"Hello Bot"

The Smart Voice Assistant

BY

Thanapat Lappromrattana 62011272

Warakorn Mukdaprasert 62011288

Surasakul Charoenrujijin 62011256

A PROJECT SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF BACHELOR OF ENGINEERING IN ROBOTICS AND AI

KING MONGKUT'S INSTITUTE OF TECHNOLOGY LADKRABANG

ACADEMIC YEAR 2022

FACULTY OF ENGINEERING

KING MONGKUT'S INSTITUTE OF TECHNOLOGY LADKRABANG

PROJECT CERTIFICATE

Project Title	"Hello Bot" The Smart Voice Assistant
Student Name	Mr. Thanapat Lappromrattana Student ID. 62011272 Mr. Warakorn Mukdaprasert Student ID. 62011288 Mr. Surasakul Charoenrujijin Student ID. 62011256
Degree	Bachelor of Engineering in Robotics and AI
Project Advisor	Signed: ______________ (Prof.Dr.Pitikhate Sooraksa)
Project Co-Advisor	Signed: ______________ ( )

Project Title	"Hello Bot" The Smart Voice Assistant
Student Name	Mr.Thanapat Lappromrattana
	Mr. Warakorn Mukdaprasert
	Mr. Surasakul Charoenrujijin
Degree	Bachelor of Engineering in Robotics and AI
Project Advisor	Prof.Dr.Pitikhate Sooraksa International Program, Ph.D.
Academic Years	2022

ABSTRACT

This thesis presents "Hello Bot" The Smart Voice Assistant, which aims to apply the knowledge gained by its authors to build an easy-to-use system that can improve the quality of life for elderly people. Factual data shows the population of younger generations is insufficient to meet the needs of an aging society. In addition, many elderly people struggle with using technology such as Youtube, Gmail, Video Call, and Google. They may also have difficulty remembering things and experience physical limitations that prevent them from participating in social activities and can lead to loneliness. The "Hello Bot" project has been created as a result to provide a voice assistant that gives users easy access to technology such as Youtube, Google, Gmail, Pill Reminder, Take Note, and Video Call. Furthermore, the project is equipped with the ability to converse about anything with the user through a unique and friendly robot face. This is made possible by the combination of various technologies, including Speech Recognition, NLP, Text to Speech, Animation, SMTP, Python, WebRTC, and Chat GPT. The project was evaluated through a number of test runs for each function, which resulted in a promising 90% success rate with only minor errors due to voice interference.

ACKNOWLEDGEMENTS

In this moment, we would like to take an opportunity to give our sincere thanks to all the professors we took classes for the past 4 years. Thank you KMITL for all the lessons, experience, teachers, staff, professional professors, incredible connections, and good environment. Without them, we wouldn't be able to gained enough knowledges and inspiration to developed this project. We would like to exclusively give our thank to Prof.Dr.Pitikhate Sooraksa who is our advisor for this project. With knowledges we gained over the past four years, what we lack was the right direction. Prof.Dr.Pitikhate played that role perfectly, and with sincere care that we can feel. The evidence is the fact that Prof.Dr.Pitikhate and our team have had meeting every 2 weeks over this semester for us to update and improve the project. At last, we would like to thank Ms. Pravee Chanachavarin and her team for providing "Jiraff" robot for our project real use case.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS/ABBREVIATIONS

Symbols/Abbreviations	Terms
CIE	Computer Innovation Engineering
SIIE	School of International Interdisciplinary Engineering Programs
RAI	Robotics and AI
pyttsx3	Python text to speech module
NLP	Natural Language Processing
Senior	Age 60+

CHAPTER 1
INTRODUCTION

Thailand and many countries around the world are turning into aging society is never before deniable. In addition, the works for adults, and school curriculum for kids cause the lack of interaction between older and younger generations. Not to mention the isolation that caused from COVID-19 pandemic situation that have tremendously affected people of all ages, elderly people included. Such situation created loneliness and some need support. New generations might be able to use the power of technology to help them. For example, video call, Youtube, and social medias just to name a few. However, most of the elderly people can't have such privilege due to numerous reasons. For instance, they don't know how to use, or the lack of fund. The chance of loneliness to happens in elderly people is therefore higher compare to newer generation. In addition, loneliness can affects their health. In conclusion, the aim of our project is to construct a prototype voice assistant system that can provide elderly people with support in using social media. Furthermore, providing them with a way to contact, and stay connected with their social circle. Last but not least, be a companion that stay, chat, and be a friend with elderly people.

1.1 Problem Statement and Background of the Study.

1.1.1 Thailand is becoming an aging society.

Reference to United Nation department of Economic and social affair, it states that Thailand is considered one of the world's rapidly age society and expect to double between 2015 – 2050. (Prachuabmoh, V., n.d.)

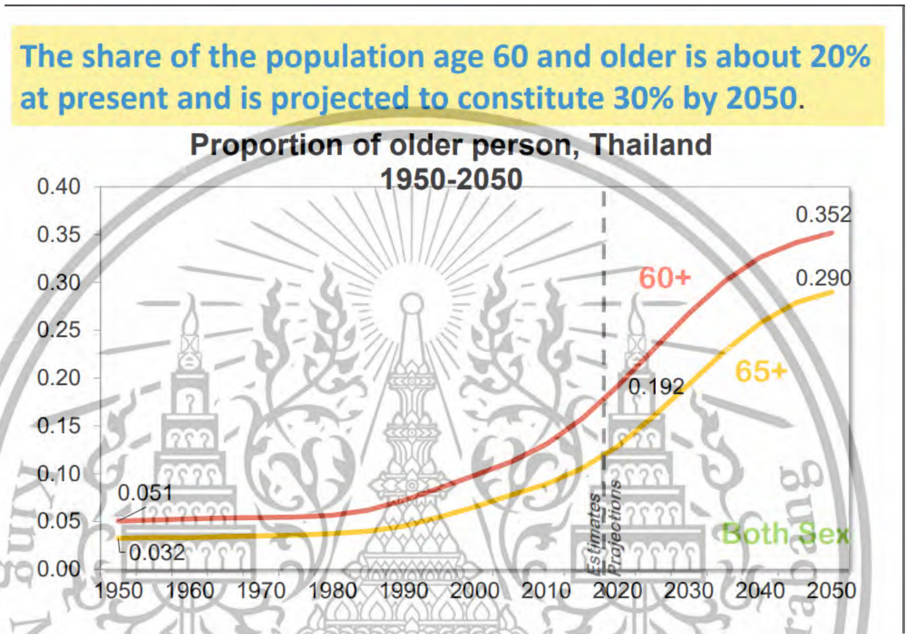


Fig 1. Graph proportion of older person in Thailand.

(Prachuabmoh, V., n.d.)

1.1.2 The world is becoming an aging society.

In addition, United Nation department of Economic and social affair has published that not only Thailand, but many more countries are facing the same problem. For instance, Japan, China, and Korea just to name a few. (Prachuabmoh, V., n.d.)

Proportion of Population age 60+ in Asia

	2015	%	2035	%
1	Japan	32.8	Japan	39.9
2	Hong Kong, China	21.8	Korea	35.5
3	Georgia	20.3	Hong Kong, China	35.3
4	Chinese Taipei	18.6	Singapore	34.1
5	Korea	18.4	Chinese Taipei	33.1
6	Singapore	17.9	Thailand	30.2
7	Cyprus	17.7	Macao	28.8
8	Israel	15.8	China	28.5
9	Arminian	15.8	Cyprus	26
10	Thailand	15.6	Georgia	25.9

Source: *United Nations, Department of Economic and Social Affairs, Population Division (2017). World Population Prospects: The 2017 Revision*

Fig 2. Proportion of population age 60+ in Asia.

(United Nation, 2017)

1.1.3 Older generations struggle.

As many as 77% of seniors struggle to use new technology and required an assistant to learned. Furthermore, as many as 56% of seniors who already up online also need assistant to connect with friends and family on social media. Last but not least, most technology are not designed for seniors. (No Isolation, n.d.)

1.2 Rationale of the Study.

There are many reasons why technology is useful to a person, but that is exactly the case when it applies to seniors.

1) Senior has lower income and reduced mobility, and that makes meeting in person is challenging.
2) New generation experience and senior's experience are not the same which cause the difficulty in catching up with new technology

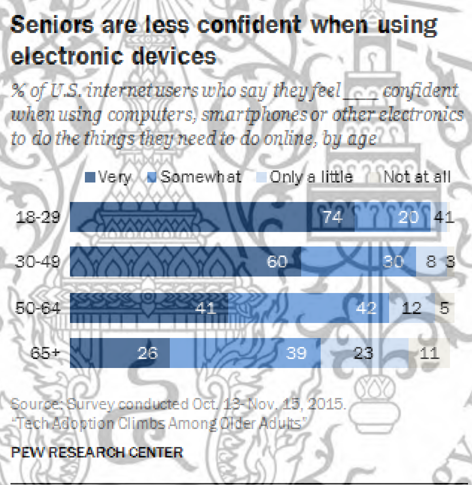


Fig 3. Seniors are less confident when using electronic devices.
(Pew Research Center, 2017)

1.3 Purpose of the Study

- To improve the quality of life for seniors and people in need.
- To apply knowledge of Robotics and AI in real life application.
- To create a voice assistant that easy to use, but provides effective results.

1.4 Scope of the Study

- Use voiced to search Youtube, and open a specific video on Youtube.
- Use voice to search Google.
- Use voice to use Gmail.
- Use voice to set an alarm. For instance, remind me to take a medicine in five minutes.

- Use voice to take a note.
- Use voice to chat with the robot
- Use voice to make a video call

1.5 Method of the Study

Coding Language : Python

Code editor : Visual Studio Code

Animation : Adobe After Effect

Table I: Required module names and descriptions.

Table I.

Modules name	**Description**
Speech recognition	Convert speech to text.
Pyttsx3	Text to speech, convert text to speech.
Time	Use timer.
Pywhatkit	Play a video Youtube.
Webbrowser	Open Youtube & Google. Search Youtube & Google.
Pygame	Create UI, and robot face animation.
Threading	Multitasking. (For reminder function)
Email.message	Send Gmail.
Random	Random the speech feedback to create a variety of outcome.
Datetime	To save the note with specific date and time. To give feedback result that differ, depending on date and time.
Openai	To chat with the robot

Table II: Required hardware names and descriptions.

Table II

Name	Description
Computer	To code & To run code.
Microphone	To receive speech / command.
Speaker	For output speech feedback.
Touch Screen Monitor	Display Robot Face

1.6 Outline of the Thesis

The rest of this report is organized as follows:

Chapter 2 reviews the state-of-the-art of NLP.

Chapter 3 describes the design methodology.

Chapter 4 demonstrates how the voice assistant functions

Chapter 5 closes the report, reviewing the work undertaken and draws conclusions about key parts of the work that was undertaken. Finally, future work is discussed with particular focus on improving the project.

CHAPTER 2
Literature Review

This chapter discusses the state-of-the-art of NLP and any knowledges considered during the analysis and design phase of this project. The investigation served to understand how robot understand human language purposes: firstly, to identify the concept of NLP (Section 2.1); and secondly, to showed the related works of NLP that is the predecessor of this thesis. (Section 2.2). Thirdly, explore the established steps and processes for a computer to understand human language in NLP (Section 2.3). Fourth, explore how to implement animation with code application. (Section 2.4). Fifth, how to send email via code. (Section 2.5). Then, section 2.6 -2.10 are WebRTC, PeerJS, HTTP server, Speech Recognition ,and Text to Speech respectively. Finally, section 2.11 summarizes the chapter.

2.1 What is NLP?

NLP stands for Natural Language Processing. It is a technique which combine linguistics and computer science to create a decipher language structure that can comprehend the human languages, and respond back appropriately either by action or word. In order to develop a NLP related project, an individual need data that can represents the human behavior. Those data need to be used to teach the computer to understand the command and give out the result that can be configured, depending on the coder. The idea is similar to teaching a baby to understand human language. In this situation, however, the baby is the computer. NLP, in conclusion, can replace human in many applications. If the application is a repetitive step that required to follow in pattern. For example, call center, smart assistant just to name a few, and we'll explore further in the next session. Therefore, NLP is a technology that can help in reducing cost, and improve time efficiency as well. (Brownlee, J, 2019)

2.2 Related Works

NLP is useful in many real-life applications. For example, Chat bot, smart assistant, call center, search engine, predictive text, grammar checker, and plagiarism checker. They all have something in common. They all are computing processes related to text.

However, in this thesis, the focus will be smart voice assistant that can chat with human. Therefore, in this section will list all related works. Such works capabilities must include but not limited to or included only some of the scope of study listed in the chapter one. The work should be able to let the users use voice to command Youtube in some way. Secondly, should be able to use voice to use Google function. Thirdly, can use voice to use Gmail function. Fourth, can use voice to set the time reminder for specific need. Fifth, can take a note. Sixth, to make a video call, Seventh, to voice chat like a chatbot. Eight, to has a robot face. And the most important thing is such technology must be designed for assisting senior.

1) **Dinsaw Robot**

Fig 4 Dinsaw Robot
(CT Asia Robotics, n.d.)

Descriptions:

Dinsaw is a robot designed for providing seniors with entertainments and connect them with their relatives. Most of the process is easy. However, it had to trade-off with the limited number of contracts up to only five. It comes with cute face which emphasized the gentleness, and cuteness. The price for this robot built by Thai people is 85,000 Baht. (CT Asia Robotics, n.d.)

Functions:

1) Can Video call in app. However, only limited to 5 saved contracts.

(Can’t use voice command, the way to call is touch the face of the one he/she wants to call)

2) Open music from its memory. (Can use voice command)
3) Simple guessing game pre-installed. (Can’t use voice command)
4) Use heat sensor to detect when the senior is missing or fall. Then, send alarm to the user connected to the app.
5) Pill reminder set in app
6) Has face.

2) Kaitomm Robot

Fig 5 Kaitomm Robot
(Obodroid, n.d.)

Descriptions

A robot developed by Obodroid and MQDC.

Functions

1) Connect with health care device.
2) Command Light on-off.
3) Pill reminder system.
4) Has face.
5) No entertainment functions.

6) VDO call

Table III Compare Dinsaw & Kaitom functions with the scope of this thesis functions

Table III.

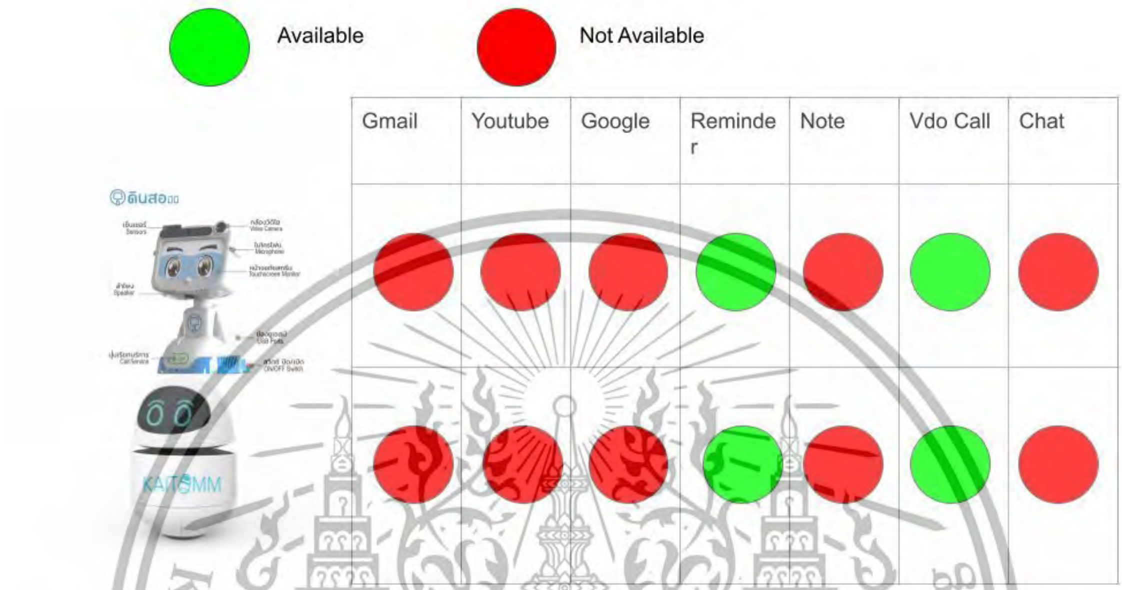
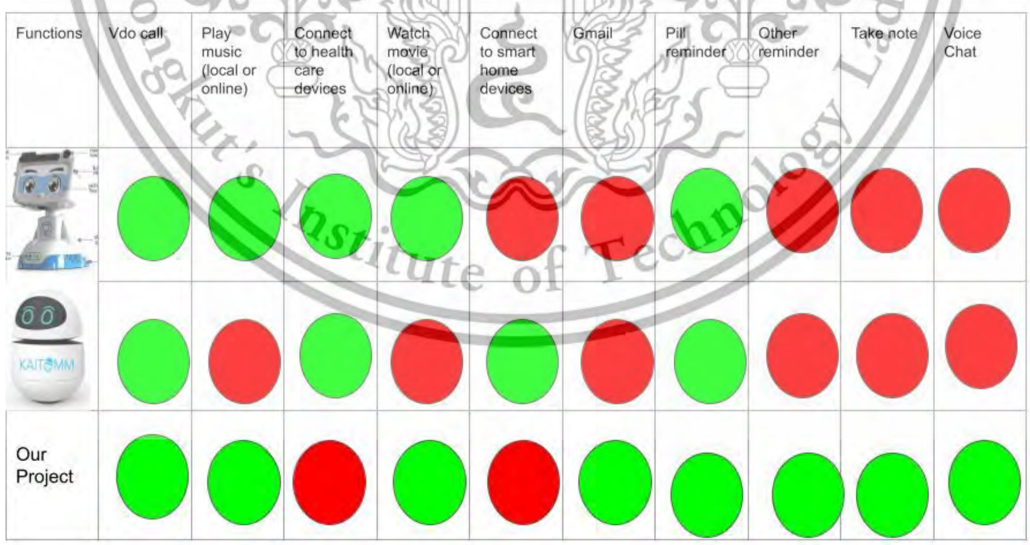


Table IV Compare all function of Dinsaw, Kaitom, and this thesis.

Table IV.



2.3 Frameworks

In general, NLP framworks can be divided into seven elements. (Lutkevich, B., 2021).

1) Segmentation
2) Tokenizing
3) Stop word
4) Stemming
5) Lemmatization
6) Speech tagging
7) Name entity tagging

These elements will be further discussed down below.

1) Segmentation: This process is to break the entire document down into its constituent sentences.

For example:

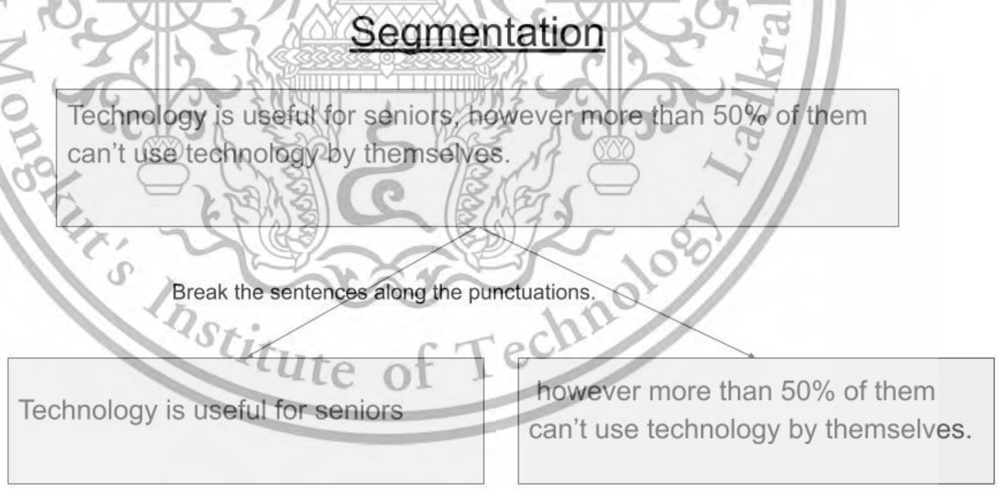


Fig 6. Segmentation.

2) Tokenizing: break down a sentence into words.

For example:

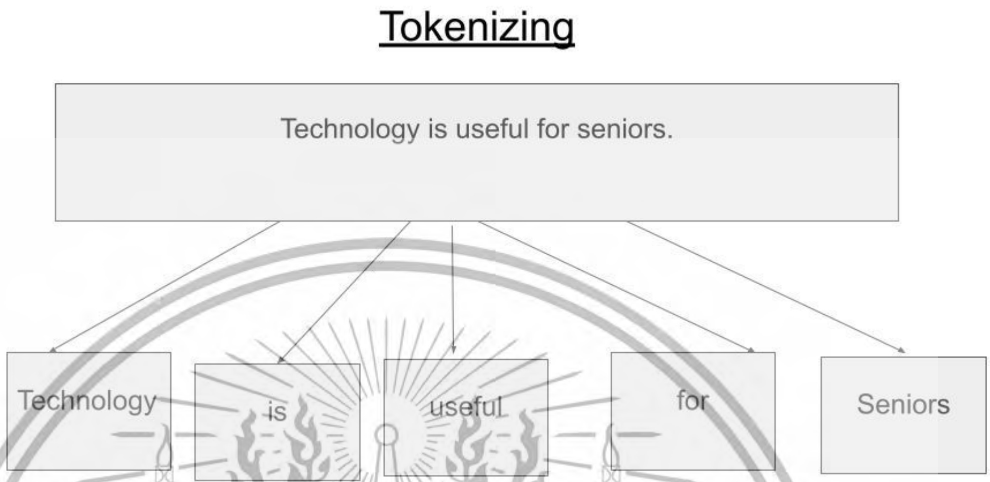


Fig 7. Tokenizing

3) Stop words: Get rid of the word that is just there to make our statement sound more cohesive.

For example:

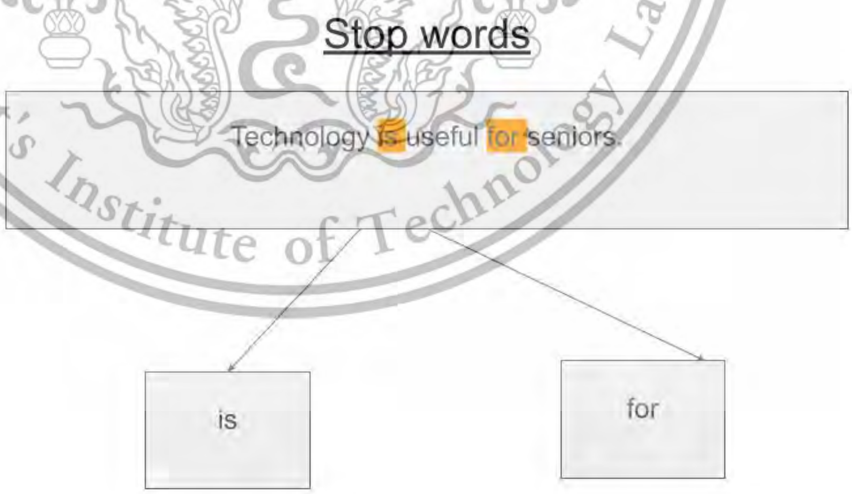


Fig 8. Stop Words

4) Stemming: Teach the idea of prefixes and suffixes. Although writing is slightly differed with the addition of prefixes and suffixes, they all stemmed from the same origin.
   For example:

Fig 9. Stemming

5) Lemmatization: Group the words into the same category.
   For example:

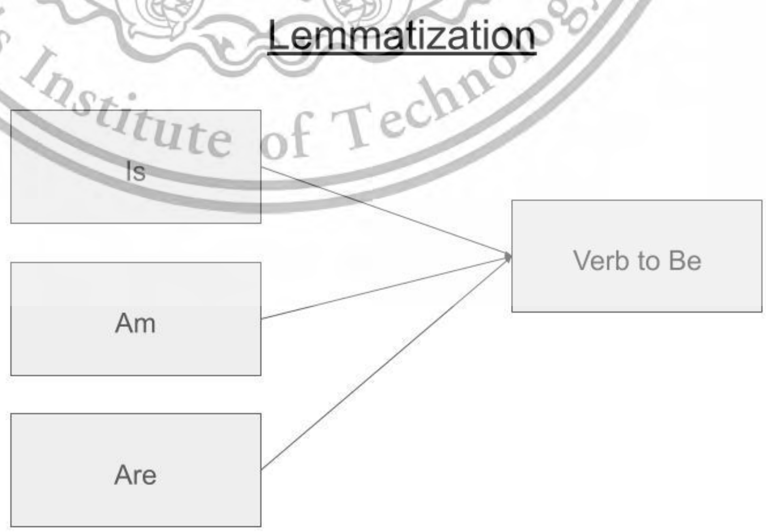


Fig 10. Lemmatization

6) Speech tagging: Teach the concept of Noun, Verb, and article to the machine.
For example:

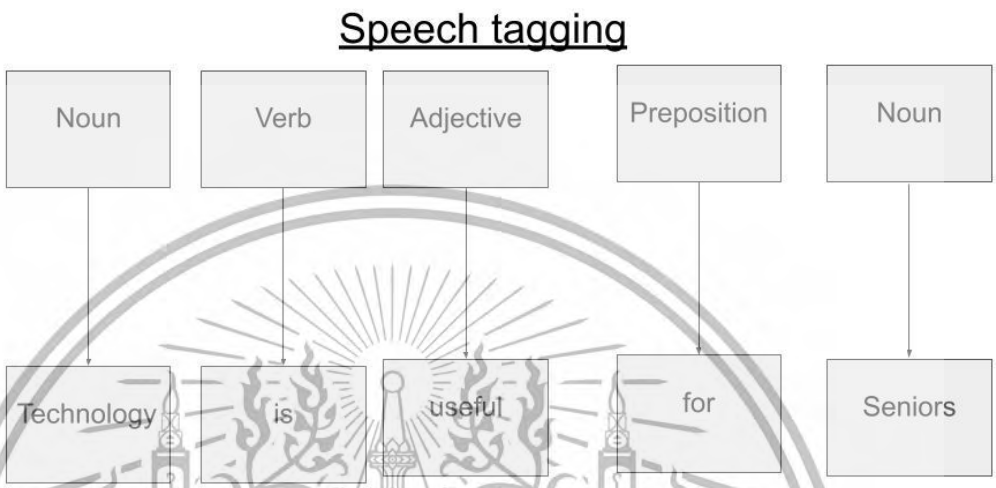


Fig 11. Speech Tagging

7) Named entity tagging: A process to tag identity needed for the project.
For example:

Fig 12. Named Entity Tagging

2.4 Animation

Fig 13. Animation's Example

(Koro, P., 2019)

Animation is an illusion of movement. It's technically a technique which consist of multiple solid pictures arrange in a way that it seems like it's moving when those pictures change pretty fast in the same position in a certain time. Therefore, the idea of FPS that stands for Frame per second is crucial. Frame per second defines how many frames changed in one second. For example, it can be one frame per second up to twenty frames per second. The number of frames depending on the applications. However, it's generally believed that the greater number of frames per second, the smoother the animation will be.

2.4.1 Concept of Frame Per Second

The picture below shows nine pictures of a runner in different posture. If one second consists of these nine pictures changing, meaning 9 fps. Another example is shown below.

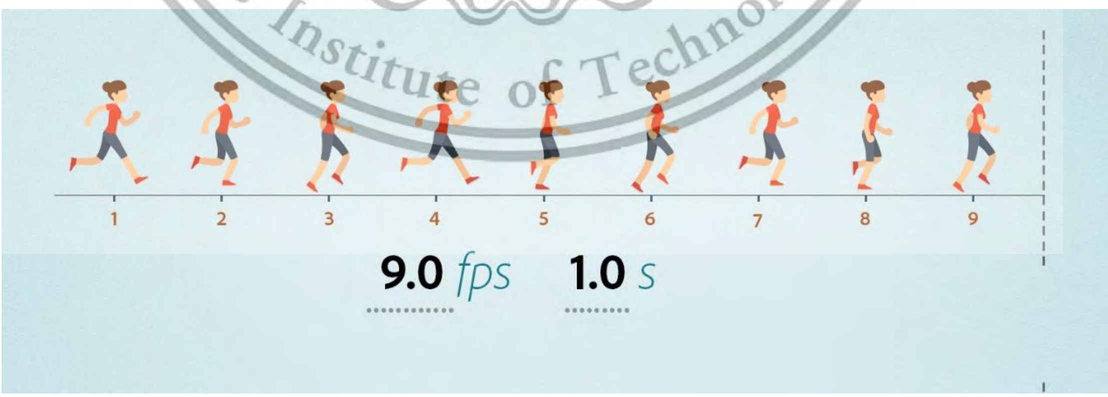


Fig 14. Nine Frames Per Second

(Adobe, 2021)

Now the picture only consists of six pictures in one second. Therefore, the number of fps is 6 fps.

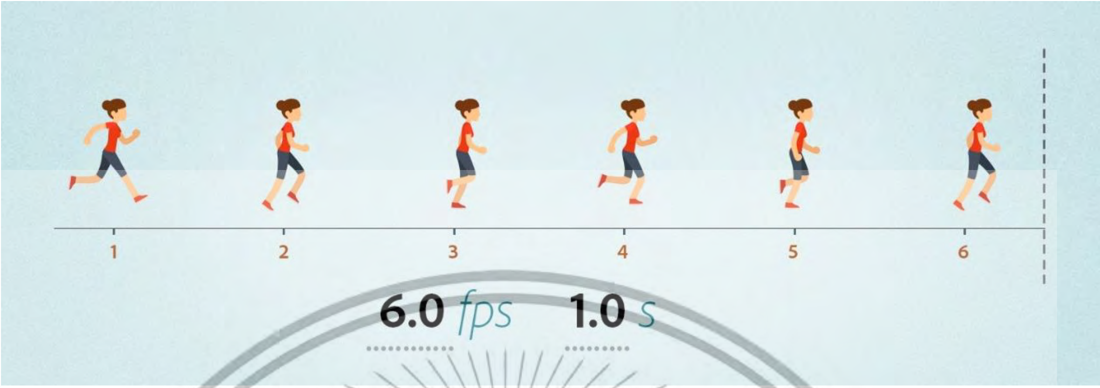


Fig 15. Six Frames Per Second
(Adobe, 2021)

2.5 Simple Mail Transfer Protocol.

Today sending mail had became basic communications between people, and the technique used to send email is called Simple Mail Transfer Protocol or in short SMTP. SMTP has server acts as a middle man between two users. For example, Mr.A is a sender, and Mr. B is a receiver. Mr. A need to send mail to server then server will send mail to Mr. B in Mr.A stead.

SMTP also has SMTP port which play a major role in operation. Port is what define the services, or type of the protocol. For example, 80 or 443 for web browsing, 21 or 20 for FTP, and 25, 487 or 2525 for SMTP. However, nowadays people mostly approved of using port 587 for SMTP. The reason being that some of other port are block from many cloud services, and in some case unsecure. Therefore, the port being used in this thesis is 587. (Philip, A., 2021)

2.6 WebRTC

WebRTC, which stands for Web Real-Time Communications, is an open-source project that enables real-time voice, text, and video communication capabilities between web browsers and devices.

2.7 PeerJS

PeerJS simplifies WebRTC peer-to-peer data, video, and audio calls. PeerJS wraps the browser's WebRTC implementation to provide a complete, configurable, and easy-to-use peer-to-peer connection API. Equipped with nothing but an ID, a peer can create a P2P data or media stream connection to a remote peer.

2.8 HTTP server

Software that implements the server part of the HTTP protocol. It waits for client requests and replies each request with requested information.

2.9 Speech Recognition

Speech Recognition convert human voice into a digital data that a computer could compute. It's a technology that required a combination of hardware (microphone) and software (pattern recognition).

2.10 Text to Speech

A technology that converts text into spoken word. It uses algorithm to identify the phoneme of each word to generate a sound that mimic human speech.

2.11 Chapter Summary

In Chapter 1, an introduction to the project, including the motivation and goal of the project were mentioned.

In this chapter the state-of-the-art was categorized into understand the concept of NLP, understand all the established steps and processes of NLP that makes computer recognized human language. Furthermore, the fundamental of Animation, SMTP for sending email, and all required components for making a WebRTC communication are also discussed. In addition, text to speech use for making computer understand human voice, and text to speech a technology for making a human-like speech also mentioned.

The next chapter presents the implemented design of NLP used in this "Hello Bot" voice assistant project. It includes plan and system methodology. The project development plan, and the designed of the voice system for both receive and respond. The chapter will include box diagram, explanation, some examples, and will be ended with chapter summary.

CHAPTER 3
METHODOLOGY

3.1 Introduction

In Chapter 2, the theory of NLP, Animation, SMTP, WebRTC, Speech Recognition, and Text to speech were explained.

This chapter, on the other hand, describes both system and plan methodology of this paper. First, the chapter describes the project's plan and system design methodology (section 3.2), and interesting problems (section 3.3). A proposed solution is then discussed (section 3.4), followed by a summary of what we discussed in this chapter.

3.2 Design Methodology

3.2.1 Plan Methodology

The design and development of this paper is based on agile design. It's a type of methodology in which required minimal planning, but allowed software to be released in iteration. The primary benefit of such is that the bug and any problems can be fixed early on for the developer while user, on the other hand, can see the true potential of the project. Furthermore, the feedback of users received by developer can be put to good use very easily. The concept of agile is that it breaks the plan into multiple small steps. First, Select the project. Second, initiate the project. Then, iteration. Fourth, release. Fifth, go back to step three periodically to fix and add according to user requirement. Sixth, production to keep maintain the system. Finally, Retired. (Knowledge Jump, 2012)

The steps described can be seen visually as shown below.

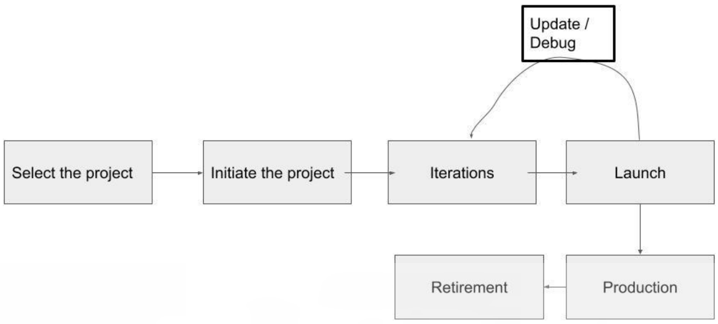


Fig 16. Plan Methodology

3.2.2 System Methodology

Voice assistant is designed based on NLP. Furthermore, the purpose of NLP is to mimic human. In this situation, the goal is to create a personal assistant that on standby 24/7. Therefore, the system must be able to understand the human speech and identify the condition. If the word is command, then execute that specific command as required per code. Therefore, the whole system can be summarized into four simple steps. Firstly, Speech recognition to convert human speech to text. Secondly, compare the text with conditions. For instance, if the word "open youtube" included in the speech, open Youtube. Thirdly, follow by the condition. At last, the fourth step is response back using text to speech. The third and fourth steps can be switch depending on the code. The block diagram of the system is as shown below.

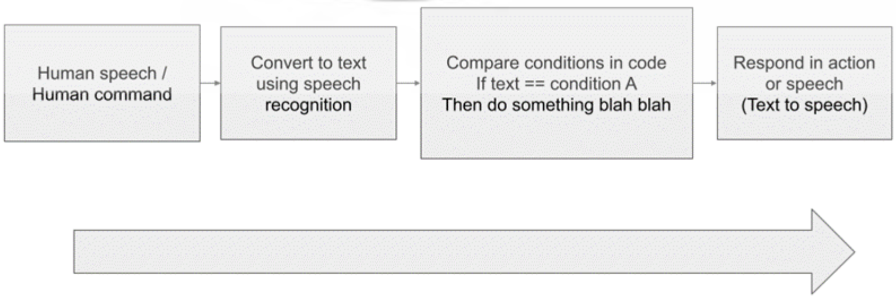


Fig 17. System Methodology

3.3 Interesting Problems

3.3.1 Plan methodology problems

The main purpose of agile design methodology is to develop, debug, and can fulfil the need of users frequently. In this case, the user is our advisor and us as developer. This designed is chosen so that functions can be fixed and added as required by the advisor easily. However, the main problem of such plan are the requirement of hard-working and good collaboration both in development team, and between user and developer. In addition, the new developer who will join the development team later on during the development won't be able to catch up easily.

3.3.2 System methodology problems

The interesting problem happens because the requirement that the system must be on-standby 24/7. However, the system is voice command. Therefore, when the system received voice that is not matched with the command, it would say error all the time. The solution is to find a way to make the system receive the command only when needed while ignore the speech the rest of the time. Otherwise, the problem was like this. While the machine is on, when someone say "He plays Cannon In D". It's obvious that the speaker didn't intend to use the voice assistant. However, it coincidentally has the word that fulfil the condition to execute the command "play Cannon in D on Youtube." Another problem is voice interference. When the machine operates in the noisy area, it usually can't receive the command.

3.4 Proposed Solution

3.4.1 Plan methodology solution

The agile plan might cause some problem if the chemistry of members development team doesn't match. Fortunately, the development team chemistry works well. Therefore, no problem about team work needed for any attention. The problem about the work-coordination between advisor and developer is none. Our advisor always gives out feedback, and add-on function while we always fulfil his feedbacks accordingly.

3.4.2 System methodology solution

The solution for always receive the command even though when it didn't have to can be fixed by adding wake-word. The concept of wake word is a name. If the machine didn't get call its name, the machine will not start receive the command even the keyword of the command is correct. Therefore, a new block diagram of the system can be shown below.

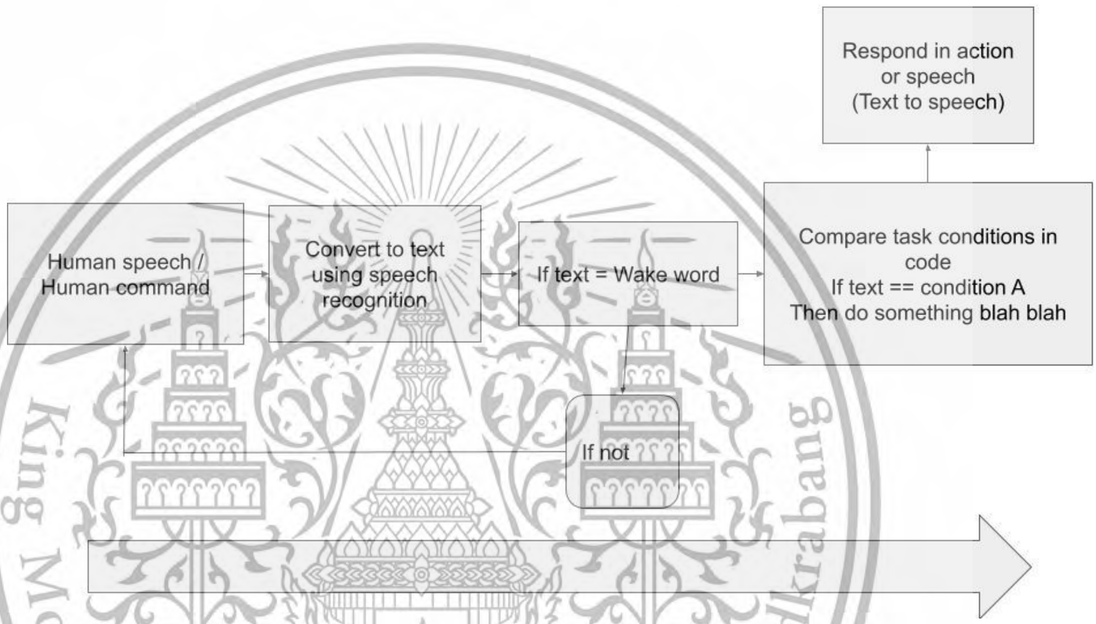


Fig 18. System Methodology Solution

The solution for noise interference is a bit tricky. A few solutions at the moment are to use this machine in quiet area. Another solution is to switch the normal microphone with a microphone that has noise cancelation feature. A solution that currently implemented is adjusting noise threshold using built in command from speech recognition library.

3.5 Summary

This chapter described the high-level requirements and design of both a system and development plans which had been used in developing this smart voice assistant. The chapter started by describing the methodologies use in system design and development plan. The interesting problems also mentioned and discussed at the following section. (Section3.3). The proposed solution was then discussed in section 3.4 followed by a summary of this chapter in Section 3.5.

The system design in this Chapter 3 is covered in further detail in Chapter 4 which describes the implementation of NLP, Animation, SMTP, Video Call, Voice Chat and Python coding.

CHAPTER 4
EXPERIMENTAL RESULT

4.1 Introduction

Chapters 3 and 4 described the design and implementation of NLP, Python, Animation, and other required knowledges. A system is born after all afore mentioned knowledges are combined. The system that is born is the "Hello Bot" a Smart Voice Assistant. It's a system that its purpose is to make seniors' life become better, and more convenient when using technology. The target customer of this technology is including but not limited to just senior, for example bed-ridden patient, broken arm patient, and any person who're in need of a voice assistant can use as well. This thesis's project is a result from all the knowledge acquired throughout time at Robotics and AI. The bachelor degree of Engineering at King Mongkut's Institute of Technology Ladkrabang.

In this chapter 4, presents a testing method and its results that show its ability that appropriate for its purpose of existence for senior. Furthermore, show its potential that anyone except senior can also use and gain benefit from this system. The chapter is organized as follows: Section 4.1 introduces and describes the project. The details will be covered in this order. First is why decided for the voice assistant as a project (Section 4.1.1), then explain the work flow box diagrams for all functions in the system (Section 4.1.2), and how. What are the knowledges, techniques, and tools used which made this project as is shown (Section 4.1.3). Next, section 4.2 presents the testing summary of the Smart Voice Assistant, testing as well as proving its practicality, stability, and sustainability. Then, section 4.3 will be the evaluation part which will discuss and evaluate its potential, and underlying issue of the technology. Section 4.4 will be the summary of this chapter 4.

4.1.1 Reason for Developing "Hello Bot" Voice Assistant.

The reason that sparked the development is that our team personal experience with our own family. Each of us used to help open movie for our grandparents when they want to watch some movie on the internet. Most of the time they watch Drama on

Youtube. In addition, our grandparent stayed at home most of the time. However, we have other activities that prevent us from staying home all the time. Therefore, our grandparent won't be able to enjoy watching movies when we're not presented. Moreover, because the ageing some time cause them to forget to do some important activities. For instance, forget to take the pill prescribed by doctor etc. So, that brought us to the idea that how good would it be if we can assign a personal assistant to them that can be on standby 24/7 ready to service. Hiring a real personal assistant seem less appropriate in term of cost and privacy wise. In addition, after researching, the situation all around the world is not much different from ours. The detail of research is as provided in chapter 1 of this paper. Therefore, in order to help more senior and people in need, we decided to develop a technology that is easy to use, designed for them, and cost effective. That's why developing "Hello Bot" the Smart Voice Assistant is our final conclusion.

4.1.2 Smart Voice Assistant System (From 1st capstone)

This system was developed in the 1st semester as an individual project. The functions are listed below.

1) Voice commands for using Youtube.
    - Voice command to open application.
    - Voice command to search.
    - Voice command to play a video
2) Voice command for using Google.
    - Voice command to open application.
    - Voice command to search.
3) Voice command for using Gmail.
    - Voice command to send Gmail.
    - Voice command to name the Subject.
    - Voice command to select who to send a Gmail to base on name.
    - Voice command to name the detail of the Gmail.
4) Voice command for reminder system.
    - Voice command to initiate the reminder system.
    - Can choose what to be reminded freely.
    - Can set timer.

- Work parallelly with main program

5) Voice command for taking note system.
   - Voice command to initiate the take note system.
   - Voice command to write the detail of the note.

4.1.3 Functions' Box Diagrams.

4.1.3.1 Youtube Box Diagram

Use wake word to make the robot on stand-by for command. After saying the wake word, in this case the wake word is "Hello". The user can command this hello bot voice assistant to open Youtube, play any video on Youtube based on voice input, and search any video on Youtube base on any voice input. The system flow is shown in fig 19.

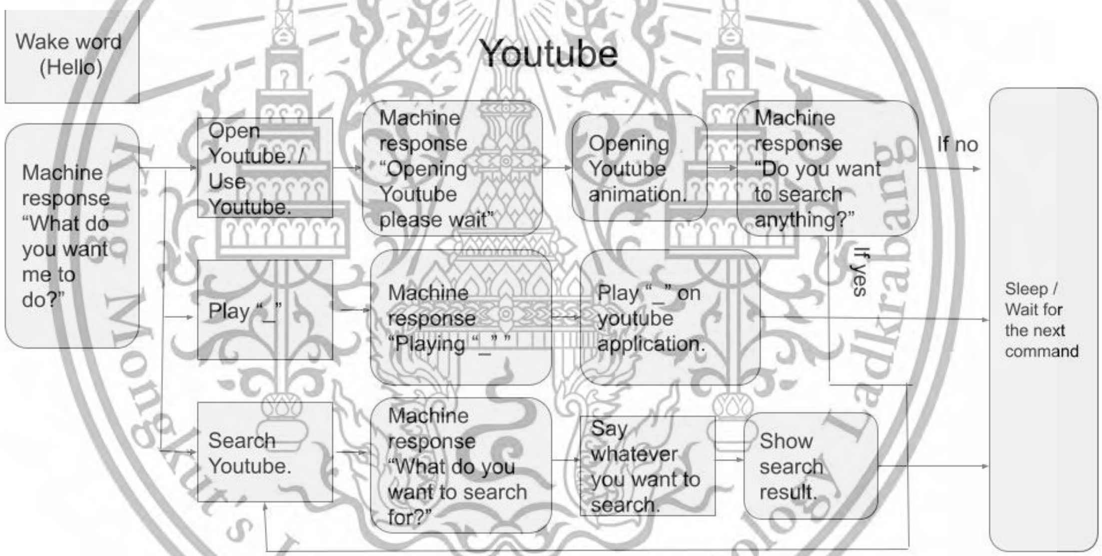


Fig 19. Youtube System Flow Box Diagram

For instance, if a user wants to use Hello bot to open Youtube, all he/she has to say is first "Hello" to wake the robot up. Then, Hello bot will ask "what do you want me to do?". In this example, to open Youtube, so the input must be any sentence as long as it includes the word "open youtube" or "use youtube". For example, "I want to use youtube", "open youtube for me", or even just by saying keyword is fine as well. After the robot recognized command, the response is "opening youtube please wait", and at the same time displaying the animation to shows that it's currently opening the Youtube application. Then, the user will be landed on Youtube application home page. At this stage, the robot will ask "do you want to search anything?". At this point, the user can give either a yes or a no. If no is an input, the system will go to sleep while waiting for the next command, letting user freely search on the Youtube home page. However, if yes is an input, the system will ask "what do you want to search". Users can give any search input. For example, "search robot". After the system received "search robot" as the command, all search result relevant to the input will be shown, and then the system go to sleep while waiting for the next command. As shown in fig 20.

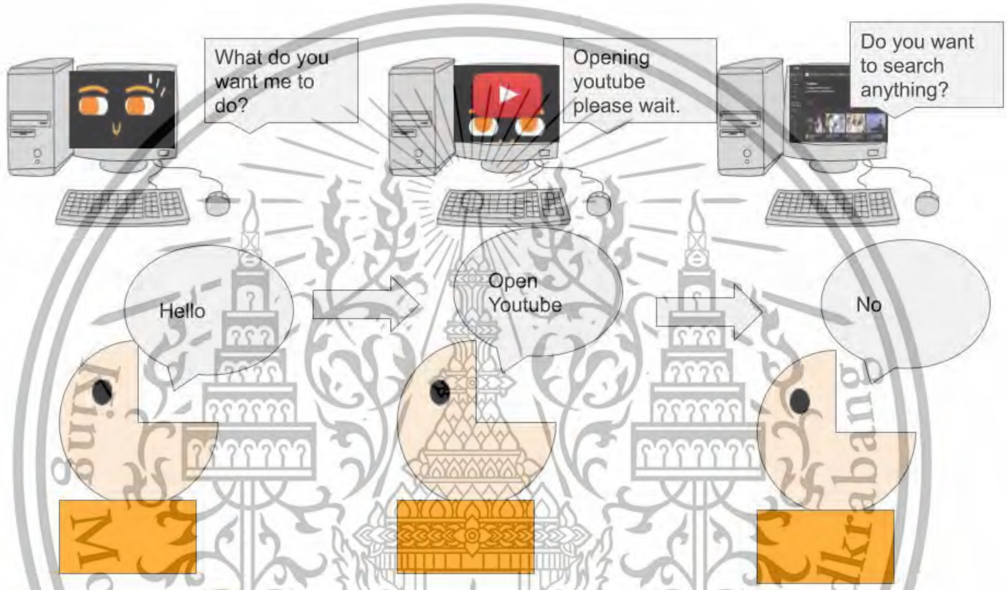


Fig 20. Youtube Open Command

Another two functions relevant to using Youtube application are play and search. Search function is basically to search any video on Youtube based on voice input. If the user wants to use search function, for example, he/she needs to say wake word first. The system, then, will response "what do you want me to do?". Actually, the system response has more than just "what do you want me to do?". The machine response is set so that it gives out random response with similar meaning. For example, "yes sir", "affirmative", and "what do you want me to do?". The reason for doing so is to shorten the different between Smart Voice Assistant bot and the actual living voice assistant. However, in this paper will only mention "what do you want me to do" as the machine feedback after the wake word in the following sections for simplicity and easy to understand. Come back to search youtube function, after machine gave the response "what do you want me to do", the keyword for using search Youtube function is any sentence that includes "search youtube" in it. He/she can say "search youtube for me please". The machine response will be "What do you want me to search". At this point, the user can say anything that he/she wants to search (no keyword required). As shown in fig 21.

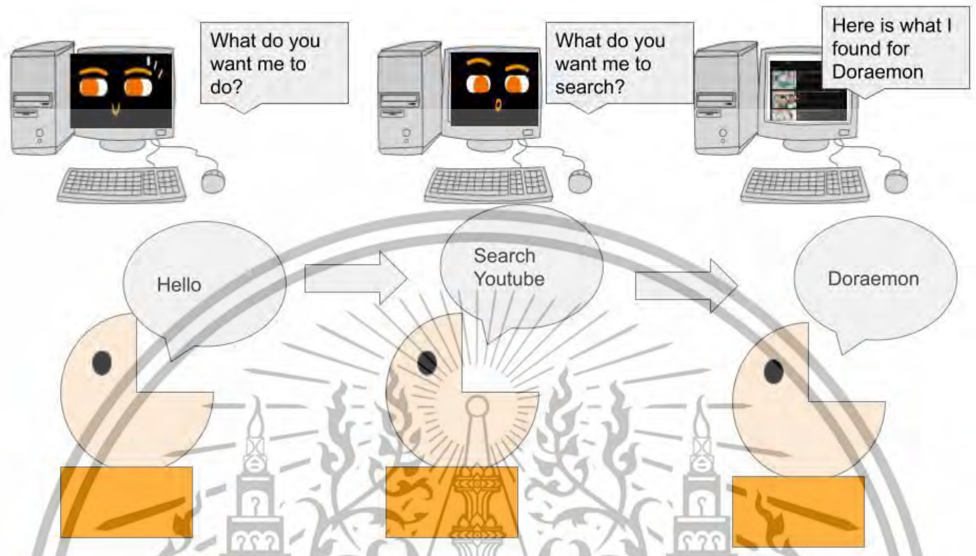


Fig 21. Youtube Search Command

Play Youtube is the last function relevant to using Youtube application. The result of this keyword "play" is to play the specific video on the application using voice to name what video the user wants the system to play. As shown in fig 22.

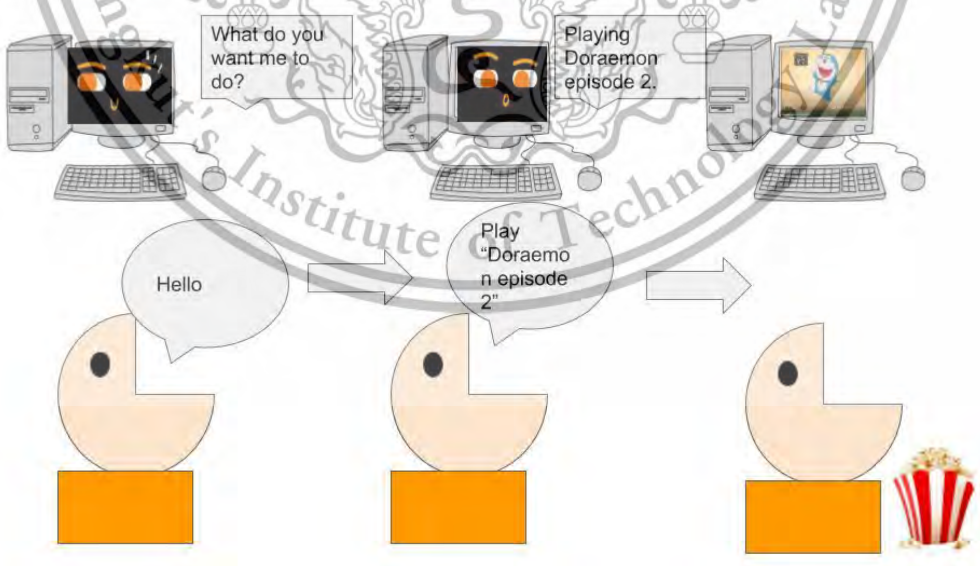


Fig 22. Youtube Play Command

4.1.3.2 Google Box Diagram

Using google function in “Hello Bot” is easy. Suitable for people of all ages. Simply saying wake word to start the system. After the system ready to receive user command, saying “what do you want me to do?”. To use Google on “Hello bot”, saying “open google” is the keyword for such command. The flow after that will be as shown in fig 23.

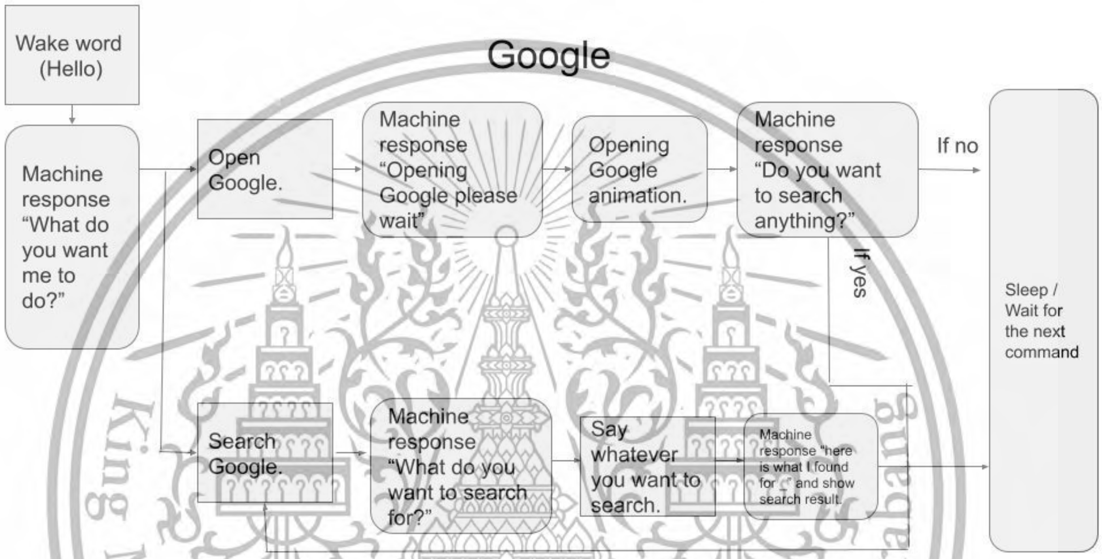


Fig 23. Google System Flow Box Diagram

That is, after the system received “open google” as a command, the system then displays opening google animation and at the same time give out a response in voice “opening google please wait”. Then, the system will ask “do you want to search anything after the google application is opened. If the input is no, user will be landed on google home page. User can search or use google freely while the system goes to sleep and waiting for the next command. If yes, on the other hand, user will be sent to Search google function.

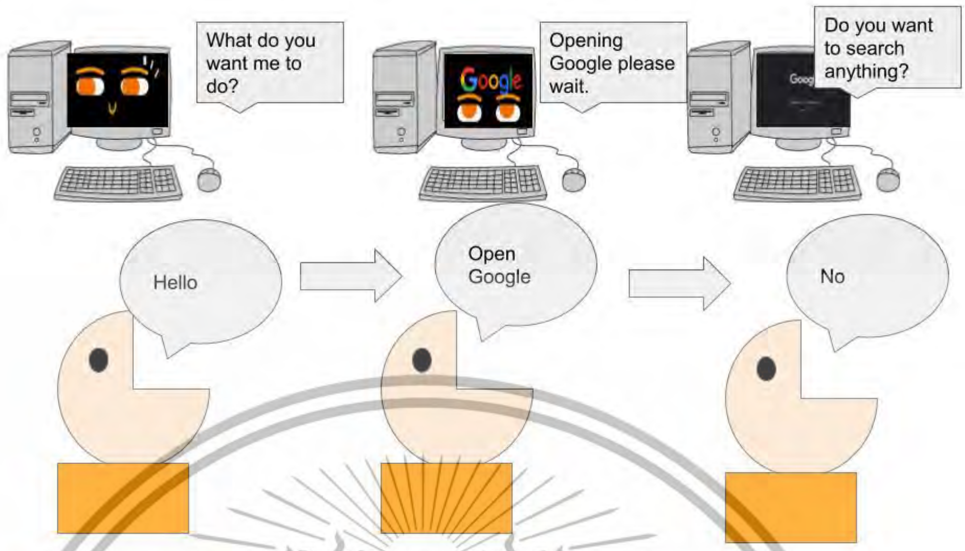


Fig 24. Open Google Command.

For Search google function, it's a function where it designed to help user search anything on google application based on any voice input it received. The flow of how this system works is as shown in fig 25. To put it simply, search google either follow from previous "open google" after user saying "yes" as the input after the system asks "If you want to search anything?" or such system can be initialized straightaway right of the bat after saying wake word follow by "search google". After the "search google" is received as the command. The system, then will ask "what do want to search for?". At this point, user can give any voice input, and the system will search google based on the voice input given.

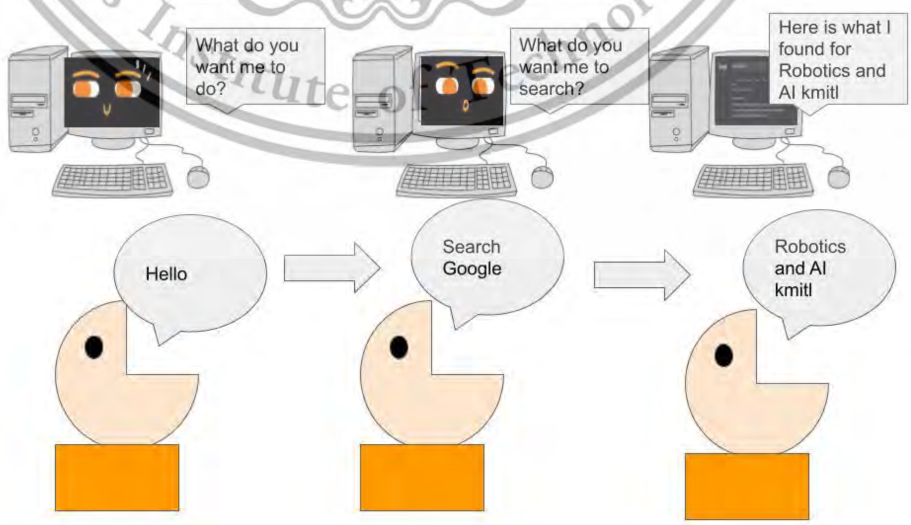


Fig 25. Search Google Command

4.1.3.3 Gmail Box Diagram

The function for using Gmail in "Hello Bot" is designed so that the user who're not tech oriented can use Gmail to send text message to another person, using only voice with ease. Metaphorically speaking, this function is designed so that only saying the name of the person user wants to send text message to follow by naming the subject and detail of the email is all that required for the user. The rest of the progress are done by the system automatically. The system flow is shown in fig 26.

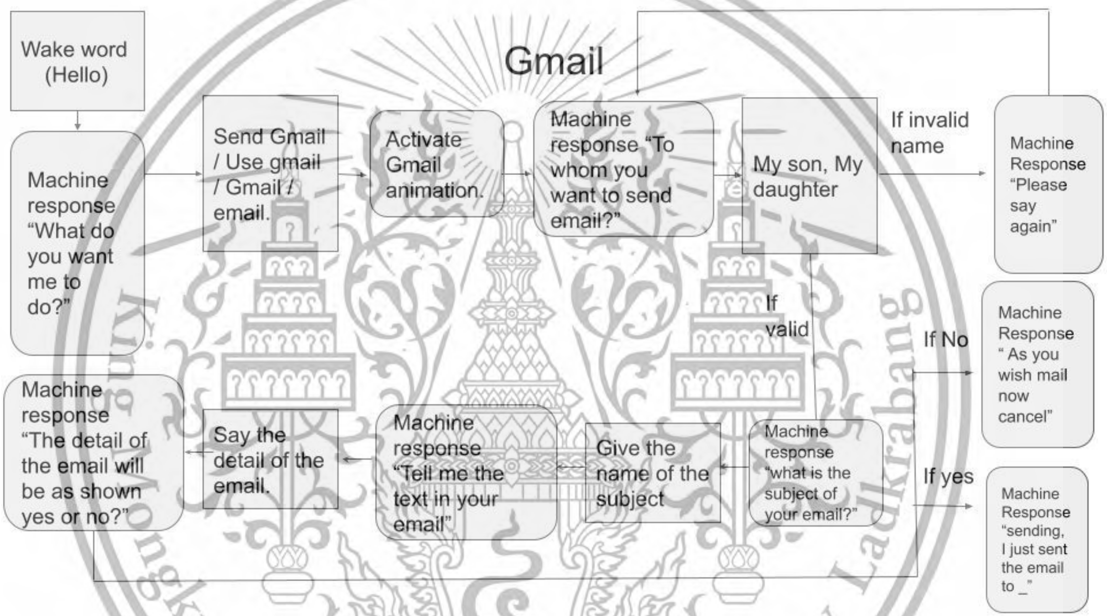


Fig 26. Gmail System Flow Box Diagram

For instance, to use this function on "Hello Bot" smart voice assistant, simply waking it up as usual, and saying any sentence that has gmail or email includes in it after the machine gave out voice response "what do you want me to do?". An animation of Gmail function activating will follow, and machine will ask "To whom you want to send email?". At this point, the system required user to say the name of the person that user wants to send the email to. The name of the person required are linked with the email address. Therefore, it's required to be set up beforehand. However, after it's set up, the user can just say the name that is saved and the system will link the name with the designated email address automatically. In addition, the number of contacts is unlimited, and the name of the contact can be named freely. If in case the input given by user doesn't match up with any name in the database, the system will give out response saying "Please say again" to prompt user to say name of the receiver again. If the name of the user is correct, on the other hand, system will ask user to name the subject of the email. After the user names the subject, system will ask user to say the text detail of the email. Finally, after everything is done, the system then displays the content of the email, including the subject, receiver, and details of the email. At this stage, system asks for the confirmation of the process from the user. If user confirm by saying "yes" after double checked all the details of the email, the system will send to the designated email automatically. As the confirmation after the system double checked that the email is sent successfully, it gives response "I just sent the email to (name of the contact)". If user saying "no" or "cancel", on the other hand, system will terminate the subject, and detail. Then, system goes back to sleep, waiting for the next command input.

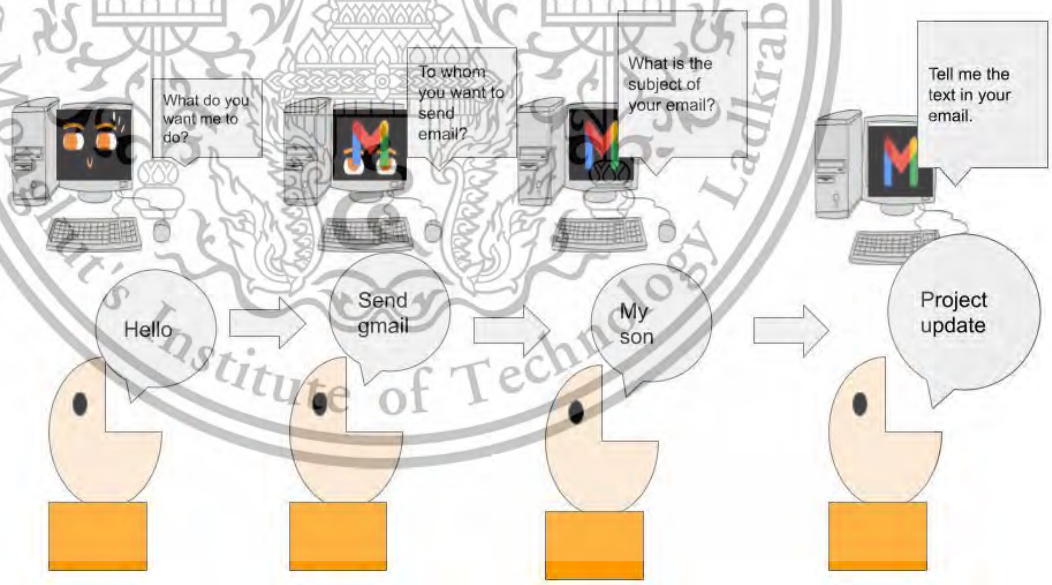


Fig 27. Gmail Voice Command Part 1

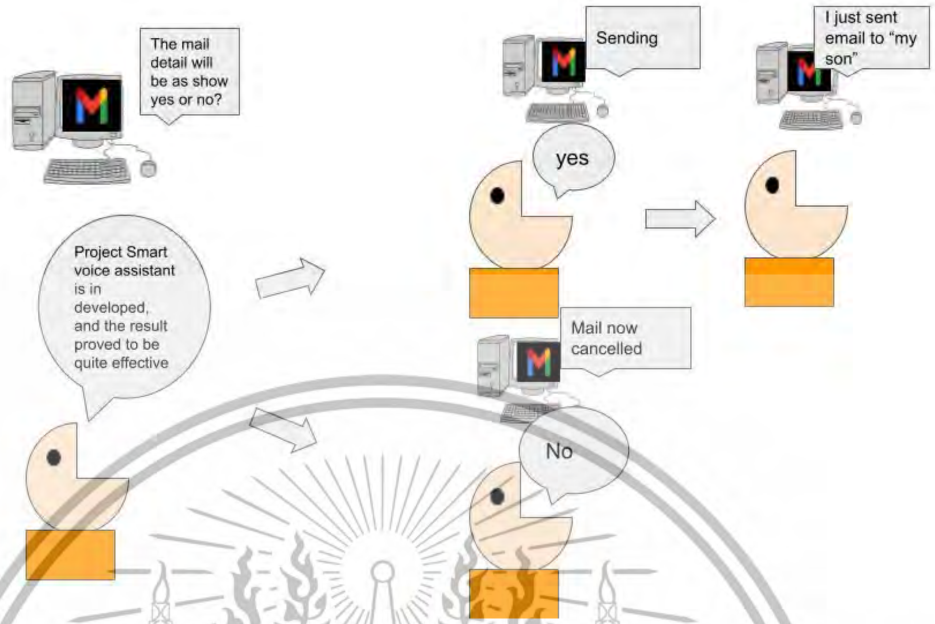


Fig 28. Gmail Voice Command Part 2

4.1.3.4 Reminder Box Diagram

A function called reminder is as its name suggested. It's a system that designed to remind user of anything based on voice input, and can set the timer as well. The system flow is shown in fig 29.

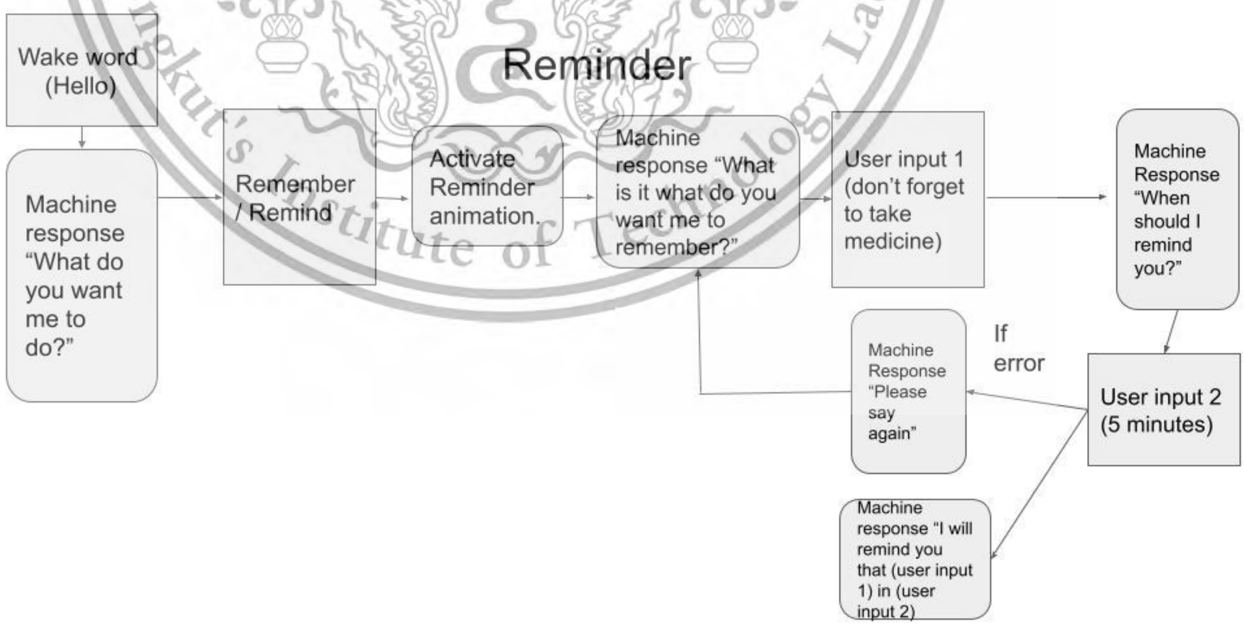


Fig 29. Reminder System Flow Box Diagram.

The system is designed such that it works simultaneously with other function. Meaning is that in basic code ladder work flow where it flows from first line to second line, any code will not be executed in jumping manner. For example, if the timer is set in a reminder system, it will count until the timer reached zero before moving on to execute other command. However, if that is the case, it obstructs the users to use other services this "Hello Bot" can provides. Therefore, a technique called "threading" is developed and implemented in this system. The result is that after the timer is set, the system starts counting down the timer at the background while it main stream system provides services to user normally like how it's designed to do so. If one word is needed in order to describe such technique. It must be "Multitasking".

In order to use reminder function from "Hello Bot", user must wake up the robot first same as any other previous functions. Then the keyword to activate this function is either remember or reminder. Any sentence includes either of those two words can activate this function. After the function is activated, machine will ask "what is it what do you want me to remember". The user is prompt to say anything he/she is desired for the robot to help he/she remembers. For example, "Do not forget to take medicine", "Don't forget to close the light", or "Don't forget the food is being heated in the oven" just for a few examples. After the system received user's input of what it must remember, the system will ask when to remind. At this point, the system is asking user to set the timer. The timer can be set staring from 1 minute up to multiple hours. In addition, while the timer is ticking. It works in background. Therefore, users can use other services available on this "Hello Bot" normally.

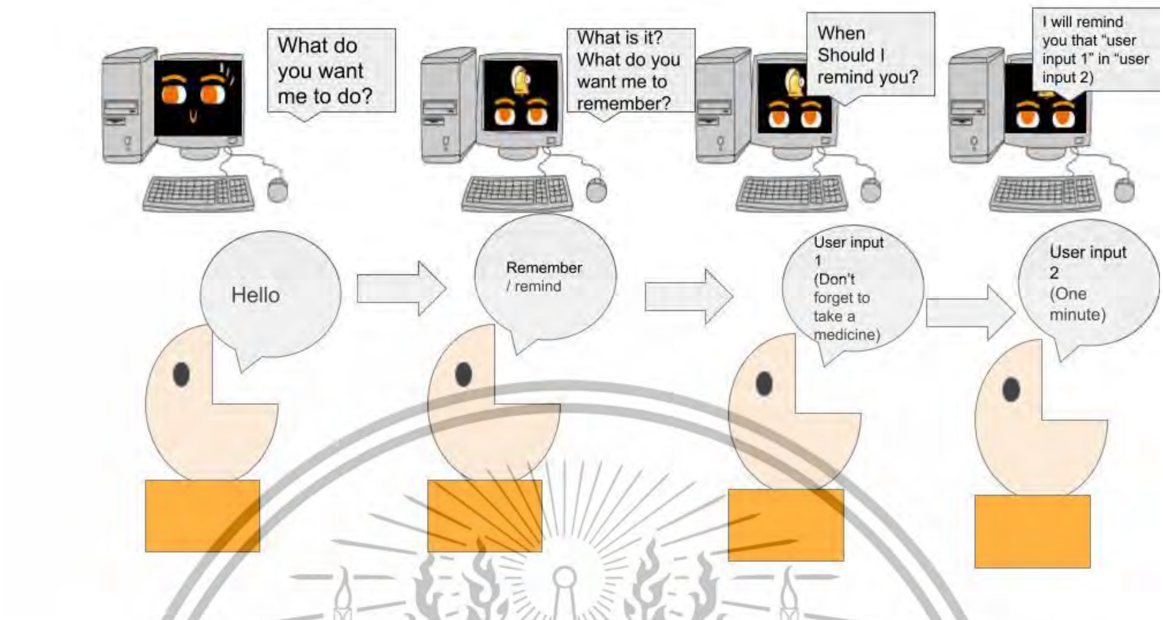


Fig 30. Reminder Voice Command

4.1.3.5 Take Note Box Diagram

Take Note function is a system designed as a replacement for an actual living personal assistant who is capable of writing what was told in text. This system is integrated into "Hello Bot" which enable its user to command the bot to note down whatever the user wishes, and the bot will write and save automatically. The system flow is shown in fig 31.

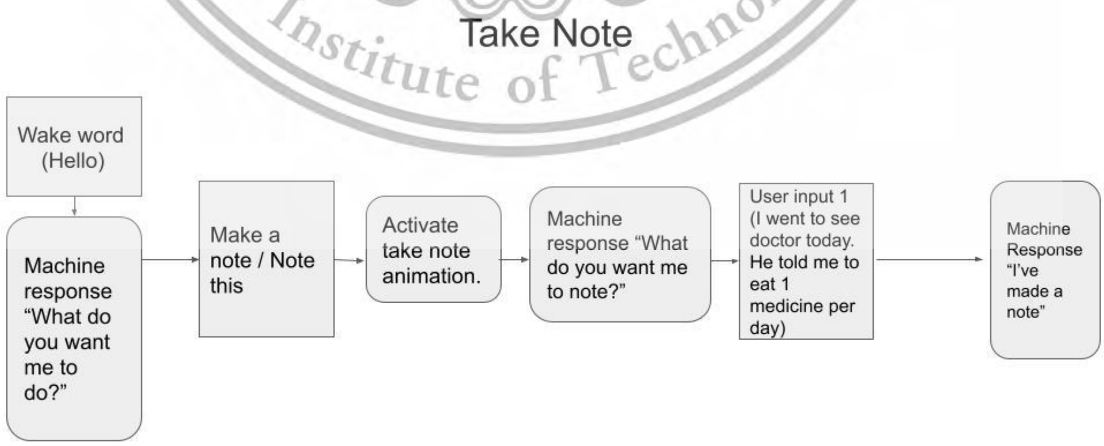


Fig 31. Take Note System Flow Box Diagram

The file name of the note being saved is based upon date and time of when it's saved, so that the user can come back to check out the note pretty easily. The use case of this function is for example, for senior who received doctor recommendation on how to treat themselves, and afraid to forget what was told by the doctor. This function can help taking note, and left as a long-term usage. It can be used to remind seniors or even other people. For example, doctor, senior's children, and senior grandchildren.

The way to use this function is simply activate the robot by saying wake word as shown in fig 32. After the robot prompt the user "what do you want me to do", keywords to active take note function are "make a note", and "note this down". The system will then ask user "what do you want me to note?". User can say all the note detail he/she desires. Then, system will automatically save the note for user in date and time named format.

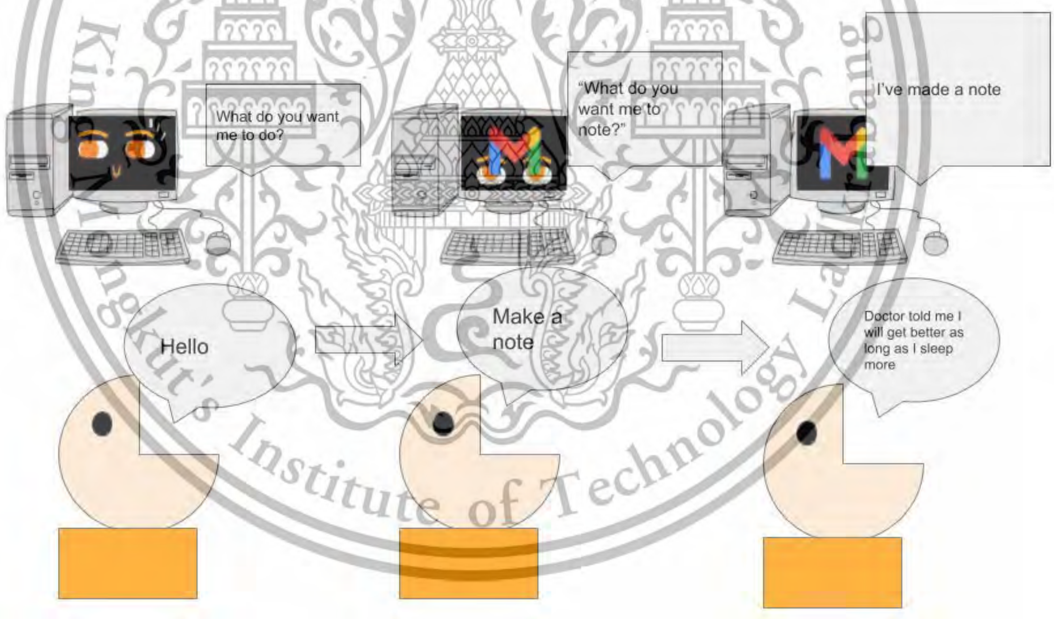


Fig 32. Take Note Voice Command

4.1.4 Hello Bot Implementation

<u>Youtube Function</u>

Code Language: Python

The following are module imported to use in operating all function related to Youtube in “Hello Bot”.

- speech_recognition as sr is a module used to convert human voice input into text format.
- Pyttsx3 is a module used to convert text into machine voice to give out the machine voice response.
- Pywhatkit is a module used to play a video on youtube based on code. In this case, the video will be played according to voice input.
- Webbrowser is a module used to open youtube on webbrowser, and it is also used for browsing youtube based on voice input as well.

```
Import speech_recognition as sr
import pyttsx3
import pywhatkit
import webbrowser
```

Fig 33. Youtube Function’s Modules.

<u>Speech_recognition Code</u>

A function to convert voice to text is called `get_audio():`. So that developer can simply use text = get_audio() if want to convert any voice into text format in further development. In its detail of this `def get_audio():`. It enables microphone as input device. Then, using `audio = r.listen(source)` to saves what is received from microphone into audio while letting said empty as “_” (`said = “”`) for the upcoming steps. Next putting audio into said r.recognize_google(audio) (`said = r.recognize_google(audio)`) to change from voice to text then save the text into “said”. Furthermore, to avoid any upcoming problems caused from upper case and lower case. Code the return of said to be all lower as shown in fig 34 `return said.lower()`.

```
Def get_audio():
    r = sr.Recognizer()
    with sr.Microphone() as source:
        audio = r.listen(source)
        said = “”

        try:
            said = r.recognize_google(audio)
            print(said)
        except Exception as e:
            print(“Exception: “ + strI)
    return said.lower()
```

Fig 34 Speech_recognition Code Implementation

Pyttsx3 Code

To make code easy to implement and develop, a function speak(text) is made, and shown in fig 35. So, the text in the blanket is the text for machine to speak. For example, speak(“hello”) for the machine to say hello.

For the code. Rate is the speed of voice the machine speaks (`rate = 100`). `Engine = pyttsx3.init()` is to initialize the pyttsx3. `Voices = engine.getProperty(‘voices’)` is to get all the available voices installed for machine to use. `Engine.setProperty(‘voice’,voices[0].id)` is to select what type of voice for machine to use. In this example, voice number is zero. It’s a male voice. `Engine.setProperty(‘rate’, rate+50)` is to use the rate set before with the voice number zero. The rate is set to 100. Furthermore, the rate can be further enhanced both faster and slower as shown. In this case, it’s set to be +50. So, the rate in total is 150. `Engine.say(text)` is a code to command machine to say according to the text. `Engine.runAndWait()` is simply just run the code and wait.

```
Def speak(text):
    rate = 100
    engine = pyttsx3.init()
    voices = engine.getProperty(‘voices’)
    engine.setProperty(‘voice’,voices[0].id)
    engine.setProperty(‘rate’, rate+50)
    engine.say(text)
    engine.runAndWait()
```

Fig 35 Pyttsx3 Code

Pywhatkit Code

The code shown in fig 36 is a code use to execute play video command on Youtube. It the code that use pywhatkit to play a video based on what was saved in song (`pywhatkit.playonyt(song)`). The data in song is saved after the input is named after "play" (`YT_STRS = ["play"]`). For example, "play Doraemon episode two". What will be saved in the song only include "doraemon episode two" because the word play is be replaced as empty by `song = text.replace(phrase, '')`.

```
YT_STRS = ["play"]
    for phrase in YT_STRS:
        if phrase in text:
            song = text.replace(phrase, '')
            speak('playing' + song)
            pywhatkit.playonyt(song)
```

Fig 36. Pywhatkit Code Implementation

Webbrowser Code

The code shown in fig 37 is how webbrowser is used to open youtube on web browser. In addition, it also shows usages and implementation of what was discussed in previous modules.

```
OP_YOUTUBE = ['open youtube']
    for phrase in OP_YOUTUBE:
        if phrase in text:
            speak('opening youtube please wait')
            url = 'https://youtube.com'
            webbrowser.get().open(url)
            speak('youtube now open')
            speak('do you want to search anything?')
            text = get_audio()
            Yans = ['yes', 'i want to','search']
            for phrase in Yans:
                if phrase in text:
                    search_youtube()
            Nans = ['no','i don\'t want']
            for phrase in Nans:
                if phrase in text:
                    speak('as you wish')
```

Fig 37. Webbrowser Code Implementation

```
url = 'https://youtube.com'
```

Fig 38. URL for Web Designation

```
webbrowser.get().open(url)
```

Fig 39. Opening Web URL.

Google Function

Code Language: Python

All import modules as shown in Fig 40 are necessary to operate Google Function in "Hello Bot". They are exactly the same as Youtube function for `speech_recognition as sr` and `pyttsx3`.

```
Import speech_recognition as sr
import pyttsx3
import webbrowser
```

Fig 40. Google Function's Modules.

The different is how to code using in webbrowser is different from youtube. The code is as shown in fig 41. This code is for search function. As shown, search target is what will be used to search in google, and it's saved such that it can search anything. The code to doing so is `url = 'https://google.com/search?q=' + search_target`. Then the url will be according to what the user wants to search. For example, if a word "robot" is what perceived in get_audio(), the url will be `'https://google.com/search?q=robot'`.

```
Def search_google():
    speak('What do you want to search for?')
    search_target = get_audio()
    url = 'https://google.com/search?q=' + search_target
    webbrowser.get().open(url)
    speak('Here is what I found for' + search_target)
```

Fig 41. Code for Google Search.

The search result will be robot as shown in fig 42.

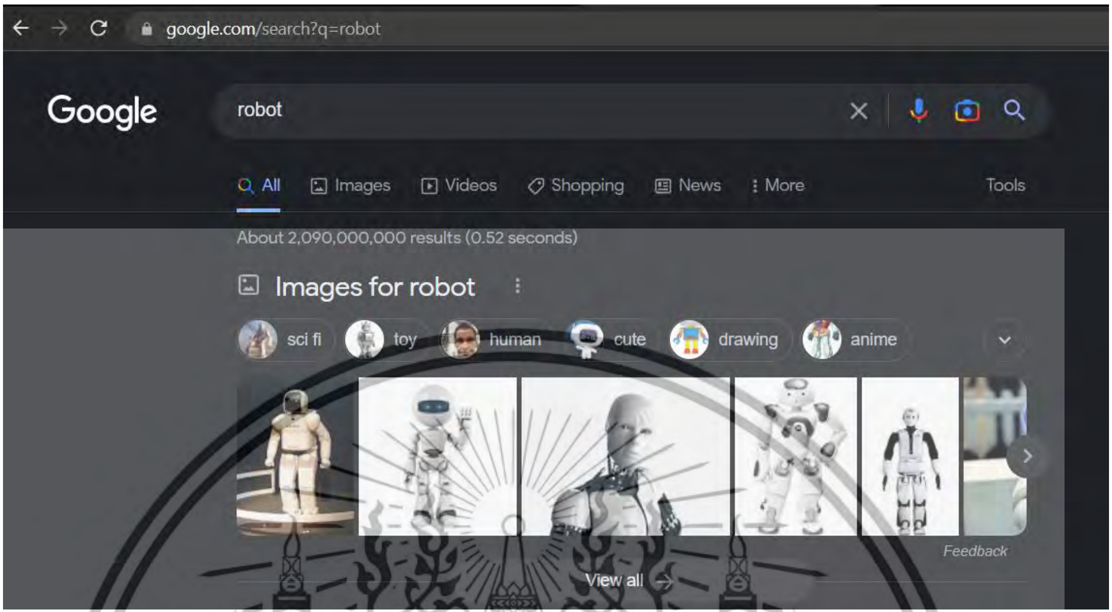


Fig 42. Search Target's Search Result.

Gmail Function

Code Language: Python

All required modules are listed in Fig 43. Speech_recognition and pyttsx3 are modules used for received and converted voice from human into text for computing in case of speech_recognition, and pyttsx3 is for converting text into machine voice. Their processes are as shown previously in Section Youtube Function.

Gmail function used `smtplib` and `from email.message import EmailMessage` for sending message via gmail. Speech_recognition is used to convert human voice into command or text for machine to put into mail. Pyttsx3 is used for machine voice interaction between machine and user. The purpose is to make the "Hello bot" easy to use.

```
Import speech_recognition as sr
import pyttsx3
import time
import smtplib
from email.message import EmailMessage
```

Fig 43. Gmail Function's Modules

This Gmail Function is programmed to make machine receive command, write email, and send email following user's voice input. The Gmail command initialized after the word email or gmail is detected. The code for that is shown in fig 44.

```
EMAIL_STRS = ["email","gmail"]
    for phrase in EMAIL_STRS:
        if phrase in text:
            get_email_info()
```

Fig 44. Initiate Gmail.

After "Hello Bot" acknowledged the to use Gmail as a command from user, user will be landed at `def get_email_info():`. In this function, machine is programed to prompt user to input the receiver's name. The step is as shown in fig 45.

```
Def get_email_info():
    speak('To whom you want to send email')
    name = get_audio()
    try:
        receiver = email_list[name]
        print(receiver)
```

Fig 45. Get Info For Gmail Function.

The name is linked to the email_list shown in fig 46. That being said, if the user's voice input is "my son", machine will put 62011272@kmitl.ac.th as the receiver.

```
Email_list = {
    "my son" : '62011272@kmitl.ac.th',
    "daughter": 'fachaijp@gmail.com'
}
```

Fig 46. Email List

After the name of the email's receiver is acquired and saved in the "receiver" variable. Machine next step is to prompt user to input the subject name of the email, and the detail of the email systematically. The code for that is as shown in fig 47. Subject name is saved in "subject variable" while text in the email is saved in "message" variable.

```
        Speak('what is the subject of your email?')
        subject = get_audio()
        speak('Tell me the text in your email')
        message = get_audio()
```

Fig 47. Detail of the Email.

After all variables are saved, prepared, and ready to send to the receiver. "Hello Bot" will ask for confirmation of the email. Of course, user can choose to send right away, or cancel the email as shown in fig 48.

```
        Speak("the mail detail will be as shown")
        speak("yes or no ?")
        text = get_audio()
        mail_yes = ['yes','ok','correct','confirm']
        for phrase in mail_yes:
            if phrase in text:
                speak('sending')
                send_email(receiver, subject, message)
                speak("I just sent the email to" + name)
        mail_no = ['no','don/t','cancel']
        for phrase in mail_no:
            if phrase in text:
                speak('as you wish')
                speak('mail now cancel')
```

Fig 48. Confirmation

If decided to cancel, simply saying "no", "don't", or "cancel" will suffice for machine to delete all saved data. However, if wished to send, simply say "yes", "ok", "correct", or "confirm" for machine to run `def send_email(receiver, subject, message):`. As you can see in fig 49, this def used all the saved data acquired from previous `def get_email_info():`. To put it simple, it will send to email address saved in receiver, the subject of the email will be what was saved in subject, and the text in the email will be what was saved in message.

```
Def send_email(receiver, subject, message):
    server = smtplib.SMTP('smtp.gmail.com', 587)
    server.starttls()
    server.login('fachai62011272@gmail.com', 'xspdfroyagjneanv')
    email = EmailMessage()
    email['From'] = 'fachai62011272@gmail.com'
    email['To'] = receiver
    email['Subject'] = subject
    email.set_content(message)
    server.send_message(email)
```

Fig 49. Def Send Email

For further explanation of the above code, it used "Simple mail transfer protocol" to send the email. It required `server.login('fachai62011272@gmail.com', 'xspdfroyagjneanv')` to set up the sender email.

In case of user's input for receiver doesn't match with any of the name listed in the email list, the program is coded so to prompt user to repeat the name again as shown in fig 50.

```
Except:
        speak("Error the person name" + name + "is not in the system")
        speak("please try again")
        get_email_info()
```

Fig 50. Gmail Please Say Again.

Reminder Function

Code Language: Python

All modules used in Reminder function is listed in fig 51.

```
Import speech_recognition as sr
import pyttsx3
import time
import threading
```

Fig 51. Reminder Function's Modules

The key feature of this function is to ask what to remind, and when to remind. User is prompted by the machine to give voice input what is it for machine to remember. After doing so, user is required to state when should it be for machine to remind. This function full code is shown in fig 52.

```
Def help_remind():
    speak("what is it? What do you want me to remember")
    42azy = get_audio()
    try:
        speak("when should I remind you")
        time_remind = get_audio()
        time_sleep = time_list[time_remind]
        print(time_sleep)
        speak("I will remind you that   " + 42azy + "in   " +
time_remind)
        time.sleep(time_sleep)
        speak("master you asked me to remind you to   " + 42azy)
    except Exception as e:
        print("Exception: " + strI)
        speak('please say again')
        help_remind()
```

Fig 52. Def Help Remind

The amount of time for user to wait until it relays back to reminder message is acquired from time list shown in fig 53. The time list as shown in Fig 53 only has two variables.

They are one minute, and five minutes. However, that is not the limit. The time list can be added further. For instance, one hour, two hours, and even ten hours just for some example.

```
Time_list = {
    'one minute' : 60,
    'five minute' : 300,
}
```

Fig 53 Time List

In case user's input is "three minutes" that is not matched with any of the time listed in the time list, the code will prompt user to give the input again as shown in fig 54.

```
Except Exception as e:
        print("Exception: " + strI)
        speak('please say again')
        help_remind()
```

Fig 54 Reminder Please Say Again.

The last must have module is threading. Threading use for multitasking, meaning it runs the reminder function parallelly with other function. Therefore, user can use other function while letting machine run the timer waiting to remind user. The code show threading's usage is shown in Fig 55.

```
Thread_help_remind = threading.Thread(target=help_remind)
thread_help_remind.start()
```

Fig 55 Threading.

<u>Note Function</u>

Code Language: Python

All required modules for note function shown in Fig 56.

```
Import speech_recognition as sr
import pyttsx3
import subprocess
import datetime
```

Fig 56 Note Function's Modules

Note Function's ideal purpose is as a notetaker replacement, meaning user give voice input for all detail of the note user's desires to notetaker for taking note. To initiate the function simply say either "make a note" or "note this". Program will automatically prompt user to fill all the detail for the rest of the process. The text in the note is saved in `note_text = get_audio().lower()`. As shown in Fig 57.

```
NOTE_STRS = ["make a note",'note this']
```

```
    for phrase in NOTE_STRS:
        if phrase in text:
            speak("What do you want me to note?")
            note_text = get_audio().lower()
            note(note_text)
            speak("I've made a note")
            notedown_demo()
```

Fig 57 Note Main Code

The `note_text` will be saved as note.txt file format. The file name will be according to the date and time of when the note is being saved. The code is as shown in Fig 58.

```
Def note(text):
    date = datetime.datetime.now()
    file_name = str(date).replace(':',"-") + '-note.txt'
    with open(file_name, "w") as f:
        f.write(text)
    subprocess.Popen(["notepad.exe", file_name])
```

Fig 58 Save Note Code

<u>Animation</u>

Animation required developer to create a set of multiple images to create animation (Fig 59). In this thesis, Adobe After Affect is used to create the images.

Fig 59 Example of Set of Image for Animation

Code implementation for using animation required developer to import pygame. Furthermore, developer need to set up what images to be used and in what order. The step is simply setting up the picture saved folder's location as shown in Fig 60. In Fig 60, it only consists of ytube1 up to ytube4. However, as shown in Fig 59, the total number of images exceeded 69. Therefore, the code shown in Fig 60 must be duplicated until more than "ytube69".

```
Def ytubeupload():
    handler.loadFromFile ( "D://Final Project For Graduation/TestGUserinterface/youtube/ytube001.jpg",  "ytube1"
    handler.loadFromFile ( "D://Final Project For Graduation/TestGUserinterface/youtube/ytube002.jpg",  "ytube2"
    handler.loadFromFile ( "D://Final Project For Graduation/TestGUserinterface/youtube/ytube003.jpg",  "ytube3" )
    handler.loadFromFile ( "D://Final Project For Graduation/TestGUserinterface/youtube/ytube004.jpg",  "ytube4" )
```

Fig 60 Set Up Pictures for Animation

Previously, we discussed the location and order of images used in creating animation. The next step is to load those images in order and specific time, so that the animation will be created. Code implementation is shown in Fig 61. The variable x and y represent the size of the display image in imagery x and y coordinate. `Handler.render` is where developer put the picture designed to display, as you can see in Fig 61 first picture in handler is "ytube45". It's the name of one of the pictures saved and named in Fig 60. In between `handler.render` of "ytube45" and "ytube46", there is time delay `time.sleep(.1)` used to create Frame Per Second.

```
Def ytubeupdisplay():
    A = 0
    B = 0
    x = 1920
    y = 1080
    handler.render ( screen, "ytube45", ( A, B ), True, ( x, y ) )
    pygame.display.update(A,B,x,y)
    time.sleep(.1)
    handler.render ( screen, "ytube46", ( A, B ), True, ( x, y ) )
    pygame.display.update(A,B,x,y)
    time.sleep(.1)
```

Fig 61 Load Pictures for Animation

To display all those load and setup pictures discussed previously in specific location of the code only required a simple code as `ytubeupload()`, then `ytubeupdisplay()`.

Prior to all those set up, load, and display, required to setup Pygame. A class called imageHandler (Fig 62) is used to setup so that those set up, load, and display can be used later on easily.

```
Class imageHandler:
  def __init__ ( self ):
    self.pics = dict()

  def loadFromFile ( self, filename, id=None ):
    if id == None: id = filename
    self.pics [ id ] = pygame.image.load ( filename ).convert()

  def loadFromSurface ( self, surface, id ):
    self.pics [ id ] = surface.convert_alpha()

  def render ( self, surface, id, position = None, clear = False, size
= None ):
    if clear == True:
      surface.fill ( (5,2,23) )

    if position == None:
      picX = int ( surface.get_width() / 2 - self.pics
[ id ].get_width() / 2 )
    else:
      picX = position [0]
      picY = position [1]

    if size == None:
      surface.blit ( self.pics [ id ], ( picX, picY ) )

    else:
      surface.blit ( pygame.transform.smoothscale ( self.pics [ id ],
size ), ( picX, picY ) )
```

Fig 62. Pygame Imagehandler

In order to use Pygame, developer also required to initiate it using what is shown in Fig 63. First is to initialize pygame with `pygame.display.init()`. Then, configure the size of the display screen image. In Fig 65 is shown using 1920, 1080, meaning it’s programmed to shown in full-screen mode. Finally is to use `handler =`

`imageHandler()` to call set up as shown in Fig 62 to use in implementation for the rest of pygame's animation implementation (Fig 59 - 61).

```
Pygame.display.init()
screen = pygame.display.set_mode((1920,1080), pygame.RESIZABLE)
handler = imageHandler()
```

Fig 63. Initiate Pygame

4.2 Testing Summary

In total ten tests were executed. Nine out of ten test cases were perfectly satisfied the coder expectation, and this data was then used to further develop this project. The tests illustrate that the quality of microphone play a major role in project related to voice assistant. In the next section, this project will be evaluated and the extent to which it supports the thesis is discussed.

Table V.

Total Number of Test Cases Planed	Total Number of Test Cases Executed	Total Number of Passed Test Cases	Total Number of Failed Test Cases
10	10	9	1

4.3 Evaluation

The testing described in section 4.2 demonstrates that not only software is important. Although code is undeniably important in creating Robotics and Ai such like voice assistant, hardware is also another element which can't be ignored. As it's crystal clear in section 4.2 shown a 10% error. After some investigation, the cause of such error is due to microphone. The microphone using in this project doesn't include noise canceling. Therefore, the machine might receive wrong voice command if operates in the crowded environment. The implementation of NLP, SMTP, Animation, and Python also yield and beautifully satisfied developer expectation. The 90% rate of success is promising as it is, and it will be even more promising if this project can increase the rate of success by fixing the cause of error as stated, and develop even further. In this section therefore, evaluated the implementation and discussed issues in the underlying technologies that the implementation has highlighted.

4.4 Testing and Evaluation Summary

This chapter introduced the implementation of NLP, SMTP, Animation, and Python into creating a “Hello Bot” voice assistant in pursue of assisting senior. All features of this voice assistant include the interaction between human and machine. All the development process were discussed and shown in concept, system flow, and code in section 4.1. In section 4.2 a series of tests were described which demonstrated the practicality of this project.

An evaluation was then presented in section 4.3. Section 4.4 is a chapter summary of chapter 4 for capstone 1 individual project. In the next couple section, section 4.5 – 4.8 will covers the detail of what being added in capstone 2. Capstone 2 is a group project, the main goal is to improve the system, and add new functions.

4.5 Capstone 2

Section 4.5 onward will be the add-on project to what previously discussed in previous chapters. Capstone 2 is a group project that we decided to add three new functions into the original “Hello-bot”. Those functions are ChatGPT, New voice assistant model to work with ChatGPT, and Video Call. Moreover, we developed in Ubuntu operating system instead of window which was the OS that used in the original. The details will be covered in this order. First is why we decided to add those functions (Section 4.5.1), then explain those functions. What are the knowledges, techniques, and tools we used which made this project as is shown (Section 4.5.2). Next, section 4.6 presents the testing summary of the Smart Voice Assistant with new functions, testing as well as proving its practicality, stability, and sustainability. Then, section 4.7 will be the evaluation part which we will discuss and evaluate its potential, and underlying issue of the technology. Section 4.8 will be the summary of section 4.5 to section 4.7.

4.5.1 Reason for adding three new functions

For ChatGPT, it's being added to enabling the voice assistant to chat like a chatbot. Therefore, the "Hello bot" with ChatGPT can talk to senior when they are bored in addition to being the artificial assistant.

For new voice assistant model, the project requires to make new voice assistant model to make user's experience seemless, and work in harmony with ChatGPT. The new model is as shown in Fig 64, and the original model is as shown in Fig 65.

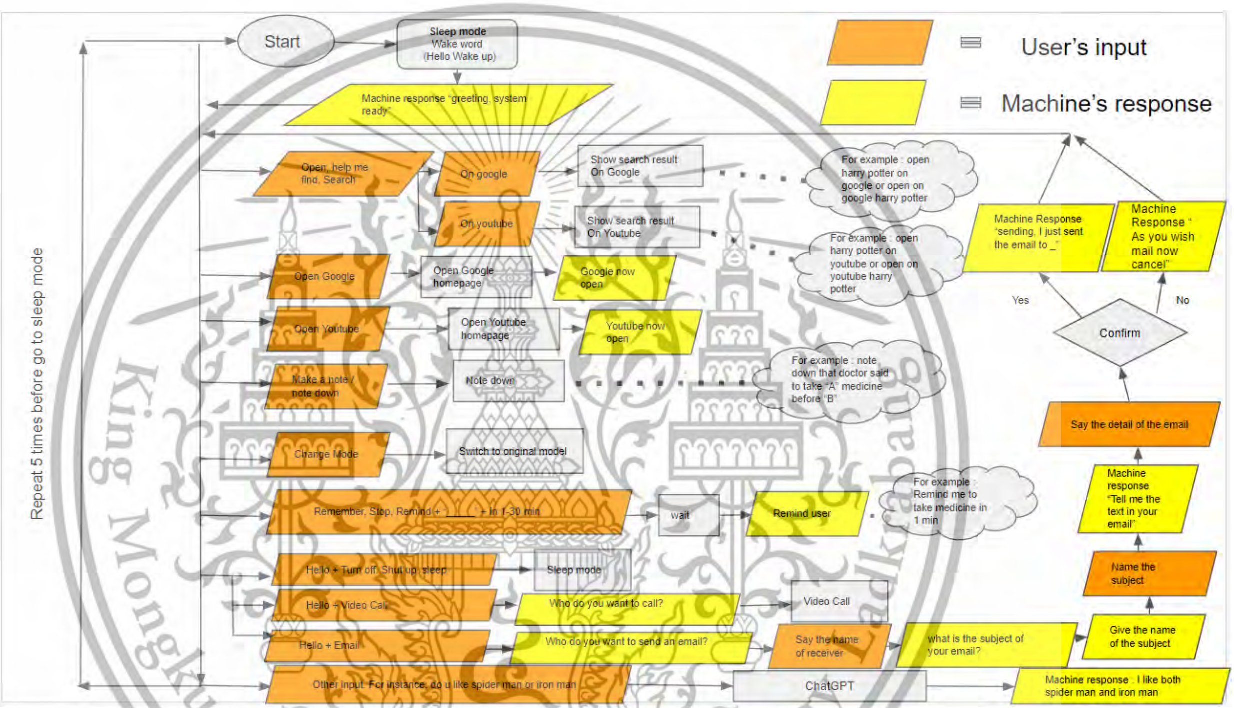


Fig 64. "Hello Bot" Model Diagram (FINAL)

When compare with the original model in Fig 65, the final model enable user to do commands in one step while the original required user to give input step by step. Both of them have pros and cons. However, it's required to make new model so the system can differentiate input for voice assistant command and input for ChatGPT more accurately. When user say something that is not matched with the condition of the commands, ChatGPT will answer that back to the user automatically, and that includes conversation dialogues, enabling ChatGPT to acts like Chat bot.

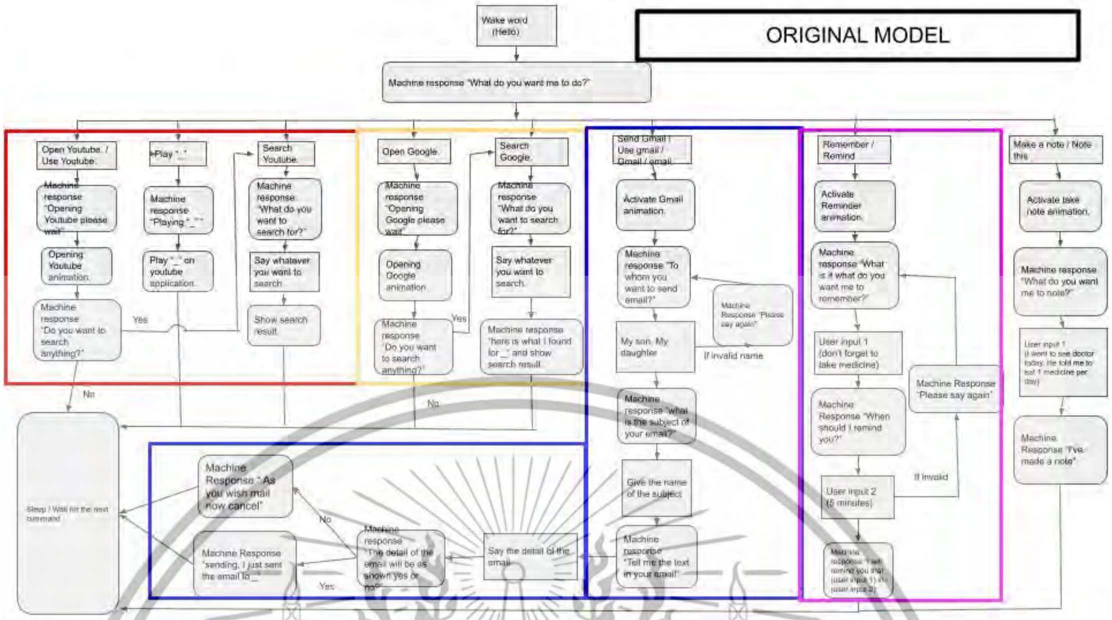


Fig. 65 Original Model

For video call, it's added because seniors and seniors' family can use this video call function to see and talk to each other free of charge, and face to face.

Video Call Function

Code Languages: Python, Javascript, CSS, HTML

There are two parts of the Video Call feature. The one written in Python is for sending an email invitation. And another is for running the server and making connections.

All modules used in Video Call Invitation function are listed in fig 43. This function uses the Gmail function to send an email invitation message to a receiver listed in the email list with a pre-defined URL to the video call website.

The flowchart in Fig 66 shows how to use the voice command.

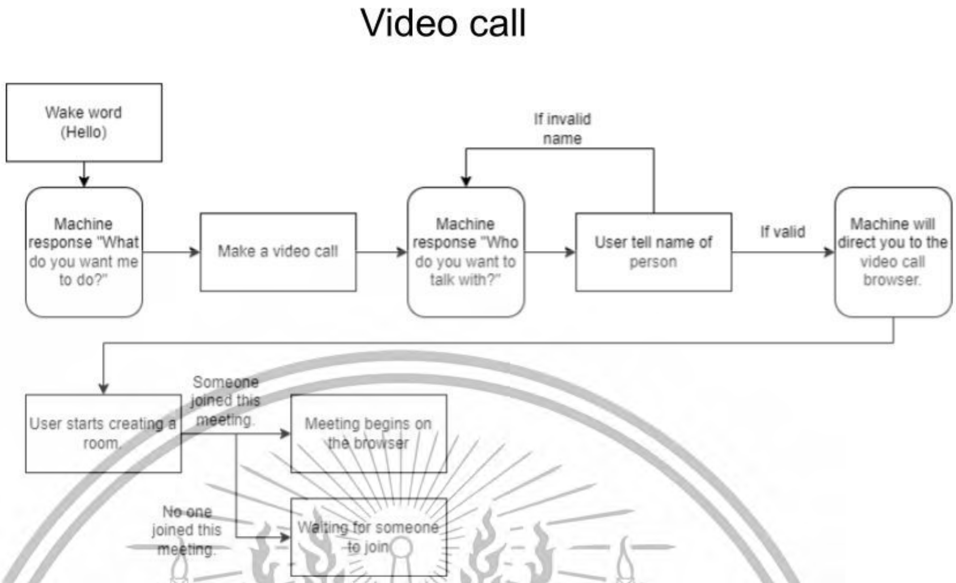


Fig 66 How to use voice command.

The script that runs the server and makes connections is described in the sequential diagram below (Fig 67).

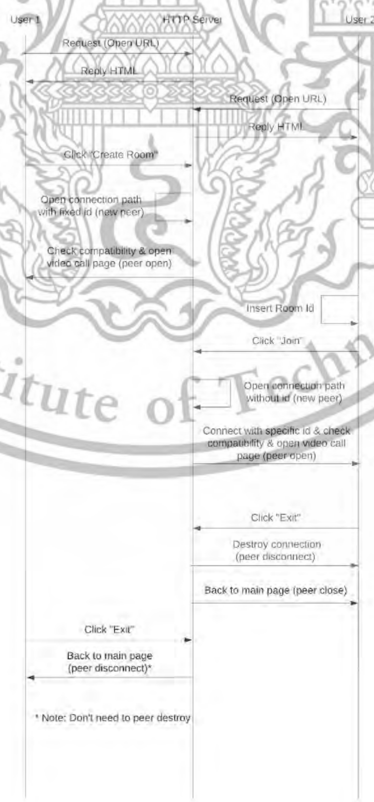


Fig 67 Sequential diagram of script.js.

ChatGPT

Code Language: Python

ChatGPT is a trained model created by openai. It's a Chatbot that is trained and can be interact in a conversational way which is suitable for many tasks. For instance, normal conversation, answer question, up to create a code if the user wish for. In addition, ChatGPT provides many models, and they all have different purposes. By the time of developing this project. The latest version is ChatGPT 3 or "text-davinci-3". This model's capability is from normal conversation, mathematical related, up to creating a code. However, using ChatGPT required API, and the cost for "text-davinci-3" in response to its capabilities is not little. Therefore, we decided to use "text-curie-001" instead. Because the main purpose for adding ChatGPT to the "Hello bot" is to enable the robot to have normal conversation with human. The model "text-curie-001" is suffice for this task with a reasonable cost.

ChatGPT has many pros, but it also has some drawback due to the trained data were up to 2021. In addition, ChatGPT can't perform task that voice assistant can perform. For example, open or make a search on google or youtube, make a note, make a video call, send an email etc. ChatGPT itself is a chatbot.

4.5.2 Hello Bot Implementation

New Voice Assistant model

Youtube Functions

Play command

"Hello bot" to play a specific video right away can be commanded so by saying "play 'on the floor' on youtube". Not only that users can use this command in many other manners. For instace, "on youtube play 'on the floor'". As long as there are the word play, and "on youtube, in youtube, at youtube" presents in the speech. What made this command possible is as shown below in fig 68. In addition, to erase the word play and "on youtube, in youtube, at youtube" presents in our search result, adding `text.replace(phrase, '')` can get rid of those unrequired words.

```
YT_STRS = ["play"]
        for phrase in YT_STRS:
```

```
            if phrase in text:
             song = text.replace(phrase, '')
             LAST_CHECK = ["on youtube","in youtube","at youtube"]
             for phrase in LAST_CHECK:
                if phrase in song:
                   song = text.replace(phrase,'')
                   speak('playing' + song)
                   pywhatkit.playonyt(song)
                   sleep_wake()
```

Fig 68 Youtube Funtion “play” command (new)

Search command

“Hello bot” is designed to recognized “help me find what movie should I watch” is not a command but a question for ChatGPT to generate the answer by making “Hello bot” recognize that when the user only say “help me find, open, use, search, show” is not a command unless combine with “on google” or “on youtube”. The code that made this possible is shown in Fig 69.

```
      OPEN_ALL_STRS = [“open”,”use”,”help me find”,”search”,”show”]
        for phrase in OPEN_ALL_STRS:
            if phrase in text:
               GOOGLE_CHECK = [“on google”,”in google”]
               for phrase in GOOGLE_CHECK:
                  if phrase in text:
                     googleload()
                     googledisplay()
                     speak(“opening on google”)
                     search_target = text.replace(phrase, ‘’)
                     url = ‘https://google.com/search?q=’ +
search_target
                     webbrowser.get().open(url)
                     speak(“Here is what I found”)
                     AUTOCHAT()

               YOUTUBE_CHECK = [“on youtube”,”in youtube”]
               for phrase in YOUTUBE_CHECK:
                  if phrase in text:
                     ytubeupload()
                     ytubeupdisplay()
                     speak(“opening on youtube”)
                     search_target = text.replace(phrase, ‘’)
                     url = ‘https://youtube.com/search?q=’ +
search_target
                     webbrowser.get().open(url)
                     speak(“Here is what I found”)
```

```
                    AUTOCHAT()
```

Fig 69 Search Command (new)

Reminder Function new

New version of reminder function is designed to be activated after user say "help me remember, stop me, remind" and then specify the time for "Hello bot" to remind the message back to user. Using this function for example, "stop me from doing homework in 1 minute" or user can say "stop me in 1 minute from doing homework" either ways work for this command due to newly designed NLP model. Moreover, the "Hello bot" automatically convert "me" and "I" to "you" which makes the sound playback better. In addition, the message runs using threading, meaning while the time is ticking for the message to be played back, user can still use other functions or chat with robot. The code for this function is as shown in Fig 70.

```
NOTE_X_REMIND1 = ["help me remember","remind","stop me"]
        for phrase in NOTE_X_REMIND1:
           if phrase in text:
              FOR_ONE = ["in 1 minute","one minute"]
              for phrase in FOR_ONE:
                 if phrase in text:
                    text = text.replace("me","you")
                    text = text.replace("i","you")
                    text = text.replace("in 1 minute","")
                    time_remind = "1 minute"
                    time_sleep = time_list[time_remind]
                    speak("I will remind you that   " + text +
"in   " + time_remind)
                    thread_help_remind =
threading.Thread(target=AUTOCHAT)
                    thread_help_remind.start()
                    time.sleep(time_sleep)
                    speak("user you asked me to   " + text)
                    thread_help_remind.join()

              FOR_TWO = ["in 2 minute", "two minute"]
              for phrase in FOR_TWO:
                 if phrase in text:
                    text = text.replace("me","you")
                    text = text.replace("i","you")
                    text = text.replace(phrase,"")
                    time_remind = "2 minutes"
                    time_sleep = time_list[time_remind]
```

```
                    speak("I will remind you that   " + text +
"in   " + time_remind)
                    thread_help_remind =
threading.Thread(target=AUTOCHAT)
                    thread_help_remind.start()
                    time.sleep(time_sleep)
                    speak("user you asked me to   " + text)
                    thread_help_remind.join()
```

Fig 70 Reminder Function (new)

Take Note Function (new)

In the original version of "Hello bot". After activating take note function, the robot will ask what do you want to note. However, in this new version, the user can command the robot to make a note in 1 step. For example, "make a note that I have to take "A" medicine every day until next month." The code for this is as shown in Fig 71.

```
Def note(text):
    date = datetime.datetime.now()
    file_name = str(date).replace(':',"-") + '-note.txt'
    with open(file_name, "w") as f:
        f.write(text)
    subprocess.Popen(["notepad.exe", file_name])
NOTE_FINAL = ['make a note','take note','new note','take a note','make
note']
        for phrase in NOTE_FINAL:
            if phrase in text:
                NOTE_SAVE = ['and save','save','and save it']
                for phrase in NOTE_SAVE:
                    note_text = text.replace(phrase,'')
                    note(note_text)
                    AUTOCHAT()
```

Fig 71 Take Note Function (new)

Open Youtube and Open Google (new)

For this new version, using open youtube or open google simply open their homepage for user after use the command word. The command word for this function is "open google" for open google homepage, and "open youtube" for open youtube homepage. The code is as shown in Fig 72.

```
      OPEN_YOUTUBE_STRS = ["open youtube","use youtube"]
        for phrase in OPEN_YOUTUBE_STRS:
            if phrase in text:
```

```
                ytubeupload()
                ytubeupdisplay()
                speak("opening")
                url = 'https://youtube.com'
                webbrowser.get().open(url)
                speak("open")
                AUTOCHAT()

        OPEN_GOOGLE_STRS = ["open google","use google"]
        for phrase in OPEN_GOOGLE_STRS:
            if phrase in text:
                googleload()
                googledisplay()
                speak("opening")
                url = 'https://google.com'
                webbrowser.get().open(url)
                speak("open")
                AUTOCHAT()
```

Fig 72 Open Youtube and Open Google (new)

Turn off, Email, Video call

To turn off, send an email, or make a video call, user has to say the name “Hello” anywhere in the command. For instane, if the user want to turn off the “Hello bot”, user has to say “hello turn off” or “hello shut up” or “go to sleep hello”. For your information, the sequence of how user say doesn’t matter. User can say both “turn off hello” or “hello turn off”.

For sending an email, user has to say “hello send an email” or “hello send a gmail” as long as there are both “hello” and “gmail” or “email” present in the speech. The reason for “hello” to be required prior to using these commands is to prevent the robot from misunderstanding user’s purpose. Because user can ask “tell me what is email” for ChatGPT to answer. Adding “hello” makes the system can differentiate between this potential issue, increase the accuracy of the robot. The rest of the step is the same as original model.

Video call also similar to two previous commands. Saying hello follows by make a video call to activate. Then, the system will ask to who do you want to make a video call. At this point, user has to say the name of the one he/she wants to call. Then the call invitation will be sent automatically to that person email address. After that, “Hello bot” will open video call browser for user automatically to make a call.

The code for all afore mentioned funtions is as shown below in Fig 73.

```
GO_TO_SLEEP = ['hello']
        for phrase in GO_TO_SLEEP:
           if phrase in text:
              speak('i am here')
              SLEEP_NOW = ['turn off','power off','sleep','shut up']
              for phrase in SLEEP_NOW:
                 if phrase in text:
                    speak('go to sleep mode')
                    wakingupload()
                    backtosleepdisplay()
                    sleep_wake()

              EMAIL_STRS = ["email","gmail"]
              for phrase in EMAIL_STRS:
                if phrase in text:
                    gmailload()
                    gmaildisplay()
                    get_email_info()
                    AUTOCHAT()
              CALL_STRS = ['video call']
              for phrase in CALL_STRS:
                 if phrase in text:
                    invite_video_call()
                    wakingupload()
                    backtosleepdisplay()
                    sleep_wake()
```

Fig 73 Turn Off, Email, Video Call (new)

For asking video call receiver and opening video call website is as shown in Fig 74. The video call website use WebRTC. Therefore, the user is required to connect to the pre-defined URL. As shown in Fig 74 is `webbrowser.get().open("https://1901-171-97-128-207.ap.ngrok.io")`

```
def invite_video_call():
    speak('Who you want to video call')
    name = get_audio()
    try:
        receiver = email_list[name]
        subject = 'Hello Bot - Video call invitation'
        message = f'Please open {"https://1901-171-97-128-
207.ap.ngrok.io"} and follow the below instructions to join.\n\n'
        message += 'Instruction: \n'
        message += '\t1. Open the link in browser\n'
```

```
        message += "\t2. Click the 'Visit Site' blue button, if there is any.\n"
        message += "\t3. Click the 'xxx.html' url, if there is any.\n"
        message += "\t4. Enther the meeting code, click 'Join'.\n"
        message += '\t5. To exit the call, simply close the browser tab.\n\n\n'
        message += '-------------------------------------------------\n'
        message += 'This message is auto generated and sent by ... team.'

        send_email(receiver, subject, message)
        speak('Video call invitation is sent to ' + receiver)
        speak('Opening video call website')
        webbrowser.get().open("https://1901-171-97-128-207.ap.ngrok.io")
        speak('Click create room to create room and join')

    except:
        errorload()
        errordisplay()
        speak('Error the person name' + name + 'is not in the system')
        speak('do you want to try again')
```

Fig 74 Invite and Open Video Call

ChatGPT

In order to use ChatGPT with "Hello bot" voice assistant, an api is required which can be acquired from openai official website. Moreover, a new library call `import openai` is needed. Then specify the API key as `openai.api_key = "your api here"`. The concept is for speech recognition that already has to convert the speech of the user into text, then send those text using openai api key to ChatGPT to generate the text feedback. Finally, makes the machine speaks those text back to user in machine voice. Having text that is converted from speech as `text`, then customize the name of the robot as `bot_name`, and also makes the robot knows user name by giving `user_name`. Then, save all those variables in `prompt`. The last step before making ChatGPT generate the response is to combine with `conversation` that is set to be blank originally. The final result will be like in Fig 75.

```
conversation = ""
user_name = "user"
bot_name = "Jiraff"
```

```
prompt = user_name+":"+text+"\n"+bot_name+":"
conversation += prompt
```

Fig 75 ChatGPT Receive Input

Now for the ChatGPT to generate the response the code in Fig 76 is used which can be acquired from openai playground. This code enable user to select the model. In this project `model="text-curie-001"` is selected. Set input to be `prompt=conversation`. The diversity of output set to be none `temperature=0`, meaning if the output is "hello my name is hello". The next time it answers will also be "hello my name is hello". For how long the sentence ChatGPT generates, to make the output to be precise not too long or short. Therefore, it's set to be `max_tokens=256`. This number can be increased up to 4000.

```
response = openai.Completion.create(
            model="text-curie-001", # select model
            prompt=conversation,    # Select prompt for input
            temperature=0,          # Diversity of output
            max_tokens=256,         # Maximum word for output
            top_p=1,
            frequency_penalty=0,
            presence_penalty=0
        )
```

Fig 76 ChatGPT Customization

After customizing the ChatGPT parameter, to makes it speak out the response. Its required using the code provided below in Fig 77.

```
response_str = response["choices"][0]["text"].replace("\n","")
response_str = response_str.split(
     user_name + ":", 1)[0].split(bot_name+ ":",1)[0]
conversation+=response_str+"\n"
        print(response_str)
        speak(response_str)
```

Fig 77 ChatGPT Output

New Text to Speech gTTS

In order to use pyttsx3 that was used in "Hello bot" original version on Ubuntu, an espeak module must be installed. However, the quality of sound is bad which is uncomparable to the sound from window OS. Therefore, an alternative TTS is a must. In this case, another free text to speech call gTTS is chosen. In order to use gTTS, `from gtts import gTTS` and `import playsound`. Then create new function call `def speak(text):`. Configure it to be english by adding "en", `tts = gTTS(text=text, lang='en')` Save the Speech into mp3 file `filename = "audio.mp3"` and then `tts.save(filename)`. Then, play the file `playsound.playsound(filename)`.

The way this function looks like is as shown in Fig 78.

```
def speak(text):
    print(text)
    tts = gTTS(text=text, lang='en')
    filename = "audio.mp3"
    tts.save(filename)
    playsound.playsound(filename)
```

Fig 78 New Text to Speech gTTS

4.6 Testing Summary

In total of ten test has been conducted for each function. The environment that was conducted in was a bit crowded and not 100% quiet all the time. The result is as shown in table VI. In summary, the results were perfectly satisfied our expectation. In addition to the testing that was being conducted as shown in section 4.2. The problem with microphone was mentioned. However, in this project, an improvement was made by making the sound input automatically adjust for the ambient voice, so the system can differentiate what is the input and what's not more accurately. In the next section, section 4.7, the project will be evaluated and the extent to which it supports the thesis is discussed.

Table VI.

Functions	Total Number of Test	Total Number of Passed Test	Total Number of Failed Cases
Youtube	10	9	1
Google	10	9	1
Reminder	10	7	3
Make Note	10	10	0
ChatGPT	10	10	0
VDO call	10	9	1
Email	10	8	2

4.7 Evaluation

The testing described in section 4.6 demonstrates that improving sound input by adjusting the amplifier does helps with the accuracy of the input which affects the output's accuracy. However, it's still unable to yield the 100% accuracy due to time of adjusting of the sound amplifier. The way it works is by making the system auto adjust in 1 second time frame. In that time frame, the system will not receive any sound as an input, and later adjust the sound volume threshold to be according to that time frame sound's volume. Meaning, if there is a crowded environment nearby in that 1 second time frame, the threshold will be auto adjusted to 400 for example. The default ideal quiet environment is 300. That means after the system auto adjusted the threshold, the system can operate even in a crowded environment. However, here comes a cause of small issue. The system auto adjusted for 1 second, and then wait to receive command input for another random second (> 1 second) before going back to auto adjust again. If the user coincidently give command in auto adjusting time, the voice command will not be recognized. However, improving the "Hello bot" to be able to operates in the crowded area does increase the possibility in many fields.

4.8 Testing and Evaluation Summary

This chapter introduced 3 add-on functions – ChatGPT, VDO call, and New Voice Assistant Model (same functions as the original but simplified to be 1 step instead of multiple steps for each command) – into "Hello bot". All of which were discussed and shown in code, reason of development, concept, and system flow in section 4.5. In section 4.6, a series of test were described which demonstrated the practicality of the project.

An evaluation was then presented in section 4.7. Section 4.8 is a chapter summary of chapter 4.5 – 4.8. The next chapter will be the conclusion of this thesis. The summary of each chapter up to chapter 5 (original) in addition to section 4.5 – 4.8 that was being added for Capstone 2 will be discussed. In addition, the key points of this thesis will also be presented. Finally, the future work plan of this thesis will be discussed at the end.

CHAPTER 5
CONCLUSION

5.1 Introduction

In this Chapter, the summary of this paper is mention (Section 5.2). Then, conclusions about key points of the work undertaken is describe in section 5.3, and finally in section 5.4 discuss future work, and how NLP, Animation, Python, Video Call, ChatGPT, Speech Recognition, Text to Speech, and SMTP technologies helping support projects such as this one.

5.2 Summary

This is a summary of each chapter intro and summary

Chapter 1 introduced the reason of developing this project. The reasons were clearly stated and displayed according to research about senior's pain point in using technology. Furthermore, the scope of this project also discussed. The requirement of both software and hardware also presented in table.

Chapter 2 reviewed the state-of-the-art in NLP, Animation, SMTP. NLP, Animation, WebRTC, Speech Recognition, Text to Speech, and SMTP were introduced, and described. The potential for those technologies were highlighted and shown in the form of related works in present days.

The comparison between the scope of study, and related work also provided. In addition, the comparison of all functions between related work and "Hello Bot" voice assistant also presented.

Chapter 3 describes the design of development plan and system methodology. Their interesting problems also discussed, and provided with suitable solution. The details provided in both box diagram and written text for explanation in each section.

Chapter 4 is divided in to two parts. Capstone 1 and Capstone 2. Capstone 1 is an individual project which cover section 4.1 – 4.4 developed in the first semester. Capstone 2 is a group project cover section 4.5 -4.8 developed in the second semester. Capstone 1 part described the implementation NLP, Animation, SMTP, Speech

Recognition, Text to Speech, and Python. These technologies' implementation method was described in system work flow box diagram with text explanation. Code implementation also presented and thoroughly described in core concept of each technology. For instance, the process of NLP using code, the process of SMTP using code, the processs of Speech Recognition using code, the process of Text to speech using code, and the process of running custom animation with code. Furthermore, the testing and evaluation of this thesis also discussed in section 4.2 – 4.4.

Chapter 4, section 4.5 – 4.8, described the add-on project for Capstone 2 Group project as well as the reason for adding them. The new technologies present include Voice Amplifier, WebRTC, ChatGPT, and new NLP model. All of them were described in system work flow box diagram with text explanation. Code implementation also presented and thoroughly described in core concept of each technology. For instance, the new process of NLP using code, the process of activating VDO call using code, the process of using ChatGPT as a chat bot using code. Furthermore, the testing and evaluation of this thesis also discussed in section 4.6 – 4.8.

In this chapter 5, all four previous chapters were discussed. Then, the conclusion will be in section 5.3, including key point which linked to this thesis's purpose presented in chapter 1 (Section 1.3). The future work is in section 5.4

5.3 Conclusions

The aim of this project was to create a voice assistant in aiding senior in using technologies such as Google, Youtube, Gmail. Of course, without forgetting to provide solution for senior's general pain points such forgetful (reminder, and take note), and not so good at technology. NLP, Speech Recognition, Text to Speech, and Animation are backbone of this project because they combined as a process that can make machine understand human. For Speech Recognition is to convert human voice into digital form for computer to compute. For NLP is to compute the digital data after Speech Recognition did its job. For Text to Speech is to make a machine voice to communicate with user. Lastly, animation makes suitable user interface for easy-to-use purpose not to mention robot face can make robot becomes more live like creature to increase the quality of interaction between Human and Machine.

In capstone 2 group project, ChatGPT is added to make Chat bot like function, making the “Hello Bot” being not just an assistant but a friend. Moreover, a new voice assistant model is required to work with ChatGPT while keeping the original version so user can choose which funtion is best for themselves. Furthermore, VDO Call funtion is added to provide another mean of communication in addition to Gmail from the original version.

In conclusion, all the functions of this paper system are listed below:

1) Voice commands for using Youtube.
   - Voice command to open application.
   - Voice command to search.
   - Voice command to play a video
2) Voice command for using Google.
   - Voice command to open application.
   - Voice command to search.
3) Voice command for using Gmail.
   - Voice command to send Gmail.
   - Voice command to name the Subject.
   - Voice command to select who to send a Gmail to base on name.
   - Voice command to name the detail of the Gmail.
4) Voice command for reminder system.
   - Voice command to initiate the reminder system.
   - Can choose what to be reminded freely.
   - Can set timer.
   - Work parallelly with main program
5) Voice command for taking note system.
   - Voice command to initiate the take note system.
   - Voice command to write the detail of the note.
6) Voice command to work with new NLP model.
   - Voice command to make note.
   - Voice command to open google.
   - Voice command to open youtube.

- Voice command to open something on youtube.
- Voice command to play a video on youtube.
- Voice command to open something on google.
- Voice command to make a reminder.
- Voice command to make a video call.
- Voice command to turn off the “Hello Bot”.
- Voice command to change mode to original.

7) Voice command for making a video call

- Voice command to initiate
- Voice command to specify a receiver
- Voice command to send an email invitation with a URL to the video call website

8) ChatGPT

- Chat bot

These combined functions created a robot named “Hello Bot”.

In Chapter 1, the general hypothesis is stated that the quality of life for seniors and people in need will improve, developer can apply knowledge of Robotics and AI in real life application, and this thesis “Hello Bot” voice assistant should be easy to use but provides effective result. 10 tests were conducted for each funtions. The result yield 90% success rate with 10% minor founded issue.

Furthermore, in capstone group project, adding new functions made robot more helpful to user with voice amplifier adding more possibility to the robot.

5.3.1 Improvement in the Quality of Life for Senior and People in Need.

The “Hello Bot” is tested and proved to be able to functions without using hand for most of all available functions. Some exceptions those required hands are only scrolling Google page after searched, and turn on and off the machine. Therefore, this technology is appropriate for not only elderly people, but people who has difficulty in using hands as well. Moreover, the improvement of the quality of life is without a doubt as the combination of NLP, Speech Recognition, Text to Speech, and Animation created a live like interaction that prompt user to have interaction with the machine to give input step by step made it easy and effortless for user to use “Hello Bot”. The multiple key words required for activating functions, random machine feedback and unlimited length of each voice input sentences are integrated in this project and helped in the improvement as well.

5.3.2 Applied Knowledge of Robotics and AI by Developer

NLP or Natural Language Processing is a form of an Artificial Intelligence where a developer teaches a machine to understand human language in their own way based on shared state of art discussed in chapter 2. In addition, developer can use the knowledge of coding python in creating “Hello Bot” voice assistant as well as increase more experience in branch knowledge like Animation, SMTP, and Modules just to name a few.

5.3.3 Easy to Use

The combination of NLP, Speech Recognition, Text to Speech, and Animation together created an artificial voice assistant. It creates a voice interaction between human and machine. The code is focused to create the interaction step by step for straightforwardness. Animation helps creating UI which give a visual aid in usability as well.

5.4 Future Work

In semester 1, The plan for future work of this “Hello Bot” voice assistant was to put it in hardware. This semester developing a software while in the next semester (2^{nd} semester) put what was developed into a hardware. The hardware that considered of using was raspberry pi. Some functions will be adjusted or added appropriately, depending on the situation.

However, in second semester got to work as a group, and assigned to install into NUC that running Ubuntu version 18.04 OS instead of a Window OS. Due to many challenges and opportunities made us succesfully developed the “Hello bot” as discussed in previous chapters. Our future plan is to make “Hello Bot” beable to chat without using ChatGPT API because if we can do so, we can customize and give personality to the robot further.

References

No Isolation.(n.d.). *Why do many seniors have trouble using technology*. https://www.noisolation.com/research/why-do-many-seniors-have-trouble-using-technology#:~:text=As%20many%20as%2077%20percent,use%20a%20smartphone%20or%20tablet.

Knowledge Jump. (2012, August 25). *Design Methodologies.* http://www.nwlink.com/~donclark/design/design_models.html

Prachuabmoh, V. (n.d.). *Aging in Thailand*. https://www.un.org/development/desa/pd/sites/www.un.org.development.desa.pd/files/unpd_egm_201902_s3_vipanprachuabmoh.pdf

Pew Research Center. (2017, May 17). *Barriers to adoption and attitudes towards technology*.https://www.pewresearch.org/internet/2017/05/17/barriers-to-adoption-and-attitudes-towards-technology/

Brownlee, J. (2019, August 7). *What is natural language processing.* https://machinelearningmastery.com/natural-language-processing/

Lutkevich, B. (2021, March). *Natural language processing.* https://www.techtarget.com/searchenterpriseai/definition/natural-language-processing-NLP

Koro, P. (2019, November 26). *How to make Simple & Funny & Hilarious DIY animation.*https://www.youtube.com/watch?v=1Xop0scjCWA

Adobe. (2021, December 23). *Use this article to learn about the various components of time in animate*. https://helpx.adobe.com/animate/using/time.html

Philip, A. (2021, October 25). *What's an SMTP port & how to choose the right one.* https://postmarkapp.com/blog/choosing-the-right-smtp-port#smtp-port-587-the-right-choice-for-nearly-every-situation

CT Asia Robotics. (n.d.). *Dinsaw Robot.* https://www.dinsow.com/dinsaw-mini-ai/

Obodroid. (n.d.). *Kaitomm personal assistant.* https://www.obodroid.com/product/personal-assistant/

APPENDICES

APPENDIX A

Code

```
import speech_recognition as sr
import pyttsx3
import time
import pywhatkit
import webbrowser
import os
import smtplib
from email.message import EmailMessage
import subprocess
import datetime
import threading
import random
import pygame
import openai
import playsound
from gtts import gTTS



class imageHandler:
  def __init__ ( self ):
    self.pics = dict()

  def loadFromFile ( self, filename, id=None ):
    if id == None: id = filename
    self.pics [ id ] = pygame.image.load ( filename ).convert()

  def loadFromSurface ( self, surface, id ):
    self.pics [ id ] = surface.convert_alpha()

  def render ( self, surface, id, position = None, clear = False, size
= None ):
    if clear == True:
      surface.fill ( (5,2,23) )

    if position == None:
      picX = int ( surface.get_width() / 2 - self.pics
[ id ].get_width() / 2 )
    else:
      picX = position [0]
      picY = position [1]

    if size == None:
```

```
      surface.blit ( self.pics [ id ], ( picX, picY ) )

    else:
      surface.blit ( pygame.transform.smoothscale ( self.pics [ id ],
size ), ( picX, picY ) )


pygame.display.init
screen = pygame.display.set_mode((1920,1080), pygame.RESIZABLE)
handler = imageHandler()

def speak(text):
    tts = gTTS(text=text, lang="en")
    filename = "voice.mp3"
    tts.save(filename)
    playsound.playsound(filename)

def get_audio():
    r = sr.Recognizer()
    with sr.Microphone() as source:
        r.adjust_for_ambient_noise(source,duration=1)
        audio = r.listen(source, phrase_time_limit=10)
        said = ""

        try:
            said = r.recognize_google(audio)
            print(said)
        except Exception as e:
            print("Exception: "+str(e))
    return said.lower()

def get_audio_auto():
    r = sr.Recognizer()
    with sr.Microphone() as source:
        r.adjust_for_ambient_noise(source,duration=1)
        audio = r.listen(source)
        said = ""

        try:
            said = r.recognize_google(audio)
            print(said)
        except Exception as e:
            print("Exception: " + str(e))
            AUTOCHAT()
    return said.lower()
```

```
def sleepload():
    handler.loadFromFile ( "D://Final Project For Graduation/TestGUserinterface/sleep/sl0.jpg",  "sl0" )
    handler.loadFromFile ( "D://Final Project For Graduation/TestGUserinterface/sleep/sl1.jpg",  "sl1" )
    handler.loadFromFile ( "D://Final Project For Graduation/TestGUserinterface/sleep/sl2.jpg",  "sl2" )
    handler.loadFromFile ( "D://Final Project For Graduation/TestGUserinterface/sleep/sl3.jpg",  "sl3" )
    handler.loadFromFile ( "D://Final Project For Graduation/TestGUserinterface/sleep/sl4.jpg",  "sl4" )
    handler.loadFromFile ( "D://Final Project For Graduation/TestGUserinterface/sleep/sl5.jpg",  "sl5" )
    handler.loadFromFile ( "D://Final Project For Graduation/TestGUserinterface/sleep/sl6.jpg",  "sl6" )

def wakingupload():
    handler.loadFromFile ( "D://Final Project For Graduation/TestGUserinterface/wakingup/wup00.jpg",  "wup0" )
    handler.loadFromFile ( "D://Final Project For Graduation/TestGUserinterface/wakingup/wup01.jpg",  "wup1" )
    handler.loadFromFile ( "D://Final Project For Graduation/TestGUserinterface/wakingup/wup02.jpg",  "wup2" )
    handler.loadFromFile ( "D://Final Project For Graduation/TestGUserinterface/wakingup/wup03.jpg",  "wup3" )
    handler.loadFromFile ( "D://Final Project For Graduation/TestGUserinterface/wakingup/wup04.jpg",  "wup4" )
    handler.loadFromFile ( "D://Final Project For Graduation/TestGUserinterface/wakingup/wup05.jpg",  "wup5" )
    handler.loadFromFile ( "D://Final Project For Graduation/TestGUserinterface/wakingup/wup06.jpg",  "wup6" )
    handler.loadFromFile ( "D://Final Project For Graduation/TestGUserinterface/wakingup/wup07.jpg",  "wup7" )
    handler.loadFromFile ( "D://Final Project For Graduation/TestGUserinterface/wakingup/wup08.jpg",  "wup8" )
    handler.loadFromFile ( "D://Final Project For Graduation/TestGUserinterface/wakingup/wup09.jpg",  "wup9" )
    handler.loadFromFile ( "D://Final Project For Graduation/TestGUserinterface/wakingup/wup10.jpg",  "wup10" )
    handler.loadFromFile ( "D://Final Project For Graduation/TestGUserinterface/wakingup/wup11.jpg",  "wup11" )
    handler.loadFromFile ( "D://Final Project For Graduation/TestGUserinterface/wakingup/wup12.jpg",  "wup12" )
    handler.loadFromFile ( "D://Final Project For Graduation/TestGUserinterface/wakingup/wup13.jpg",  "wup13" )
```

```
    handler.loadFromFile ( "D://Final Project For Graduation/TestGUserinterface/wakingup/wup14.jpg",  "wup14" )

def reminderload():
def recognizeload():
def errorload():
def googleload():

    handler.loadFromFile ( "D://Final Project For Graduation/TestGUserinterface/google/google00.jpg",  "google00" )
    handler.loadFromFile ( "D://Final Project For Graduation/TestGUserinterface/google/google01.jpg",  "google01" )
    handler.loadFromFile ( "D://Final Project For Graduation/TestGUserinterface/google/google02.jpg",  "google02" )
    handler.loadFromFile ( "D://Final Project For Graduation/TestGUserinterface/google/google03.jpg",  "google03" )
    handler.loadFromFile ( "D://Final Project For Graduation/TestGUserinterface/google/google04.jpg",  "google04" )
    handler.loadFromFile ( "D://Final Project For Graduation/TestGUserinterface/google/google05.jpg",  "google05" )
    handler.loadFromFile ( "D://Final Project For Graduation/TestGUserinterface/google/google06.jpg",  "google06" )
    handler.loadFromFile ( "D://Final Project For Graduation/TestGUserinterface/google/google07.jpg",  "google07" )
    handler.loadFromFile ( "D://Final Project For Graduation/TestGUserinterface/google/google08.jpg",  "google08" )
    handler.loadFromFile ( "D://Final Project For Graduation/TestGUserinterface/google/google09.jpg",  "google09" )
    handler.loadFromFile ( "D://Final Project For Graduation/TestGUserinterface/google/google10.jpg",  "google10" )
    handler.loadFromFile ( "D://Final Project For Graduation/TestGUserinterface/google/google11.jpg",  "google11" )
    handler.loadFromFile ( "D://Final Project For Graduation/TestGUserinterface/google/google12.jpg",  "google12" )
    handler.loadFromFile ( "D://Final Project For Graduation/TestGUserinterface/google/google13.jpg",  "google13" )
    handler.loadFromFile ( "D://Final Project For Graduation/TestGUserinterface/google/google14.jpg",  "google14" )
    handler.loadFromFile ( "D://Final Project For Graduation/TestGUserinterface/google/google15.jpg",  "google15" )
    handler.loadFromFile ( "D://Final Project For Graduation/TestGUserinterface/google/google16.jpg",  "google16" )
    handler.loadFromFile ( "D://Final Project For Graduation/TestGUserinterface/google/google17.jpg",  "google17" )
    handler.loadFromFile ( "D://Final Project For Graduation/TestGUserinterface/google/google18.jpg",  "google18" )
```

```
    handler.loadFromFile ( "D://Final Project For Graduation/TestGUserinterface/google/google19.jpg",  "google19" )
    handler.loadFromFile ( "D://Final Project For Graduation/TestGUserinterface/google/google20.jpg",  "google20" )
    handler.loadFromFile ( "D://Final Project For Graduation/TestGUserinterface/google/google21.jpg",  "google21" )
    handler.loadFromFile ( "D://Final Project For Graduation/TestGUserinterface/google/google22.jpg",  "google22" )

def noteload():
    handler.loadFromFile ( "D://Final Project For Graduation/TestGUserinterface/note/note00.jpg",  "note00" )
    handler.loadFromFile ( "D://Final Project For Graduation/TestGUserinterface/note/note01.jpg",  "note01" )
    handler.loadFromFile ( "D://Final Project For Graduation/TestGUserinterface/note/note02.jpg",  "note02" )
    handler.loadFromFile ( "D://Final Project For Graduation/TestGUserinterface/note/note03.jpg",  "note03" )
    handler.loadFromFile ( "D://Final Project For Graduation/TestGUserinterface/note/note04.jpg",  "note04" )
    handler.loadFromFile ( "D://Final Project For Graduation/TestGUserinterface/note/note05.jpg",  "note05" )
    handler.loadFromFile ( "D://Final Project For Graduation/TestGUserinterface/note/note06.jpg",  "note06" )
    handler.loadFromFile ( "D://Final Project For Graduation/TestGUserinterface/note/note07.jpg",  "note07" )
    handler.loadFromFile ( "D://Final Project For Graduation/TestGUserinterface/note/note08.jpg",  "note08" )
    handler.loadFromFile ( "D://Final Project For Graduation/TestGUserinterface/note/note09.jpg",  "note09" )
    handler.loadFromFile ( "D://Final Project For Graduation/TestGUserinterface/note/note10.jpg",  "note10" )
    handler.loadFromFile ( "D://Final Project For Graduation/TestGUserinterface/note/note11.jpg",  "note11" )
    handler.loadFromFile ( "D://Final Project For Graduation/TestGUserinterface/note/note12.jpg",  "note12" )
    handler.loadFromFile ( "D://Final Project For Graduation/TestGUserinterface/note/note13.jpg",  "note13" )
    handler.loadFromFile ( "D://Final Project For Graduation/TestGUserinterface/note/note14.jpg",  "note14" )
    handler.loadFromFile ( "D://Final Project For Graduation/TestGUserinterface/note/note15.jpg",  "note15" )
    handler.loadFromFile ( "D://Final Project For Graduation/TestGUserinterface/note/note16.jpg",  "note16" )
    handler.loadFromFile ( "D://Final Project For Graduation/TestGUserinterface/note/note17.jpg",  "note17" )
```

```
def gmailload():
def ytubeupdisplay():
    A = 0
    B = 0
    x = 1920
    y = 1080

    handler.render ( screen, "ytube45", ( A, B ), True, ( x, y ) )
    pygame.display.update(A,B,x,y)
    time.sleep(.1)
    handler.render ( screen, "ytube46", ( A, B ), True, ( x, y ) )
    pygame.display.update(A,B,x,y)
    time.sleep(.1)
    handler.render ( screen, "ytube47", ( A, B ), True, ( x, y ) )
    pygame.display.update(A,B,x,y)
    time.sleep(.1)
    handler.render ( screen, "ytube48", ( A, B ), True, ( x, y ) )
    pygame.display.update(A,B,x,y)
    time.sleep(.1)
    handler.render ( screen, "ytube49", ( A, B ), True, ( x, y ) )
    pygame.display.update(A,B,x,y)
    time.sleep(.1)
    handler.render ( screen, "ytube50", ( A, B ), True, ( x, y ) )
    pygame.display.update(A,B,x,y)
    time.sleep(.1)
    handler.render ( screen, "ytube51", ( A, B ), True, ( x, y ) )
    pygame.display.update(A,B,x,y)
    time.sleep(.1)
    handler.render ( screen, "ytube52", ( A, B ), True, ( x, y ) )
    pygame.display.update(A,B,x,y)
    time.sleep(.1)
    handler.render ( screen, "ytube53", ( A, B ), True, ( x, y ) )
    pygame.display.update(A,B,x,y)
    time.sleep(.1)
    handler.render ( screen, "ytube54", ( A, B ), True, ( x, y ) )
    pygame.display.update(A,B,x,y)
    time.sleep(.1)
    handler.render ( screen, "ytube55", ( A, B ), True, ( x, y ) )
    pygame.display.update(A,B,x,y)
    time.sleep(.1)
    handler.render ( screen, "ytube56", ( A, B ), True, ( x, y ) )
    pygame.display.update(A,B,x,y)
    time.sleep(.1)
    handler.render ( screen, "ytube57", ( A, B ), True, ( x, y ) )
```

```
    pygame.display.update(A,B,x,y)
    time.sleep(.1)
    handler.render ( screen, "ytube58", ( A, B ), True, ( x, y ) )
    pygame.display.update(A,B,x,y)
    time.sleep(.1)
    handler.render ( screen, "ytube59", ( A, B ), True, ( x, y ) )
    pygame.display.update(A,B,x,y)
    time.sleep(.1)
    handler.render ( screen, "ytube60", ( A, B ), True, ( x, y ) )
    pygame.display.update(A,B,x,y)
    time.sleep(.1)
    handler.render ( screen, "ytube61", ( A, B ), True, ( x, y ) )
    pygame.display.update(A,B,x,y)
    time.sleep(.1)
    handler.render ( screen, "ytube62", ( A, B ), True, ( x, y ) )
    pygame.display.update(A,B,x,y)
    time.sleep(.1)
    handler.render ( screen, "ytube63", ( A, B ), True, ( x, y ) )
    pygame.display.update(A,B,x,y)
    time.sleep(.1)
    handler.render ( screen, "ytube64", ( A, B ), True, ( x, y ) )
    pygame.display.update(A,B,x,y)
    time.sleep(.1)
    handler.render ( screen, "ytube65", ( A, B ), True, ( x, y ) )
    pygame.display.update(A,B,x,y)
    time.sleep(.1)
    handler.render ( screen, "ytube66", ( A, B ), True, ( x, y ) )
    pygame.display.update(A,B,x,y)
    time.sleep(.1)
    handler.render ( screen, "ytube67", ( A, B ), True, ( x, y ) )
    pygame.display.update(A,B,x,y)
    time.sleep(.1)
    handler.render ( screen, "ytube68", ( A, B ), True, ( x, y ) )
    pygame.display.update(A,B,x,y)
    time.sleep(.1)
    handler.render ( screen, "ytube69", ( A, B ), True, ( x, y ) )
    pygame.display.update(A,B,x,y)
    time.sleep(.1)
    handler.render ( screen, "ytube70", ( A, B ), True, ( x, y ) )
    pygame.display.update(A,B,x,y)
    time.sleep(.1)
    handler.render ( screen, "ytube71", ( A, B ), True, ( x, y ) )
    pygame.display.update(A,B,x,y)
    time.sleep(.1)
    handler.render ( screen, "ytube72", ( A, B ), True, ( x, y ) )
    pygame.display.update(A,B,x,y)
```

```
    time.sleep(.1)

def sleepdisplay():
    A = 0
    B = 0
    x = 1920
    y = 1080
    handler.render ( screen, "sl0", ( A, B ), True, ( x, y ) )
    pygame.display.update(A,B,x,y)
    time.sleep(.1)
    handler.render ( screen, "sl1", ( A, B ), True, ( x, y ) )
    pygame.display.update(A,B,x,y)
    time.sleep(.1)
    handler.render ( screen, "sl2", ( A, B ), True, ( x, y ) )
    pygame.display.update(A,B,x,y)
    time.sleep(.1)
    handler.render ( screen, "sl3", ( A, B ), True, ( x, y ) )
    pygame.display.update(A,B,x,y)
    time.sleep(.1)
    handler.render ( screen, "sl4", ( A, B ), True, ( x, y ) )
    pygame.display.update(A,B,x,y)
    time.sleep(.1)
    handler.render ( screen, "sl5", ( A, B ), True, ( x, y ) )
    pygame.display.update(A,B,x,y)
    time.sleep(.1)
    handler.render ( screen, "sl0", ( A, B ), True, ( x, y ) )
    pygame.display.update(A,B,x,y)
    time.sleep(.1)
    handler.render ( screen, "sl1", ( A, B ), True, ( x, y ) )
    pygame.display.update(A,B,x,y)
    time.sleep(.1)
    handler.render ( screen, "sl2", ( A, B ), True, ( x, y ) )
    pygame.display.update(A,B,x,y)
    time.sleep(.1)
    handler.render ( screen, "sl3", ( A, B ), True, ( x, y ) )
    pygame.display.update(A,B,x,y)
    time.sleep(.1)
    handler.render ( screen, "sl4", ( A, B ), True, ( x, y ) )
    pygame.display.update(A,B,x,y)
    time.sleep(.1)
    handler.render ( screen, "sl5", ( A, B ), True, ( x, y ) )
    pygame.display.update(A,B,x,y)
    time.sleep(.1)
    handler.render ( screen, "sl0", ( A, B ), True, ( x, y ) )
    pygame.display.update(A,B,x,y)
    time.sleep(.1)
```

```
    handler.render ( screen, "sl1", ( A, B ), True, ( x, y ) )
    pygame.display.update(A,B,x,y)
    time.sleep(.1)
    handler.render ( screen, "sl2", ( A, B ), True, ( x, y ) )
    pygame.display.update(A,B,x,y)
    time.sleep(.1)
    handler.render ( screen, "sl3", ( A, B ), True, ( x, y ) )
    pygame.display.update(A,B,x,y)
    time.sleep(.1)
    handler.render ( screen, "sl4", ( A, B ), True, ( x, y ) )
    pygame.display.update(A,B,x,y)
    time.sleep(.1)
    handler.render ( screen, "sl5", ( A, B ), True, ( x, y ) )
    pygame.display.update(A,B,x,y)
    time.sleep(.1)
    handler.render ( screen, "sl0", ( A, B ), True, ( x, y ) )
    pygame.display.update(A,B,x,y)
    time.sleep(.1)
    handler.render ( screen, "sl1", ( A, B ), True, ( x, y ) )
    pygame.display.update(A,B,x,y)
    time.sleep(.1)
    handler.render ( screen, "sl2", ( A, B ), True, ( x, y ) )
    pygame.display.update(A,B,x,y)
    time.sleep(.1)
    handler.render ( screen, "sl3", ( A, B ), True, ( x, y ) )
    pygame.display.update(A,B,x,y)
    time.sleep(.1)
    handler.render ( screen, "sl4", ( A, B ), True, ( x, y ) )
    pygame.display.update(A,B,x,y)
    time.sleep(.1)
    handler.render ( screen, "sl5", ( A, B ), True, ( x, y ) )
    pygame.display.update(A,B,x,y)
    time.sleep(.1)
    handler.render ( screen, "sl0", ( A, B ), True, ( x, y ) )
    pygame.display.update(A,B,x,y)
    time.sleep(.1)
    handler.render ( screen, "sl1", ( A, B ), True, ( x, y ) )
    pygame.display.update(A,B,x,y)
    time.sleep(.1)
    handler.render ( screen, "sl2", ( A, B ), True, ( x, y ) )
    pygame.display.update(A,B,x,y)
    time.sleep(.1)
    handler.render ( screen, "sl3", ( A, B ), True, ( x, y ) )
    pygame.display.update(A,B,x,y)
    time.sleep(.1)
    handler.render ( screen, "sl4", ( A, B ), True, ( x, y ) )
```

```
    pygame.display.update(A,B,x,y)
    time.sleep(.1)
    handler.render ( screen, "sl5", ( A, B ), True, ( x, y ) )
    pygame.display.update(A,B,x,y)
    time.sleep(.1)
    handler.render ( screen, "sl6", ( A, B ), True, ( x, y ) )
    pygame.display.update(A,B,x,y)
    time.sleep(.1)

def wakingdisplay():
    A = 0
    B = 0
    x = 1920
    y = 1080
    handler.render ( screen, "wup0", ( A, B ), True, ( x, y ) )
    pygame.display.update(A,B,x,y)
    time.sleep(.1)
    handler.render ( screen, "wup1", ( A, B ), True, ( x, y ) )
    pygame.display.update(A,B,x,y)
    time.sleep(.1)
    handler.render ( screen, "wup2", ( A, B ), True, ( x, y ) )
    pygame.display.update(A,B,x,y)
    time.sleep(.1)
    handler.render ( screen, "wup3", ( A, B ), True, ( x, y ) )
    pygame.display.update(A,B,x,y)
    time.sleep(.1)
    handler.render ( screen, "wup4", ( A, B ), True, ( x, y ) )
    pygame.display.update(A,B,x,y)
    time.sleep(.1)
    handler.render ( screen, "wup5", ( A, B ), True, ( x, y ) )
    pygame.display.update(A,B,x,y)
    time.sleep(.1)
    handler.render ( screen, "wup6", ( A, B ), True, ( x, y ) )
    pygame.display.update(A,B,x,y)
    time.sleep(.1)
    handler.render ( screen, "wup7", ( A, B ), True, ( x, y ) )
    pygame.display.update(A,B,x,y)
    time.sleep(.1)
    handler.render ( screen, "wup8", ( A, B ), True, ( x, y ) )
    pygame.display.update(A,B,x,y)
    time.sleep(.1)
    handler.render ( screen, "wup9", ( A, B ), True, ( x, y ) )
    pygame.display.update(A,B,x,y)
    time.sleep(.1)
    handler.render ( screen, "wup10", ( A, B ), True, ( x, y ) )
    pygame.display.update(A,B,x,y)
```

```
    time.sleep(.1)
    handler.render ( screen, "wup11", ( A, B ), True, ( x, y ) )
    pygame.display.update(A,B,x,y)
    time.sleep(.1)
    handler.render ( screen, "wup12", ( A, B ), True, ( x, y ) )
    pygame.display.update(A,B,x,y)
    time.sleep(.1)
    handler.render ( screen, "wup13", ( A, B ), True, ( x, y ) )
    pygame.display.update(A,B,x,y)
    time.sleep(.1)
    handler.render ( screen, "wup14", ( A, B ), True, ( x, y ) )
    pygame.display.update(A,B,x,y)
    time.sleep(.1)

def reminderdisplay():
    A = 0
    B = 0
    x = 1920
    y = 1080
    handler.render ( screen, "reminder00", ( A, B ), True, ( x, y ) )
    pygame.display.update(A,B,x,y)
    time.sleep(.1)
    handler.render ( screen, "reminder01", ( A, B ), True, ( x, y ) )
    pygame.display.update(A,B,x,y)
    time.sleep(.1)
    handler.render ( screen, "reminder02", ( A, B ), True, ( x, y ) )
    pygame.display.update(A,B,x,y)
    time.sleep(.1)
    handler.render ( screen, "reminder03", ( A, B ), True, ( x, y ) )
    pygame.display.update(A,B,x,y)
    time.sleep(.1)
    handler.render ( screen, "reminder04", ( A, B ), True, ( x, y ) )
    pygame.display.update(A,B,x,y)
    time.sleep(.1)
    handler.render ( screen, "reminder05", ( A, B ), True, ( x, y ) )
    pygame.display.update(A,B,x,y)
    time.sleep(.1)
    handler.render ( screen, "reminder06", ( A, B ), True, ( x, y ) )
    pygame.display.update(A,B,x,y)
    time.sleep(.1)
    handler.render ( screen, "reminder07", ( A, B ), True, ( x, y ) )
    pygame.display.update(A,B,x,y)
    time.sleep(.1)
    handler.render ( screen, "reminder08", ( A, B ), True, ( x, y ) )
    pygame.display.update(A,B,x,y)
    time.sleep(.1)
```

```
    handler.render ( screen, "reminder09", ( A, B ), True, ( x, y ) )
    pygame.display.update(A,B,x,y)
    time.sleep(.1)
    handler.render ( screen, "reminder10", ( A, B ), True, ( x, y ) )
    pygame.display.update(A,B,x,y)
    time.sleep(.1)
    handler.render ( screen, "reminder11", ( A, B ), True, ( x, y ) )
    pygame.display.update(A,B,x,y)
    time.sleep(.1)
    handler.render ( screen, "reminder12", ( A, B ), True, ( x, y ) )
    pygame.display.update(A,B,x,y)
    time.sleep(.1)
    handler.render ( screen, "reminder13", ( A, B ), True, ( x, y ) )
    pygame.display.update(A,B,x,y)
    time.sleep(.1)
    handler.render ( screen, "reminder14", ( A, B ), True, ( x, y ) )
    pygame.display.update(A,B,x,y)
    time.sleep(.1)
    handler.render ( screen, "reminder15", ( A, B ), True, ( x, y ) )
    pygame.display.update(A,B,x,y)
    time.sleep(.1)
    handler.render ( screen, "reminder16", ( A, B ), True, ( x, y ) )
    pygame.display.update(A,B,x,y)
    time.sleep(.1)

def recognizedisplay():
    A = 0
    B = 0
    x = 1920
    y = 1080
    handler.render ( screen, "rn00", ( A, B ), True, ( x, y ) )
    pygame.display.update(A,B,x,y)
    time.sleep(.1)
    handler.render ( screen, "rn01", ( A, B ), True, ( x, y ) )
    pygame.display.update(A,B,x,y)
    time.sleep(.1)
    handler.render ( screen, "rn02", ( A, B ), True, ( x, y ) )
    pygame.display.update(A,B,x,y)
    time.sleep(.1)
    handler.render ( screen, "rn03", ( A, B ), True, ( x, y ) )
    pygame.display.update(A,B,x,y)
    time.sleep(.1)
    handler.render ( screen, "rn04", ( A, B ), True, ( x, y ) )
    pygame.display.update(A,B,x,y)
    time.sleep(.1)
    handler.render ( screen, "rn05", ( A, B ), True, ( x, y ) )
```

```
    pygame.display.update(A,B,x,y)
    time.sleep(.1)
    handler.render ( screen, "rn06", ( A, B ), True, ( x, y ) )
    pygame.display.update(A,B,x,y)
    time.sleep(.1)
    handler.render ( screen, "rn07", ( A, B ), True, ( x, y ) )
    pygame.display.update(A,B,x,y)
    time.sleep(.1)
    handler.render ( screen, "rn08", ( A, B ), True, ( x, y ) )
    pygame.display.update(A,B,x,y)
    time.sleep(.1)
    handler.render ( screen, "rn09", ( A, B ), True, ( x, y ) )
    pygame.display.update(A,B,x,y)
    time.sleep(.1)
    handler.render ( screen, "rn10", ( A, B ), True, ( x, y ) )
    pygame.display.update(A,B,x,y)
    time.sleep(.1)
    handler.render ( screen, "rn11", ( A, B ), True, ( x, y ) )
    pygame.display.update(A,B,x,y)
    time.sleep(.1)
    handler.render ( screen, "rn12", ( A, B ), True, ( x, y ) )
    pygame.display.update(A,B,x,y)
    time.sleep(.1)
    handler.render ( screen, "rn13", ( A, B ), True, ( x, y ) )
    pygame.display.update(A,B,x,y)
    time.sleep(.1)
    handler.render ( screen, "rn14", ( A, B ), True, ( x, y ) )
    pygame.display.update(A,B,x,y)
    time.sleep(.1)
    handler.render ( screen, "rn15", ( A, B ), True, ( x, y ) )
    pygame.display.update(A,B,x,y)
    time.sleep(.1)

def inrecogdisplay():
    A = 0
    B = 0
    x = 1920
    y = 1080
    handler.render ( screen, "rn00", ( A, B ), True, ( x, y ) )
    pygame.display.update(A,B,x,y)
    time.sleep(.1)
    handler.render ( screen, "rn01", ( A, B ), True, ( x, y ) )
    pygame.display.update(A,B,x,y)
    time.sleep(.1)
    handler.render ( screen, "rn02", ( A, B ), True, ( x, y ) )
    pygame.display.update(A,B,x,y)
```

```
    time.sleep(.1)
    handler.render ( screen, "rn03", ( A, B ), True, ( x, y ) )
    pygame.display.update(A,B,x,y)
    time.sleep(.1)
    handler.render ( screen, "rn04", ( A, B ), True, ( x, y ) )
    pygame.display.update(A,B,x,y)
    time.sleep(.1)
    handler.render ( screen, "rn05", ( A, B ), True, ( x, y ) )
    pygame.display.update(A,B,x,y)
    time.sleep(.1)

def outrecogdisplay():
    A = 0
    B = 0
    x = 1920
    y = 1080
    handler.render ( screen, "rn05", ( A, B ), True, ( x, y ) )
    pygame.display.update(A,B,x,y)
    time.sleep(.1)
    handler.render ( screen, "rn04", ( A, B ), True, ( x, y ) )
    pygame.display.update(A,B,x,y)
    time.sleep(.1)
    handler.render ( screen, "rn03", ( A, B ), True, ( x, y ) )
    pygame.display.update(A,B,x,y)
    time.sleep(.1)
    handler.render ( screen, "rn02", ( A, B ), True, ( x, y ) )
    pygame.display.update(A,B,x,y)
    time.sleep(.1)
    handler.render ( screen, "rn01", ( A, B ), True, ( x, y ) )
    pygame.display.update(A,B,x,y)
    time.sleep(.1)
    handler.render ( screen, "rn00", ( A, B ), True, ( x, y ) )
    pygame.display.update(A,B,x,y)
    time.sleep(.1)

def errordisplay():
    A = 0
    B = 0
    x = 1920
    y = 1080
    handler.render ( screen, "er0", ( A, B ), True, ( x, y ) )
    pygame.display.update(A,B,x,y)
    time.sleep(.1)
    handler.render ( screen, "er1", ( A, B ), True, ( x, y ) )
    pygame.display.update(A,B,x,y)
    time.sleep(.1)
```

```
    handler.render ( screen, "er2", ( A, B ), True, ( x, y ) )
    pygame.display.update(A,B,x,y)
    time.sleep(.1)
    handler.render ( screen, "er3", ( A, B ), True, ( x, y ) )
    pygame.display.update(A,B,x,y)
    time.sleep(.1)

def googledisplay():
    A = 0
    B = 0
    x = 1920
    y = 1080
    handler.render ( screen, "google00", ( A, B ), True, ( x, y ) )
    pygame.display.update(A,B,x,y)
    time.sleep(.1)
    handler.render ( screen, "google01", ( A, B ), True, ( x, y ) )
    pygame.display.update(A,B,x,y)
    time.sleep(.1)
    handler.render ( screen, "google02", ( A, B ), True, ( x, y ) )
    pygame.display.update(A,B,x,y)
    time.sleep(.1)
    handler.render ( screen, "google03", ( A, B ), True, ( x, y ) )
    pygame.display.update(A,B,x,y)
    time.sleep(.1)
    handler.render ( screen, "google04", ( A, B ), True, ( x, y ) )
    pygame.display.update(A,B,x,y)
    time.sleep(.1)
    handler.render ( screen, "google05", ( A, B ), True, ( x, y ) )
    pygame.display.update(A,B,x,y)
    time.sleep(.1)
    handler.render ( screen, "google06", ( A, B ), True, ( x, y ) )
    pygame.display.update(A,B,x,y)
    time.sleep(.1)
    handler.render ( screen, "google07", ( A, B ), True, ( x, y ) )
    pygame.display.update(A,B,x,y)
    time.sleep(.1)
    handler.render ( screen, "google08", ( A, B ), True, ( x, y ) )
    pygame.display.update(A,B,x,y)
    time.sleep(.1)
    handler.render ( screen, "google09", ( A, B ), True, ( x, y ) )
    pygame.display.update(A,B,x,y)
    time.sleep(.1)
    handler.render ( screen, "google10", ( A, B ), True, ( x, y ) )
    pygame.display.update(A,B,x,y)
    time.sleep(.1)
    handler.render ( screen, "google11", ( A, B ), True, ( x, y ) )
```

```
    pygame.display.update(A,B,x,y)
    time.sleep(.1)
    handler.render ( screen, "google12", ( A, B ), True, ( x, y ) )
    pygame.display.update(A,B,x,y)
    time.sleep(.1)
    handler.render ( screen, "google13", ( A, B ), True, ( x, y ) )
    pygame.display.update(A,B,x,y)
    time.sleep(.1)
    handler.render ( screen, "google14", ( A, B ), True, ( x, y ) )
    pygame.display.update(A,B,x,y)
    time.sleep(.1)
    handler.render ( screen, "google15", ( A, B ), True, ( x, y ) )
    pygame.display.update(A,B,x,y)
    time.sleep(.1)
    handler.render ( screen, "google16", ( A, B ), True, ( x, y ) )
    pygame.display.update(A,B,x,y)
    time.sleep(.1)
    handler.render ( screen, "google17", ( A, B ), True, ( x, y ) )
    pygame.display.update(A,B,x,y)
    time.sleep(.1)
    handler.render ( screen, "google18", ( A, B ), True, ( x, y ) )
    pygame.display.update(A,B,x,y)
    time.sleep(.1)
    handler.render ( screen, "google19", ( A, B ), True, ( x, y ) )
    pygame.display.update(A,B,x,y)
    time.sleep(.1)
    handler.render ( screen, "google20", ( A, B ), True, ( x, y ) )
    pygame.display.update(A,B,x,y)
    time.sleep(.1)
    handler.render ( screen, "google21", ( A, B ), True, ( x, y ) )
    pygame.display.update(A,B,x,y)
    time.sleep(.1)
    handler.render ( screen, "google22", ( A, B ), True, ( x, y ) )
    pygame.display.update(A,B,x,y)
    time.sleep(.1)

def notedisplay():
    A = 0
    B = 0
    x = 1920
    y = 1080
    handler.render ( screen, "note00", ( A, B ), True, ( x, y ) )
    pygame.display.update(A,B,x,y)
    time.sleep(.1)
    handler.render ( screen, "note01", ( A, B ), True, ( x, y ) )
    pygame.display.update(A,B,x,y)
```

```
    time.sleep(.1)
    handler.render ( screen, "note02", ( A, B ), True, ( x, y ) )
    pygame.display.update(A,B,x,y)
    time.sleep(.1)
    handler.render ( screen, "note03", ( A, B ), True, ( x, y ) )
    pygame.display.update(A,B,x,y)
    time.sleep(.1)
    handler.render ( screen, "note04", ( A, B ), True, ( x, y ) )
    pygame.display.update(A,B,x,y)
    time.sleep(.1)
    handler.render ( screen, "note05", ( A, B ), True, ( x, y ) )
    pygame.display.update(A,B,x,y)
    time.sleep(.1)
    handler.render ( screen, "note06", ( A, B ), True, ( x, y ) )
    pygame.display.update(A,B,x,y)
    time.sleep(.1)
    handler.render ( screen, "note07", ( A, B ), True, ( x, y ) )
    pygame.display.update(A,B,x,y)
    time.sleep(.1)
    handler.render ( screen, "note08", ( A, B ), True, ( x, y ) )
    pygame.display.update(A,B,x,y)
    time.sleep(.1)
    handler.render ( screen, "note09", ( A, B ), True, ( x, y ) )
    pygame.display.update(A,B,x,y)
    time.sleep(.1)
    handler.render ( screen, "note10", ( A, B ), True, ( x, y ) )
    pygame.display.update(A,B,x,y)
    time.sleep(.1)
    handler.render ( screen, "note11", ( A, B ), True, ( x, y ) )
    pygame.display.update(A,B,x,y)
    time.sleep(.1)
    handler.render ( screen, "note12", ( A, B ), True, ( x, y ) )
    pygame.display.update(A,B,x,y)
    time.sleep(.1)
    handler.render ( screen, "note13", ( A, B ), True, ( x, y ) )
    pygame.display.update(A,B,x,y)
    time.sleep(.1)
    handler.render ( screen, "note14", ( A, B ), True, ( x, y ) )
    pygame.display.update(A,B,x,y)
    time.sleep(.1)
    handler.render ( screen, "note15", ( A, B ), True, ( x, y ) )
    pygame.display.update(A,B,x,y)
    time.sleep(.1)
    handler.render ( screen, "note16", ( A, B ), True, ( x, y ) )
    pygame.display.update(A,B,x,y)
    time.sleep(.1)
```

```
def donenotedisplay():
    A = 0
    B = 0
    x = 1920
    y = 1080
    handler.render ( screen, "note17", ( A, B ), True, ( x, y ) )
    pygame.display.update(A,B,x,y)
    time.sleep(.1)
    handler.render ( screen, "note13", ( A, B ), True, ( x, y ) )
    pygame.display.update(A,B,x,y)
    time.sleep(.1)
    handler.render ( screen, "note17", ( A, B ), True, ( x, y ) )
    pygame.display.update(A,B,x,y)
    time.sleep(.1)
    handler.render ( screen, "note13", ( A, B ), True, ( x, y ) )
    pygame.display.update(A,B,x,y)
    time.sleep(.1)
    handler.render ( screen, "note17", ( A, B ), True, ( x, y ) )
    pygame.display.update(A,B,x,y)
    time.sleep(.1)
    handler.render ( screen, "note13", ( A, B ), True, ( x, y ) )
    pygame.display.update(A,B,x,y)
    time.sleep(.1)
    handler.render ( screen, "note17", ( A, B ), True, ( x, y ) )
    pygame.display.update(A,B,x,y)
    time.sleep(.1)
    handler.render ( screen, "note13", ( A, B ), True, ( x, y ) )
    pygame.display.update(A,B,x,y)
    time.sleep(.1)
    handler.render ( screen, "note17", ( A, B ), True, ( x, y ) )
    pygame.display.update(A,B,x,y)
    time.sleep(.1)

def gmaildisplay():
    A = 0
    B = 0
    x = 1920
    y = 1080
    handler.render ( screen, "gmail00", ( A, B ), True, ( x, y ) )
    pygame.display.update(A,B,x,y)
    time.sleep(.1)
    handler.render ( screen, "gmail01", ( A, B ), True, ( x, y ) )
    pygame.display.update(A,B,x,y)
    time.sleep(.1)
    handler.render ( screen, "gmail02", ( A, B ), True, ( x, y ) )
```

```
    pygame.display.update(A,B,x,y)
    time.sleep(.1)
    handler.render ( screen, "gmail03", ( A, B ), True, ( x, y ) )
    pygame.display.update(A,B,x,y)
    time.sleep(.1)
    handler.render ( screen, "gmail04", ( A, B ), True, ( x, y ) )
    pygame.display.update(A,B,x,y)
    time.sleep(.1)
    handler.render ( screen, "gmail05", ( A, B ), True, ( x, y ) )
    pygame.display.update(A,B,x,y)
    time.sleep(.1)
    handler.render ( screen, "gmail06", ( A, B ), True, ( x, y ) )
    pygame.display.update(A,B,x,y)
    time.sleep(.1)
    handler.render ( screen, "gmail07", ( A, B ), True, ( x, y ) )
    pygame.display.update(A,B,x,y)
    time.sleep(.1)
    handler.render ( screen, "gmail08", ( A, B ), True, ( x, y ) )
    pygame.display.update(A,B,x,y)
    time.sleep(.1)
    handler.render ( screen, "gmail09", ( A, B ), True, ( x, y ) )
    pygame.display.update(A,B,x,y)
    time.sleep(.1)
    handler.render ( screen, "gmail10", ( A, B ), True, ( x, y ) )
    pygame.display.update(A,B,x,y)
    time.sleep(.1)
    handler.render ( screen, "gmail11", ( A, B ), True, ( x, y ) )
    pygame.display.update(A,B,x,y)
    time.sleep(.1)
    handler.render ( screen, "gmail12", ( A, B ), True, ( x, y ) )
    pygame.display.update(A,B,x,y)
    time.sleep(.1)
    handler.render ( screen, "gmail13", ( A, B ), True, ( x, y ) )
    pygame.display.update(A,B,x,y)
    time.sleep(.1)
    handler.render ( screen, "gmail14", ( A, B ), True, ( x, y ) )
    pygame.display.update(A,B,x,y)
    time.sleep(.1)
    handler.render ( screen, "gmail15", ( A, B ), True, ( x, y ) )
    pygame.display.update(A,B,x,y)
    time.sleep(.1)
    handler.render ( screen, "gmail16", ( A, B ), True, ( x, y ) )
    pygame.display.update(A,B,x,y)
    time.sleep(.1)
    handler.render ( screen, "gmail17", ( A, B ), True, ( x, y ) )
    pygame.display.update(A,B,x,y)
```

```
    time.sleep(.1)
    handler.render ( screen, "gmail18", ( A, B ), True, ( x, y ) )
    pygame.display.update(A,B,x,y)
    time.sleep(.1)
    handler.render ( screen, "gmail19", ( A, B ), True, ( x, y ) )
    pygame.display.update(A,B,x,y)
    time.sleep(.1)
    handler.render ( screen, "gmail20", ( A, B ), True, ( x, y ) )
    pygame.display.update(A,B,x,y)
    time.sleep(.1)
    handler.render ( screen, "gmail21", ( A, B ), True, ( x, y ) )
    pygame.display.update(A,B,x,y)
    time.sleep(.1)
    handler.render ( screen, "gmail22", ( A, B ), True, ( x, y ) )
    pygame.display.update(A,B,x,y)
    time.sleep(.1)

def backtosleepdisplay():
    A = 0
    B = 0
    x = 1920
    y = 1080
    handler.render ( screen, "wup14", ( A, B ), True, ( x, y ) )
    pygame.display.update(A,B,x,y)
    time.sleep(.1)
    handler.render ( screen, "wup13", ( A, B ), True, ( x, y ) )
    pygame.display.update(A,B,x,y)
    time.sleep(.1)
    handler.render ( screen, "wup12", ( A, B ), True, ( x, y ) )
    pygame.display.update(A,B,x,y)
    time.sleep(.1)
    handler.render ( screen, "wup11", ( A, B ), True, ( x, y ) )
    pygame.display.update(A,B,x,y)
    time.sleep(.1)
    handler.render ( screen, "wup10", ( A, B ), True, ( x, y ) )
    pygame.display.update(A,B,x,y)
    time.sleep(.1)
    handler.render ( screen, "wup9", ( A, B ), True, ( x, y ) )
    pygame.display.update(A,B,x,y)
    time.sleep(.1)
    handler.render ( screen, "wup8", ( A, B ), True, ( x, y ) )
    pygame.display.update(A,B,x,y)
    time.sleep(.1)
    handler.render ( screen, "wup7", ( A, B ), True, ( x, y ) )
    pygame.display.update(A,B,x,y)
    time.sleep(.1)
```

```
    handler.render ( screen, "wup6", ( A, B ), True, ( x, y ) )
    pygame.display.update(A,B,x,y)
    time.sleep(.1)
    handler.render ( screen, "wup5", ( A, B ), True, ( x, y ) )
    pygame.display.update(A,B,x,y)
    time.sleep(.1)
    handler.render ( screen, "wup4", ( A, B ), True, ( x, y ) )
    pygame.display.update(A,B,x,y)
    time.sleep(.1)
    handler.render ( screen, "wup3", ( A, B ), True, ( x, y ) )
    pygame.display.update(A,B,x,y)
    time.sleep(.1)
    handler.render ( screen, "wup2", ( A, B ), True, ( x, y ) )
    pygame.display.update(A,B,x,y)
    time.sleep(.1)
    handler.render ( screen, "wup1", ( A, B ), True, ( x, y ) )
    pygame.display.update(A,B,x,y)
    time.sleep(.1)
    handler.render ( screen, "wup0", ( A, B ), True, ( x, y ) )
    pygame.display.update(A,B,x,y)
    time.sleep(.1)

def blackout():
def search_google():
    speak('What do you want to search for?')
    search_target = get_audio()
    url = 'https://google.com/search?q=' + search_target
    webbrowser.get().open(url)
    speak('Here is what I found for' + search_target)

def search_youtube():
    speak('What do you want to search for?')
    y_search_target = get_audio()
    url = 'https://youtube.com/results?search_query=' + y_search_target
    webbrowser.get().open(url)
    speak('Here is what I found for' + y_search_target)

def send_email(receiver, subject, message):
    server = smtplib.SMTP('smtp.gmail.com', 587)
    server.starttls()
    server.login('fachai62011272@gmail.com', 'xspdfroyagjneanv')
    email = EmailMessage()
    email['From'] = 'fachai62011272@gmail.com'
    email['To'] = receiver
    email['Subject'] = subject
    email.set_content(message)
```

```
    server.send_message(email)

email_list = {
    "my son" : '62011272@kmitl.ac.th',
    "daughter": 'fachaijp@gmail.com'
}

def get_email_info():
    speak('To whom you want to send email')
    name = get_audio()
    try:
        receiver = email_list[name]
        print(receiver)
        speak('what is the subject of your email?')
        subject = get_audio()
        speak('Tell me the text in your email')
        message = get_audio()
        print("Send to" + receiver)
        print("The subject is " + subject)
        print("The detail is as follow " + message)
        speak("the mail detail will be as shown")
        speak("yes or no ?")
        text = get_audio()
        mail_yes = ['yes','ok','correct','confirm']
        for phrase in mail_yes:
            if phrase in text:
                speak('sending')
                send_email(receiver, subject, message)
                speak("I just sent the email to" + name)
        mail_no = ['no','don/t','cancel']
        for phrase in mail_no:
            if phrase in text:
                speak('as you wish')
                speak('mail now cancel')
    except:
        errorload()
        errordisplay()
        speak("Error the person name" + name + "is not in the system")
        speak("please try again")
        get_email_info()

def note(text):
    date = datetime.datetime.now()
    file_name = str(date).replace(':',"-") + '-note.txt'
    with open(file_name, "w") as f:
        f.write(text)
```

```
    subprocess.Popen(["notepad.exe", file_name])

time_list = {
    'one minute' : 60,
    '5 minutes' : 300,
}

def help_remind():
    speak("what is it? What do you want me to remember")
    lazyy = get_audio()
    try:
        speak("when should I remind you")
        time_remind = get_audio()
        time_sleep = time_list[time_remind]
        print(time_sleep)
        speak("I will remind you that   " + lazyy + "in    " +
time_remind)
        time.sleep(time_sleep)
        speak("master you asked me to remind you to    " + lazyy)
    except Exception as e:
        print("Exception: " + str(e))
        errorload()
        errordisplay()
        speak('please say again')
        help_remind()
def invite_video_call():
    speak('user')
    speak('hey to Who do you want to video call')
    name = get_audio_auto()
    speak("i understand")
    try:
        receiver = email_list[name]
        subject = 'Hello Bot - Video call invitation'
        message = f'Please open {"https://4c69-171-97-128-
207.ap.ngrok.io"} and follow the below instructions to join.\n\n'
        message += 'Instruction: \n'
        message += '\t1. Open the link in browser\n'
        message += "\t2. Click the 'Visit Site' blue button, if there
is any.\n"
        message += "\t3. Click the 'xxx.html' url, if there is any.\n"
        message += "\t4. Enther the meeting code, click 'Join'.\n"
        message += '\t5. To exit the call, simply close the browser
tab.\n\n\n'
        message += '-----------------------------------------------
-\n'
```

```
        message += 'This message is auto generated and sent by ...
team.'

        send_email(receiver, subject, message)
        speak('Video call invitation is sent to ' + receiver)
        speak('Opening video call website')
        webbrowser.get().open("https://4c69-171-97-128-
207.ap.ngrok.io")
        speak('Click create room to create room and join')

    except:
        errorload()
        errordisplay()
        speak('Error the person name' + name + 'is not in the system')
        speak('please try again')

def greeting():
    hour = int(datetime.datetime.now().hour)
    if hour >= 0 and hour <12:
        speak("Good morning master")
    elif hour >= 12 and hour <18:
        speak("Good afternoon master")
    else:
        speak("Good evening master")
openai.api_key = "skdpVAdgdgdfolktdfgdgkkjhbkFJYBUPkuiipoipiipyu"

WAKE = 'hello'
welcome_speech=['master please say hello when you required my
service','my name is hello','I am built to serve']
Recognized = ['yes sir','what do you want me to do','what may I
help','I am glad you called me']

def Hello_bot():

    while True:
        print("Listening")
        text = get_audio()

        if text.count(WAKE) > 0:
            recognizeload()
            inrecogdisplay()
            x = random.choice(Recognized)
            speak(x)
            text = get_audio()
            recognizeload()
            outrecogdisplay()
```

```
        YT_STRS = ["play"]
        for phrase in YT_STRS:
            if phrase in text:
                song = text.replace(phrase, '')
                speak('playing' + song)
                pywhatkit.playonyt(song)
                Hello_bot()


        GOOGLE_STRS = ["search google"]
        for phrase in GOOGLE_STRS:
            if phrase in text:
                search_google()
                Hello_bot()

        YOUTUBE_STRS = ["search youtube","music"]
        for phrase in YOUTUBE_STRS:
            if phrase in text:
                search_youtube()
                Hello_bot()


        OP_YOUTUBE = ['open youtube']
        for phrase in OP_YOUTUBE:
            if phrase in text:
                speak('opening youtube please wait')
                ytubeupload()
                ytubeupdisplay()
                url = 'https://youtube.com'
                webbrowser.get().open(url)
                speak('youtube now open')
                speak('do you want to search anything?')
                text = get_audio()
                Yans = ['yes', 'i want to','search']
                for phrase in Yans:
                    if phrase in text:
                        search_youtube()
                        Hello_bot()

                Nans = ['no','i don\'t want']
                for phrase in Nans:
                    if phrase in text:
                        speak('as you wish')
                        Hello_bot()
```

```
        OP_GOOGLE = ['open google']
        for phrase in OP_GOOGLE:
            if phrase in text:
                speak('opening google please wait')
                googleload()
                googledisplay()
                url = 'https://google.com'
                webbrowser.get().open(url)
                speak('google now open')
                speak('do you want to search anything?')
                text = get_audio()
                YGans = ['yes', 'i want to','search']
                for phrase in YGans:
                    if phrase in text:
                        search_google()
                        Hello_bot()

                NGans = ['no','i don\'t want']
                for phrase in NGans:
                    if phrase in text:
                        speak('as you wish')
                        Hello_bot()

        EMAIL_STRS = ["email","gmail"]
        for phrase in EMAIL_STRS:
            if phrase in text:
                gmailload()
                gmaildisplay()
                get_email_info()
                Hello_bot()


        NOTE_STRS = ["make a note",'note this']
        for phrase in NOTE_STRS:
            if phrase in text:
                noteload()
                notedisplay()
                speak("What do you want me to note?")
                note_text = get_audio().lower()
                note(note_text)
                noteload()
                donenotedisplay()
```

```
                    speak("I've made a note")

                    Hello_bot()

            TIME_STRS = ["what time",'current time']
            for phrase in TIME_STRS:
                if phrase in text:
                    time = datetime.datetime.now().strftime('%I:%M %p')
                    print(time)
                    speak('current time is' + time)
                    Hello_bot()

            REMEMBER_STRS = ['remember','remind']
            for phrase in REMEMBER_STRS:
                if phrase in text:
                    reminderload()
                    reminderdisplay()
                    thread_help_remind =
threading.Thread(target=help_remind)
                    thread_help_remind.start()
                    Hello_bot()

            SELF_STRS = ["who are you"]
            for phrase in SELF_STRS:
                if phrase in text:
                    speak('I am a robot')
                    speak("I am made to serve my master")
                    wakingupload()
                    backtosleepdisplay()
                    Hello_bot()

            else:
                errorload()
                errordisplay()
                speak('Sorry please say again')
                gmailload()
                blackout()



def AUTOCHAT():
    conversation = ""
    user_name = "user"
    bot_name = "Jiraff"
    i = 0
```

```
    while True:
        print("\n Listening...")
        text = get_audio()
        recognizeload()
        inrecogdisplay()

        CHANGE_TO_DO_STR = ['switch mode', "change mode",'switch
more','change more','change to another mode','switch to another
mode','switch to another more','change to another more']
        for phrase in CHANGE_TO_DO_STR:
            if phrase in text:
                speak("ok")
                speak("switching to do as command mode")
                Hello_bot()

        YT_STRS = ["play"]
        for phrase in YT_STRS:
          if phrase in text:
            song = text.replace(phrase, '')
            LAST_CHECK = ["on youtube","in youtube","at youtube"]
            for phrase in LAST_CHECK:
               if phrase in song:
                  song = text.replace(phrase,'')
                  speak('playing' + song)
                  pywhatkit.playonyt(song)
                  speak('go to sleep mode')
                  sleep_wake()

        OPEN_ALL_STRS = ["open","use","help me
find","search","show","make a search"]
        for phrase in OPEN_ALL_STRS:
            if phrase in text:
               google_search_text = text.replace(phrase,'')
               GOOGLE_CHECK = ["on google","in google"]
               for phrase in GOOGLE_CHECK:
                  if phrase in google_search_text:
                     googleload()
                     googledisplay()
                     speak("opening on google")
                     search_target = google_search_text.replace(phrase,
'')
                     url = 'https://google.com/search?q=' +
search_target
                     webbrowser.get().open(url)
                     speak("Here is what I found")
                     AUTOCHAT()
```

```
                YOUTUBE_CHECK = ["on youtube","in youtube"]
                for phrase in YOUTUBE_CHECK:
                    if phrase in google_search_text:
                        ytubeupload()
                        ytubeupdisplay()
                        speak("opening on youtube")
                        search_target = google_search_text.replace(phrase,
'')
                        url = 'https://youtube.com/search?q=' +
search_target
                        webbrowser.get().open(url)
                        speak("Here is what I found")
                        AUTOCHAT()


        OPEN_YOUTUBE_STRS = ["open youtube","use youtube"]
        for phrase in OPEN_YOUTUBE_STRS:
            if phrase in text:
                ytubeupload()
                ytubeupdisplay()
                speak("opening")
                search_target = text.replace(phrase, '')
                url = 'https://youtube.com'
                webbrowser.get().open(url)
                speak("open")
                AUTOCHAT()

        OPEN_GOOGLE_STRS = ["open google","use google"]
        for phrase in OPEN_GOOGLE_STRS:
            if phrase in text:
                googleload()
                googledisplay()
                speak("opening")
                #search_target = text.replace(phrase, '')
                url = 'https://google.com'
                webbrowser.get().open(url)
                speak("open")
                time.sleep(2)
                AUTOCHAT()



## important continue from here
        NOTE_X_REMIND1 = ["help me remember","remind","stop me"]
        for phrase in NOTE_X_REMIND1:
```

```
        if phrase in text:


            FOR_ONE = ["in 1 minute","one minute"]
            for phrase in FOR_ONE:
                if phrase in text:
                    text = text.replace("me","you")
                    text = text.replace("i","you")
                    text = text.replace("in 1 minute","")
                    time_remind = "1 minute"
                    time_sleep = time_list[time_remind]
                    speak("I will remind you that   " + text + "in    "
+ time_remind)
                    thread_help_remind =
threading.Thread(target=AUTOCHAT)
                    thread_help_remind.start()
                    time.sleep(time_sleep)
                    speak("user you asked me to   " + text)
                    thread_help_remind.join()

            FOR_TWO = ["in 2 minute", "two minute"]
            for phrase in FOR_TWO:
                if phrase in text:
                    text = text.replace("me","you")
                    text = text.replace("i","you")
                    text = text.replace(phrase,"")
                    time_remind = "2 minutes"
                    time_sleep = time_list[time_remind]
                    speak("I will remind you that   " + text + "in    "
+ time_remind)
                    thread_help_remind =
threading.Thread(target=AUTOCHAT)
                    thread_help_remind.start()
                    time.sleep(time_sleep)
                    speak("user you asked me to   " + text)
                    thread_help_remind.join()

            FOR_THREE = ["in 3 minute", "three minute"]
            for phrase in FOR_THREE:
                if phrase in text:
                    text = text.replace("me","you")
                    text = text.replace("i","you")
                    text = text.replace("you","me")
                    text = text.replace(phrase,"")
                    time_remind = "3 minutes"
                    time_sleep = time_list[time_remind]
```

```
                    speak("I will remind you that   " + text + "in   "
+ time_remind)
                    thread_help_remind =
threading.Thread(target=AUTOCHAT)
                    thread_help_remind.start()
                    time.sleep(time_sleep)
                    speak("user you asked me to   " + text)
                    thread_help_remind.join()

              FOR_FOUR = ["in 4 minute", "four minute"]
              for phrase in FOR_FOUR:
                 if phrase in text:
                    text = text.replace("me","you")
                    text = text.replace("i","you")
                    text = text.replace("you","me")
                    text = text.replace(phrase,"")
                    time_remind = "4 minutes"
                    time_sleep = time_list[time_remind]
                    speak("I will remind you that   " + text + "in   "
+ time_remind)
                    thread_help_remind =
threading.Thread(target=AUTOCHAT)
                    thread_help_remind.start()
                    time.sleep(time_sleep)
                    speak("user you asked me to   " + text)
                    thread_help_remind.join()

              FOR_FIVE = ["in 5 minute", "five minute"]
              for phrase in FOR_FIVE:
                 if phrase in text:
                    text = text.replace("me","you")
                    text = text.replace("i","you")
                    text = text.replace("you","me")
                    text = text.replace(phrase,"")
                    time_remind = "5 minutes"
                    time_sleep = time_list[time_remind]
                    speak("I will remind you that   " + text + "in   "
+ time_remind)
                    thread_help_remind =
threading.Thread(target=AUTOCHAT)
                    thread_help_remind.start()
                    time.sleep(time_sleep)
                    speak("user you asked me to   " + text)
                    thread_help_remind.join()

              FOR_TEN = ["in 10 minute", "ten minute"]
```

```
            for phrase in FOR_TEN:
                if phrase in text:
                    text = text.replace("me","you")
                    text = text.replace("i","you")
                    text = text.replace("you","me")
                    text = text.replace(phrase,"")
                    time_remind = "10 minutes"
                    time_sleep = time_list[time_remind]
                    speak("I will remind you that   " + text + "in   "
+ time_remind)
                    thread_help_remind =
threading.Thread(target=AUTOCHAT)
                    thread_help_remind.start()
                    time.sleep(time_sleep)
                    speak("user you asked me to   " + text)
                    thread_help_remind.join()

            FOR_FIFTENN = ["in 15 minute", "fifteen minute"]
            for phrase in FOR_FIFTENN:
                if phrase in text:
                    text = text.replace("me","you")
                    text = text.replace("i","you")
                    text = text.replace("you","me")
                    text = text.replace(phrase,"")
                    time_remind = "15 minutes"
                    time_sleep = time_list[time_remind]
                    speak("I will remind you that   " + text + "in   "
+ time_remind)
                    thread_help_remind =
threading.Thread(target=AUTOCHAT)
                    thread_help_remind.start()
                    time.sleep(time_sleep)
                    speak("user you asked me to   " + text)
                    thread_help_remind.join()

            FOR_TWENTY = ["in 20 minute", "twenty minute"]
            for phrase in FOR_TWENTY:
                if phrase in text:
                    text = text.replace("me","you")
                    text = text.replace("i","you")
                    text = text.replace("you","me")
                    text = text.replace(phrase,"")
                    time_remind = "20 minutes"
                    time_sleep = time_list[time_remind]
                    speak("I will remind you that   " + text + "in   "
+ time_remind)
```

```
                    thread_help_remind =
threading.Thread(target=AUTOCHAT)
                    thread_help_remind.start()
                    time.sleep(time_sleep)
                    speak("user you asked me to   " + text)
                    thread_help_remind.join()

              FOR_TWENTYFIVE = ["in 25 minute", "twenty five minute"]
              for phrase in FOR_TWENTYFIVE:
                 if phrase in text:
                    text = text.replace("me","you")
                    text = text.replace("i","you")
                    text = text.replace("you","me")
                    text = text.replace(phrase,"")
                    time_remind = "25 minutes"
                    time_sleep = time_list[time_remind]
                    speak("I will remind you that   " + text + "in   "
+ time_remind)
                    thread_help_remind =
threading.Thread(target=AUTOCHAT)
                    thread_help_remind.start()
                    time.sleep(time_sleep)
                    speak("user you asked me to   " + text)
                    thread_help_remind.join()

              FOR_THIRTY = ["in 30 minute", "thirty minute","half an
hour","half hour"]
              for phrase in FOR_THIRTY:
                 if phrase in text:
                    text = text.replace("me","you")
                    text = text.replace("i","you")
                    text = text.replace("you","me")
                    text = text.replace(phrase,"")
                    time_remind = "30 minutes"
                    time_sleep = time_list[time_remind]
                    speak("I will remind you that   " + text + "in   "
+ time_remind)
                    thread_help_remind =
threading.Thread(target=AUTOCHAT)
                    thread_help_remind.start()
                    time.sleep(time_sleep)
                    speak("user you asked me to   " + text)
                    thread_help_remind.join()
```

```
TEST_HEAR = ['can you hear me','can you still hear me']
for phrase in TEST_HEAR:
   if phrase in text:
      speak("yes I can hear you")
      AUTOCHAT()



NOTE_FINAL = ['make a note','take note','new note','take a note','make note']
for phrase in NOTE_FINAL:
   if phrase in text:
      NOTE_SAVE = ['and save','save','and save it']
      for phrase in NOTE_SAVE:
         note_text = text.replace(phrase,'')
         note(note_text)
         AUTOCHAT()




GO_TO_SLEEP = ['hello']
for phrase in GO_TO_SLEEP:
   if phrase in text:
      speak('i am here')
      SLEEP_NOW = ['turn off','power off','sleep','shut up']
      for phrase in SLEEP_NOW:
         if phrase in text:
            speak('go to sleep mode')
            wakingupload()
            backtosleepdisplay()
            sleep_wake()

      EMAIL_STRS = ["email","gmail"]
      for phrase in EMAIL_STRS:
        if phrase in text:
            gmailload()
```

```
                    gmaildisplay()
                    get_email_info()
                    AUTOCHAT()
              Call_STRS = ['make a video call','video call']
              for phrase in Call_STRS:
                  if phrase in text:
                      invite_video_call()
                      wakingupload()
                      backtosleepdisplay()
                      sleep_wake()


        prompt = user_name+":"+text+"\n"+bot_name+":"
        conversation += prompt

        response = openai.Completion.create(
            model="text-curie-001",
            prompt=conversation,
            temperature=0,
            max_tokens=250,
            top_p=1,
            frequency_penalty=0,
            presence_penalty=0
        )

        response_str = response["choices"][0]["text"].replace("\n","")
        response_str = response_str.split(
            user_name + ":", 1)[0].split(bot_name+ ":",1)[0]

        conversation+=response_str+"\n"

        print(response_str)

        speak(response_str)
        i = i + 1
        print(i)
        if i == 10:
            speak('go to sleep mode')
            wakingupload()
            backtosleepdisplay()
            sleep_wake()
        recognizeload()
        outrecogdisplay()
```

```
wakie = "hello wake up"
nowwake = ['system ready','ready for your service','greeting']

def sleep_wake():

   while True:
        print("Listening")
        text = get_audio_auto()

        if text.count(wakie) > 0:
            speak('yes')
            talk_when_wake = random.choice(nowwake)
            wakingupload()
            wakingdisplay()
            speak(talk_when_wake)
            AUTOCHAT()


#-----------Run Code-----------
sleep_wake()
```

“Hello Bot” The Smart Voice Assistant

Thanapat Lappromrattana
Warakorn Mukdaprasert
Surasakul Charoenrujijin
Robotics and AI Department of Engineering
King Mongkut's Institute of Technology Ladkrabang
Bangkok, Thailand
62011272@kmitl.ac.th
62011288@kmitl.ac.th
62011256@kmitl.ac.th

***Abstract*—This thesis presents “Hello Bot” The Smart Voice Assistant, which aims to apply the knowledge gained by its authors to build an easy-to-use system that can improve the quality of life for elderly people. Factual data shows the population of younger generations is insufficient to meet the needs of an aging society. In addition, many elderly people struggle with using technology such as Youtube, Gmail, Video Call, and Google. They may also have difficulty remembering things and experience physical limitations that prevent them from participating in social activities and can lead to loneliness. The “Hello Bot” project has been created as a result to provide a voice assistant that gives users easy access to technology such as Youtube, Google, Gmail, Pill Reminder, Take Note, and Video Call. Furthermore, the project is equipped with the ability to converse about anything with the user through a unique and friendly robot face. This is made possible by the combination of various technologies, including Speech Recognition, NLP, Text to Speech, Animation, SMTP, Python, WebRTC, and Chat GPT. The project was evaluated through a number of test runs for each function, which resulted in a promising 90% success rate with only minor errors due to voice interference.**

Keywords*—*Aging society, Voice assistant robot, Chat Bot

I. Introduction

Thailand and many countries around the world are turning into an aging society is never before deniable. In addition, the works for adults, and school curriculum for kids cause the lack of interaction between older and younger generations. Not to mention the isolation that caused from COVID-19 pandemic situation that has tremendously affected people of all ages, elderly people included. Such situation created loneliness and some need support. New generations might be able to use the power of technology to help them. For example, video call, youtube, and social medias just to name a few. However, most of the elderly people can’t have such privileges due to numerous reasons. For instance, they don’t know how to use, or the lack of fund. The chance of loneliness to happens in elderly people is therefore higher compared to the newer generation. In addition, loneliness can affects their health. In conclusion, the aim of our project is to construct a prototype voice assistant system that can provide elderly people with support in using social media. Furthermore, providing them with a way to contact, and stay connected with their social circle. Last but not least, be a companion that stays, chat, and be a friend with elderly people.

II. Problem

A. Problem Statement and Background of the Study.

1) Thailand is becoming an aging society.

Reference to the United Nation department of Economic and social affairs, it states that Thailand is considered one of the world’s rapidly age society and expect to double between 2015 – 2050. (Prachuabmoh, V., n.d.)

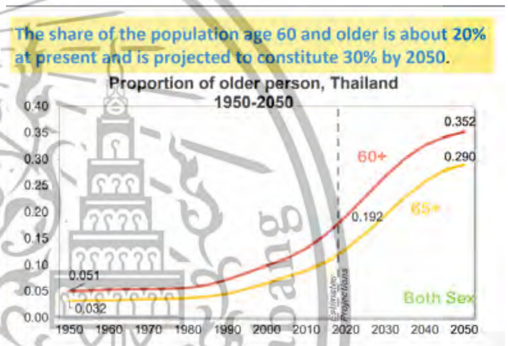
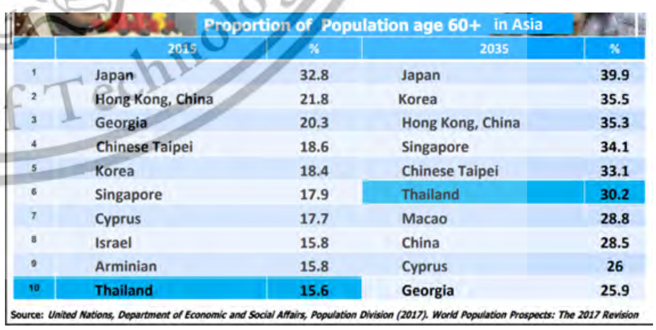


Fig1: Graph proportion of older person in Thailand. (Prachuabmoh, V., n.d.)

2) The world is becoming an aging society.

In addition, the United Nation department of Economic and social affairs has published that not only Thailand, but many more countries are facing the same problem. For instance, Japan, China, and Korea just to name a few. (Prachuabmoh, V., n.d.)



Proportion of Population age 60+ in Asia

	2015	%	2035	%
1	Japan	32.8	Japan	39.9
2	Hong Kong, China	21.8	Korea	35.5
3	Georgia	20.3	Hong Kong, China	35.3
4	Chinese Taipei	18.6	Singapore	34.1
5	Korea	18.4	Chinese Taipei	33.1
6	Singapore	17.9	Thailand	30.2
7	Cyprus	17.7	Macao	28.8
8	Israel	15.8	China	28.5
9	Arminian	15.8	Cyprus	26
10	Thailand	15.6	Georgia	25.9

Source: United Nations, Department of Economic and Social Affairs, Population Division (2017). World Population Prospects: The 2017 Revision

Fig2: Proportion of population age 60+ in Asia. (United Nation, 2017)

3) Older generations struggle.

As many as 77% of seniors struggle to use new technology and require an assistant to learn. Furthermore, As many as 56% of seniors who are already up online also need assistance to connect with friends and family on social media. Last but not least, most technologies are not designed for seniors. (No Isolation, n.d.)

B. Rationale of the Study.

There are many reasons why technology is useful to a person, but that is exactly the case when it applies to seniors.

- Elderly people have lower income and reduced mobility, and that can make meeting in person a challenge.
- New generation experience and senior's experience are not the same which cause the difficulty in catching up with new technology

C. Purpose of the study

- To improve the quality of life for seniors and people in need.
- To apply knowledge of Robotics and AI in real life applications.
- To create a voice assistant that is easy to use, but provides effective results.

III. Solution

The solution is to make keywords easy to understand, and easy to remember for users to command the system. This project's keyword for activating each command can be described as coherence. For example when asking an actual person to search for something on Youtube, one would say "help me find "something" on Youtube or one might say "search for "something" on Youtube". That's exactly how to command this project, using this project is as simple as described. More examples, and all system flow for each system are provided in the following section A - G.

A. Voice commands for using Youtube

Voice commands for using YouTube are able to open YouTube, search videos and play videos based on voice input.

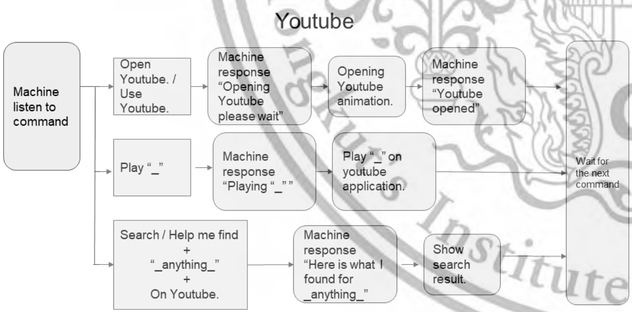


Fig79: Youtube System Flow Box Diagram (NEW)

B. Voice commands for using Google

Voice commands for using google is a function designed to help elders open the Google application and search on the Google application based on the voice input.

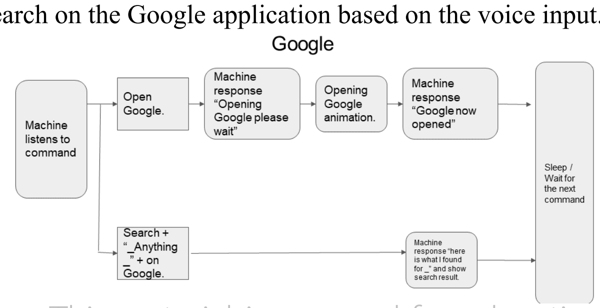


Fig80: Google System Flow Box Diagram (NEW)

C. Voice command for using Gmail

Voice command for using Gmail is designed to assist elders who are not tech-oriented in using Gmail to send text messages to another person using only voice with ease. Metaphorically speaking, this function is designed so that only saying the name of the person to whom the user wants to send a text message, followed by naming the subject and details of the email, is all that is required for the elders.

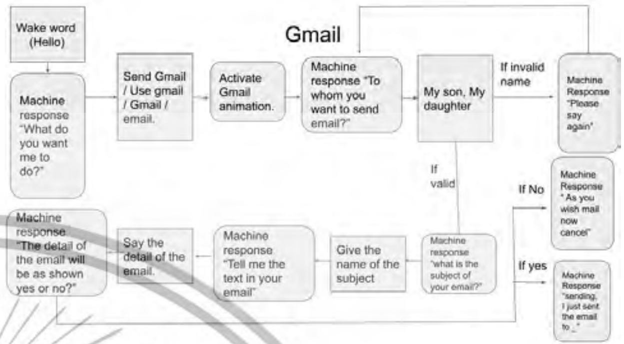


Fig5: Gmail System Flow Box Diagram

D. Voice command for reminder system

Voice command for the reminder system is a system that's designed to remind the elders of anything based on voice input and can set the timer as well. It has a function to initiate the reminder system, can choose what to be reminded of freely, can set a timer, and works parallel with the main program.

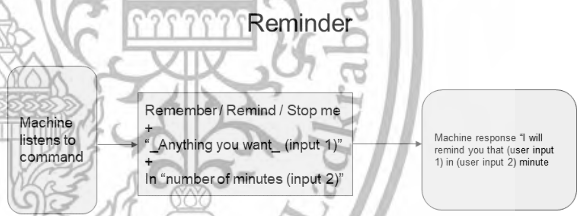


Fig81: Reminder System Flow Box Diagram (NEW)

E. Voice command for taking note system

The Take Note function is a system designed as a replacement for an actual living personal assistant who is capable of writing things down. It has the capability to initiate the take-note system and write the details of the note.

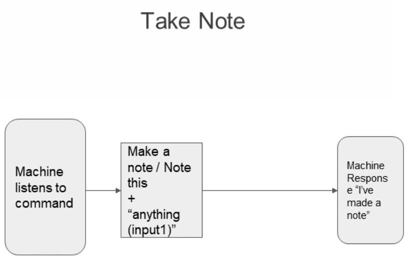


Fig82: Take Note System Flow Box Diagram (NEW)

F. Voice command for video call system

A video call function is designed to assist elderly in communicating with the doctor or nurse. It has the function of having a free meeting with a doctor displayed on the robot. This function integrates with the email function, which will send an invitation email to the receiver.

Fig8: Video Call System Flow Box Diagram

G. Voice command for Chat with ChatGPT

This voice Chat function is mainly designed for the project to act as an artificial friend to elderly people. Therefore, the system will send the user's voice input to ChatGPT to generate output when the user's voice input doesn't match up with the aforementioned conditions from A-F, and that's a big challenge of this project. The Authors of this project had to use NLP to create such a system that was intelligent enough to differentiate "This reminds me of my good time at KMITL" or "Is there harry potter related video on Youtube" , just for examples, are not commands, but questions for this voice chat function to generate output to talk back to the user.

IV. Methodology

The goal of this project is to make a prototype artificial assistant that can understand, change facial expressions, have conversation and provide assistance to elderly people. Therefore, the process of this thesis project can be divided into 4 steps.

Step 1) Convert human voice into computable data format.
Step 2) Identify conditions.
Step 3) Perform operation.
Step 4) Convert digital data into voice.

The block diagram of the system is as shown below.

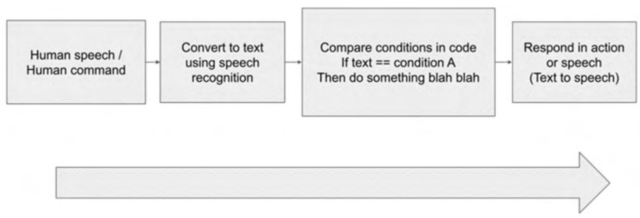


Fig 17: System Methodology

V. Theory

A. Natural Language Processing

Natural Language Processing or NLP is a branch of Artificial Intelligence that makes computers understand human language. In general, NLP frameworks can be divided into seven elements. (Lutkevich, B., 2021).

1) Segmentation
2) Tokenizing
3) Stop word
4) Stemming
5) Lemmatization
6) Speech tagging
7) Name entity tagging

B. Speech recognition

Speech recognition is a process of converting speech into a digital format that can be computed by computer. Speech recognition required a combination of hardware and software to perform well. That's because the way speech recognition works is when a microphone equipped with Analog Digital Converter (ADC) converts a human's voice which is in analog form and not computable by computer into digital data. Then, perform a pattern recognition process. In other words, when the computer receives digital data that's converted from the human voice, it plots that data on the spectrogram to identify the corresponding word by looking for a matching phoneme. After all the processes mentioned were done, the system will get text that developers can use to perform various tasks, and that's why some people call speech recognition as Speech to Text.

C. Text to Speech

Text to speech is a process of converting computed data into sound, sending it back to the user. This can be varied depending on the developer's choice of programming. Some may play the sound from the computer itself like pyttsx3, and some may play the sound after it's generated from a 3rd party like gTTS provided by Google. In simple terms, text to speech converts text into generated voice, and that's why it's called text to speech.

D. Simple Mail Transfer Protocol

Simple Mail Transfer Protocol or SMTP is a technique used to send email. SMTP has a server that acts as a middle man between two users. SMTP also has an SMTP port which plays a major role in operation. Port is what defines the services, or type of the protocol. For example, 80 or 443 for web browsing, 21 or 20 for FTP, and 25, 487 or 2525 for SMTP. However, nowadays people mostly approve of using port 587 for SMTP. Therefore, the port being used in this thesis is 587. (Philip, A., 2021)

E. Animation

Animation is being used in creating robot face system in this project. In theory, Animation is an illusion of movement. It's technically a technique which consists of multiple solid pictures arranged in a way that it seems like it's moving when those pictures change pretty fast in the same position in a certain time.

F. WebRTC

WebRTC stands for Web Real-Time Communications, is an open-source project that enables real-time voice, text, and video communication capabilities between web browsers and devices.

G. ChatGPT

ChatGPT is a trained model created by Openai. It's a Chatbot that is trained and can interact in a conversational way which is suitable for many tasks. For instance, normal conversation, answer questions, up to create a code if the user wishes for. In addition, ChatGPT provides many models, and they all have different purposes. By the time of developing this project. The latest version is ChatGPT 3 or "text-davinci-3". This model's capability is from normal conversation, mathematical related, up to creating a code. However, using ChatGPT requires an API, and the cost for "text-davinci-3" in response to its capabilities is not small. Therefore, "text-curie-001" is being used instead. Because the main purpose for adding ChatGPT to the "Hello bot" is to enable the robot to have normal conversations with humans. The model "text-curie-001" is sufficient for this task with a reasonable cost.

ChatGPT has many pros, but it also has some drawbacks due to the training of data only up to 2021. In addition, ChatGPT can't perform tasks that voice assistants can perform. For example, open or make a search on google and youtube, make a note, make a video call, send an email etc. ChatGPT itself is a chatbot.

VI. EXPERIMENTAL RESULT

A. Full System Work Flow Box Diagram

This project's full system is as shown in the Box diagram below.

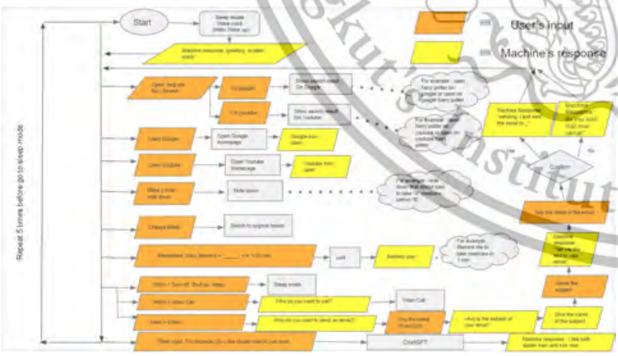
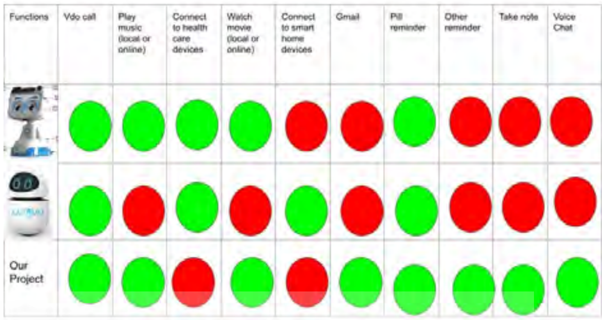


Fig 64: "Hello Bot" Model Diagram (FINAL)

B. Comparison

In this comparison section, similar products will be compared with this thesis project. The focus will be the scope of study listed in this thesis chapter 1, section 1.4. The products are Dinsaw and Kaitomm which are voice assistants that are mainly designed to give assistance to elderly people. (Obodroid, n.d.)(CT Asia Robotics, n.d.)

Table IV.
Compare all functions of Dinsaw, Kaitom, and this thesis.



C. Testing Summary

In total, ten tests have been conducted for each function. The environment that was conducted in was a bit crowded and not 100% quiet all the time. The result is as shown in table VI. In summary, the results perfectly satisfied the authors expectations. In addition to the testing that was being conducted as shown in the previous semester. The problem with the microphone was mentioned. However, in this project, an improvement was made by making the sound input automatically adjust for the ambient voice, so the system can differentiate what is the input and what's not more accurately. In the next section, Evaluation, the project will be evaluated and the extent to which it supports the thesis is discussed.

TableVI. Testing Summary

Functions	Total Number of test	Total Number of Passed test	Total Number of Failed Test
Youtube	10	9	1
Google	10	9	1
Reminder	10	7	3
Make Note	10	10	0
ChatGPT	10	10	0
Video call	10	9	1
Email	10	8	2

D. Evaluation

The testing described in the testing summary demonstrates that improving sound input by adjusting the amplifier does help with the accuracy of the input which affects the output's accuracy. However, it's still unable to yield the 100% accuracy due to time of adjusting of the sound amplifier. The way it works is by making the system auto adjust in 1 second time frame. In that time frame, the system will not receive any sound as an input, and later adjust the sound volume threshold to be according to that time frame sound's volume. Meaning, if there is a crowded environment nearby in that 1 second time frame, the threshold will be auto adjusted to 400 for example. The default ideal quiet environment is 300. That means after the system auto adjusts the threshold, the system can operate even in a crowded environment. However, here comes a small issue. The system auto adjusted for 1 second, and then waited to receive command input for another random second (> 1 second) before going back to auto adjust again. If the user coincidently gives commands in auto adjusting time, the voice command will not be recognized. However, improving the "Hello bot" to be able to operate in crowded areas does increase the possibility in many fields.

E. Real Use Case

This project got an opportunity from Prof.Dr.Pitikhate who contacted Ms. Pravee Chanachavarin and her team from AutoMe Motionworks to let this project be installed in a real robot called "Jiraff". Jiraff robot will be used in the real hospital. The pictures of Jiraff robot are shown below.

Fig 83. Real use case (Jiraff robot)

ACKNOWLEDGMENTS

At this moment, we would like to take an opportunity to give our sincere thanks to all the professors we took classes for the past 4 years. Thank you KMITL for all the lessons, experiences, teachers, staff, professional professors, incredible connections, and good environment. Without them, we wouldn't be able to gain enough knowledge and inspiration to develop this project. We would like to exclusively give our thank to Prof.Dr.Pitikhate Sooraksa who is our advisor for this project. With the knowledge we gained over the past four years, what we lacked was the right direction. Prof.Dr.Pitikhate played that role perfectly, and with sincere care that we can feel. The evidence is the fact that Prof.Dr.Pitikhate and our team have had meetings every 2 weeks over this semester for us to update and improve the project. At last, we would like to thank Ms. Pravee Chanachavarin and her team for providing "Jiraff" robot for our project real use case.

REFERENCES

[1] No Isolation.(n.d.). Why do many seniors have trouble using technology. https://www.noisolation.com/research/why-do-many-seniors-have-trouble-using-technology#:~:text=As%20many%20as%2077%20percent,use%20a%20smartphone%20or%20tablet.

[2] Knowledge Jump. (2012, August 25). Design Methodologies. http://www.nwlink.com/~donclark/design/design_models.html

[3] Prachuabmoh, V. (n.d.). Aging in Thailand. https://www.un.org/development/desa/pd/sites/www.un.org.development.desa.pd/files/unpd_egm_201902_s3_vipanprachuabmoh.pdf

[4] Pew Research Center. (2017, May 17). Barriers to adoption and attitudes towards technology.https://www.pewresearch.org/internet/2017/05/17/barriers-to-adoption-and-attitudes-towards-technology/

[5] Brownlee, J. (2019, August 7). What is natural language processing? https://machinelearningmastery.com/natural-language-processing/

[6] Lutkevich, B. (2021, March). Natural language processing. https://www.techtarget.com/searchenterpriseai/definition/natural-language-processing-NLP

[7] Koro, P. (2019, November 26). How to make Simple & Funny & Hilarious DIY animation.https://www.youtube.com/watch?v=1Xop0scjCWA

[8] Adobe. (2021, December 23). Use this article to learn about the various components of time in animate. https://helpx.adobe.com/animate/using/time.html

[9] Philip, A. (2021, October 25). What's an SMTP port & how to choose the right one. https://postmarkapp.com/blog/choosing-the-right-smtp-port#smtp-port-587-the-right-choice-for-nearly-every-situation

[10] CT Asia Robotics. (n.d.). Dinsaw Robot. https://www.dinsow.com/dinsaw-mini-ai/

[11] Obodroid. (n.d.). Kaitomm personal assistant. https://www.obodroid.com/product/personal-assistant/