

# WhatBot – A Chatbot System as Online Support Tutor

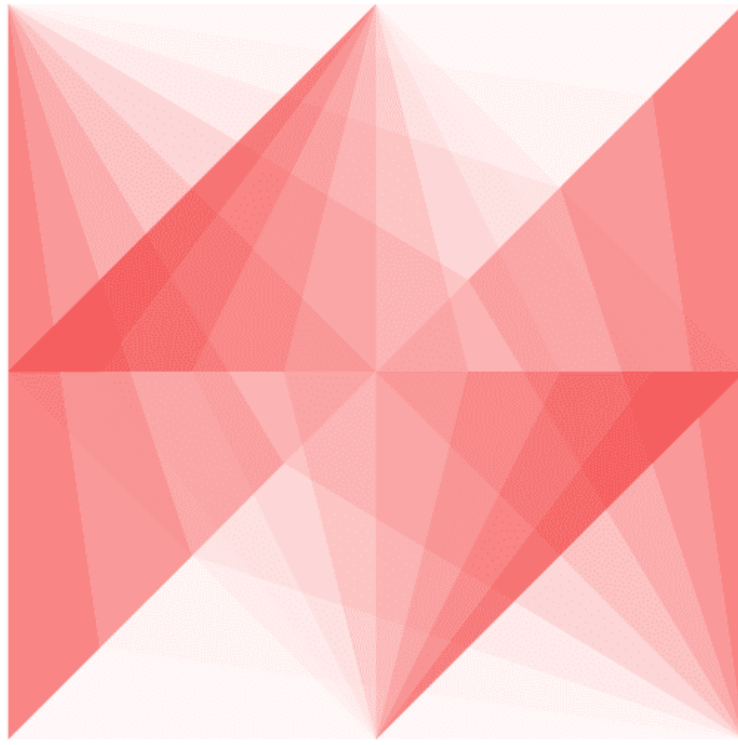
COMP9900 Final Report by z5002318, z5129269, z5137895, z5175539

28 April 2019 (last updated: 30 April 2019)

## Contents

<b>1</b>	<b>Quick Overview</b>	<b>4</b>
<b>2</b>	<b>Introduction</b>	<b>4</b>
2.1	Background . . . . .	4
2.1.1	History . . . . .	4
2.1.2	Existing Systems . . . . .	5
2.2	Aim . . . . .	5
<b>3</b>	<b>System and Software</b>	<b>7</b>
3.1	Software Architecture and Design Overview . . . . .	7
3.2	Core Components and Features . . . . .	8
3.2.1	Query Module . . . . .	8
3.2.2	Utility Module . . . . .	9
3.2.3	Management Module . . . . .	12
3.2.4	Search Module . . . . .	20
3.2.5	Database and File Storage . . . . .	20
3.2.6	Data Extractor . . . . .	22
3.2.7	Conversation Handling and Answering User Questions . . . . .	23
3.2.8	User Interface . . . . .	31
<b>4</b>	<b>Third Party Functionalities Used</b>	<b>35</b>
4.1	Dialogflow . . . . .	35
4.1.1	Description and Usage Justification . . . . .	35
4.1.2	Licensing Impact . . . . .	35
4.2	ECharts.js . . . . .	36
4.2.1	Description and Usage Justification . . . . .	36
4.2.2	Licensing Impact . . . . .	36
4.3	AWS . . . . .	36
4.3.1	Description and Usage Justification . . . . .	36
4.3.2	Licensing Impact . . . . .	36
4.4	Heroku . . . . .	36
4.4.1	Description and Usage Justification . . . . .	36
4.4.2	Licensing Impact . . . . .	37
<b>5</b>	<b>Development Methodology and Software Implementation Challenges</b>	<b>37</b>
5.1	Development Methodology . . . . .	37
5.1.1	Continuous Integration and Continuous Delivery (CI/CD) Infrastructure . . . . .	37
5.2	Implementation Challenges . . . . .	39
5.2.1	Sparse and Unclean Data Source . . . . .	39
5.2.2	Dialogflow NLP AI Model Accuracy and Agent Training . . . . .	39

5.2.3	Fallbacks and Error Handling . . . . .	39
5.2.4	Architecture And Infrastructure Challenges . . . . .	39
<b>6</b>	<b>User Documentation and Manual</b>	<b>41</b>
6.1	For Developers . . . . .	41
6.1.1	Running the main software . . . . .	41
6.1.2	How to train a new intent using command line . . . . .	42
6.2	Training Data Configuration . . . . .	43
6.2.1	Intents . . . . .	43
6.2.2	Entities . . . . .	46
6.3	Using the Software (For typical user) . . . . .	47
6.3.1	Admin Users . . . . .	47
6.3.2	Student Users . . . . .	48
<b>7</b>	<b>Conclusion</b>	<b>48</b>



# WhatBot

Your mighty online support tutor

# 1 Quick Overview

WhatBot is a student-tutor support chatbot developed for UNSW CSE Postgraduates to help teachers handle common student queries preemptively and also common interactions that students want to make with lecturers such as booking a course consultation. It aims to be easily scalable and can react to new situations quickly through automatic AI training such that even users with no background in AI or computer science can easily use it to enable its powerful functions. Therefore, while WhatBot currently only supports UNSW CSE Postgraduate courses, the current version act as a proof of concept such that WhatBot can be easily configured for other faculties or education institutes to use as long as data source is available.

The live version of WhatBot can be found at: <https://whatbot9900.herokuapp.com/>

Before using the web app, users are required to sign in first. There are two types of users, admin and students, and one of their credentials are:

- Admin:
  - Username: z0000000, Password: 0000000
- Student:
  - Username: z1234567, Password: 123456
  - Username: z8888888, Password: 123456
  - Username: z9999999, Password: 9999999

Admin has full access to all features, which are chatroom, automation training and data analytic dashboard. On the other hand, student only has access to the chatroom.

Note: These login credentials are simply made up as this software is created for the purpose of proof of concept and is not linked to real UNSW database.

## 2 Introduction

### 2.1 Background

With recent advancement of technology, artificial intelligence software has started to gain a lot of attention. Most of these software are built for a purpose of reducing manual and repetitive work for humans. One of these task is customer support where people often have similar question for a product or service as information they can find online is typically not clear. As a result, chatbots have become increasingly popular for businesses to provide automated 24/7 support.

#### 2.1.1 History

While chatbots seems to be an emerging technology, they actually have a long history and have been around since computers existed. The very first chatbot was created at the MIT in 1966 by the name of Elizza which was later called the “chater bot”<sup>1</sup>. Elizza was first created for the purpose of human to machine communication and was able to impersonated the responses of a non-directional psychotherapist in an initial psychiatric interview. This increased people’s attention on to chatbot systems and there became more focus on making chatbot systems more natural and be able to hold conversation just like a human being. As such, the next notable advancement in chatbot came with PARRY, which was able to act like Elizza but with a more human attitude<sup>2</sup>. However, these early day chatbots had

---

<sup>1</sup>(2018, July 20). The Evolution of ChatBots – BlueEast – Medium. Retrieved April 25, 2019, from <https://medium.com/blueeast/the-evolution-of-chatbots-d2753e3b0b6a>

<sup>2</sup>(n.d.). A History of Chatbots - ChatBot Pack. Retrieved April 25, 2019, from <https://www.chatbotpack.com/a-history-of-chatbots/>

a clear weakness such that they were designed based on some hand crafted rules and could not react to unplanned scenarios which often lead to bad user experiences<sup>3</sup>.

### 2.1.2 Existing Systems

With recent advancement in AI and NLP, we can see businesses today have all started to incorporate chatbot into their service especially in customer service or sales as they are no longer based on fixed rules like the early day ones. Furthermore, chatbot has also becoming increasingly popular for mental health applications as well. Some existing chatbot include Woebot, XiaoIce made by Microsoft and Google Duplex<sup>4</sup>. These chatbot aims to simulate human speech pattern to a point that the end user won't even notice that they are speaking to an AI. Furthermore, they also aim to be able to complete tasks that seems to be difficult for traditional AI, such as making a reservation for a restaurant.

#### 2.1.2.1 Problems and Drawbacks of Existing Systems

While chatbot system has become increasingly more powerful in recent years, their advancement has being more focused on creating actions rather than reacting to situations. Reacting to situations is for tasks such as answering a customer's question where conversation is initiated by user and chatbot has less control of the flow of the conversation since it is not the one asking the questions, while creating actions is tasks like making a reservation for a restaurant where the chatbot makes the initiative and have more control of the conversation as the initiator since it can expect different forms of replies. As a result, reactive chatbot such as customer service and sales chatbot are still not as powerful since queries from user comes in many different form. Due to this, chatbot in customer service are currently more focused on preemptive customer service and the more complex tasks are still handled by humans<sup>5</sup>.

Furthermore, many of the chatbots today in customer service still rely on a keyword driven system and this leads to user having to be very specific in their queries for the system to perform<sup>6</sup>. They also can only handle very simple question and answer interactions. This often lead to bad user experiences. From this, we can see that there is a need to make chatbot better at reacting to different situations to enhance customer experiences and also provide features that is different from simple question and answer.

## 2.2 Aim

WhatBot aims to solve the problem of chatbot systems not being able to quickly react to new situations in the context of a student-tutor support system. It is a chatbot that will help scaling up a small teaching team, like one to two lecturer in charge with a few casual academic tutors, to serve courses with large number of students as it won't require users to spend lots of time on updating their system when new situation or requirements arises.

WhatBot will be a modern, user friendly and effective student-tutor support system that aims to solve the student support service problem where there are too many questions from students and too few lecturers or academic tutors to answer them. These situations are especially evidence during enrollment period where many students have questions on various courses and are planning their course work schedule. While many of the answers to their questions can be found online, the information source is too spread out usually resulting them being hard to find or people not willing to search for it. Therefore, this software will provide a high level of utility such that it not only benefit students but also the teaching staffs and have all the information to answer their queries in one place.

---

<sup>3</sup>(2019, January 2). Development of social chatbots – Towards Data Science. Retrieved April 25, 2019, from <https://towardsdatascience.com/development-of-social-chatbots-a411d11e5def>

<sup>4</sup>(2018, May 22). Microsoft also has an AI bot that makes phone calls to ... - The Verge. Retrieved April 25, 2019, from <https://www.theverge.com/2018/5/22/17379508/microsoft-xiaoice-chat-bot-phone-call-demo>

<sup>5</sup>(2018, May 23). How The Development Of AI Has Advanced The Technology ... - Forbes. Retrieved April 25, 2019, from <https://www.forbes.com/sites/forbestechcouncil/2018/05/23/how-the-development-of-ai-has-advanced-the-technology-available-for-chatbots/>

<sup>6</sup>(2018, May 15). When and why do chatbots make a mess of customer service .... Retrieved April 25, 2019, from <https://www.mycustomer.com/service/channels/when-and-why-do-chatbots-make-a-mess-of-customer-service>

The software will provide automatic AI training feature that allows user to quickly scale up the system to be able to handle new type of queries. Furthermore, the system also provide performance feedback through the form of data analytic dashboard such that users can know how the system is performing and what they need to do to improve the system. Furthermore, it will not only handle common question and answer feature that exist in most chatbots, but also be able to handle course consultation booking interaction to show that it is also a management system and not just a dumb question and answer chatbot. WhatBot is also able to hold conversation with users and understand the context of the conversation such that it will ask user for information required to answer a question if user did not provide them in their queries.

From the limited time of the project, WhatBot only supports UNSW CSE Postgraduate courses as data extraction and cleansing of all courses in UNSW was estimated to take too much time and the team decided to put more focus on other more important features.

## 3 System and Software

### 3.1 Software Architecture and Design Overview

WhatBot uses a client-server architecture with a React.js frontend and Python Flask backend. The backend compose of several modules and utilizes AWS cloud service for data (RDS) and file storage (S3). Both components runs on nginx web server in Docker containers in the live deployed app hosted on a cloud service, Heroku, which is based on AWS but has built in tools and infrastructure for web server and deployment<sup>7</sup>. Diagram below gives a brief overview of the components and system architecture:

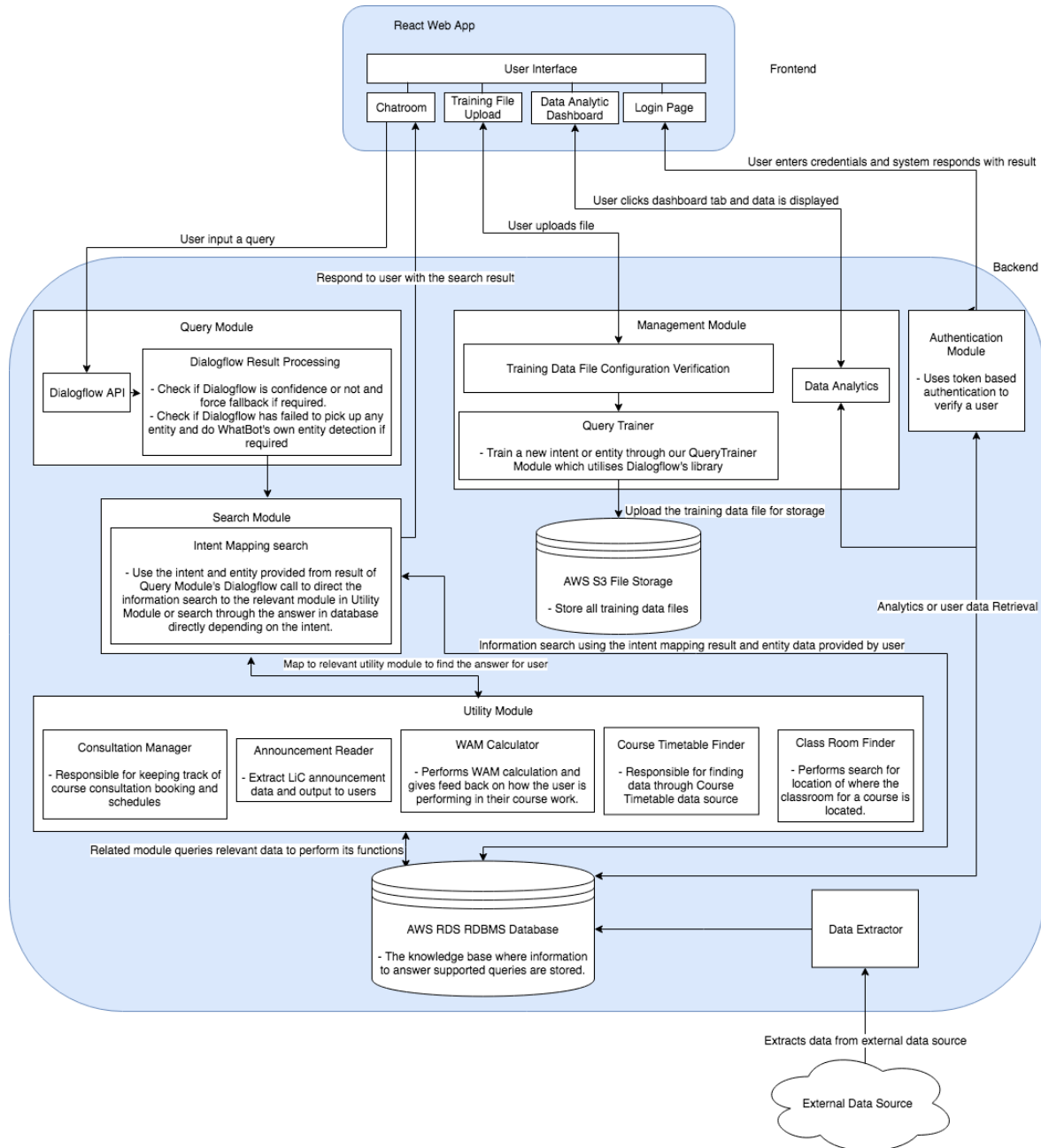


Figure 1: WhatBot System Architecture Overview

<sup>7</sup>(2015. Oct 15.). Why do people use Heroku when AWS is present? What distinguishes .... Retrieved April 25, 2019, from <https://stackoverflow.com/questions/9802259/why-do-people-use-heroku-when-aws-is-present-what-distinguishes-heroku-from-aws>

## 3.2 Core Components and Features

### 3.2.1 Query Module

#### 3.2.1.1 Overview

Query module is a crucial component that allows the chatroom feature to function. It is the backend component that is responsible for doing query processing and interacting with the Dialogflow AI agent through Dialogflow library API. It is also responsible for processing the result of Dialogflow API calls as there are often cases when Dialogflow Agent is not so confident on its own result or when it has failed to detect the right entity/key words in user's query. As a result, query module does another layer of analysis on the query and Dialogflow result to enhance entity detection before passing the information for search module to look for answer to user's query. It also sends the user query data and Dialogflow API call result data to management module for data collection purposes.

#### 3.2.1.2 Features

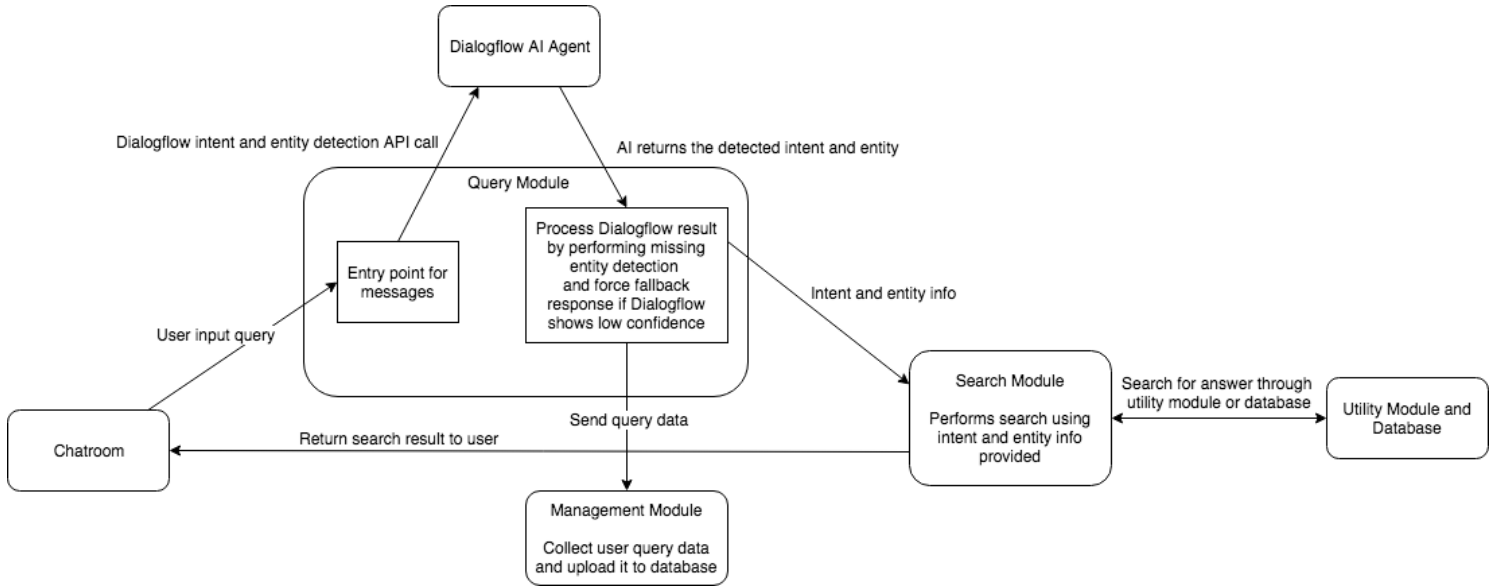


Figure 2: Query Module Data and Workflow

##### 3.2.1.2.1 User Input Intent Classification and Entity Extraction

As illustrated in figure 2, when user enters a query through the chatroom, an API call to Dialogflow with the input is made and this is where intent classification is made. This API call gives us access to the AI NLP model we have pre-trained already on Dialogflow and will return us result of which intent Dialogflow thinks the user query belong to and also the entities that exist in the user input. The Dialogflow response is consist of three main components: the intent it detected the query to be, a confidence level and a response message. The intent it detected is essentially considered as the intent the query is classified as.

Query module checks the confidence level of the API result as Dialogflow API responses are of a `client.response` class which has an attribute for confidence level as indicated by it's documentation<sup>8</sup>. In the situation where Dialogflow shows low confidence, the query module forces the query response into a fallback response and tells the search module this information so that it returns a fallback response.

Finally, query module checks the message of the Dialogflow API response as the AI has been setup to respond with entity values if it is an intent that requires entity in user queries. Entities can essentially be considered keywords

<sup>8</sup>(n.d.). Dialogflow Python Client v2 | Read the Docs. Retrieved April 26, 2019, from <https://readthedocs.org/projects/dialogflow-python-client-v2/>



and some queries require key words so that they can be used to search for information. When query module checks the response message, it checks if all keywords are picked up by Dialogflow and if not, it process the original query string using regular expressions to pick up those keywords and then update this information to search module for information retrieval. However, it also failed to pick up the keywords then it could mean that user simply did not provide the entity values that are crucial for their query and query module forces a fallback response like the low confidence level case and ask user to be more specific in providing the missing info.

Meanwhile, the result of Dialogflow API call and user queries are passed into management module in another thread for data collection purpose which is explained in more detail in the management module's data analytics section.

### 3.2.1.2.2 Query Intent AI Training

The query trainer class in train.py is responsible for AI model training. This allows user to train new intent by simply providing some training data which follows the configuration in the instruction below. Through this feature, even when user only provide little training data, the query trainer can also automatically generate variations of the training data to increase number of training phrases, which enhance the accuracy of the models.

This feature provided by query module is in general not used in standard question and answer operations. However, it is responsible for the accuracy of the user input intent classification and is used for the automatic AI training feature in management module. More detail on the power of this feature in management module's automatic AI training section.

## 3.2.2 Utility Module

### 3.2.2.1 Overview

Utility module provides many utility classes like WAM calculator, course consultation booking manager, classroom finder, course timetable finder, email system and announcement reader. These classes do more than just simple database queries such that they perform some form of analysis or processing of the data before returning the result to users.

### 3.2.2.2 Features

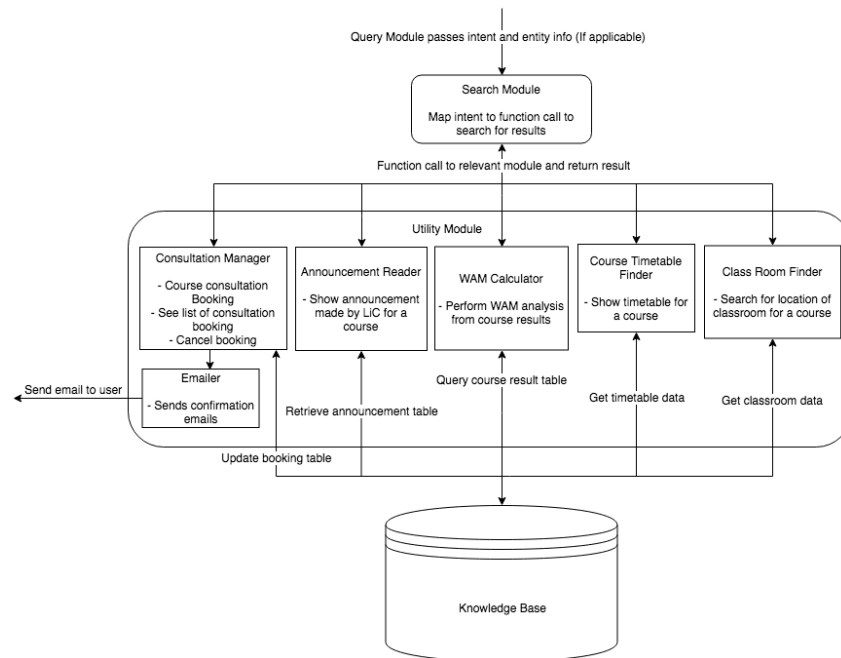


Figure 3: Utility Module Structure and Workflow

### **3.2.2.2.1 Course Consultation Booking**

With this system, there is no more need for students to manually look for LiC's contact to email and ask for when a lecturer would be available for a consultation as WhatBot's booking management system will handle all these trivial tasks for lecturer to pre-advertise what time they would be available such that if a student tries to book for a time that has already been booked by another student, WhatBot will let them know straight away instead of having the student to wait for a few days just to find out that the lecturer wasn't available for that time.

This is WhatBot's course consultation booking management system. It allows users to do the following:

- See list of course consultation booking they've made
- Book a course consultation
- Cancel a course consultation booking they've made

With a successful course consultation booking or cancellation, users also receives confirmation emails. However, as we do not have real email data, it is currently configured to only send email to WhatBot itself. Nevertheless, a simple change in variable name in the code will make it send email to real users.

As you can see, it makes the consultation booking process a lot smoother and reduce workload and waiting time for both students and lecturers. Furthermore, it also imposes the following rules to make the system more robust:

- No course consultation can be booked on the weekends.
- No course consultation booking can be made for a date that is more than 7 days away since it is difficult for lecturers themselves to even know their plans for times more than a week away.
- Pre-allocated time slots at every hour from 9am-5pm such that these are the only times people can book consultation for because having time precision up to minutes is hard for anyone to manage and is difficult scheduling problem.

This feature works by having a preset of rules in the program and a database specifically designed to suit its operations. Upon a request, the program checks for the conditions mentioned above and perform the related operations. Confirmation email system works by using SMTP mail server to connect to the Google gmail server and using user provided information for their booking, it fills these information onto a message template before sending it out.

### **3.2.2.2.2 Announcement Reader**

This feature offers students to read announcement made by lecturers without the need of going to course websites. With students taking so many courses each semester, each course typically have their own course website, students will need to spend lots of time and effort to go through each of them to check daily for all the updates. However, with this feature, they can simply ask WhatBot what the latest announcements are for a course and WhatBot will summarise it for him/her.

When a lecturer/admin wants to make a new announcement, details of the announcement is added to the database and this feature will then automatically grab the latest one.

### **3.2.2.2.3 WAM Calculator**

Students often want to find out how they are progressing in their current degree and the current process for this is quite exhausting as it requires students to students are required to log into myUNSW and download the course transcript to see their marks for all courses and their overall grade. This results in unnecessary downloads quite often. With this feature, student can quickly gain a view on all their course grade and also be able to see the overall result.

This feature works by going into the database and look for the student's course results then perform computations to get their overall grade.

Furthermore, this feature is also different for lecturers/admin users as they can gain information on any students by simply including the targeted student's ID inside their query.

#### **3.2.2.2.4 Course Timetable Finder**

Course timetable is often a common concern for students as they need to pre plan their schedule for a semester. The current method of going to UNSW timetable website and opening multiple tabs for all the course a student is interested in is exhausting and inefficient. This feature allow students to get all the timetable information they need inside WhatBot such that they can enter consecutive queries and compare the results instantly without going back and forth between tabs.

This feature works by going through database to search for the required information and reformat the timetable to be easier to read for humans.

#### **3.2.2.2.5 Classroom Locator**

Classroom locator allows student to quickly find information on where their classes are. Usually, in the beginning of a semester, this is one of the most common questions students have to Google search for as they are not familiar with their timetable. Searching for timetable right now requires student to log into myUNSW and navigate to timetable which takes some time to load as myUNSW is a more heavyweight application. WhatBot is much more light weighted and loads faster to be able to provide this information in a more instant manner.

WhatBot is faster as this feature has being optimized for handling this type of query since we have database specifically dedicated to classroom locations and expensive joins are not used in our queries. Therefore, this feature works by having denormalized database tables for efficiency.

### 3.2.3 Management Module

#### 3.2.3.1 Overview

The management module is responsible for performance and feedback features such as automatic AI training, data collections and data analytics. It provides the interface for automatic AI training APIs and also makes connection to file storage, which we choose to use AWS S3. File storage is used as it is inefficient to store files in RDBMS databases and we require it in situations where our deployed docker container dies, the uploaded files stored inside the docker containers will be lost so we need to be able to retrieve back user uploaded files so we can retrain AI model if required or do other development using their uploaded files.

#### 3.2.3.2 Features

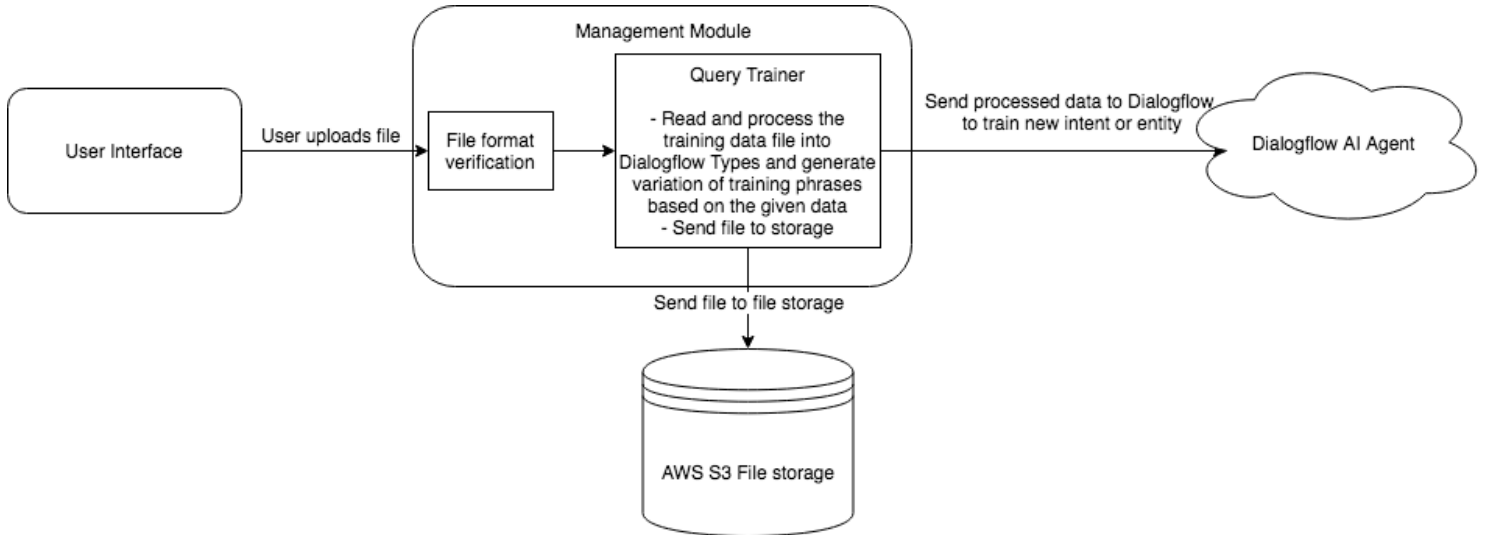


Figure 4: Automation Training Workflow

##### 3.2.3.2.1 Automation AI Trainer

This feature allows admin users or engineers to update WhatBot’s NLP models by creating a new intent or entity without manually going into Dialogflow console. It leverages Dialogflow APIs so that users can simply upload a training data file, which follows the configuration described in user manual section of this report, through the “Upload training data” tab in WhatBot’s UI and the intent or entity will be trained in Dialogflow. This automation process makes WhatBot unique as it means developers or administrators can quickly react and adapt WhatBot to new type of queries as long as the data required to answer those queries is available in the database. Furthermore, it also generate significant more amount of training queries with only a few training data and this remove the manual process of having user to enter variations of training queries by themselves on Dialogflow console. In summary, automatic AI model training provides these advantages:

- Rapid Dialogflow NLP model update making it easily adaptable to new use cases by creating or updating intents and entities.
- Generate training queries. For example, given 10 training data, it can produce more than 200 training queries to be used to make the NLP model more robust.
- Useable through web interface or command line, making it user friendly for even user without any computer knowledge.

#### Usage example

For a typical user to use our own training pipeline to train new intent with just one click. Specifically, instead of submit training corpus phrase by phrase in Dialogflow console, our auto training pipeline can generate hundreds of training phrase with a template file within 15 lines, and afterwards it will pipe them into dialog flow training API.

User steps:

1. User sign in as an admin using the credentials above.
2. User clicks the “Upload training data” tab on the top of the page.
3. User uploads their training template file by clicking at the cloud icon in the middle of the page. (template file format is provided in this page as well)
4. User clicks “Upload”, then new intent training will start.

Results:

1. New intent in Dialogflow, which is used as the intent/entity detector/extractor in query understanding.
2. New chat feature enabled by the new intent and user can perform new type of queries related to that intent.

### How is it different from Dialogflow console?

Dialogflow also provide a feature for user to upload conversation logs into it's [console](#). However, WhatBot's automatic AI training feature is a lot more powerful and this will be explained in this section with the following four points.

#### 1. Dialogflow console's upload file cannot create a new intent directly

While conversation logs can be uploaded onto Dialogflow console, it cannot be used to create a new intent. Dialogflow will only try to match the lines in conversation logs to an existing intent and add the lines as training queries into those existing intent. Therefore, you cannot create a new intent using console's feature without having to manually create that intent yourself and enter some training queries for the lines in the conversation logs to be matched to it. If you do not do so, you have to manually label each line in the conversation log you uploaded to an intent yourself.

However, with WhatBot's automatic AI training feature, user's can create brand new intent or update an existing one by simply specifying the intent name along with some training queries.

Example scenario: User is trying to create an intent for course outline.

- User using Dialogflow console
  - Step 1. User prepares the conversation log which looks like a .txt file with the following content:

```
I want to get the course outline for COMP9900

Show me the course outline for COMP9900

Course outline for COMP9318

COMP9415's course outline

Outline for COMP9331
```

- Step 2. On Dialogflow console, user navigates to the “Training” tab and click “Upload”.

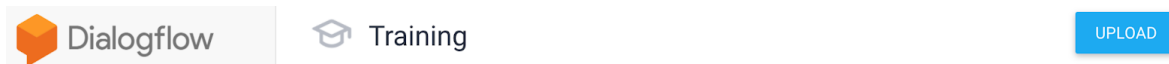


Figure 5: Dialogflow manual training upload button

- Step 3. After upload, they see the 5 requests because we entered 5 lines above and we also see 5 no match because the intent for course outline related queries did not exist before.

Conversation	Requests	No match	Date	
I want to get the course outline for COMP9900	5	5	Today	>

- Step 4. User is then required to go into it and manually create the new intent themselves and also label the entities as well.

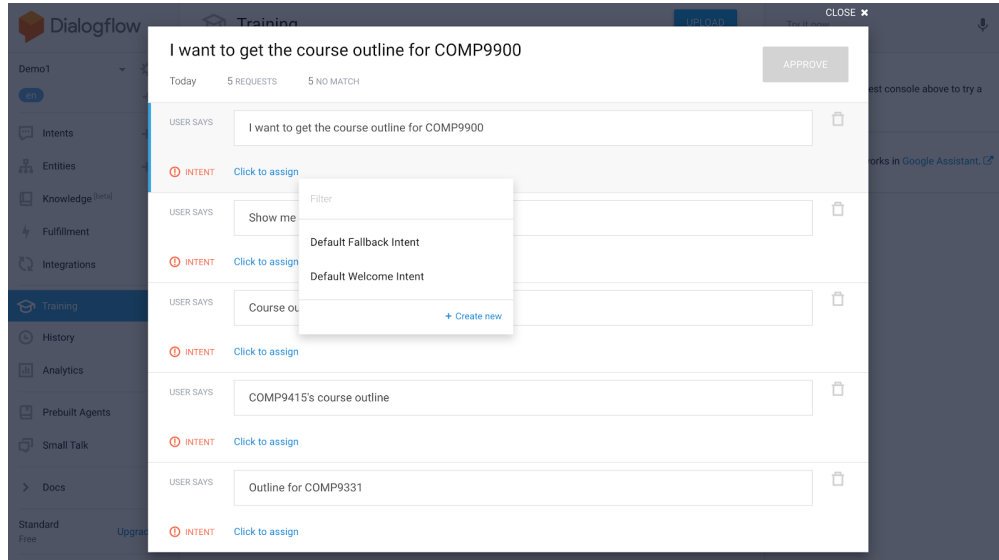


Figure 6: User creating a new intent themselves by clicking create new.

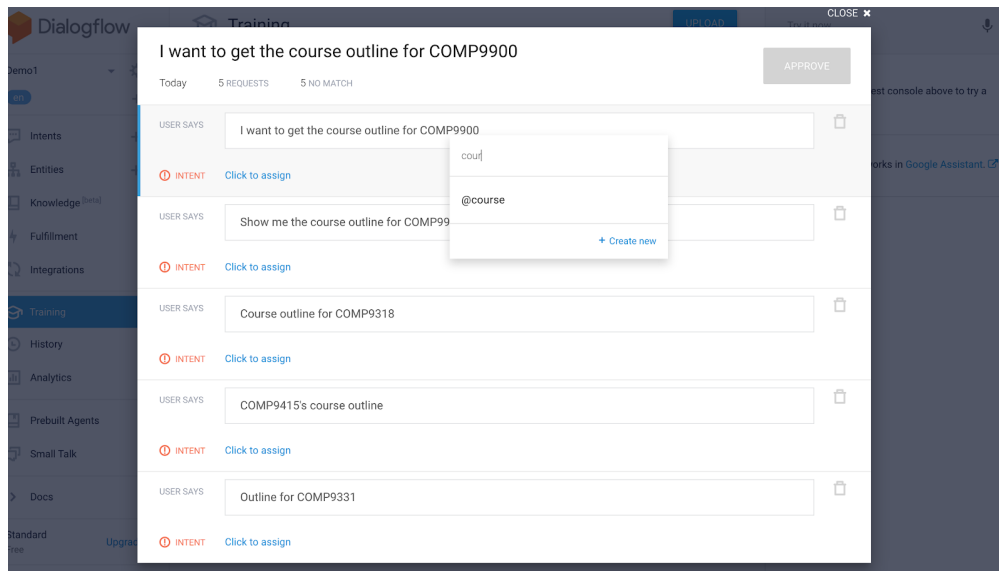


Figure 7: User doing manual entity labelling as Dialogflow console often does not pick up custom entities.

- User using WhatBot's automatic AI training feature
  - Step 1. User prepares the training data file where they can specify the intent's attributes like: display\_name, message\_texts, parent\_followup, input\_context, reset\_contexts, output\_context. More de-

tail on training data file configuration click in the user manual section of this report. In this example, we will use the an example configuration file which is also for the same scenario above:

```
display_name course_outline_queries

message_texts $course

intent_types course_outline

reset_contexts

What is {course code} about?

Tell me more about {course code}

What can I learn in {course code}

Description of {course code}

Course outline for {course code}

Give me the course outline of {course code}

Give me a description of {course code}

...

I want to know about {course code}

I want to find out more about {course code}

Tell me about {course code}

{course code} description
```

- Step 2. User uploads the file through web interface.

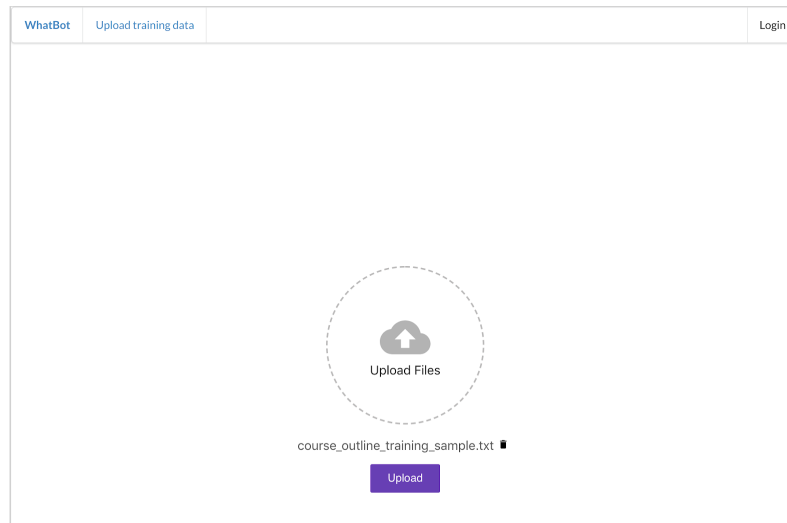


Figure 8: User uploads the file through web interface.

- Step 3. User can see that the intent has being created on Dialogflow console with more than 300 training phrases that was generated by WhatBot’s training query generation feature. User can also see that entities has being labelled through WhatBot’s backend doing the query analysis.

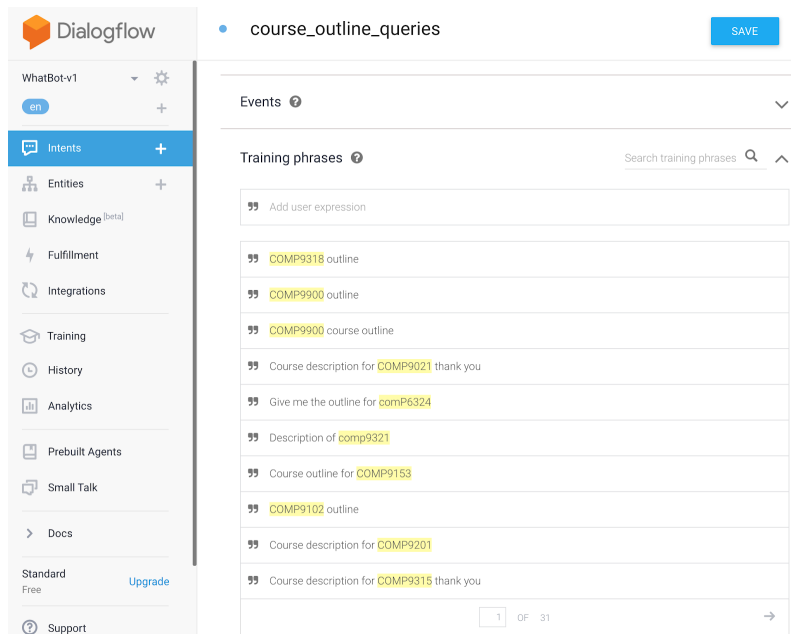


Figure 9: console shows generated intent

## 2. Dialogflow console’s upload file cannot label custom entites accurately

Using the example used above, we can also see that Dialogflow console’s upload file feature cannot label custom entities accurately very often while WhatBot’s automatic AI training features leverages it’s entity training feature and having user provide where these entities should be to accurately label them using Dialogflow APIs for training.

## 3. Dialogflow console’s upload file cannot be used for creating new entities or updating existing entities

The upload file for training feature in Dialogflow cannot be used for training entities while WhatBot’s automatic AI training feature can be by following the entity training file configuration mentioned in [here](#). This is also the feature that allow accurate entity labelling mentioned in the last difference.



#### 4. Dialogflow console's upload file cannot be used to configure an intent, e.g. followup, response and contexts

From the usage example above, you can see that parameters such as `message_texts` and `reset_contexts` were specified in WhatBot's configuration file as a requirement. This allowed WhatBot users to setup various parameters for the intent right away upon creation. Parameters like `message_texts` is used to specify how the intent should respond and `reset_contexts` is used to specify whether this intent will mark the end of a conversation. WhatBot also offer context and followups to be configured as well by specifying fields such as `parent_followup`, `input_context`, `output_context` and `action`. For more detail on the training data configuration see the user manual section of this report. All of these, cannot be done on Dialogflow automatically.

### 3.2.3.2.2 System Feedback and Data Analytics

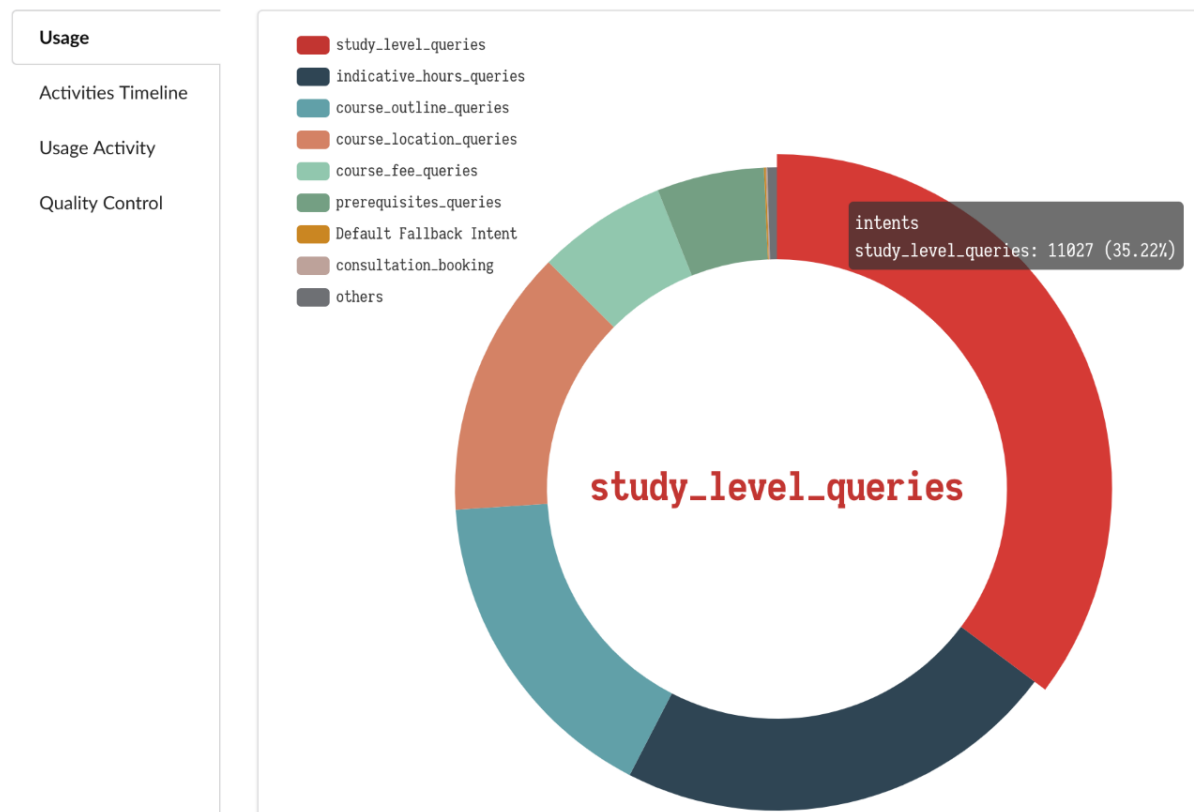
WhatBot's system is developed to assist the daily work of student office, for example, CSE student service.

So it can provide insights on user usage stories in the form of a data dashboard can be like this and allow admins to gain feedback on how the system is performing and improve the system for new and current use cases. A usage workflow looks something like:

1. Admin opens up the dashboard on the web app and sees an overview of recent user activity.
2. Admin finds something interesting/abnormal, she can dive into details by seeing the timeline data.
3. Admin finds some trend in the usage activities. For example, singhe found `study_level_queries` are very popular, she would be wondering the reasons behind these activities. The manager would then decide if they should take some action to reduce the students' "worries" about these areas.
4. Admin finds some intent have low confidence and initializes a Performance Improve Plan to improve the AI and creates training data to fix this low confidence issue.

This features queries relevant data from the database and perform analytics to produce results that are in some form of data structure that matches what WhatBot's frontend is requesting form to be used for visualisation. Some of the analytics include

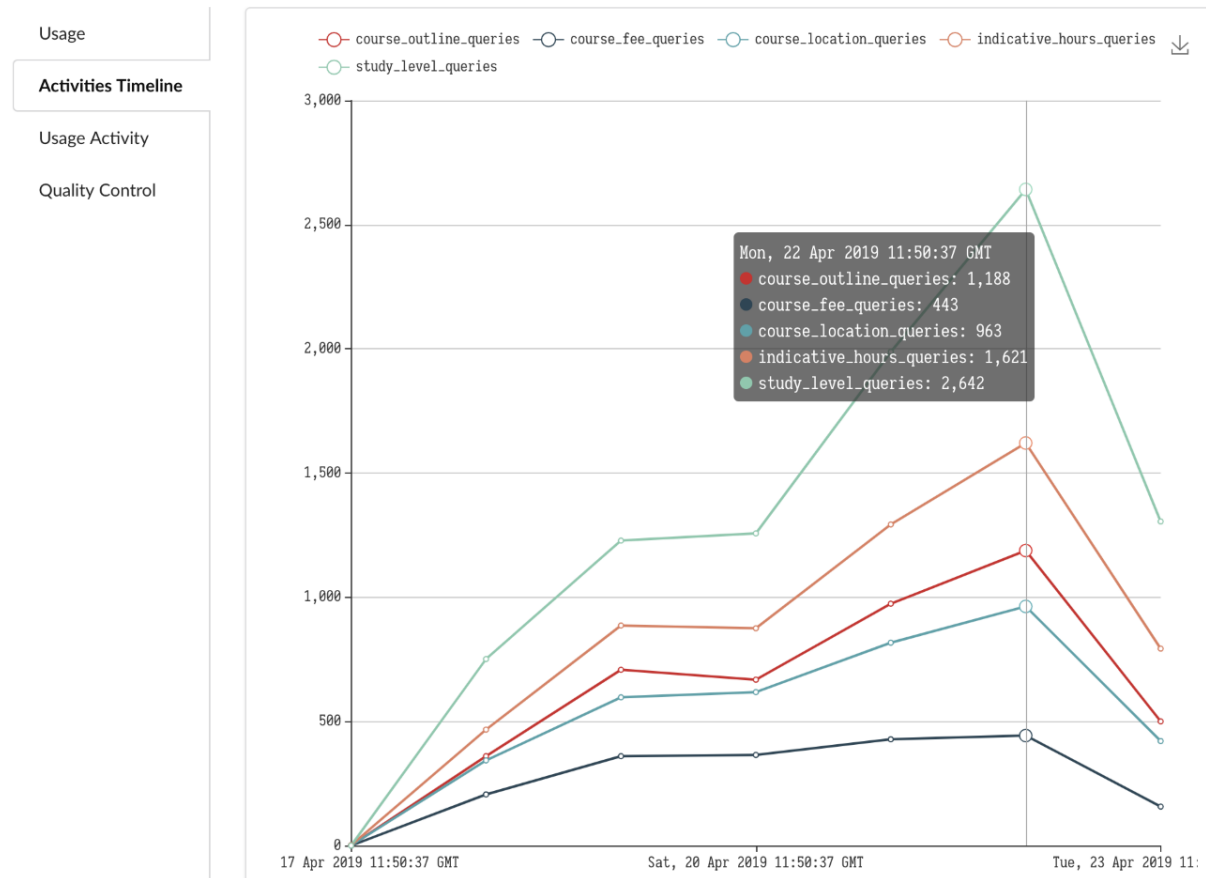
#### Intent Usage



Provides data on the amount of usage for each intent and return the result to the user. Therefore, user who calls this can get a better idea of how their system is being used and what the consumer cares most about. Provides the following insights:

- Reveals the most popular topics, thus you can know what your students are thinking/caring/worrying about in a given time period.
- Find out what the current hotspot feature is, this can help you make a future development plan like features improving/expanding based these facts.

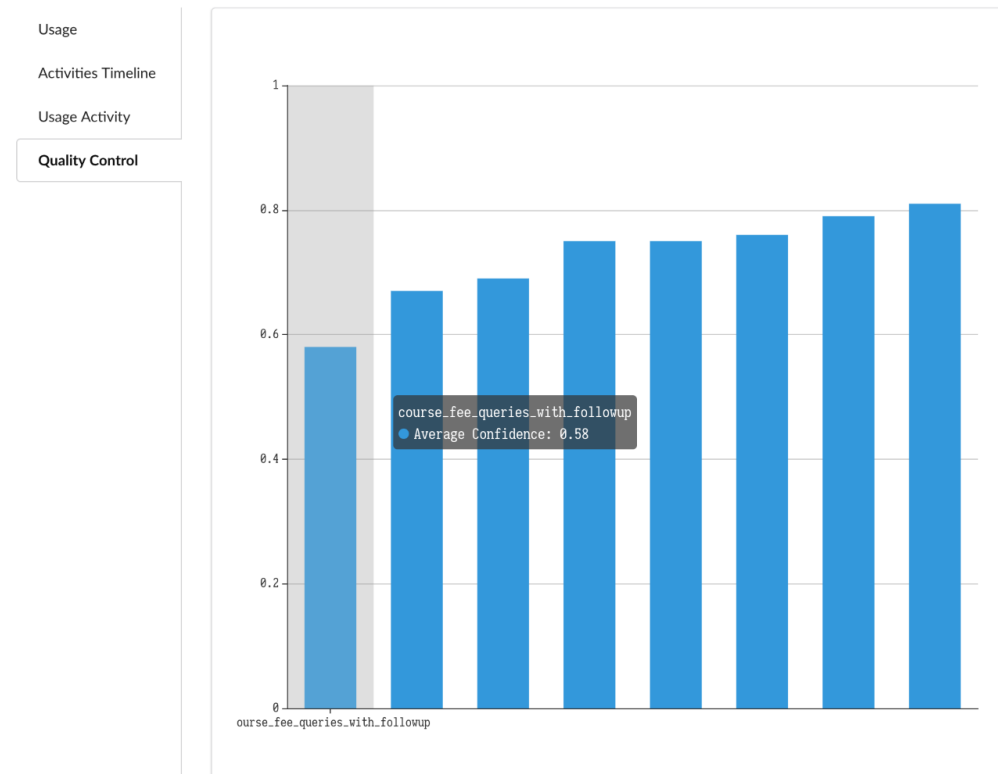
## Intent Timeline



Provides a timeline data (based on last 7 day) on how much each intent has been used. From this, user can see what is the current trend of their consumer and be able to react to more people's questions faster. Provides the following insights:

- See how exactly each topic is growing for N= days.
- Reveals what is trending right now, which is a pretty similar idea with Google

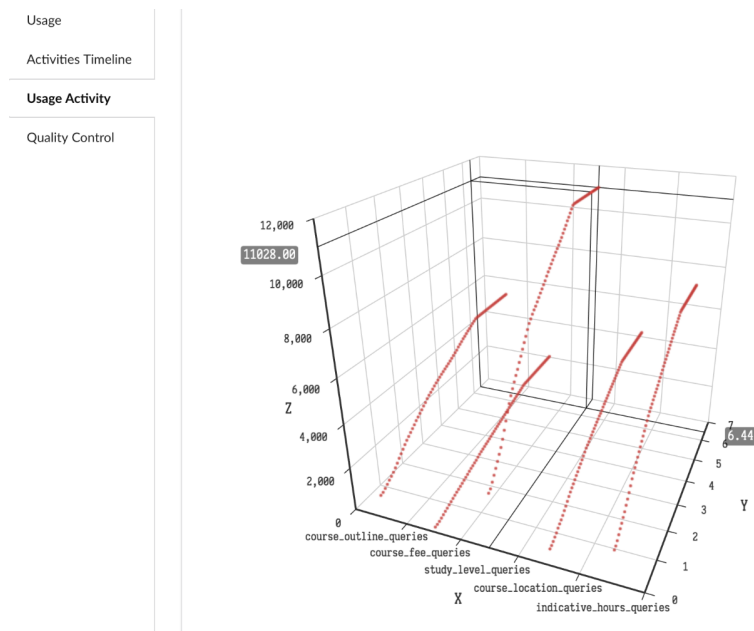
## Intent Confidence



Provides the average confidence level that Dialogflow has for the bottom n intent. This lets the user know how well the system is performing so that they can use the automation training feature to improve the system if required. Provides the following insight:

- Find out which intents are the AI not performing well in and may require additional training.

## 3D Timeline Intent Usage



Provides a 3D view of top n intent's accumulative usage in the last 7 days. Provides insight on:

- A better understanding of user behaviour growth over the period instead of the density of each day.

### 3.2.4 Search Module

#### 3.2.4.1 Overview

Search module is responsible for searching for result to queries and formulating responses to user queries. It provides an entry point for query module to give the intent, entities and user message to search module so it has some information on how it can find the result user is asking for. Search module also uses the utility module in some situations to perform the appropriate tasks for some functions such as when it finds user is asking to see their WAM.

#### 3.2.4.2 Features

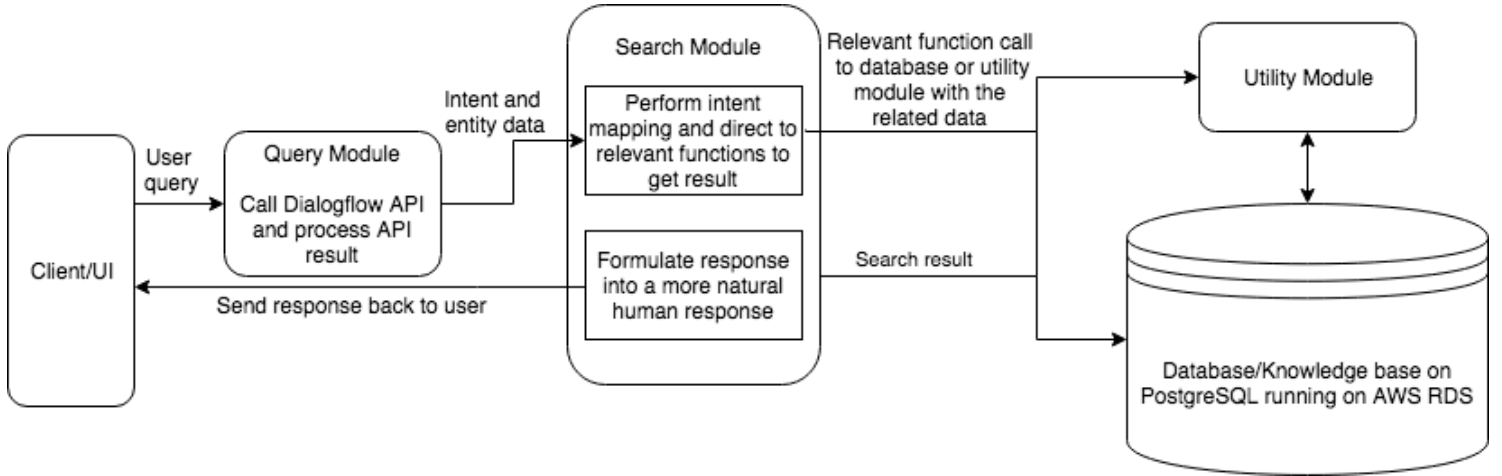


Figure 10: Information Search and Response Workflow

#### 3.2.4.2.1 Information Search and Response Formulation

When the query module provide an intent and entity data, search modules uses a query map to process on how to perform information search. It takes the intent and map it to its corresponding function with the entity values as function arguments, if applicable, to get query results. Depending on the intent being triggered, sometimes search module uses utility module, which queries database to get the information it needs to perform its relevant function, to get the right results. At other times, the search module will make direct query to database to extract required data to formulate responses.

After getting the results from either database or utility module, search module takes the direct result and format them into strings, which are made of templated sentences, to make the response sound more human like and return the resulting string to client.

### 3.2.5 Database and File Storage

#### 3.2.5.1 RDBMS - Knowledge base

##### 3.2.5.1.1 Database Tables

###### *Announcement Table*

The announcement table composes 4 columns as seen in data tables figure. This table is mainly for the announcement reader feature described in the utility module section above.

###### *Classroom Table*

This table enables the class room finder of the utility module to work. It provides information for various courses' locations.

Announcement	Classroom	Consultation	Courses	Course Timetable	Handbook Information
Course code	Course code	Course code	Course code	Course code	Course code
Course name	Location	Student ID	Course name	Course name	Course name
Date		Date	ADK	Time	Credit
Content		Time	Rating		Prerequisites
			Content		Outline PDF URL
Intent Collection	Intent Collection	Student Results	Users		Faculty
User query text	User query text	Student ID	Username		School
Intent	Intent	Course ID	Password		Offering terms
Time	Time	Marks			Campus
Confidence	Confidence				Outline text
					Indicative hours
					Fees

Figure 11: Data Tables

#### *Consultation Table*

This table is the backbone of the consultation booking management system. It stores all currently booked times and upon cancellation, data is deleted. Furthermore, when a booking has expired for a long time, we perform periodic clean up to clear data storage.

#### *Courses Table*

This table contains information on a course such as course code, course name, whether it is ADK or not, student's rating and description content. This table is used for answering some intent's questions.

#### *Handbook Information Table*

This table contains information grabbed from UNSW course handbook. This is one of the main database that WhatBot relies on to answer common student queries since many student queries answer is typically found here.

#### *Courses Timetable Table*

This table contains information on timetable information and it is mainly used by the course timetable feature described in the utility module section.

#### *Intent Collection Table*

This table stores all the user data. It stores the user input, what Dialogflow classified the query as and its confidence level and also the timestamp of when the query was made. This data drives the management module's data analytic feature.

#### *Student Results Table*

This table stores information on student's course result and is the key data source of the WAM calculator feature.

#### *Users Table*

This table simply stores user credentials. It is used for login and authority level check.

### **3.2.5.1.2 Database Design**

Overall there are 10 tables inside our database as the team has decided to denormalize as much when designing the database. The team chose to denormalize and create redundancy because we chose to prioritize speed and performance over database storage cost. This is because the scope of the project was focused on UNSW CSE postgraduate courses so we knew that we won't need to store large amount of data so storage cost became less of a concern.

The denormalization was based on requirements of the different features such that we have dedicated tables for the key features as they will be utilized more and speed is important for them.

As a result, you can see from figure 5 above, course code is used in many different tables as keys very often and introduces lots of redundancy. However, also due to this design, we have less expensive joins in our SQL queries making our system have better performance.

### 3.2.5.2 File Storage

In WhatBot's system, file storage is used to store all the training data file for the automation AI training feature mentioned previously in the management module section. It is required as the system is deployed using Docker and when user uploads a training data file, it is first stored locally in the Docker container before being read by management module to perform automated Dialogflow AI training. If we don't store it in an external file storage, in situation where the Docker container or web server dies, all the file stored locally will be lost. This is obviously problematic as we have no way of retrieving these files in situations where we want to retrain the AI.

The file storage chosen is AWS S3 as we have also choose AWS RDS for our other storage and using another AWS services goes well with it.

### 3.2.6 Data Extractor

#### 3.2.6.1 Overview

Data extractor is essentially what WhatBot use to get external data source into our knowledge base. It is essentially a web scraper that is configured to extract information from UNSW course handbook website and performs data cleansing before uploading the data into the database.

#### 3.2.6.2 Features

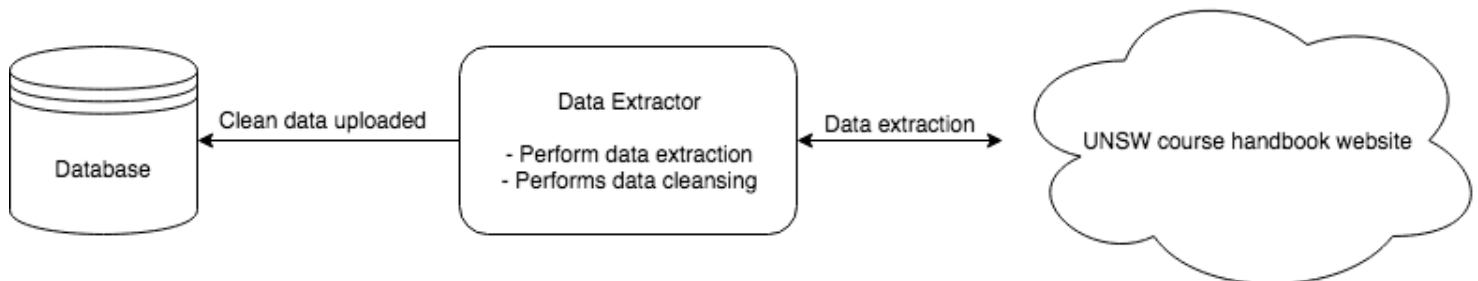


Figure 12: Data Extraction Process

#### 3.2.6.2.1 Data Extraction and Cleansing

Data extraction feature works by using the Python web scraping library, beautiful soup, to crawl through the web. This feature is used by internal system and for development where an url will be given as the input and the data extractor will search for the url and perform web scraping to get key contents out. The data extractor is then also responsible for reformatting the contents scraped into a usable format that is usable by our database schema and uploading them into the database.

As the feature works by having data extractor configured for the HTML of a specific website, this feature will not work on non UNSW course handbook related website as it is not designed to be an universal one as mentioned in the project proposal.

The feature for periodic update discussed in the project proposal can be made into production through this module as user can periodically run data extractor to update the system database easily.

## 3.2.7 Conversation Handling and Answering User Questions

### 3.2.7.1 Overview

This is the core feature of WhatBot and it is what makes the system a chatbot. This feature is empowered by a combination of components, such as query module, search module and utility module as mentioned above. It shows that WhatBot is able to not only handle simple question and answers, but also be able to hold conversations through context passing to pass information between inputs and also ask for missing information to perform the required operations user is asking for. Below are some use cases to demonstrate WhatBot's capable of.

### 3.2.7.2 Usage scenarios

To try these scenarios yourself, users are required to sign in first. There are two types of users, admin and students, and one of their credentials are:

- Admin:
  - Username: z0000000, Password: 0000000
- Student:
  - Username: z1234567, Password: 123456
  - Username: z8888888, Password: 123456
  - Username: z9999999, Password: 9999999

Some usage cases scenarios are only applicable to admins and some are only applicable to students. Unless specified, the use case is applicable for both type of user.

#### 1. Consultation booking

**Explanation:** User wants to book a course consultation with the lecture in charge for a course but cannot find information on how to contact the lecturer directly. He/She can use our system to make a booking directly. However, user can only make a booking for a date within one week's range.

Assuming today's date is 19/04/2019, when the user does the following, the first booking fails because the date requested is outside the one week range while the second one succeeds as it is within one week's range. The third one fails because booking has already been made for that time and date for that course.

*Conversation-1: booking success*

- User inputs 1:
  - I want to book a course consultation for COMP9801 at 10am on 25/04/2019
- System outputs 1:
  - Your booking is on Wednesday 2019-04-25
  - User also receive a confirmation email in their inbox.

---

*Conversation-2: booking-failed, time too far away*

- User inputs 2:
  - I want to book a course consultation for COMP9801 at 10am on 28/05/2019
- System outputs 2:
  - Sorry you can only booking consultation up to next one week. Your booking date must before 2019-04-26

---

*Conversation-3: booking-failed, time slot already booked*

- User inputs 3:

- I want to book a course consultation for COMP9801 at 10am on 25/04/2019
- System outputs 3:
  - Sorry this time slot has been booked, please choose another one from following time slots on 2019-04-25: 09:00:00, 11:00:00, 12:00:00, 13:00:00, 14:00:00, 15:00:00, 16:00:00, 17:00:00

*Conversation-4: booking with followup context*

*our chatbot surely can build up some knowledge from previous questions and answers the next questions within the same conversation accordingly.*

```
- [user input]: I want to book a course consultation
- [System output]: Sure! What is the course code of the course you would like to
  book it for? Also, what time and date?
- [User input]: Book consultation for COMP9415
- [System output]: Sure! Please tell me which date and time you would like to book
  a course consultation for COMP9415.
- [User input]: 29 April 2019 at 10am please
- [System output]: Your booking has been made on Monday 2019-04-29
```

## 2. See list of consultation bookings made

**Explanation:** User wants to see the list of consultation bookings they have made as they cannot remember what they are.

- User inputs:
  - Show me the list of consultation bookings I've made
- System outputs:

```
The list of consultation bookings you 've made are:
COMP9801 on 2019-04-24 at 10:00:00
COMP9901 on 2019-04-25 at 10:00:00
```

## 3. Consultation cancel

**Explanation:** User made a consultation booking before but want to cancel it now

- User inputs:
  - Cancel my consultation booking for COMP9331 at 11am on 23/04/2019  
(Assuming COMP9331 at 11am on 23/04/2019 is already booked)
- System outputs:
  - You have cancelled the booking at 11:00:00 on 2019-04-23

## 4. Finding out where the classroom for a course is located

**Explanation:** User is not sure where to go for a course and can quickly find out this information through our chatbot.

- User inputs:
  - Where is the classroom for COMP9321
- System outputs:
  - Tyree Energy Technology G16, Kensington



## 5. Finding out the course outline of a course

**Explanation:** User what to find out what a course is about without having to do their own online search, which involves navigating difficult to use handbooks.

The system can also understand context and hold brief conversations when under the situation where user fails to provide complete set of information for a query.

*Conversation-1: (without followup)*

```
- [User]: What is COMP9318 about?

- [Chatbot]:
  Data Warehouse: (a) Data Model for Data Warehouses. (b) Implementing Data
  Warehouses: data extraction, cleansing, transformation and loading, data
  cube computation, materialized view selection, OLAP query processing. Data
  Mining: (a) Fundamentals: data mining process and system architecture,
  relationship with data warehouse and OLAP systems, data pre-processing. (b)
  Mining Techniques and Application: association rules, mining spatial
  databases, mining multimedia databases, web mining, mining sequence and
  time-series data, text mining, etc. The lecture materials will be
  complemented by projects /assignments.
```

*Conversation-2: (with followup feature)*

```
- [User]: I want to see the course outline for a course

- [Chatbot]: Of course! What is the course code of the course you want to know the
  outline for?

- [User]: COMP9414 thanks

- [Chatbot]:
  Overview of Artificial Intelligence. Topics include: the representation of
  knowledge, search techniques, problem solving, machine learning, expert
  systems, natural language understanding, computer vision and an Artificial
  Intelligence programming language (Prolog or LISP). Students may be required
  to submit simple Artificial Intelligence programs, or essays on an aspect of
  A.I, for assessment, in areas such as robotics, computer vision, natural
  language processing, and machine learning.
```

## 6. Finding out course fee for a course

**Explanation:** User wants to plan their budget for next semester but is not sure how much the cost is for a course.

- User inputs:
  - How much does COMP9444 cost?
- System outputs:

```
Commonwealth student: '$1170
```

```
Domestic student: $4350
```

```
International student: $5730
```

## 7. Finding out when a course is offered

**Explanation:** User wants to take a course but is not sure when a course would be offered. They can find this information through our application without having to look through the whole timetable and handbook website.

- User inputs:

```
When is COMP9331 offered?
```

- System outputs:

```
Summer Term, Term 1, Term 2, Term 3
```

## 8. Finding out which school runs a course

**Explanation:** User wants to find out which school manages a course so they can contact them for their own personal queries.

- User inputs:

```
Which school runs COMP9414
```

- System outputs:

```
Faculty of Engineering manages this course.
```

Or

```
This course belongs to Faculty of Engineering
```

Or

```
This course is run by Faculty of Engineering
```

## 9. Admin wants to see the WAM of a particular student

**Explanation:** Admin wants to find out the current grade of a particular student without having to go to the database to search through all that information and manually calculate that student's grades

- User inputs:

1. User is logged on as an admin using the credential provided at introduction section (username: z0000000, password: 0000000)
2. User enters the message "I want to see WAM of z1234567" in the chatroom

- System outputs:

```
Course name: COMP6441
```

```
Number of credits: 6
```

```
Result: 95.0
```

```
Course name: COMP9020
```

```
Number of credits: 6
```

```
Result: 98.0
```

```
Course name: COMP9021
```

```
Number of credits: 6
```

```
Result: 100.0
```

```
Course name: COMP9032
```

```
Number of credits: 6
```

```
Result: 78.0
```

```
Course name: COMP9311
```

```
Number of credits: 6
Result: 88.0

Course name: COMP9414
Number of credits: 6
Result: 66.0

Course name: COMP9814
Number of credits: 6
Result: 79.0

Course name: COMP9511
Number of credits: 6
Result: 85.0

Wam is: 86.1

Grade is: D
```

## 10. Student want to see their own WAM

**Explanation:** Student want to find out how well they are doing in their degree without having to go to myUNSW to download their transcript to see the whole summary.

- User inputs:
  1. User is logged on as a student using the credential provided at introduction section (username: z1234567, password: 123456)
  2. User enters the message “What is my current WAM?” in the chatroom
- System outputs:

```
Course name: COMP6441
Number of credits: 6
Result: 95.0

Course name: COMP9020
Number of credits: 6
Result: 98.0

Course name: COMP9021
Number of credits: 6
Result: 100.0

Course name: COMP9032
Number of credits: 6
Result: 78.0

Course name: COMP9311
Number of credits: 6
Result: 88.0

Course name: COMP9414
Number of credits: 6
Result: 66.0
```

```
Course name: COMP9814
Number of credits: 6
Result: 79.0

Course name: COMP9511
Number of credits: 6
Result: 85.0

Wam is: 86.1

Grade is: D
```

## 11. Request course outline file

**Explanation:** User can ask the system to send the course outline file to them so they can keep it as a record, such that they won't have to go online and search for it everytime. Currently, it is very difficult to find a place to download the pdf file for course outline. Due to lack of real email addresses, all emails are sent to the system's own email, which is "[whatbot9900@gmail.com](mailto:whatbot9900@gmail.com)" but if deployed for real usage, the system will use user's email, which will be available from the database.

- User inputs:  
Could you please send me the course outline for COMP9332?
- System outputs:
  1. Whatbot replies with "The course outline for COMP9332 has been sent to [whatbot9900@gmail.com](mailto:whatbot9900@gmail.com)"
  2. User sees a new mail in their inbox.

## 12. Finding out prerequisites of a course

**Explanation:** User wants to know the prerequisites for a course that they are interested in taking.

*Conversation-1: without followup*

- User inputs:
  - Tell me the prerequisites for COMP9415
- System outputs:
  - Prerequisite: COMP9024.

*Conversation-2: with followup*

```
- [User input]: I want to find out the prerequisite for a course
- [System output]: Sure! Please tell me the course code of the course you want to
  find out prerequisites for.
- [User input]: COMP9444 please
- [System output]: Prerequisite: COMP9024.
```

## 13. Find out if a course is an ADK (Advanced Disciplinary Knowledge) course

**Explanation:** User wants to know if a course is an ADK course as he/she need to set study plan to make sure he/she meets the ADK requirements to graduate.

- User inputs:
  - Is COMP9900 an ADK?
- System outputs:
  - Yes, it is an ADK course

- User inputs:
  - Is COMP9020 ADK?
- System outputs:
  - This course is not an ADK course

#### 14. Finding out list of courses in CSE UNSW

**Explanation:** User wants to know what are the courses available in CSE at UNSW as currently, there is no clear website that shows this.

- User inputs:
  - Show me list of postgrad courses in cse
- System outputs:
  - The list of courses is:

```
COMP6441 COMP9020 COMP9021 COMP9032 COMP9311 COMP9414 COMP9814
COMP6451 COMP6324 COMP9024 COMP9041 COMP9101 COMP9801 COMP9222
COMP9331 COMP9334 COMP9415 COMP9447 COMP4141 COMP4161 COMP6452
COMP6733 COMP6752 COMP6771 COMP6841 COMP9102 COMP9151 COMP9161
COMP9283 COMP9211 COMP9313 COMP9315 COMP9318 COMP9319 COMP9332
COMP9337 COMP9417 COMP9431 COMP9444 COMP9517 COMP4121 COMP6443
COMP6448 COMP6741 COMP6843 COMP6845 COMP9242 COMP9243 COMP9323
COMP9511 COMP4418 COMP9321 COMP9322 COMP6714 COMP6721 COMP9153
COMP9201 COMP9333 COMP9336 COMP6445 COMP6447 COMP9418 COMP9596
COMP9900 COMP9945
```

#### 15. Find out if a course is for postgraduates or undergraduate

**Explanation:** User wants to find out if a course is offered for his/her study level.

- User input:
  - Is COMP6741 for postgraduates?
- System output:
  - COMP6741 is a CSE course for postgraduate

#### 16. User wants to find out the timetable for a course

**Explanation:** User wants to find out the timetable for a course, but going through the timetable website for each course is quite time consuming.

- User inputs:
  - Show me the timetable for COMP9313
- System outputs:

- Timetable for Big Data Management is: Wed 13:00 - 15:00 (Weeks:1-10), Thu 16:00 - 18:00 (Weeks:1-10)

**17. Find out how many hours of classes for a course (data based on indicative hour on handbook)**

**Explanation:** User wants to find out the amount of workload expected for a class, but going through the handbook for many courses can be time consuming.

- User inputs:
  - How many hours of classes are there for COMP9414?
- System outputs:
  - 5

**18. Check latest announcement made by a lecturer for a course**

**Explanation:** User can get the latest news for a course so they can keep up to date is what is happening.

- User inputs:
  - See latest announcement for COMP9414
- System outputs:
  - Latest announcement made on 2019-04-07: We will have a lab next Monday

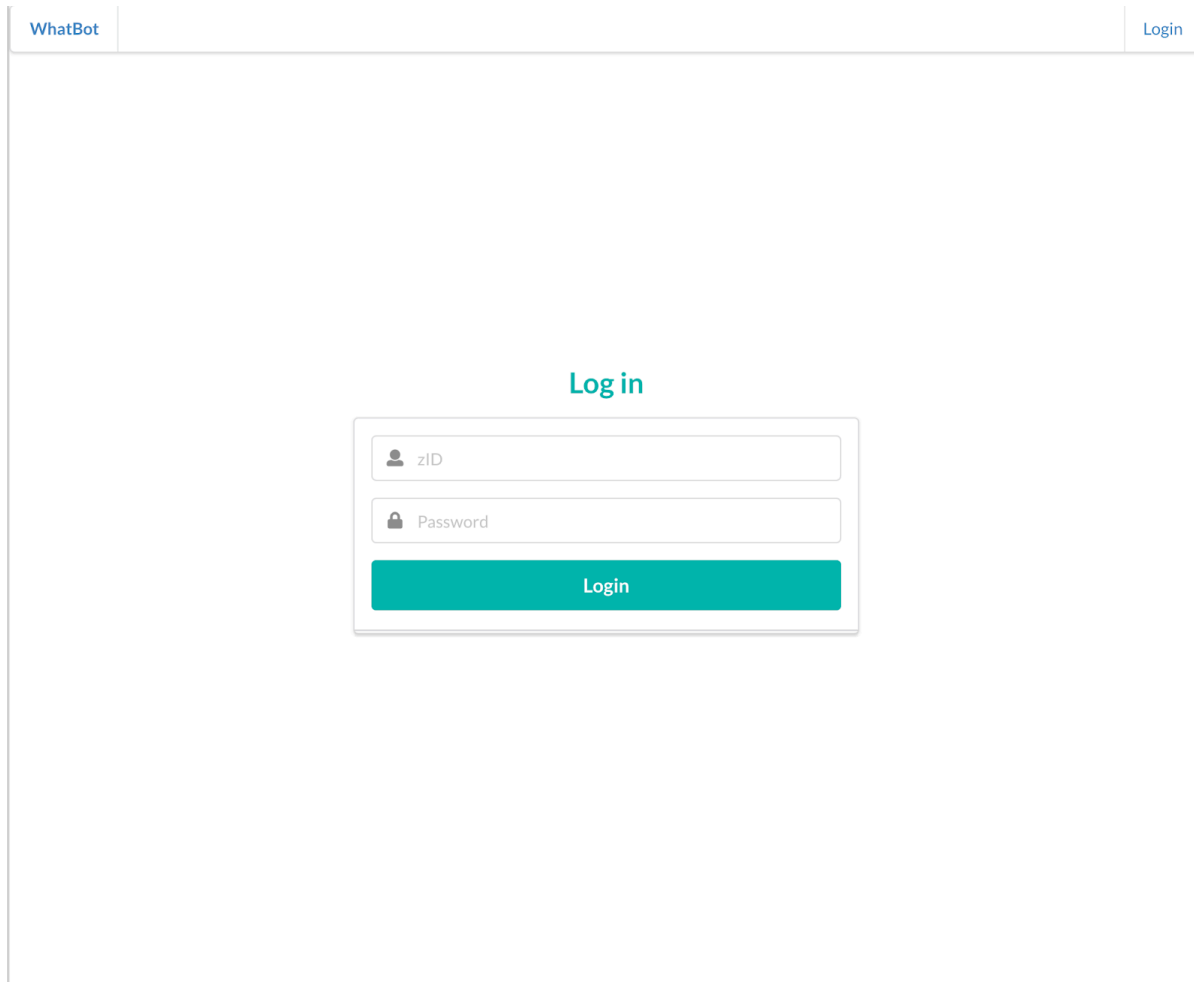
### 3.2.8 User Interface

WhatBot’s user interface is simple and clear. It compose of three main tabs, which are chatroom, file upload, and dashboard, each for a specific feature. The user interface is designed such that people can easily use the software without having to look up at external sources for manuals.

As the system has a client-server architecture, this user interface is completely independent of the backend such that developers who want to use WhatBot’s service can design their own frontend as well while still using WhatBot’s backend features.

#### 3.2.8.1 Design Overview

- When user first open the app, they will be directed to a simple login page:



The screenshot shows a web application interface for 'WhatBot'. At the top, there is a header bar with 'WhatBot' on the left and 'Login' on the right. The main body of the page is white and contains a centered login form. The form has a title 'Log in' in teal. Below the title are two input fields: the first is labeled 'zID' with a person icon, and the second is labeled 'Password' with a lock icon. Below these fields is a teal button labeled 'Login'.

- After logging in, they see WhatBot's main component which follows a simple and clear chatroom design.

WhatBot	Upload training data	Dashboard	Logout
---------	----------------------	-----------	--------

- For student user's only this feature will be available and they will not see the "Upload training data" tab and "Dashboard" shown in the screenshot above. Below are some sample of what a conversation look like:

WhatBot

Sure! What is the course code of the course you would like to book it for? Also, what time and date?

WhatBot

Sure! Please tell me which date and time you would like to book a course consultation for COMP9415.

WhatBot

Your booking has been made on Monday 2019-04-29

You

I want to book a course consultation

You

Book consultation for COMP9415

You

29 April 2019 at 10am please



You

I want to find out the prerequisite for a course

WhatBot

Sure! Please tell me the course code of the course you want to find out prerequisites for.

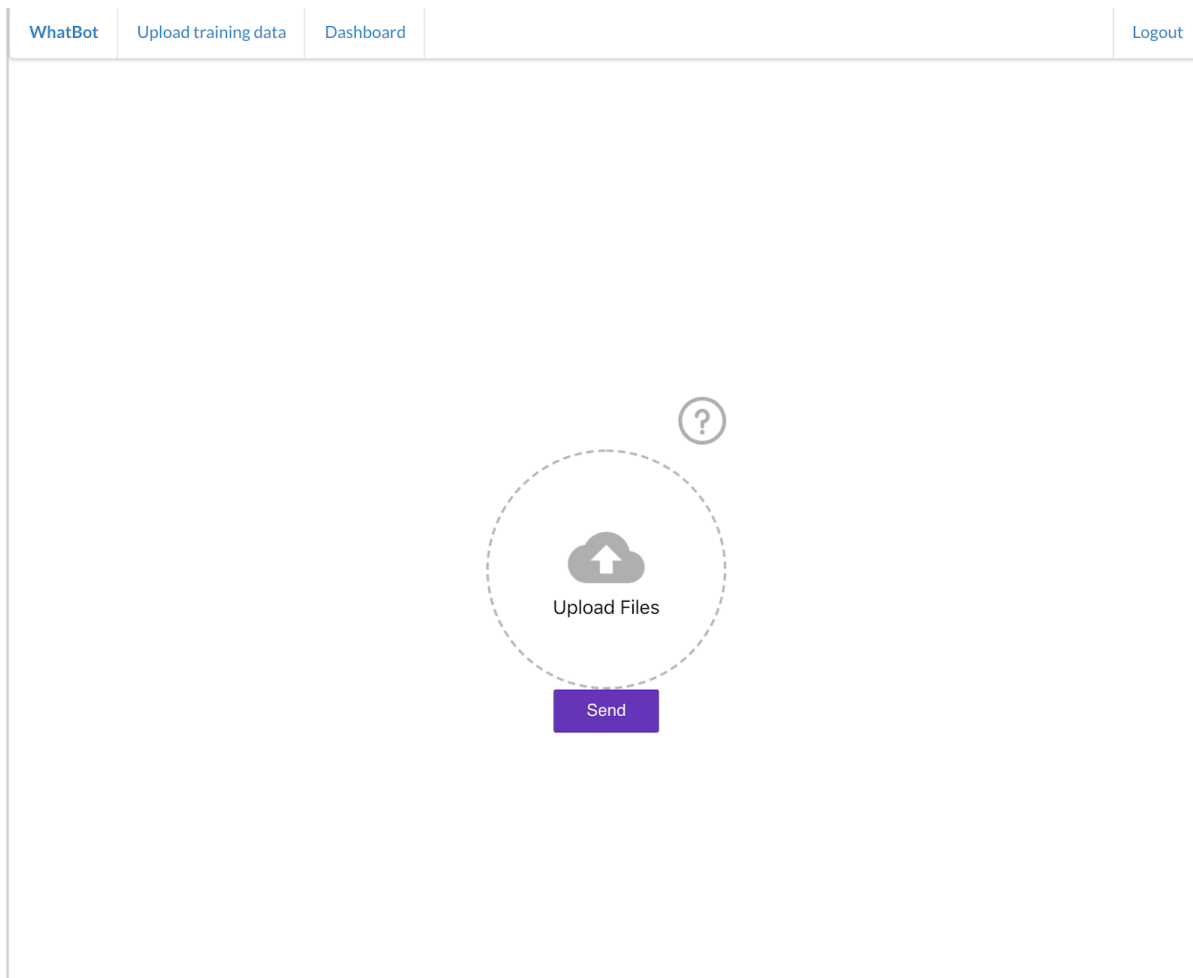
You

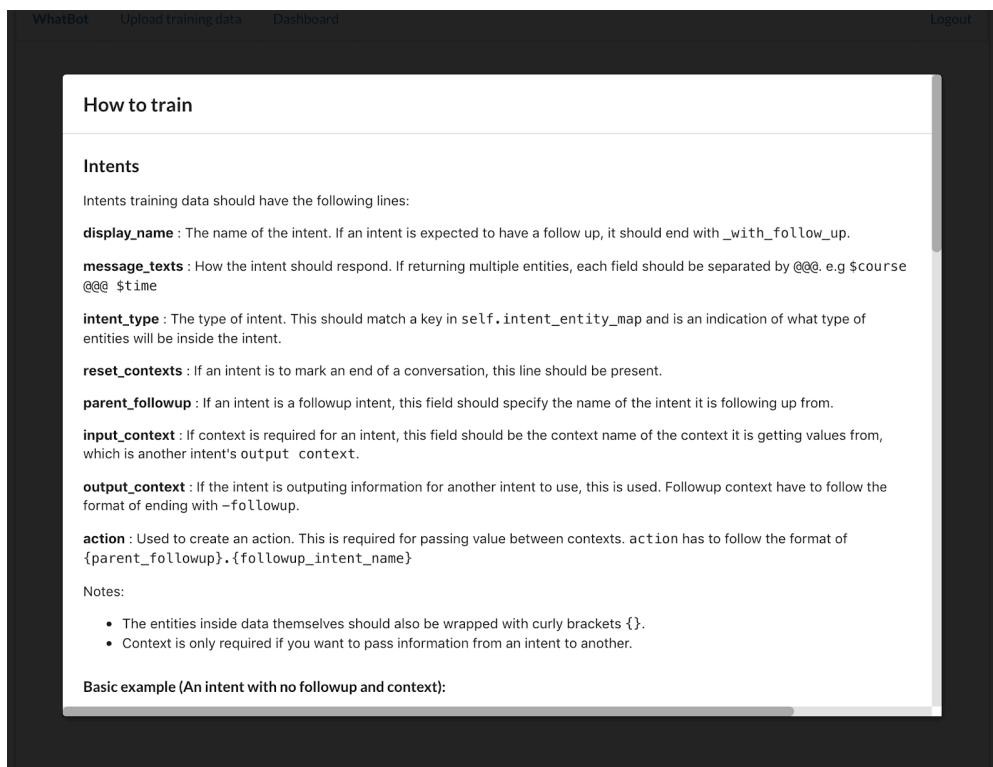
COMP9444 please

WhatBot

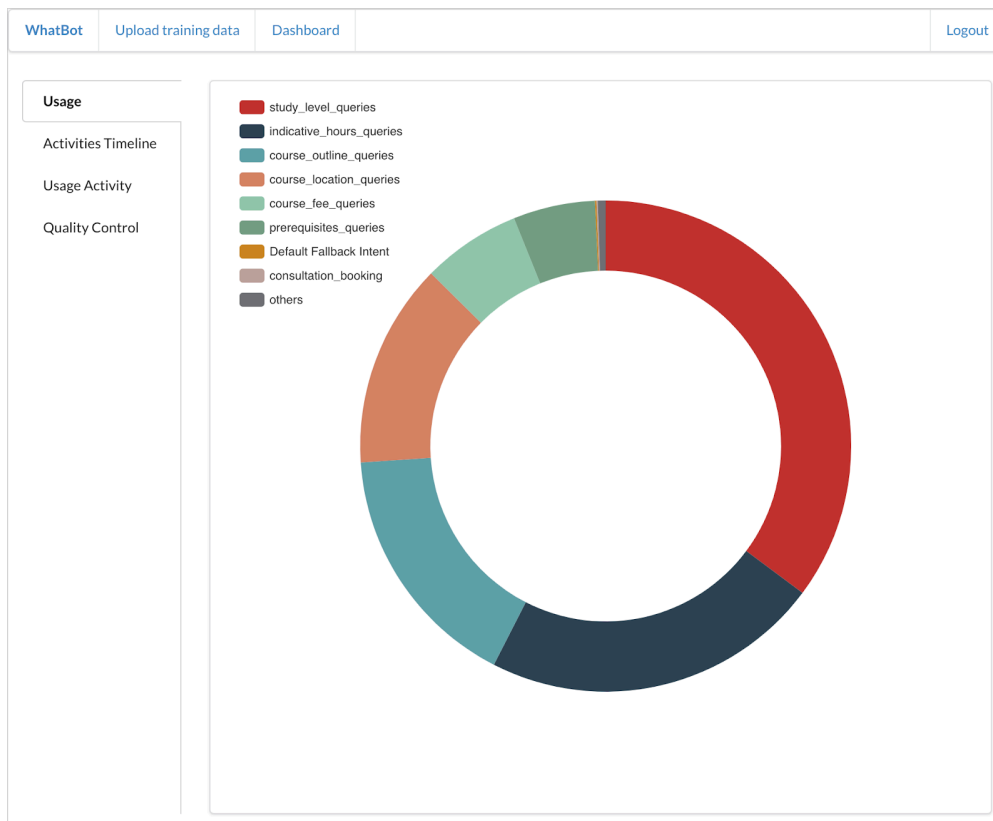
Prerequisite: COMP9024.

- For automation training, admin users will navigate to the “Upload training data” tab.





- This is where automation training can be done and clicking the question mark icon provides a pop up modal with instruction on how to use the feature.
- The dashboard also follow a clear and easy to use design such that users won't be overwhelmed with information.



## 4 Third Party Functionalities Used

This section describes all clouds services, external APIs and libraries used by WhatBot.

### 4.1 Dialogflow

#### 4.1.1 Description and Usage Justification

Dialogflow is Google's NLP based artificial intelligence service. Dialogflow's SDKs and underlying APIs allow users to perform three key functions which are: reacting based on conversational input by providing fulfillment responses and have contexts, intent detection, and dynamically change AI agent's behavior by modifying or adding new intent and entity<sup>9</sup>. WhatBot fully utilise the first two to enhance user interaction with WhatBot's chat system as intent detection is the backbone of WhatBot's intent classification function that allow it understand how to deal with different queries and context passing between messages is how WhatBot enhance user experiences. While fulfillment text responses are what WhatBot uses to extract entities from queries to get keywords for information search and retrieval. The third function, dynamically changing AI agent's behaviour is essentially the core of WhatBot's automation training feature.

Google's Dialogflow was chosen instead of other similar conversational NLP service like Wit.ai owned by Facebook<sup>10</sup> because Dialogflow provided more functionality for developers to add new behaviours, maintain conversation flow, manage context behaviour and perform entity labelling through code<sup>11</sup>. Overall, it was determined that Dialogflow was easier to use and more powerful at the same time.

#### 4.1.2 Licensing Impact

Dialogflow's library has an Apache 2.0 License<sup>12</sup>. This indicate that the library little impact on this project because Apache 2.0 License allow us to use the library for the following purposes<sup>13</sup>:

- Commercial Use
- Modify
- Distribute
- Sublicense
- Private Use
- Use Patent Claims
- Place Warranty

However, we must include notice for:

- Copyright
- Apache License usage
- Any code modification

Since the must include legal requirements have no impact on the lifetime or the use of our software. There is little impact of using Dialogflow library in our software and we are free to turn it into a commercial software.

---

<sup>9</sup>(n.d.). Dialogflow. Retrieved April 26, 2019, from <https://dialogflow.com/doc/sdks>

<sup>10</sup>(2015, January 5). Facebook Acquires Wit.ai To Help Its Developers With ... - TechCrunch. Retrieved April 26, 2019, from <https://techcrunch.com/2015/01/05/facebook-wit-ai/>

<sup>11</sup>(n.d.). How is DialogFlow different from Wit.ai? - Quora. Retrieved April 26, 2019, from <https://www.quora.com/How-is-DialogFlow-different-from-Wit-ai>

<sup>12</sup>(n.d.). GitHub - googleapis/dialogflow-python-client-v2: Python client for .... Retrieved April 26, 2019, from <https://github.com/googleapis/dialogflow-python-client-v2>

<sup>13</sup>(2013, November 25). Apache License 2.0 (Apache-2.0) Explained in Plain ... - TLDRLegal. Retrieved April 26, 2019, from [https://tldrlegal.com/license/apache-license-2.0-\(apache-2.0\)](https://tldrlegal.com/license/apache-license-2.0-(apache-2.0))

## 4.2 ECharts.js

### 4.2.1 Description and Usage Justification

Echart.js is an open source graphing library developed by Chinese Technology Giant, Baidu<sup>14</sup>. It provides APIs for a wide variety of graphs and is the backbone of WhatBot's data analytic dashboard.

Echart.js was chosen as the visualisation library instead of other well known libraries like D3.js because it was a library that our team member has experience with and it had a smaller learning curve as well.

### 4.2.2 Licensing Impact

Echart.js also has Apache 2.0 license<sup>15</sup>. Therefore, similar to Dialogflow, uses of this library has little impact on our software as long as we do what Apache 2.0 license require us to do and these requirements were already mentioned in Dialogflow's Licensing Impact section.

## 4.3 AWS

### 4.3.1 Description and Usage Justification

Amazon Web Service, AWS, was the chosen cloud service platform to host WhatBot's database and file storage. It provides cloud computing for the software to access storage, databases and a broad set of application services over the Internet.

WhatBot uses AWS RDS to host its knowledge base which is a PostgreSQL database and AWS S3 for file storage to store training data for automation training. As you can see, these two components both play an important role in WhatBot's core functions.

AWS was chosen over other cloud service, like Google Cloud Platform (GCP), for hosting database and file storage because it was a platform the team member were more familiar with and also AWS was believed to be easier to use and configure compared to GCP.

### 4.3.2 Licensing Impact

As AWS is a commercial cloud platform, there is no licensing issue with using its service. However, due to it being a commercial product, if WhatBot is to be scaled up to serve many more users, we will need to consider the cost for our data and file storages.

Furthermore, the AWS S3 SDK, boto, and PostgreSQL Python library, psycopg2, used inside our code are under Apache 2.0 license<sup>16</sup> and BSD license<sup>17</sup> respectively. Both of them are free license that allow commercial use so there is little impact on using them in our code.

## 4.4 Heroku

### 4.4.1 Description and Usage Justification

Heroku is a cloud platform for building, scaling and running apps. It is an AWS based service but provides well configured infrastructure for user to use so users don't need to spend too much effort on setting up servers and other

---

<sup>14</sup>(n.d.). ECharts ·About. Retrieved April 26, 2019, from <https://echarts.baidu.com/echarts2/doc/about-en.html>

<sup>15</sup>(n.d.). License - ECharts - The Apache Software Foundation!. Retrieved April 26, 2019, from <https://echarts.incubator.apache.org/license.html>

<sup>16</sup>(n.d.). GitHub - boto/boto3: AWS SDK for Python. Retrieved April 26, 2019, from <https://github.com/boto/boto3>

<sup>17</sup>(n.d.). psycopg/psycopg2 - GitHub. Retrieved April 26, 2019, from <https://github.com/psycopg/psycopg2>

infrastructure things<sup>18</sup>. It is for this reason, the team decided to use Heroku instead of AWS EC2 for deploying and running WhatBot as this saved significantly good amount of time. Furthermore, Heroku integrated well with Docker containers as it has its own Docker registry<sup>19</sup> which eased the deployment process significantly as Docker is one of the main tool the team has decided to use for deployment.

#### 4.4.2 Licensing Impact

As Heroku is a commercial cloud service, there is no direct impact on our code. However, similar to all cloud services, if we choose to scale up our application, we will need to evaluate the cost of doing so. Currently, the free tier only provides 500 hour of run time per month<sup>20</sup> such that if WhatBot becomes a very popular software that is up and running all the time, we will need to decide on some sort of business model to have some form of cash flow to even out the server hosting cost.

## 5 Development Methodology and Software Implementation Challenges

### 5.1 Development Methodology

Agile software development practices are used by our team with fortnightly sprints and stand-ups every two days. Github project board and issues were used to track tasks and progresses.

#### 5.1.1 Continuous Integration and Continuous Delivery (CI/CD) Infrastructure

##### 5.1.1.1 Why we need this?

CI/CD was setup to ensure software quality and help deployment be made easy. It drive development teams to implement small changes and check in code to version control repositories frequently. It aims to establish a consistent and automated way to build, test code and deploy application. With this consistency team members can commit more code changes more frequently, which leads to better collaboration and software quality<sup>21</sup>.

For example, a team member has made some changes to a part of a code or refactored. However, these changes may have potentially broke another feature or change its behaviours somehow. Typically, the team member will have to manually build the software to run the unit tests and integration tests to ensure the new changes did not induce any unexpected behaviour or bugs into the software. This process is time consuming and inefficient. As a result, CI/CD move these repetitive but important tasks to a remote build machine so the team member can continue working on other tasks.

##### 5.1.1.2 Our CI/CD Infrastructure

Team WhatBot fully utilized the power of Docker and CircleCI for our CI/CD infrastructure. An account was setup on CircleCI such that every time a commit to a pull request or master branch was made, the new changes will be tested against the unit tests we have setup, which are located in frontend/test and backend/tests. Below are some of the screen shots of changes that passed and failed the unit tests, where successful changes lead to deployment.

---

<sup>18</sup>(n.d.). What is Heroku | Heroku. Retrieved April 26, 2019, from <https://www.heroku.com/what>

<sup>19</sup>(2019, April 24). Container Registry & Runtime (Docker Deploys) | Heroku Dev Center. Retrieved April 26, 2019, from <https://devcenter.heroku.com/articles/container-registry-and-runtime>

<sup>20</sup>(n.d.). Pricing | Heroku. Retrieved April 26, 2019, from <https://www.heroku.com/pricing>

<sup>21</sup>(2018, May 10). CI/CD? Continuous integration and continuous delivery ... - InfoWorld. Retrieved April 27, 2019, from <https://www.infoworld.com/article/3271126/what-is-cicd-continuous-integration-and-continuous-delivery-explained.html>

✓ SUCCESS	#48 Dev/fix extractor (#58)	workflow build
! FAILED	#45 Use setup build to try to fix import errors.... (#45)	workflow build
! FAILED	#38 Integrate query and response fallback part (#49)	workflow build
✓ SUCCESS	#31 query followup and context mess... (#48)	workflow build
! FAILED	#19 configured to stop sql injection (#47)	workflow build
✓ SUCCESS	#15 CI/CD pipeline integration (#46)	workflow build
✓ SUCCESS	#494 consultation test fix and consultation cancel intent data (#11...	build_and_test frontend_build_and_test
✓ SUCCESS	#493 consultation test fix and consultation cancel intent data (#11...	build_and_test backend_build_and_test
✓ SUCCESS	#470 upload and training automation made available	build_and_test deploy
✓ SUCCESS	#469 upload and training automation made available	build_and_test backend_build_and_test
✓ SUCCESS	#468 upload and training automation made available	build_and_test frontend_build_and_test
✓ SUCCESS	#465 improve consultation test code quality	build_and_test deploy
✓ SUCCESS	#464 improve consultation test code quality	build_and_test frontend_build_and_test
✓ SUCCESS	#463 improve consultation test code quality	build_and_test backend_build_and_test

To run the tests for the whole software, we simulate how the live application will run by spinning up two Docker container, one for frontend and another for backend. Inside these two containers, we run their respective unit tests. Containers allow us to package up the application with all modules, libraries and dependencies, and deploy it all out as one package<sup>22</sup>. If both test passes and the pull request is merged into master, we go into the deploy stage where we also spin up two Docker containers with nginx web server inside to run the components in production mode then push the two containers onto Heroku.

<sup>22</sup>(n.d.). What is Docker? | Opensource.com. Retrieved April 27, 2019, from <https://opensource.com/resources/what-docker>

## 5.2 Implementation Challenges

### 5.2.1 Sparse and Unclean Data Source

In the project proposal, it was proposed that we will make WhatBot available for all courses in UNSW. However, as the project progressed and we gained more understanding of the features we are implementing and their data source requirements, it became difficult to maintain the idea of doing all courses in UNSW for the following reasons:

- Data source content from different school and faculty varies.
- Data source location for different things like handbook info and timetable are too spread out for different school and faculty.
- Cleaning and filtering out noise from data is a very time consuming task.

As a result, the team decided to reduce the scope down to postgraduate CSE courses in UNSW as our aim was to give a proof of concept of our vision for WhatBot.

### 5.2.2 Dialogflow NLP AI Model Accuracy and Agent Training

As our software relied on Dialogflow to handle queries and intent classification, it was important that Dialogflow performs well. However, it soon became evident that not knowing how to use its SDK for development would impose a bottleneck for scaling up the software. We needed a way to be able to train intent in a fast and accurate way. Having developers to manually do them on the Dialogflow console was an exhaustive process. Furthermore, it is difficult to train accurate AI with just few training phrases as well and having develop manually entering hundreds and thousands of training phrases is unrealistic. As a result, we studied extensively on how to use Dialogflow SDK and it eventually turned into our automation training feature and this feature was not mentioned in our initial proposal. Therefore, from this we learnt how features can be come up with during development.

### 5.2.3 Fallbacks and Error Handling

In the initial project proposal, it was proposed that upon not understanding user's queries, when intent classification detects a fallback intent, the system will first aim to perform online search for result for information or give user options on choosing what they want to do based on what our AI understands, but as the project progressed, it was decided that this feature is too difficult to implement in the given amount of time.

This feature would require the server to keep a state of what is happening with each user and what their last queries are to be able to provide options for user to clarify their queries. Keeping state on server side is an extremely difficult tasks and is almost never done any server application. Quite often, the user can change their mind between queries such that two inputs may have no relationship with each other and this is difficult for a normal server application to recognize. As a result, the team decided not to implement this feature ourselves and rely on Dialogflow's AI which understands and can distinguish context between conversations.

Performing Google search as fallback was also not implemented since the challenge with doing so is that the quality of the result is quite often inconsistent as part of Google's search engine utilizes user profiles to identify what the user is most likely looking for<sup>23</sup>. Therefore, it was decided it would be too difficult to get consistent results that the user will be satisfied with. As a result, the team decided to reply on Dialogflow fallbacks to ask user to be more specific with their queries for handling the situation where Dialogflow's AI agent did not understand the query.

### 5.2.4 Architecture And Infrastructure Challenges

Our choice of CI/CD platform and deployment platform, CircleCI and Heroku respectively, required significant amount of configuration to work together as CircleCI is a relatively new platform that did not support deployment

---

<sup>23</sup>(2015, September 29). 7 Reasons Google Search Results Vary Dramatically. Retrieved April 27, 2019, from <https://www.webpresencesolutions.net/7-reasons-google-search-results-vary-dramatically/>

to Heroku using Docker well. As CircleCI build containers are based on their own pre-built images, it allowed less flexibility for user to customize their own containers<sup>24</sup> and to use Heroku's services, we required Heroku CLI to be installed inside the container which did not work well<sup>25</sup>.

As a result, we had to spend quite a lot of effort into working out a script that help us install Heroku CLI into CircleCI containers for automatic deployment to work. The solution we produced can be seen in `setup_heroku.sh` file in the main project repository.

Furthermore, due to our client-server architecture, the ideal way to deploy would have been to use Docker-Compose to launch both containers into the same network into one application on Heroku. However, it was later discovered that Heroku does not support Docker-Compose<sup>26</sup>, the team had to change the deployment strategy to deploy frontend and backend separately into two different applications. A downside of this is that it is more expensive to run and introduce more latency between frontend and backend communication. On the other hand, an upside is that backend essentially becomes an API server that not only our frontend will be using.

---

<sup>24</sup>(n.d.). Pre-Built CircleCI Docker Images - CircleCI. Retrieved April 27, 2019, from <https://circleci.com/docs/2.0/circleci-images/>

<sup>25</sup>(2018, October 29). Deployment on CircleCI + Docker + Heroku does not work - Bug .... Retrieved April 27, 2019, from <https://discuss.circleci.com/t/deployment-on-circleci-docker-heroku-does-not-work/26106>

<sup>26</sup>(n.d.). How to push Docker containers managed by Docker-compose to Heroku .... Retrieved April 27, 2019, from <https://stackoverflow.com/questions/46904060/how-to-push-docker-containers-managed-by-docker-compose-to-heroku>



## 6 User Documentation and Manual

### 6.1 For Developers

This section is for developers who are interested in seeing how WhatBot works and potentially fork the project for their own development. First clone the repository:

```
git clone https://github.com/comp3300-comp9900-term-1-2019/capstone-project-whatbot.git
```

#### 6.1.1 Running the main software

##### 6.1.1.1 Running locally

To make running the whole software simple and easy, a Makefile has been created for this purpose. In the terminal, go to the main project repository and run the Makefile with:

```
make
```

and you should see all the dependencies being installed and the whole app running afterwards.

Alternatively, developers can run the components manually separately with the following steps:

- Launching backend:

```
# Prerequisites: Python 3 installed  
# Inside the backend directory:  
# Setup and install all the dependencies  
python3 -m pip install -r requirements.txt  
python3 setup.py install  
  
# Run  
python3 app.py
```

- Launching frontend:

```
# Prerequisites: npm installed  
# Inside the frontend directory:  
# Setup and install dependencies  
npm install  
  
# Run  
npm start
```

##### 6.1.1.2 Running in Production

Docker files has being setup for the software to run in production mode on nginx web servers. You will need to have a cloud service, like Heroku or AWS EC2, setup first to run the docker containers. In this section, we will provide instruction on how to deploy with Heroku.

To manually deploy to Heroku, you need to have Heroku CLI and Docker installed.

Install Heroku CLI here: <https://devcenter.heroku.com/articles/heroku-cli>.

Install Docker here: <https://docs.docker.com/install/>.

After you have installed both of them, you need to first create two apps, one for frontend, and another for backend.

After you have done so, with your Heroku credentials:

- Log into Heroku CLI:

```
heroku login
```

- Log into Heroku container registry with:

```
heroku container:login
```

- Deploy frontend

```
cd frontend
```

```
heroku container:push web -a {your frontend app name here}
```

```
heroku container:release web -a {your frontend app name here}
```

- Deploy backend

```
cd backend
```

```
heroku container:push web -a {your backend app name here}
```

```
heroku container:release web -a {your backend app name here}
```

### 6.1.2 How to train a new intent using command line

The QueryTrainer class in train.py is responsible for AI model training. This file allows user to train new intent by simply providing some training data which follows the configuration in the instruction below. Through this feature, even when user only provide little training data, the QueryTrainer can still automatically generate variations of the training data to increase number of training phrases. More detail on the power of this feature in ManagementModule's automatic AI training section.

#### 6.1.2.1 How to train

The Dialogflow agent can be trained by putting training data inside training\_data directory inside query\_module directory. So go into the query\_module directory first. Also, make sure you have Python and Dialogflow library installed first. You can do this by going to the backend directory and running: `pip install -r requirements.txt`

To train an entity:

This will retrain all the entities with data inside training\_data/entities

```
python3 train.py --retrain_entities True
```

Alternatively, to train a specific intent, you can modify the “For development use” section of the code at the end of train.py to read the file where your entity training data is located and train the entity. For example:

```
display_name, entity_values, synonyms =  
    query_module_trainer.read_entities_data({Path to training data})  
  
query_module_trainer.create_entity(self, display_name=display_name,  
    entity_values=entity_values, synonyms=synonyms)
```

You can import the QueryTrainer class into your own code and call the functions as shown in the example above.

Note: Will throw an error if entity already exist, so make sure you call `query_module_trainer.delete_entity(display_name)`

To train an intent:

This will retrain all the intents with data inside training\_data/intents

```
python3 train.py --retrain_intents True
```

Alternatively, similar to retraining a specific entity, to train a single intent:

```
display_name, message_texts, intent_types, data =  
    query_module_trainer.read_intents_data({Path to training data})  
  
query_module_trainer.create_intent(display_name=display_name,  
    message_texts=message_texts, intent_types=intent_types, training_data=data,  
    data_is_parsed=True)
```

**You can import the QueryTrainer class into your own code and call the functions as shown in the example above.**

Note: Will throw an error if entity already exist, so make sure you call `query__module__trainer.delete__intent(display__name)`

To train the whole agent completely:

```
python3 train.py --retrain_all True
```

## 6.2 Training Data Configuration

### 6.2.1 Intents

Intents training data should have the following lines:

- **display\_name** : The name of the intent. If an intent is expected to have a follow up, it should end with `_with_follow_up`.
- **message\_texts** : How the intent should respond. If returning multiple entities, each field should be separated by `@@@`. e.g `$course @@@ $time`
- **intent\_type** : The type of intent. This should match a key in `self.intent_entity_map` and is an indication of what type of entities will be inside the intent.
- **reset\_contexts**: If an intent is to mark an end of a conversation, this line should be present.
- **parent\_followup**: If an intent is a followup intent, this field should specify the name of the intent it is following up from.
- **input\_context**: If context is required for an intent, this field should be the context name of the context it is getting values from, which is another intent's output context.
- **output\_context**: If the intent is outputting information for another intent to use, this is used. Followup context have to follow the format of ending with `-followup`.
- **action**: Used to create an action. This is required for passing value between contexts. action has to follow the format of `{parent_followup}.{followup_intent_name}`

Notes:

- The entities inside data themselves should also be wrapped with curly brackets `{}`.
- Context is only required if you want to pass information from an intent to another.

Basic example (An intent with no followup and context):

```
display_name course_fee_queries  
  
message_texts $course  
  
intent_types course_fee
```

```
reset_contexts
```

```
How much does {course code} cost?
```

```
How much does {course code} cost to do?
```

```
What is the cost of {course code}?
```

Advance example (An intent with followup but no context):

- Parent intent

```
display_name course_fee_queries_with_followup
```

```
message_texts Sure! What is the course code of the course you want to know course  
fee for?
```

```
intent_types course_fee
```

```
output_context course_fee_queries_with_followup-followup
```

```
I want to find out the course fee for a course
```

```
I have a question about course fee
```

```
I have a query regarding course fee
```

```
Course fee question
```

```
I want to find information about course fee
```

```
I want information for course fee
```

```
Information for course fee
```

```
Information for course fee please
```

```
Information for course fee thanks
```

```
Find information about course fee
```

```
Course fee info
```

```
Course fee info please
```

```
Course fee info thanks
```

- Followup intent

```
display_name course_fee_queries_with_followup-user_input_course_code
```

```
message_texts $course
```

```

intent_types course_fee

parent_followup course_fee_queries_with_followup

input_context course_fee_queries_with_followup-followup

reset_contexts

{course code}

{course code} please

{course code} thanks

I want to get the course fee for {course code}

I want to get the course fee for {course code} thanks

I want to get the course fee for {course code} please

course fee for {course code} please

course fee for {course code} thanks

```

Advance example (An intent with followup and context):

- Parent intent to trigger context (May not be necessary depending on your usage case, but this example aims to show a case where you want to just output an context initially in an intent).

```

display_name consultation_booking_with_followup

message_texts Sure! What is the course code of the course you would like to book it
for? Also, what time and date?

intent_types consultation_booking

output_context consultation_booking_with_followup-followup

I want to book a consultation

I want to book a course consultation

Book course consultation

Make a course consultation booking

Book a course consultation

```

- Followup intent that takes in the context to recognize the situation and output the information it gets

```

display_name consultation_booking_with_followup-user_input_course_code_with_followup

message_texts Sure! Please tell me which date and time you would like to book a
course consultation for $course.

```

```

intent_types consultation_booking

parent_followup consultation_booking_with_followup

input_context consultation_booking_with_followup-followup

output_context
    consultation_booking_with_followup-user_input_course_code_with_followup-followup

action
    consultation_booking_with_followup.consultation_booking_with_followup-user_course_code

{course code}

{course code} please

{course code} thanks

I want to book {course code}

```

- Final followup that retrieves missing information that user didn't provide and also gather information from the previous followup's context

```

display_name
    consultation_booking_with_followup-user_input_course_code_with_followup-user_input_time

message_texts
    #consultation_booking_with_followup-user_input_course_code_with_followup-followup.course_code
    @@@ $time @@@ $date

intent_types consultation_booking

parent_followup
    consultation_booking_with_followup-user_input_course_code_with_followup

input_context
    consultation_booking_with_followup-user_input_course_code_with_followup-followup

action
    consultation_booking_with_followup.consultation_booking_with_followup-user_course_code

reset_contexts

I want to book on {date} at {time}

{time} {date}

```

### 6.2.2 Entities

The first line of any entities training data should be the name of the entity and starts with `display_name`. All following lines are the data themselves. If an entity has a synonym, it should be separated with `@@@` on the same

line. Entity training data files should have the following lines:

`display_name:` name of the entity

For example:

```
display_name course  
  
COMP6441@@@Security Engineering and Cyber Security  
  
COMP9020@@@Foundations of Computer Science  
  
COMP9021@@@Principles of Programming  
  
COMP9032@@@Microprocessors and Interfacing  
  
COMP9311@@@Database Systems
```

## 6.3 Using the Software (For typical user)

### 6.3.1 Admin Users

To access admin user features, you need to be first be logged in as an admin user. One of the account for admin user is:

- Username: z0000000
- Password: 0000000

#### 6.3.1.1 How to train a new intent or entity through web interface

1. First create a training data file for an intent or entity following the format described in the “Training data configuration” section above.
2. Go to the “Upload Training Data” tab
3. User should see a screen similar to below



4. Click “Upload Files” icon, as indicated by 1 in the above diagram, to select the file they want to upload. And example of a file content looks like this:

```
display_name rating_queries
message_texts $course
intent_types course_code
reset_contexts
Can you please tell me if {course code} is a good course?
Is {course code} a good course?
Is {course code} a good course
What is the rating for {course code}?
Can you tell me if {course code} is a good course?
Tell me if {course code} is a good course
{course code} a good course?
What is {course code}'s rating?
What do you think about {course code}?
Rating of {course code}
{course code}'s rating
Tell me if {course code} is a good or bad course
How good is {course code}?
Tell me if {course code} is a good or not
```

5. Click “Send” to upload as indicated by 2 in the above diagram
6. After a few minute (depending on size of Dialogflow Agent and training data file), user will see WhatBot being able to handle new intents.
7. If user is not sure on how to configure the training data file, they have access to information on “How to configure training data file” through the question mark icon indicated by 3. The information provided through there is similar to the “Training data configuration” section of this report.

#### 6.3.1.2 How to see system feedback through data analytics dashboard

1. Click the “Dashboard” tab
2. On the left bar, click the tab you are interested in viewing and it’s graph will appear.

#### 6.3.2 Student Users

Student users only have access to the chatroom which handles all of their queries and allow them to book course consultation, see their course consultation booking times and also cancel a course consultation booking. They can user all these features by simply typing their queries.

## 7 Conclusion

In conclusion, WhatBot is a software developed by following good software design patterns and principles. It is a software that is easily extensible and maintainable as it is developed with good testing and delivery infrastructure setup already. its automation AI training feature that offers users limitless potential, such that it does not have to just be a student-tutor support system, and they can configure it to however they want WhatBot makes it a truly unique software.