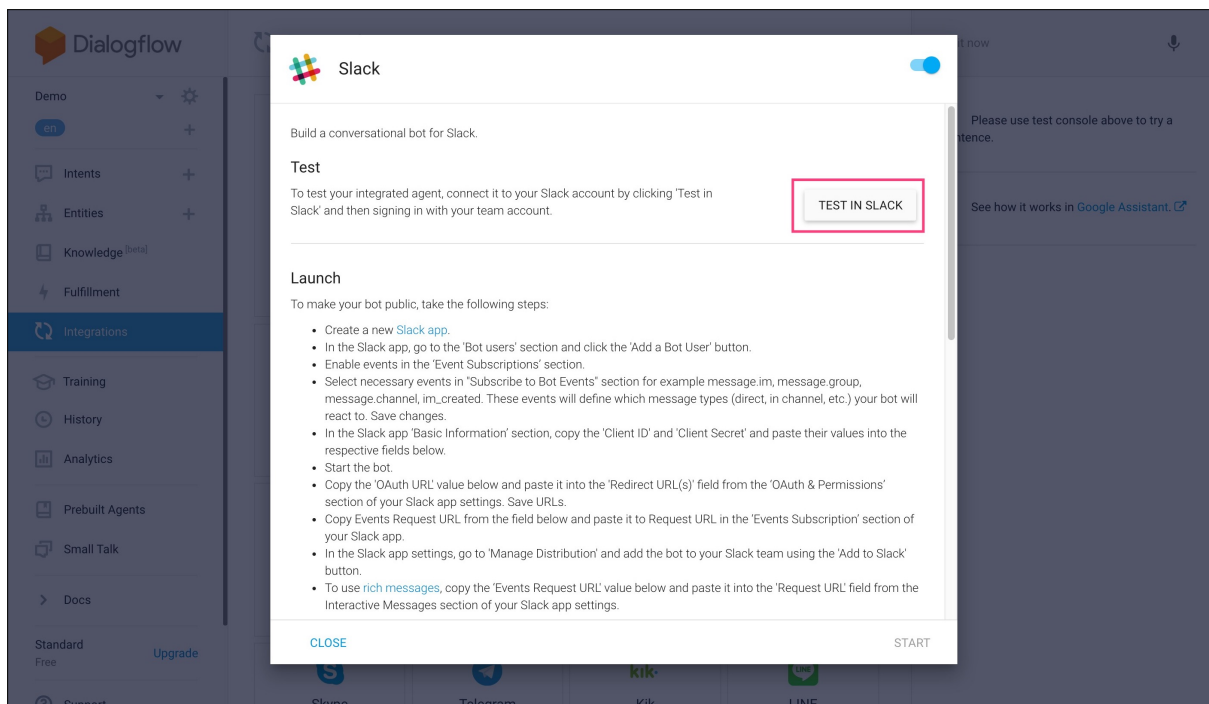


A.3.3 Integration

1. Click “Integrations” and then enable “Slack”



2. Follow the “Launch” to use, or click “TEST IN SLACK” with proper Slack set-up for testing






A.4 Test Demo

If Dialogflow is successfully integrated with Slack, you can say “hi” or other welcome words, and COVID-19 Assistant will answer you enquiries.

app.slack.com/client/TE9HYMU12/D011AM0FB71

Search mtech2019pt

mtech2019pt  

 Dai Yirui

- Threads
- Mentions & reactions
- Drafts
- Saved items
- People
- Apps
- Files
- Show less

Channels +

- # general
- # random
- + Add a channel

Direct messages +

- Slackbot
- Yirui (you)
- + Invite people

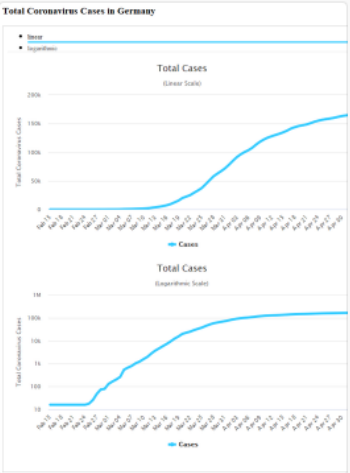
Apps +


- Dialogflow Bot**


Dialogflow Bot ☆

Messages About

Total Coronavirus Cases in Germany











 **Yirui** 7:23 PM
hi

 **Dialogflow Bot** APP 7:23 PM

****Hello, worldwide COVID-19 Status:****
 Total Confirmed Cases: 3484502,
 Total Death Cases: 244781,
 Total Recovered Cases: 1121608.
****Input one of below options to continue:****
 "Upload Temperature" - Upload your temperature to NUS Temperature Declaring System.
 "Update Subscription" - Update your subscription preference.
 "COVID-19 Data by Country" - Check COVID-19 Data by Input Country.
 "Latest News" - Get latest News regarding COVID-19.

Message Dialogflow Bot

 **B** *I*       

B Individual Report

B.1 GUO FENG

B.1.1 Personal Contribution

At the brainstorming session for project ideas, I have proposed a potential project which serves as a career coaching agent for graduates: based on their personal interest and background, to find out their skill gap with the market job requirements, and recommend the training courses to them, so that they are better prepared for their job hunting.

Later the team had put up a joint discussion and we decided to take another idea which is closely related with our day-to-day situations, we need an COVID-19 intelligent assistant to keep us updated and save us the hassle of updating temperatures everyday.

In the implementation phase, I worked on the setting up of backend infrastructure, which covered database schema design, DB management utility functions and services APIs to integrate with various components such as diagflow agents, crawlers.

B.1.2 What I have learnt

- RestfulAPI Framework (Flask)
- Lightweight DB: SQLite
- Crawler tools such as selenium, TagUI
- Typical tricks to bypass website anti-crawling mechanisms.
- Integration of chatbot agent (dialogflow) to slack front end and Python Flask back-end
- Overall system design and integration experience, so as to build functions as loosely coupled components but could be easily integrated to deliver the end-to-end solution.

B.1.3 How I can apply the knowledge and skills in other situations

In my company, our business users need to do a lot of tedious manual work daily, e.g:

- Have to switch to multiple websites to get the respective content, and manually consolidate them together
- Receive multiple data files in various formats from multiple sources, do eyeball checking and ensure correctness
- Need to constantly maintain the versioning but still run into data inconsistency among themselves.

I could propose to them to adopt the approaches in our project:

- Build crawler scripts to get the data from company internal web applications periodically, and store into a shared DB
- Build a Restful API with flask which offers the data services for all users to query
- Build automation scripts with TagUI/UIPath, to reduce the manual effort of manipulating files

B.2 DAI YIRUI

B.2.1 Personal Contribution

At the brainstorming session for project ideas, I have proposed a potential project which functions as a dressup and makeup advisor that gives recommendations to users regarding what to wear before they hangout. Users need to upload all images of his/her clothes, pants, makeup and shoes into the database. Then the system will crawl the latest trend on fashion websites and propose to the user with a closest match. The system also takes other factors into consideration such as forecasted weather and temperature, purpose of hangout (party, ceremony etc) and recommends accordingly. This system involves features like image recognition, fuzzy classification, RPA and supervised learning. It is fairly difficult to implement in a short period of time.

Later the team had put up a joint discussion and we decided to take another idea which is closely related with our day-to-day situations, we need an COVID-19 intelligent assistant to keep us updated and save us the hassle of updating temperatures everyday.

In the implementation phase, I worked on the Dialogflow setup and training, Slack integration with Dialogflow, coding of services on server side and integration of different components, and setup the scheduler with email function.

B.2.2 What I have learnt

- Dialogflow with Slack and its limitation
- Crawler tools such as selenium, TagUI and their goods/bads
- Session, cache and thread management at server side.
- Standalone scheduler with SMTP function
- Cloud AI with Google Cloud API - translate and OCR
- Overall system design and integration experience, so as to build functions as loosely coupled components but could be easily integrated to deliver the end-to-end solution.

B.2.3 How I can apply the knowledge and skills in other situations

- It's good to consider Bot (Slack, Facebook Messenger) integrated with Dialogflow for NLP purpose
- Use of RPA tools for web crawling. Use of Cloud AI for translation and OCR purpose
- Use of scheduler plus RPA for tedious daily operations
- System design and infrastructure. Good to analyze limitations of different systems and come up with the design (e.g. sync or async)
- Deeper understanding of quick POC on server side - Flask, session, cache and thread

B.3 LU JIAHAO

B.3.1 Personal Contribution

At the brainstorming session for project ideas, I have proposed a potential project which could be used to update NUS DAILY TEMPERATURE DECLARATION based on user

uploaded images. However, we tried some OCR models but did not get an ideal result. Most temperature numbers in body thermometer images cannot be correctly recognized. Because of limited time, an ideal OCR model will be hard to be trained.

If only automating temperature submission with human input numbers, the scope will be a bit small. Since this temperature declaration is related to COVID-19, we came up with an idea that both COVID-19 information check and temperature declaration were put together to be a COVID-19 Assistant.

In the implementation phase, I worked on automated COVID-19 information scraping from web pages. It includes tabular data, svg charts and news. The major RPA tools used is TagUI and Selenium.

I proposed ideas I learnt from the Chatbot project in the first semester. It helped us to quickly define the workflow.

As the report is compiled by LaTeX, I also helped the team to finalize the report in a structured format.

B.3.2 What I have learnt

- Crawler tools such as selenium, TagUI and their limits
- Render html element by imgkit in Python
- Understand the limit in Dialogflow with Webhook
- Make automated news summary after data scraping
- Cloud AI with Google Cloud API - translate and OCR
- OCR is quite limited with the real life images

B.3.3 How I can apply the knowledge and skills in other situations

- Find many potential automation projects in daily life and get the idea on how to automate them with what I learnt.
- Use of RPA tools for web crawling. Use of Cloud AI for translation to quickly get the information in other languages. Then I can send to parents
- Use a scheduler to automate some daily process triggered easily
- Some new LaTeX understanding can be applied to other reports

B.4 LI DUO

B.4.1 Personal Contribution

At the brainstorming session for project ideas, I have pain points of daily Covid-19 data checking especially for country data, and everyday temperature uploading, and proposed a potential project which could be used to update NUS DAILY TEMPERATURE DECLARATION.

We came up with an idea that both COVID-19 information check and temperature declaration were put together to be a COVID-19 Assistant.

In the implementation phase, I worked on an automated COVID-19 Temperature upload to NUS temperature declaration system. This time I use Selenium technology to do RPA workflow. This is because NUS is smart to detect webdriver, and TagUI cannot bypass the check, and Selenium is able to do it. I figure out the xpath of each element and use Selenium functions to auto-fill in and upload the temperature.

I also helped the team to finalize the video and shared to internet.

B.4.2 What I have learnt

- Crawler tools such as selenium, TagUI and their limits
- Understand the limit in Dialogflow with Webhook
- Make automation of upload information
- Cloud AI with Google Cloud API - translate

B.4.3 How I can apply the knowledge and skills in other situations

- Find some potential Web automation job in my life to reduce time to do routine work
- Use of RPA tools for web crawling and auto filling. Use of Cloud AI for translation to quickly get the information in other languages.
- Use a scheduler to automate some daily process triggered easily
- Some new Video editing technology

C Project Related Files

Presentation Video: https://youtu.be/w2INw_zed8g

Project Files: <https://github.com/ddgone2far/ISA-IPA-2020-03-11-IS1PT-GRP-High5.git>