DASH

Diagnosis and Symptoms Helper
An intelligent triage and pre-diagnosis system

Team members
Tadhg Kennedy
Meng Chenxi
Wang Hongtao
Yang Yizhou

November 3, 2021

Contents

1 Executive Summary

The COVID-19 pandemic has changed the way we approach doctor's visits and medical institution as a whole. With a highly transmissible disease out in the population having sick people congregate in a location is the exact opposite of what doctors and immunologists advise. While the COVID-19 pandemic is waning the impacts it has had on our approach to medicine can be seen across the industry, tele-medicine services are now common place and efforts to reduce patient congregation are still in place to reduce the chances of a potential spike.

Our diagnosis and symptoms helper (DASH) seeks to address this need and provide additional benefits to the institutions it's deployed in. By leveraging the power of machine learning and chatbots we hope to be able to reduce the need for patients to congregate in waiting rooms and reduce some of the workload from the doctors in their already busy workdays. Our system will collect the patient symptoms via a chatbot, these symptoms will then be used to intelligently generate questions to help narrow down the possible diagnoses, once the system has sufficient information the symptoms will be used to provide a potential diagnosis in a report to the doctor. This report will allow the doctor to decide if the patient requires an in person consultation, tele-medicine session, provide prescription or potentially skip the visit to the GP and proceed straight to a specialist.

This project will provide a minimum viable product (MVP) for the system, focusing on the core functionality in order to deploy, test in a real world scenario, and gather feedback to help improve the product. The product has the potential to grow to encompass many useful functionalities such as prescription recommendation, automated booking systems, and improved diagnosis based on the patients medical history.

2 Business Case

2.1 Problem Statement

With COVID still a major concern, people congregating while waiting to see the doctor increases the risk of transmission to other potentially vulnerable patients. Some patients may even have illnesses which can't be treated with antibiotics or prescription medications, such as cold or flu, in these cases it would be better that the person does not come to the clinic and risk exposure, and instead remains at home to rest and recuperate.

Additionally health care workers are stretched thin and over-worked due to the demands of the pandemic, this can lead to overwhelming workloads, mounting backlogs and eventually burnout. Further reducing the manpower available at a time where it is so desperately needed.

2.2 Proposed Solution

Our proposed solution to tackle these issues is to use a AI chatbot to gather patients symptoms before they go to the clinic in person, intelligently generate questions to further narrow the potential diagnosis before finally generating a report for a doctor to review and make recommendations for how the patient should proceed. The system will focus on acute illnesses which can be pre-diagnosed by unquantifiable symptoms (sore throat, itchiness, rash).

The doctors can then recommend: tele-medicine services for those with illnesses which don't require further in person tests and prescribe medication as necessary, in person consultation for illnesses which require test like blood pressure or samples, or potentially direct referral to specialists for time sensitive or severe illnesses.

Our solution has the following objectives

1. **Reduce unnecessary visits to clinics** Help patients and doctors avoid visiting clinics and potentially exposing themselves to infectious disease.

2. **Assist doctors** Reduce some of the workload from doctors by collecting symptoms in advance and providing possible diagnosis.

3. **Reduce wait times** By reducing the number of people visiting clinics in person, patients who need it can be seen quicker and potentially skip clinic visits to be referred to a specialist saving precious time in cases of critical illness.

2.3 Market Research

AI healthcare is a rapidly growing market, reaching \$6.7 Billion in 2021 [Kwo21] and is forecast to reach a market size of USD 61.59 Billion by 2027 [RD21]. This ever expanding industry along with Singapore's commitment to the Smart Nation projects in the medical sector show that AI is a major focus for Singapore's future.

Based on our research we have found there are several other products which attempt to provide medical diagnosis via chatbots and apps. However, these products are either focused on providing the diagnosis and suggested treatments to the patient or to gather information about the patient before sending them to the companies own private tele-medicine services [Fut17]. We intend to focus on provided value to existing brick and mortar locations and the public sector with our product, we believe this approach will provide the best value for the community and work best in Singapore due to the nationalised healthcare system.

We have also conducted our own market research starting by interviewing a doctor to gather information about their day to day work in a polyclinic here in Singapore and to get their opinions of the usefulness and viability of our idea. Next we surveyed patients globally and doctors both in Singapore and in China, using the information we gained from the interview to decide what questions to ask. The surveys were split into two parts: for patients the first was to gather their general experiences with doctors and approach to illness, second was to get an understand of their exposure to AI and tech in medicine and hear their concerns; for doctors the first was to understand their general day to day work and experience with patients, second was to their experience with AI in medicine, what they want from a product like ours and what concerns they have.

2.3.1 Doctor Interview

We were lucky enough to be able to interview Dr. Li Kim Selby, a two year family medicine resident at the Geylang Polyclinic. She was kind enough to take the time to speak with us and answer our questions, from this we were able to get a general estimate of how many patients with acute cases would be seen in a day along with an understanding of how long doctors had to deal with these patients and the percentage of patients who didn't need prescriptions for their illness. She also explained that AI isn't something she uses in her day-to-day but sh does believe it could help. She did however raise concerns over the model creating a confirmation bias with doctors, we have taken this feedback under advisement and based on the results from our MVP test will investigate potential solutions to combat this. The full set of interview questions is available in the appendix D, we'd like to thank Dr. Selby for her time in helping guide the design of DASH.

2.3.2 Survey Results

Patients Survey Based on our survey with 73 respondents, we were able to get a general feeling for the public's interactions with doctors and opinions about AI in medicine. Our sample was 66% over 45 years of age, with approximately the same split in the age date, most visited the doctor fewer than 5 times a year. 70% of the respondents had not used a tele-medcine service, 53% of respondents would not engage with an AI over a doctor, with 30% being neutral and the remaining 17% being willing. General concerns were primarily accuracy with 75% citing this as their major concern. Based on these responses we feel that we have made the correct decision is designing the product to provide a report to the doctor, as this will address any of the fears raised by the public over an AI making the final decision.

Doctor Survey We received 16 response from doctors in Singapore all within the GP or family medicine discipline, across many years of experience ranging from 3 to 34 years. We gathered a further 22 responses from doctors and medical students in China.

65% of responding doctors had not used AI as part of their work, while 35% had, this shows that AI is still an emerging field in medicine. We can also see from the results that the doctors with the most experience were the least likely to trust AI, while younger doctors were more neutral, this may be due to an unfamiliarity with what AI is for older doctors or past experience with less advanced AI systems. 68% of responding doctors indicated that a chatbot to collect patient symptoms would be helpful in their day to day work. With the 57% of respondents taking 3-5 minutes to collect symptoms per visit, and 20-40 visits per day, our system could save between 1 to 2.5 hours a day for the doctor.

The legal implications of engaging an AI was also raised by some doctors in the comments, as previously stated we've chosen to design the system as an assistant for doctors to make the final diagnosis and in all cases the patient will be seen by a doctor which avoids any potential occurrences of misdiagnosis of critical illnesses. When the system is deployed we will work to ensure the system strictly follows the guidelines set out by the MOH regarding AI in medicine. [oH21]

2.4 Project Team

Full Name	Roles
Tadhg Kennedy	Administrative lead Market Research Intelligent question generation Marketing Video Project report writing
Meng Chenxi	Model development and testing Project report writing
Wang Hongtao	UI design and development Backend Function Project report writing
Yang Yizhou	Technical Lead Chatbot development Backend framework development Project report writing

3 Product Design

3.1 Main Features

Just like the results of our surveys, we can find that main pain points are that patients often have to spend a lot of time before they get a treatment and when AI is involved in treatment, it will trigger the distrust of doctors. Therefore, we mainly propose the following features.

- **A chatbot that collects patient information.**
  Because the cornerstone of diagnosis in the modern healthcare system is trust. From this perspective, we don't let the user notice too much the presence of AI in the diagnosis, so we use a chatbot for patient information capture.

- **An intelligent diagnostic system.**
  Doctors often use symptoms to speculate about illnesses, and the reasoning is often based on medical knowledge and previous cases. So we can reduce it to a process of inductive reasoning, also called knowledge representation and search. In this way, we can give enough possible conditions for the doctor to make a diagnosis, and even some conditions that the doctor has not considered.

- **An app for doctors.**
  We also designed an application for doctors to search for patients information and use AI to advise on the disease and even provide referable treatments.

3.2 Process Flow

Figure 1 shows the workflow of this system. It can be found that the system provides two external interfaces, one is a chat window in the form of Chatbot, which is mainly provided to patients. One is an interface in the form of a desktop application. Through this interface, doctors can query, update, modify, and delete entries in the database.

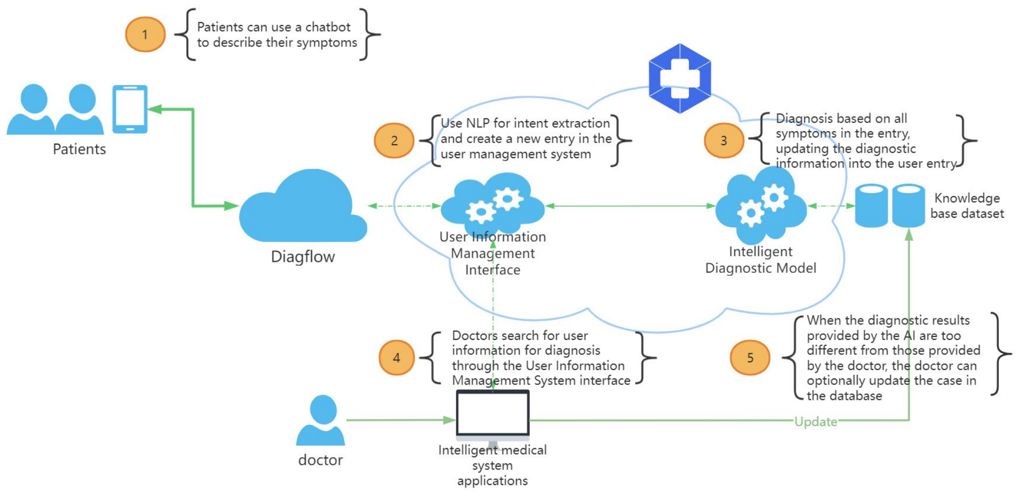

1
Patients can use a chatbot to describe their symptoms
Patients
2
Use NLP for intent extraction and create a new entry in the user management system
3
Diagnosis based on all symptoms in the entry, updating the diagnostic information into the user entry
Diagflow
User Information Management Interface
Intelligent Diagnostic Model
Knowledge base dataset
4
Doctors search for user information for diagnosis through the User Information Management System interface
5
When the diagnostic results provided by the AI are too different from those provided by the doctor, the doctor can optionally update the case in the database
doctor
Intelligent medical system applications
Update

Figure 1: Process Flow

3.3 Systems Highlights

3.3.1 Short-term memory based chatbot

Our intelligent medical system is a short-term memory base system which can remember the symptoms described by the patient and narrows them down to a specific symptom based on the results of the next questioning, for example, if the patient says, ”I have a fever,” we will write down the symptom of fever, but it is uncertain whether the fever is high or low, and we will narrow down the symptom in the next round of Ask, ”Is it a high fever or a low fever”, to narrow down the symptoms.

3.3.2 Natural Response

We use a template-based response generation module to respond to patient questions.The input to this module is a specific symptom and the output is a natural flowing sentence as a response to the patient’s question.

3.3.3 Extensible software framework design

Figure 2 shows our software architecture design, which can be seen in three main modules, user information management module, core module, and function module. This framework is designed to keep the modules separate from each other. Modules and databases are independent of each other, with high cohesion and low coupling and easy extension, making it possible for each member of the team to do some of the work independently.

- **Core module**
  The core module has two main processes. One process acts as the back-end process for Diagflow, storing, filtering and processing the intent returned by Diagflow. The other process is mainly used to provide a visualization application for the doctor to view the patient’s specific situation.

- **User information management module**
  User Information Management Module contains a set of API for operating the database and a processing process. The API is mainly used by other modules to safely operate the database, and the process monitors the data submitted by other modules continuously and calls functions in the function module to process the data that needs to be processed.

- **Functional module**
  Functional module is designed to be extensible, now we have implemented the diagnosis function and subsequently if you want to implement more functions in this framework can also be easily added.

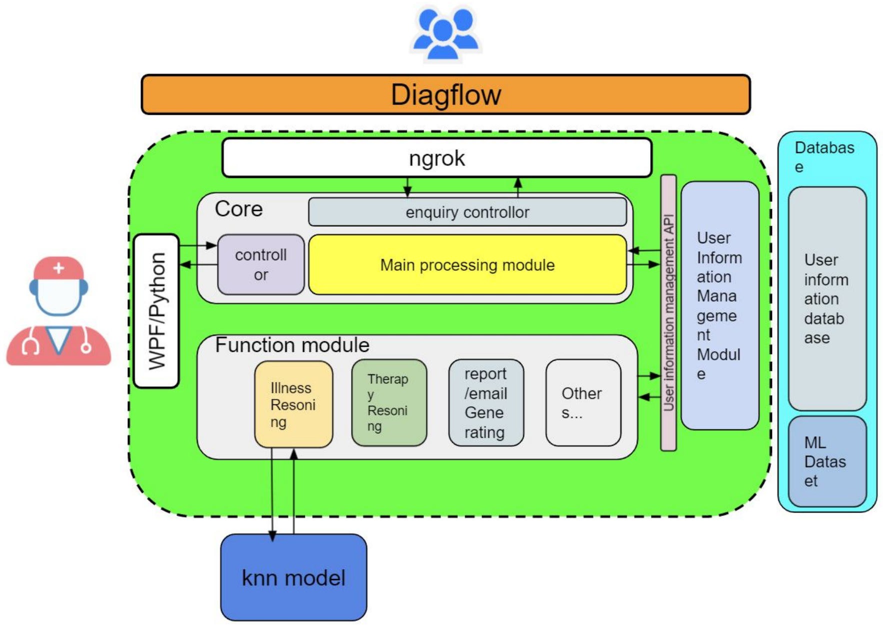


Figure 2: Framework

- **Exterior Design**
  Exterior Design is similar to a plug-and-play design, which means we can design and train the models independently, update the database, and then replace them with the original model and database and then they will work fine.

4 System Modeling

4.1 Data Preparation

We collect the data on Kaggle, and each piece of data is a patient's symptoms and its disease. We do the preprocessing to the dataset and divide it into training set and test set, which will be convenient to our later work. There are 132 symptoms and 41 diseases in this training set, and 4920 patients to train the model. The testing set contains only 42 pieces of data but covers all disease so we won't miss one single situation. The details of our datasets like this: the symptoms which patients have are marked as 1 and other symptoms are marked as 0. The last column records the diseases of each patients. The symptoms are very common and detailed, such as headache, cough and high fever. The diseases range from simple cold to AIDs.

4.2 Model Choosing

We look up some papers and found that most similar projects use KNN(k-nearest neighbors)[ESG16] [XB19] and DT(decision tree)[CCH19] to build the model. Therefore we decide to try both of them and then find the better one.

We first use KNN to build the model and the function 'knn.score' to check its performance. We set the number of neighbors which is 'k' in the algorithm from 20 to 100, and record the computing time. The most suitable k is about 50, which has a high score of 94.9% on the training set and 92.9% on the testing set. At the same time, the computing time is acceptable which is important for several times of training and we believe it's a useful point in real world case.

Then we use DT to build the model and the function 'dt.score' to check its performance. We change the parameter 'max_depth' to find the best depth of DT and find 38 is the most suitable depth, which has a score of 91.8% on the training set and 90.6% on the testing set and the computing time of DT model is much longer than KNN model. Moreover, based on the actual problem of this case, it's difficult for DT to get the next symptom during the talk with patient. At last, we decide to choose KNN model and set k as 50. We train KNN model on the training set and get the final model quickly and then save it for future use.

Now that we have the trained model, the next step is to build other functions to complete the process. Here is how the system works: we first get the symptom that user provides and use model to predict potential illnesses and their probability. Then we pick symptom with highest fluency in those potential illnesses and set it as the next question to be asked. All symptoms that user gives yes answer will be marked as 1. Once the sequence of potential illness come to one single illness or all symptoms have been asked, the chat will come to an end and the system will save the final result.

4.3 Model Testing

We start to test it using specific illness. Here I take common cold as an example.

We first collect all 17 symptoms of common cold in the training data and create a sequence to simulate user's input. First we set 'cough' as 1 while other symptoms as 0, which means the first user input is 'cough'. Second we put this sequence into our model and get a result of potential illnesses which are 'Bronchial Asthma', 'Common Cold' and so on. Moreover, we can get the symptoms to be asked next which is 'chest pain', but it don't match 'common cold' so we skip it and find another symptom which is 'high fever'. Same as the last step, we set 'high fever' as 1 and put the sequence into model to make prediction. Mostly, the number of potential illnesses will reduce step by step and finally reach the predicted result. After we integrate the whole system, we use diagflow to do the same test and at the same time check the outputs in the command window and find the same result.

During the course of the testing we had to tackle many problems. Initially we found that the distance function being used didn't yield the correct results. Through experimentation we were able to find that the "russellrao" setting yielded much better accuracy. We also found that initially the way the questions were being generated wasn't a very good experience for the patient as in some cases the chatbot would end up asking questions for too long. Investigating in the backend we found that even when the diagnosis was certain if there were many symptoms associated with the illness the chatbot would ask all of them even if it didn't help to increase the accuracy. We made updates to the logic such that when the diagnosis is narrowed to only one possible outcome the chatbot stops asking further questions, this made the experience much better for the patient and increased efficiency.

5 System Development & Implementation

5.1 User Interface

Initially, we decide to develop a UI system for doctor to see information of patients including patient user information, the symptoms they provided and the illness that our chatbot had predicted.

The front end was developed using pyqt. This system was divided into three parts: search, view and exit.

Once the doctor enter into this app, it will display search page and required to input patient id or name in order to get the information from the database and then click the "submit" button, it will then show the view page to display the symptoms and illness that our chatbot predicted.

Figure 3 show the search page, which is based on the id or name to search the information of patients.

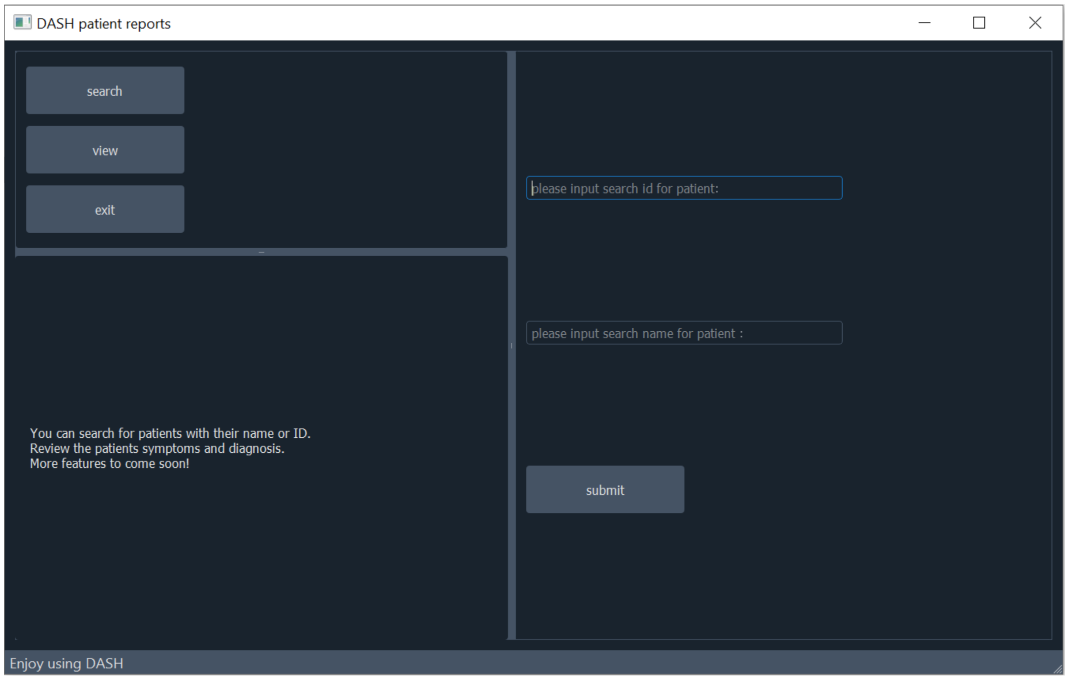
Figure 4 show the view page, which can display the symptoms and illness that our system collected and predicted.

5.2 Chatbot

5.2.1 Dialogflow

Dialogflow is a natural language understanding platform used to design and integrate a conversational user interface into mobile apps, web applications, devices, bots, interactive voice response systems and related uses.

Entities Design. First we divided the different symptoms into parts and symptoms, for example, "eye_pain" is a symptoms in the dataset, so we divided it into "eye" part of our body and "pain" the


DASH patient reports
search
view
exit
please input search id for patient:
please input search name for patient :
submit
Enjoy using DASH

You can search for patients with their name or ID.
Review the patients symptoms and diagnosis.
More features to come soon!

Figure 3: UI Search page

description of this symptom. As analyzed, we created two new entities and divided all the symptoms in the dataset into these two entities, and added some synonyms to make the identification more accurate, like "fever" and "high temperature".

Intents Design. Dialogflow can help us extract intents form a sentence. In this task, the main intention is to identify two intentions: 1. the illness described by the patient, which we named "illness", and 2. the attitude of the patient in answering the question, whether it is positive or negative. For example when the patient says "I have a fever", the specific symptom "fever", will be returned by Dialogflow to the system, and when the patient answers a question generated by the system in the chat window, we can determine if they answered yes or no.

5.2.2 Questioning Strategy

We want the chatbot to automatically respond more naturally, even like a real person communicating with a patient, so the logic of asking is essential. Based on our daily query experience, we divided the query logic into three steps.

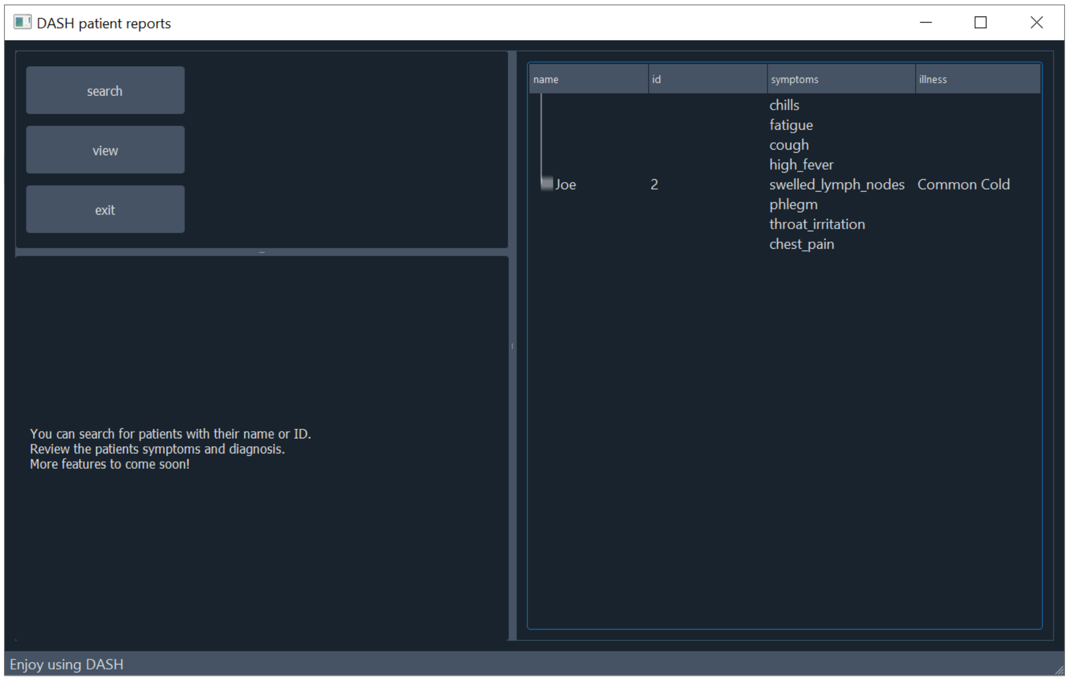
Figure 5, you can find that we summarized our daily experience, firstly, describing the condition and doctor questioning are relatively well understood, and it is worth mentioning that we designed step 2 to narrow down the symptoms based on short-term memory. For example, if in the first step of describing the illness, the patient says his eyes feel uncomfortable, but does not express his specific symptoms. And then we will search all the symptoms related to the eyes in the database and ask the patient with these symptoms, then we achieve the purpose of obtaining specific symptoms. In the third step of doctor questioning, we used the KNN model to get the possible illnesses under the asked symptoms and searched the symptom common to these illnesses as the next question asked by the chatbot.

5.3 Backend Process

5.3.1 User Information Management Process

Before describing this process, first we need to explain the data structure in the database.

Figure 6 shows the data structure of the user information in the database, including information on the 142 symptoms, as well as the predicted conditions and corresponding treatments.



name	id	symptoms	illness
Joe	2	chills fatigue cough high_fever swelled_lymph_nodes phlegm throat_irritation chest_pain	Common Cold

You can search for patients with their name or ID.
Review the patients symptoms and diagnosis.
More features to come soon!

Enjoy using DASH

Figure 4: UI View page

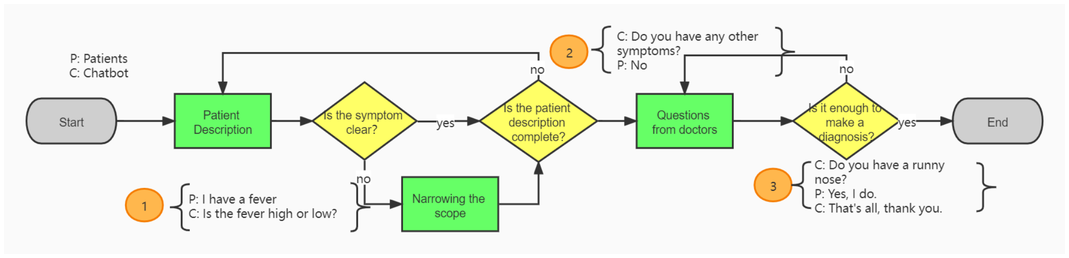

P: Patients
C: Chatbot
Start
Patient Description
Is the symptom clear?
yes
no
Narrowing the scope
Is the patient description complete?
no
Questions from doctors
Is it enough to make a diagnosis?
no
yes
End
1
P: I have a fever
C: Is the fever high or low?
2
C: Do you have any other symptoms?
P: No
3
C: Do you have a runny nose?
P: Yes, I do.
C: That's all, thank you.

Figure 5: Query Logic

The user information management system (UM) provides a series of interfaces for other such as UI, ngrok to make changes to the user information database.

In Figure 5, you can notice that unlike the usual API design, behind the interface for adding, deleting, and checking, we designed a process to handle the submitted entries. The reason for this design is that the entry submission is not simply a modification to the database, but requires a call to 'get_nextquestion' to get the next question asked and also 'get_illness' to get the most likely current illness and record it to the database, both functions require the involvement of the model.

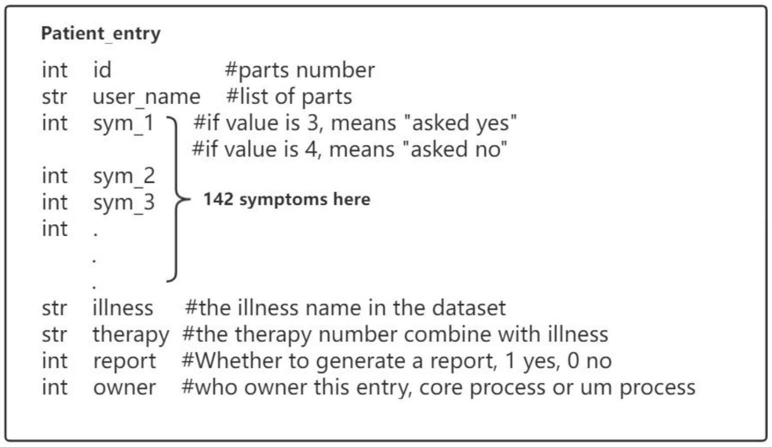
Figure 8 shows the processing of data by the user information management process (**UM** for short). First UM makes a judgment on the owner field, and only processes this entry if it is judged to belong to UM. After that specific functions will be called to populate the entry, for example the illness field, UM will iterate through all the symptoms with the value of 1 and then input them into the model, after getting the illness predicted by the model, we input the illness into the illness field.

5.3.2 Ngrok with chatbot integration

The main questioning strategy has been covered in section 5.2.2, and here we focus on how chatbot is implemented in software.

For a chatbot, the input is a sentence from the user and the output is a reply. In our framework above, Diagflow has helped us extract the intent from the input sentence, so the remaining step is to generate a reply based on these intents.

Figure 9 illustrates the data structure of the symptoms.The symptom class contains information



```
Patient_entry
int   id              #parts number
str   user_name    #list of parts
int   sym_1  ⎫ #if value is 3, means "asked yes"
             ⎪ #if value is 4, means "asked no"
int   sym_2  ⎪
int   sym_3  ⎬ 142 symptoms here
int   .      ⎪
      .      ⎪
      .      ⎭
str   illness    #the illness name in the dataset
str   therapy  #the therapy number combine with illness
int   report   #Whether to generate a report, 1 yes, 0 no
int   owner    #who owner this entry, core process or um process
```

Figure 6: data structure in the database

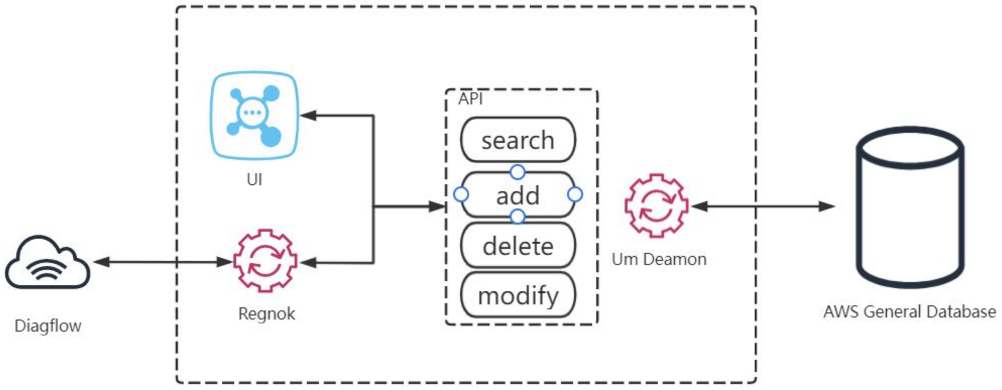


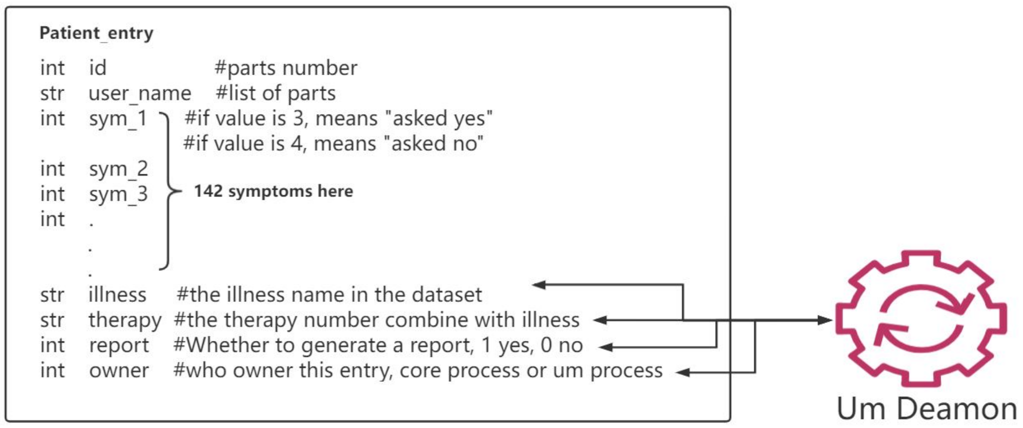
Figure 7: Work flow

such as the number of parts, the name of the part, the description of the symptom, etc. The task of our chatbot is to compose this information into a sentence to express it, and here we use the simplest method – a template to generate the sentence. For example, "eye_pain" is a symptom name in our dataset. First we divide it into parts "eye" and description "pain", there are 1 part and 1 description so this example match the "1 part 1 description" template – "do you have description in part", the reply will be "Do you have pain in eyes?".

Now we just have a simple division of these symptoms, just to meet the needs of MVP. If we want this chatbot to be more intelligent, we can add more attributes to divide these symptoms and add more templates to match these symptoms.

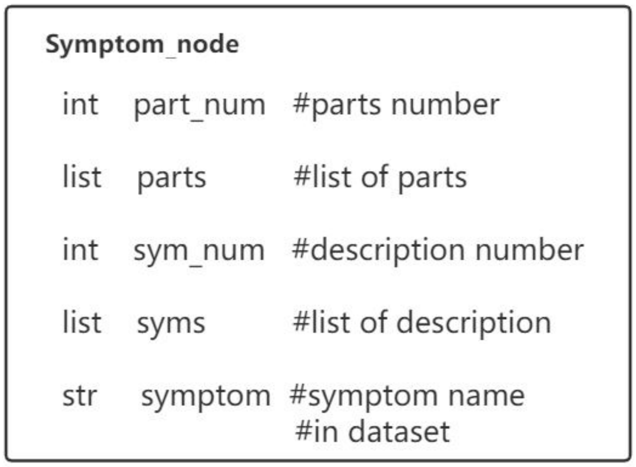
5.4 Conclusion

We've successfully implemented a chatbot which is able to gather a patients symptoms, intelligently generate questions and provide a potential diagnosis to a doctor. We've had to overcome some challenges in order to do it and with more time we hope to further improve our product and approach clinics about conducting a trial run for the service. With feedback from these implementations we can get DASH ready for market and ready to help our front line medical workers.



```
Patient_entry
int   id          #parts number
str   user_name   #list of parts
int   sym_1   #if value is 3, means "asked yes"
              #if value is 4, means "asked no"
int   sym_2
int   sym_3       142 symptoms here
int   .
      .
      .
str   illness   #the illness name in the dataset
str   therapy  #the therapy number combine with illness
int   report   #Whether to generate a report, 1 yes, 0 no
int   owner    #who owner this entry, core process or um process
```

Figure 8: UM deamon



```
Symptom_node
int    part_num   #parts number
list   parts      #list of parts
int    sym_num    #description number
list   syms       #list of description
str    symptom    #symptom name
                  #in dataset
```

Figure 9: symptom node

5.4.1 Future Development

The system provide as part of this project serves as an MVP, in order to get the product in the hand of users early and gather feedback. We see the potential for many further enhancements to the product over time:

- **Prescription recommendation and interaction analysis**
  Recommend a prescription for the patients diagnosis and take into account the potential drug interactions for prescription they are already taking.
- **Including the patients medical history in the model**
  If we have access to the patients medical history we can use this to better diagnose their symptoms.
- **Expanding the possible diseases with a larger dataset**
  Increasing the database to include more possible diagnoses to cover a greater number of illnesses.
- **Taking in doctor's feedback to improve the accuracy of the model**
  Using doctor's feedback to confirm later if the original diagnosis from the system was correct and if it was not using this information to improve the accuracy of the model.

- **Using phone sensors or smart devices to measure information about the patient**
  With additional information coming from the sensor data we could expand our capabilities to take in blood pressure, heart rate or temperature into the diagnosis process. Additionally we could expand the system to continuously monitor the patient in their day to day for any anomalies.

References

[CCH19] Ching-Chin Chern, Yu-Jen Chen, and Bo Hsiao. Decision tree–based classifier in providing telehealth service. *BMC medical informatics and decision making*, 19(1):1–15, 2019.

[ESG16] I Ketut Agung Enriko, Muhammad Suryanegara, and Dadang Gunawan. Heart disease prediction system using k-nearest neighbor algorithm with simplified patient's health parameters. *Journal of Telecommunication, Electronic and Computer Engineering (JTEC)*, 8(12):59–65, 2016.

[Fut17] The Medical Futurist. The top 12 health chatbots. *The Medical Futurist*, 2017.

[Kwo21] Liz Kwo. Contributed: Top 10 use cases for ai in healthcare. *mobihealthnews*, 2021.

[oH21] Ministry of Health. Artificial intelligence in healthcare. *MOH*, 2021.

[RD21] Reports and Data. Artificial intelligence (ai) in healthcare market size worth $61.59 billion by 2027 — cagr of 43.6%. *globenewswire*, 2021.

[XB19] Wenchao Xing and Yilin Bei. Medical health big data classification based on knn classification algorithm. *IEEE Access*, 8:28808–28819, 2019.

A APPENDIX A - Project Proposal

FILING REF:

PROJECT PROPOSAL

Date of proposal: 28/10/2021
Project Title: Medical chatbot
Group ID (As Enrolled in LumiNUS Class Groups): Group Members (Name , Student ID): Yang Yizhou A0231320A Wang Hongtao A0195158U **Tadhg Kennedy** A0231552N Meng Chenxi A0231546J
Sponsor/Client: *(Company Name, Address and Contact Name, Email, if any)* *None*
Background/Aims/Objectives: Initially we come up with creating a medical chatbot because nowadays we are facing the situation that patients are hard to know what diseases that they are facing so that they also cannot register the correct department in the hospital. In order to solve the problem we come up with the idea that creating a medical chatbot to make people easier to know their illness by answering the symptoms questions. Also the COVID-19 pandemic has changed the way we approach doctor's visits and medical institution as a whole. With a highly transmissible disease out in the population having sick people congregate in a location is the exact opposite of what doctors and immunologists advise. While the COVID-19 pandemic is waning the impacts it has had on our approach to medicine can be seen across the industry, tele-medicine services are now common place and efforts to reduce patient congregation are still in place to reduce the chances of a potential spike. Our medical chatbot system seeks to address this need and provide additional benefits to the institutions it's deployed in. By leveraging the power of machine learning and chatbots we hope to be able to reduce the need for patients to congregate in waiting rooms and reduce some of the workload from the doctors in their already busy workdays. Our system will collect the patient symptoms via a chatbot, these symptoms will then be used to intelligently generate questions to help narrow down the possible diagnoses, once the system has sufficient information the symptoms will be used to provide a potential diagnosis in a report to the doctor. This report will allow the doctor to decide if the patient

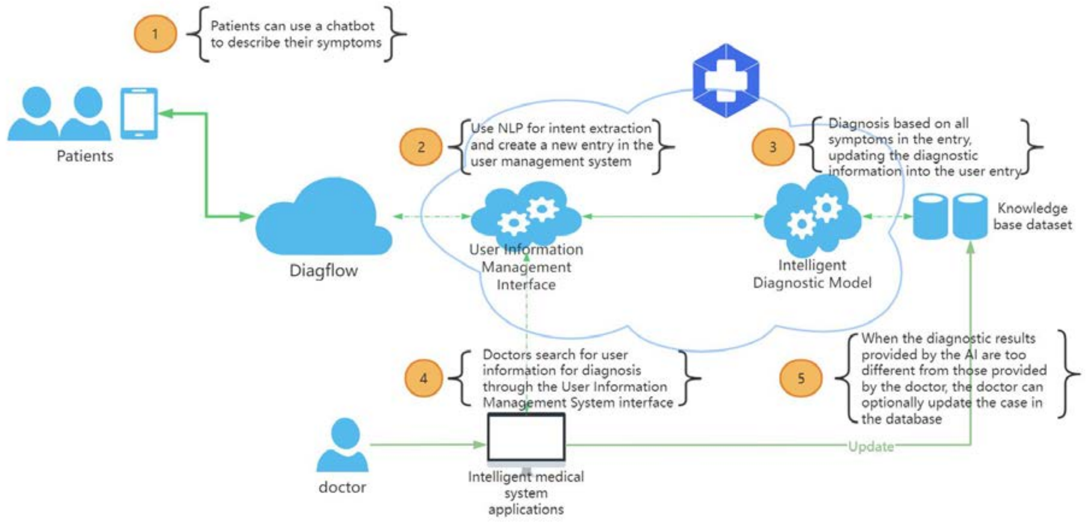
requires an in person consultation, tele-medicine session, provide prescription or potentially skip the visit to the GP and proceed straight to a specialist.

Project Descriptions:

We can divide our medical chatbot into following parts:

1. Train the model using decision tree and KNN. Dataset is based on the symptoms and illness of online dataset.
2. Use chatbot to collect the symptoms from the patients.
3. Use the model to predict the illness based on the symptoms that chatbot collected and confirmed.
4. For doctors we create a chatbot which can display the result of patients information, symptoms and illness.

The flowchart:


1
Patients can use a chatbot to describe their symptoms
Patients
2
Use NLP for intent extraction and create a new entry in the user management system
3
Diagnosis based on all symptoms in the entry, updating the diagnostic information into the user entry
Diagflow
User Information Management Interface
Intelligent Diagnostic Model
Knowledge base dataset
4
Doctors search for user information for diagnosis through the User Information Management System interface
5
When the diagnostic results provided by the AI are too different from those provided by the doctor, the doctor can optionally update the case in the database
doctor
Intelligent medical system applications
Update

B APPENDIX B

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

Module Courses	System Functionalities/Techniques Applied
Machine Reasoning(MR)	**Knowledge Representation** **Supervised Learning Algorithms(KNN and decision tree)** **Rule Based System**
Reasoning Systems(RS)	**Analytic and Synthetic tasks**
Cognitive System(CGS)	**NLP** **Chatbot** **diagflow**

C APPENDIX C - Installation & User Guide

Installation and User Guide

Step 1: Download the DASH code from GitHub.

Step 2: Install python in your computer.

For Window: You can download Python 3.9 from the Microsoft Store
For Linux: You can install using:

```
sudo apt update
sudo apt -y upgrade
sudo apt install python3.9
```

For Mac: Can be downloaded from this link:
https://www.python.org/downloads/macos/

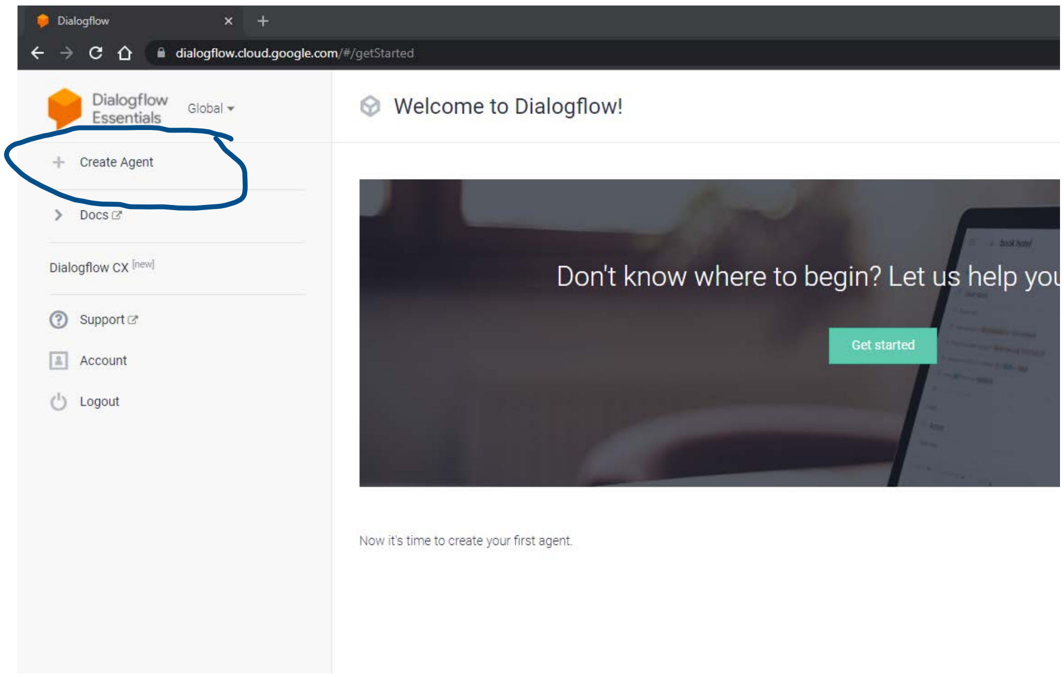
Step 2:

In the command line navigate to the directory you have downloaded to code into.
Run the command:

```
pip install -r requirements.txt
```

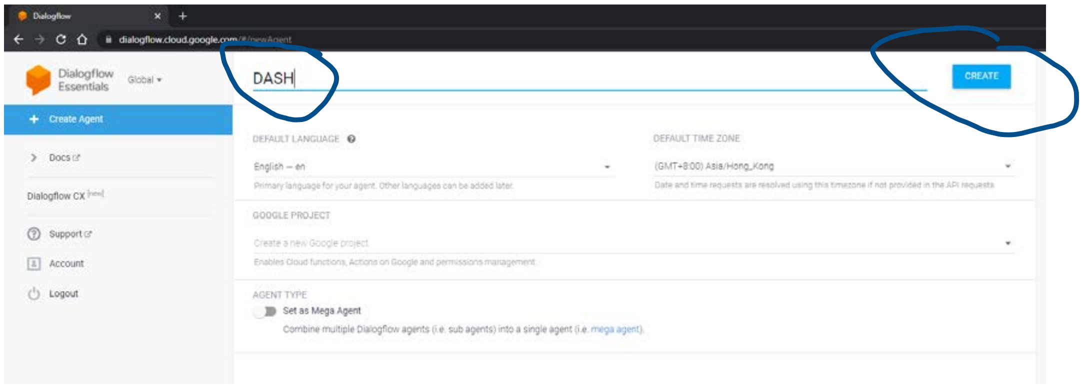
Step 3: login your diagflow dashboard
https://dialogflow.cloud.google.com/

Step 4: Click "create agent"



Now it's time to create your first agent.

Step 5: Name the agent “DASH” and click create



DASH

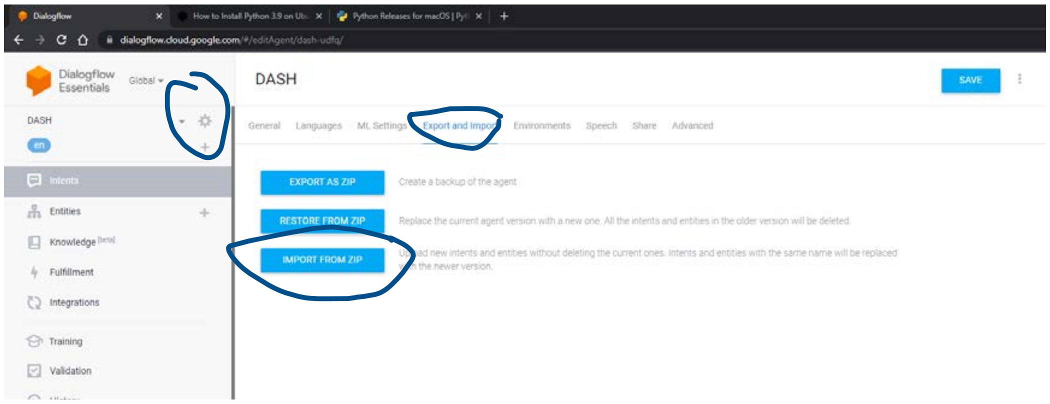
DEFAULT LANGUAGE
English — en
Primary language for your agent. Other languages can be added later.

DEFAULT TIME ZONE
(GMT+8:00) Asia/Hong_Kong
Date and time requests are resolved using this timezone if not provided in the API requests.

GOOGLE PROJECT
Create a new Google project
Enables Cloud functions, Actions on Google and permissions management.

AGENT TYPE
Set as Mega Agent
Combine multiple Dialogflow agents (i.e. sub agents) into a single agent (i.e. mega agent).

Step 6: import SmartAgent.zip

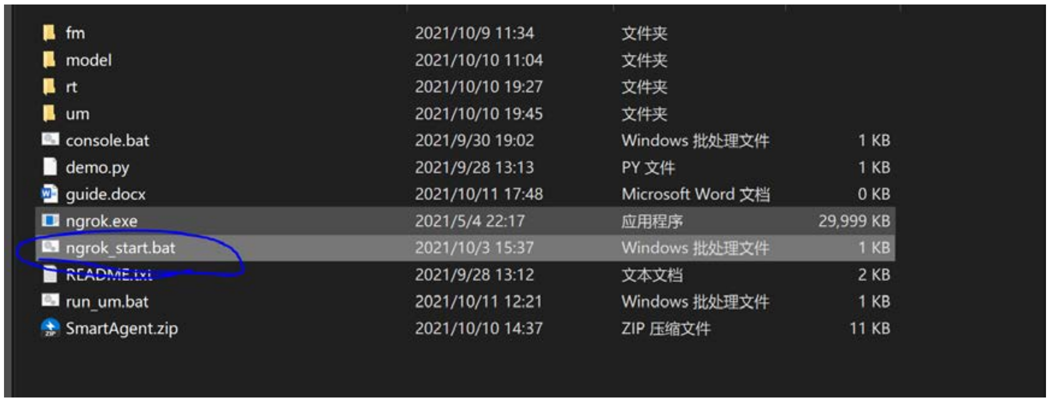


EXPORT AS ZIP — Create a backup of the agent

RESTORE FROM ZIP — Replace the current agent version with a new one. All the intents and entities in the older version will be deleted.

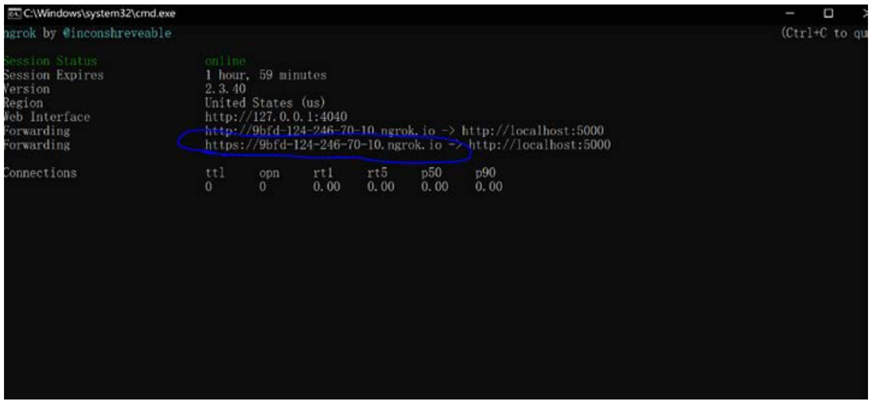
IMPORT FROM ZIP — Upload new intents and entities without deleting the current ones. Intents and entities with the same name will be replaced with the newer version.

Step 7: click ngrok_start.bat

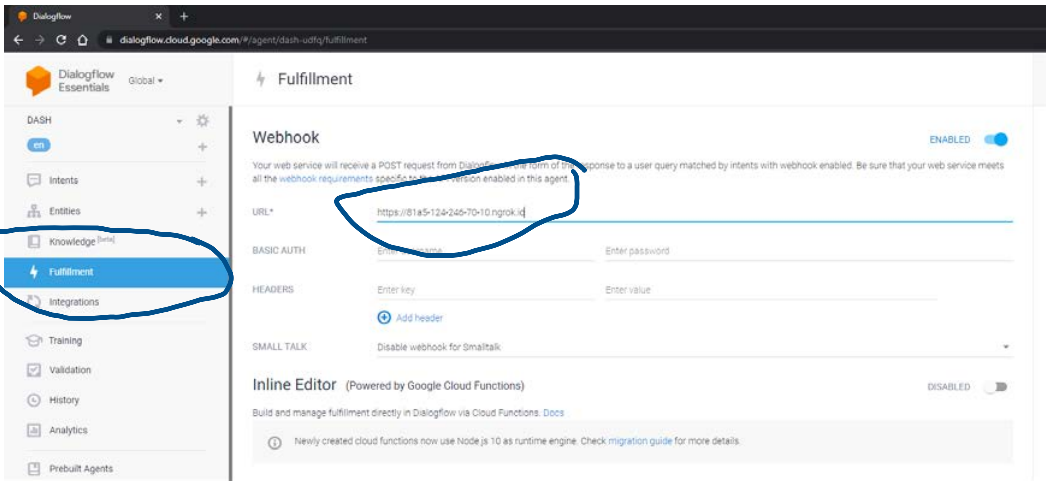


```
fm               2021/10/9 11:34    文件夹
model            2021/10/10 11:04   文件夹
rt               2021/10/10 19:27   文件夹
um               2021/10/10 19:45   文件夹
console.bat      2021/9/30 19:02    Windows 批处理文件        1 KB
demo.py          2021/9/28 13:13    PY 文件                   1 KB
guide.docx       2021/10/11 17:48   Microsoft Word 文档       0 KB
ngrok.exe        2021/5/4 22:17     应用程序              29,999 KB
ngrok_start.bat  2021/10/3 15:37    Windows 批处理文件        1 KB
README.txt       2021/9/28 13:12    文本文档                  2 KB
run_um.bat       2021/10/11 12:21   Windows 批处理文件        1 KB
SmartAgent.zip   2021/10/10 14:37   ZIP 压缩文件             11 KB
```

Step 8: copy to the https address into the “URL” field on the “Fulfillment” tab in Dialogflow



```
ngrok by @inconshreveable                                                          (Ctrl+C to qu

Session Status                online
Session Expires               1 hour, 59 minutes
Version                       2.3.40
Region                        United States (us)
Web Interface                 http://127.0.0.1:4040
Forwarding                    http://9bfd-124-246-70-10.ngrok.io -> http://localhost:5000
Forwarding                    https://9bfd-124-246-70-10.ngrok.io -> http://localhost:5000

Connections                   ttl     opn     rt1     rt5     p50     p90
                              0       0       0.00    0.00    0.00    0.00
```



Webhook

ENABLED

Your web service will receive a POST request from Dialogflow in the form of the response to a user query matched by intents with webhook enabled. Be sure that your web service meets all the webhook requirements specific to the API version enabled in this agent.

URL* https://81a5-124-246-70-10.ngrok.io

BASIC AUTH Enter username Enter password

HEADERS Enter key Enter value

Add header

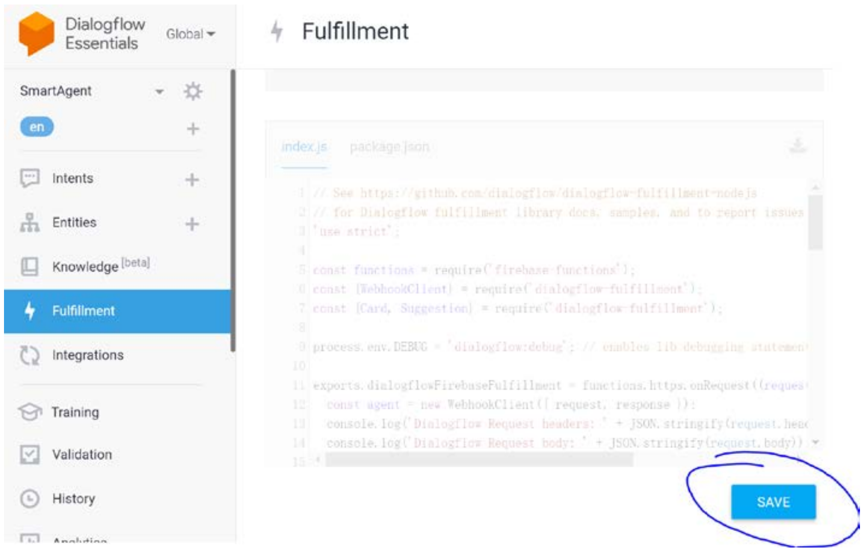
SMALL TALK Disable webhook for Smalltalk

Inline Editor (Powered by Google Cloud Functions)

DISABLED

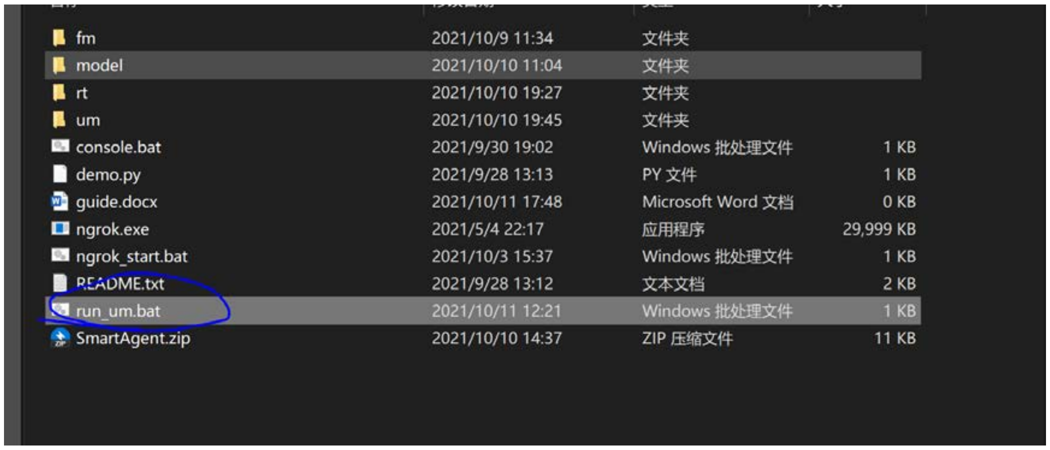
Build and manage fulfillment directly in Dialogflow via Cloud Functions. Docs

Newly created cloud functions now use Node.js 10 as runtime engine. Check migration guide for more details.



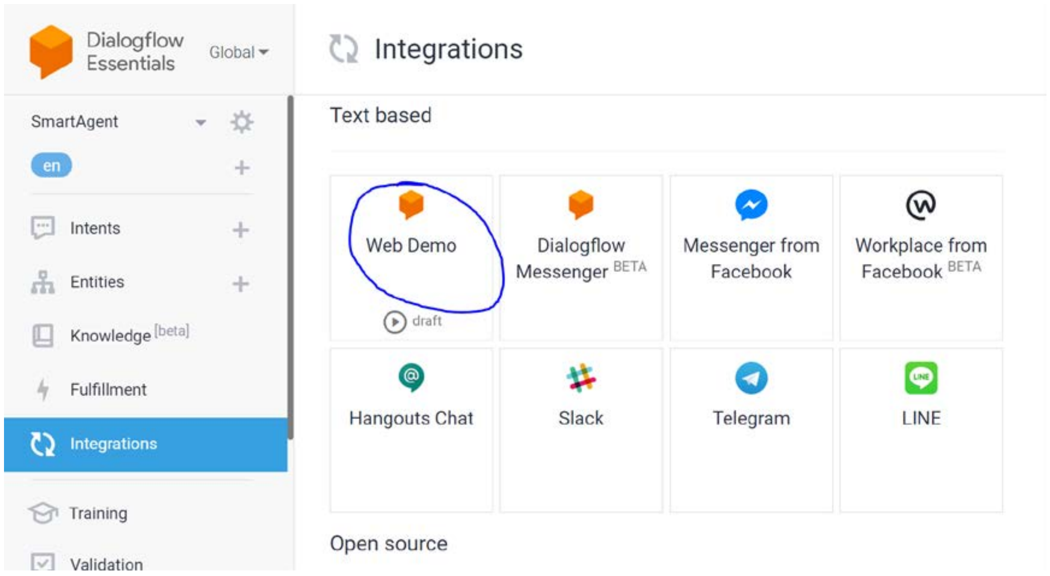
```
// See https://github.com/dialogflow/dialogflow-fulfillment-nodejs
// for Dialogflow fulfillment library docs, samples, and to report issues
'use strict';

const functions = require('firebase-functions');
const {WebhookClient} = require('dialogflow-fulfillment');
const {Card, Suggestion} = require('dialogflow-fulfillment');

process.env.DEBUG = 'dialogflow:debug'; // enables lib debugging statemen

exports.dialogflowFirebaseFulfillment = functions.https.onRequest((reques
  const agent = new WebhookClient({ request, response });
  console.log('Dialogflow Request headers: ' + JSON.stringify(request.head
  console.log('Dialogflow Request body: ' + JSON.stringify(request.body));
```

Step 9: click run_um.bat and go to the "Integrations" Tab in Dialogflow



```
fm                2021/10/9 11:34     文件夹
model             2021/10/10 11:04    文件夹
rt                2021/10/10 19:27    文件夹
um                2021/10/10 19:45    文件夹
console.bat       2021/9/30 19:02     Windows 批处理文件        1 KB
demo.py           2021/9/28 13:13     PY 文件                   1 KB
guide.docx        2021/10/11 17:48    Microsoft Word 文档       0 KB
ngrok.exe         2021/5/4 22:17      应用程序             29,999 KB
ngrok_start.bat   2021/10/3 15:37     Windows 批处理文件        1 KB
README.txt        2021/9/28 13:12     文本文档                  2 KB
run_um.bat        2021/10/11 12:21    Windows 批处理文件        1 KB
SmartAgent.zip    2021/10/10 14:37    ZIP 压缩文件             11 KB
```

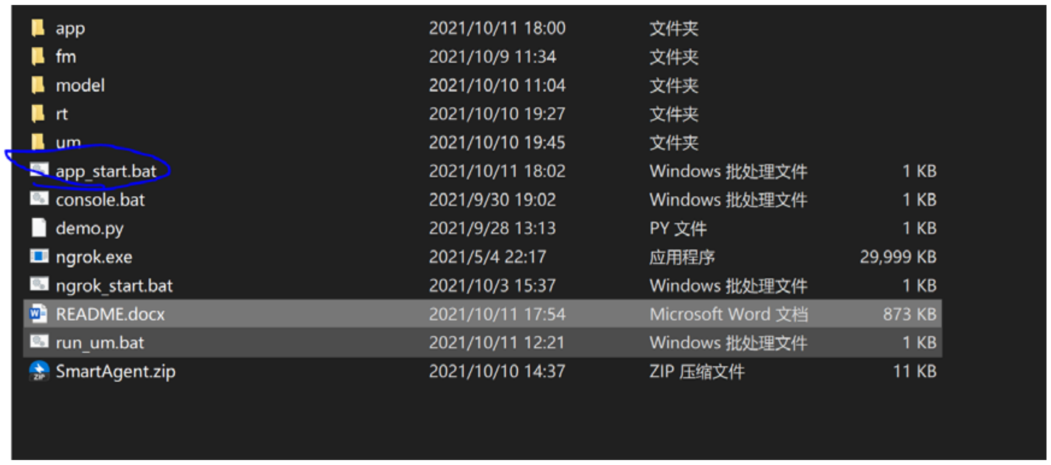
Step 10: Scroll down to Web Demo and try out the chatbot



Note:

- Please provide your symptoms one by one to the chatbot as it asks information.

Step 11： click app_start.bat to start app and view the results of your answers



app	2021/10/11 18:00	文件夹	
fm	2021/10/9 11:34	文件夹	
model	2021/10/10 11:04	文件夹	
rt	2021/10/10 19:27	文件夹	
um	2021/10/10 19:45	文件夹	
app_start.bat	2021/10/11 18:02	Windows 批处理文件	1 KB
console.bat	2021/9/30 19:02	Windows 批处理文件	1 KB
demo.py	2021/9/28 13:13	PY 文件	1 KB
ngrok.exe	2021/5/4 22:17	应用程序	29,999 KB
ngrok_start.bat	2021/10/3 15:37	Windows 批处理文件	1 KB
README.docx	2021/10/11 17:54	Microsoft Word 文档	873 KB
run_um.bat	2021/10/11 12:21	Windows 批处理文件	1 KB
SmartAgent.zip	2021/10/10 14:37	ZIP 压缩文件	11 KB

Note: Patient ID starts from 2 when using the app, at this time.

Referring to the following GITHUB link: https://github.com/wanghongtaonus/IRS_PM_2021_07_05_IS03FT_GRP_17_DASH/blob/main/Miscellaneous/Installation_guide.docx

D APPENDIX D - Doctor Interview

Second year family medicine resident at Geylang Polyclinic.

How many patient's would you see typically in a day at the poly clinic?

It can range from around 20 to low 30s, on average about 25 per day. This is slightly lower than usual because the clinic recently changed the computer operating system and in order to ensure time to properly record everything the number of slots has been reduced to 4 per hour, where previously it was 6.

Is this pretty typical for all clinics or are there more high traffic locations?

Generally, pretty similar across the country.

How many patients would you say come in with ailments that don't require a prescription? Could be treated with bedrest or painkillers?

It's difficult to say, because different rooms are assigned different types of cases. Some will be allocated chronic cases with follow-ups, and some will be assigned acute cases. But in my experience for acute cases this can be as high as 60

How much time do you have with each patient?

This differs between chronic and acute cases, for acute cases about 10 minutes is allocated.

Do you have a set list of questions you ask or is it more by intuition?

It's a mix, we're taught a set of question you should ask but based on how the patient looks and how they answer questions it changes. Every doctor has a list in their head to rule out or in illnesses.

Have you ever used any kind of AI in your work?

Never, it's rarely mentioned in teaching either. It may be a case where schools want to ensure doctors have the full set of skills and don't rely on AI as a crutch/become dependent on it.

More specifically have you ever used or see any diagnosis aids using AI?

Question is moot from above answer.

Do you feel like something which gathered patients symptoms in advance and gave you a potential diagnosis could help you in our day to day work?

I feel like it could be useful in helping reduce the number of patients. I would, however, be concerned that the model's diagnosis could lead to bias in the diagnosis with the doctors. I've have seen instances where going into a consultation with a preconceived diagnosis can impact the questions asked and how the patients answers were interpreted to fit idea.

E APPENDIX E - Individual Report

E.1 Personal Report: Tadhg Kennedy

E.1.1 Personal Contribution

Based on my previous experience working in software and product development, I feel I brought some important insight into the importance for ensuring everyone is on the same page and working towards the same overall goal. I worked to make sure we discussed our ideas early so we could refine the scope to a workable project plan.

My experience with development and understanding the importance of testing practices meant that we were able to identify critical bugs in the system.

I took on the role of administrative leader during the course of the project, using the skills I previously developed to work with the team to set up meetings for in person discussion (when restrictions allowed) and keeping the team up to date on progress. I worked with the team in these meetings to identify the next steps we needed to take and divide out the work between everyone with clear deadlines or checkpoints to make sure everyone was progressing and didn't have any roadblocks which needed help.

E.1.2 Lessons Learned

Through discussion and working with my team I learned a lot about the different models we investigated using and was introduced to functions I hadn't previously used in the course studies so far.

I also learned about the different medical care structure in China, and how it differed from Singapore during our discussion about the project when my team mates didn't know about the GP clinics in Singapore and the importance of a referral letter. This lead to important discussions about initial target market for the product, as the different structures would require different approaches.

Coming from a professional environment with large teams working together it took me a while to understand that it was important to keep scope manageable for a smaller team, especially when the group was split between different class schedules and work for other projects.

I also learned the importance of speaking clearly and not falling into old habits using terminology or acronyms from my previous employment which weren't commonly used elsewhere. I can naturally fall into speaking very quickly and with my accent this can be difficult for those not familiar with it.

E.1.3 Future

I'm sure the knowledge I've gained about models such as decision trees and kNN will be useful to me going forward in my ML career.

Additionally, I think that what I've learned from working cross culturally with my team mates will better prepare me for working with a greater diversity of teams and team members.

E.2 Personal Report: Wang Hongtao

E.2.1 Personal Contribution

During this project, I was involved the whole process of the system design and integration including use case design and development, back end function and UI design and development.

E.2.2 Lessons Learned

I learn a lot from this project, firstly we collect suggestion from domain expert, so we build a survey to do marketing in order to know requirements of patients and doctor, and i think it is essential because pain point is the key for your product and grabbing the customer.

In addition, Our team find dataset online and try different models such as decision tree and knn, we evaluate the performance and choose the better, sometimes need trade-off.

The main part for me is Ui design and development, i learn pyqt and put it into practice.

One of the interesting part in this project is during the discussion i know the differences of medical system between different regions, so that i know when solving problem we need to treat problem case by case.

And Group members are from different regions, so it help a lot for collaborating with diverse team members.

My experience with development and understanding the importance of testing practices meant that we were able to identify critical bugs in the system.

E.2.3 Future

After this intelligent reasoning system practice module, I understood the whole process of system design and development which can help me a lot in the working field, from gathering requirements, use case design and development, group discussion...

I also investigate different models, these practice will help me better understand the concepts which i am sure will useful for me.

E.3 Personal Report: Meng Chenxi

E.3.1 Personal Contribution

At the beginning of IRS module, I gathered this team considering everyone's excellent capability and working experience. As a fresh graduate, I don't have much working experience, which means I always performed a learner and executor working with other team members during the whole project.

I sometimes did minutes of the project meeting. Mostly, I did the system modeling. After investigation of some relevant papers and projects, I build KNN model and decision tree model, which is a very important part in this system and I also did model testing, which contained parameters adjusting and debugging, in order to optimize the system efficiency. What's more, I made a demo of how the dialogflow use, which was performed in our commercial video.

E.3.2 Lessons Learned

I totally involved in this project using my abilities based on previous experience and passion to complete this useful system. As a graduate of statistics, I am quite unfamiliar with python and how to build a machine learning system. During this project, I learned some practical techniques of building machine learning models by actual operations. In addition, I learned how to organize a whole system and how to combine chatbot with model from my teammates while having meeting with them.

E.3.3 Future

I am so glad to see our system work well and solve real world problems, which inspires me to go deeper into artificial intelligent area. I will learn more knowledge and skills about modeling and system integration, which are very helpful in my future work. I also feel very lucky and grateful of having those wonderful teammates who help me a lot and show kind patience. I am sure this project experience will have great influence on myself and my future.

E.4 Personal Report: Yang Yizhou

E.4.1 Personal Contribution

Propose a Chatbot solution: For the whole solution, I propose to extract the intent in the sentences through Dialogflow and design a whole backend framework to handle these intents and reply based on templates to implement a simple bot. I also propose to use KNN model as an inference model, because KNN can predict categories based on distance. The main feature of a disease is the symptoms, and KNN can suitably determine which disease a collection of symptoms belongs to, while KNN is also well suited for expandable and continuously updated databases.
Backend framework & database design and implementation: I designed a low-coupling and high-cohesion backend framework, where modules and modules are separated from each other, making it possible to do the coding together. At the same time, this backend framework is also highly extensible, as long as different functions are added to the functional modules, then it makes the system handle different situations. The framework design of the database is inspired by the registers in the embedded system, although I don't have much is database experience, but I have a lot of experience in embedded development, I know how to design and use the registers. If you think of the database as a hardware device, then each entry is a register in it, and the background program is the firmware of this device, so I designed such a structure.

E.4.2 Lessons Learned

I don't actually have any experience designing and implementing a system in Python. But it's not that hard for me to build a simple system with some knowledge of locks, inter-thread communication, and inter-process communication implemented in Python. In terms of machine learning, this project was very rewarding for me, and we did a lot of research during the initial discussions of the proposal. The main problem was how to find the next problem when we know some of the symptoms. In this process, we investigated KNN model and D-tree, and found that KNN model is better than D-tree in terms of data set and interpretability, so we chose KNN as the inference model. We also took a lot of detours in the later implementation process, but this detour also allowed us to learn a lot of machine learning engineering stuff. For example, the accuracy of our model is high but the next problem we get in chatbot is often unsatisfactory. Finally, we found out that the metric function was wrong, because our data is a binary vector, so using Euclidean distance to calculate the distance would be very poor, but if we use "russellrao" to calculate the distance, the performance is much better. In summary, this project has given me the ability to translate machine learning theory into practical engineering, and has increased my experience in machine learning engineering.

E.4.3 Future

This project has given me engineering experience in NLP. I got familiar with how to use Dialogflow to deploy projects and publish my chatbot. I also learned how to use ngrok to make the local computer a server to handle diagflow requests. During model deployment, I also learned how to use sklearn library to train and deploy KNN models. Also, I had a lot of debug experience in this project. In addition to the technical aspects, I also learned a lot about the market, how to research user needs. How to analyze the market situation, and so on. All in all, with the cooperation between me and my team members, we were able to complete this relatively complex project. And everyone delivered their own module as module leader and learned something new, which I believe is meaningful for every member of the group.