VIETNAM NATIONAL UNIVERSITY OF HOCHIMINH CITY

THE INTERNATIONAL UNIVERSITY

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

A picture containing text, room, gambling house

Description automatically generated

# Building A Microsoft Teams Application For Online In-Class Activities Management

By

Phạm Nguyễn Trường Thịnh – ITITIU15068

A thesis submitted to the School of Computer Science and Engineering

In partial fulfillment of the requirements for the degree of

Bachelor of Computer Science

Ho Chi Minh City, Viet Nam

2021-2022

# Building A Microsoft Teams Application For Online In-Class Activities Management

APPROVED BY:

Tran Thanh Tung, Ph.D.

THESIS COMMITTEE

**ACKNOWLEDGMENTS**

I sincerely want to give the most respect to my supervisor – Dr. Trần Thanh Tùng, who has guided me through this semester and also spent his time to let me receive what I wish for the Thesis. His deep knowledge is the thing which helped me a lots during the process and I want to thank you about that.

I regard my friend – Luong Huynh Huy Hoang, who I want to give my gratitude and appreciation to. His technical help and contribution helped me to understand deeply what I need to improve the understanding of workflow of the BOT technology and how to handle all the cases of the operation. This is an unforgettable experience. My honor goes to the faculty of the School of Computer Science of the International University, especially Dr. Trần Thanh Tùng for his support since the beginning of my studies. I also want to give my respect to the members of the examination committee.

**TABLE OF CONTENTS**

Contents

[Building A Microsoft Teams Application For Online In-Class Activities Management 1](#_Toc93907912)

[Building A Microsoft Teams Application For Online In-Class Activities Management 2](#_Toc93907913)

[**ACKNOWLEDGMENTS** 3](#_Toc93907914)

[**TABLE OF CONTENTS** 4](#_Toc93907915)

[**LIST OF FIGURES** 6](#_Toc93907916)

[**LIST OF TABLES** 8](#_Toc93907917)

[**ABBREVIATIONS** 9](#_Toc93907918)

[**ABSTRACT** 10](#_Toc93907919)

[CHAPTER 1. INTRODUCTION 11](#_Toc93907920)

[**1.1.** **Problem Statement** 11](#_Toc93907921)

[**1.2.** **Motivation** 11](#_Toc93907922)

[**1.3.** **Objective** 12](#_Toc93907923)

[**1.4.** **Assumption** 12](#_Toc93907924)

[CHAPTER 2. BACKGROUND / APPLIED TECHNOLOGIES 13](#_Toc93907925)

[**2.1.** **Microsoft Teams (MS Teams)** 13](#_Toc93907926)

[**2.2.** **Node.JS** 14](#_Toc93907927)

[**2.3.** **NPM and Nodemon** 15](#_Toc93907928)

[**2.4.** **Ngrok** 16](#_Toc93907929)

[**2.5.** **Javascript (JS)** 18](#_Toc93907930)

[**2.6.** **Teams Toolkit** 19](#_Toc93907931)

[**2.7.** **Bot Framework SDK** 20](#_Toc93907932)

[**2.8.** **Azure Active Directory (AAD)** 22](#_Toc93907933)

[**2.9.** **Microsoft Graph API** 24](#_Toc93907934)

[**2.10.** **Overview of Bot apps for education** 25](#_Toc93907935)

[CHAPTER 3. SYSTEM REQUIREMENT 27](#_Toc93907936)

[**3.1.** **System Description** 27](#_Toc93907937)

[**3.2.** **System use cases** 28](#_Toc93907938)

[**3.3.** **System Functionality** 29](#_Toc93907939)

[**3.4.** **System Workflow** 30](#_Toc93907940)

[CHAPTER 4. IMPLEMENTATION 38](#_Toc93907941)

[**4.1.** **Setup environment** 38](#_Toc93907942)

[**4.2.** **Required tools** 42](#_Toc93907943)

[**4.3.** **Implementation and Result** 44](#_Toc93907944)

[CHAPTER 5. DISCUSSION 59](#_Toc93907945)

[CHAPTER 6. CONCLUSION AND FUTURE WORK 60](#_Toc93907946)

[**5.1.** **Summary** 60](#_Toc93907947)

[**5.2.** **Future work** 60](#_Toc93907948)

[**LIST OF REFERENCES** 61](#_Toc93907949)

[**Bibliography** 61](#_Toc93907950)

**LIST OF FIGURES**

[Figure 1: MS Teams architecture belong to Microsoft Document 13](#_Toc101263762)

[Figure 2: Node.js in details 14](#_Toc101263763)

[Figure 3: Node.js system 15](#_Toc101263764)

[Figure 4: Local Dependencies managed by NPM 16](#_Toc101263765)

[Figure 5: Example of ngrok on terminal 17](#_Toc101263766)

[Figure 6: Example of ngrok – URL, output 17](#_Toc101263767)

[Figure 7: How ngrok works 17](#_Toc101263768)

[Figure 8: JS 18](#_Toc101263769)

[Figure 9: Example of JS code for comparing answer in the question 19](#_Toc101263770)

[Figure 10: Teams toolkit 19](#_Toc101263771)

[Figure 11: Example of Bot belong to Microsoft Document 21](#_Toc101263772)

[Figure 12: Example of building bot belong to Microsoft Document 21](#_Toc101263773)

[Figure 13: Azure AD architecture belong to Azure 22](#_Toc101263774)

[Figure 14: Azure AD resources 23](#_Toc101263775)

[Figure 15: Microsoft Graph belong to Microsoft Document 24](#_Toc101263776)

[Figure 16: Relationship between Microsoft Graph and others belong to Microsoft Document 25](#_Toc101263777)

[Figure 17: System use cases 28](#_Toc101263778)

[Figure 18: BOT Entity Relationship 29](#_Toc101263779)

[Figure 19: Bot's workflow belong to Microsoft Document 31](#_Toc101263780)

[Figure 20: Create Question 32](#_Toc101263781)

[Figure 21: Create Attempt for Answer base on Question ID 33](#_Toc101263782)

[Figure 22: Choose Question 33](#_Toc101263783)

[Figure 23: Send Answer message 34](#_Toc101263784)

[Figure 24: Check Answer Attempt 34](#_Toc101263785)

[Figure 25: Close Question section 35](#_Toc101263786)

[Figure 26: Create Result 35](#_Toc101263787)

[Figure 27: Compare Answer and Result 36](#_Toc101263788)

[Figure 28: Show Value of Question and Answer 36](#_Toc101263789)

[Figure 29: Record into file 37](#_Toc101263790)

[Figure 30: Open Teams Toolkit 38](#_Toc101263791)

[Figure 31: Create section 39](#_Toc101263792)

[Figure 32: Create new app 39](#_Toc101263793)

[Figure 33: Bot section 40](#_Toc101263794)

[Figure 34: Programming language 40](#_Toc101263795)

[Figure 35: App name 40](#_Toc101263796)

[Figure 36: App files 41](#_Toc101263797)

[Figure 37: Teams Toolkit features 43](#_Toc101263798)

[Figure 38: teamsBot’s file running 44](#_Toc101263799)

[Figure 39: Add Bot local 45](#_Toc101263800)

[Figure 40: After added 46](#_Toc101263801)

[Figure 41: Bot added into a channel 46](#_Toc101263802)

[Figure 42: Bot announcement 47](#_Toc101263803)

[Figure 43: Fetch user info 47](#_Toc101263804)

[Figure 44: User's token 48](#_Toc101263805)

[Figure 45: Mention Bot 48](#_Toc101263806)

[Figure 46: Running message in terminal 48](#_Toc101263807)

[Figure 47: Question 49](#_Toc101263808)

[Figure 48: Question message 50](#_Toc101263809)

[Figure 49: Answer code 51](#_Toc101263810)

[Figure 50: Question ID 52](#_Toc101263811)

[Figure 51: Choose question ID 52](#_Toc101263812)

[Figure 52: Choose and Answer command 52](#_Toc101263813)

[Figure 53: Result for Question ID 53](#_Toc101263814)

[Figure 54: End Question ID 53](#_Toc101263815)

[Figure 55: End all Questions 54](#_Toc101263816)

[Figure 56: Question after closed 54](#_Toc101263817)

[Figure 57: Compare Question ID 55](#_Toc101263818)

[Figure 58: Compare Message 55](#_Toc101263819)

[Figure 59: Show Answer 56](#_Toc101263820)

[Figure 60: Print out message 57](#_Toc101263821)

[Figure 61: Record into file 57](#_Toc101263822)

[Figure 62: Answer file 58](#_Toc101263823)

**LIST OF TABLES**

[Table 1: Characteristics of Bot 20](#_Toc93480261)

[Table 2: Azure AD characteristics 23](#_Toc93480262)

[Table 3: Functionality of Bot 30](#_Toc93480263)

**ABBREVIATIONS**

MS Teams Microsoft Teams

APP Application

API Application Programming Interface

AAD Azure Active Directory

NPM Node Package Management

JS Javascript

VS Code Visual Studio Code

**ABSTRACT**

Since the first computer has been created, the development of the technology is known to be the greatest breakthrough of humanity. Due to the circumstance of technology development, many aspects have been developed such as media, Internet, etc. According to the Internet’s development, teaching and studying have become easier than the old days. Because everyone can approach with a computer or laptop. They just need to open an MS Team application to learn online class or teaching through this app. Moreover, this app keeps the interaction between people more effective without meeting directly outside. But the disadvantage of this connection is that it is difficult for the lecturer to keep an eye on students all the time. Therefore, Bot has been created, which base on this kind of aspect. Bot will allow the lecturer to manage activities for students in Online-class.

This thesis will express the communication between lecturer and bot or student and bot. All the features and functions of the Bot will be represented in the Demo, which will let the engineers know how to interact with and understand the workflow of the bot.

# CHAPTER 1. INTRODUCTION

* 1. **Problem Statement**

Due to the circumstances of the COVID-19, teaching and studying at school have been canceled. This is a significant problem in education which can lead to the collapse of the country. If people don’t have knowledge, the problem won’t be solved correctly. To solve this problem, the Ministry of Education has come to a solution which is Online class.

To keep the interaction between lecturer and student smoothly, we need to use MS Teams app, which allows lecturer to teach students online. However, it is impossible for lecturer to always keep an eye on students all the time. Lecturers can make many mistakes in collecting study data. Whenever a question has been created, lecturer has problems in collecting answers from student base on the time. Also the lecturer has to scroll down for each answer and compare between the result and answer, which is very time-consuming. Besides, technical difficulties with online teaching tools are the things need to be considered as problems too.Because some lecturers are not used to use the modern technology depend on their ages.

* 1. **Motivation**

Therefore, to deal with these kinds of problems, Bot app has been invented which can execute the commands of the lecturer and students correctly and smoothly without any problems.

Because of the correct command’s execution , Bot can eliminate all the human mistakes as many as possible for user. The result will be demonstrated after compiling by Bot which represent as the activities of the lecturer’s management for students.

* 1. **Objective**

The purpose of this thesis is to let users learn about fundamental workflow by using Bot app, and interact with Bot smoothly by sending commands such as creating Question, Result and Comparing answers, etc. The students can also send commands to Bot to answer the Question which will be stored by Bot correctly.

This thesis will show what system requirements that user needs to implement and understand the Bot’s workflow after constructed.

* 1. **Assumption**

However, Bot app technology is not yet widely popular in education nowadays. Ministry of Education still does not apply this app frequently in education depending on the efficiency of this app.

In summary, this thesis assumes the effective of Bot app for educating. This paper also demonstrates the convenience and usefulness of it.

# CHAPTER 2. BACKGROUND / APPLIED TECHNOLOGIES

**2.1. Microsoft Teams (MS Teams)**

Microsoft Teams is an application built for hybrid work, which its platform supports communication, meeting, app sharing, etc. Ms teams is a very convenient and useful application. Because of this, people from anywhere in the world can meet each other without going outside. It is widely used for teaching, studying, working, talking, etc.



Figure 1: MS Teams architecture belong to Microsoft Document

**2.2. Node.JS**

NodeJs is defined as an open-source and cross-platform which operates in a single thread, rather than establishing a new one for each request. Node.js standard library includes a set of asynchronous I/O primitives that prevent JavaScript code from blocking, and libraries in Node.js are often created following non-blocking paradigms, thus blocking behavior is the exception rather than the rule.

Node.js is a scalable network application framework that uses an asynchronous event-driven JavaScript engine which is the main structure that constructs Node.js. Node.js is designed for developing server-side and networking applications.



Figure 2: Node.js in details

Below is the simple description of Node.js system.

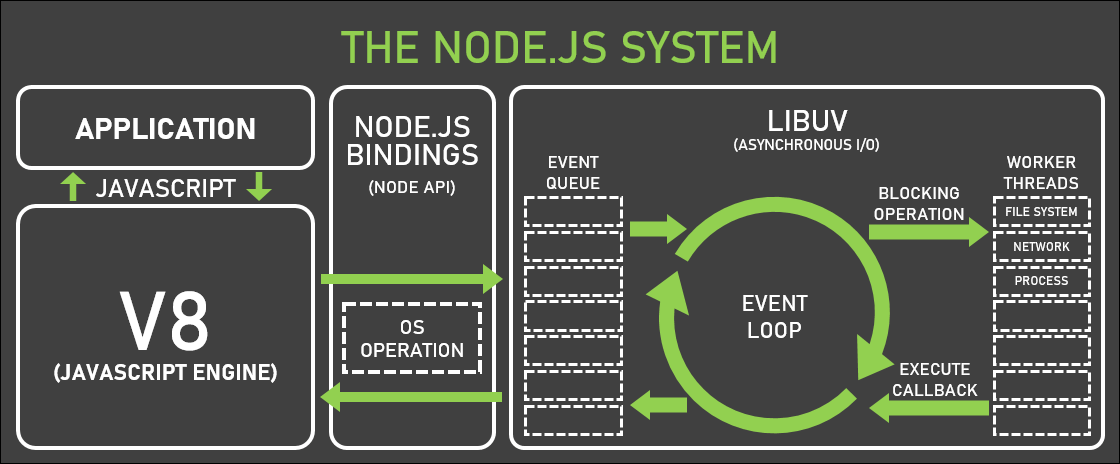


Figure 3: Node.js system

**2.3. NPM and Nodemon**

NPM stands for Node Package Manager. When you install Node.js application, NPM is also included in the installation. NPM becomes an online registry that hosts packages, as well as a manager of those packages and their dependencies on your workstation, so you may utilize them anytime you want on your project.

Nodemon is a tool or a package which helps user to save a lots of time when programming in Node.js. Whenever a single code has been modified, user have to restart the server by command “node app.js” to update new changes. With nodemon, the server will automatically restart the application for the user whenever a change happens in the file. In summary, any additional changes to your code or method of development, nodemon does not require.

Text

Description automatically generated

Figure 4: Local Dependencies managed by NPM

**2.4. Ngrok**

Ngrok is a tool which can create tunnels between your localhost and Internet. This tool allows other users to access your localhost through the custom domain of ngrok.

The benefit of ngrok is to allow you to test responsive on mobile easily through URL provided by ngrok. Not only this benefit, but also you can show the demo for the customer effectively without deploying it on the server. It supports IP whitelist, http, https, tcp. The disadvantage of it is whenever there is a change in the system, it will create a new domain to access to localhost. Basically, it is mutable.

A screenshot of a computer

Description automatically generated with medium confidence

Figure 5: Example of ngrok on terminal

Graphical user interface, text, application

Description automatically generated

Figure 6: Example of ngrok – URL, output

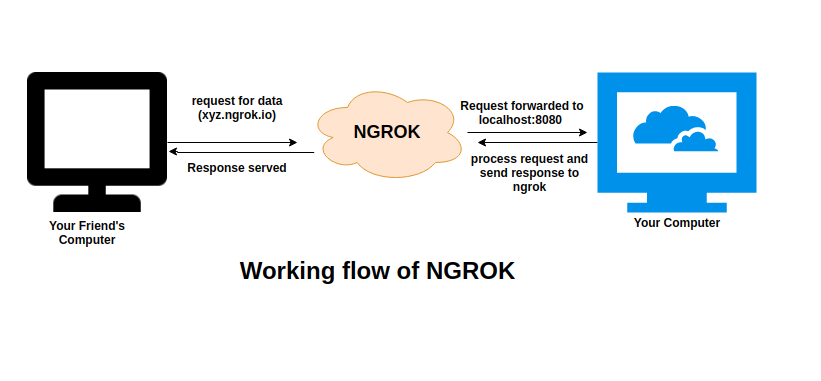


Figure 7: How ngrok works

**2.5. Javascript (JS)**

Graphical user interface, diagram

Description automatically generated

Figure 8: JS

Javascript is a programming language which is well-known in the world. Javascript plays a significant role in developing client side for web page behavior such as logging data, web page animation, etc, also with third-party libraries frequently incorporated.

Originally only used in web browsers, JavaScript engines are now found in a wide range of servers and apps. Node.js is the most popular runtime system for this language.

JavaScript is an ECMAScript-compliant high-level, frequently just-in-time compiled language. Its multi-paradigm which allows you to program in event-driven, functional, and imperative styles. Dynamic typing, prototype-based object orientation, and first-class functions are among its features.

Text, letter

Description automatically generated

Figure 9: Example of JS code for comparing answer in the question

**2.6. Teams Toolkit**

Teams toolkit is an extension for Visual Studio Code application. This extension helps developers construct and deploy Teams apps with integrated identity, cloud storage, data from Microsoft Graph, and other services in Azure effectively on Ms Teams without configuration.

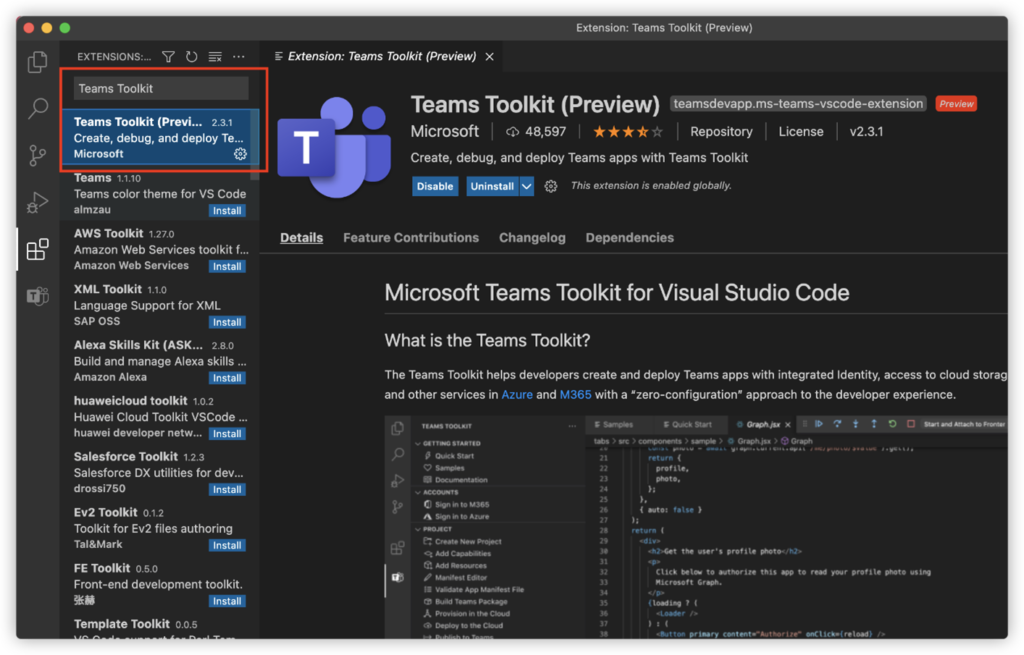


Figure 10: Teams toolkit

**2.7. Bot Framework SDK**

Bot framework provides a tool which helps developers create a bot. Bot can be also tested, deployed and managed through this framework. Not only that, but also developers can design bot as the way they want such as interpreting natural language, performing question and answer, etc.

Bot is an app that let people experience the communication between each other which is kinda like dealing with a real person but an intelligent bot. A bot interaction can be as simple as asking a question and receiving an answer, or as complex as a discussion that intelligently gives access to resources.

|  |  |
| --- | --- |
| Advantage | Disadvantage |
| The use of mechanisms generally reserved for human-to-human communication. | Time for implementation |
| Proactive Customer Interaction | Maintenance |
| Manage activities easily | Not a real person |
| Reduce human’s mistakes | Execute command 1 by 1. |

Table 1: Characteristics of Bot

Diagram

Description automatically generated

Figure 11: Example of Bot belong to Microsoft Document

Graphical user interface, diagram, application, Teams

Description automatically generated

Figure 12: Example of building bot belong to Microsoft Document

**2.8. Azure Active Directory (AAD)**

Azure Active Directory is a platform which its workspace is used for storing data of user on Microsoft’s cloud and also access management service. Azure AD services allow people to sign in and access resources such as external resources – Microsoft 365, Azure portal, etc, and internal resources.

Whenever an application has been deployed on AAD, the cloud will execute all the commands that have been sent from user to server.

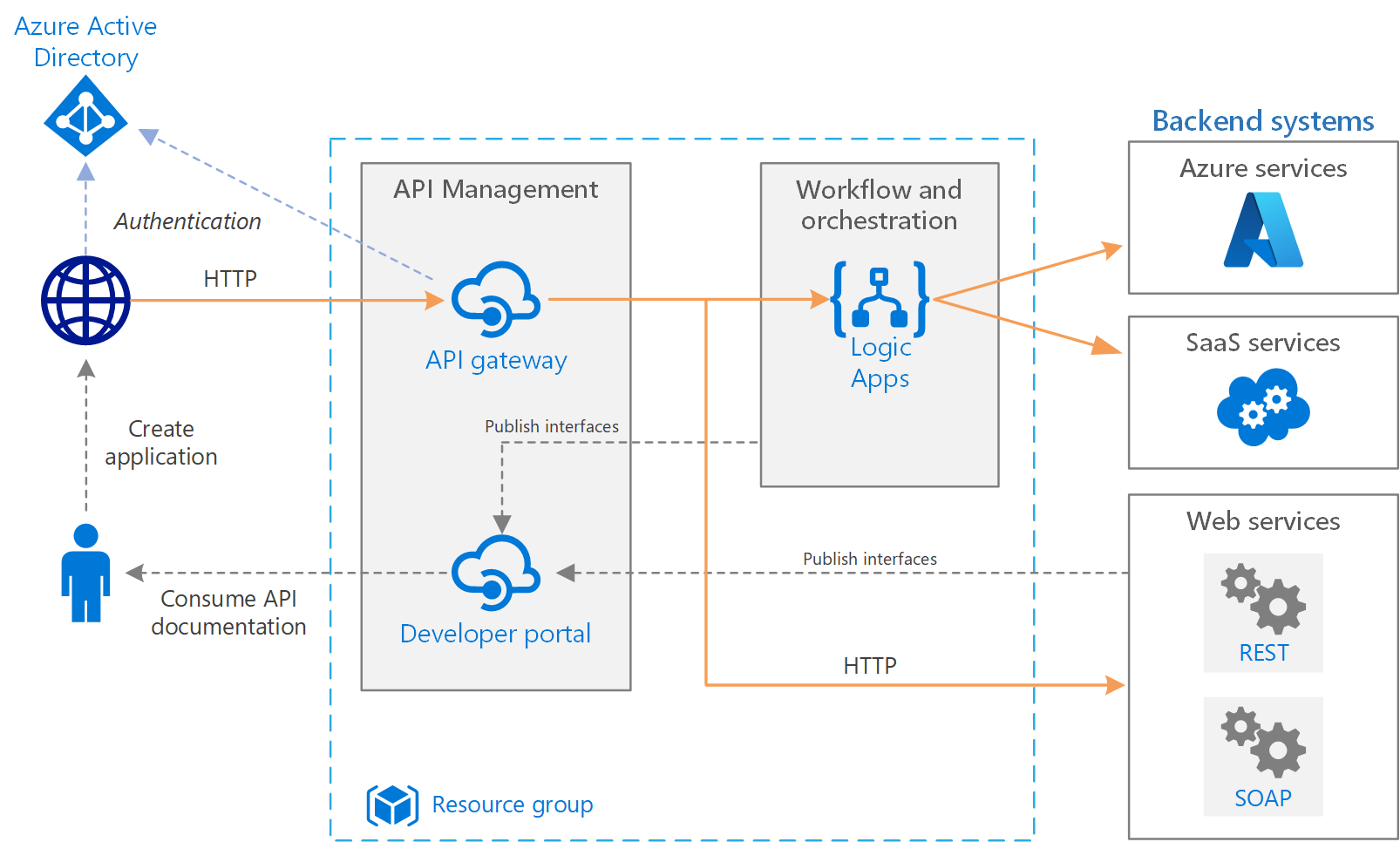


Figure 13: Azure AD architecture belong to Azure

Here is my source code that has been deployed on Azure AD. AAD will compile my source code and run it on the server automatically. It doesn’t need to restart the server on localhost after changes, it just needs to be deployed again to the cloud.

Graphical user interface, text, application, email

Description automatically generated

Figure 14: Azure AD resources

|  |  |
| --- | --- |
| Advantage | Disadvantage |
| Easy to deploy | Cost fee for subscription to be used. |
| Implement anytime | Functions are not possible for managing in details |
| Multiple platform functionality | Not support LDAP. Only via REST API. |

Table 2: Azure AD characteristics

**2.9. Microsoft Graph API**

A picture containing watch

Description automatically generated

Figure 15: Microsoft Graph belong to Microsoft Document

Microsoft Graph is a RESTful web API which allows you to access resources from Microsoft Cloud service. When an app has been registered and authenticated, you can access the Microsoft Graph API by the obtained authentication token.

Graphical user interface, application

Description automatically generated

Figure 16: Relationship between Microsoft Graph and others belong to Microsoft Document

**2.10. Overview of advanced Bot apps for education**

**2.10.1. Mongoose Harmony by Drift**

An intelligent chatbot and virtual assistant that was designed specifically for higher education applications to help meet growing demand for engagement and access.

Drift's chatbot helps higher education institutions evolve to meet the demand of the younger generation and can quickly route website visitors to appropriate staff and relevant content.

**2.10.2. Amazon’s QnABot**

Amazon’s artificial intelligence bot that makes use of Amazon Alexa and Amazon Lex provides a conversational platform where students can ask questions and easily sort through information.

Prioritizes the idea that students should have quick access to institutional answers that can provide tremendous value during the enrollment process.

**2.10.3. IBM’s Watson**

IBM Watson has been optimized as an interactive chatbot by universities across the world, from the UK to Europe and the US.

Expedite student responses, download and provide documents when needed, and answer subject-specific questions.

**2.10.4. HubBot by HubSpot**

An artificially intelligent chat service similar to IBM’s Watson and Amazon’s QnABot.

In addition to having the capability to answer basic questions per a pre-loaded script, HubBot can also book meetings, integrate with your existing HubSpot CRM, and track communications in an easy-to-filter way.

HubSpot's HubBot is geared to automate conversations and appear authentic, almost as if the answers were being provided via a live human interaction.

# CHAPTER 3. SYSTEM REQUIREMENT

**3.1. System Description**

Since this is a Bot custom app which is built to run on Ms Teams, the main requirement must follow these criterions:

+ The application’s package must be uploaded to Ms teams or at least be published on store.

+ The system must be deployed to work along with Ngrok and Ms Teams.

+ All the dependencies must be added which let the Bot work.

The command must be clear and input correctly as it should be by the user to let the Bot app execute the correct function.

**3.2. System use cases**

Diagram

Description automatically generated

Figure 17: System use cases

**3.3. System Functionality**

Diagram

Description automatically generated

Figure 18: BOT Entity Relationship

These functionalities of Bot will be shown as table below:

|  |  |  |
| --- | --- | --- |
|  | Function | Description |
| Client side | Ask Question | Send the question command to bot which let bot create question for student. |
| Reply Answer | Send the answer command to Bot which allow Bot to collect Answer from student. |
| Give Result | Send the result command to Bot which allow Bot to create a Result for the Question. |
| Compare Answer | Send compare command to Bot which let Bot to compare Answer and Result for that Question. |
| Attempt | Send attempt command to Bot which let Bot create the number of attempts for Answer. |
| End Question | Send the end command to Bot which let Bot close the Question and Answer can’t be replied anymore. |
| Export | Send export command to Bot which allows Bot to store all the datas that have been collected into a text file. |
| Server - side | Compiling all the source code | Compiling all the commands that have been sent by user. |

Table 3: Functionality of Bot

**3.4. System Workflow**

This diagram below describes how users have installed the Bot app in Ms Teams and communicate with it through messages. Its function will be compiled through Azure service. Then, a responding message will be sent back to the user on Ms Teams interface after executing commands.

Diagram

Description automatically generated

Figure 19: Bot's workflow belong to Microsoft Document

To be more specific, our Bot for education will be deployed on local server. After installed/uploaded the app package into Ms Teams, it will have an announcement of running. Running on local will reponse the message faster and you can check the response message in the console anytime. But on Azure AD server, it will respond slower than local. This problem of working on Azure AD will be explained in the “***Discussion***” section.

These diagrams below will describe how user interact with Bot app through Ms Teams interface by sending command’s messages such as “*Creating Question*”, “*Creating Attempt for the Question*”, “*Choose Question to add Answer*”, “*Adding Answer after choosing Question*”, “*Creating Result for the Questions*”, “*How to close the Question field*”, “*Compare between the Results and Answers*”, “*Print out the study’s data*”, “*Record all the datas into file on your local*”.

* + 1. **Create Question**

Diagram

Description automatically generated

Figure 20: Create Question

* + 1. **Create Attempt**

Diagram

Description automatically generated

Figure 21: Create Attempt for Answer base on Question ID

* + 1. **Choose Question**

Diagram

Description automatically generated

Figure 22: Choose Question

* + 1. **Send Answer and Check Answer’s Attempt**

Diagram

Description automatically generated

Figure 23: Send Answer message

Diagram

Description automatically generated

Figure 24: Check Answer Attempt

* + 1. **Close Question**

Diagram

Description automatically generated

Figure 25: Close Question section

* + 1. **Create Result For Question**

Diagram

Description automatically generated

Figure 26: Create Result

* + 1. **Compare Answer and Result**

Diagram

Description automatically generated

Figure 27: Compare Answer and Result

* + 1. **Show Question And Answer**

Diagram

Description automatically generated

Figure 28: Show Value of Question and Answer

* + 1. **Record into File**

Diagram

Description automatically generated

Figure 29: Record into file

# CHAPTER 4. IMPLEMENTATION

**4.1. Setup environment**

***Step 1:***

You need to install the Teams Toolkit extension on Visual Studio Code, which allows you to create a basic Hello Bot easily without configuration. After it is installed, you open the extension

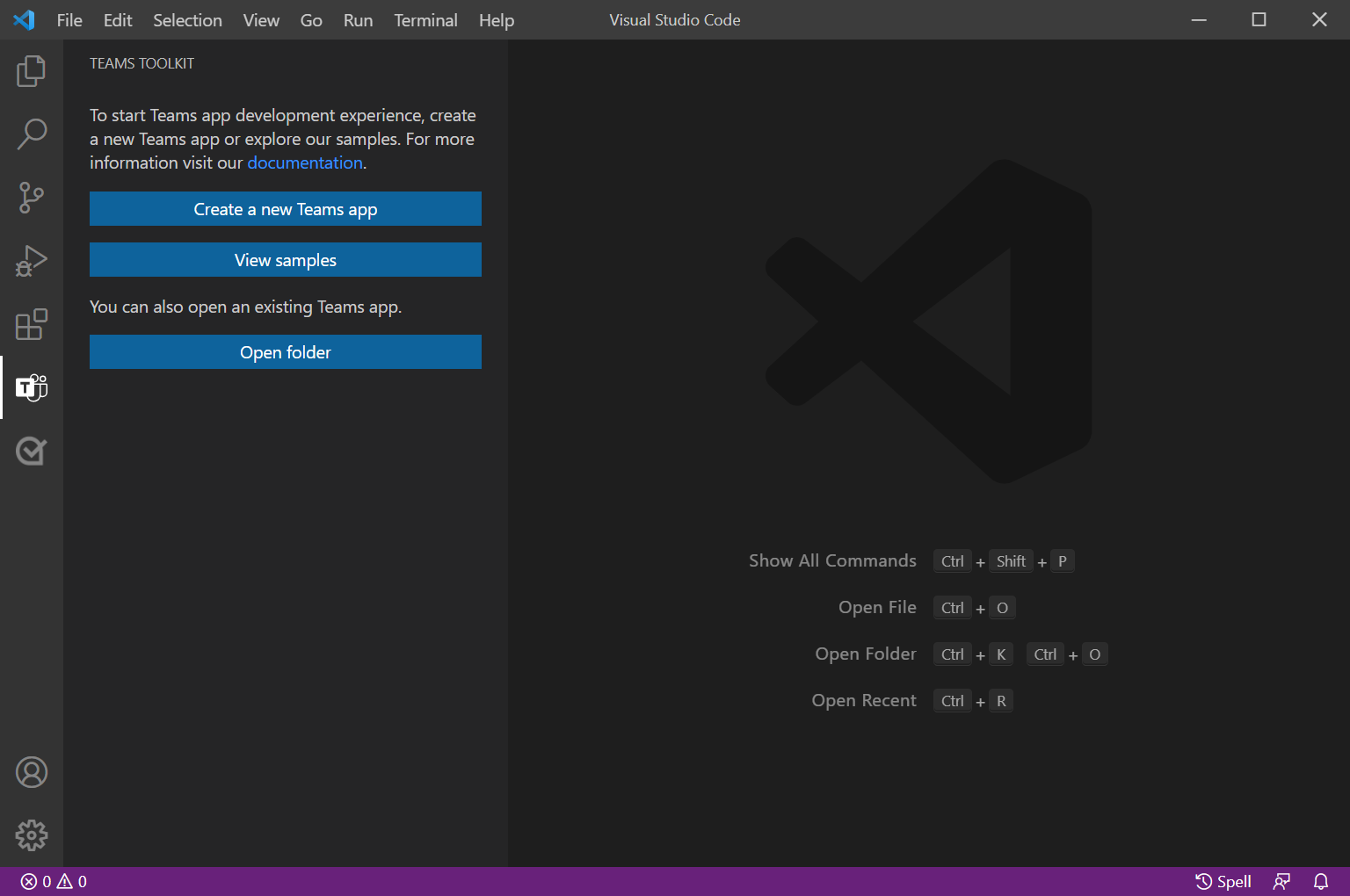


Figure 30: Open Teams Toolkit

***Step 2:***

You choose the section “***Create a new Teams app***” which allows you to construct the Bot.

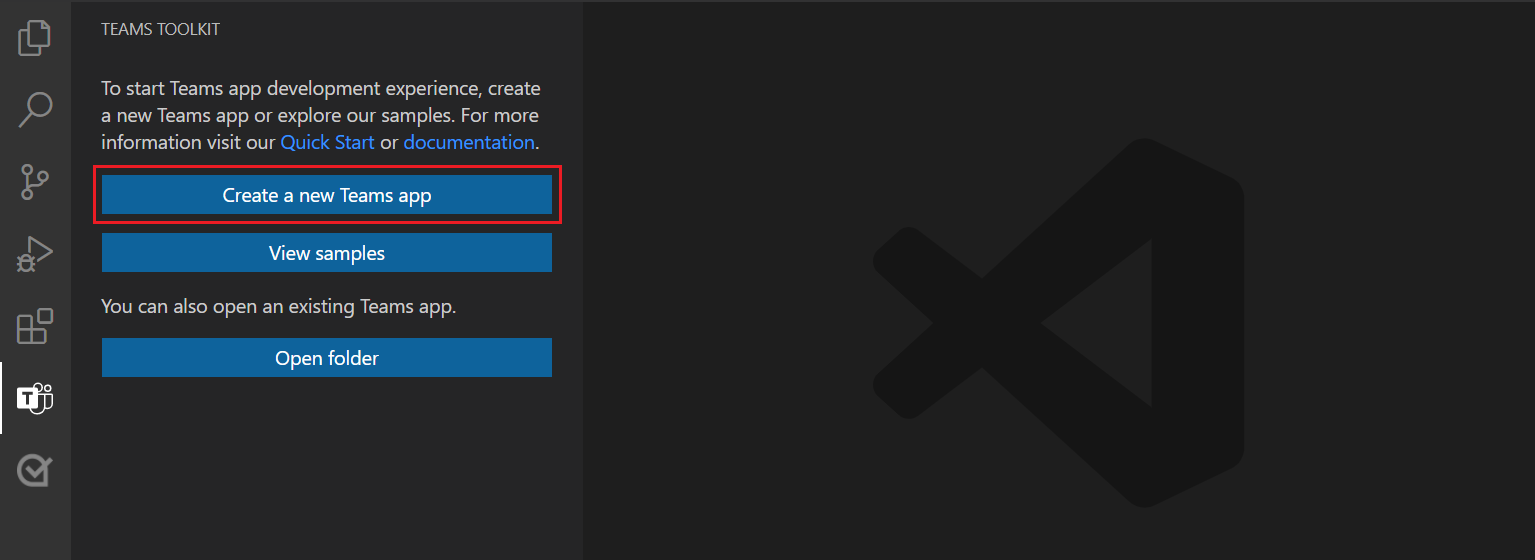


Figure 31: Create section

***Step 3:***

It will pop out a field with 2 options for you to choose. Because we need to create a totally new app, so we choose the first option.

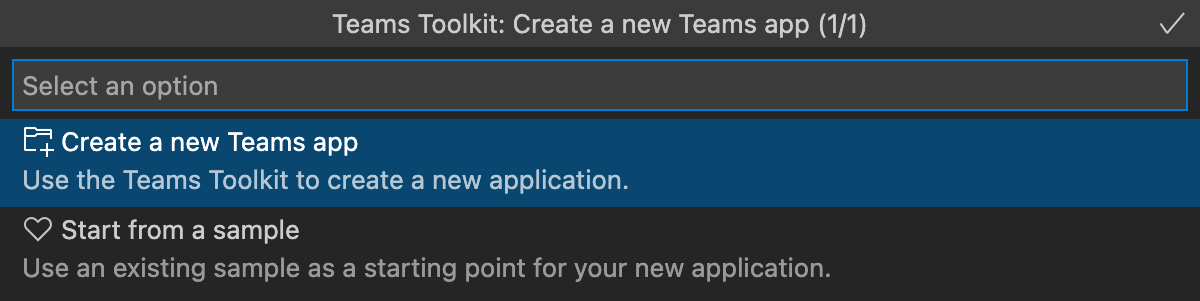


Figure 32: Create new app

***Step 4:***

After chosen “***Create a new Teams app***”, it will show the type of creation for you to choose. Because this project is about Bot, so we select “***Bot***” option and deselect all other options. After that, choose “***OK***” on the right angle at the top corner to continue.

Text

Description automatically generated

Figure 33: Bot section

***Step 5:***

Select the language which you prefer in programming. Here, I choose “***Javascript***” language which I’m familiar with for my Bot app.

Graphical user interface

Description automatically generated with medium confidence

Figure 34: Programming language

***Step 6:***

The final step will be giving a name for the application after you have finished all the steps before.

Graphical user interface, text, application

Description automatically generated

Figure 35: App name

If you’ve done all these steps correctly, the app would have these files excluding *answer* and *answerfile*. After that, you can program your functions which you want Bot app to have in “***teamsBot.js***”.

Graphical user interface, application

Description automatically generated

Figure 36: App files

**4.2. Required tools**

* + 1. **Node.JS and NPM**

Node.js is a platform that allows you to create asynchronous and event-driven network applications. It comes with built-in HTTP server libraries, which allows developers to build their own web server and highly scalable web applications on top of it. Because we are building a Bot app, so this must be installed. You can download it through this website:

<https://nodejs.org/en/download/>

When you’ve finished the installation, Node.js and NPM are now running on your device.

* + 1. **Ngrok**

Because we have installed the Teams toolkit before, which has included ngrok application. This saves a lots time for downloading and configuring the thing you need to run the project. Or if you want to download manually, check out this link:

<https://ngrok.com/download/>

* + 1. **Visual Studio Code (VS Code)**

Visual Studio Code is a software which helps developers to program a project by a programming language. Its platform can let you arrange the code more beautiful as you want it to be. It also supports many languages that you are familiar with. To download it, check out this link:

[https:// code.visualstudio.com/download/](https://ngrok.com/download/)

* + 1. **Teams Toolkit**

Teams Toolkit is an extension which helps developers to create and manage Bot app easily. It also has many useful features such as deploy, zip app package, publish app, etc. You can install it through Visual Studio Code’s extension. Because we are building Bot app to run on Ms Teams, so this must be installed. After installed, it also included Ngrok. Whenever you run Bot app, Ngrok also will be run on the VS Code platform by Teams Toolkit, which you don’t have to configure Ngrok again.

Graphical user interface, text

Description automatically generated

Figure 37: Teams Toolkit features

* + 1. **Microsoft Teams (Ms Teams)**

Microsoft Teams is a software which allows people to meet and work online without going outside. Ms Teams is very effective nowadays. It is also a platform that has many custom apps that you can work with. It can be used on website or on an application. To download this app, check out this link:

<https://microsoft.com/vi-vn/microsoft-teams/download-app/>

**4.3. Implementation and Result**

In this section, I will demonstrate for you how the Bot app works in details as possible as I can.

First of all, when you’ve created a new Bot app and want to run it locally to test it, you have to open the file “teamsBot.js” and press “F5” to make the Bot running at localhost. If it runs correctly, it will show up the MS Teams on website to let you add the bot for local use. Because ngrok has been included in teams toolkit, so whenever you run it, it will automatically create a tunnel on ngrok for you.

A screenshot of a computer

Description automatically generated

Figure 38: teamsBot’s file running

Graphical user interface, text, application

Description automatically generated

Figure 39: Add Bot local

Then click “Add” section to continue to add and chat privately with Bot app. In Figure 40, it will show you the result after adding it for private chat. To add the Bot app in a channel of a group, you need to upload the Bot app package through the channel’s apps by “Upload a custom app” button. In the Figure 41, the BOT app has been added into channel. There will be an announcement in the “Post” section of the channel, which let you know the BOT is running as Figure 42.

Graphical user interface, application, Teams

Description automatically generated

Figure 40: After added

A screenshot of a computer

Description automatically generated

Figure 41: Bot added into a channel

Graphical user interface, application, Teams

Description automatically generated

Figure 42: Bot announcement

Because the Bot app just has a simple function such as “Hello”. Therefore, we need to add the new functions we want into the app. However, we need to fetch the user information on MS Teams first to be able to move to the next section. In Figure 43, this function will get the token from the user when the user sends a message to Bot, which has a lot of information about the user such as user ID, name, email, content of message, etc.

Graphical user interface, text, application

Description automatically generated

Figure 43: Fetch user info

Text

Description automatically generated

Figure 44: User's token

In order to call the Bot and send commands in a channel of a group in MS Teams, we have to mention Bot every time when we want Bot to do the work. Then, Bot will execute and respond to the command that has been received in terminal in Figure 45 and Figure 46. To modify for Bot app to execute the command “Question” and create questions, we will have function for this section in Figure 47.

A picture containing text

Description automatically generated

Figure 45: Mention Bot

Text

Description automatically generated

Figure 46: Running message in terminal

Graphical user interface, text, application, email

Description automatically generated

Figure 47: Question

This function allows the bot to create multiple questions, whenever there is a user input many questions at the same time. However, if you want to send a command to Bot in a channel, you must mention it and send the command next to the mention tag. Or else it won’t recognise the type of message. It will execute command in terminal and response with the message on the Client-side in Figure 48. Next, we continue to the “Answer” section. Whenever a lecturer give the question, the Bot will execute “Question” command and create question. Students have to answer this question. To execute the “Answer” command, we program as in the Figure 49.

Graphical user interface, text, application

Description automatically generated

Figure 48: Question message

Text

Description automatically generated

Figure 49: Answer code

However, because of multiple questions, students have to choose the “Question ID” which is the ID of a specific question that has been created. This “Question ID” is the number of index of Map that I’ve configured for the Bot which will start from 1 to many. In Figure 50, the ID of Question will stand before the Question which let you know what question base on that ID. We program as Figure 51 to choose ID of Question to store the Answer. After choosing a question to answer, you can send “Answer” through private chat for this Bot. It works normally in both the channel of a group or private. The replied “Answers” will be collected by bot and stored on the server.

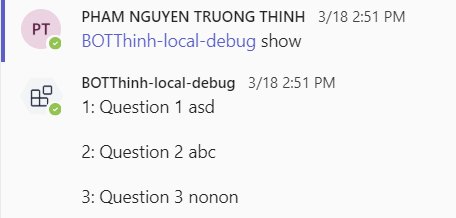


Figure 50: Question ID

Text

Description automatically generated

Figure 51: Choose question ID

Graphical user interface, application, Teams

Description automatically generated

Figure 52: Choose and Answer command

After “Answers” have been replied by Students and collected by Bot, Lecturer will give the Result for Bot to collect and compare with the Answers of students. To give the specific “Result” for “Question”, we have to include the “Result” with “Question ID” as the same as “Choose Question” section. The “Result” will be stored by Bot on the server as the same as “Answer”. After all these steps have been finished, the lecturer must use the command “End Question” for a specific “Question” to close the field which will not allow students to give “Answer” anymore for Bot to collect. Because each question has a different time that Lecturer gives to students. Or you can use command “End all” to close all the Questions at the same time. After closing the “Question” field, if a student sends “Answer” to Bot app. The Bot app will not recognise anymore and response with message as Figure 56.

Graphical user interface, text

Description automatically generated with medium confidence

Figure 53: Result for Question ID

Text

Description automatically generated

Figure 54: End Question ID

Text

Description automatically generated

Figure 55: End all Questions

Graphical user interface, text, application

Description automatically generated

Figure 56: Question after closed

The next function we are going to discuss is “Compare” the “Result” with “Answer” after collected by Bot app. We also use command “Compare Question ID” to compare every “Answer” of each user in a specific “Question” field. Bot will filter the “Answer” of each student every time and compare it with the Result that has been collected. If a student has a correct or wrong Answer, the Bot will response with message as Figure 58.

Text

Description automatically generated

Figure 57: Compare Question ID

Graphical user interface, application

Description automatically generated

Figure 58: Compare Message

Whenever the lecturer wants to check who has “Answer” first of a specific “Question”, the command “Show Question ID” will be used as Figure 59. This command will let the bot print out “Answer” message of students. Bot app will run the command and response with a message. Because the Bot executes one by one command, therefore who was the first one that had sent “Answer”, would be stored on the server. The “Answer” message will be printed in order as Figure 60. When the class has finished, the lecturer wants to export all the datas which have been collected from the beginning of the lesson, thus we send the command “export” to Bot app. Bot will execute and store all the datas to a text file on your local. The content of the file such as student’s name, timestamp of message, ID of the student, student’s answer will be Object as Figure 62.

Text

Description automatically generated

Figure 59: Show Answer

Graphical user interface, text, application, email

Description automatically generated

Figure 60: Print out message

Graphical user interface, text, application

Description automatically generated

Figure 61: Record into file

Graphical user interface

Description automatically generated

Figure 62: Answer file

# CHAPTER 5. DISCUSSION

Azure AD is a service that helps developers a lot. If you are using one platform of Microsoft, then you can access data easily through Azure AD server which is called multiple platform functionality. Its security is very high and valuable for developers. It also has single sign-on for multiple applications. It’s very popular because of global availability. In summary, it worths a try at least once.

However, all the benefits of it are suitable for experienced developers in working on Azure AD or working for a big company that using Azure services. For a newcomer or freelancer, it takes time to get used to it and hard to apply it in the project. Because it costs a lots of money to maintain the server and run the project. Not only that, Azure server’s responding time is also not stable. It sometimes responds quickly, but sometimes very slowly or not responding at all. Azure AD also needs to verify identity which you need to be Azure’s partner to access secured datas in the cloud. The process of becoming a partner takes a very long time to complete.

Besides, one of the major drawbacks of Bot is the number of queries it can resolve. At a certain point in time, it will have to connect to an actual human to resolve the issues. They also have limited replies and solutions which can leave a customer unsatisfied. It is imperative to understand that a well-designed chatbot that can handle more tasks than others will also cost higher. So this pushes the investment of the company for the chatbot further. Chatbots are effective in many industries and for varied applications, but they cannot handle all of them.

Now, there are certain services like hospitals, fire brigades, and the police that require immediate responses. In such cases, investment in chatbots might not yield any sustainable advantage.

Essentially, chatbots are machines. They will make mistakes and sometimes even provide wrong solutions to the customers. So companies have to be wary of it and work on improving their chatbots further to avoid such incidents.

# CHAPTER 6. CONCLUSION AND FUTURE WORK

* 1. **Summary**

This thesis already demonstrated how to apply the Bot technology into teaching and studying for online-class on Ms teams. Also we have learnt a lots of knowledge about the Bot’s technology and the basic workflow of it. This is an unforgettable experience.

Thanks to this, it can help the student to solve the Question smoothly without any problem. And Bot will help both lecturer and student in class in a convenient way.

Besides, Bot technology still needs a long way to go. Not only for Question and Answer, but also it can be used in many study’s fields such as health, social, etc.

* 1. **Future work**

This Bot still needs to improve more functions in the future. The experiences can be improved for the user after adding more advanced features. For example, Allow students to send command messages in the channel without mentioning the Bot which is very convenient for the user. Because this function is only available for developers and not usable for the user. The new function it should have which is to recognise the speech as the command to let it work in the same way as Siri, Cortana, etc.

In summary, Ms Teams still does not have many advanced features to configure for the Bot technology. I hope that Microsoft will improve this and make Bot’s technology more popular in the future. Not only just solve problems for education, but also help people to interact with Bot as a real person.

**LIST OF REFERENCES**

**Bibliography**

[1] Bot Framework SDK Documentation. Available: <https://docs.microsoft.com/en-us/azure/bot-service/index-bf-sdk?view=azure-bot-service-4.0>. Accessed: January 19, 2022.

[2] Microsoft Graph API Documentation. Available: <https://docs.microsoft.com/en-us/graph/use-the-api?view=graph-rest-1.0>. Accessed: January 19, 2022.

[3] Microsoft Teams Documentation. Available: <https://docs.microsoft.com/en-us/microsoftteams/teams-overview>. Accessed: January 20, 2022.

[4] Azure Active Directory Documentation. Available: <https://docs.microsoft.com/en-us/azure/active-directory/>. Accessed: January 20, 2022.

[5] Chatbots applications in education: A systematic review. Available: <https://www.sciencedirect.com/science/article/pii/S2666920X21000278?via%3Dihub>. Accessed: January 23, 2022.