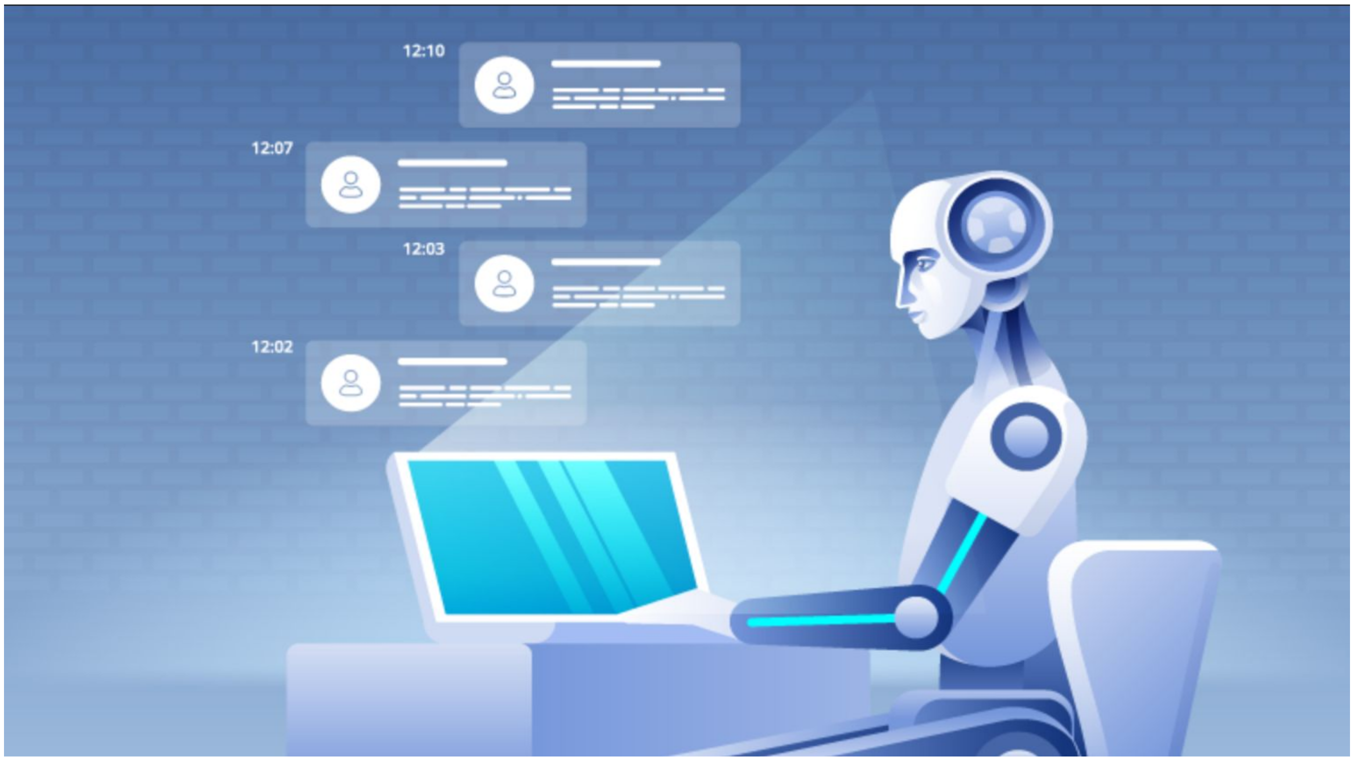


COMP9900 Project Proposal: WhatBot

A support tutor chatbot that scales up a small teaching team to serve thousands of student!

Submission Date: 08.03.2019

Team: WhatBot
Scrum Master: Wenxiang Tang - z5002318
Developer: Kuan-Chun Hwang - z5175539
Developer: Zhuowen Deng -z5137895
Developer: Mingyue Wei - z5129269

Background

Customer support is the provision of service to customers before, during and after a purchase. However, today customer support has much broader definition. Why do we need a service or support? If a client has some issues using the product, where are they expected to consult? Everyone is familiar with the frustration of waiting hours for an email reply or being put on hold for over 30 minutes when all you want is to get an answer to a simple question about a product or service.

With the development of AI, the internet makes the problem more extreme, but it also offers new opportunities for solutions. Though it took almost 100 years for phones to become one of the main customer service channels. Call centers, email, live chat, sophisticated CRM systems and social media have all become common for customer support. Customer service interactions have taken one step forward with the proliferation of social media. The most customer-oriented business started using Facebook, Instagram and Twitter to connect with their customers in a personal and public manner.

This surge in popularity to using internet to provide fast support without having the user to wait on the phone or go to the store themselves have revolutionised how customer services are provided these day. However, this sudden load has also increased the need for automated customer support as many companies simply do not have the capacity to provide 24 hours, 7 days instant support. As a result, chatbots have become increasingly more popular in recent years with the rise of machine learning and NLP. However, while they can provide some basic functions, they simply do not have enough capabilities right now to provide sufficient support to replace human customer service as many of them still have issues handling common queries.

Student support service is no exception to the customer service problem. With attending university becoming the norm of the society today, student support services are already showing signs that the workload is too much with slow and uninformative responses. Part of this issue is contributed by the fact that information for courses are not in a centralised location. While there are many different website resources for the separate functions. They are not consistent enough for the students to access the services easily, especially for the new students. Therefore, we need to merge several applications and information sources into one system. For example, if someone wants to get the outline or basic information related to a specific course, he/she has to go to one of the website, and then if he/she want to find class time table, he/she will have to go to another website. In addition, if the student also wants to book a consultation timeslot, he or she has to get into the student centre service. These processes are too inefficient and contribute to the student support service issue.

Aim

Our team aims to produce 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. For example, considering a course like *Introduction to Programming by Python*, which is open for whole university and available online for MOOC enrollment, it will approximately at least serves 400 in class students and 1000 online MOOC students. Except for that, there might be more than 10000 students interested in taking the class and will ask pre-enrolment consultation. This will produce huge workload to a small teaching team. With a support chatbot system act like a *firewall*, the teaching team can improve service quality to the new height, comparing to serve traditional static web pages.

In terms of our project, we are going to implement a dialog framework for conversational agents for the student-tutor support system. For example, for beginning of every semester, student might ask similar questions, such as, what is the outline of the course, are there any available spaces for the course, where is the lecture room of this course, etc. If every student comes to the student office or emails them to tutors, the process will definitely inefficient. However, if we have an intellectual chatbot for students and tutor, it will dramatically enhance the work efficiency. Our chatbot will cover almost all questions related to all content from a specific course. What’s more, tutors will be able to get the data analyzed by our software based on what students ask and the chatbot will also provide functionality such as consultation booking.

We will create a modern, user friendly and effective student tutor support system for the education assistance that aims to solve the student support service problem. The software will provide a high level of utility such that it not only benefit students but also the teaching staffs.

Technical Scope of the Project

After properly understanding the requirement of the project, the team has decided the application will be a web chatbot application, which will act as a student-tutor support system. The chatbot will be easily configurable by teachers such that teachers can upload relevant information about a course for the chatbot to learn and perform its functions. This flexibility in configuration will make the application more maintainable and powerful because course contents are constantly being updated and lecturer themselves can provide the most accurate information that they think are about a course. Furthermore, this also increases the scalability of the application since lecturer will be able to configure it to any course and the chatbot software will work.

While Natural Language Processing, NLP, is expected to be crucial on how well the chatbot can learn a course content and understand queries, the team has currently decided to leave implemented our own NLP model out of the scope and rely on Dialogflow's service because NLP is a complex topic and creating our own implementation may not be possible given the amount of time we have for the project. Furthermore, for this project, we will focus on text inputs and not voice inputs as the team has decided that the complexity of voice input is too hard to solve with the given time we have.

Furthermore, the application, WhatBot, is expected to be able to handle high volume of users, which can occur during enrollment periods when students have many questions about the courses that they can choose. Therefore, WhatBot is expected to offer basic information such as course outlines, LiC and prerequisites for a course. WhatBot will also provide data analytic features to teaching staff on query statistics so that they can use them to improve their course website etc.

As the software is a real-time interaction application, we will need to make sure our APIs have low latency and are efficient so that response time is low. This will be a major factor on the overall user experience so code efficiency and quality will be crucial. To ensure high quality code, the team has decided to build well tested CI/CD pipeline and code review will be conducted for every pull request.

Team Skill Set

Languages: Python, Java, C, JavaScript, HTML/CSS

Frontend Frameworks: React, Express, Angular

Backend Frameworks: Flask, Django, Node

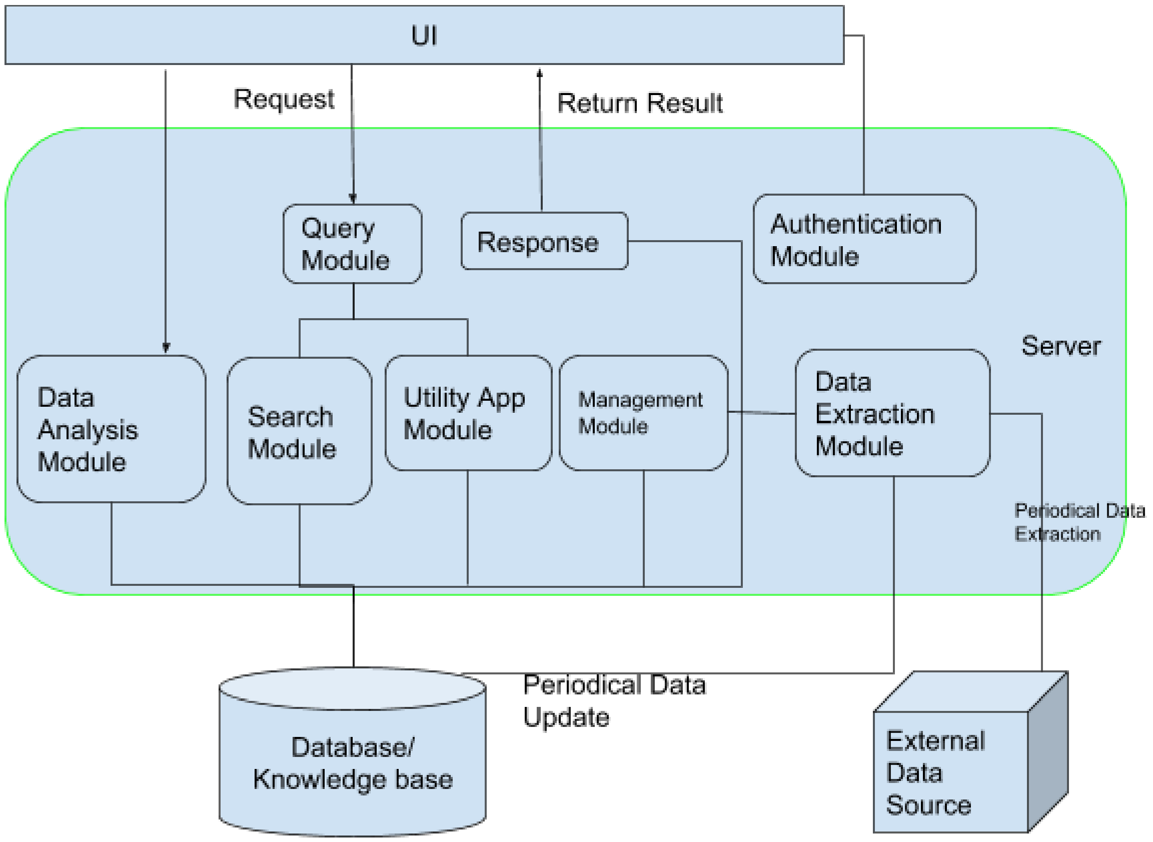
DevOps: Git, Jenkins, Docker

Databases:

- SQL: PostgreSQL
- NoSQL: MongoDB, Redis

System Design

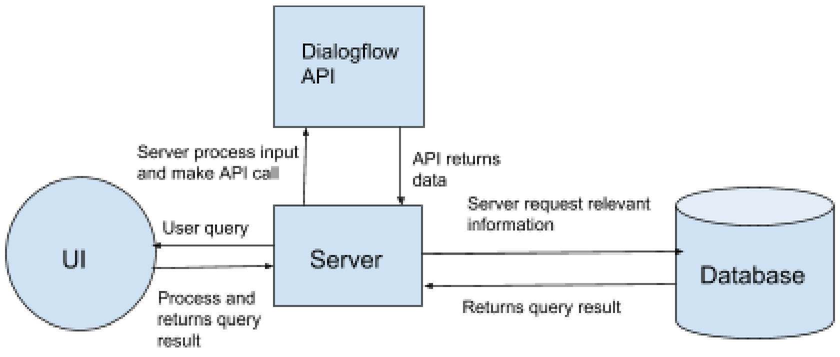
Architecture



The software will follow a standard server-client architecture where the server will handle most data processing including Dialogflow API calls, communication to database and controlling how information will be presented on the UI (See more detail in diagram below and backend section below for description). We are also expecting to leverage AWS RDS’s capabilities for running our database.

Data Flow

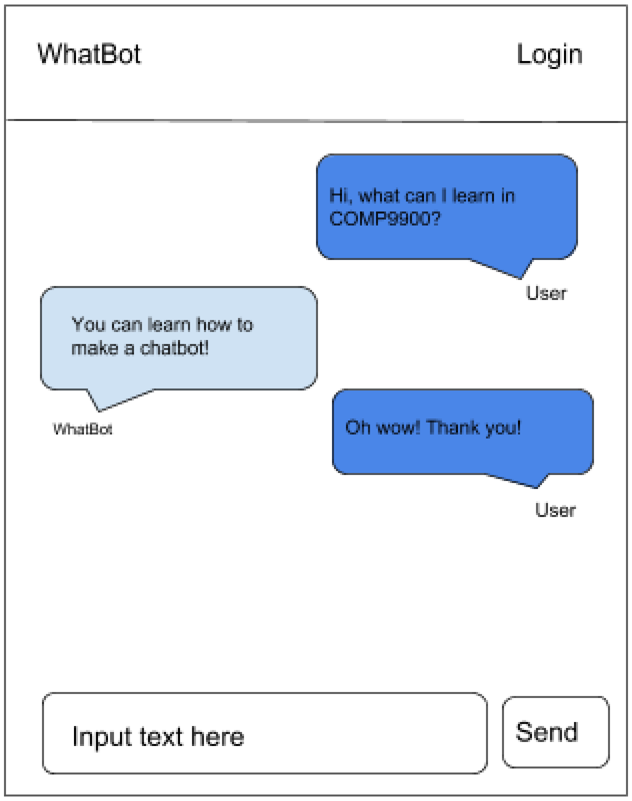
Data flow through the software architecture is described by the diagram below:


Dialogflow API
Server process input and make API call
API returns data
UI
User query
Server
Server request relevant information
Database
Process and returns query result
Returns query result

Frontend

The frontend plays an important part of the application as the user interface often plays a bit part in user experiences. Creating a fluid design and easy to use interface plays will make the application much more appealing, which leads to a better overall experience. Therefore, a strong focus has being put on making navigation easy, making smooth animations and using easy to look at styles and colours to create the user interface.

As the main feature for this project is the chatbot itself, the main user interface will follow a chatroom like design with additional functions to be added to the navigation bar at the top of the page to provide functions such as authentication functions e.g login and logout. A proof of concept for this design is shown on the diagram on the right.


WhatBot
Login
Hi, what can I learn in COMP9900?
User
You can learn how to make a chatbot!
WhatBot
Oh wow! Thank you!
User
Input text here
Send

Furthermore, this authentication function will allow the system to distinguish between students and administrators (lecturers) so that administrators will have access to a dashboard like page which provide analytic information on the usage of WhatBot, such as which students have submitted the most queries and what courses received were mentioned the most. More information on how analytics and authentication may be implemented in the backend section.

To achieve these key points and functionality, we make sure the core features are easily accessible and we also make sure good choice of colours and image designs are used. To study how to achieve this goal, the team will put a strong focus on examine how other existing chat services functions, how their interface is laid out and how they use colours to ensure a good viewing experience.

In terms of the technology that we will use for the frontend, the current frontend stack will be React only because it is an modern development framework that is fast and powerful. The frontend will then communicate to backend via API calls and the current design involves API calls being triggered every time the user inputs some information such that the input will be passed to backend as a parameter for further processing.

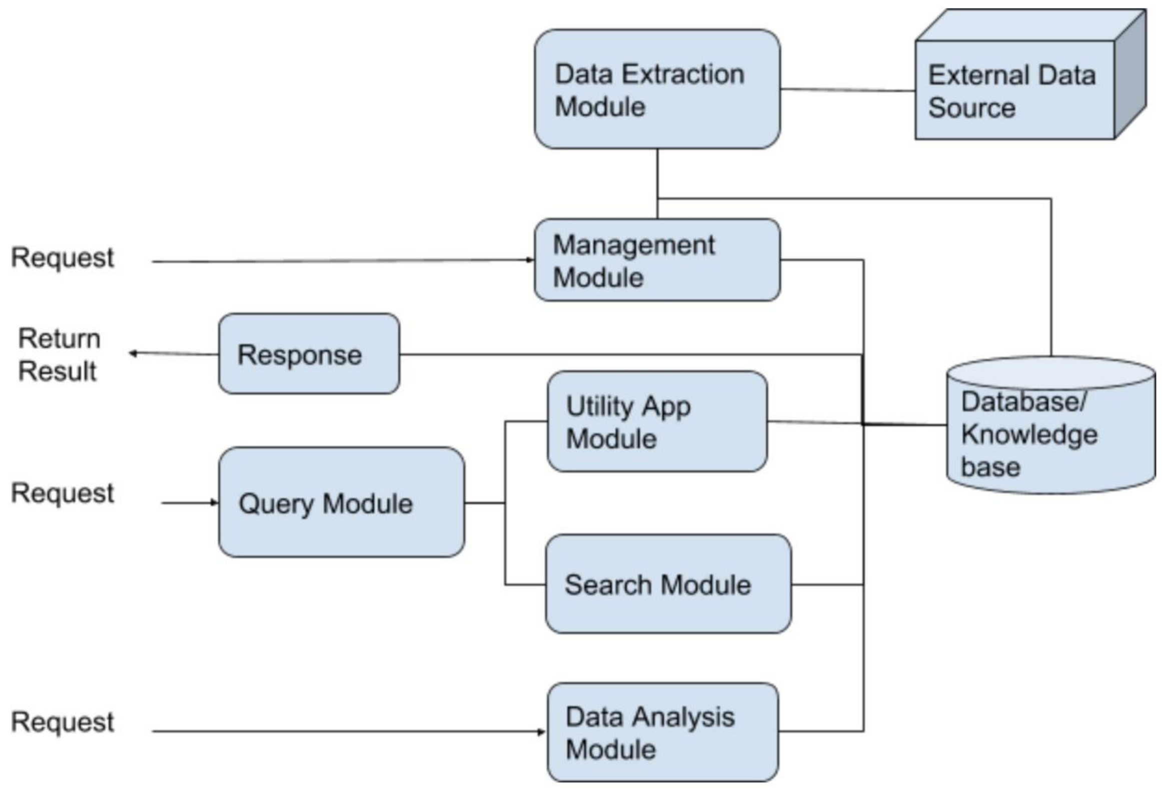
Backend

The backend server will handle all the data processing and contains the knowledge base, query module, utility app module, search module, data analytic module, management module, authentication module, data extraction module and a database as shown in the diagram below. It will also be where REST API calls are made to Dialogflow as the data flow diagram earlier suggests. Creating an efficient and effective backend will be crucial for user experience so much resources will be put into this area.

The stack we have decided to use for backend is Python Flask and Django as it aligns more with the team's skill set. Furthermore, the database, which is the knowledge base and database, we are planning on using is Postgres through Amazon RDS. The difference between knowledge base and database is that knowledge base would be used for course content informations that helps answer student queries while the database will store contents of student queries for the data analytic module to work.

It is expected that input message will be passed in through frontend as a post request with the user input that is to be processed. The message will then be parsed through the query module. In the query module, the query is classified and keywords are extracted. The query module may be implemented using Dialogflow APIs due to the powerful NLP ability it provides. For example, if key information for a query is not provided, Dialogflow's dialog function can help us request for it. The search module will then use the keywords to search through the knowledgebase for relevant information which will be passed into the response module, which will be implemented through Dialogflow, for the data to be processed into a more natural response.

The data analytic module is added for the epic of "Admin data analytics" so that admin users can get insights on various things described in the epics section of this proposal. For this module, efficiency is important as we may be potentially processing large amount of data. This module will allow our tool to provide even more solutions than other existing services as we will be using unique data that only our application possess.



Epics

I. Data Extractor

The data extractor will control what information goes into our knowledge base. It is the module that will perform data extraction from various online sources. The module will be used by internal system 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.

The data extractor functionality will be critical to the application as it controls the quality of responses that we can produce. Its extraction capability will be controlled by the number of data sources that our data extractor is compatible for since web scraping require us to know the HTML format of the website beforehand. Therefore, at current stage, we have decided to make our data extractor compatible to sources such as WebCMS3 and UNSW course handbook with more expected to be added on as development progresses. Therefore, what is out of scope for the data extractor is not a universal extractor that is compatible with all websites as that is simply too hard to implement.

We also plan to setup data extractor to perform periodic data update to our database by extracting the latest information from the sources that we have set. This will also ensure that the most recent and accurate results will be provided by other components that depends this module for their functions.

This epic is estimated to have a difficult rating of 3/10 and time estimate of 3/10.

II. Query Module

As the main source of communication between our software and the client would be through input text, a component is required to handle and analyse the user inputs. This query module will be the first form of contact of any requests that comes in from user input. Its performance will be core to how the rest of the system performs. The query module is responsible for classifying the type of input. For example, is the user input a question or a command. The query classification will determine the data flow and which component will be responsible for the query. Furthermore, it will perform name entity extraction where the extracted information will be passed onto the relevant module for them to perform their operations. Also, it will be responsible for formulating the response from other components into natural language before responses are returned to user interface.

For this epic, the Dialogflow API will be an important component as Dialogflow provides powerful NLP functionality. Therefore, it is crucial to setup effective intents and train powerful AI agent on its platform so that user queries can be processed. After query classification, the intent component will be responsible for parsing the user input. The Dialogflow agent will also be responsible for entity extraction to relevant modules. For example, the search module can use the results to perform relevant searches if the query requires it or the *Utility App Module* can use the results to do relevant computations or operations.

Overall, the query module's input will be some input from user and it will classify the input to decide on which module will handle the query and then produce relevant information for the relevant module to function. Therefore, it's functionality will play a big part on how well the system performs.

Nevertheless, this module will dependent on Dialogflow's NLP model and no implementation of our NLP will be made.

This epic is estimated to have a difficult rating of 8/10 and time estimate of 7/10 due to the team's unfamiliarity with Dialogflow.

III. Search Module

The search module will be responsible for information retrieval from the knowledge base. Key terms will be passed from the query module if the query is classified as a question type. With this module, we will be able to retrieve information for queries such as Which semester is this course offered? What is the learning outcome of this course? And what are the prerequisites for this course? Search module will be responsible for the results of common FAQs that students have for a class.

This module provides value for both student and lecturers. Students will be able to get answers instantly without having to spend more time looking for the answers elsewhere as there is currently no central place where all information for a course is stored. Lecturers will also have more time for their own work as they won't have to spend more time answering the same questions that many students are frequently asking.

The scope of this epic will include implementing efficient search algorithms that goes through our database using the keywords provided from query module. However, what is out of the scope for this module would be using the it to perform Google searches in situations where no information in our knowledge base can be found.

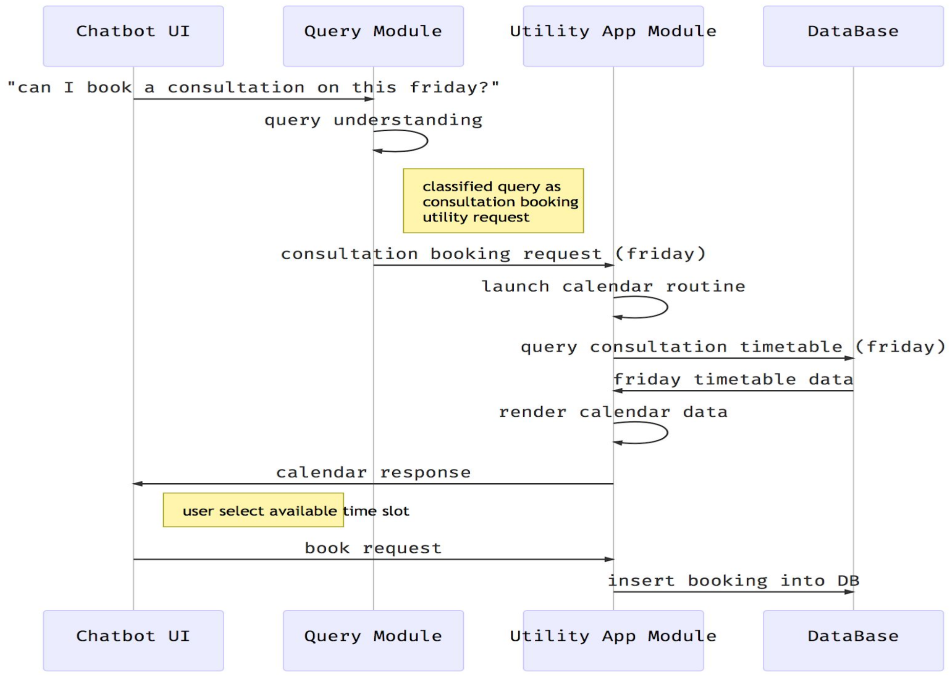
This epic is estimated to have a difficult rating of 6/10 and time estimate of 7/10.

IV. Utility App Module

Apart from the Q&A core feature, the Utility App Module of the chatbot will provide some other in-class support utilities as extensions, like:

1. Consultation booking calendar
2. WAM Calculator
3. Class room finder

After query understanding by query module, if the request is classified as an utility request (other than knowledge QA), this module will launch the corresponding routine to handle the request. Unlike Q&A, which is focused on information retrieval and correlated computation, the utilities are more like stand-alone apps embedded in the chat and they provide their service via natural language interaction. For example, if a student want to book a consultation then the calendar utility will be triggered. The process would be like this:



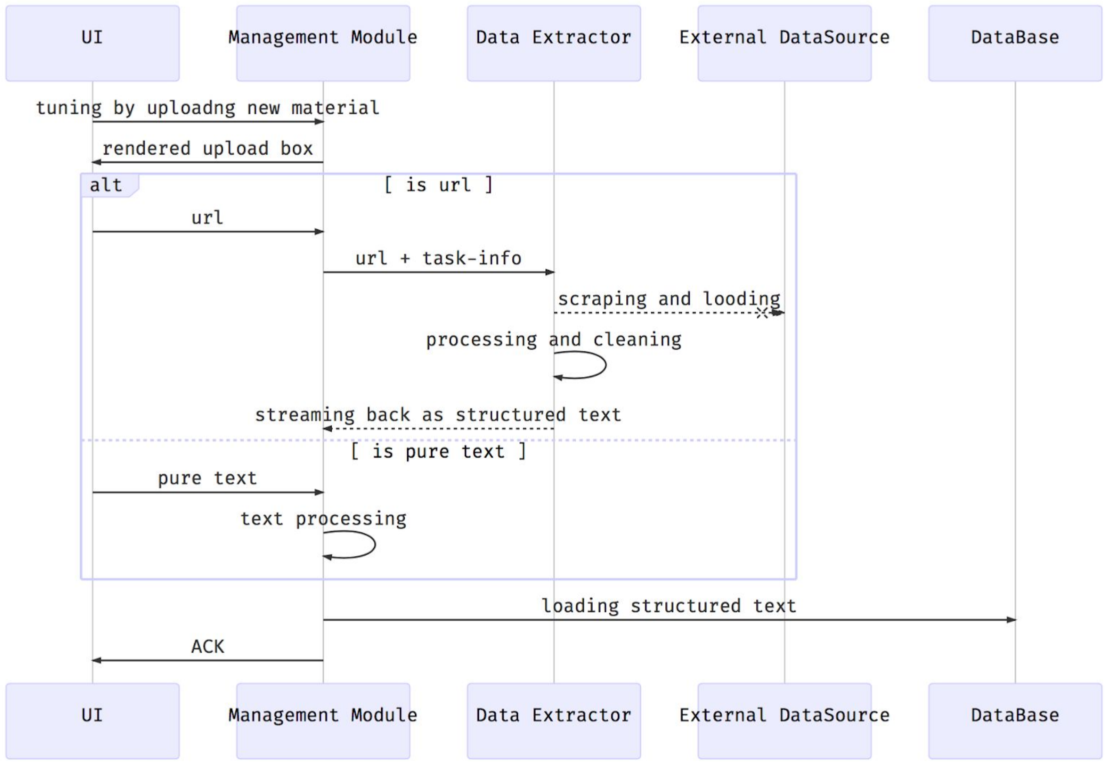
The utilities provided by this epic will be the ones presented in this report such that users should not be able to put external apps as plug-ins into our chatbot as this functionality is out of the scope of this epic.

This epic is estimated to have a difficult rating of 6/10 and time estimate of 7/10.

V. Management Module

Management module is essentially the backstage of the chatbot system, where course staff member can tune the chatbot, re-adjust the consultation booking schedule and configure other settings. The input of this system will be the configuration and setting changes coming in from the UI page.

For example, if the teacher wants to tune the bot by uploading new material to this system, the process would be like:



This epic provides a panel like interface for staff member to manage all the logistics, thus it can be implemented as a ordinary single page web application. The main challenge is on how to give out a compact and user friendly UI for teaching staff to use as part of their daily education activity. Therefore, for users with no NLP or even computer science background, the learning curve of this tool should be reasonably flat. To make this tool easy to use, users will not be able to add in extra configuration not listed in the inputs that the software allow as this will be out of scope for this epic.

This epic is estimated to have a difficult rating of 5/10 and time estimate of 5/10.

VI. Data Analytic Module

The data used for data analysis will mainly come from user inputs and the responses our software provide. This module will be responsible of providing insight to the data collected to help teaching staff to rationally think and make decisions.

It will collect the most popular words or sentences that students would like to ask and analyse the data for them to be visualised through a dashboard to reflect the real needs of the students.

It will not only analyze the requests from students, but it is also expected to periodically analyze the data. As a result, the user will be able to see the trends of student’s preferences and concerns.

In terms of the input, the operation in WhatBot is simple. When the user is authenticated to be a lecturer or tutor, they will be able to access the admin dashboard through clicking a tab on the web page. When they perform this action, it will trigger the data analytic module to present information through graphs and charts onto the dashboard.

However, what is out of scope is a customizable dashboard as the aim of the project is not to create another data dashboard software like Tableau so only a fixed preconfigured dashboard that is based on the data present in our database will be available.

This epic is estimated to have a difficult rating of 4/10 and time estimate of 6/10.

VII. Authentication Module

The authentication module is responsible for identifying which users are students and teaching staff. All the users have to login to the application before they start using. An authentication module is required as features like the admin dashboard will only be available for teaching staff so they can see backend data on queries being made.

This module will consist of a login page that requires users to enter their credentials with the admin uses (teaching staff) being provided access to the data analysis dashboard provided by the data analytic module through an extra tab on their web page.

With these data, the lecturer in charge can also gain benefit from the chatbot such that he/she can improve their course contain to attract more students into their course. This is because, he/she can learn where he/she is not clear on course outline or other information and also gain insight on what most student's concerns are about their course.

While this module will manage user login and logout, it will not handle sign up or forget password features as the software should be used and configured by the university such that all user related operations credential management should not be done through our application.

This epic is estimated to have a difficult rating of 3/10 and time estimate of 4/10.

Final Epic Selection

From the epics presented above, we have chosen to deliver the following:

- High Priority: Query Module, Search Module, Utility App Module, Data Extractor, Data Analytic Module
- Medium Priority: Management Module
- Low Priority: Authentication Module

The high priority epics were chosen because they provide the core capabilities of our software, so the success of the project depends on them. Furthermore, the components being tightly coupled was another factor taken into consideration. This resulted in the management module being medium priority as it acts as more of an enhancement that tunes our search module and algorithms with some features exclusive to admin user, which is a small set of functions. Therefore, the application will still be able to operate without the management module to a great extend with only little loss of features. Lastly, the authentication module is placed with a lower priority as it is easier to implement and no real features is dependent on it. For example, if it is not implemented, the data analytic module will still work but students will also be able to see what queries are being asked, which may not actually be a bad thing. Therefore, authentication module only acts as a user identifier which is not as important to the core functionalities.

The team's aim is to deliver all of the epics chosen but the amount of attention for each epic will be based on the priorities listed above. Therefore, in the situation where time does not allow, the lower priority epics may be only partially or not implemented.

Development methodology

Agile software development practices are used by our team with fortnightly sprints and stand-ups every two days. Github project board will also be used to track project and sprint work. All members will contribute to committing code to Github repository that has already being set up.

Proper software development practices will be in place such that code commits to main branch will be made through pull requests so that they go through CI/CD pipeline and ensure a fully functional main branch. Furthermore, the team has decided to put emphasis onto code review to ensure a consistent coding style, well documented and quality code such that a high quality and maintainable product is produced at the end as software quality is a highly valued component.

Stand-up schedule:

Every Tuesday, Thursday and Sunday at 10:30pm.

Sprints schedule:

1. Week 3, 4
2. Week 5, 6
3. Week 7, 8
4. Week 9, 10

Appendix: Core Features and Scenes

I. Pre-enrolment Consultation

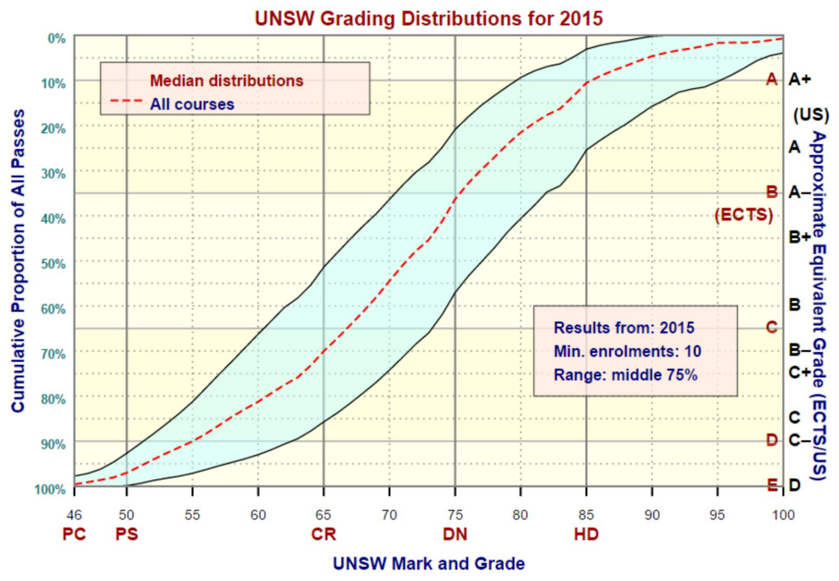
Usually in the middle of a semester, enrolments open for next term. At this time, there are usually high demand for course consultation. Students want to know what course they should do, and if a specific course is a good option for them. Lecturers may also do marketing or pre-sale work for their courses.

The chatbot will be able to answer common student queries like the following:

1. What are the courses I can take in next semester?
2. What is the learning outcome of this course?
3. How are the marks calculated from each part?
4. In what form would this course be taught in?
5. What are the assessment components in the course?
6. How hard is this course? What is the WAM stats in past years? What is the fail rate?
7. Based on my prerequisite(s), what is the predicted mark that I can get on this course?

Pre-enrolment consultation is not only important to students, but also important to educators. Having a chatbot to answer FAQ to students can effectively reduce the workload for student service. Besides, following points might also interest the lecturer in charge:

1. What are the WAM distribution of the potential students like?
2. What are their expectation on the learning outcomes?
3. What are the top 3 concerned questions of the student market?



Sample distribution of WAM

II. Class FAQ

For many common queries the answers are usually already available in the corpus of data available for the course or in the knowledge database for a specific service. These common queries include:

1. What is the penalty for submitting a specific assignment late?
2. In what form will the lab/quiz will be conducted?
3. What are the core topics of this course?
4. What is the knowledge graph/map like for a given point?

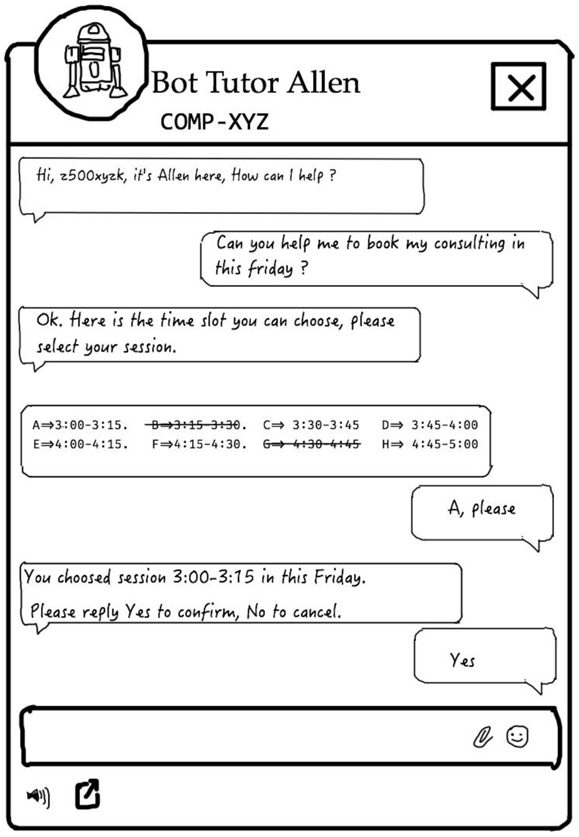
With chatbot taking care of these questions, lecturer and tutor can save time to focus on the questions that really matter. If meeting any question that can't be effectively answered by the chatbot, the teachers and tutors will be informed to take further actions. For example, a notification will be generated on admin dashboard or an email will be sent to teaching staff. And the the chatbot will show the consultation booking interface and ask the student if he or she needs to book a consultation and enter the details for booking.

III. Consultation Booking

Consultation is usually booked by email communication, but manual email reading is tedious and doesn't scale very well. Specifically, consider a big course which owns large number of students and the corresponding high volume of consulting demand, not only the communication will be a headache, but also scheduling the consultation will arise to be a O(n*log(n)) complex work.

Besides, email replying is usually delayed for uncertain time, especially when the teaching staff are busy and the request quantity is high. As the increasing of request, the quality of service will drop drastically.

However, chatbot, as a virtual assistant to tackle this, can serve 7*24 hour per week and reply to student consultation requests in real-time. And even every students in class send a request to book consultation, it can be easily handled by a computer server, making this process from O(n*log(n)) to be felt like O(1);



Hi, z500xyzk, it's Allen here, How can I help ?

Can you help me to book my consulting in this friday ?

Ok. Here is the time slot you can choose, please select your session.

A⇒3:00-3:15. ~~B⇒3:15-3:30~~. C⇒ 3:30-3:45 D⇒ 3:45-4:00
E⇒4:00-4:15. F⇒4:15-4:30. ~~G⇒ 4:30-4:45~~ H⇒ 4:45-5:00

A, please

You choosed session 3:00-3:15 in this Friday.
Please reply Yes to confirm, No to cancel.

Yes

For example, there are 15 minutes for each timeslot, and the consultation will start from 9 AM to 16 PM on workdays.

- The students can reserve it by their student ID at least one day in advance.

- The UI interface only shows the available time slots of that day that students choose.
- Once confirmed, the students will receive an email as receipt and also as a thread to postpone or cancel the booking.
- If there is any exception in teaching stuff side, these email threads can also be served as notification to re-adjust the consultation appointment.

IV. In-Class Support

The chatbot will also act as a management feature for students so they can find out information on their current progress and help them manage their time. Some of the features include:

1. A WAM calculator
2. Assignments and projects due date tracker
3. Latest progress made in lecture.
4. Get lecturer announcements.
5. Classroom and tutorial locator

V. Admin Dashboard

Admin dashboard is a Single Page Application that provides an admin dashboard for lecturer in charge and tutors. Apart from the logistics features like login authentication, it will also provide an interface that can show the insight of chatbot system based on data such as chat history and running log. Specifically, the insight can be like:

1. What are the top x questions in a given time period.
2. The score of user experience
3. Feedback collector