

<Restaurant Recommending Chat-Bot>

Final Project Report

DT228

BSc (Ord) in Computer Science

**<YONG HO MOON>**

**<MICHAEL COLLINS>**

School of Computing

Dublin Institute of Technology

**<30 APR 2019>**



Declaration

I hereby declare that the work described in this dissertation is, except where otherwise stated, entirely my own work and has not been submitted as an exercise for a degree at this or any other university.

Signed:

YONG HO MOON

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

<Student Name>

YONG HO MOON

<Date>

30 APR 2019

Table of Contents

Abstract ………………………………………………………………………………. 5

1. Introduction

1.1. Overview of the project and the background behind it ………………….. 6

1.2. Project Aims and Objectives …………………………………………….. 7

1.3. Project Scope …………………………………………………………….. 8

1.4. Thesis Roadmap …………………………………………………………. 8

2. Background

* 1. Introduction of this Chapter ……………………………………………. 10
  2. Introduction of HEROKU ………………………………………….…... 10
  3. Introduction of Dialogflow ……………………………………………... 12
  4. Introduction of PostgreSQL ………………………………………….… 12
  5. Introduction of Node.js …………………………………………….…... 13
  6. Conclusions …………………………………………….…………….… 14

1. Design
   1. Introduction of this Chapter …………………………….…………….… 15
   2. Methodology …………………………………………………………… 15
   3. System Architecture (3-tier) …………………………………………… 17
   4. Front-End Design and Use Case …………………………….…….……. 19
   5. Middle-Tier Design …………………………….…….…….….….……. 21
   6. Back-End Design (E-R diagram) …………………………….…….…… 21
   7. Conclusions …………………………….…….………….……….….…. 22
2. Development
   1. Introduction of this Chapter …….………….………….………….……. 23
   2. Front-End Development ………………………………….…….…….… 23
   3. Middle-Tier Development ………………………………….…….……. 26

4.3.1. Code Explanation ……………………………………………….……. 26

4.3.2. Chat-Bot Platform ……………………………………………………. 30

* 1. Back-End Development ………………………………….….…….….... 32
  2. Conclusions ….….….……….……….….….…….….…….…………… 33

1. System Validation
   1. Introduction of this chapter …….……….……………………………… 34
   2. Testing Methodology ….……….……….……………………………… 34
   3. Testing Result ……….……….……….….……….……….…………… 35
   4. Conclusions …………….….……….……….….……….……………… 36
2. Conclusion
   1. Introduction of this Chapter ...………………………….…….…….…… 37
   2. Conclusion ……………………………………….….….……………… 37
   3. Future Work ……………………………………………………….…… 38

References ………………………………………………………………….….….… 39

Abstract



On 2018 May, Google announced Google Duplex which makes a reservation through Google AI. The announcement enthralled people by showing artificial intelligence that allowed the Google AI to talk naturally with people and making a reservation. Although this announcement was performed in the shape of phone call reservation, if we just change the I/O tools from phone to messenger, it can work same as it did on the phone.

Under the heading of Google and Facebook, there were many announcements of AI development and many chat-bot platforms were released to the public. Naturally, interest toward chat-bot has increased and I also was interested and started this project.

The meaning of this project is to make user not using a command to activate chat-bot’s function but from the conversation between user and chat-bot, chat-bot can activate its restaurant recommending function naturally. So, the user can feel that the chat-bot is running smoothly.

1. Introduction

* 1. Overview of the project and the background behind it.

For myself, as the newcomer of Dublin, Ireland the first thing that worried about was food. Including me, when most of the people becomes a newcomer in certain country, their natural concern will be the local food. If the food there doesn’t taste good for them, they miss their home food or tries to find decent restaurant. From that point, I tried to find a way to help people with building a software program that people can easily utilize. One of the best ways was to build a chat-bot running on the Social Network Service that can recommend those people a decent or even great restaurant that they can enjoy.

As chat-bot development area is in still development, the most interested area in the development is about making the chat-bot work smoothly so that when people have a conversation with the chat-bot, they don’t feel any awkwardness during the conversation. I also wanted to make the chat-bot run smoothly, so I thought that if the chat-bot run based on command such as “/find food” or “/food” it will be awkward and tried not to use any command to run certain function of the chat-bot.

Since Facebook is at the top of the global SNS popularity rankings, I thought the chat-bot running on the Facebook will be easy for user to access and utilize it. Also, to get the information of the restaurant and send it to user, web crawling was needed. All the information was gathered from <https://www.yelp.com>, a website which has a good deal of information of restaurant. There was much information in there and I couldn’t show all the information to the user because it might seem messy. So, I tried to make the restaurant listed in the order of users review evaluation. Also, I tried to let user know if they are having biased meal throughout their search history.

* 1. Project Aims and Objectives

As mentioned above, the main purpose of this project is to build a chat-bot running on the Facebook Messenger that helps people by recommending restaurant when they are having hard time with selecting a restaurant. The main targeted user can be a tourist in that country or the newcomer to any country who are having hard time with the food and having less information in that place. The chat-bot should be easily accessible and easy usable to all people using SNS. In this project, three mainly focused problems were set up.

First problem was an interlock between server, database, chat-bot platform and Facebook messenger page. Especially, as the chat-bot is deployed on the server, if the server breaks down, all the connections does break down. So, finding an adequate server and programming language for dynamic programming was needed.

Second problem was to make chat-bot provide appropriate answers to user input values. If it’s about conversation between user and chatbot, to make chat-bot run smoothly, it needs to be trained with lots of datasets to make it reply adequately. Also, the database needs to get the right input value of user and keep it in the database.

Third problem was finding an information of restaurants when the input values were given by user. As users input values can be differed anytime, the chat-bot needed to start searching when the input values are given. Also, during the usage of the chat-bot, user should not feel any inconvenience in getting an information of restaurant and having a conversation with chat-bot, which means server need to remain actively while users are using the chat-bot program.

* 1. Project Scope

This project covers the connection between server and Facebook Application Programming Interface, server and database, and programming language to connect server, front-end and database. It also covers the web crawling and the way to get the specific information through web crawling. In this project, the workflow of server and chat-bot platform will be included. As it will be mentioned more specifically in Chapter 7, the Future Work, this program is a placeholder for the future work so that it will be keep managed with version upgraded.

As mentioned above, this messenger stores database to save users input value and check users’ unbalanced eating habit. If user keeps searching same food over and over again, based on the data in the database, this messenger will let user know that he or she is keep searching the same food. This personal information will be prohibited from commercial use but only for research purpose of finding out users’ taste.

On the other hand, this project does not cover finding adequate algorithm of creating sentence in the chatbot and training algorithms to make chatbot’s artificial intelligence upgraded.

* 1. Thesis Roadmap

Including this chapter, there are 6 chapters.

In the chapter 1, I mentioned the project background, its aim and scope of the project.

In the chapter 2, I will mention the backgrounds of each tool and module that I used.

In the chapter 3, I will mention mainly about system design. I will mention the system architecture, front-end design, back-end design and the conclusion.

In the chapter 4, I will talk about the development, front-end development, middle-tier development, back-end development and conclusion of the chapter.

In the chapter 5, results and evaluation of the program will be mentioned. Evaluation and result for white and black box test that have been done will be mentioned.

For the final chapter, the chapter 6, I will talk about the conclusion of this project, and future work that will be done afterward.

2. Background

* 1. Introduction of this chapter

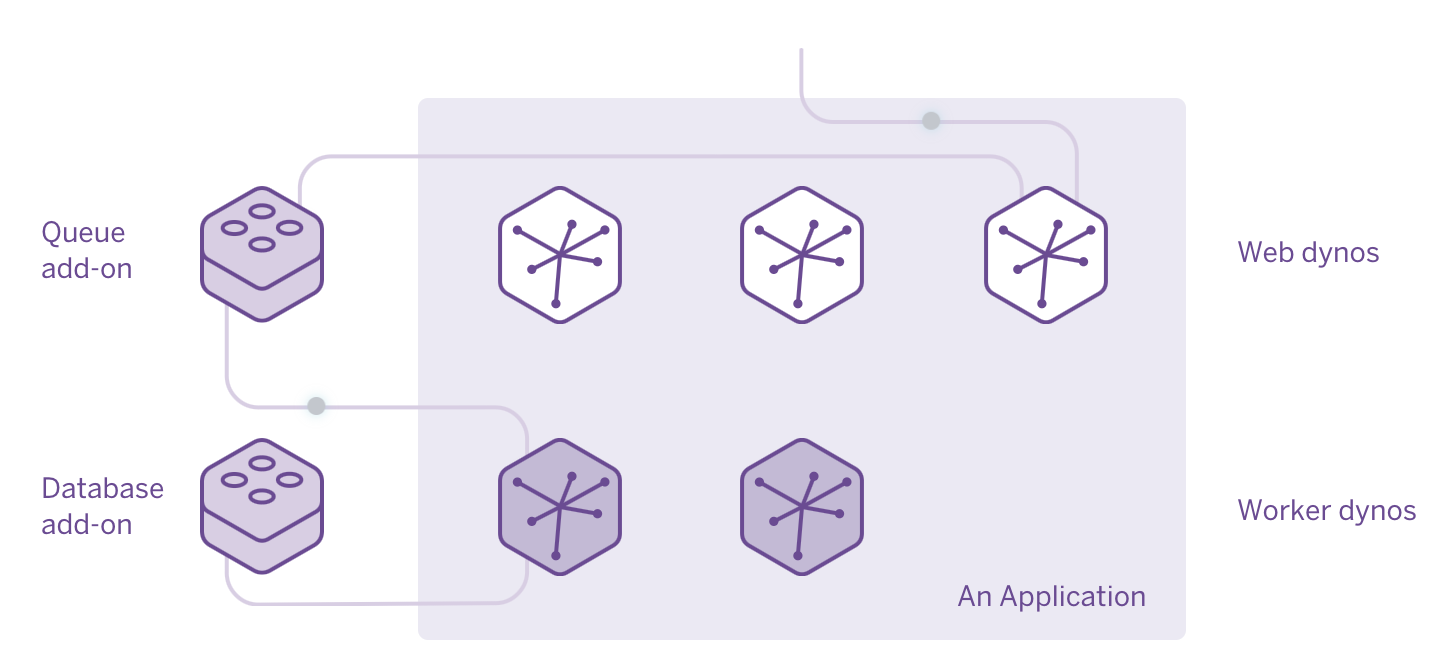
In this chapter, the background of a server, database and platform for training chat-bot will be introduced. For chapter ‘2.2’, background of server, HEROKU, will be introduced. In chapter ‘2.3’, background of database, PostgreSQL, will be introduced. In chapter ‘2.4’, background of chat-bot platform, Dialogflow, will be introduced. In chapter ‘2.5’, the background of programming language, Node.js, will be introduced. For the last chapter, conclusion and connection of the backgrounds will be mentioned briefly.

* 1. Introduction of Server - HEROKU

Heroku is one of the first cloud platforms that has been developed since June 2007. It is a cloud platform as a service (PaaS). PaaS is a really useful tool for developers as they don’t need to install or setup the base development environment. Network configuration or the auto-scaling is also not needed. By some codes written by developer, this platform executes it for the user. It also supports lots of programming languages such as Java, Python, Node.js.



*Figure 2.1 HEROKU platform workflow [1]*



*Figure 2.2 HEROKU – Dynos [2]*

On *Figure 2.1* and *Figure 2.2*, you can see the workflow of the HEROKU platform. The HEROKU uses container model to run an application that are deployed. It is called as dynos. Dynos are isolated, virtualized Linux containers that are designed to execute code based on a user-specified command. By deploying this application, HEROKU manages the dyne and frees user from managing infrastructure.

When the application gets web request, the request is sent to random web dynos. From there, the request is sent to a queue that is connected as an add-on and when it is placed in the queue safely, web dynos send user a success message.

The worker dyno picks up the request from the queue and performs the request. If the worker dyne is connected to a database, it can persist the result of the work in the database and returns user a different request. One of the advantages of HEROKU is that it offers a good quality of free dyno. Dyno which has 512MB sized RAM and 2 processors that is always turned on is offered for free. Even there is a little disadvantage, which when the deployed application is not in use, the dyno goes into sleep mode so if the user is trying to use it during its sleep mode it takes a little time to wake it up, it still was a good option for me to run my chat-bot.

* 1. Introduction of Database - PostgreSQL

PostgreSQL is open-source, object- relational database management system. As an object-relational database, it offers transaction and database’s property - ACID (Atomicity, Consistency, Isolation, Durability). There are two advantages of PostgreSQL, first, not like Oracle DB, MS-SQL, it does not need any payment for the license usage. Secondary, it is famous for its good stability and reliability. Although there are some relative disadvantages in PostgreSQL when it is compared to Oracle DB, MySQL, MS-SQL, such as inefficiency in data replication, inefficient architecture in writing. But considering its open-source feature, when other databases are maintained in the company such as Oracle Corp. and Microsoft, it offers good quality.

* 1. Introduction of Chat-Bot Platform - Dialogflow

Dialogflow formerly known as Api.ai is a Google-owned developer of human–computer interaction technologies based on natural language conversations. After Google buys this company in September 2016, it started to support Google Assistant which has voice based interactive interface and this becomes the infrastructure of Google Duplex that I mentioned in Abstract chapter. The company offers a platform which helps natural language processing. By using this platform, developer can set up their own conversation flow and the context management. With conversation flow, developer can decide how to make the dialogue go on and with context management, developer can make the chat-bot study the context and upgrade its artificial intelligence. As the platform is connected with front-end page, it receives request from the page and give response to the page. One of the advantages of the platform is that it supports many front-end pages such as Facebook Messenger, Google Assistant, Slack, Twitter and so on.

* 1. Introduction of Programming Language - Node.js

Node.js is an open-source, cross-platform JavaScript run-time environment that executes JavaScript code outside of a browser. One of the features of Node.js is that it runs asynchronously. Normally, if the server runs synchronously and if clients try to connect server and send requests, as it is thread-based synchronous, memory usage goes higher, and it concludes server down. But, in Node.js by, using event-based asynchronous feature, only one thread is enough to take care of clients’ requests which gives less burden to the server. Also, Node.js covers front-end, back-end page with JavaScript language. This feature gives user utility when he needs to deal with server or front-end page. That is, on the development page of project, or on the page of maintenance, it gives lots of benefit in time, human resource, and efficiency. Although, there are some points that need to be aware such as if it takes too much time in one request then overall system performance goes low and if looped callback happens (callback hell), code readability drops significantly, but these problems can be settled with frameworks and testing.

* 1. Conclusions

In this chapter, I mentioned a server, database, bot platform and programming language. There are some advantages and disadvantages for each tool, but I thought each selection for server, database and platform will be good. About the brief design for the program, which will be explained in the next chapter, Node.js will be connecting server, database and bot platform. Data will be saved locally and also in the server. Queries to save data will be sent by Node.js from server to database. Details will be explained more in the next chapter.

3. Design

3.1. Introduction of this chapter

In this chapter, design of the software will be given. Development Methodology will be introduced in ‘3.2’. In this project, test driven development method was used so the concept of Test-Driven Development will be explained. In ‘3.3’, system architecture of client and server will be introduced. In ‘3.4’, front-end design of software, which is a Facebook Messenger and use case of it will be mentioned. In ‘3.5’, the Middle-Tier Design, method of connecting messenger and server will be introduced. In ‘3.6’, which is Back-End design, the E-R diagram of PostgreSQL database will be introduced. Finally, in ‘3.7’, conclusion of this chapter will be mentioned.

3.2. Methodology

In this project, Agile, Test-Driven Development method has been chosen for the project process. Between the Agile and test-driven development, both are almost same except the scale of it. So, in this report, I will only mention a test-driven development. In test-driven development, short cycle of development is repeated. For my case, every week was my development cycle so that test plan for each week was needed. There are three reasons of choosing test-driven development.

First, test-driven development mainly has two advantages, feedback and cooperation. Normally, cooperation is useful for group project. Tester makes the lists need to be tested as document so that developer can figure out where the fault occurred. But, in this project I worked as developer and as tester, so that I was able to find out tested list from the document and revise code from it, and this means that the development method gives a good feedback. Throughout the documentation work, the result becomes visualize and makes me easy to find out the result of each unit test.

Secondly, this development method is optimized when the project’s topic is not familiar with a developer. As the topic is not familiar with the developer, uncertainty of developer goes high. In this situation, test-driven development helps developer with their development by setting test-case before the development and giving guidelines for the development phase.

About the concept of test-driven development, as it says, test becomes the subject of development. Normally, in software development, test is done after the development is finished, but in test-driven development during the development phase, test-cases are made for each cycle and development focuses on the pass of the test. In each cycle, with result of the test, developer gets feedback from the test-case and keeps checking if the unit is working properly. More specifically, each cycle of test-driven development goes like this –



*Figure 3.1 Cycle of Test-Driven Development [3]*

1. Create test plan for a unit development, this test is called flawed test so need to be reworked later. This test plan helps developer with understanding what the developer has to do to complete successful test result.

2. Write down code to fulfill the test expectation. It means that, the code does not need to be perfect. There are expected grade result in each test and the code only need to exceed the minimum expected grade result. Also, in this section, developer must focus on what is written in the test case and what isn’t.

3. Last section is refactoring section. In this section, developer need to clean the code and put comment with the code. During the procedure, developer must aware of damaging the code or making critical system error during the process. After the cleaning and commenting procedure, testing need to be done again to ascertain no error has occurred.

After completing a cycle for test-driven development, other cycles are taking place until the project’s increment process is done. Although this development methodology takes more time than other development methodology, I can expect more specific result for each unit testing and less faults occurrence with using this methodology.

3.3. System Architecture (3-tier Architecture)

The chat-bot messenger is consisted of 3-tier architecture. Each tier represents client tier, application tier and data tier. The reason of using 3-tier architecture instead of 2-tier architecture is that if the 2-tier architecture is chosen, as all the clients need to be connected to the database, it gives a limitation with consisting of certain number of clients. But in 3-tier architecture, as application server takes care of the queries of database, the database gets less connection numbers of client so that it can run effectively.

First of all, about the client tier, which also represented as the GUI or the Front-End tier, offers the user interface. In this project, the user interface will be the Facebook Messenger. By the interaction with the application tier, it will send request to the application tier and refer the data. As mentioned above, as application tier takes care of all the queries, this client tier does not cover any data management source code or business logic.

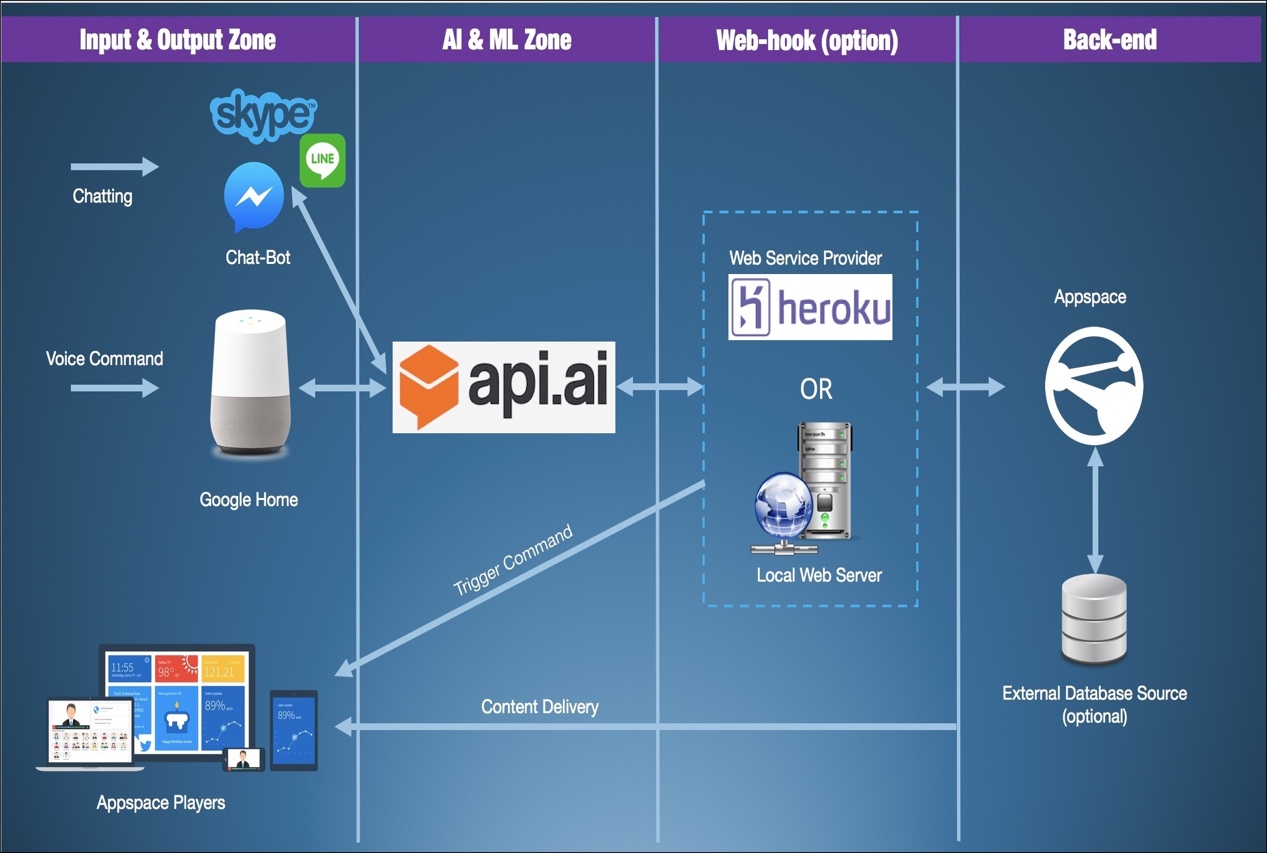
The application tier, also called as transaction tier, takes care of information processing. There are two zones. First zone is an AI & ML zone and second zone is a web-hook zone. In the AI & ML zone, the chat-bot platform – Dialogflow takes care of the training of the chat-bot. As developer gives lots of training data to the platform, the platform helps with a training with the given data sets.

In the case of the Facebook Messenger application, all the input and output values are communicated via web-hook. The server, HEROKU, takes care of the connection between server, client and also the AI & ML zone by the way of web-hook events. The way that Web-hook works will be explained more specifically later.

In the data tier, database is stored and access to the database and reading or writing data are done. In this project, the PostgreSQL is used for the database and data of users’ input will be stored. Queries to ask data from the database will not be included in this tier as the queries will be written by Node.js code.

On the next page, for the easy understanding, you can see the system architecture.

To summarize it short, Input & Output Zone will take only Chatting tool which is Facebook Messenger. The AI & ML Zone will take Dialogflow(api.ai) for the chatbot training. For the web server, HEROKU has chosen and work by web-hook events. Back- end is consisted of the database, PostgreSQL.



*Figure 3.2 System Architecture [4]*

3.4. Front-End Design and Use Case

For the front-end design, to make chat-bot running on the Facebook, first, the developer needs to register at the Facebook for developer page. After registration, I was able to make my own page for the chat-bot development. After setting the page, with a button called “Add a new App”, it becomes able to create app and you can connect Facebook messenger with the page. Also, on the set-up page, for the middle tier development, you can get page access token, and make a webhook.

I also included three Use case diagrams. First use case is initial and basic use case. As messenger is running on the Facebook page, user need to log in to the Facebook. Then the user needs to access chat-bot’s web page and send message. After user finishes finding any restaurant, he can log out from the Facebook page.



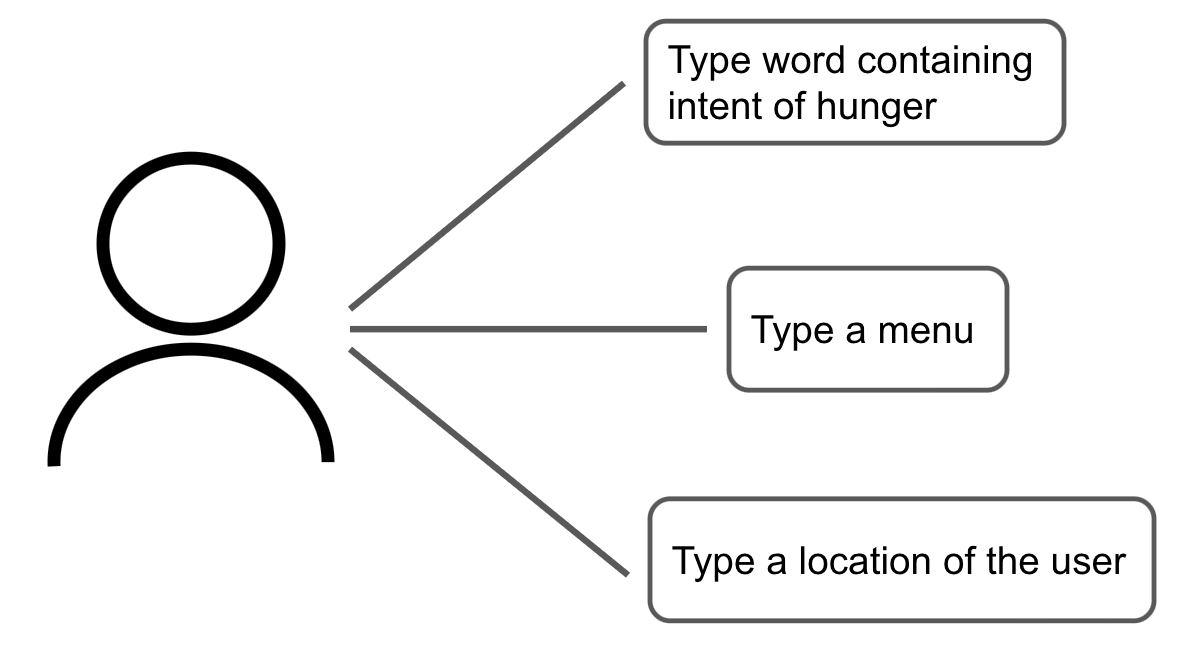
*Figure 3.3 Basic Use Case*

Next use case is about sending message for the first time. If user access to the web page for the first time, and tries to use the chat-bot, they need to press the <Get Started> button. This Button helps the first connection between client and the server. If the user once clicks the button, the button helps user with sending a message to bot automatically. After a few seconds, the user gets reply from the bot and this concludes the connection tries.



*Figure 3.4 <Get Started> Use Case*

Last use case is about using restaurant finding function. After the connection, user can have a conversation with chatbot. If the user writes down any word that contains intention of hunger, the chatbot activates its restaurant recommending function. To find adequate data, chat-bot asks users menu, and their location. Based on the menu and location, chat-bot searches information of 5 top rated restaurants and gives it to user.



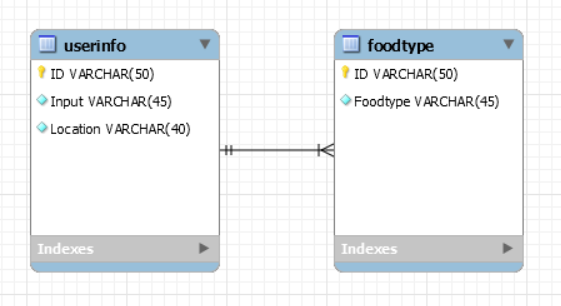
*Figure 3.5 Use Case during the conversation with chat-bot*

3.5. Middle-Tier Design

On the middle tier, throughout webhook and token that has been setup on the front-end design, the messenger is connected to the server. In this process, Node.js is chosen for the program language and ‘require’ function has used to include modules. This will be explained more specifically on the Development chapter. For the chat-bot platform, it is also included by ‘require’ function through its token and this included module helps the server to interact with the platform and the Facebook page.

3.6. Back-End Design (E-R diagram)

On the back-end tier, I designed two tables to keep the information of users’ input value and type of the food they searched. As you can see on the *figure 3.6*, the table ‘userinfo’ contains the input value of users’ containing their id, menu, and location. When user uses the chat-bot, as each of them has their unique id, I used the id to distinguish their respective menu and location value. For the ‘foodtype’ table, it contains the id of the user and the food type that they searched. As same as the ‘userinfo’ table, to distinguish each user I used user id attribute. ‘Foodtype’ attribute contains the type of crawled food information so that I can warn the user their unbalanced food habit.



*figure 3.6 E-R Diagram of the database*

3.7. Conclusions

In this chapter, I explained about the designs for front, middle, and back-end tier. To summarize it, on the front-end design, I explained about the ways to make the Facebook page and activate messenger on the page. On the middle tier design, I briefly explained about the way the server, bot platform, and the Facebook page interacts. Lastly on the back-end design, I explained the way that the database is connected and tables of users’ information and food type inside the database by showing E-R diagram.

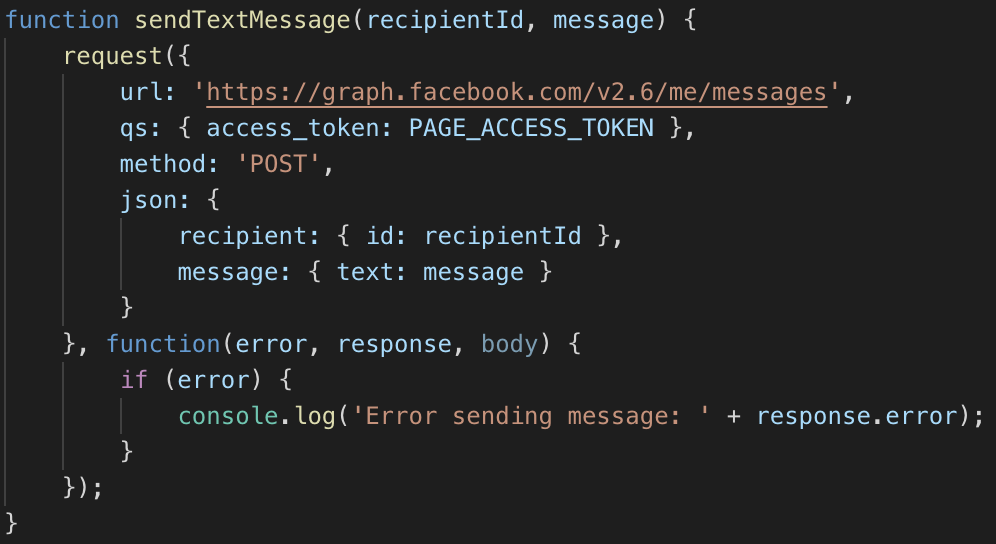
4. Development

4.1. Introduction of this chapter

In this chapter, description for the development phase of front, middle, and back-end tier will be explained. Some changes were happened when it is compared with the initial architecture. These changes from the initial architecture will also be explained in each chapter. In the front, back-end and middle-tier development chapter, each code of Node.js that used in each chapter will be explained. In the front-end development chapter, codes for getting postback, sending text message and images will be explained. In the middle-tier development, server connection, app reacting with server, crawling function will be explained. Finally, in back-end development chapter, how the queries are sent to the database and its result will be explained.

4.2. Front-End Development

In this chapter, I will explain my codes about sending and receiving messages and postback in messenger. First, there are two functions that I’m trying to explain. I named it ‘sendTextMessage’ and ‘sendImgMessage’. In Facebook, to use messenger in the developer’s own page, the developer needs a token called page access token. Using this token, the developer can access to his page and make the page to react on the developer’s request. You can see the ‘sendTextMessage’ function specifically on the *Figure 4.1*, the PAGE\_ACCESS\_TOKEN is variable containing the token of my page. To send message, I need two values. First is the id of the recipient and the second one is the message that the chat-bot is trying to send. I figured out the users’ id from the postback that I receive from the user. This will be explained later in the ‘receivedPostback’ function. There are many types of message that the user or either chat-bot can send, but in this function, I only took care of the text message as each message has unique format of JSON file. Below the request, you can see the function to check the error. By using ‘console.log’, I can check all the errors easily on my server.



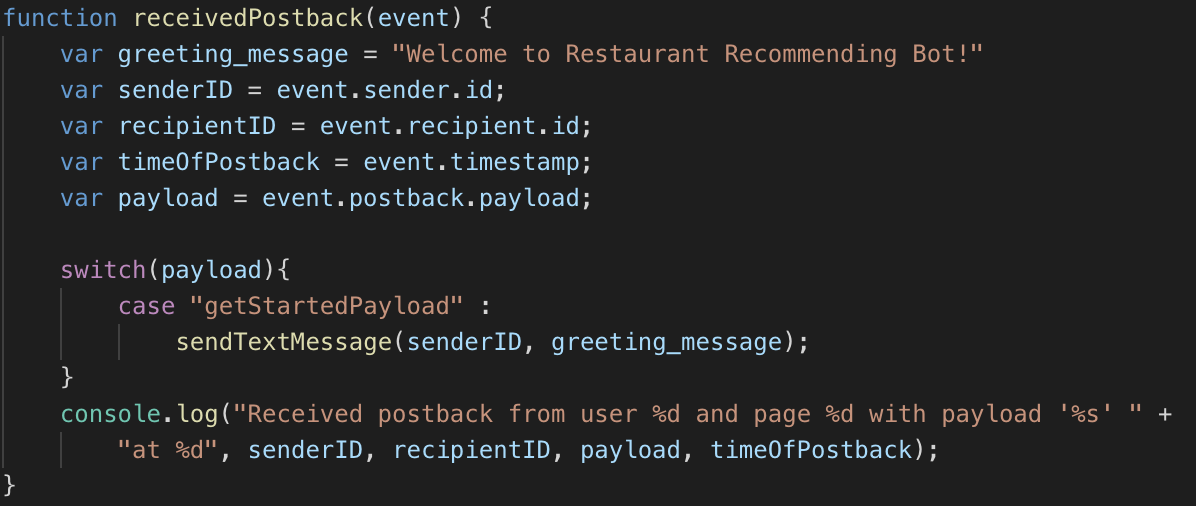
*Figure 4.1 Function – sendTextMessage*

Next function that I’m trying to explain is the ‘sendImgMessage’ function. This function is used to show result of the restaurants that has been crawled. Just like the sendTextMessage function, base URL, the token to access page, method, function of sending the error to the server are all same, but the difference is the JSON part. As you can see on the *figure 4.2*, it has restaurant’s name, it’s location, the URL for more specific information and the phone call button. Each button has different type. For the URL button, when the button is clicked, as it should show the web page, I chose URL payload attribute for the button. For the phone call button, I chose phone number payload to activate the function of the button.



*Figure 4.2 Function – sendImgMessage*

‘receivedPostback’ function was developed for ‘get started’ button. Unlike the simple text message, buttons are sent and received through postback method. Each postback can be set by unique payload. In this function, I set the payload for the ‘get started’ button as ‘getStartedPayload’.



*Figure 4.3 Function – receivedPostback*

4.3. Middle-Tier Development

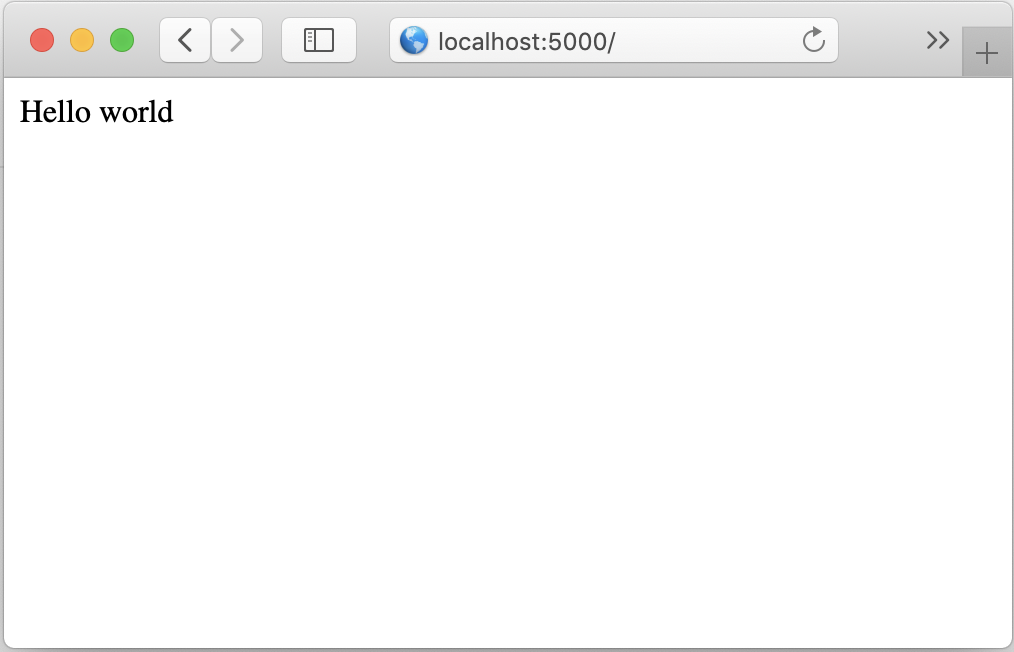
In this chapter, I divided chapter into two small chapters. First chapter will be about code explanation. In code explanation chapter, I’ll explain about the webhook function, app interlocking with server, and web crawler. The other chapter is about the platform of chat-bot. I’ll mainly explain about the platform and how I figure out the menu, location of user.

4.3.1. Code Explanation

First, setting port of server and connecting application to the server is important. On the *Figure 4.4*, you can see three functions of the app – set, get and listen. ‘set’ function is for setting port of server which will be used on starting server. Next, the ‘get’ function is used for app to response on server when the server sends any request. In the Figure 4.4, app responses to the server by sending ‘Hello world’ message. So, just like in Figure 4.5, as the port has been set as 5000, when you join the URL, https://localhost:5000, you can see the message ‘Hello world’ for the result of normal server connection. ‘listen’ function make the app to keep the connection with server by keep listening the server. By using ‘console.log’, when the command ‘npm start’ is typed, you can check the line ‘running on port 5000’ coming out. Other connection through other port except 5000, is responded by 404 Not Found error.



*Figure 4.4 app-server connection*



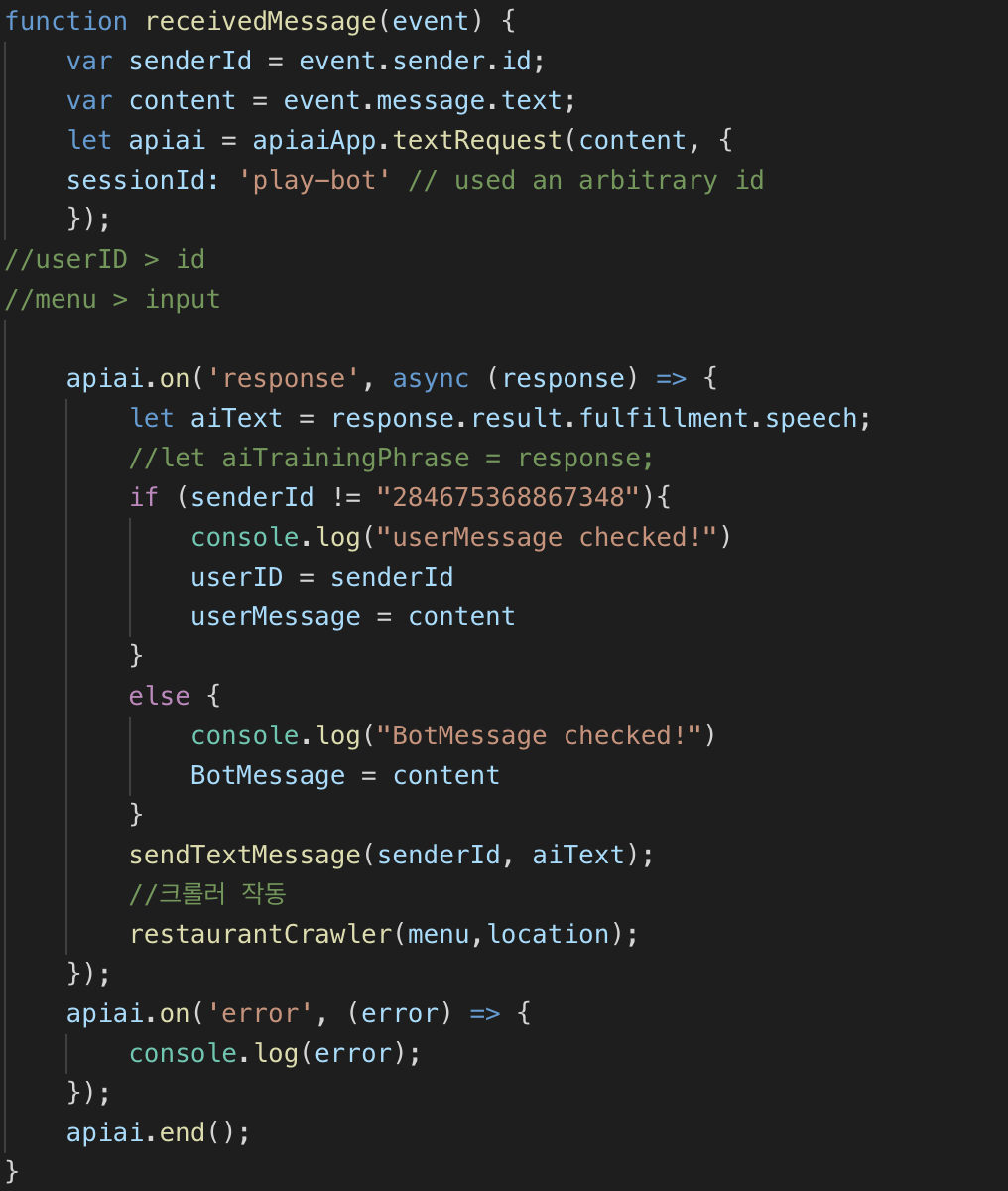
*Figure 4.5 result of app connection to server*

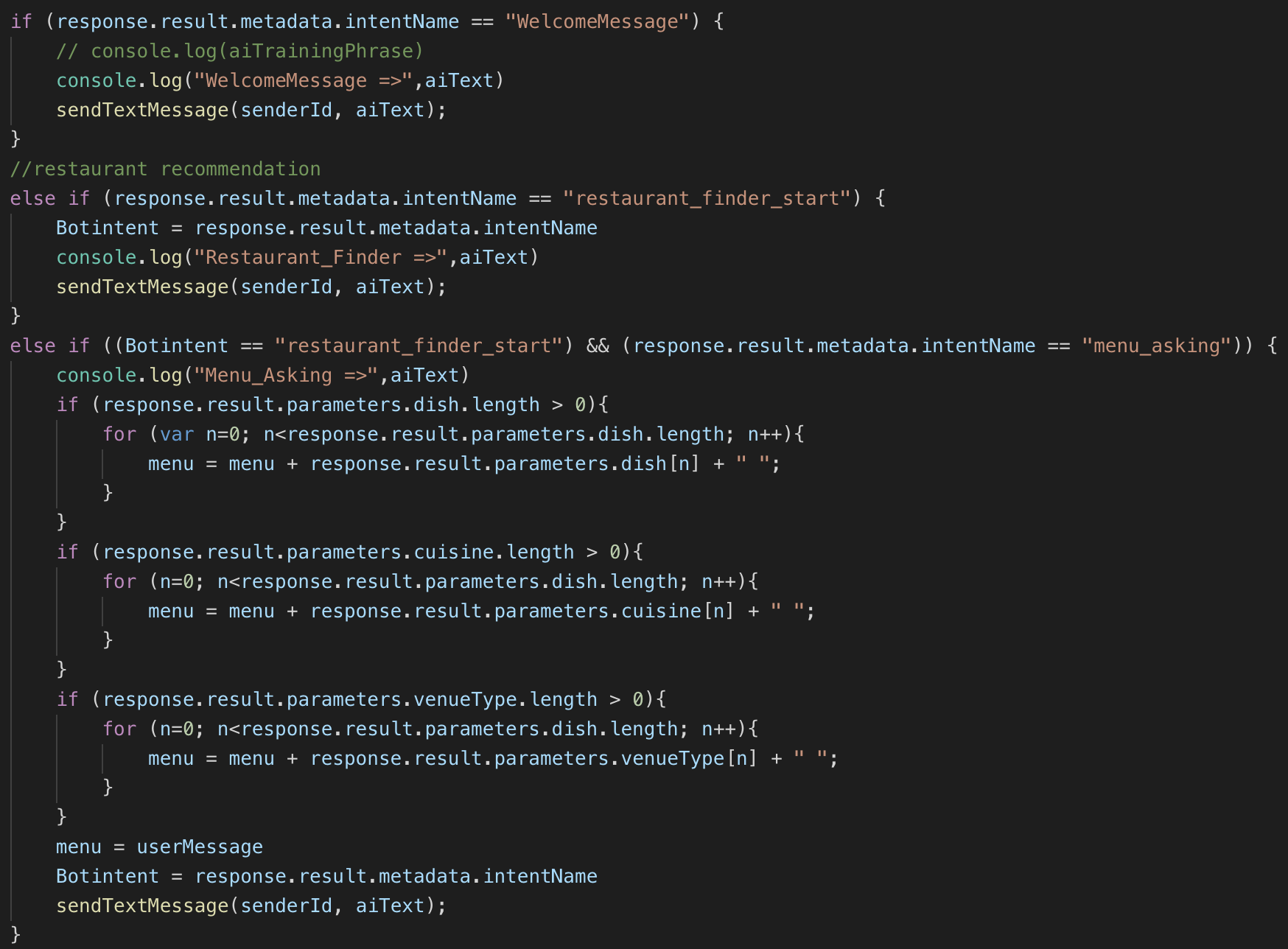
Also, there are function called ‘post’, this function is used to change certain value or status of server. In the case of *Figure 4.6*, the ‘post’ function is used when the route path is given as ‘/webhook’, so when the webhook is given, the app checks the type of messaging event. If the event is given as a form of message, then the receivedMessage function works. If it is postback, then receivedPostback function works. Also, when the post function works properly, it sends 200, HTTP status code, which means success of server dealing with the request.



*Figure 4.6 app.post function*

Next Function that I’m trying to explain is ‘receivedMessage’ function. As you can see on the *Figure 4.6*, when the message from the messagingEvent has given, the receivedMessage function works. In this function, apiai from the Dialogflow has been used to find out the intent of the message from the user and give adequate response of chat-bot. In this function, actions such as finding out menu from user’s input, finding out location from user’s input, giving welcoming message, asking users’ location or menu are done. I only added giving welcoming message, asking users’ menu part on the figure 4.7.

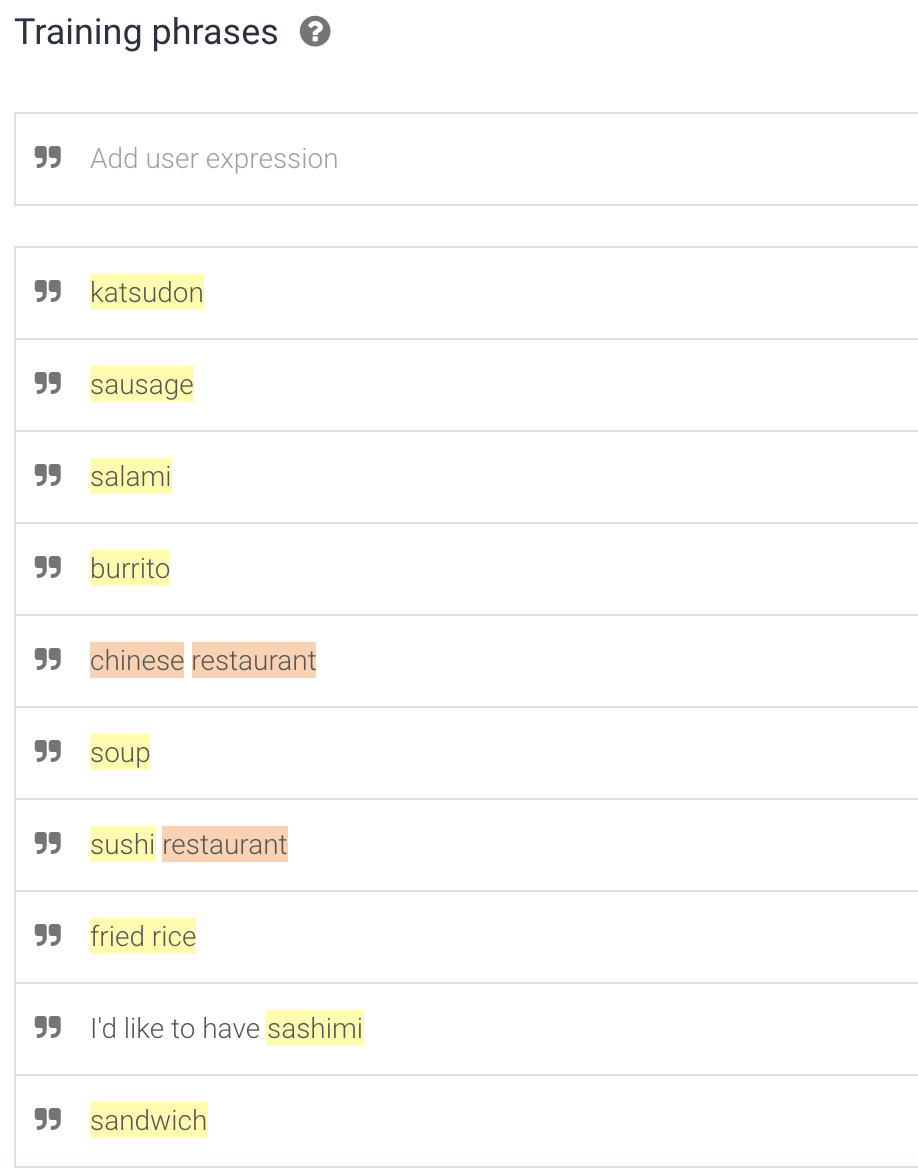




*Figure 4.7 receivedMessage function*

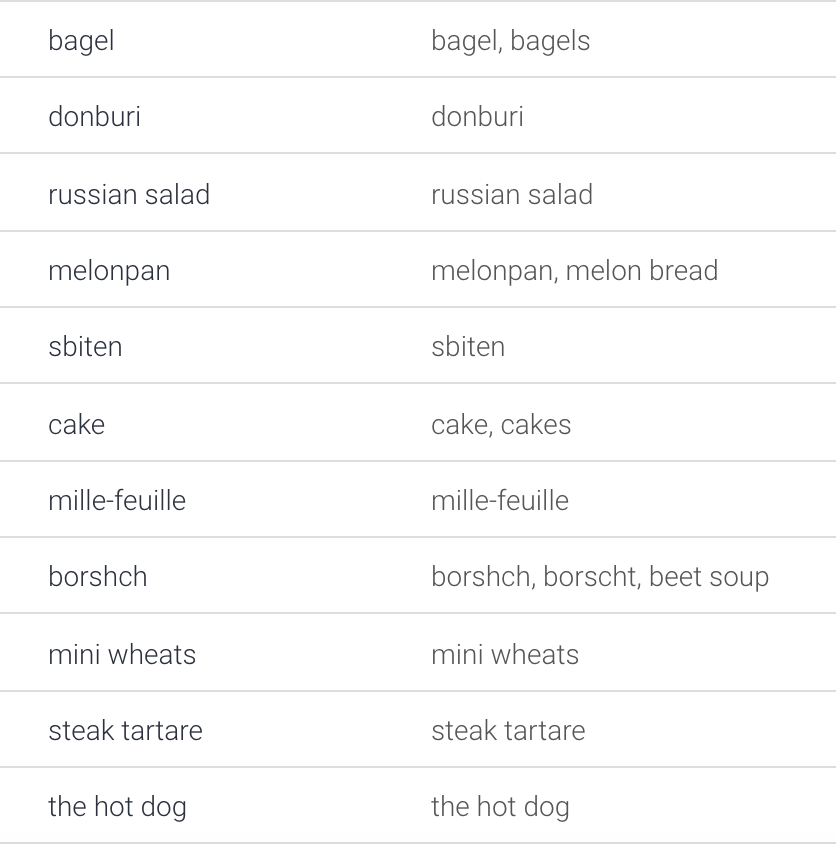
4.3.2 Chat-Bot Platform

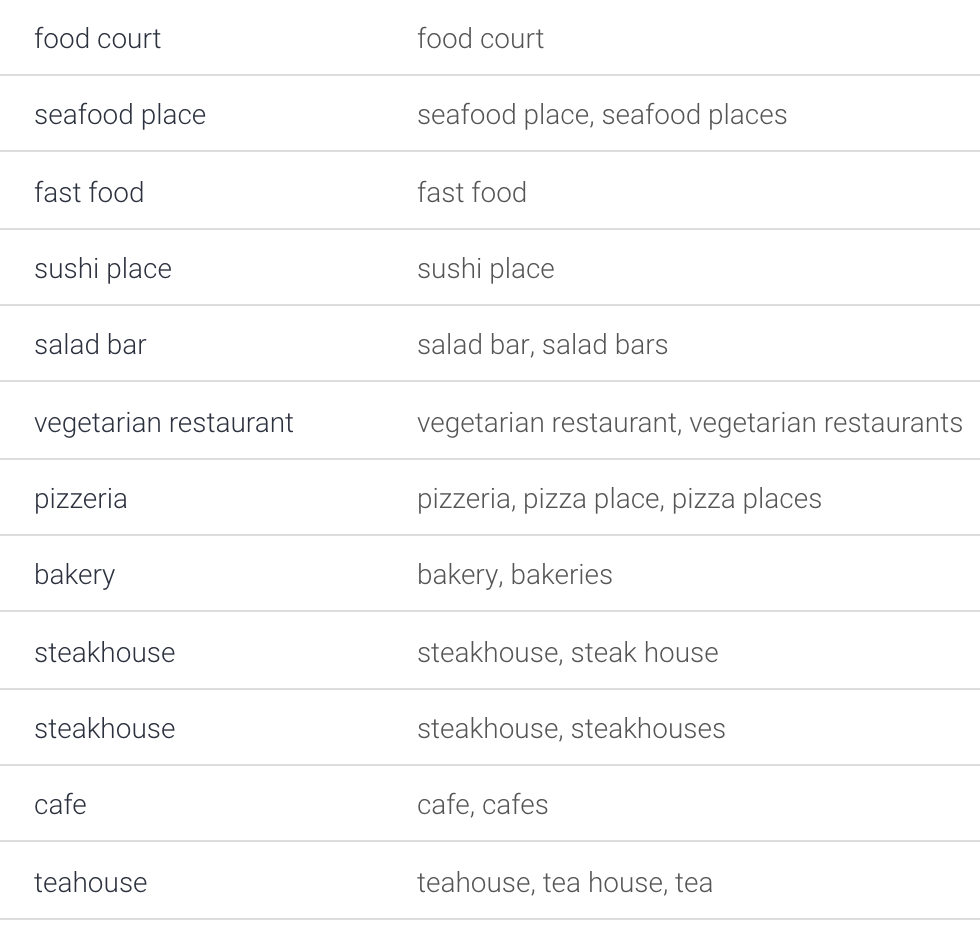
In this chapter, I’ll explain about the platform and how I’m getting menu or location from the user.



*Figure 4.8 Training Phrases*

When you see the *Figure 4.8,* You can see a portion of training phrases that I used to train my chat-bot. This portion is a training phrases I used to make chat-bot figure out the name of the food from users’ input. You also can see yellow or red color from the figure. This is called as entity. Entity is a set of data that has similar property. When you see the *Figure 4.9* below, you can see some properties of the entity. The first one is a set of dishes and second one is a set of venue types. By using these entities, I was able to figure out users’ input value more efficiently. For example, if a user inputs ‘I want a fried rice!’, then what I have to do is finding out the menu that the user wants from the whole sentence. Normally, it’s quite complicated to find out only a menu from the whole sentence. But, by using dish entity, which has fried rice in it, I can find out the menu and start crawling based on the menu in the entity.

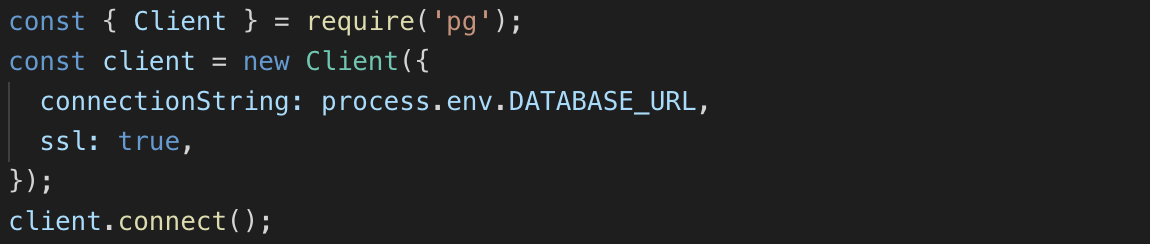




*Figure 4.9 dish, venue type entity.*

4.4. Back-End Development

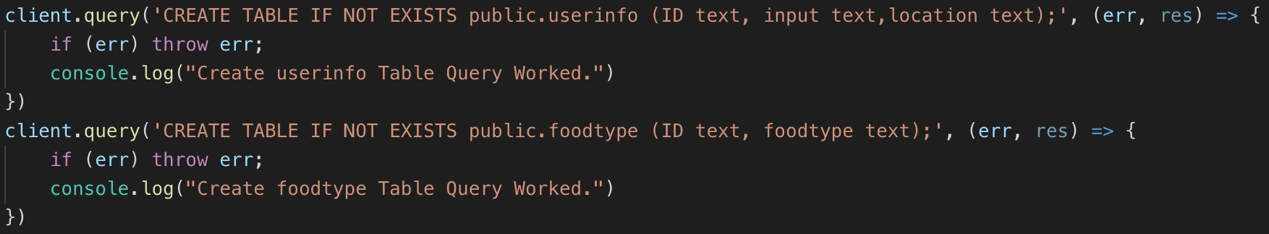
On the Back-End development chapter, I’ll explain about connecting the database, sending queries to database through Node.js. First will be the connecting part. In the case of database, as it is connected to my server, what I only have to do was getting the URL of the database that is connected with the server so that I can interlock the database. After that, I can use queries to input or output data into database.

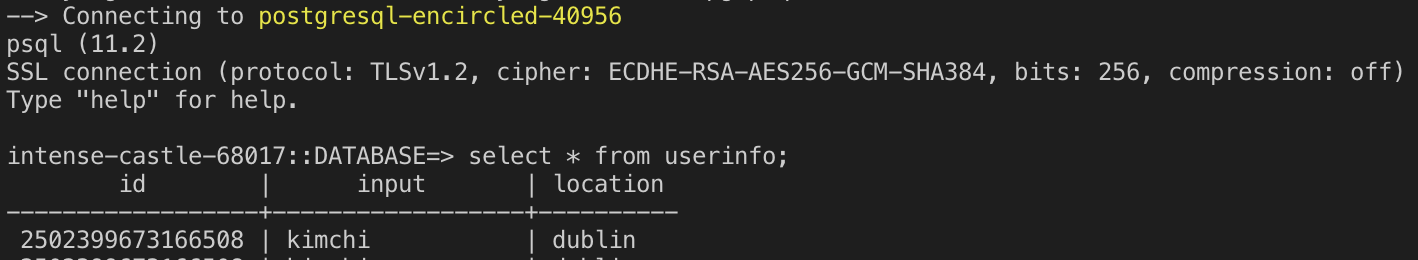


*Figure 4.8 Database Connection*

As you can see on *figure 4.8*, by using ‘require’ function, I called PostgreSQL module in Node.js and called an URL of the database. By using ‘connect’ function, I was able to connect database with server. After connecting the database, then, I can send queries.

By using ‘query’ function, I can send queries to my server and make database run. After the code is run, I can check the result on the console of my server and check whether data is correctly inserted or table has been created successfully.





*Figure 4.9 Sending queries, Result of table after the code running*

4.5. Conclusions

In this chapter, I explained some main functions that has been used in front, middle, and back-end tier. For the front-end chapter, I explained about the functions of sending and receiving messages and postback. For the middle tier chapter, I explained the server connection codes and what happens when the message is received. I also explained about the bot platform. For the back-end chapter, I explained about the connection of database and sending queries to it.

5. System Validation

* 1. Introduction of this chapter

In this chapter, testing method that has been deployed in this project, the result of the test, will be mainly explained. Both black box testing and white box testing were done to test the chat-bot messenger. Use case test were done for the black box testing, and for white box testing, function-based unit testing of MC/DC method has been used.

* 1. Testing Methodology

Black box testing for the functional features has been done in the type of use case test. Based on the users’ skill level to the application, functional and nonfunctional test cases have been planned. User who is moderately skilled and having basic knowledge with using computer and Facebook messenger is expected as main target of the user. So, I chose two tester who has basic knowledge of Facebook messenger. I asked two of my friends for the use test, so the participants are 25-year-old male with having an experience of using Facebook messenger.

White box testing has been done in the type of unit testing. The standard of the unit was each independent function of the code. Each condition statement has been tested in the coverage of MC/DC - modified condition/decision coverage. MC/DC Coverage method is the best way to check individual conditional statements’ independent aspect in the whole conditional statements and overall development. By using this method, I was able to expect the maximum test coverage for the application by using less test cases. Also, if one statement is changed, and others stay, it is easy to keep track on the changes of the application and react on the change sensitively.

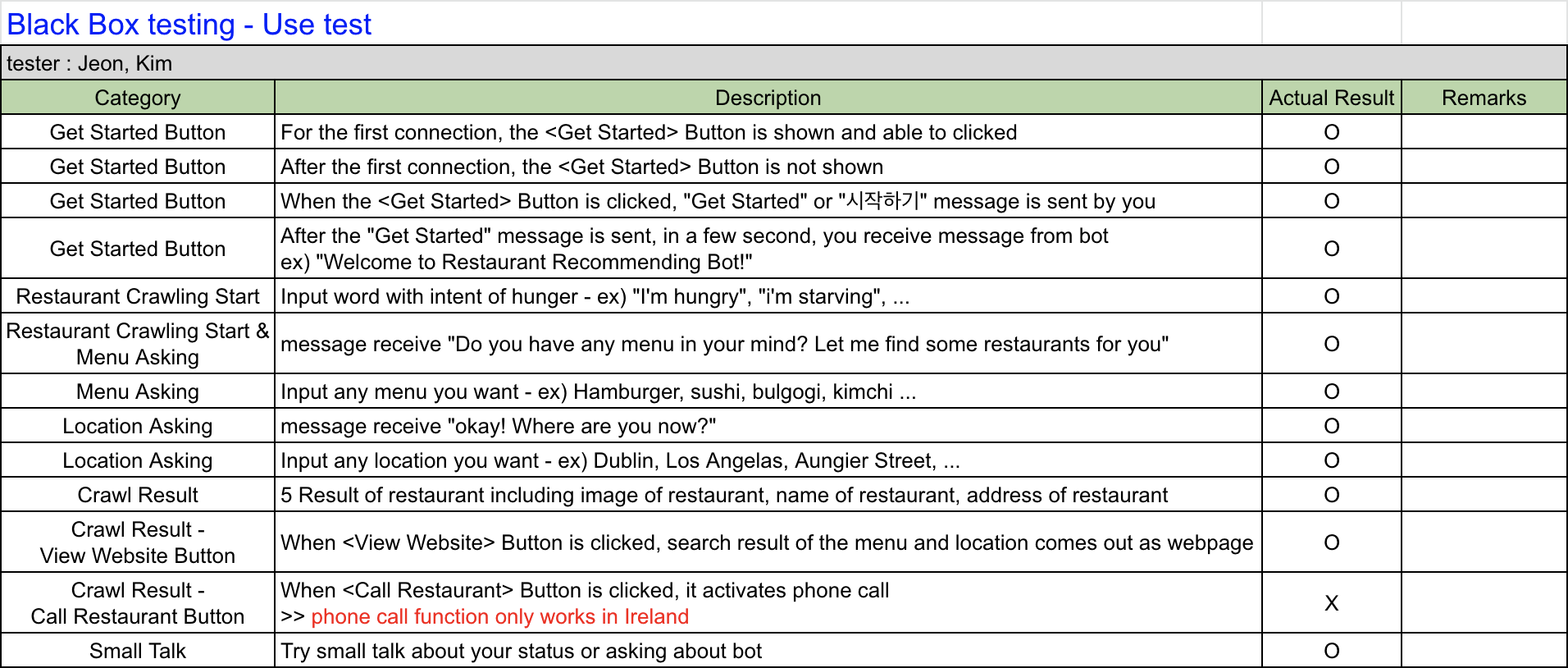
* 1. Testing Result

First, for the white box testing, result of test is saved as test case document. ‘IF’ conditions in the code were the main target of the white box testing. By making test case document such things like category, what the case condition is, expected result when the code run, and actual result was visualized so that it was easy to check the result of the test. In the category field, each statement is input. Case condition field contains the statements that will be executed. Expected result category means the expected result when the case condition is executed. This can be compared with the actual result category so that if the results are different, it can be written down on the ‘remark’ category.



*Figure 5.1 Result of White Box Testing – MC/DC Coverage*

Next, for the black box testing, as mentioned above, two testers participated in the black box testing. Main purpose of the black box testing, user test was to check the main function of Restaurant Finding. On *figure 5.2*, you can check the result of the final user test. Before the final test, there were some errors such as the crawler function did not work several times or when user uses apostrophe with menu or location, the data did not insert into database properly. With the use test, it was able to find some unexpected errors and fix it.



*Figure 5.2 Result of Black Box Testing – user test*

* 1. Conclusion`

In this chapter, I explained about the testing methodology for black box and white box testing. For the white box testing, MC/DC coverage method was used to test all the functions independently. For the black box testing, use case test has been selected to get the feedback of the application from different type of users. I also attached the result of each test by using test case documentation work.

6. Conclusions & Future work

* 1. Introduction of this chapter

In this chapter, the findings that I got during the development will be mainly discussed. Some improvements that is needed and some points that leave me a regret will also be discussed. Based on these findings and regrets, future direction of the application maintenance will also be mentioned.

* 1. Conclusions

From designing a system architecture to development and testing, the whole period of this project was really a good experience for me. I improved my proficiency of coding by using Node.js. Setting up the server, connecting the messenger application and database was also a new challenge, which I was able to learn more specifically about the web connection and application. Selecting the development method, testing method for the whole project was also a good chance for me to study more about those methods. Doing documentation of the whole test was also a good experience for me. By setting up the version of the test document and maintaining it made me easier to find errors.

On the other hand, there also were some points that I feel regret. Even though I tried hard to train my chat-bot, there are still some awkwardness with my chat-bot. It sometimes has hard time with understanding the small talk. Also, the phone call function of the application gives me some dissatisfaction. I failed to make the phone call button work for the restaurants all over the world. For now, it just works only in Ireland.

* 1. Future Work

For the future work of this project, the first work will be giving the chat-bot more training dataset. To reduce some awkwardness of the chat-bot, a sheer quantity of dataset is necessary. So, keep the current dataset and adding more data will be helpful.

Upgrading the phone call function also will be one of the future works. The problem that I met during the development was that each country has different country code. It was hard for me to have all the data about the restaurants’ location and its’ county code because that gives quite big burden to the server. If I have any chance of upgrading the server, I’ll try to finish the function.

References

* Heroku tutorial - <https://devcenter.heroku.com/articles/getting-started-with-nodejs>
* Dialogflow tutorial - <https://dialogflow.com/docs>
* Facebook for developer – <https://developers.facebook.com/docs>
* Reference of Image files

[1] - https://en.wikipedia.org/wiki/Heroku

[2] - https://www.heroku.com/dynos

[3] - https://medium.com/@MasterOfCodeGlobal/benefits-of-test-driven-development-64a24bbe743e

[4] - http://www.kwangsiklee.com/2018/01/%EA%B5%AC%EA%B8%80-ai-%ED%94%8C%EB%9E%AB%ED%8F%BC-dialogflow-%EB%88%88%EC%9C%BC%EB%A1%9C-%EB%94%B0%EB%9D%BC%ED%95%98%EB%A9%B0-%EB%B0%B0%EC%9A%B0%EA%B8%B0/