

DOCBOT

Medical ChatBot



April 23, 2023

C+++

K Hlongwane, B Sarak-Jones, Owain Dutfield, Z Toldi

Table of Contents

[Introduction 2](#_Toc133178822)

[Objectives 2](#_Toc133178823)

[Achieved and Unachieved Objectives 4](#_Toc133178824)

[Achieved Objectives 4](#_Toc133178825)

[Unachieved Objectives 4](#_Toc133178826)

[Market Requirement and Limitations 4](#_Toc133178827)

[Market Competition 5](#_Toc133178828)

[Babylon Health 5](#_Toc133178829)

[K Health 5](#_Toc133178830)

[Market Requirements 5](#_Toc133178831)

[User Guide 6](#_Toc133178832)

[Training Data 6](#_Toc133178833)

[Terms and Conditions 7](#_Toc133178834)

[Chatbot Interface 8](#_Toc133178835)

[Chatbot Interaction 9](#_Toc133178836)

[Used Components 9](#_Toc133178837)

[Python 9](#_Toc133178838)

[TensorFlow 9](#_Toc133178839)

[Numpy 10](#_Toc133178840)

[Pickle 11](#_Toc133178841)

[Natural Language Toolkit 11](#_Toc133178842)

[JSON 11](#_Toc133178843)

[TKinter 11](#_Toc133178844)

[Code Documentation 11](#_Toc133178845)

[Training Program 12](#_Toc133178846)

[Chatbot Program 16](#_Toc133178847)

[Training Data 21](#_Toc133178848)

[Conclusion 21](#_Toc133178849)

# Introduction

This report includes a summary design of a medical chatbot describing the components and libraries used for the project. In addition, a user interface guide for the medical chatbot is also available. The medical chatbot was constructed using Python language, using the libraries available in Python such as Tkinter and Tensorflow. Moreover, the report provides information about the competition that is available on the market.

In the report, a user guide for a technical and a non-technical person is available. The user guides are provided to allow users to be able to navigate within the program easily and freely. Furthermore, the documentation will give a technical user a base foundation and some knowledge of the program. The objective section explains how we envisioned the project before development.

# Objectives

The objective set out for the project is to develop an AI chatbot for the company Brainvire Ltd. This project is set up to help the overwhelmed NHS. The chatbot should be able to quickly assist patients/users with infections and diseases they may have caused by the symptoms/information they have provided to the chatbot. The quick responses from the chatbot should allow users to be able to self-diagnose from the comfort of their homes.

In addition, the program must be scalable allowing users to access the chatbot on different devices. The abbreviation KISS (Keep it simple stupid) must be followed allowing people with no technical background to easily navigate the program without needing assistance.

Due to LESPI and legal requirements terms and conditions policy must be set out before the user can use the application. This is to protect the company from any legal situations. Due to GDPR, the application should not store any personal information about the user such as name, date of birth and home address etc. Research must be done for quality assurance to provide a correct diagnosis for the user. In addition, the user should be prompted to contact a doctor if the application diagnoses them with a serious infection or disease.

Communication between the team members must be done using Microsoft Teams and GitHub. Evidence of communication is provided in a separate document (Team Portfolio.) The project must play on the strengths and weaknesses of the team members to improve coding skills and gain new knowledge on artificial intelligence. The code for the program should be made available on GitHub to allow the addition of code from different team members.

To complete these objectives a business plan was set and followed. The application must go through thorough testing, this is to avoid any glitches in the program before it is released to the public.

# Achieved and Unachieved Objectives

## Achieved Objectives

All basic objectives were achieved in the chatbot application. The application was designed in Python as agreed in our business plan at the start of the project. In addition, the application can question a user with questions about the symptoms they present to gain information to diagnose them with a disease. A large data set using a JSON file was used to train the AI chatbot with the different infections. The information on the diseases was taken from a reliable resource (NHS website).

To avoid any legal actions the application does not store any information about the user. Following GDPR rules the application does not ask the user any unnecessary questions only asking direct questions concerning the symptoms the user has. The terms and conditions text box are displayed before the user can use the application to allow the user to understand that the company does not hold any responsibilities if the chatbot misdiagnosis them.

The graphical user interface is very simple and easy to use. Allowing people with no technical knowledge to be able to use the application easily. All communication was done through Microsoft Teams and the code is available on GitHub. Each team member gained new knowledge of the Python Language and Artificial Intelligence.

## Unachieved Objectives

All basic objectives were met however additional features that the company requested could not be met due to the complexity of the program and the allowed time for the project.

Our main objective was to make the chatbot scalable to allow users to access this from different types of devices. We were unable to meet this objective as the application is only available on laptops and PC devices.

The company had decided to provide the service for different languages such as Welsh however the application is only able to provide its service to English speakers. The application is also not accessible to all the demographics such as users with disabilities. The application only allows text input and not any vocal input making it unavailable to people with disabilities.

# Market Requirement and Limitations

In this section, we will discuss the different types of medical chatbots that are available on the market in the United Kingdom. In addition, we will discuss the requirements needed before releasing a medical chatbot to the public.

The market for a medical chatbot is well populated now due to COVID-19 which prevented patients to go to the hospital in person. The more it gets populated the better it is for everyone as this will help decrease the amount of pressure on the NHS staff and company. Chatbots provide a convenient and accessible way for patients to receive medical advice. Each chatbot in the market provides similar services and the end goal of the chatbot is to help users.

The majority of the surgeries in Wales and the NHS website provide a service that questions a user about the different symptoms they have. It will narrow down the answers a user gives to a specific disease or infection. A well-known service that provides this type of technology is HealthWeb Solutions.

## Market Competition

### Babylon Health

This company was created in the United Kingdom offering digital health services to the public. The company provides an AI-powered chatbot which available 24/7 helping patients with a range of health concerns including mental health, sexual health and general well-being. The company is currently in partnership with the NHS organisation. However, this service is only available in some parts of the United Kingdom.

### K Health

K Health is a US-based company that has expanded to the United Kingdom. This chatbot provides personalised medical advice for its users. The personalised medical chatbot advises patients based on their medical history. K Health also offers telemedicine consultations with healthcare professionals allowing users to be able to use the service through a phone call.

## Market Requirements

Before releasing a medical chatbot in the United Kingdom to the public, it is important to meet the guidelines and legal actions to ensure regulatory compliance and protect patient safety. These considerations include:

Data Protection Regulations: Medical chatbots collect sensitive patient data, so it is critical to comply with data protection regulations such as the General Data Protection Regulation (GDPR) and the Data Protection Act 2018. This includes obtaining patient consent and ensuring the chatbot's technology is secure enough to safeguard patient information.

Compliance with Healthcare Regulations: As medical devices, chatbots must comply with regulations set by the Medicines and Healthcare Products Regulatory Agency (MHRA) and the Care Quality Commission (CQC). Depending on the chatbot's level of risk, regulatory clearance or certification may be necessary.

Ethical and Professional Guidelines: To maintain high standards, medical chatbots must follow ethical and professional guidelines from regulatory bodies such as the General Medical Council (GMC) and the Nursing and Midwifery Council (NMC). This involves ensuring that the chatbot offers evidence-based medical advice, promotes safe practices, and doesn't suggest dangerous or ineffective treatments.

Patient Safety: Patient safety should be a top priority when designing and deploying a medical chatbot. This means implementing appropriate safeguards to prevent the chatbot from providing harmful advice, referring patients to healthcare professionals when necessary, and providing clear information about the chatbot's limitations.

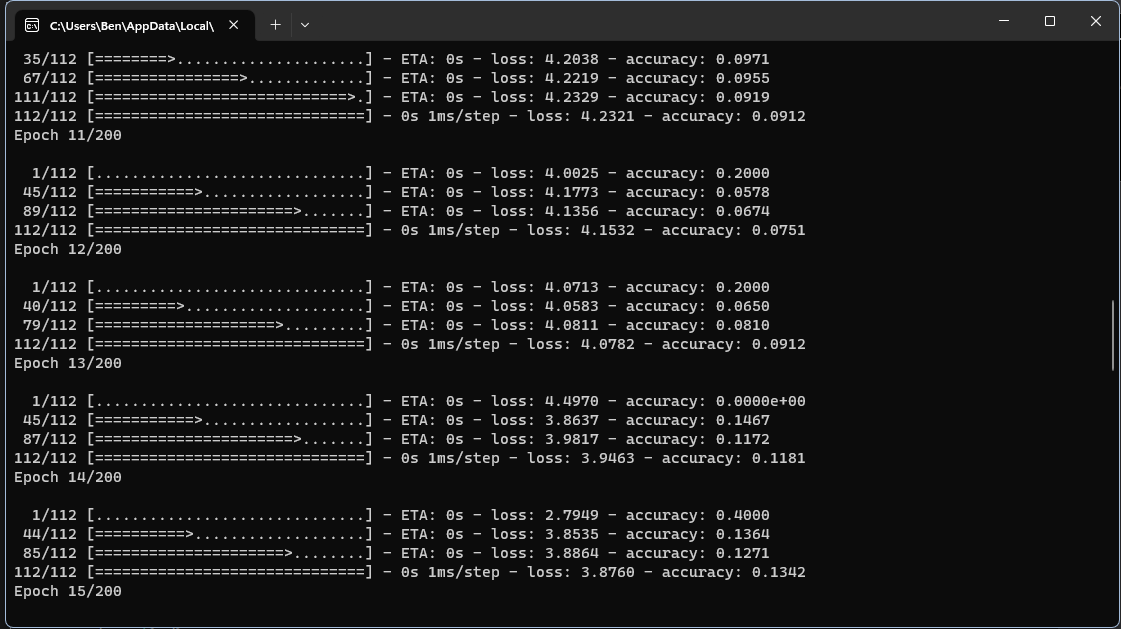
Accessibility: Medical chatbots should be available to all patients, regardless of age, ethnicity, or disability status. Therefore, it is important to consider designing the chatbot with a diverse range of users in mind and ensuring that it is available in multiple languages and formats.

# User Guide

In this section a well detailed user guide will be provided to assist first time users of the chatbot. Due to following the KISS methodology the application is very simple and straight forward to use.

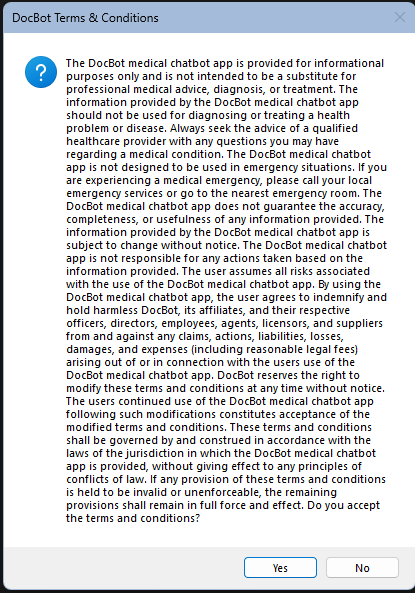
### Training Data

Before running the program at the beginning the chatbot needs to run its training data and after completion and the user will have the ability to see a graphical user interface designed using TensorFlow.



### Terms and Conditions

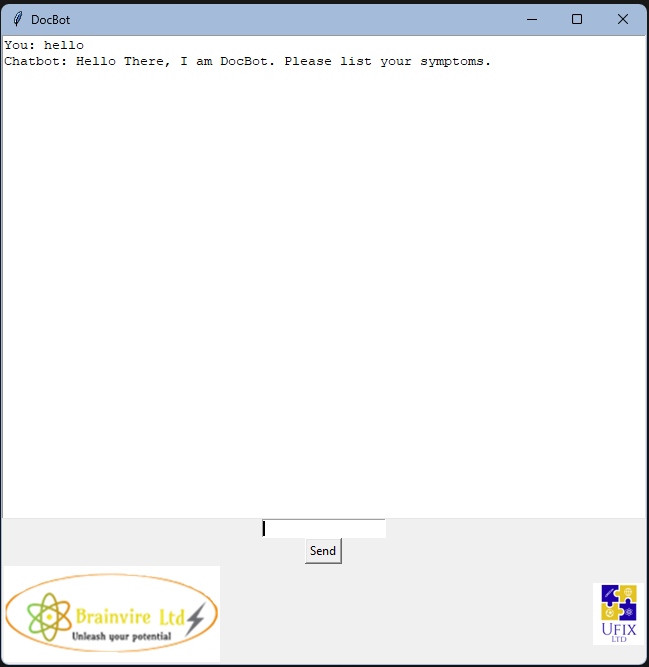
The first sight of the chatbot graphic user interface. The user will be allowed to read the terms and conditions of the chatbot before using it. This to notify the user of what information the program will provide and how it is not a substitute medical professional advice. The user will be able to agree or disagree with the terms of conditions.



### Chatbot Interface

If the user agrees the terms and conditions seen above the user will be prompted to begin the conversation with the chatbot by typing in usual casual introductions such as hello, hi etc. The introduction information should be typed in the textbox just above the send button. When the user has typed this, they should click send to allow the chatbot to receive the information.

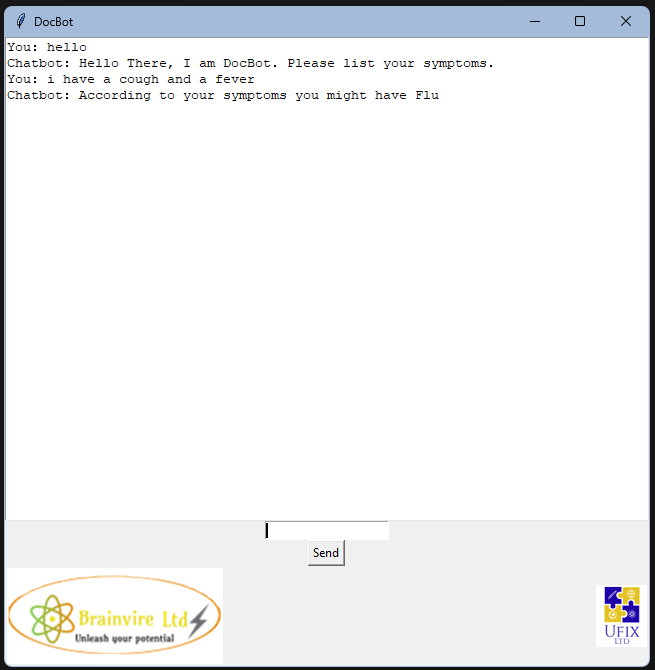
The chatbot will reply to this and ask the user what symptoms they have present. The user should type the information into the textbox and click send after typing.



### Chatbot Interaction

After the user has given the chatbot the information about their symptoms the chatbot will quickly run this information through its data set and come with a conclusion of the potential infection or disease the user may have.

The complete conversation between a user and the chatbot can be seen in the screenshot below.



# Used Components

This section will outline the programming tools that were used to create this project. The tools that will be listed were used to either directly help with the project or helped with the management of the project.

## Python

The Python programming language was the most important part of our project as it was the base programming language that we used to build our entire program. It was important that we chose Python as it had the most libraries relevant to our project available.

## TensorFlow

This tool is a Python library that is a major component in the project. It handles the machine learning part of the program as it provides many useful and relatively easy to use features, which helps the creation of AI based programs develop faster as a lot of base systems are already in place.

## Numpy

This library allows for the complex manipulation of matrices and arrays that base Python doesn’t allow. The reason this library is needed for the chat bot is because it is much more efficient then the built in Python lists, which is crucial as training and getting answers can be quite laborious.

## Pickle

This is a built-in library that is present in Python and its main job is to Serialise data, which means saving the data that is in memory into the solid state drive so it can be De-serialised which is the read the data back into the memory later. This module is used to store the pre-trained data so it then can be loaded into the chat bot.

## Natural Language Toolkit

This library is used to help read in human language such as English and convert it to something that the computer can understand. This library is used to convert prompts that are inputted by the user into something simple which can then be used to determine what kind of answer the chatbot should give.

## JSON

This is the file format that the training data will be held in, which includes the diseases with a set answers and the responses. The reason for using JSON is that it is a very widely used file format used for data storage, which means that there are all sorts of tools and support available when using it.

## TKinter

This is a built-in library in Python that allows for the creation of GUI applications. This library is used for making the interaction between the user smoother by allowing them to type messages into an easy-to-use window.

# Code Documentation

This section will showcase the code and explain what the code does. There are two programs that will be documented here, which is the Training program and the Chatbot program. This section will additionally briefly go over the training data that is used for the chatbot.

The training program is responsible for training the chatbot based on the input JSON data containing all the possible diseases. This program will generate the trained data when it is done so that it can be used by the chatbot program.

The Chatbot program is the main program that the user will be interacting with. This is where messages can be sent to the chatbot so it can then give a response based on the data that it was trained on.

## Training Program

Text

Description automatically generated

This is section of the code involves importing all the libraries that will be used when training the chatbot. It includes most of the libraries that were previously mentioned and some that are haven’t such as random and ctypes. The random library is just a basic random generator and the ctypes library provides a datatype that is needed for the training.

Text

Description automatically generated

This next section declares some global variables that will be used in the program. It initialises the “lemmatizer” variable which will be used to lemmatize the words so that the program can train the chatbot with. It also initialises the JSON file that will contain all the training data and some arrays that will be used later. The words array is used to store all of the words that can be used in a prompt meaning this array represents the model’s vocabulary, classes is the array containing all the unique disease names and other prompts such as greetings and the documents array contains all the diseases along with their corresponding words.

Text

Description automatically generated

This part of the program loops through the entire JSON file to find all the intents which in this case are the diseases then it tokenizes the patterns which are the words or set of words that are typed in by the users that corresponds to that disease then it puts them into a list. Then after the words have been put in the list, the intent tag(which is the disease name) gets added into the documents array along with the list of words that correspond to it. If the disease is not int the classes array already, it will add it in.

A screenshot of a computer

Description automatically generated with medium confidence

This section is mainly used to filter out any unnecessary words that have no use for the AI, which are punctuation marks and common words and then store the words that are important into a pickle file which will then later be used in the training of the AI.

The first line breaks down all the useful words into their root from which is their simplest form so that it makes it easier to understand while also removing those unnecessary words and punctuation marks. Then the unique words will be then saved into the words array along with the classes array before they are saved to a pickle file.

Text

Description automatically generated

First the training array is initialised that will be used in the storage of the training data for machine learning. Then another array gets initialised with 0, the number of zeros initialised is the number of unique intents that are in the classes array. Then a loop is initiated that goes trough each document that is saved (which is the intents with its associated words). After creating the bag array which will hold the bag of words for the document, the words get extracted from the document into the word\_patterns array and then the words will get simplified even more and made lower case. Then a loop is initiated where it loops trough the entire words array of the program then it will add a one if the word is present in the current document or a 0 of it isn’t. Then the program creates a list using the array of zeros and then sets the 0 to a 1 that corresponds to the same index as the current document. Then that list and the bag-of-words array gets added to the training data. The reason why these zeros and ones are created is because it helps the machine learning process easily learn the relation between the words and the class labels.

Text

Description automatically generated

In this part of the code, the first line shuffles the training data so that the model can recognize the pattern without just memorizing the patterns’ order. Then the training data gets converted into a more efficient numpy array. Then two lists are created, the train\_x list will contain all the bag-of-words representations of the documents’ words and train\_y will take the one-hot encoded representation of the intents.

Text

Description automatically generated

This is where the sequential model is created which is a stack of layers that will be used to create the neural network. The first layer is added on the second line, which is the input layer, this layer is responsible for getting the input and then passing it to the next layer. Then a dropout layer is added which will randomly dropout a fraction of the data, which is used to force the model to learn a stronger representation of data. Then the next layer is added which is the hidden layer which is responsible for converting the input data into something that will help the model make accurate predictions with. Then another dropout layer is added and finally the output layer is added which will produce the final prediction based on the input.

Text

Description automatically generated

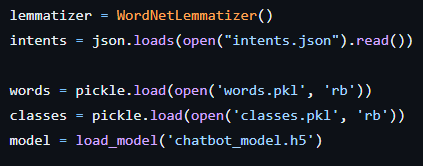
The first line initialises the optimizer for the model, which is the Stochastic gradient descent (SGD) optimizer which will help it minimize loss. Then the model is compiled with the “categorical\_crossentropy” loss function with the SGD optimiser and the accuracymetric. Next line is the when the training of the model starts. First the two arrays that were created to store the training data are given in, then the epochs are specified which is the number of times the model will go over the training data set. Then the batch size is given, which is how many examples the model will go through before its parameters based on the loss generated by that current batch. Then the verbose level is set, which defines if the training progress should be shown in the console or not. After the model is done training it will save it to a .h5 file that can later be used by the chatbot program.

## Chatbot Program

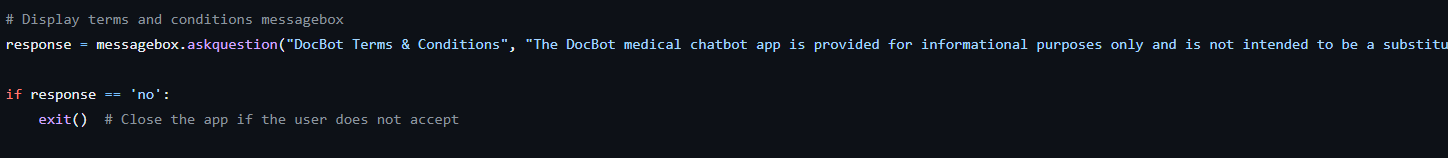
Text

Description automatically generated

This section includes some libraries again, some that were used in the previous training program and some new ones such as tkinter.



This snippet of code initialises some variables that will be used later. The first one being the lemmatizer, which will be used on the user input. The intents JSON to get all the responses for the chatbot which include things like greetings and disease descriptions. Then it initialises three other variables, one being the words variable which will have the words loaded as its vocabulary, then the second one being the classes, which are the unique answers that the chatbot can give and then finally the pre-trained model into the model variable, which will do most of the answering.



This section will ask the user to agree with the terms and conditions that we have for this chatbot, if the user disagrees, the program will exit.

Text

Description automatically generated

This function’s main job is to clean up the sentences that it gets sent in. This cleaning process includes reducing words to their simplest form and removing any redundant or non-useful words from the sentence.

A screenshot of a computer

Description automatically generated with medium confidence

This function’s purpose is to take in a sentence and then return a bag-of-words representation of the sentence. First it uses the “clean\_up\_sentence” function from earlier to tokenize and lemmatize the sentence. Then it will create and initialise an array called bag with a lot of zeros based on the size of the global words array. Then the function does a loop, which essentially just sets zeros to ones in the bag if the word in the sentence is present in the global words array. Then it will return the finished bag as a numpy array.

Text

Description automatically generated

This function’s job is to predict which prompt is the sentence most likely related to, a prompt being a disease or a greeting. First a bag-of-words representation of the sentence is created using the previously mentioned “bag\_of\_words” function. Then that bag-of-words array gets fed into the model using its “predict” function, which will try to find the closest intent that relates to the sentence and will store the prediction results in the “res” variable. After the prediction was made a variable is set which is the “ERROR\_THRESHOLD” variable which means that any result that is bellow this threshold value, will be removed as it will be considered as noise data. Then another list is initialised called the “results” array, which will hold a pair of values, the first value being the index of the intent(which Is one of the responses that the model can have) and the second one is the probability of it being the correct intent. When the results array has been initialised, it gets sorted into descending order, meaning that the most likely responses will be at the front of the list. Then the function creates a list of dictionaries in which there is an intent and its probability, the intent is obtained by the index that was stored earlier in the pair, then the string representation of the probability is stored in the dictionary. After the dictionary is done it is returned by the function.

Text

Description automatically generated

This function’s main purpose is to get the correct response based on the users input. First the function takes in an “intents\_list” which is the list generated by the previous “predict\_class” function. And then it takes In the second parameter which is the “intents\_json” which is the JSON file containing all of the intents with their responses. First it extracts the intent’s name into the tag variable, then it will get all the intents form the JSON file and put them in a variable. Then a loop is created that will search for the extracted tag in the JSON file, if a match is found, then it will chose from one of the responses randomly, then return that response.

Text

Description automatically generated

This part of the code initialises the Graphical User Interface of the program. It will first create the root, which is the main window of the program, then it will set its title to “DocBot”. It will then create a textbox into a variable called “conversation” where the responses will be shown to the user, and then it will create an entry textbox into a variable called “message\_entry” where the user can input their questions into, it will store this input into the “message” variable. This section also creates the “message” variable which will hold a Tkinter string which will be used to store the user’s input that was typed into “message\_entry”.

A screenshot of a computer

Description automatically generated with medium confidence

This function will be invoked when the user will press the send button. The function will first get the message that the user typed and store it in a variable, then it will pass that variable into the “predict\_class” function that was mentioned earlier. After that it will use the get response function to get the appropriate response. When the response was obtained, it will insert the messages into the textbox with indicators that will help distinguish who sent the message.

Text

Description automatically generated

First this part of the code loads in the company logos, then it will create the send button, that has the function “send\_message” assigned to it as a command, then the images get placed into the window. After everything has been done, the main loop is initiated where the program starts running allowing the user to start using it. The last part with the “\_\_name\_\_” is just a debug leftover that isn’t executed.

## Training Data

Text

Description automatically generated

This is the structure of the JSON file that is used for the training of the AI, note that this is only a part of the entire document and that there are many more of these sections. The first major section is the intents section, which defines all other proceeding section as an intent. All intents have a value and two lists a part of them, the first value is the “tag”, which is essentially the name of the intent which helps it identify it. Then there is the “patterns” list, which holds the words that are associated with the intent, these are the words that the AI model will look for when it is trying to predict if this intent is the correct one. Then finally there is the “responses” list, which holds the different messages that the chatbot can send back to the user.

# Conclusion

In conclusion, the report summarises the design of the medical chatbot designed by the C+++ team. The report includes information about the components and libraries used for the application. A well-detailed summary of the libraries used and Python language syntax explaining the base foundation of the program allowing a technical user to have the ability to understand the structure of the program is provided in the report. Furthermore, a user interface guide and information on the competition the product may face if released for public usage. In addition, the objective set up by the company is explained and how the objectives set out have been met. Overall, the report provides insight into the development and potential of medical chatbots to assist the NHS by providing accessible healthcare advice to patients.

Near the beginning of the project, we didn’t have much experience in Python besides the very basics. However, this project has pushed us to learn and improve our skills in the language by presenting us with complex problems that helped us gain experience. The project also helped improve our teamworking skills as well since there were times where some of us had struggled with a particular problem but always had the team to provide help with most issues. The project also helped us improve our organisation skills as it made us plan out how we will do each of our parts, making us decide when to study a specific part of the language and then to use that gained knowledge in the code. Our communication skills have also improved since it made us keep track of each other’s progress by messaging in the group chat. This better communication also allowed us to make quicker progress in the project as we could more efficiently plan out the rest of the project, which as a result made our planning skills better. Most importantly, this project helped us gain a greater insight on how AI works and what goes into making one work, which in turn will help us make an even better AI in the future.