G52GRP Final Group Report

Project Title:

Virtual Patient Chatbot for Pharmacists

Date: 28/4/2021

Group 8A

Student ID	Name	Course
20089417	Khor Yong Teng	Comp Sci with AI
20129374	Law Khye Yueh	Comp Sci
20129377	Lee Ren Jin	Comp Sci with AI
20150707	Ang Jia Hau	Comp Sci with AI
20194491	Steven Ho Chu Leong	Comp Sci
20212765	Alya Amirah Muhammad Saufee	Comp Sci with AI

Supervisor : **Mohammad Aazam**

Final Word Count : 7240Words

Table of content

1. **Abstract**
2. **Introduction**
    - 2.1 Aims & Objectives
    - 2.2 Project Description
    - 2.3 Client
3. **Literature Review**
    - 3.1 Background & History
    - 3.2 Similar Systems
    - 3.3 Technologies
4. **Project Description**
    - 4.1 Product Features
    - 4.2 General Constraints
5. **Requirement Specifications**
    - 5.1 Functional Requirements
    - 5.2 Non-Functional Requirements
6. **User Interface Design**
    - 6.1 Log In Page
    - 6.2 Registration Page
    - 6.3 Select Exercise
    - 6.4 Chatbot
    - 6.5 Mobile Responsiveness
7. **Implementation & Methodology**
    - 7.1 Initial Design
    - 7.2 Use Case
    - 7.3 Architectural Design
    - 7.4 Activity Diagram
    - 7.5 Sequence Diagram
    - 7.6 Implementing DialogFlow for Conversation
    - 7.7 Feedback Generation
    - 7.8 Google cloud platform, Dialogflow API and Server.js
8. **Time Plan**
    - 8.1 Autumn Semester
    - 8.2 Spring Semester
9. **Group Organization**
    - 9.1 Individual Roles
    - 9.2 Workload Division
    - 9.3 Meeting System

10. Discussion

11. Conclusion

12. Summary of Achievements

13. Reflective Comments

14. Appendix

14.1 Test Case

14.2 Meeting Minutes

14.3 User Manual

15. References

1. Abstract

In this project, we need to create a chatbot that acts as a patient to train and improve pharmacy students' skills by seeking consultation from them. The chatbot will give feedback at the end of the session which shows the overall performance of the student. We are using DialogFlow API for this project since it has extensive language processing capabilities and can produce the appropriate outputs needed to generate the feedback. In addition, a database is also required which helps to record user inputs as well as for login authentication. Further improvements and understandings are required in order to improve our chatbot functionality.

2. Introduction

2.1 Aims & Objectives

The main purpose of this project is to create a chatbot which acts as a patient, seeking the consultation and assistance of a pharmacy student to give medical advice. This project aims to be an exercise which will train and optimize the skills of pharmacy students in their prescription and counselling skills.

2.2 Project Description

The automated system is for users, such as pharmacists or pharmacy students, that allow them to practice their knowledge through a chatbot. It provides real world scenarios including patient interactions which could help them upgrade and enhance their understanding in a very comfortable environment at any time.

To work with the chatbot, the student needs to log in first by using his/her university's email and password. For a student who does not register yet, he/she has to register beforehand by filling up the registration details. After logging in, the user can choose between 3 different exercises based on his/her option. By then, the student should start questioning the bot which acts as a virtual patient based on the medical status and history of the bot in order to give an accurate conclusion.

Subsequently, the student will be assessed based on whether all required questions have been asked during the session and the chatbot will then provide some feedback based on the sequencing and replies of the student. Feedback is important and very needed in order to help students to improve their learning. Consequently, the feedback can assist students if there is any forgotten information that has not been included during the session as well as if the questions have been asked in the exact sequence.

2.3 Client

1. University of Nottingham Pharmacy Students: This group of users comprise of Pharmacy Students in The University of Nottingham Malaysia. They also undergo exams which involve theory and practical studies which require real life training to prescribe the correct medication but this is something they can't get which is why this chatbot is created in order to help them.

2. University of Nottingham Pharmacy Lecturers: This group of users are the module convenors who wish to keep track of the student's progress and their ability to get the required information and ask the right questions in order to prescribe the right medication. The lecturers can go through the history of the chatbot in order to help the student improve too instead of just the chatbot providing feedback.

3. Literature Review

3.1 Background and History

The term chatbot may be a commonly heard term in this technology filled society but chatbots actually goes way back to 1966, where the first chatbot, Eliza[1], named after Eliza Doolittle was developed. Later, more and more people started following this technology which led to the creation of other chatbots such as Parry in 1972 and Julia by Lycos,Inc. in 1994. Obviously, as time went by, chatbots were improving too. This includes the implementation of AI technology in chatbots such as Siri that appeared in 2011 and FB Messenger bot.

The first chatbot, Eliza, is a chatbot that emulates a Rogerian psychotherapist. This shows that even the first chatbot has some ties to the healthcare industry. Chatbots have been implemented in many different fields and one of the significant fields that was affected is the pharmacy field. Medical chatbots have huge potential to improve the life of people whether it is in terms of well being or education. In the aspect of education, the current pandemic situation caused by covid-19 has significantly affected the pharmacy students' ability to train with real life standardized patients. Therefore, an online virtual patient chatbot that would allow students to achieve active learning and practice their knowledge will greatly improve the overall standard of healthcare.

3.2 Similar Systems

Virtual patients are interactive computer simulations that aid in healthcare education (Poulton, T., 2009) by complementing clinical training (Huang, G., 2007). These simulations may encompass various types of forms, including Virtual Standardized Patients (VSP), which are simulated patients with natural language processing capabilities, enabling training in provider-patient communication skills. . (Kononowicz, A. A. et al, 2015)

Such a VSP can be seen in the technical report of “Developing a Conversational Virtual Standardized Patient to Enable Students to Practice History-Taking Skills” (Maicher K, et al, 2017). The VSP was developed in the Unity game engine as emotionally responsive 3D characters and integrating ChatScript, a dialogue management system for conversing and providing replies. Medical students were to converse and take the history of the VSP, develop a differential diagnosis and document the experience in the electronic medical record. However, the assessment of the student’s performance was required to be done manually. The accuracy of the responses from the VSP ranged from 79% to 86% and students were able to accurately develop a differential diagnosis based on the interaction with the VSP.

Aside from that, a similar system but for Pharmacy education can be seen in “A Computer Simulation of Community Pharmacy Practice for Educational Use” (Bindoff, I, 2016). The simulation was developed using Unity3D game development environment, simulating a complete community pharmacy. Students are allowed to freely interact with the environment and converse with the virtual patient. However, the conversation with the virtual patient can only be done through preset choices as dialogue options. The simulation scenarios can also be created and customized by educators through a builder tool developed by the developers. To start a session, the students are allowed to choose one or more scenarios from prewritten ones.

3.3 Technologies

In the new century as we are now living is a world of constant digital transformation. We can see evidence of AI in almost every environment, especially assistants like Microsoft's Cortana and Apple's Siri to our conversations with chatbots[8]. Chatbots are easily one of the most well-known examples of artificial intelligence. It appears at a rapid pace throughout the communication history. According to Gartner, by 2020, around 85% of our interactions will be handled by bots instead of humans. It helps to answer questions and complete repetitive tasks through text messages or instant messaging online.

Chat Interface is the face of the system through which the entire communication takes place. It is responsible for collecting user queries from the user which are the input to the system as it is an important role at the front end of the system. From the very first chatbot, ELIZA[9], it was developed in 1966 at MIT. It uses a simple decision tree and a template-based response mechanism. Until now, artificial intelligence has improved a lot as the next step of creation of virtual personal assistants like Google Assistant[10]. To perform its functions, Google Assistant relies on natural language processing and machine learning to understand what the user is saying, and to make suggestions or act on that language input.

For the chatbot algorithm, Natural Language Processing (NLP)[11] and Natural language Understanding (NLU)[12] are also taking part as an important role that should be implemented in the chatbot system. In the context of bots, it receives input from users and creates responses based on a contextual analysis similar to human beings. Natural Language Understanding makes chatbot understand the input in a form of sentences in text or speech format. It enables human-computer interaction (HCI)[13]. NLU uses algorithms to reduce human speech into a structured ontology and is programmed with the ability to understand the meaning in spite of common human errors like mispronunciations or transposed letters or words. NLU techniques enable the system to identify words or sentences if the user makes use of them while conversing with the chatbot, making the user feel that the conversation is taking place between two humans but not between a human and a bot. This Artificial Intelligence improves the efficiency of conservation between chatbot and human beings.

Next, for the word segmentation[14] will also be doing the important task as it's a process of splitting text into smaller and meaningful units such as in a chatbot system. In 2017, Naeun Lee et al. explained the implementation of word segmentation using natural Language ToolKit (NLTK)[15]. NLTK is a python package which provides services for NLP which contain inbuilt tokenizers[16]. The accuracy, speed and efficiency of NLTK tokenizers is recommended. Furthermore, it does not require any algorithm implementation.

4. Project Description

In a pharmacy, it is required for a pharmacist to ask questions and take note of a patient's medical history in order to provide accurate prescriptions. In order to do so, a pharmacist requires experience in conversing with a patient. Pharmacy students are not yet qualified to prescribe actual patients and their chances of interacting with actual patients are limited. Thus, it is difficult for them to acquire practical experience.

This project mainly focuses on us creating a chatbot to assist future University of Nottingham Pharmacy students to refine their prescription skills in order to obtain information from patients in order for them to prescribe the correct medicine.

The solution is creating a chatbot which acts as a patient, seeking the consultation and assistance of a pharmacy student to give medical advice. This project aims to be an exercise which will train and optimize the skills of pharmacy students in their prescription and counselling skills. There will be multiple exercises for the student to select before starting. The chatbot will initiate the chat, and the student should start questioning the chatbot to gain more information on the medical status of the bot in order to give an accurate conclusion. After that, the student will be assessed based on whether all required questions have been asked. Finally, the chatbot will provide some feedback based on the sequencing and questions of the student.

4.1 Product Features

The main features of PharmaBot consists of, but are not limited to:
- Natural Language Responses: The responses by the chatbot will be written in the standard and understandable English with Pharmacy Terms

- Natural Language Processing: Our chatbot will be able to receive and be able to understand and process replies from user written in English

- User Authentication: The system will be able to authenticate students and lecturers who are accessing this system.

- Assessment and Feedback: The system will be able to provide feedback based on user replies as a means of assessing a students aptitude in counselling.

4.2 General Constraints

1. Language Restrictions: The system/chatbot will only be able to understand and reply back in simple basic English only.

2. Limit of the Questions: There are thousands of questions that can be asked by the pharmacist in various ways. To answer the questions properly, the chatbot needs to be trained using a wide range of datasets in order to be accurate. In initial stages of training, the chatbot might find difficulty in understanding questions asked but we will slowly improve and feed in more datasets so that the chatbot can be more accurate in understanding and responding to the questions.

3. Requires maintenance : The industry of medicine is constantly going through research & development to strive for improvements. A few years from now, there will be better medicine alternatives to be recommended by pharmacists. Since the chatbot is trained by datasets fed to it, it will have to be regularly updated following the development and updates of medicine.

4. Limited Responses : The chatbot limits the user to one response per message. Users will have to wait for the chatbot to reply to their message before sending another. The word limit per message is limited to 256 words.

5. Requirement Specifications

5.1 Functional requirements

5.1.1 Hardware requirements

- Users will need to have an internet connection to access the webpage and chatbot.
- Users need to have hardware that allows them to fully utilize the chatbot. Eg. keyboard for input,mouse to select,screen for display.

5.1.2 Basic Conversation Etiquette

- Chatbot will respond and greet users when a user initiates a conversation.

- Chatbot will reply in a polite manner while conversing with users.

5.1.3 User Authentication

- User will be prompted for authentication before starting a session

5.1.4 Chatbot sessions

- Users will be allowed to choose different chatbot sessions for different exercises.

5.1.5 Understanding Natural Language

- Chatbot will be able to understand natural language and identify keyword to respond accordingly

5.1.6 Error Handling

- Chatbot will prompt user to repeat or reword the query when keyword is unable to be identified

5.1.7 Assessment and Feedback

- Chatbot will assess the response from users and provide feedback based on them.

5.2 Non-functional requirements

5.2.1 User communication

- Users should have basic english language understanding to operate the system and chatbot.

5.2.2 Security requirements

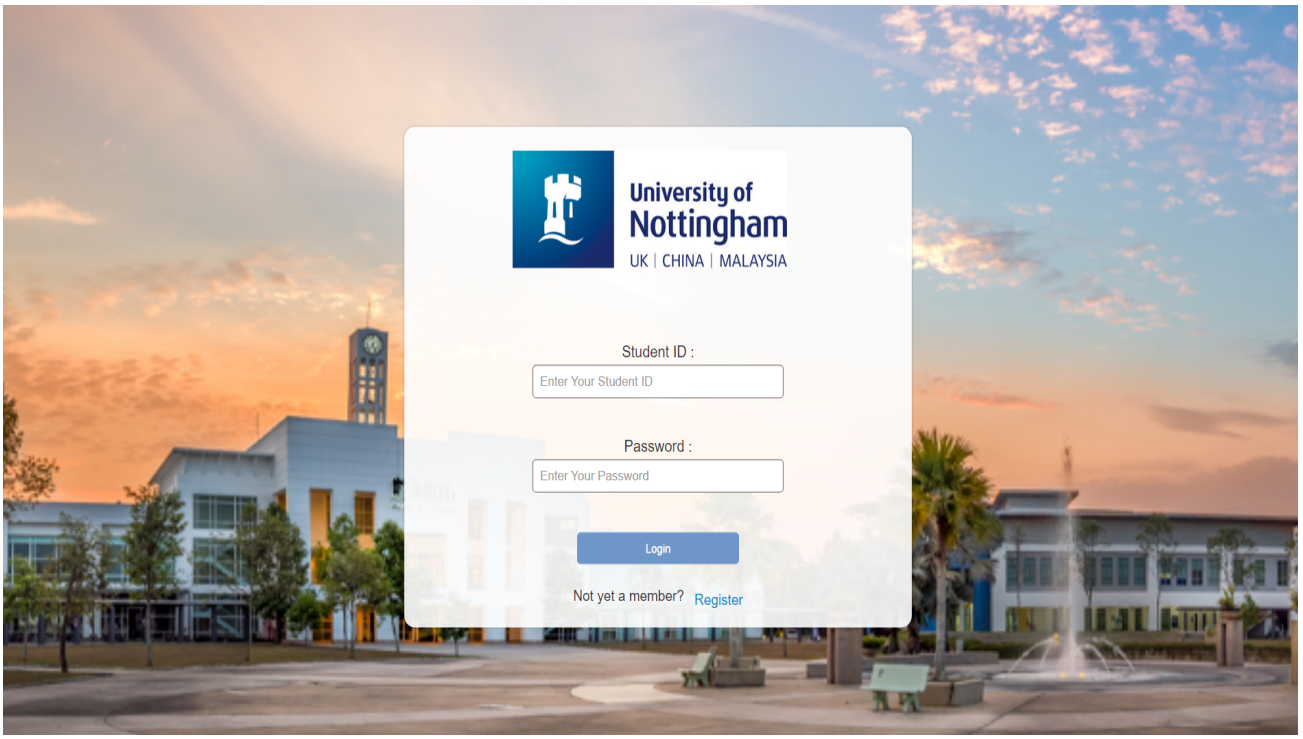
- User log-in details will be encrypted.
- Chat logs between chatbot and user will be private and confidential unless user requests to export them.

5.2.3 Chatbot response rate

- Chatbot will issue a reply within 5 seconds

6. User Interface Design

6.1 Log In Page



The front end of the login page was created using handlebars(HBS) & CSS, whilst the backend is handled by Node.js and SQL. It has a straightforward yet clean template which ensures the Student ID entered matches the password created within the database. Besides that, first time visitors are given a prompt below the login button which leads them towards the registration page. We made sure the user interface and experience was clean and simple yet practical.

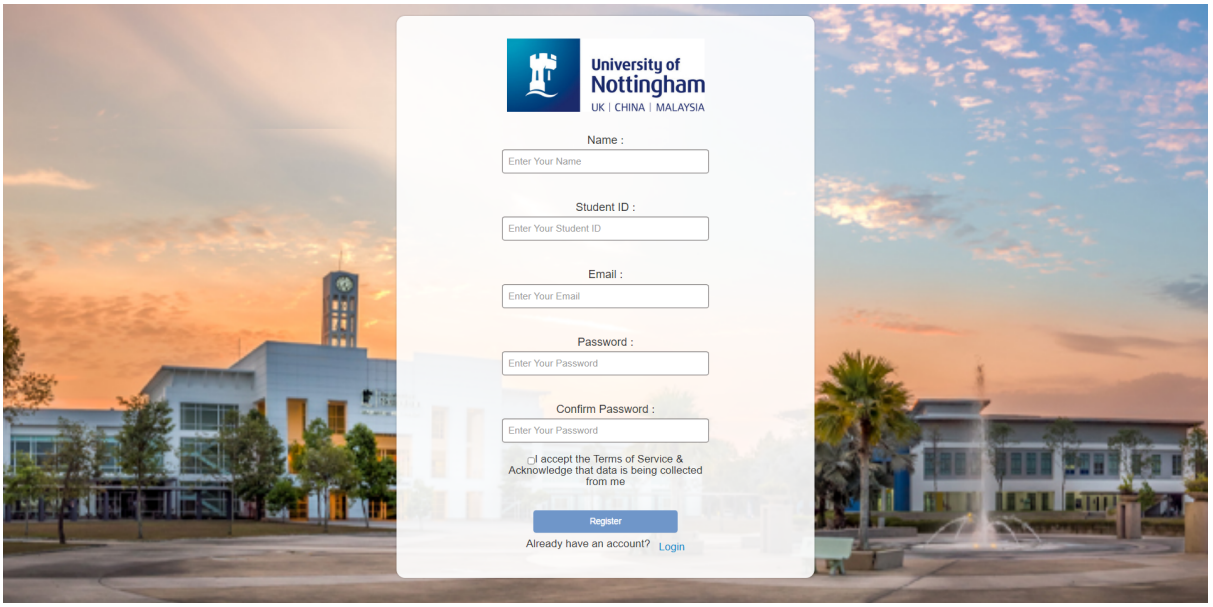
```
exports.login = async (req,res)=>{
    try{
        const { studentId,password} = req.body;

        if( !studentId || !password){
            return res.status(400).render('login.hbs', {
                message : 'Please enter both student ID and password to proceed.'
            })
        }

         //Positional parameter to avoid SQL injections, and to check student ID matches password
        con.query('SELECT * FROM users WHERE studentId = ?', [studentId], async (error,results) =>{
            console.log(results);

            if( results[0] == undefined || !(await bcrypt.compare(password,results[0].password)) ){
               res.status(401).render('login.hbs',{
                message: 'Invalid student ID or password'
            })
        }
```

```
                else{
                const name = results[0].name;
                const studentId = results[0].studentId;
                //create token, take in name as identifier to create the token,
                 //JWT variables stored in .env file, just as other important passwords are
                const token = jwt.sign({name, studentId},'process.env.JWT_SECRET',{
                    expiresIn: process.env.JWT_EXPIRES_IN
                   })

            console.log("Token is : " + token);

            const cookieOptions = {
                expires : new Date(
                    Date.now() + process.env.JWT_COOKIE_EXPIRES * 24 * 60 * 60 * 1000
                ),
                httpOnly: true
            }
            //Create cookie, and redirection after log in, change redirect.
            res.cookie('cookie',token, cookieOptions);
            res.status(200).redirect("/select");
           }

        })

     }catch (error){
        console.log(error);
    }
}
```

Figure 6.1

Upon users clicking on the “login” button, a router.post request will be sent, and the function in Figure 6.1 will be called. It selects the “users” table and finds a matching query between Student ID and password. Because the passwords stored within the user table is hashed using the bcrypt function from Node.js, we will need to use “bcrypt.compare”. If none matches, a message of “Invalid Student ID or password” will be shown. Upon successful login, users will be redirected to the “select” page, and a cookie will be created for the session. Positional parameter is used as a basic security function to avoid SQL injections.

6.2 Registration Page



The registration page has 6 queries to be filled, including the checkbox at the end which ensures students approve and are aware of their data being recorded. The overall design has the same style as the login page, as consistency is an important factor in UI/UX.

```
exports.register = (req,res) => {
    console.log(req.body);

    const{ name, studentId, email, password, passwordConfirm} = req.body;

    //Positional parameter to avoid SQL injections, and to avoid same email being registered twice.
    con.query('SELECT studentId FROM users WHERE studentId = ?', [studentId], async (error,results)
=>{

        if(error){
         console.log(error);
        }



        if(results.length > 0 ){
            return res.render('register.hbs', {
                message: 'This student ID already exists.'
            });


        } else if (password !== passwordConfirm){
            return res.render('register.hbs', {
                message: 'The passwords entered do not match, please try again.'
            });
        }

        //8 rounds of encryption
    let hashedPassword = await bcrypt.hash(password, 8);
        console.log(hashedPassword);

        con.query('INSERT INTO users SET ?', {name: name, email: email, studentId: studentId,
```

```
password : hashedPassword }, (error,results) =>{
                if(error){
                        console.log(error);
            } else{
                console.log(results);
                 return res.render('login.hbs', {
                messageSuccess: 'You have successfully registered! Please log in with your
credentials.'
            });
            }
        })
    });


        emailExistence.check(email, function(error, emailResponse){
                 if(emailResponse === false ){
                          return res.render('register.hbs', {
                                      message: 'Please enter a valid email.'
                                 });
                        }

                  });


}
```

Figure 6.2

Similar to the login page, a router.post request will be sent which calls the function in **figure 6.2** when users click on the “register” button. There are multiple error handling methods in place. Firstly, if the Student ID already exists, the student will be prompted and has to contact the admin for the fix. Secondly, the query within the fields of “Password” and “Confirm password” must match. If the account is successfully created, the password entered will be hashed when stored within the database, which acts as a security feature. Lastly, we used the emailExistence function from node.js to ensure the email registered is existing and students do not simply enter an unregistered and illegitimate email.

6.3 Select Exercise

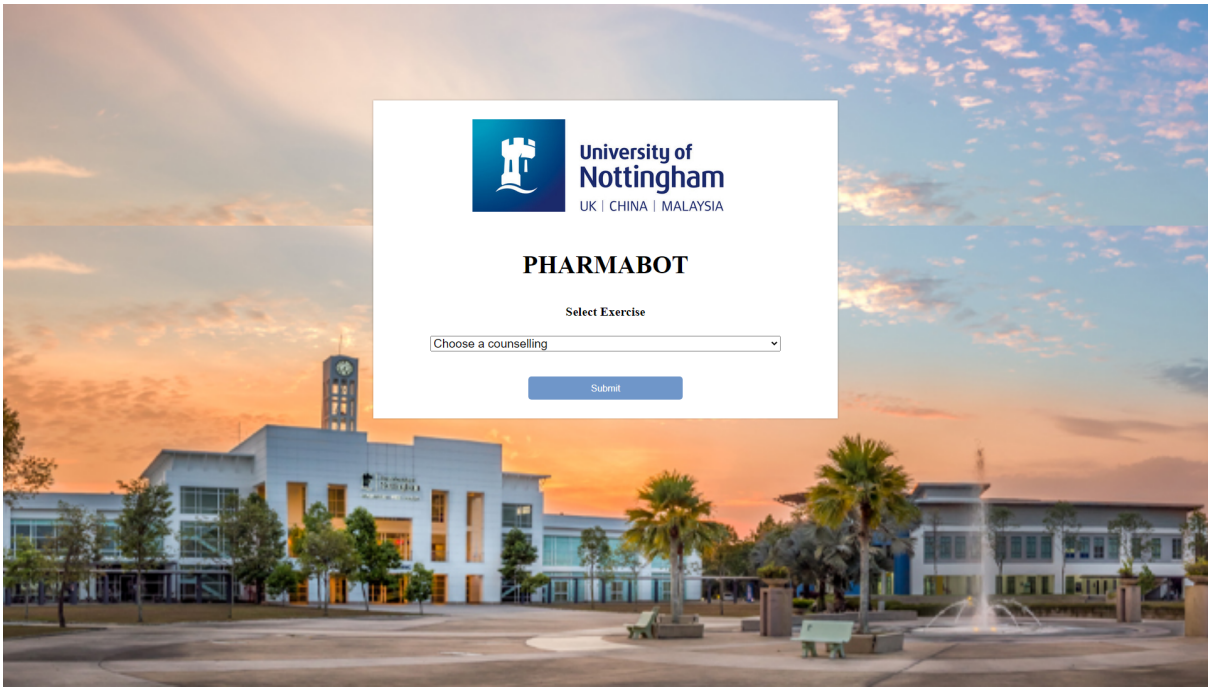


Figure 6.3

Upon successful registration and login, students will be brought to this page to select their counselling exercise and to proceed to the session with the chatbot.

6.4 Chatbot

Figure 6.4

The Chatbot's interface has 3 basic features. Sending messages, Volume Mute or Unmute , and Microphone for Text to Speech.

```
<form class="message-box" id="mymsg" method="POST" autocomplete="off">
```

Figure 6.4.1

```
function insertMessage() {
  msg = $('.message-input').val();
  if ($.trim(msg) == '') {
    return false;
  }
  $('<div class="message message-personal">' + msg +
'</div>').appendTo($('.mCSB_container')).addClass('new');
  fetchmsg();

  $('.message-input').val(null);
  updateScrollbar();

}

//Message Submission
document.getElementById("mymsg").onsubmit = (e)=>{
  e.preventDefault()
  insertMessage();
}
```

Figure 6.4.2

As seen in Figure 6.4.1, the message box at the bottom with a placeholder of "Type Message" is a form, which takes a text input with a maximum of 256 characters. Upon submission, a post request will be sent and a new div will be created which displays the message send, as seen in Figure 6.4.2.

```
function fetchmsg(){

    var url = '/send-msg';

     const data = new URLSearchParams();
     for (const pair of new FormData(document.getElementById("mymsg"))) {
         data.append(pair[0], pair[1]);
     }

    var botnum = document.getElementById("botnum").value;
    data.append("botnum",botnum);

     console.log("abc",data);
       fetch(url, {
         method: 'POST',
         body:data
       }).then(res => res.json())
```

```
        .then(response => {

          console.log(response);
        serverMessage(response.Reply);

            var botReply =  new SpeechSynthesisUtterance(response.Reply)
            if(botSpeak ==true){
             speechSynthesis.speak(botReply);
                }

         })

          .catch(error => console.error('Error h:', error));
}
```

Figure 6.4.3

Then as seen in Figure 6.4.3, the fetchmsg() function is called which will fetch the response, append the sent message to the reply and display it, followed by using Speech Synthesis for Text to Speech given the user has not muted the chatbot.

```
    function checkVolume(speechText){
            //mute button
           $('#mute-btn').on('click', function(e) {
               speechSynthesis.cancel(speechText)
              botSpeak = false;

                 document.getElementById("mute-btn").style.display = "none";
                document.getElementById("unmute-btn").style.display = "inline-block";
           });

           //unmute buton
           $('#unmute-btn').on('click', function(e) {
               botSpeak = true;
               console.log(botSpeak);
                document.getElementById("unmute-btn").style.display = "none";
                document.getElementById("mute-btn").style.display = "inline-block";
           });
    }
```

Figure 6.4.4

Lastly, the volume button is done through some basic javascript and CSS. The two buttons are first overlaid on each other. One of the buttons is then hidden whilst the other is displayed. Depending on which button is clicked, they will swap roles. If the mute button is clicked, speech synthesis will be cancelled, volume is muted, and the mute button is hidden whilst the unmute button is shown. The same procedure is carried out but in opposite parameters, for when the unmute button is clicked.

6.5 Mobile Responsiveness

6.5.1 Login

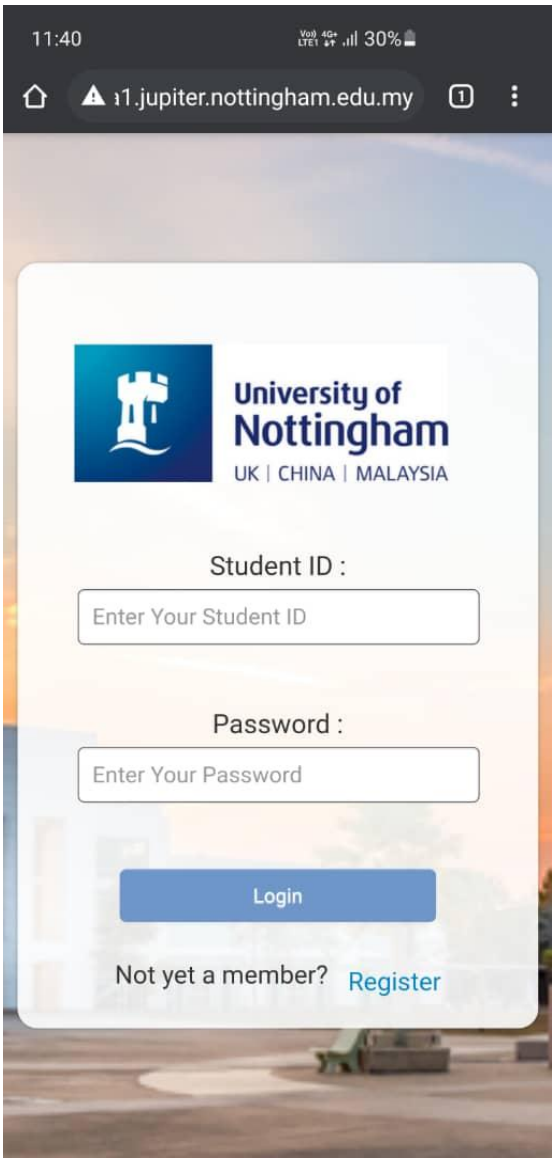


Figure 6.5.1

6.5.2 Registration

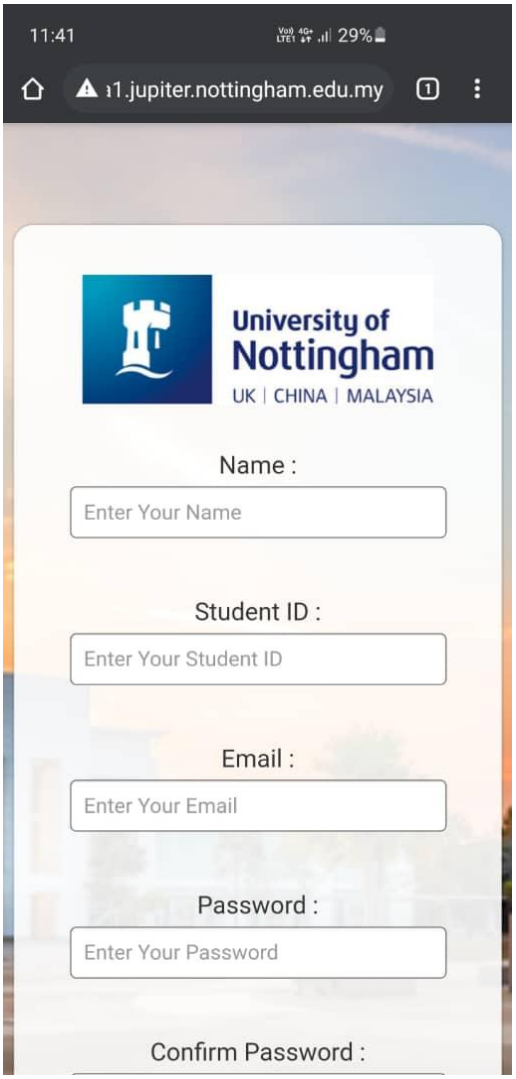


Figure 6.5.2 - Top Section

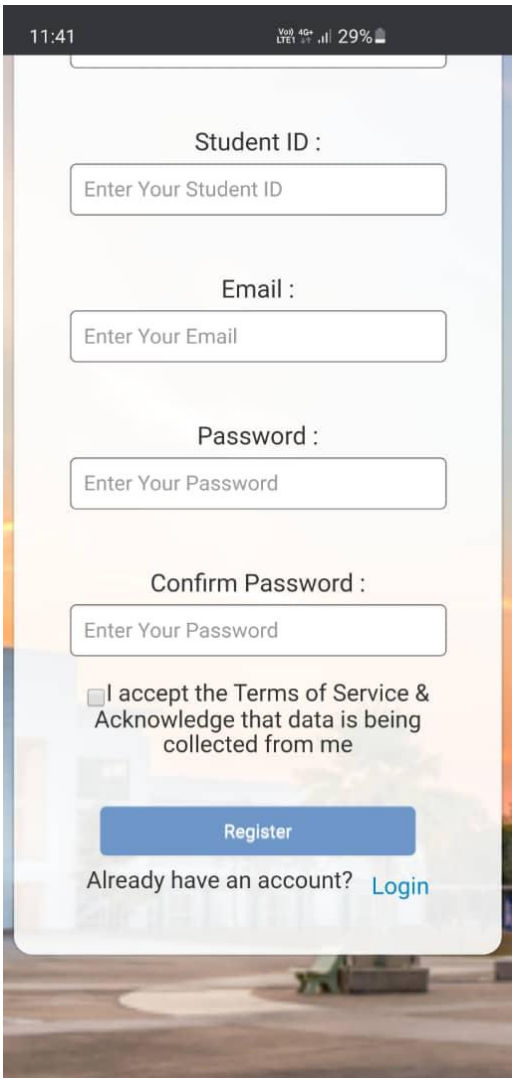


Figure 6.5.2 - Bottom Section

6.5.3 Select Exercise

Figure 6.5.3 - Static

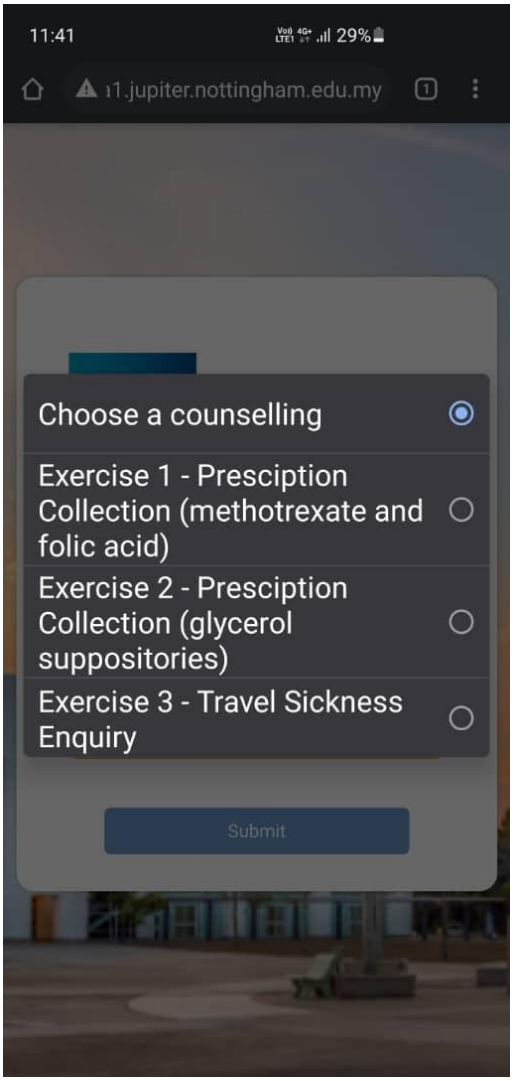


Figure 6.5.2 - Counselling Button Pressed

6.5.4 Chatbot

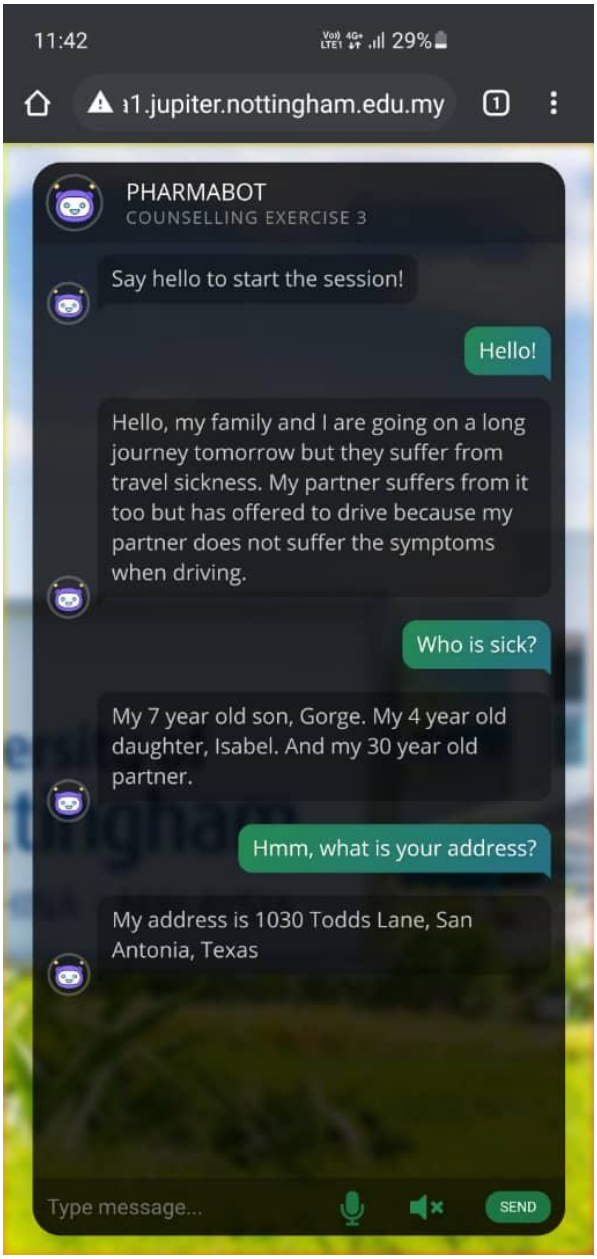


Figure 6.5.4

7. Implementation & Methodology

7.1 Initial Design

Based on the description provided by the client, the project output will be a chatbot that will be used by pharmacy students for training in advising patients.

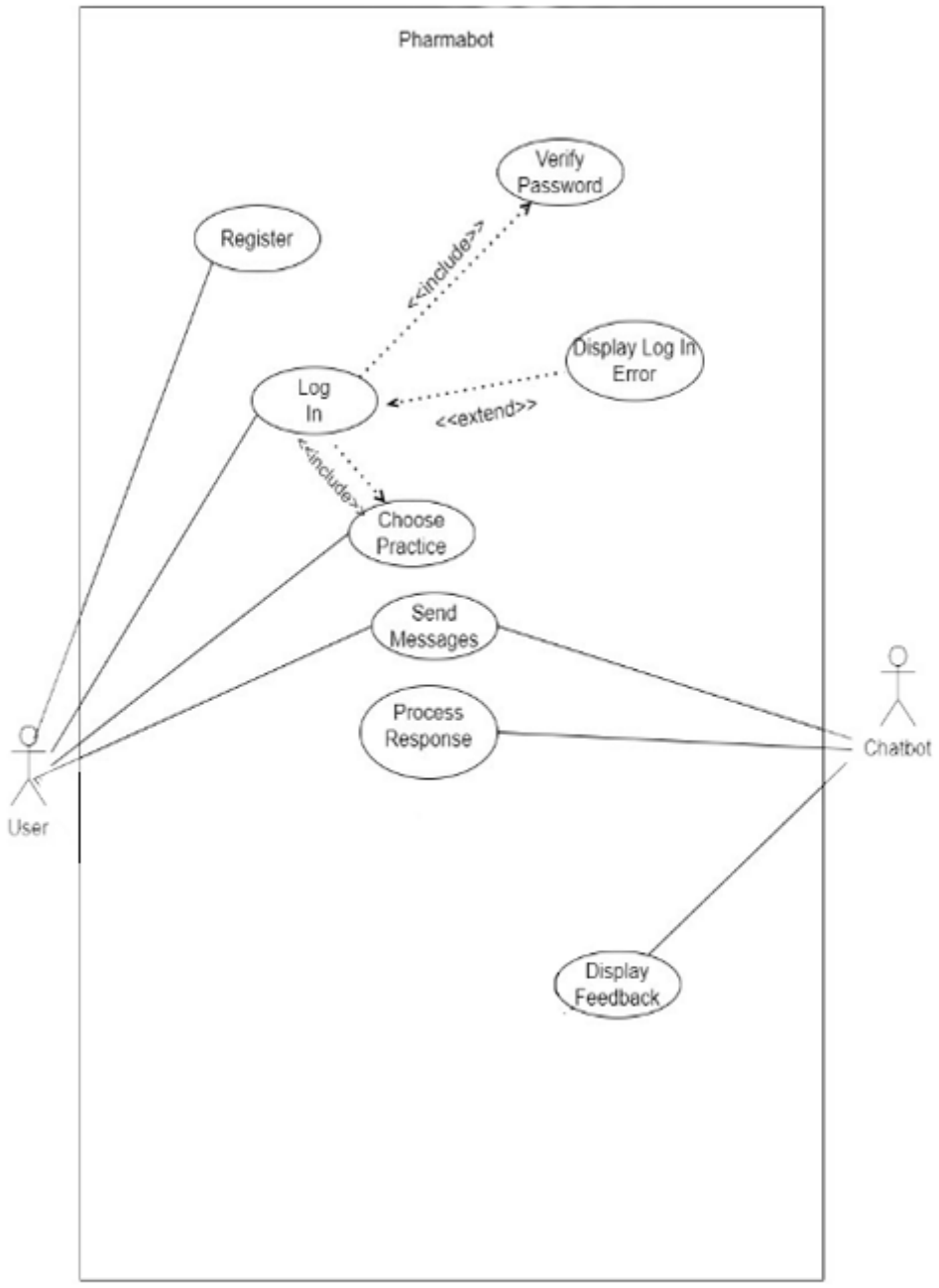
The main interface that we will be implementing will be an Instant Messaging (IM) interface due to the nature of the project, which is a virtual patient presented in the form of a chatbot. Other features such as user login and authentication, and practice question selection are also needed. We also would like the chatbot to be as accessible as possible, without the need to download any prerequisite software.

Based on these factors, the interface of the chatbot will be implemented as a website with a login page, training question selection page and Instant Messaging style page to interface with the chatbot, along with the appropriate navigation in between.

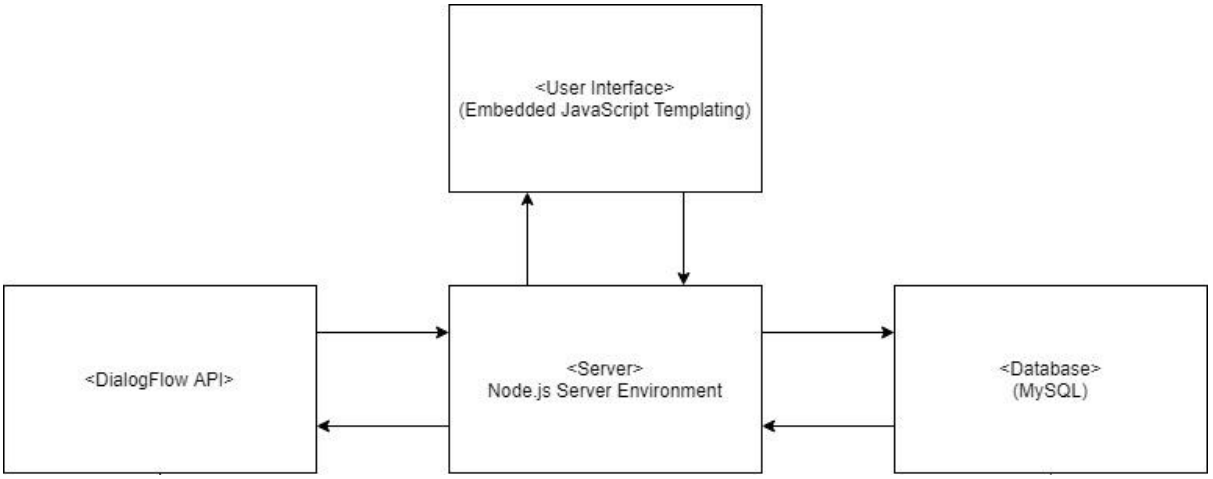
Besides that, the system should produce appropriate feedback for the students at the end of each practice for review purposes.

7.2 Use Case

Below is the expected use case of the project.



7.3 Architectural Design

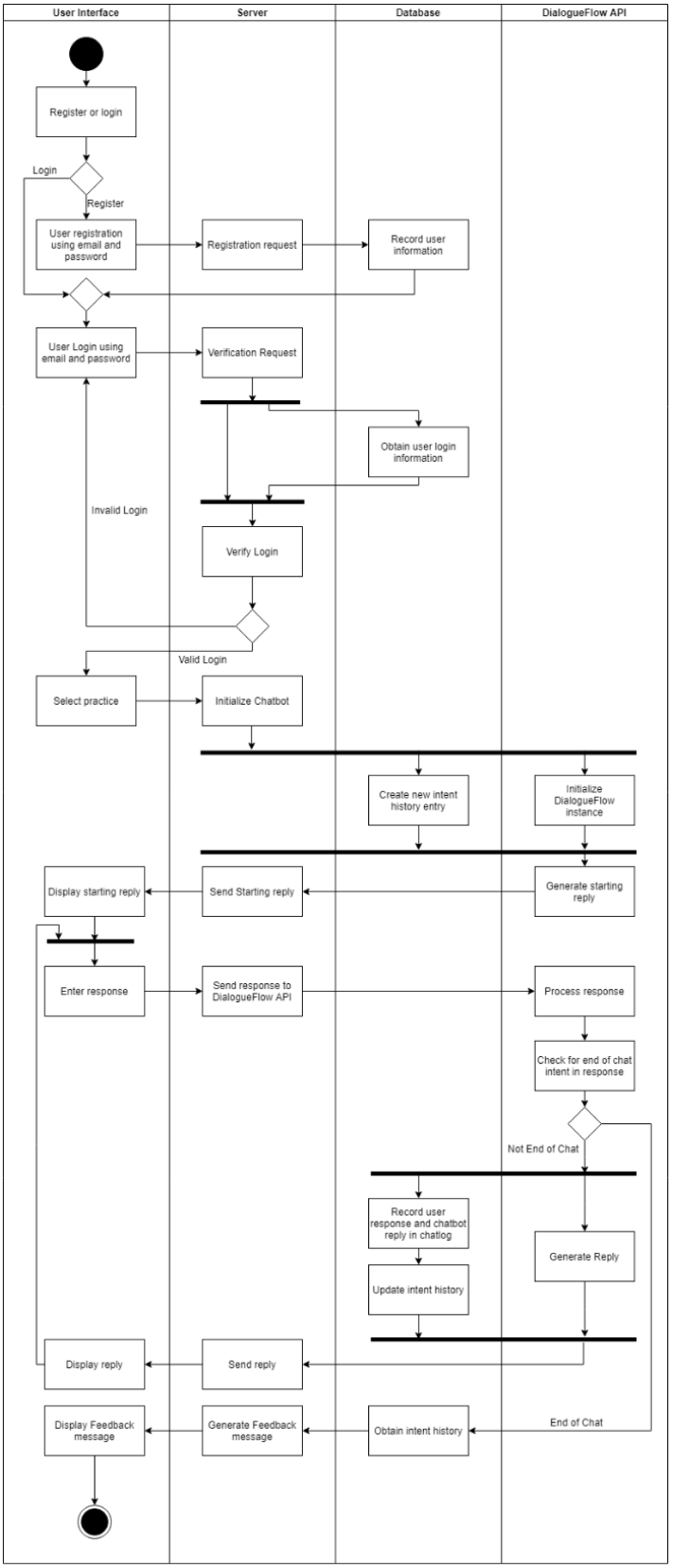


Our initial plan for the project was to use python's Chatterbot library to design a script for the chatbot. However after further research and client suggestion, Google's DialogFlow API was determined to be a better alternative and more suitable for the project as it has extensive language understanding and processing capabilities, and can produce the appropriate outputs to our users, including error handling when it cannot understand a message sent by the user. Utilizing DialogFlow as the basis for our chatbot, we then did more research in order to implement it into a website.

For our usage of the chatbot, as a virtual patient that provides automatic feedback at the end of a session, a database is also required to record the user input, in the form of question intent asked by the users that's derived by the DialogFlow API. Another database is also needed for login authentication. MySQL is selected as the database management system to fulfill this requirement as it is open source, suitable for this application, and the team has more experience utilizing it. Lastly, we selected Node.js as the server environment for our website, as it is able to integrate DialogFlow and MySQL both with relative ease utilizing DialogFlow Client API and MySQL driver packages for NodeJS. The interface of the chatbot is built on top of the server environment using Embedded Javascript Templating (EJS) that is typically utilized by Node.js. Additional Javascript scripts were also written and used for interfacing between the different main components of the above architecture. The above structure of components resulted in a Model-View-Controller (MVC) type architecture being implemented.

7.4 Activity diagram

Below is a swimlane activity diagram describing the planned flow from one activity to another. Link for higher fidelity diagram: https://drive.google.com/file/d/1WQdtikMINA0ARaJB_aWFSGooJgyhVEHE/view?usp=sharing

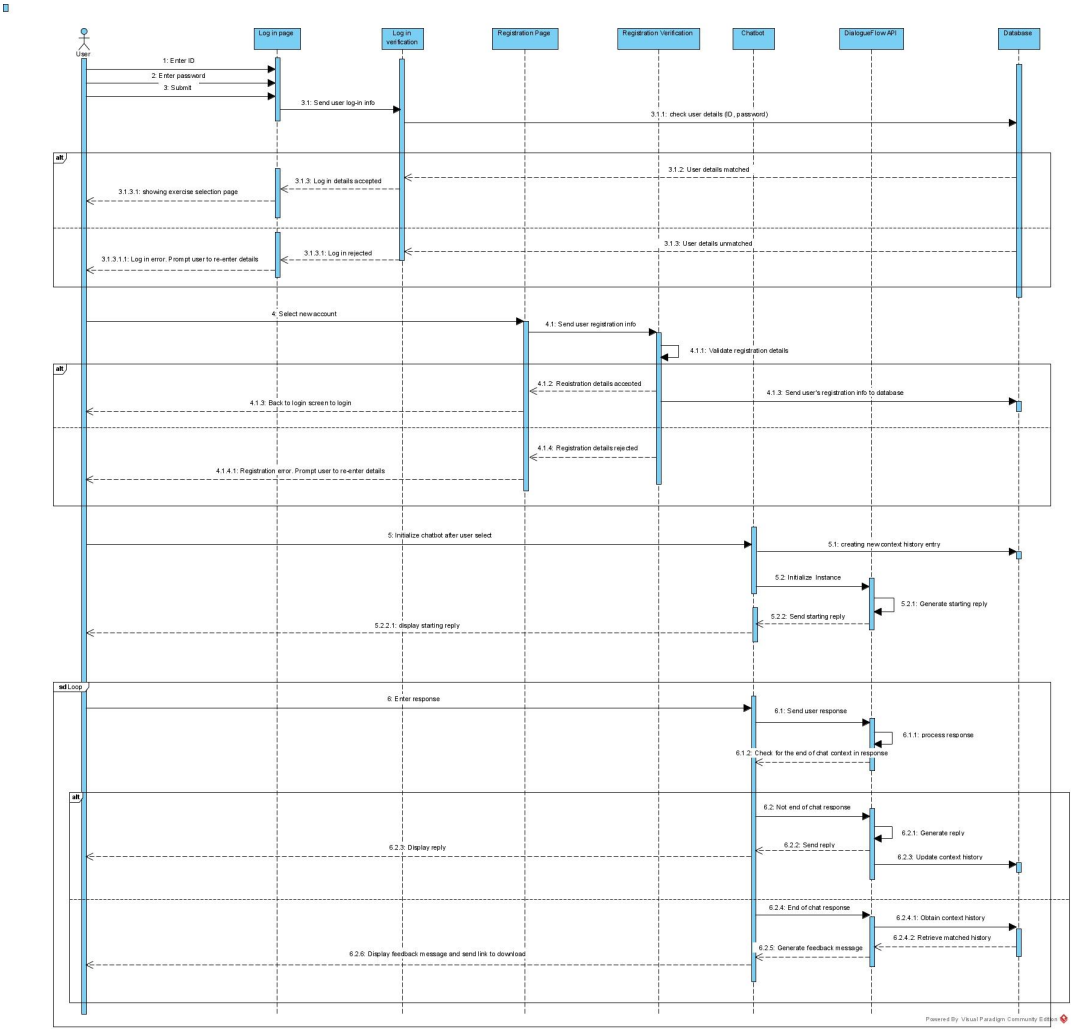


7.5 Sequence Diagram

Below is a sequence diagram describing the planned sequence flow of the system.

Link for higher fidelity diagram:

https://drive.google.com/file/d/1FOQrZIly-xcJqEn9KBMi1ycYlhXo-FpK/view?usp=sharing



7.6 Implementing DialogFlow for Conversation

7.6.1 Agents

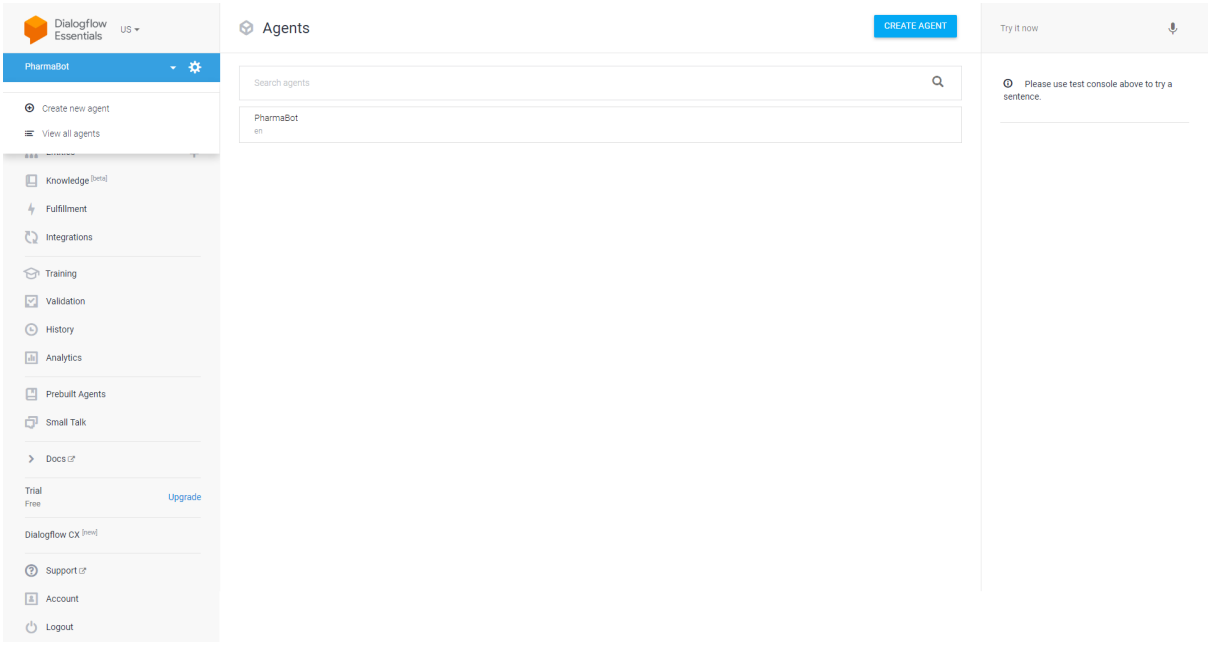


Figure 7.1

Our conversational process will be achieved through creating DialogFlow Agents which is a trained generative machine learning model that understands natural language. Agents consist of Intents, Entities, and other features to aid in this process of implementing conversation. All agents using a machine learning model will automatically learn and expand the phrase list as more users interact with the Agent. We have created an agent named “PharmaBot”, using English-US-USA(Global) as the primary language and (UTC+8) Asia/Kuala Lumpur as our default Time Zone.

7.6.2 Intents

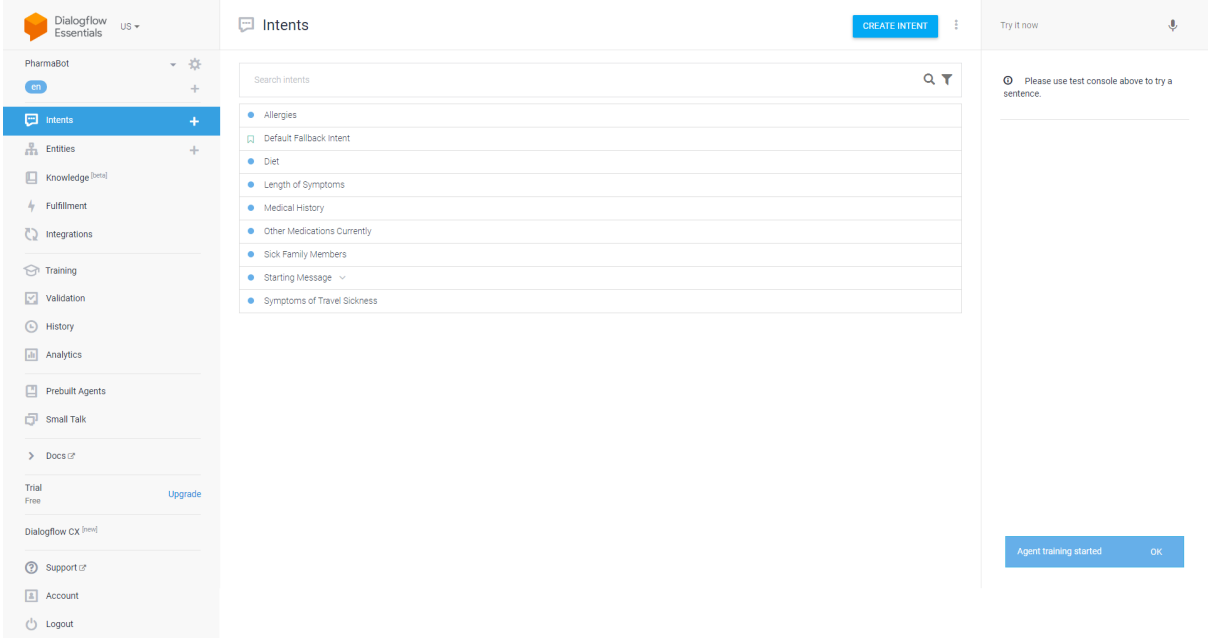


Figure 7.2

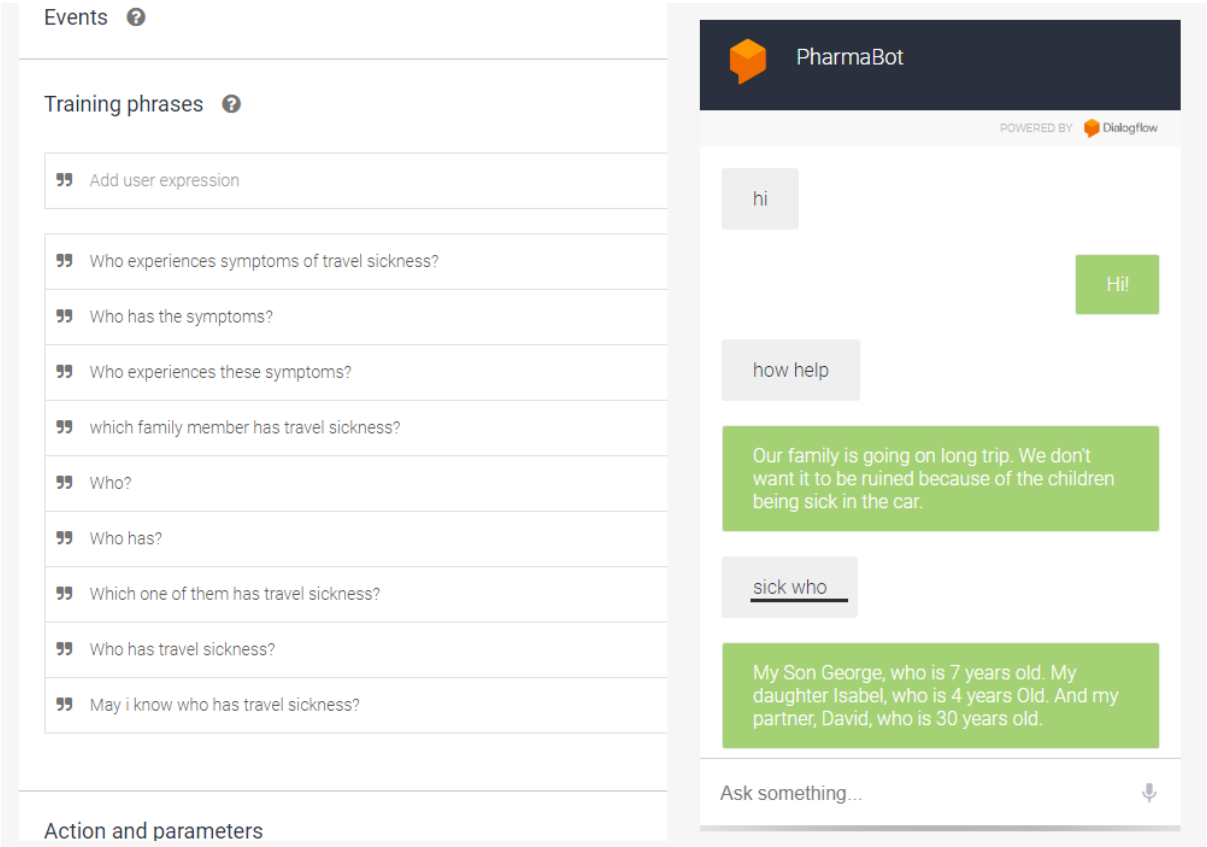


Figure 7.3

Intents are used for the DialogFlow Agents to recognize what the user wants and to understand what the users are saying. We will have to create intents for anything a user might request and each intent provides various examples in the way users might communicate, as shown in Figure 7.2. Just as their label suggests, “Starting Message” is used to give a welcoming message to users. “Default Fallback Intent” will be triggered when the chatbot does not understand what the user has input. The various other intents will be triggered depending on what the user says.

DialogFlow will use this information to train a machine learning model to understand not just the exact phrases we have entered but also numerous phrases which mean the same thing. As seen in Figure 7.3, even though the training phrase tab does not contain the words “sick who” , the “Sick Family Members” intent is still triggered, and the chatbot returns a response previously entered in the “Responses” tab from the “Sick Family Members” Intent.

7.6.3 Entities

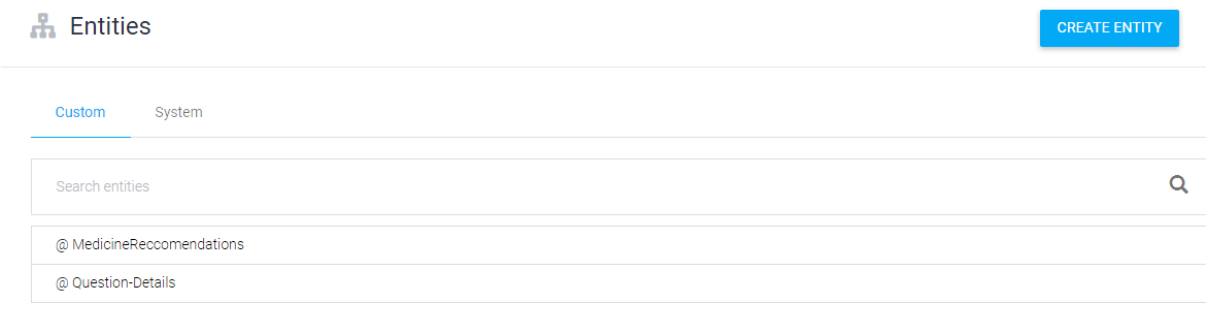


Figure 7.4

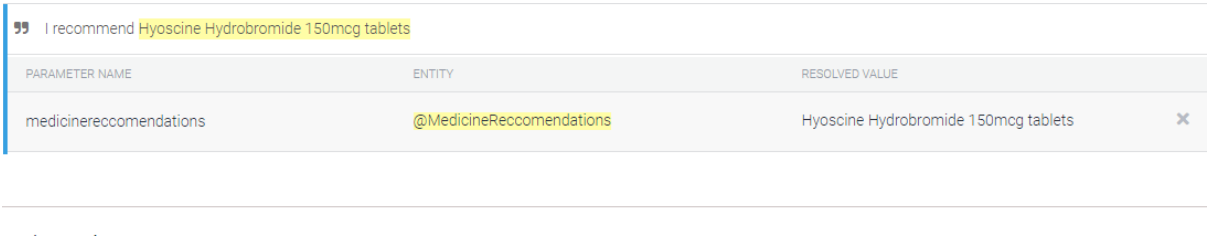


Figure 7.5

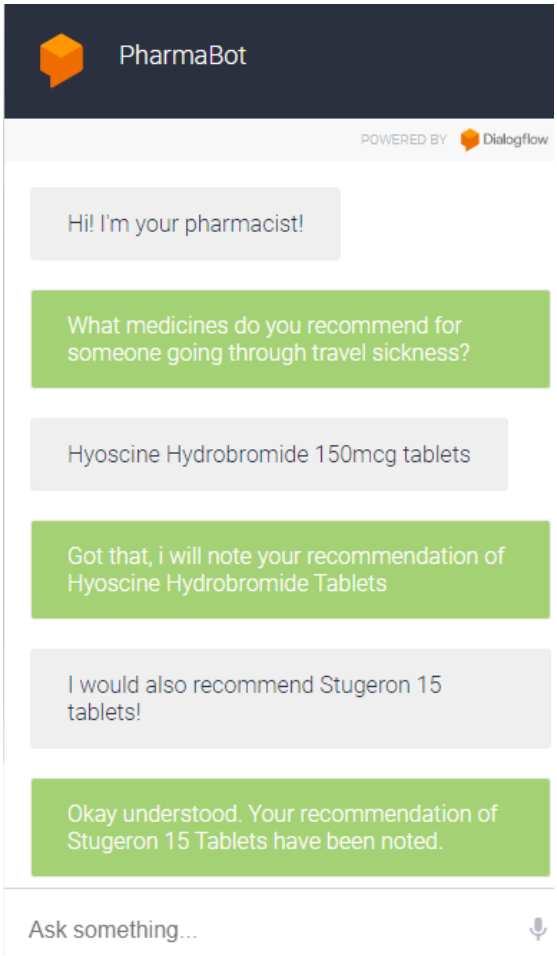


Figure 7.6

Entities are used to extract useful information from what users say to the DialogFlow agents. Such can be seen in **Figure 7.5** whereby the user's recommended prescription can be picked up within the statement and extracted as a parameter. We will want to provide a given list of words or phrases which matches the given concept and create Entities as seen in **Figure 7.4,** whereby we have defined a list of medicines and assigned them to an entity named @MedicineRecommendations. This is called Developer Entity where the entities we define are based on a list of words registered either through the DialogFlow console, API, and/or uploading JSON or CSV files. With Entities, we will be able to create flows of conversation as seen in **Figure 7.6**.

7.6.4 Fulfillment

To control DialogFlow conversations programmatically, we will be using their fulfillment feature. This allows us to use the parameter values to be implicated further within the backend. We do this by creating our web server which exposes a single http endpoint, namely webhook, where our custom logic will reside. After that, we enable fulfillment for any intents which require it, and give each intent an action

name that helps fulfillment know which intent is triggered. In the scenario that one of these intents are matched, DialogFlow will send a json request to our webhook that contains values of any entity that were extracted, what the user said, as well as the action name so that we may know which intent was matched.

When we call the DialogFlow API directly, we can provide custom information. In our webhook, we can use the custom information provided to access data stored, trigger any business logic we define, call API's, and generate responses. These responses created will be sent to the users through DialogFlow.

7.7 Feedback Generation

To generate the feedback after each counselling exercise, a script was implemented in Javascript to generate the appropriate feedback response to be sent to the front end. The script was implemented in 3 major sections, which are intent matching, intent sequence saving, and the generation of feedback messages.

7.7.1 Intent Matching

After the DialogFlow API processes the message sent by the user, the result generated which contains the response and intent detected is sent to be processed by the script. Additional information such as the ID for the counselling exercise chosen and session data was also passed to the script. Firstly, the counselling exercise ID was used to determine the relevant intent matching function (1 of 3 counselling exercises), along with the table name that represents the table to be used to save the intent sequences. The intent was then passed to the relevant matching function where the intent name was matched using a switch case, returning a column name to save the sequence values within the relevant table. The response generated from DialogFlow was returned as well if an end intent wasn't detected such that it can be sent to the front end as a response.

7.7.2 Intent Sequence Saving

The column name retrieved was then passed to another function along with relevant information such as session data and table name. The saving of intents work through a sequence of queries to the relevant table within the database. First, the session id was queried to determine if there is already an existing entry, if not a new entry will be created containing the session id and student id. Next the column was checked if it was already recorded with a sequence value, if it was, the function would terminate without updating the table. After determining that an update to the table was needed, the total intents sequence already saved that is represented by a column within the table will be queried to determine the next number within the sequence. The entry within the table is then updated with the total intents value incremented and the column having the next value within the sequence, resulting in database entries such as these.

sessionId	studentId	totalIntents	address	allergy	defaultWelcome	end	firstTime	folicAcid	furtherQuestions	keyCouns
1618671634	20089417	16	3	5	1	16	7	14	15	

7.7.3 Feedback Message Generation

When the end intent is detected during intent matching, the response generation is passed to functions called endhandlers where each counselling exercise has their own version due each of the counselling exercises being unique. The endhandler functions queries relevant table entries based on session ID to retrieve the sequence of intents detected. Strings that represent each intent are then appended based on their designated sequence number to produce the feedback message that successfully describes the actual sequence of intents inputted by the user. The feedback message is then returned by the script and sent to the front end for user review.

7.8 Google Cloud Platform, Dialogflow API and Server.js

In the user interface design section, the front end side functions for the chatbot is mentioned, but not the server and back end side.

```
async function runSample(userQuery, id, sessionData) {
var projectId

   if (id == 1){
 var fileName = require('./counselling-1-rlbm-838891a46377');
    projectId = 'counselling-1-rlbm';
  }else if (id == 2){
 var fileName = require('./counselling-2-cefa-eabaf3e74be8 ');
    projectId = 'counselling-2-cefa';

  }else if (id == 3){
 var fileName = require('./counselling-3-orlp-ffda2e217e8a');
    projectId = 'counselling-3-orlp';
  }


const credentials = {
    client_email: fileName.client_email,
    private_key: fileName.private_key,
};

const sessionClient = new dialogflow.SessionsClient(
    {
        projectId: projectId,
        credentials
    }
);

const sessionPath = sessionClient.projectAgentSessionPath(projectId, sessionId);

  // The text query request.
  const request = {
    session: sessionPath,
    queryInput: {
      text:  {
```

```
        // The query to send to the dialogflow agent
        text: userQuery,
        // The language used by the client (en-US)
        languageCode: 'en-US',
      },
    },
  };

  // Send request and log result
  const responses = await sessionClient.detectIntent(request);
  //console.log('Detected intent');

  const result = responses[0].queryResult;
  console.log(`${result.queryText}`);
  console.log(`${result.fulfillmentText}`);

    //Intent Matching
  if (result.intent) {
    var chatResponse =  intentMatcher.match(con, id, sessionData, result);
  } else {
    console.log(`  No intent matched.`);
  }

  return chatResponse;
}
```

Figure 7.8.1

To use the Dialogflow API, a project must first be created on Google Cloud Platform which gives us all the necessary files and details to do the webhook function, such as Project ID and private key. This is then connected to our system through the server.js file. After the connection is established and credentials are accepted, google gives us access to be able to send queries to the dialogflow agent project, and return the query results. As seen in **Figure 7.8.1,** we put all of these in the runSample() function.

```
app.post('/send-msg',(req,res)=>{
    console.log(req.body.botnum);

    var payload = getJWTPayload(req.cookies['cookie']);
    var sessionData = {studentId: payload['studentId'], sessionId: payload['iat']};
    //console.log(sessionData);
    runSample(req.body.MSG, req.body.botnum, sessionData).then(data=>{
        res.send({Reply:data})
    })
})
```

Figure 7.8.2

When we fill in and send the messages in our chatbot page through the form which is mentioned previously in the “User Interface Design” section of this report, a post request will be sent and the runSample() function is called, therefore carrying out the functions previously mentioned in **Section 6.4,** and the user is able to converse with the chatbot.

8. Time Plan

8.1 Autumn Semester

We started the project slow as we are still getting used to the online classes. After the third week, we realised we need to speed up on the work to catch up on the project so we have stepped up and made sure we will not be behind schedule again. Eventually, we made up to our project ahead of time and we have spare time to amend any minor issues.

Burndown Chart

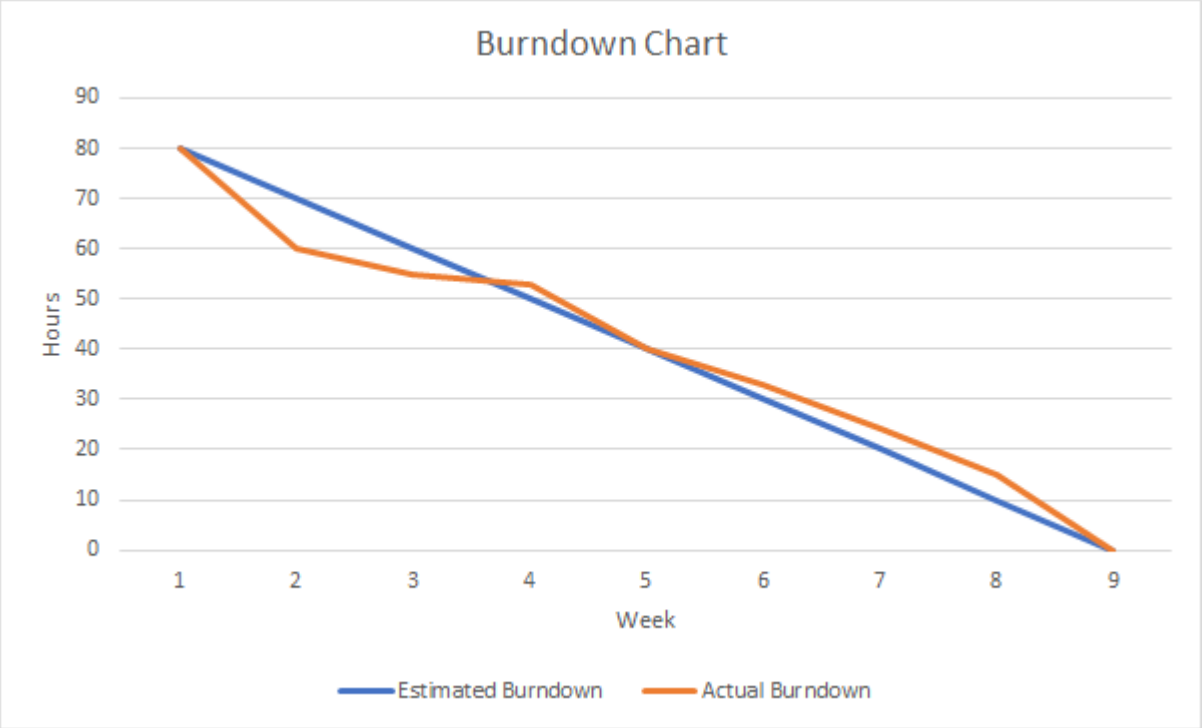


Figure 8.1.1

Gantt Chart

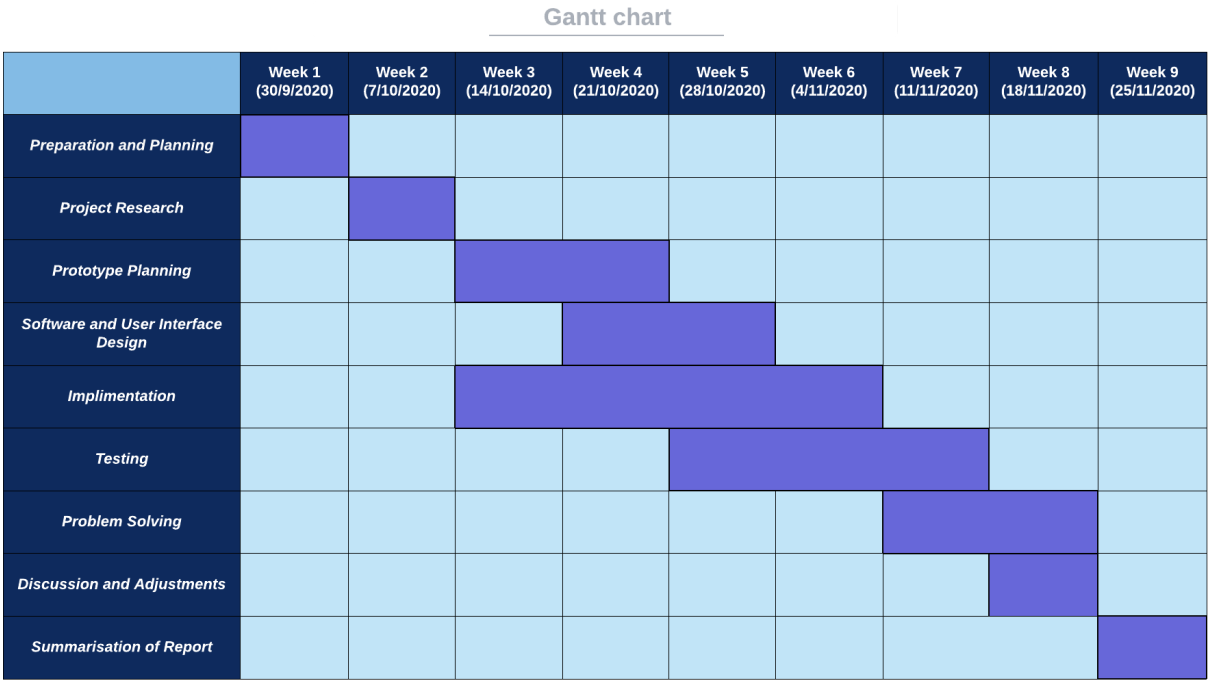


Figure 8.1.2

Detailed Task Description

Week 1 (Preparation and Planning)	We were assigned to our groups and we created a whatsapp group and microsoft teams group for easier communication between group members.
Week 2 (Project Research)	Our group had our first meeting with the client and supervisor. We listened to our client's request and requirements and proceeded with our research on it.
Week 3 (Prototype Planning)	After doing some research, our group members came back together for a meeting to discuss our prototype and plan ahead what needed to be done.
Week 4 (Software and user interface design)	We came up with a prototype using our desired software. Then, we started to design our user interface according to the requirements.
Week 5 (Implementation)	We implemented our prototype on a host and linked it with our database.
Week 6 (Testing)	We tested the prototype to ensure all the requirements work accordingly.
Week 7 (Problem Solving)	We checked for any issues with the prototype and made necessary tweeks to solve the problems.
Week 8 (Discussion and adjustments)	We had some discussions on the issues we faced whether it's on the prototype or the report and made appropriate changes to it.
Week 9 (Summarisation of report)	We summarised our report and sent it to our supervisor to check before submitting.

Figure 8.1.3

8.2 Spring Semester

Learning from the mistakes from the previous semester, we were better in our time management on the project this semester. Although there was a slight drop in week 6 to week 8 due to the release of coursework from other courses, we still managed to pull ourselves ahead and finish the project ahead of time.

Burndown Chart

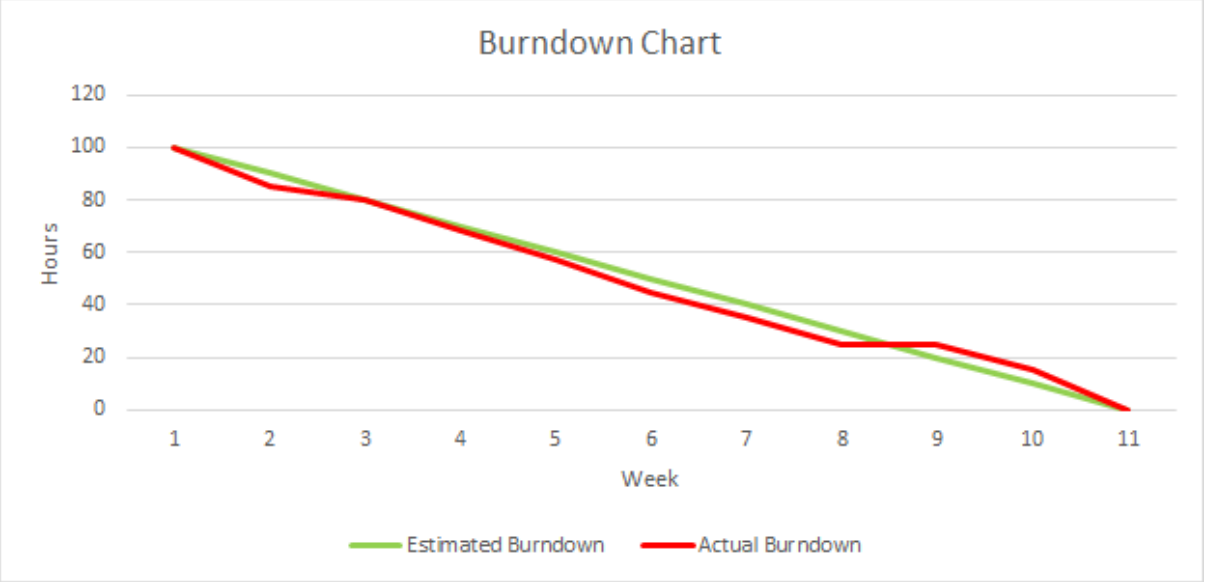


Figure 8.2.1

Gantt Chart

	Week 1 (27/1/2021)	Week 2 (3/2/2021)	Week 3 (24/2/2021)	Week 4 (3/3/2021)	Week 5 (10/3/2021)	Week 6 (17/3/2021)	Week 7 (24/3/2021)	Week 8 (31/3/2021)	Week 9 (7/4/2021)	Week 10 (14/4/2021)	Week 11 (23/4/2021)
Progress Update											
Prototype Development											
Functionality Implementation											
Testing											
Design Improvement											
Final Review											
Report Writing and Presentation											

Figure 8.2.2

9. Group Organization

9.1 Individual Roles

6 members involved in this project are Yong Teng, Khye Yueh, Ren Jin, Jia Hau, Steven and Alya. Jia Hau was elected as the leader through voting. During the first meeting, every member had to vote for someone who is capable of becoming a leader, and the most voted person is chosen. The core responsibility of becoming a leader is providing clear and transparent instructions to group members as well as addressing any concerns to avoid any task conflicts. Besides, the leader is in-charge of any group project submissions, setting up meetings, keeping track of everyone's progress and making sure everyone contributes to the project.

As for the other members, everyone needs to be committed to team objectives by executing the tasks assigned to them by giving quality outcomes in a given timeframe. In order to work effectively, all team members have to analyse carefully their own roles so that the whole team can benefit from using each other's strengths to best advantage.

During the Autumn semester, even though everyone is participating in the report, there are a few job scopes that were generally divided. Yong Teng and Steven was handling the database, Alya and Khye Yueh was in charge in front end design(UI/UX), Jia Hau and Ren Jin were in charge of DialogFlow chatbot training and everyone had a part in handling the webhook, fulfilment as well as the custom logic.

However during the Spring Semester, a few unforeseen circumstances caused a need for reallocation of job scopes, namely the increase in workload for certain jobs and the absence of Khye Yueh during the semester. After the reallocation of job scopes discussed during a meeting, Alya and Jia Hau was in charge of front end design (UI/UX), Jia Hau, Ren Jin and Yong Teng was in charge of DialogFlow chatbot training and implementation, Yong Teng and Steven was in charge of CPanel terminal implementation, lastly debugging as well as database management was handled by everyone present.

9.2 Workload Division

Leader has to make sure everyone has a fair amount of work to do. Before dividing the workload among members, having a small meeting is required to ease the division of work by listing all the tasks that need to be completed in a specific timeframe. Subsequently, each member assigned the tasks themselves based on

their strengths, capabilities and interests. To always keep track of each other, all members will be having another meeting to go through each other's progressions.

9.3 Meeting System

Meetings can be divided into two groups which are formal and informal. In the formal meeting, all members should attend the meeting together with the client. Supervisor is always welcome to join as well. It holds for only 15 to 30 minutes. In this meeting, members need to update the current progress while the client will give some feedback or list out a few suggestions that need to be changed or improved.

Whereas, informal meeting is a meeting that is attended by only group members once a week and it usually takes around 2 hours. The objective of having this meeting is to give feedback or suggestions to improve the tasks that have been completed which can benefit in keeping track of each other without leaving a person behind. If there are things that need to clarify, everyone needs to assist and help one another. Openness to ideas and input from others by sharing knowledge, ideas and information can help finding the best solutions. This meeting also includes planning future tasks and delegate them fairly so that everyone is working on the project evenly.

10. Discussion

Training pharmacy students is our scope of this project as it's able to lead a more modern path for learning communication with their patients. What we have done so far is an interface for the chatbot using DialogFlow as it's giving sample responses to what we have coded. For now, our chatbot is limited to the text response rather than adding speech recognition for our client to do so. As we are expecting that students' questions will be added into our chatbot database as chatbot history so that the lecturer can receive it and know the performance of a participant. Furthermore, the feedback from the clients can make us improve more as the code can be better. Our project has progressed smoothly as our system has started to build up.

11. Conclusion

In the Autumn Semester, we envision this chatbot to be widely used amongst pharmacy students as there is an amount of potential within its functionality and practicality. At this stage of having a working prototype, we believe the scalability of this chatbot is considerable if done properly. However, there are some weaknesses found in the preliminary model and there are room for improvements in the upcoming semester which we will definitely be working on. Besides picking up the pace to implement more functionality and to reach our client's expectations, we worked together towards a goal of optimizing our system's architecture and framework to be able to implement the necessary features and more.

As what we have done in Spring Semester, the prototype that we planned is now able to function and work as a formal training chatbot. It now has a lot of functions and is user friendly. The client has been satisfied with our product shown and has also evidently exceeded their expectations.

12. Summary of Achievements

We are happy to note that we were able to achieve close to all of the requirements specified for our project, with some extras.

As specified in the requirements, we managed to fulfil all of the functional requirements. Before the user starts a conversation, the chatbot sends a message on how to start communicating with the bot. The chatbot will respond and greet the user when a user initiates a conversation and chat bot will reply in a polite manner while conversing with users as we programmed the response of the chatbot. As for the user authentication, the user needs to register with a verified email before starting a session. When they are done conversing with the chatbot, their chat history will be saved and admins are able to retrieve and review it easily, because the session is tagged to their student ID. Students are also able to log back in using the registered credentials, as their registration details are stored in the database's "user" table. Besides, users will be allowed to choose exercises. After registering or logging in an account, users need to choose one out of three counselling exercises before proceeding with the chatbot. The chatbot utilizes Natural Language Processing(NLP) which identifies keywords, entities and intents to respond accordingly. For error handling, the chatbot has a default fallback intent which prompts users to repeat or reword the query when their intent is unable to be identified. Lastly, the chatbot is able to assess and identify the response from users and provide feedback based on the sequencing and questions asked.

As for the non-functional requirements, user log-in details are stored securely within the database and only the client is able to view the information. Even though the

personal data (name, email and student id) can be visible in the database, the password of users has been hashed. Chat logs between chatbot and user will be private and confidential as well. Only the client has the access to view the conversations in the DialogFlow. For the chatbot response rate, we stated that the chatbot will issue a reply within 3 seconds. Unfortunately, if we are using a server using CPanel's terminal, the response of the chatbot can sometimes be more than 5 seconds. But if we run using localhost, the chatbot successfully issues a response within 3 seconds.

Although all the functional requirements have been met, one of the non-functional requirements has not been verified. Specifically the chatbot response rate as we have no control on the server.

13. Reflective Comments

Pharmabot is a counselling chatbot that is able to converse and give final feedback whether the students have asked the correct questions and prescribed the correct answers in a given sequence. This chatbot aims to be an exercise that will train pharmacy students' prescription and counselling skills. However, amid this project, the main problem faced by the team was lack of technical knowledge in the field of Artificial Intelligence Chatbot and Web Development.

With the finalization of the project, we had successfully achieved the aims of the module, which includes gaining software engineering experience as well as gaining group working experience. Regardless of lacking some technical knowledge, we were able to get our implementation working after doing some research on the project with the help of team members' cooperation.

The completion of Pharmabot started off by delegating work among team members. Tasks were distributed based on each member's capabilities and interests. However, everyone needs to pull together each other's ideas and suggestions to promote greater understanding so that everyone can keep the group headed towards the goal. For the report, small sections were listed out first before team members chose their desired sections. Leader is in-charge of making sure tasks are divided evenly and keeping track of everyone's contribution to this project. Due to this pandemic, meetings can only happen virtually via Microsoft Teams. Meeting among members happens once a week mainly to discuss the progression of the project while meeting with a supervisor happens when members are having problems or difficulties while doing the project. Besides, a meeting with the client will be scheduled to show the performance of Pharmabot as well as receive some comments on what needs to be added or upgraded.

The accomplishment of Pharmabot was handled systematically and proudly to say that we managed to achieve a simulated patient in the form of chatbot that met

almost all of the functional and non-functional requirements as well as fulfilled client's requirements.

14. Appendix

14.1 Test Case

#	Action	Expected Response	Pass/Fail
1	Visit the Pharmabot website	Login page is shown	Pass
2	Enter registered student id and password correctly	No error will be shown	Pass
	Log in without or wrong student id	'Invalid student id or password' message will pop-up	Pass
	Log in without or wrong password	'Invalid student id or password' message will pop-up	Pass
	Click login button after entering with correct credentials	Select exercises page is shown	Pass
3	Click the word 'Register'	Register page shown	Pass
	Register without fill in user's email or enter gibberish email address	'Please enter valid email' will popped up	Pass
	Enter password that do not match in the 'Confirm Password'	The passwords entered do not match, please try again' message will pop-up	Pass
	Register without tick the terms and conditions box	'Please check this box if you want to proceed' message will pop-up	Pass
	Click register button after registering	Directed to Login page	Pass
4	Encrypt password	Encrypted password at database	Pass

6	Check user's details in the database	Name, student id, email and encrypted password shown in database	Pass
7	Select Exercise 1	User is redirected to counselling exercise 1 chatbot	Pass
	Select Exercise 2	User is redirected to counselling exercise 2 chatbot	Pass
	Select Exercise 3	User is redirected to counselling exercise 3 chatbot	Pass
8	User asks specific questions	The bot will reply with intended response which matches intent settings within the dialogflow dashboard	Pass
9	User inputs dummy texts	Bot will prompt user with a default fallback intent	Pass
10	User type 'end' or 'goodbye'	Feedback will be shown based on the sequencing of user's replies.	Pass
11	Check chat log at DialogFlow console	Chat history is shown together with the timestamp	Pass

14.2. Meeting Minutes

Software Engineering Group Project Meeting		
Date: 21 October 2020	Time: 15:00	Location: Microsoft Teams

Meeting called by	Ms. Neoh Siew Chin
Type of meeting	Formal
Note taker	Lee Ren Jin
Timekeeper	N/A
Attendees	1. Ang Jia Hau 2. Khor Yong Teng 3. Lee Ren Jin 4. Steven Ho Chu Leong 5. Alya Amirah Muhammad Saufee 6. Law Khye Yueh
AGENDA: Meeting with Client	
Contents	· Meeting with client · Found out about details of project · Understand the requirements and request from client

Action Items	Person Responsible	Deadline
· Note down requirements for project · Research about project	Everyone	4th November 2020

Software Engineering Group Project Meeting		
Date: 4 November 2020	Time: 15:00	Location: Microsoft Teams

Meeting called by	Ang Jia Hau
Type of meeting	Formal
Note taker	Lee Ren Jin
Timekeeper	N/A

Attendees	1. Ang Jia Hau 2. Khor Yong Teng 3. Lee Ren Jin 4.Steven Ho Chu Leong 5. Alya Amirah Muhammad Saufee 6. Law Khye Yueh
AGENDA: Assign roles for prototype and task for interim report	
Contents	· Assign roles · Distribute task for interim report

Action Items	Person Responsible	Deadline
· Do research on chatbot · Research on our task	Everyone	18 November 2020

Software Engineering Group Project Meeting		
Date: 18 November 2020	Time: 15:00	Location: Microsoft Teams

Meeting called by	Ang Jia Hau
Type of meeting	Formal
Note taker	Lee Ren Jin
Timekeeper	N/A
Attendees	1. Ang Jia Hau 2. Khor Yong Teng 3. Lee Ren Jin 4.Steven Ho Chu Leong 5. Alya Amirah Muhammad Saufee 6. Law Khye Yueh
AGENDA: Finalize interim report	

Contents	· Combine parts of interim report by each member · Finalize interim report

Action Items	Person Responsible	Deadline
· Combine interim report · Double-check for errors in report	Everyone	2 December 2020

Software Engineering Group Project Meeting		
Date: 2 December 2020	Time: 15:00	Location: Microsoft Teams

Meeting called by	Ang Jia Hau
Type of meeting	Formal
Note taker	Lee Ren Jin
Timekeeper	N/A

Attendees	1. Ang Jia Hau 2. Khor Yong Teng 3. Lee Ren Jin 4.Steven Ho Chu Leong 5. Alya Amirah Muhammad Saufee 6. Law Khye Yueh
AGENDA: Submit interim report	
Contents	· Submit interim report

Action Items	Person Responsible	Deadline
· Submit interim report	Jia Hau (submission), Everyone	16 December 2020

Software Engineering Group Project Meeting		
Date: 16 December 2020	Time: 15:00	Location: Microsoft Teams

Meeting called by	Ang Jia Hau
Type of meeting	Formal
Note taker	Lee Ren Jin
Timekeeper	N/A
Attendees	1. Ang Jia Hau 2. Khor Yong Teng 3. Lee Ren Jin 4.Steven Ho Chu Leong 5. Alya Amirah Muhammad Saufee
AGENDA: Work on prototype	
Contents	· Further improve prototype

Action Items	Person Responsible	Deadline
· Update and work on prototype	Everyone except Law Khye Yueh	30 December 2020

Software Engineering Group Project Meeting		
Date: 30 December 2020	Time: 15:00	Location: Microsoft Teams

Meeting called by	Ang Jia Hau
Type of meeting	Formal
Note taker	Lee Ren Jin
Timekeeper	N/A

Attendees	1. Ang Jia Hau 2. Khor Yong Teng 3. Lee Ren Jin 4.Steven Ho Chu Leong 5. Alya Amirah Muhammad Saufee
AGENDA: Research on project	
Contents	· Research on dialogflow and sql

Action Items	Person Responsible	Deadline
· Learn about dialogflow and sql	Everyone except Law Khye Yueh	27 January 2021

Software Engineering Group Project Meeting		
Date: 27 January 2021	Time: 15:00	Location: Microsoft Teams

Meeting called by	Ang Jia Hau
Type of meeting	Formal
Note taker	Lee Ren Jin
Timekeeper	N/A
Attendees	1. Ang Jia Hau 2. Khor Yong Teng 3. Lee Ren Jin 4.Steven Ho Chu Leong 5. Alya Amirah Muhammad Saufee
AGENDA: Progress update	
Contents	· Update on the research made · Plan timeline for project · Assign tasks for chatbot and database

Action Items	Person Responsible	Deadline
· Train chatbot · Create database	Everyone except Law Khye Yueh	17 February 2021

Software Engineering Group Project Meeting		
Date: 17 February 2021	Time: 15:00	Location: Microsoft Teams

Meeting called by	Ang Jia Hau
Type of meeting	Formal
Note taker	Lee Ren Jin
Timekeeper	N/A
Attendees	1. Ang Jia Hau 2. Khor Yong Teng 3. Lee Ren Jin 4.Steven Ho Chu Leong 5. Alya Amirah Muhammad Saufee

AGENDA: Work on frontend, backend and database	
Contents	· Research on NodeJS · Resume work on database

Action Items	Person Responsible	Deadline
· Implement NodeJS · Work on SQL	Everyone except Khye Yueh	3 March 2021

Software Engineering Group Project Meeting		
Date: 17 February 2021	Time: 15:00	Location: Microsoft Teams

Meeting called by	Ang Jia Hau
Type of meeting	Formal
Note taker	Lee Ren Jin
Timekeeper	N/A

Attendees	1. Ang Jia Hau 2. Khor Yong Teng 3. Lee Ren Jin 4.Steven Ho Chu Leong 5. Alya Amirah Muhammad Saufee
AGENDA: Implement functionality of admin page	
Contents	· Work on the functionality of admin page · Research on Express library

Action Items	Person Responsible	Deadline
· Code functions of admin page · Research on linking frontend with backend	Everyone except Law Khye Yueh	3 March 2021

Software Engineering Group Project Meeting		
Date: 3 March 2021	Time: 15:00	Location: Microsoft Teams

Meeting called by	Ang Jia Hau
Type of meeting	Formal
Note taker	Lee Ren Jin
Timekeeper	N/A
Attendees	1. Ang Jia Hau 2. Khor Yong Teng 3. Lee Ren Jin 4.Steven Ho Chu Leong 5. Alya Amirah Muhammad Saufee
AGENDA: UI Design	
Contents	· Decide on the UI design of interface · Implement Express

Action Items	Person Responsible	Deadline
· Design the interfaces of admin pages and chatbot · Link frontend with backend	Everyone except Khye Yueh	17 March 2021

Software Engineering Group Project Meeting		
Date: 17 March 2021	Time: 15:00	Location: Microsoft Teams

Meeting called by	Ang Jia Hau
Type of meeting	Formal
Note taker	Lee Ren Jin
Timekeeper	N/A
Attendees	1. Ang Jia Hau 2. Khor Yong Teng 3. Lee Ren Jin 4.Steven Ho Chu Leong 5. Alya Amirah Muhammad Saufee

AGENDA: Testing of software	
Contents	· Bug fixes and testing of software

Action Items	Person Responsible	Deadline
· Test chatbot and it's functionality	Everyone except Law Khye Yueh	31 March 2021

Software Engineering Group Project Meeting		
Date: 31 March 2021	Time: 15:00	Location: Microsoft Teams

Meeting called by	Ang Jia Hau
Type of meeting	Formal
Note taker	Lee Ren Jin
Timekeeper	N/A

Attendees	1. Ang Jia Hau 2. Khor Yong Teng 3. Lee Ren Jin 4.Steven Ho Chu Leong 5. Alya Amirah Muhammad Saufee
AGENDA: Report writing and setup client meeting	
Contents	· Assign task for final report · Setup client meeting

Action Items	Person Responsible	Deadline
· Set date for meeting with client to show progress · Prepare for final report writing	Everyone except Law Khye Yueh	14 April 2021

Software Engineering Group Project Meeting		
Date: 14 April 2021	Time: 15:00	Location: Microsoft Teams

Meeting called by	Ang Jia Hau
Type of meeting	Formal
Note taker	Lee Ren Jin
Timekeeper	N/A
Attendees	1. Ang Jia Hau 2. Khor Yong Teng 3. Lee Ren Jin 4.Steven Ho Chu Leong 5. Alya Amirah Muhammad Saufee
AGENDA: Meeting with client and prepare for presentation	
Contents	· Show project progress to client · Prepare for presentation

Action Items	Person Responsible	Deadline
· Present software and functionality to client · Do presentation slides	Everyone except Law Khye Yueh	21 April 2021

Software Engineering Group Project Meeting		
Date: 21 April 2021	Time: 15:00	Location: Microsoft Teams

Meeting called by	Ang Jia Hau
Type of meeting	Formal
Note taker	Lee Ren Jin
Timekeeper	N/A
Attendees	1. Ang Jia Hau 2. Khor Yong Teng 3. Lee Ren Jin 4.Steven Ho Chu Leong 5. Alya Amirah Muhammad Saufee

AGENDA: Finalize final report to submit and practice presentation	
Contents	· Finalize final report and submit · Practice for presentation and demo

Action Items	Person Responsible	Deadline
· Update final report · Submit final report · Do trial run for presentation and demo	Jia Hau(Submission), Everyone except Law Khye Yueh	21 April 2021

14.3. User Manual

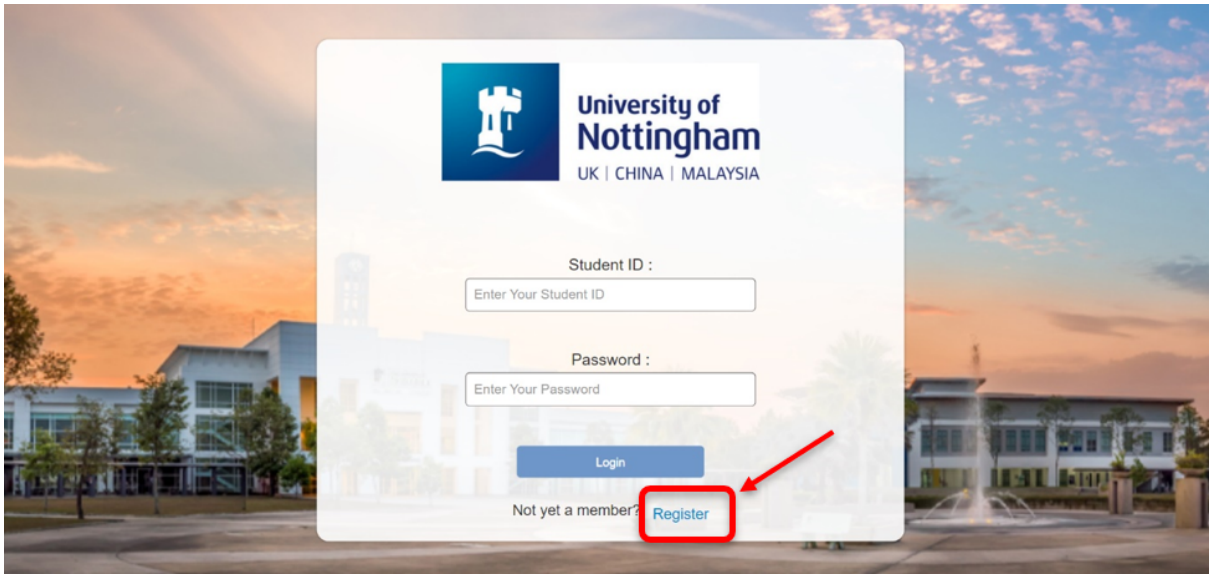
Step 1: go to http://hcyja1.jupiter.nottingham.edu.my/

Step 2.1:

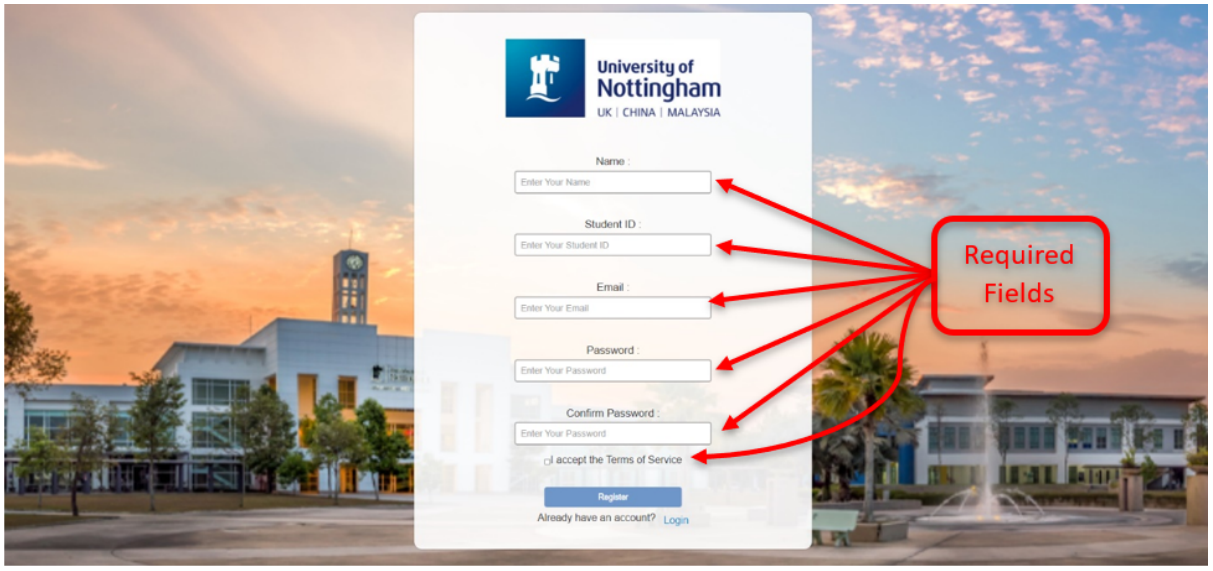
- If student is an existing user, then login with credentials (Student ID and Password)

Step 2.2:

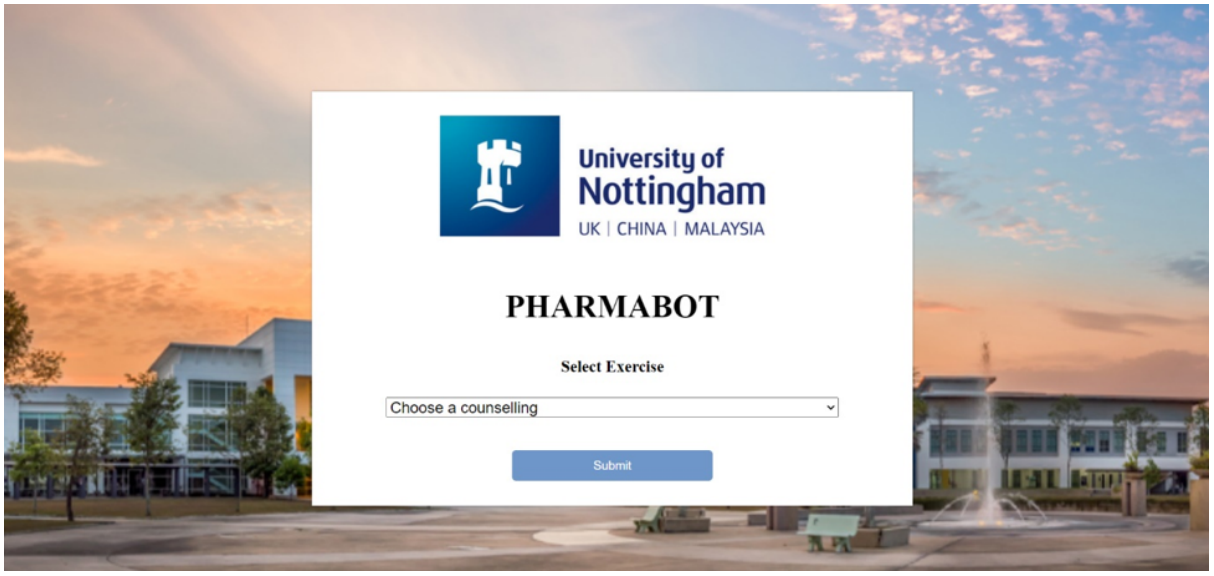
- If student is a new user, then click on “Register”



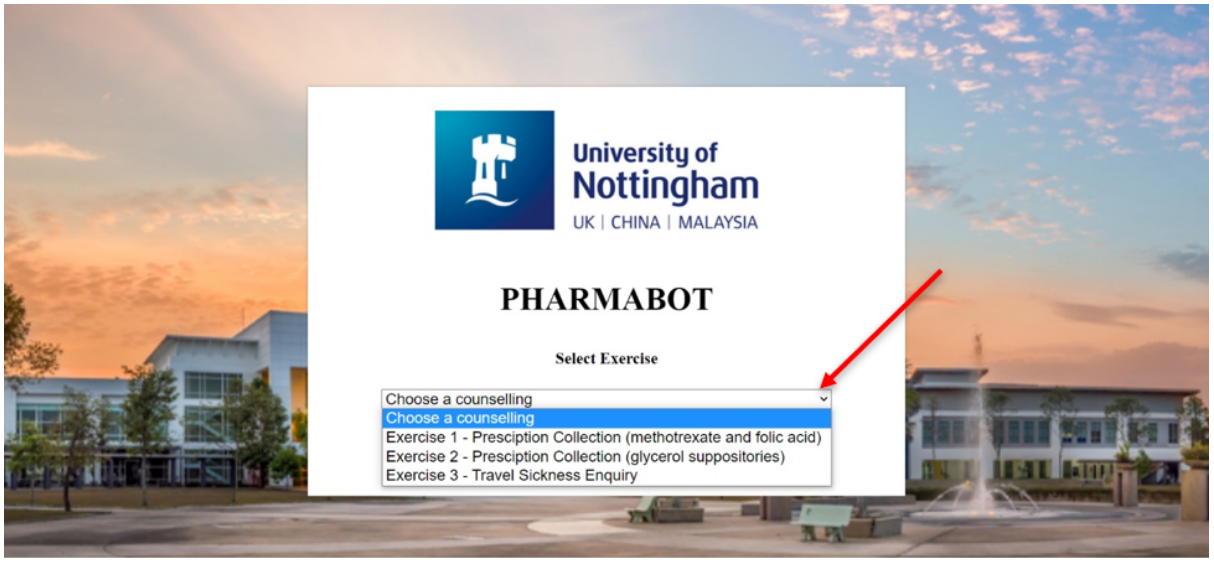
After clicking register, users will be redirected to the register page as seen below. Fill out the required fields and accept the Terms of Service. Once the user has filled it all up, click on the “Register” button. Upon successful registration, return to step 2.1 and log in.



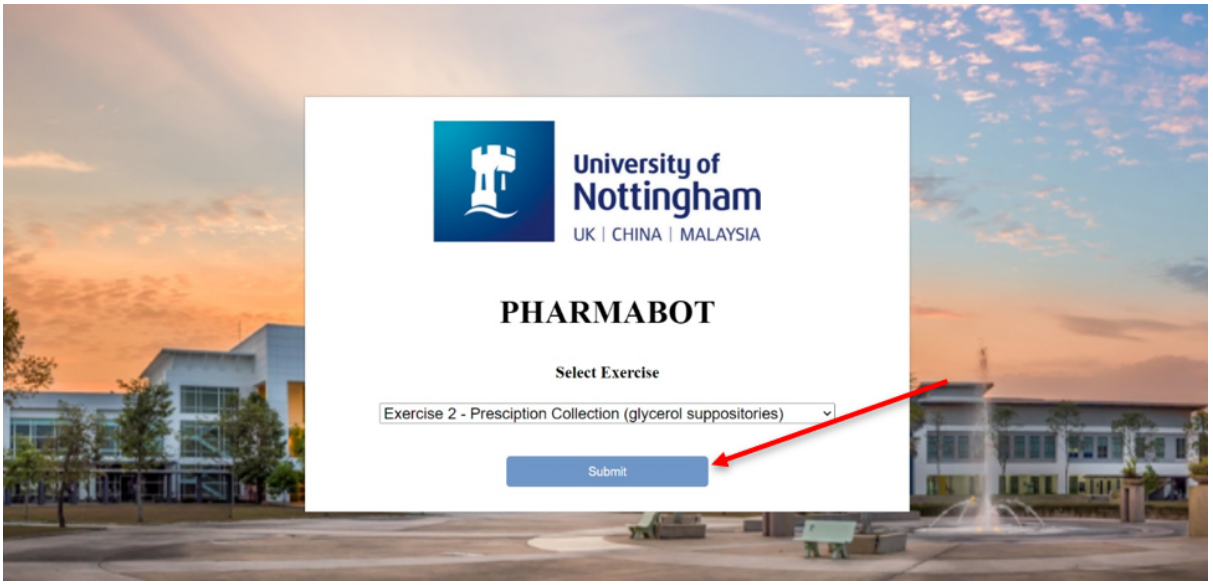
Step 3: After logging in successfully, the user will be redirected to a page to choose the desired counselling exercise.



Step 4: User can click on the dropdown menu to select counselling exercise

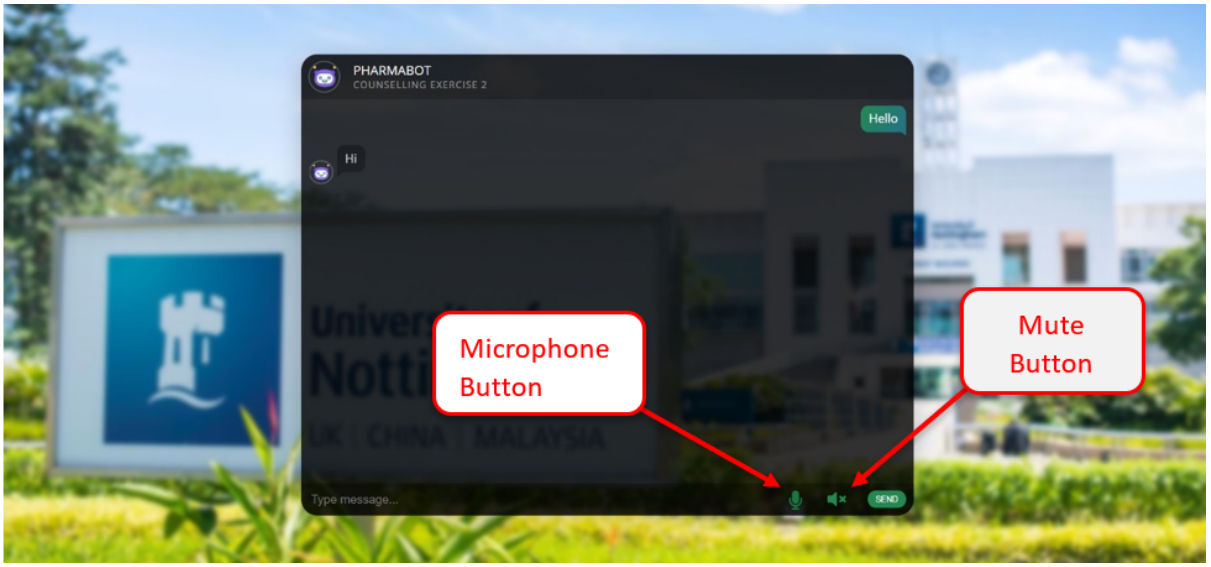


Step 5: After selecting the desired counselling exercise, the user will need to click on the “submit” button to initiate the exercise and will be directed to the chatbot page.

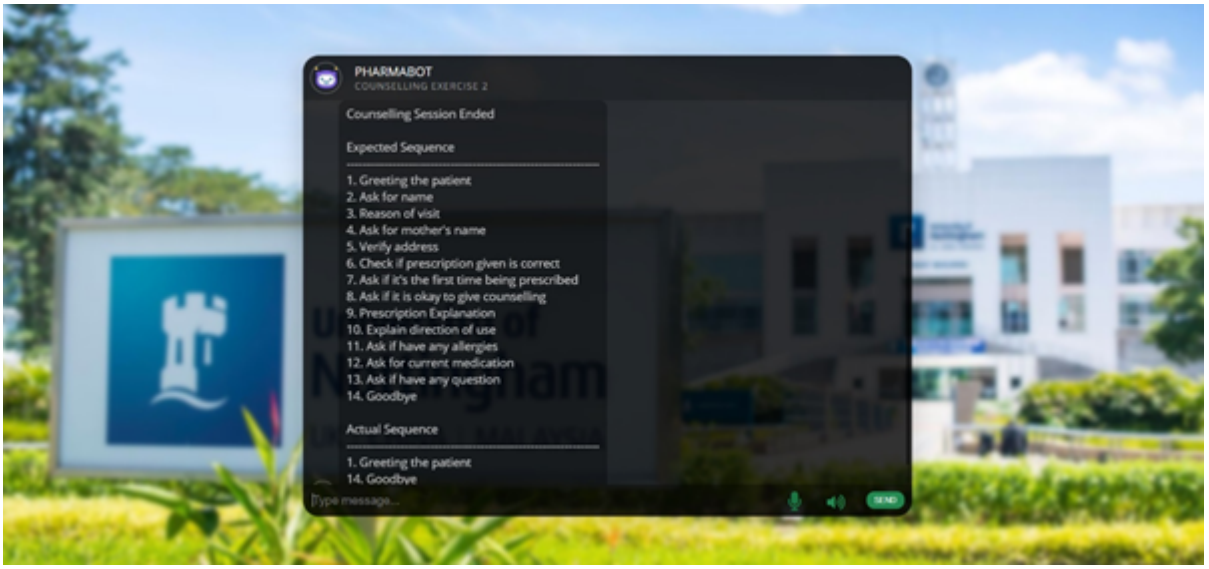


Step 6: User can start conversing with the chatbot by typing messages in the textbox and sending with the “send” button or pressing enter key.

- User can mute narration by clicking on the mute button
- User can use voice-to-text function by clicking on the microphone button. (microphone required)



Step 7: If the user wants to end the session, user can send “end” or “goodbye” to the chatbot and the chatbot will give a feedback for the counselling session.



Step 8.1: If user wishes to clear the chat, user can simply refresh the chatbot page

Step 8.2: If the user wishes to choose another exercise, the user can simply click on the back button on their browser and repeat Step 3 – Step 7.

Starting the server

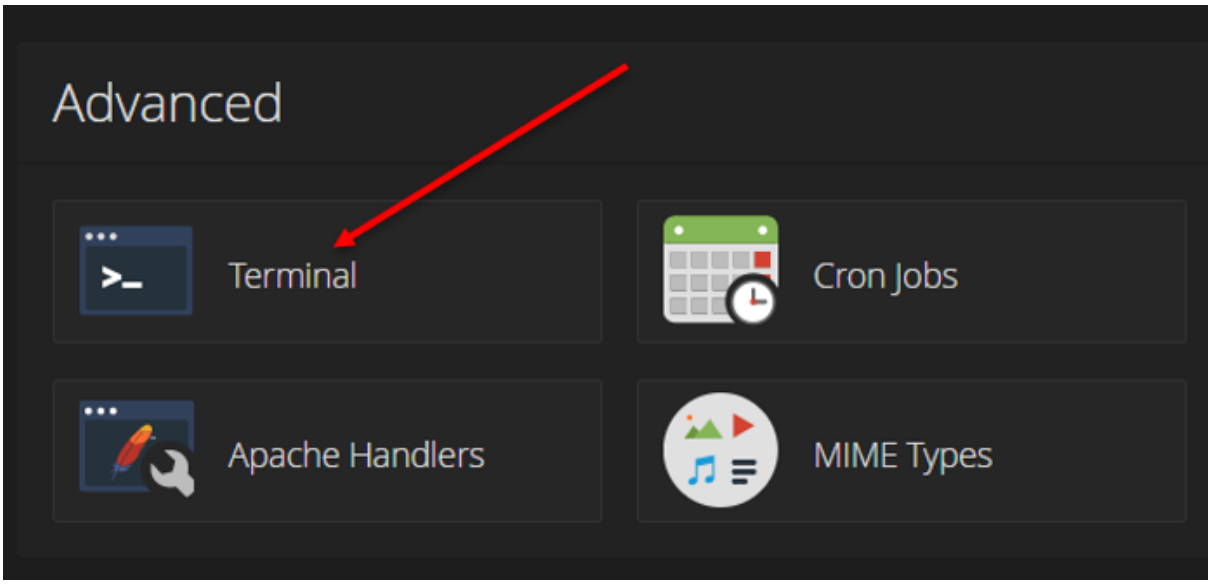
Step 1: go to https://jupiter.nottingham.edu.my:2083/

Step 2: log in to cPanel using these credentials

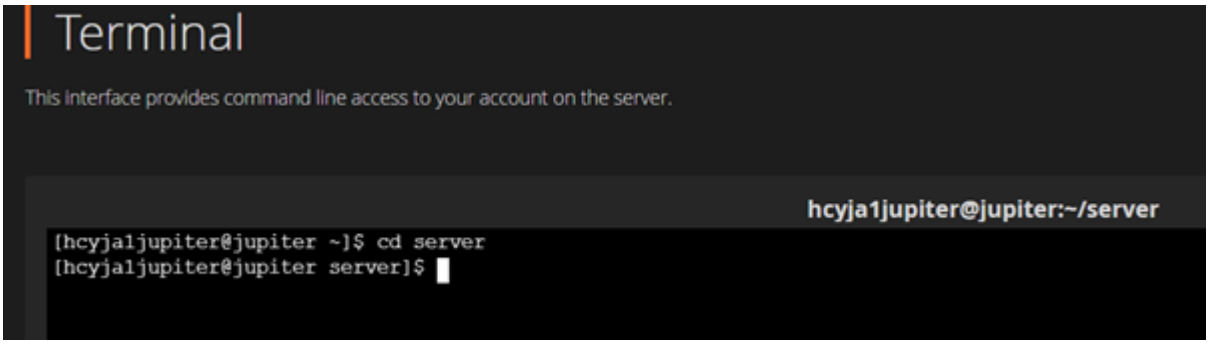
Username: hcyja1jupiter

Password: Ph@rm@botNottingham

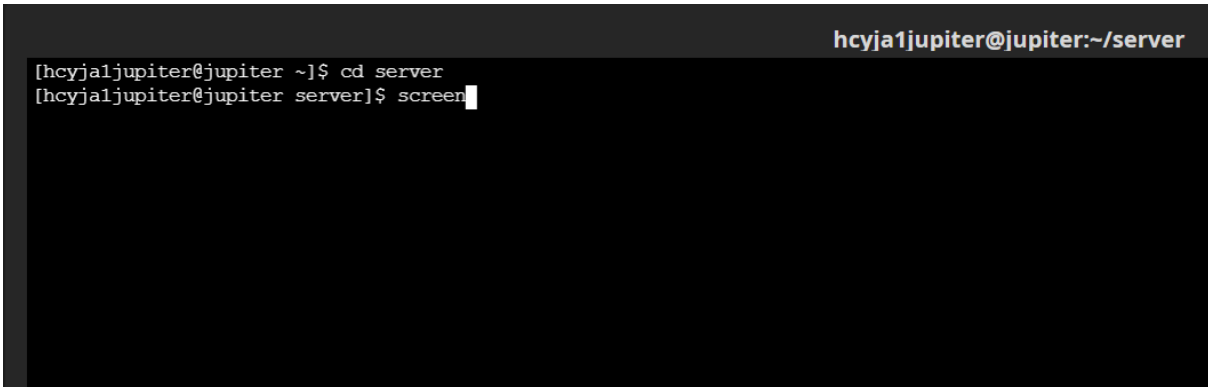
Step 3: Go to terminal



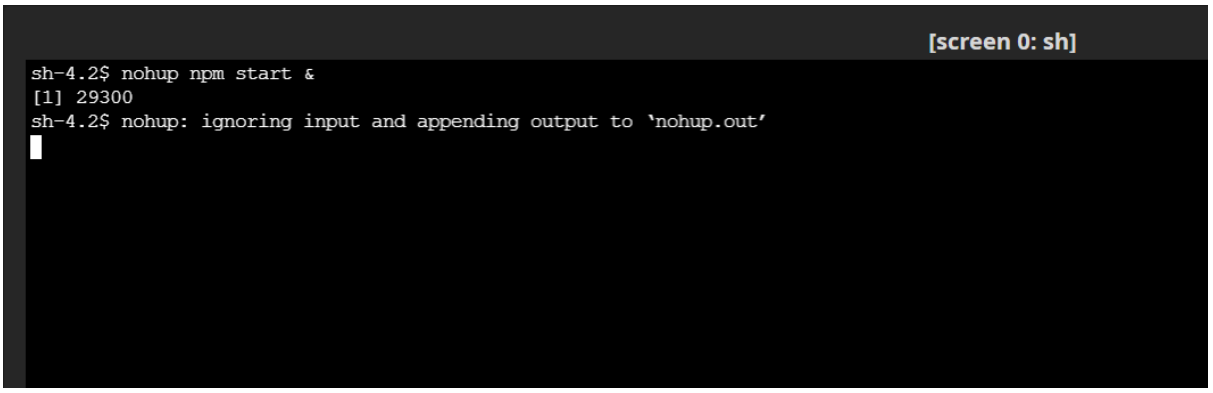
Step 4: type “cd server” and enter

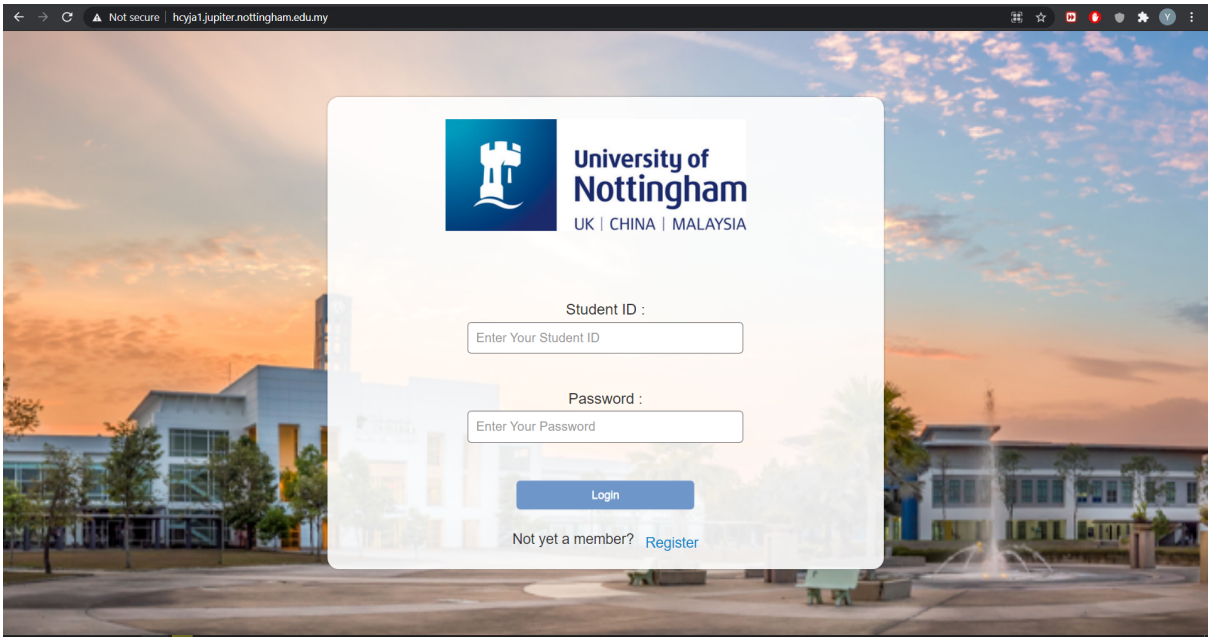


Step 5: type “screen” and enter



Step 6: type “nohup npm start &” and enter, and the server will start





Closing the Server

Step 1: Enter the commands “cd server” followed by “screen”

Step 2: Type “pkill node” to close the server

Login into DialogFlow (required before making any changes regarding chatbot)

Step 1: go to https://dialogflow.cloud.google.com/#/login

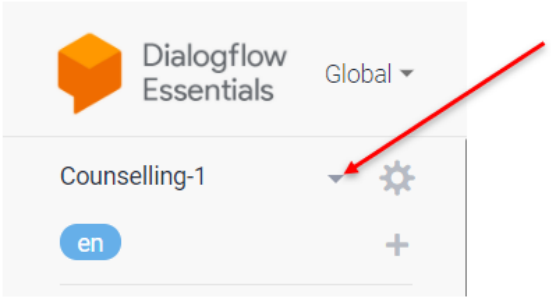
Step 2: log in using these credentials

Email: pharmabotnottingham@gmail.com

Password: Ph@rm@bot123

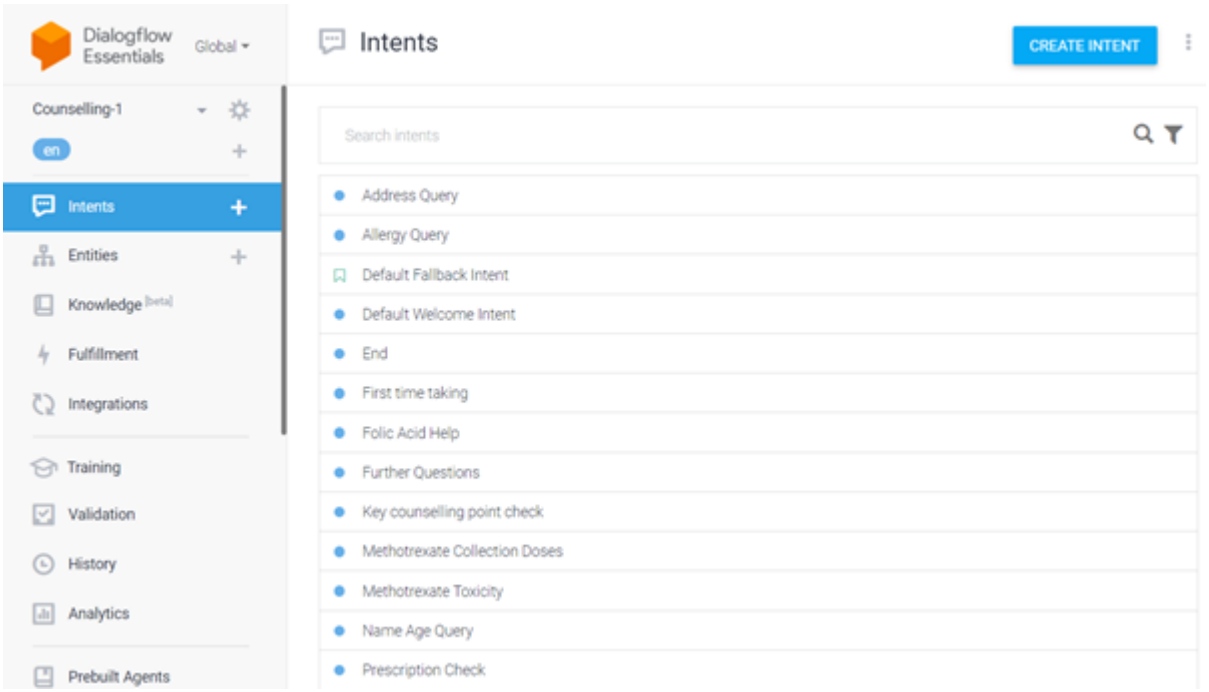
Modifying Existing Chatbot

Step 1: Click on the arrow button.

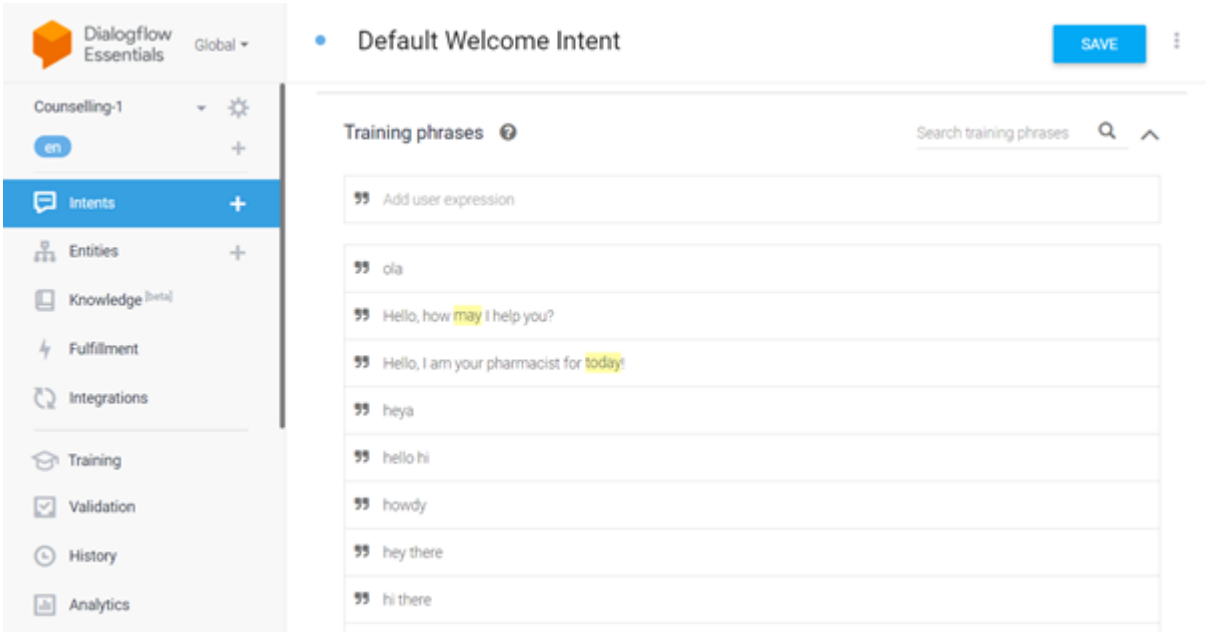


Step 2: Select which counselling exercise you wish to edit.

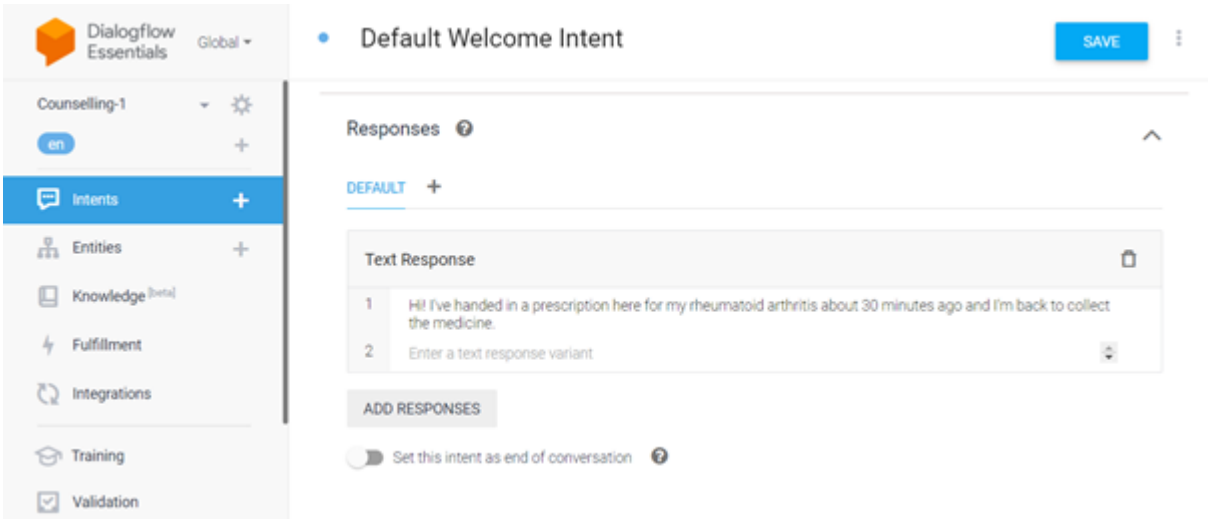
Step 3: After that go to the “Intents” section and choose which intent you would like to edit.



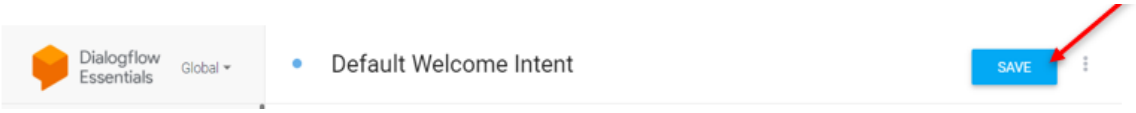
Step 4: Modify “Training phrases” for what pharmacists/students might say.



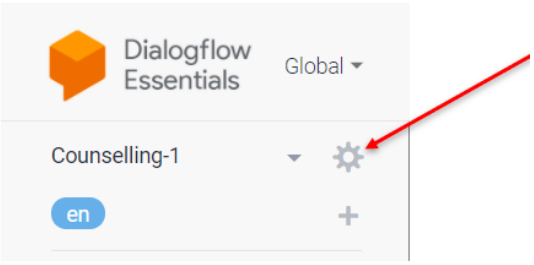
Step 5: Modify “Responses” for chatbot response when prompted by training phrase.



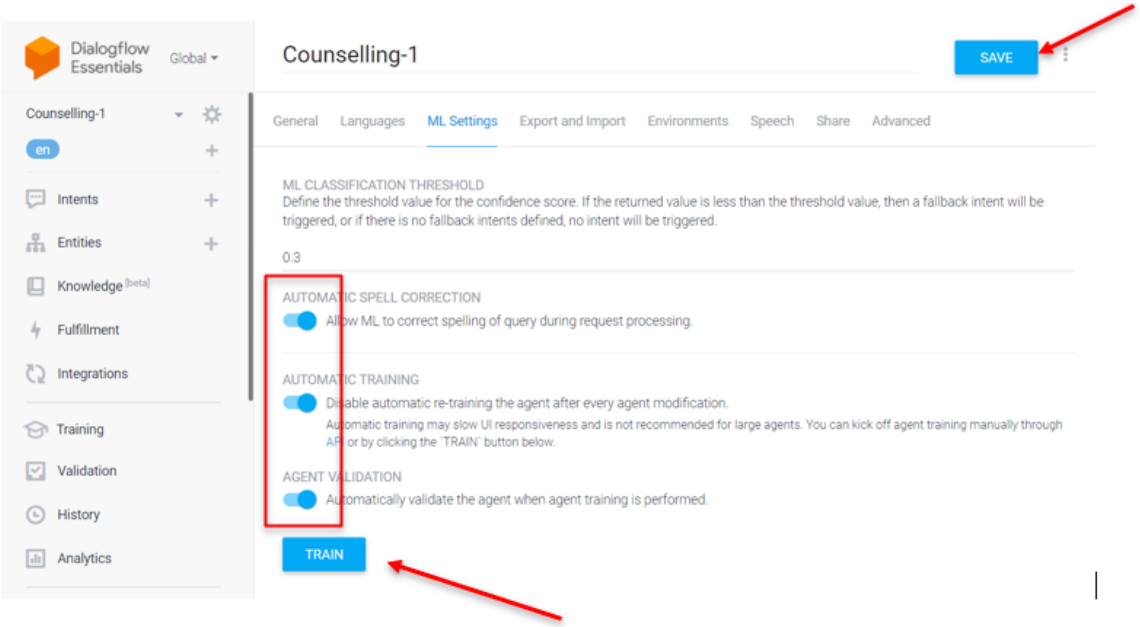
Step 6: Save changes by clicking the “SAVE” button.



Step 7: Go to the settings dashboard by clicking on the settings button.

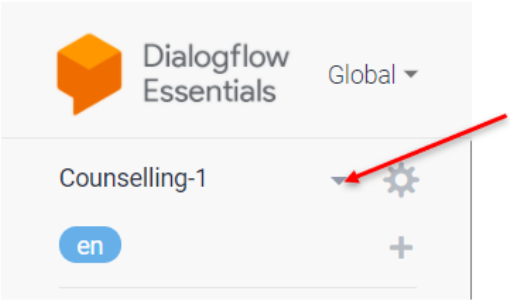


Step 8: Go to “ML Settings” tab, enable the following functions and click “TRAIN”, then click “SAVE”.

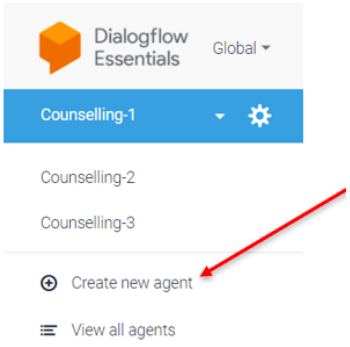


Creating new chatbot

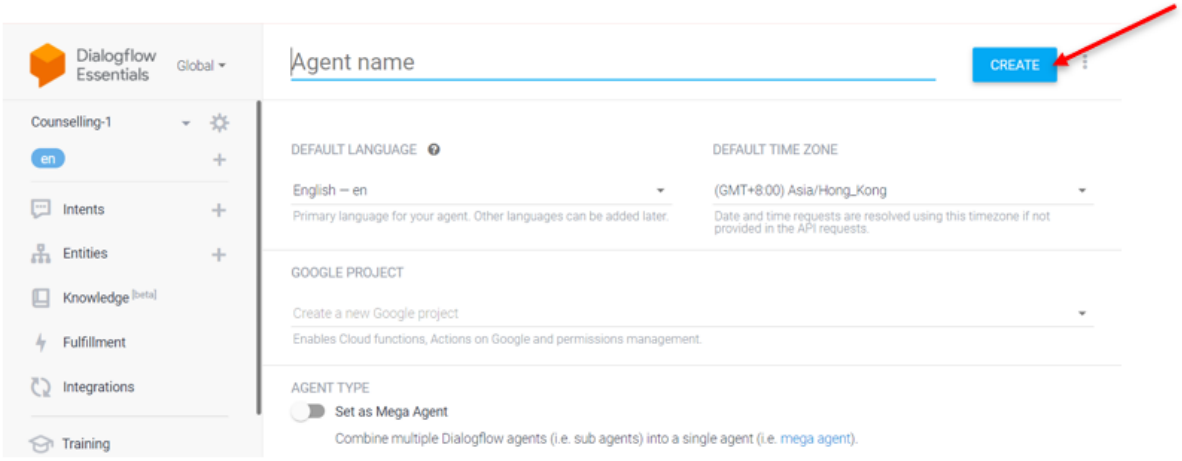
Step 1: Click on the arrow button.



Step 2: Click “Create new agent”.



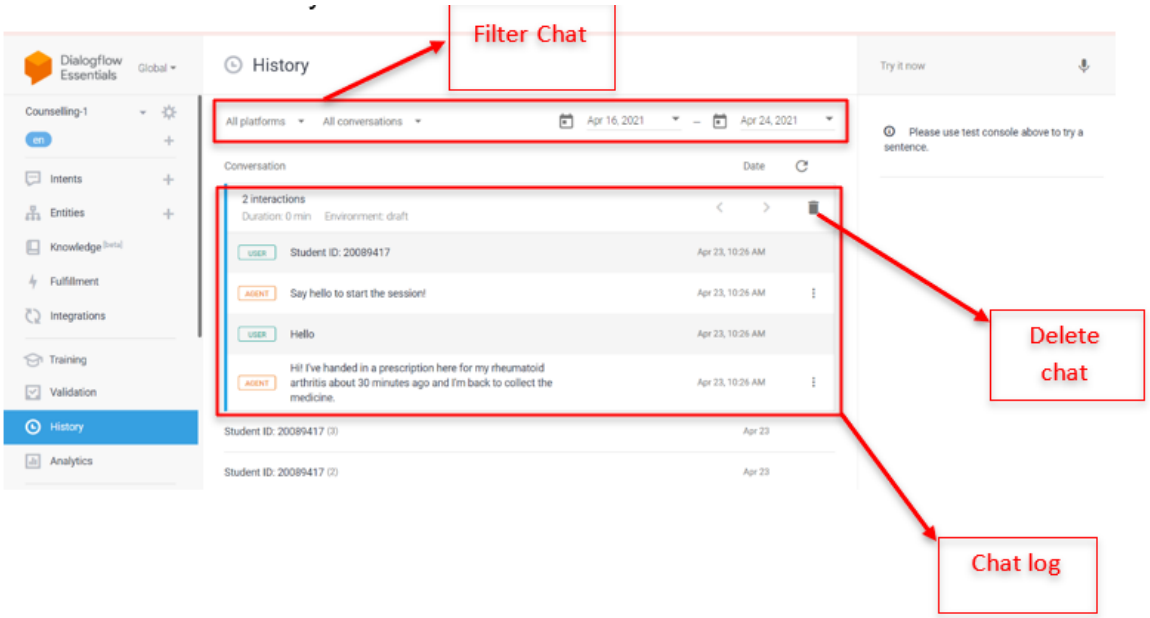
Step 3: Fill in agent name and click “CREATE”.



Step 4: Refer to “Modifying Existing Chatbot” manual to edit the chatbot.

Chat History

Admin can check and modify the chat log between user and chatbot based on their student id in the “History” section.



Running in localhost

Prerequisite

NodeJS : https://nodejs.org/en/ (get the LTS version)

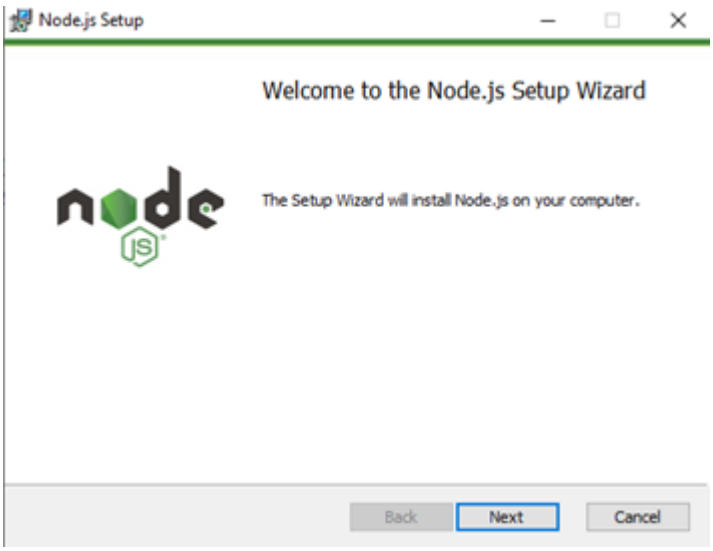
Xampp : https://www.apachefriends.org/index.html (choose respective to your Operating System)

Source Code: https://github.com/hcyja1/Pharmabot

Setup NodeJS

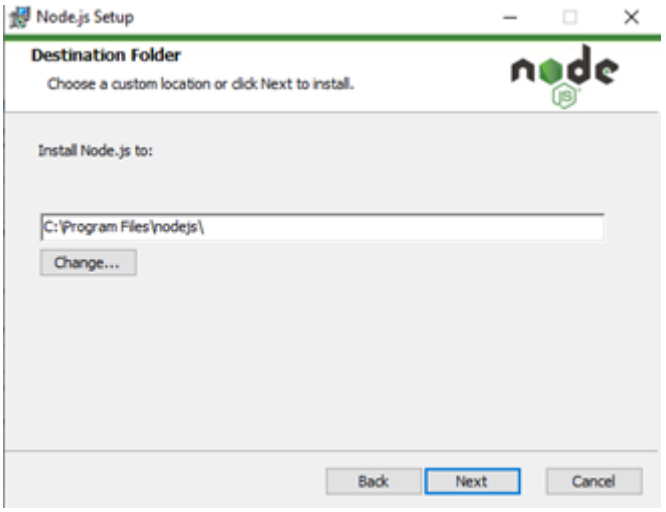
Step 1: run NodeJS installer.

Step 2: Click Next.

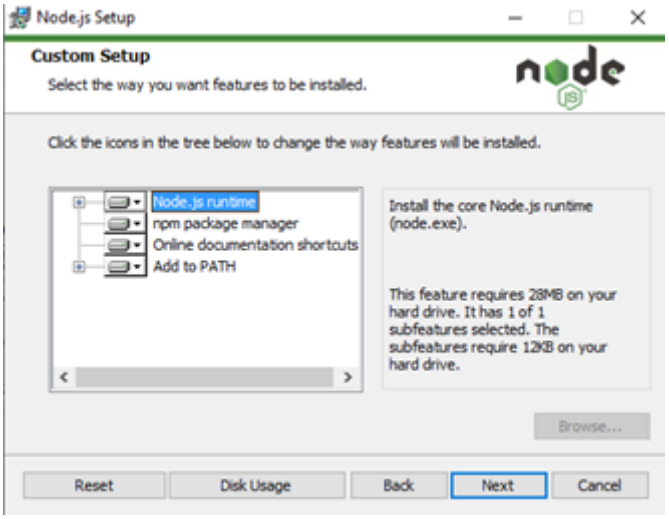


Step 3: Accept terms in License Agreement and click Next.

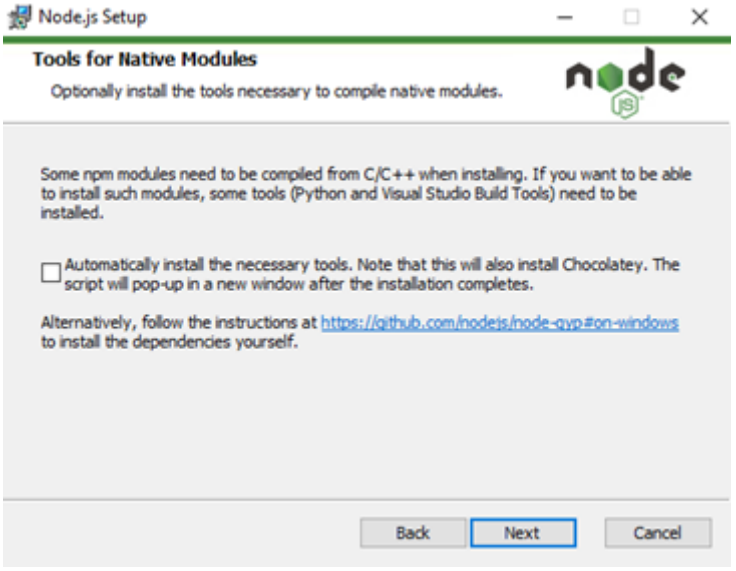
Step 4: Choose your installation path and click Next.



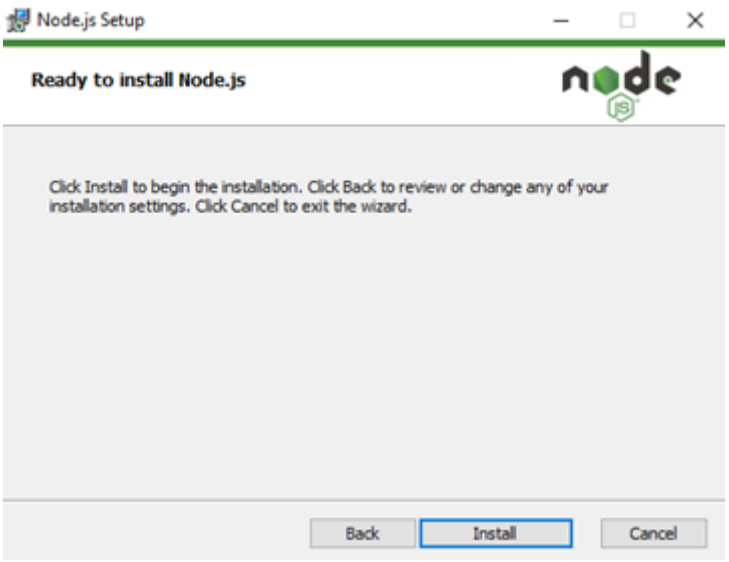
Step 5: Click Next



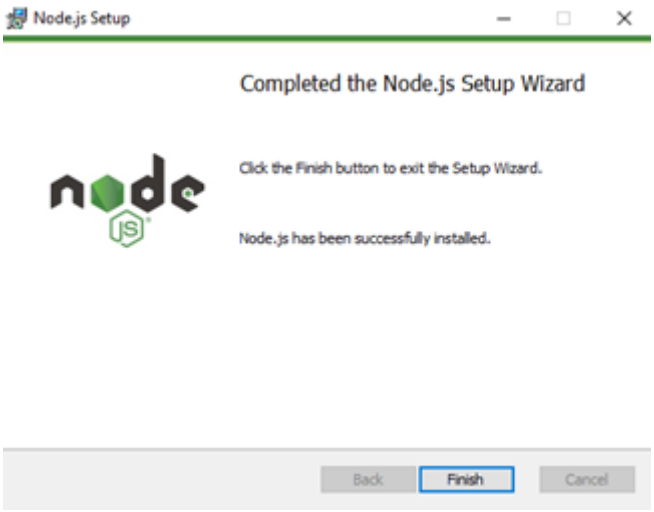
Step 6: Uncheck box and click Next.



Step 7: Click Install.

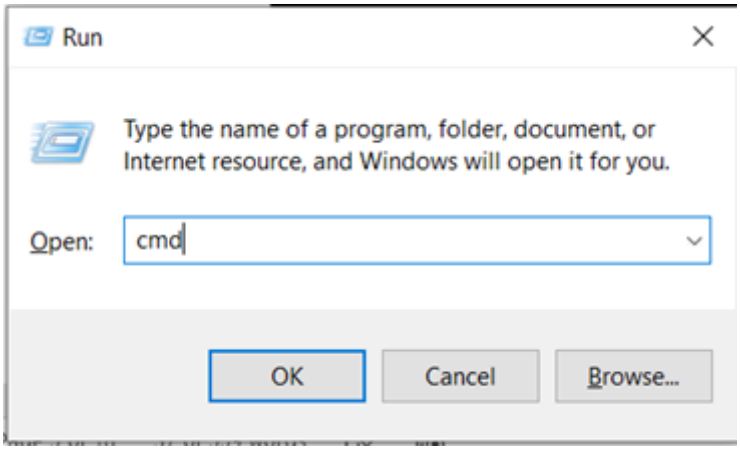


Step 8: Click Finish.



Step 9: Press window key + r

Step 10: type cmd and press OK



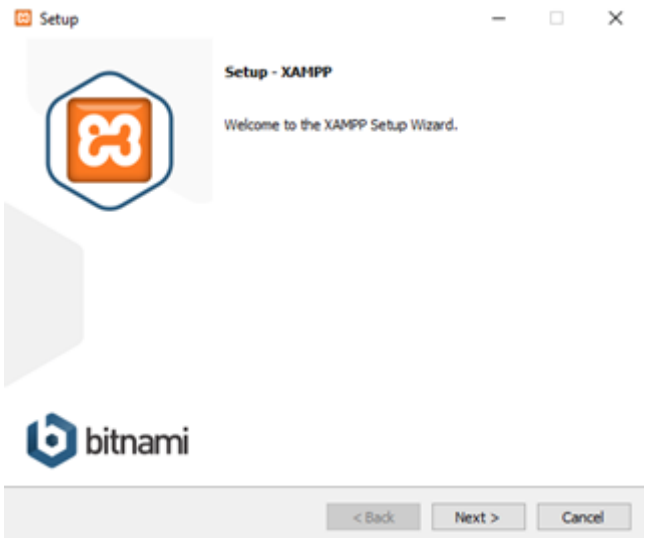
Step 11: In the cmd.exe, type “cd (folder extracted)”

Step 12: Enter command “npm install”

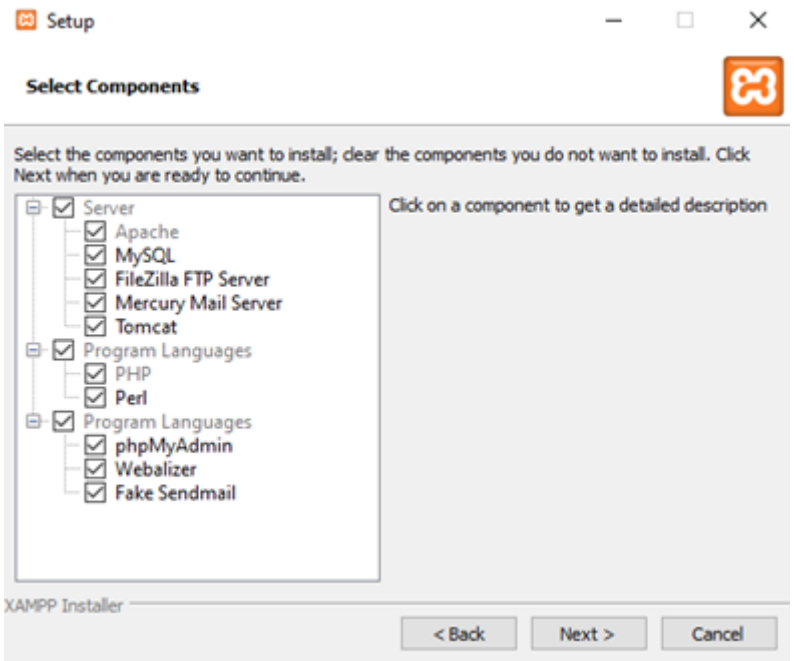
Setup XAMPP (database)

Step 1: Run XAMPP installer

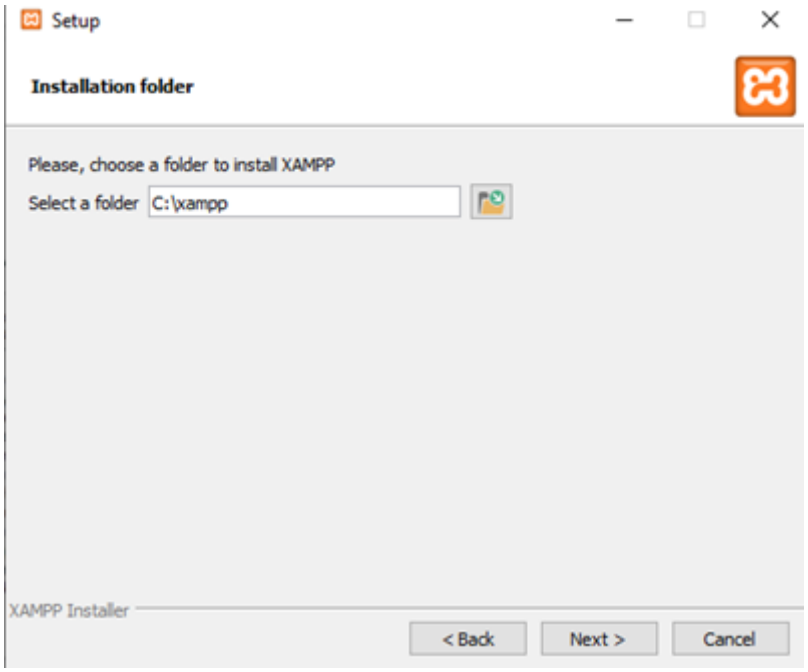
Step 2: Click Next



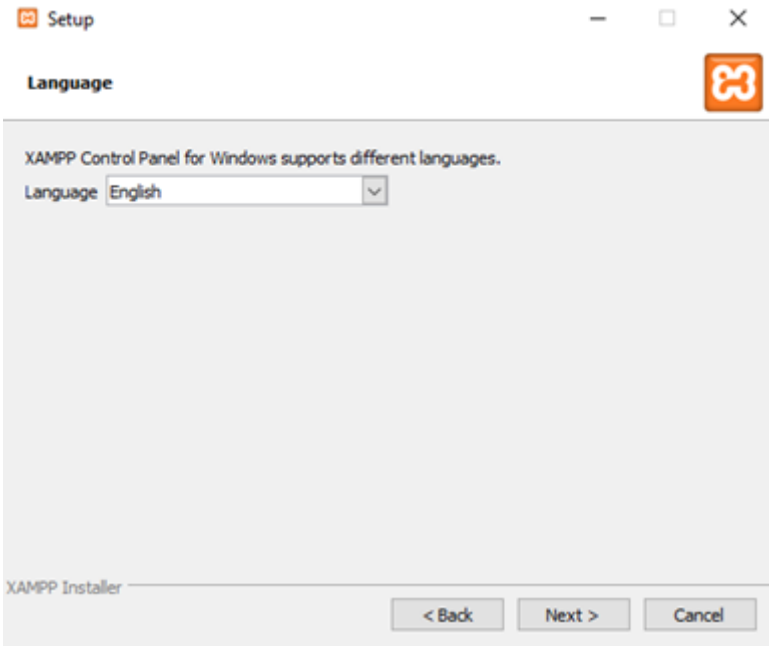
Step 3: Select all and click Next



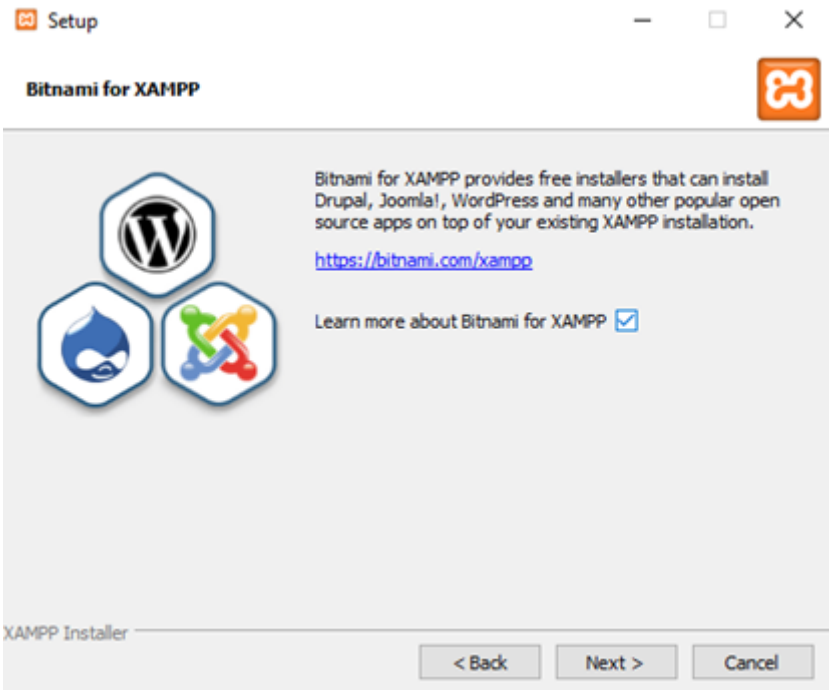
Step 4: Choose your installation path and click Next.



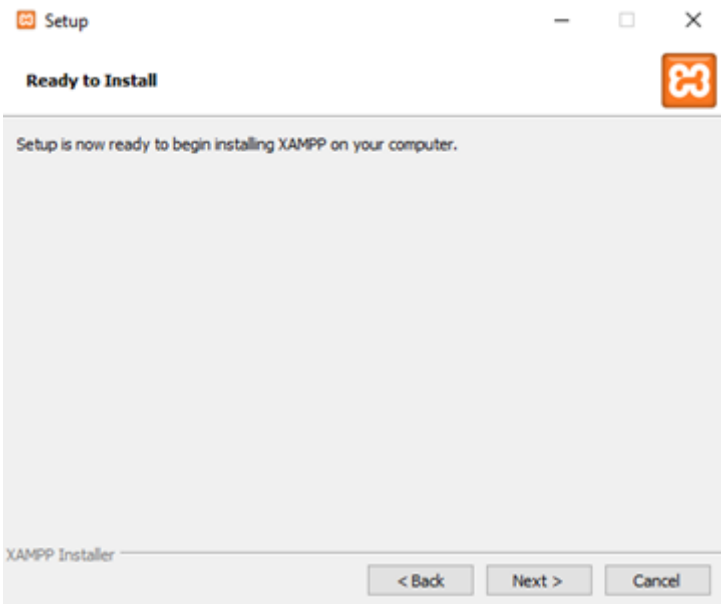
Step 5: Choose your preferred language and press Next.



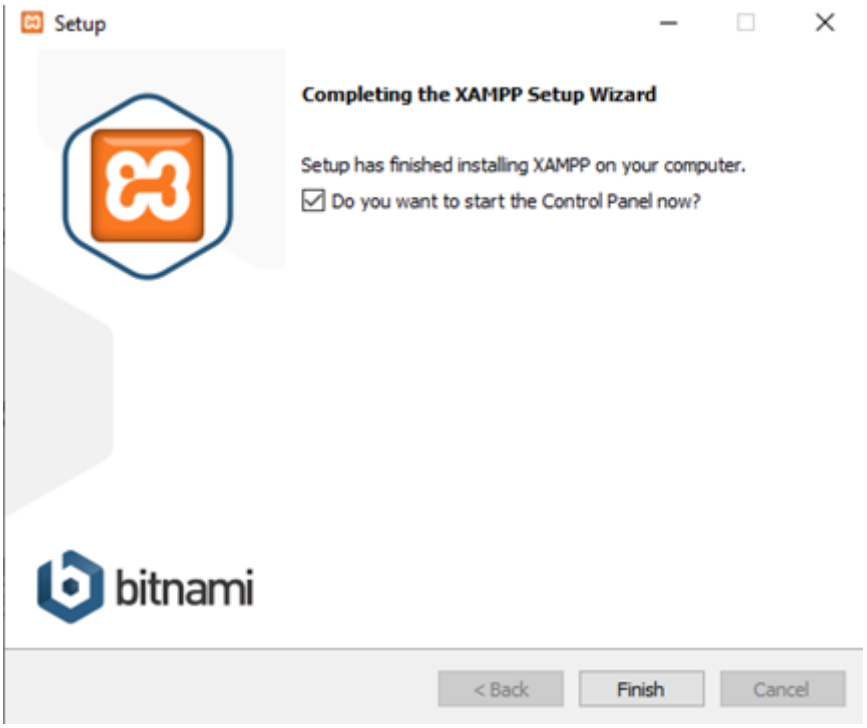
Step 6: Click Next.



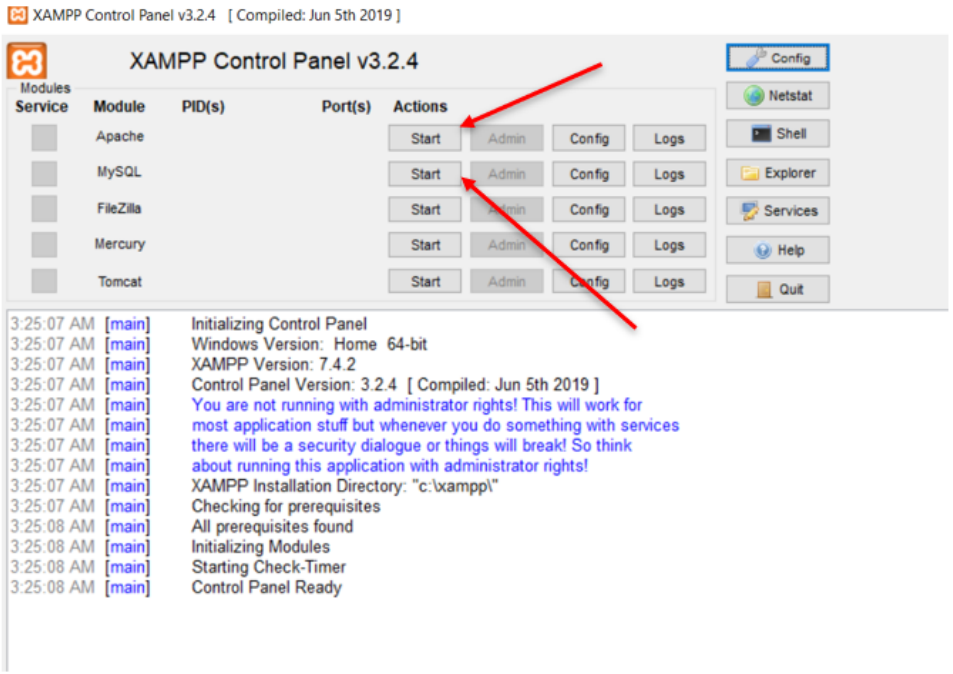
Step 7: Click Next.



Step 8: Check to start the Control Panel and click Finish.

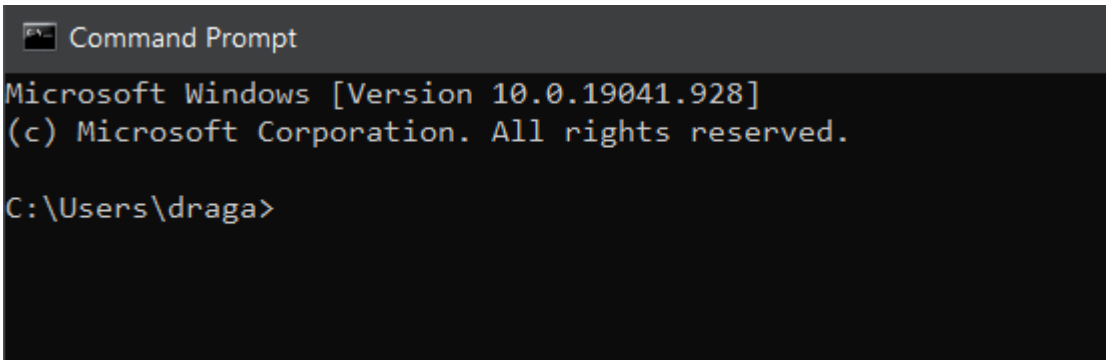


Step 9: Click Start for Apache and MySQL.

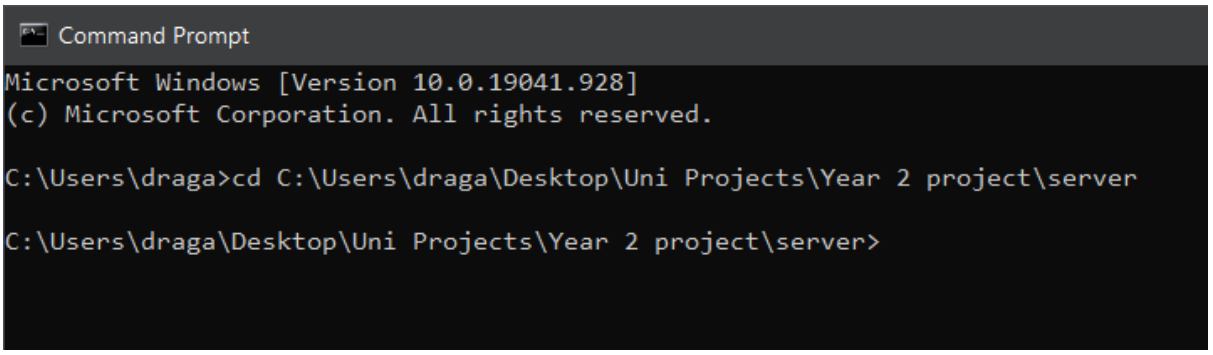


Starting the server

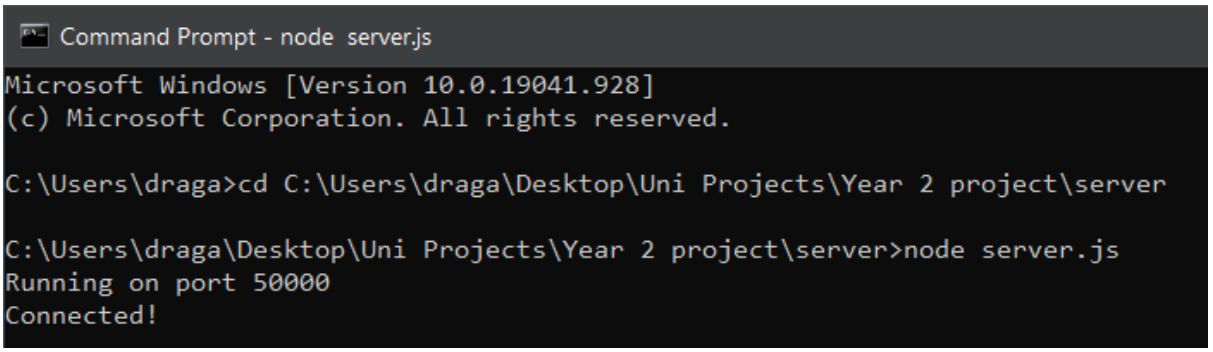
Step 1: Open command prompt



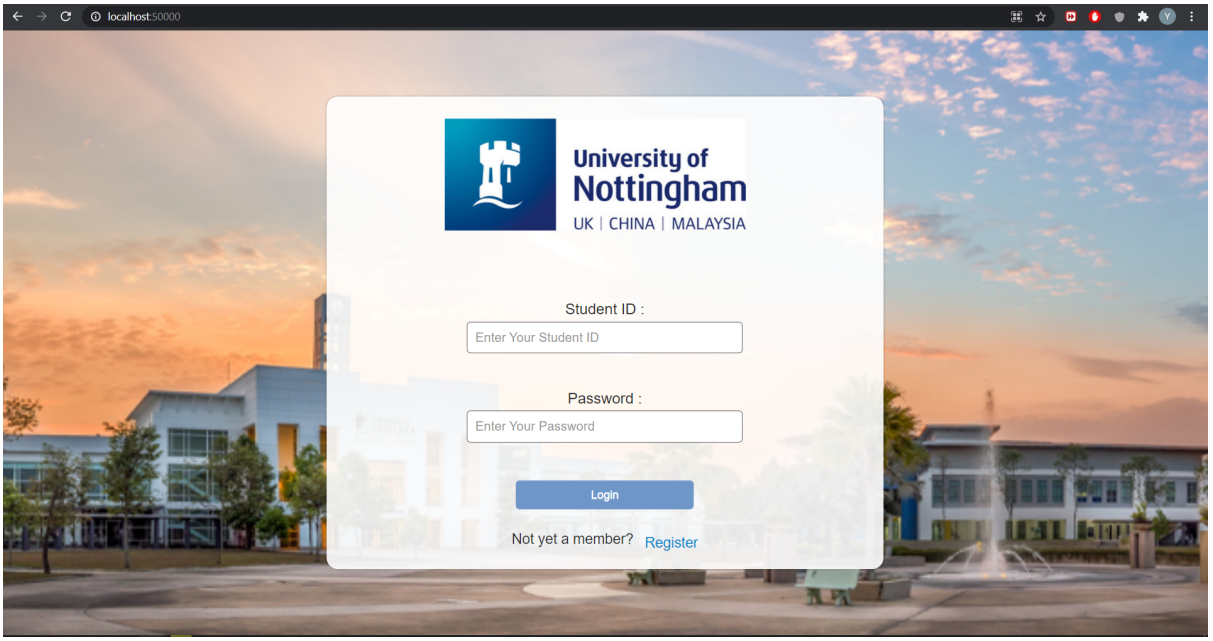
Step 2: Access directory with the server files using cd command followed by the directory



Step 3: Type “node server.js” and enter and the server will start

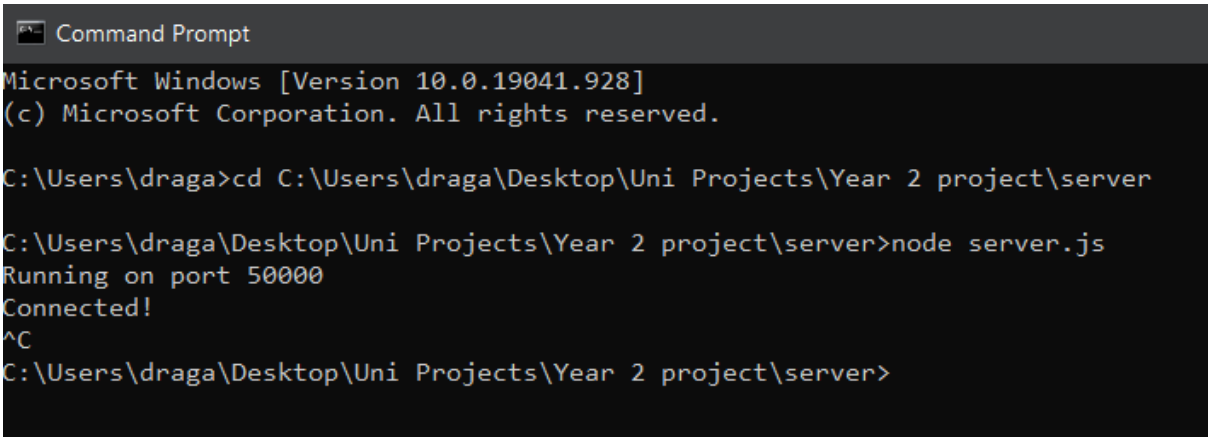


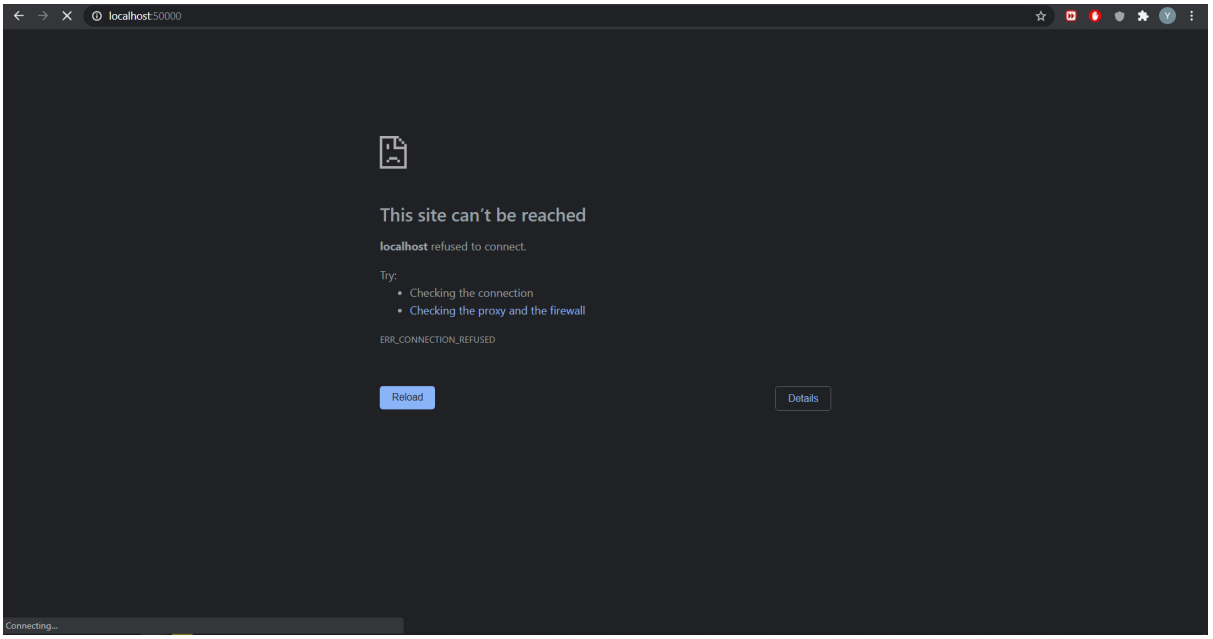
Step 4: Type in “http://localhost:50000/” in the url of google chrome and to access the login page



Closing the server

The server can be closed by either closing the command prompt or by pressing “ctrl + c” while in the window





15. Reference

[1] Shah, Huma; Warwick, Kevin; Vallverdú, Jordi; Wu, Defeng (2016). "Can machines talk? Comparison of Eliza with modern dialogue systems" (PDF). *Computers in Human Behavior*. **58**: 278–95. Available : https://curve.coventry.ac.uk/open/file/d4c5572d-3a8f-4ed1-b085-c88f8124fd74/1/Can%20Machines%20Talk_%20CHB_Shah-Warwick_2016%20%281%29.pdf

[2] Poulton, Terry. (2009). Repurposing existing virtual patients; an Anglo-German case study.

[3] Huang, Grace. (2007). Virtual Patient Simulation at U.S. and Canadian Medical Schools. Educational Strategies. 82 (5): 446–51. doi:10.1097/ACM.0b013e31803e8a0a.

[4] Kononowicz, Andrzej A., Zary, Nabil, Edelbring, Samuel, Corral, Janet, Hege, Inga (2015). Virtual patients - what are we talking about? A framework to classify the meanings of the term in healthcare education. BMC Medical Education. 15: 11. doi:10.1186/s12909-015-0296-3.

[5] Maicher K, Danforth D, Price A, Zimmerman L, Wilcox B, Liston B, Cronau H, Belknap L, Ledford C, Way D, Post D, Macerollo A, Rizer M (2017). Developing a Conversational Virtual Standardized Patient to Enable Students to Practice History-Taking Skills. Simul Healthc. 12(2):124-131. doi: 10.1097/SIH.0000000000000195.

[6] Bindoff, Ivan & Ling, Tristan & Bereznicki, Luke & Chalmers (nee Stafford), Leanne & Breen, Juanita & Gee, Peter & Garland, Tristan. (2016). Pharmacy Simulator: A 3D computer-based virtual patient simulator for training community pharmacists. Research in Social and Administrative Pharmacy. 12. e47. 10.1016/j.sapharm.2016.05.112.

[7] Basics | Dialog Flow. (n.d.). Available : https://cloud.google.com/dialogflow/docs/

[8] Matthew Finnegan(2019). New Cortana and Google Assistant updates highlight workplace potential: Microsoft and Google this week both showcased changes to their digital voice assistants designed to make them more useful in the office. computer world from idg.

[9] Manisha Salecha (2016). Story of ELIZA, the first chatbot developed in 1966: ELIZA operates by recognizing keywords or phrases using those keywords from pre-programmed responses. analytics india magazine pvt.

[10] Manuel Bronstein (2019). Bringing you the next-generation Google Assistant: To power the Google Assistant, we rely on the full computing power of our data centers

to support speech transcription and language understanding models. Google Assistant at I/O.

[11]Mitul Makadia (2019). 5 Reasons Why Your Chatbot Needs Natural Language Processing: NLP can pave the way for an easier-to-use interface to features and services. towards data science.

[12] John Ball (2016). Why Chatbots Won't Survive Without NLU: NLU has been considered an AI-hard problem that is fundamentally unsolved. Chatbot Magazine by Octane AI.

[13] Pedro Mendonca (2020). Human-Computer Interaction: What Is It, Why Does It Matter & Best Practices: Learn why this field of science can perfectly combine design, computer technology and human interaction providing a great experience to every user when well executed. Exaud Insights.

[14] Yan Shan, Christian Hardmeier, Joakim Nivre (2018). Universal Word Segmentation; Implementation and Interpretation" (PDF) page 422-423 Available: https://www.mitpressjournals.org/doi/pdf/10.1162/tacl_a_00033#:~:text=Word%20segmentation%20is%20a%20low,a%20considerable%20number%20of%20languages.&text=The%20experimental%20results%20indicate%20that,of%20unique%20non%2Dsegmental%20terms.

[15] Nivedita bhirud, Subhash Tatale, Sayali Randive, Shubham Nahar (2019) "A Literature Review On Chatbots In Healthcare Domain" (PDF) page 225-231 Available: https://www.researchgate.net/publication/334836867_A_Literature_Review_On_Chatbots_In_Healthcare_Domain

[16] Tiago Duque (2020). Building a Tokenizer and a Sentencizer: Understanding the underlying bones that give NLP its structure. Analytics Vidhya.

[17] Usman Shahid,(2019,September 15), Building a Chatbot with Google DialogFlow[Online]. Available : https://blog.datasciencedojo.com/building-a-chatbot-with-google-dialogflow/#:~:text=A%20DialogFlow%20Agent%20is%20a,apps%20and%20services%20can%20understand.