

**LEGAL ANGEL: PERSONALIZED NLP CHATBOT FOR LEGAL  
GUIDANCE**

**BOEY ZHI XUAN**


**A project report submitted in partial fulfilment of the  
requirements for the award of the degree of  
Bachelor of Technology (Hons) Electronic Systems**

**Faculty of Engineering and Green Technology  
Universiti Tunku Abdul Rahman**

**September 2024**

## DECLARATION

I hereby declare that this project report is based on my original work except for citations and quotations which have been duly acknowledged. I also declare that it has not been previously and concurrently submitted for any other degree or award at UTAR or other institutions.

Signature :  \_\_\_\_\_

Name : Boey Zhi Xuan

ID No. : 2005745

Date : 03 September 2024

**APPROVAL FOR SUBMISSION**

I certify that this project report entitled **“LEGAL ANGEL: PERSONALIZED NLP CHATBOT FOR LEGAL GUIDANCE”** was prepared by **BOEY ZHI XUAN** has met the required standard for submission in partial fulfilment of the requirements for the award of Bachelor of Engineering (Hons) Electronic Systems at Universiti Tunku Abdul Rahman.

Approved by,

Signature : \_\_\_\_\_

Supervisor: Ts. Dr. Lee Han Kee

Date : \_\_\_\_\_

The copyright of this report belongs to the author under the terms of the copyright Act 1987 as qualified by Intellectual Property Policy of Universiti Tunku Abdul Rahman. Due acknowledgement shall always be made of the use of any material contained in, or derived from, this report.

© 2024, Boey Zhi Xuan. All right reserved.

Specially dedicated to  
my beloved mother and father

## **ACKNOWLEDGEMENTS**

I would like to thank everyone who had contributed to the successful completion of this project. I would like to express my gratitude to my research supervisor, Ts. Dr. Lee Han Kee for his invaluable advice, guidance and his enormous patience throughout the development of the research.

## **LEGAL ANGEL: PERSONALIZED NLP CHATBOT FOR LEGAL GUIDANCE**

### **ABSTRACT**

This project presents a cutting-edge artificial intelligence chatbot created to help with legal inquiries. The chatbot is outfitted with an array of sophisticated functionalities designed to augment user engagement and ease of usage. Voice input, which enables users to interact with the system using natural speech, is one of its main features. This makes the system more intuitive and user-friendly. Furthermore, people with a variety of linguistic backgrounds may easily converse in their favourite languages thanks to the chatbot's capability for multilingual translation. This AI chatbot's ability to learn new things on a constant basis is one of its best qualities. The chatbot may learn new things from users, which enables it to gradually increase and improve its knowledge base. The chatbot's capacity to learn guarantees that it stays current with legal knowledge and can adjust to the changing demands of its users. By adding these functions, the chatbot promotes a more dynamic and customized contact experience while also offering precise and prompt legal aid.

## TABLE OF CONTENTS

<b>DECLARATION</b>	<b>ii</b>
<b>APPROVAL FOR SUBMISSION</b>	<b>iii</b>
<b>ACKNOWLEDGEMENTS</b>	<b>vi</b>
<b>ABSTRACT</b>	<b>vii</b>
<b>TABLE OF CONTENTS</b>	<b>viii</b>
<b>LIST OF TABLES</b>	<b>xi</b>
<b>LIST OF FIGURES</b>	<b>xii</b>
<b>LIST OF SYMBOLS / ABBREVIATIONS</b>	<b>xv</b>
<b>LIST OF APPENDICES</b>	<b>xvi</b>

## CHAPTER

<b>1</b>	<b>INTRODUCTION</b>	<b>1</b>
	1.1 Background	1
	1.2 Evolution on AI and its impact on the current society	2
	1.2.1 Overview on First, Second, and Third Industry Revolution	2
	1.2.2 Overview on Artificial Intelligence in the Fourth Industrial Revolution	6
	1.3 Problem Statements	10
	1.4 Project Objectives	11
	1.5 Project Scope	11
	1.6 Report Outline	12
<b>2</b>	<b>LITERATURE REVIEW</b>	<b>13</b>



2.1	Propositional Logic	13
2.2	AI Learning	14
2.2.1	Nearest-Neighbour Classification	14
2.2.2	Perceptron Learning	16
2.3	Context-Free Grammar	17
2.4	Reliability	18
<b>3</b>	<b>METHODOLOGY</b>	<b>20</b>
3.1	Project Overview	20
3.2	Programming Language	22
3.2.1	Programming Language Selection	22
3.2.2	Programming Environment	23
3.2.3	Programming Library	24
3.3	Design Phase	26
3.3.1	Phase 1: Development	27
3.3.2	Phase 2: Implementation	32
3.3.3	Phase 3: Analysis	40
3.4	Project Management	40
3.5	Cost Estimation	42
<b>4</b>	<b>RESULTS AND DISCUSSION</b>	<b>44</b>
4.1	Website Overview	44
4.2	Chatbot Overview	46
4.2.1	Main Chatbot	46
4.2.2	Secondary Chatbot	51
4.2.3	Data Integration Between Two Chatbot	56
4.3	Chatbot Features	60
4.3.1	Voice Input	60
4.3.2	Multilanguage Text Translation	63
4.4	Chatbot Testing Result	66
<b>5</b>	<b>CONCLUSION AND RECOMMENDATIONS</b>	<b>67</b>
5.1	Conclusion	67

	x
5.2 Recommendations for Future Improvement	68
<b>REFERENCES</b>	<b>70</b>
<b>APPENDICES</b>	<b>76</b>

**LIST OF TABLES**

<b>TABLE</b>	<b>TITLE</b>	<b>PAGE</b>
	Table 3.1: Differences Between Visual Studio and Visual Studio Code	24
	Table 3.2: The Gantt chart for FYP 1	41
	Table 3.3: The Gantt Chart for FYP 2	42
	Table 3.4: Cost Estimation of Project Materials	43
	Table 4.1: Chatbot Learning Process	58
	Table 4.2: Process of Voice Input	61

## LIST OF FIGURES

<b>FIGURE</b>	<b>TITLE</b>	<b>PAGE</b>
Figure 2.1:	Structure of Grammatical Sentence Represented in The Form of Formal Grammar	18
Figure 3.1:	The Project Overview Flowchart Part 1	21
Figure 3.2:	The Project Overview Flow Chart Part 2	22
Figure 3.3:	Design Process	26
Figure 3.4:	Code for Chatbot Protocol Part 1	27
Figure 3.5:	Code for Chatbot Protocol Part 2	28
Figure 3.6:	Python Code Snippet for Chatbot Learning	30
Figure 3.7:	Code Snippet for Chatbot Translation from English to Other Language	31
Figure 3.8:	Code Snippet for Chatbot Translation from Foreign Language to English	31
Figure 3.9:	JSON File Code Snippet for Main Chatbot	33
Figure 3.10:	JSON File Code Snippet for Secondary Chatbot	33
Figure 3.11:	Python Code Snippet for Adding New Question and Response	34
Figure 3.12:	Python Code Snippet for Adding New Question into Existing Tag	35
Figure 3.13:	Python Code snippet for NLTK	36
Figure 3.14:	Python Code for Chatbot Training Model	37
Figure 3.15:	Image Used to Enhance User Experience	39

Figure 3.16: Python Code Snippet for Numbered List	39
Figure 4.1: The Chatbot Website Part 1	44
Figure 4.2: The Chatbot Website Part 2	45
Figure 4.3: The Chatbot Website Part 3	45
Figure 4.4: Chatbot Performing Simple Conversation with User	46
Figure 4.5: Result of Chatbot Response to Offense in First Section	47
Figure 4.6: Result of Chatbot Responses to Unrelated Place	48
Figure 4.7: Result of Chatbot Responses to Neutral Action	49
Figure 4.8: Result of Chatbot Response to Offense in Second and Third Section	49
Figure 4.9: Result of Chatbot Response to Offense in Fourth Section	50
Figure 4.10: Chatbot Responses to Positive Action	50
Figure 4.11: The Login Phase for The Secondary Chatbot	51
Figure 4.12: Secondary Chatbot Performing Simple Conversation with User	52
Figure 4.13: Chatbot Unable to Provide Answer to User Question	53
Figure 4.14: User Teaching Unknown Question to The Second Chatbot (i)	53
Figure 4.15: All Tag Listed in The Database	54
Figure 4.16: Result of New Question Added into an Existing Tag	54
Figure 4.17: User Teaching Unknown Question to The Second Chatbot (ii)	55
Figure 4.18: Result of Teaching Chatbot a New Question, Tag, and Answer	56
Figure 4.19: Training Model Output	57
Figure 4.20: Result of Training (i)	57
Figure 4.21: Result of Training (ii)	58

Figure 4.22: Result of Translation in Chatbot (i)	63
Figure 4.23: Result of Translation in Chatbot (ii)	64
Figure 4.24: Chatbot Training Result	66

**LIST OF SYMBOLS / ABBREVIATIONS**

AI	Artificial Intelligent
CFG	Context-Free Grammer
CSS	Cascading Style Sheets
HTML	HyperText Markup Language
IDE	Integrated Development Environment
IR3	Industry Revolution Three
JSON	JavaScript Object Notation
kNN-TSC	K-Nearest-Neighbour-Based Time-Series Classification
NLP	Natural Language Processing
NLTK	Natural Language Toolkit
UI	User Interface
VSC	Visual Studio Code

**LIST OF APPENDICES**

<b>APPENDIX</b>	<b>TITLE</b>	<b>PAGE</b>
APPENDIX A:	JavaScript Code for User Interface	76
APPENDIX B:	CSS Code for User Interface	82
APPENDIX C:	HTML Code for User Interface	94
APPENDIX D:	Python Code for Connecting HTML	97
APPENDIX E:	Python Code for Main Chatbot	102
APPENDIX F:	Python Code for Chatbot Learning	106
APPENDIX G:	Python Code for Speech-to-Text	108
APPENDIX H:	Python Code for Translator	109
APPENDIX I:	Python Code for Chatbot Training	110
APPENDIX J:	Python Code for Connecting Both Chatbot JSON File	114



## CHAPTER 1

### INTRODUCTION

#### 1.1 Background

Artificial intelligence (AI) has emerged as one of the most significant technical advancements in history. It has demonstrated its ability to completely transform a variety of sectors, from improving healthcare diagnoses to expediting commercial procedures. However, AI's ability to upend established employment paradigms is one of its most significant effects.

Discussions concerning the nature of employment in the future have been triggered by the emergence of AI-driven automation, with many experts projecting a dramatic change in the labour market (Huang & Rust, 2018). Concerns about AI algorithms replacing a significant percentage of human occupations are growing as they become more advanced and capable of handling complicated tasks. Although the automation revolution promises more production and efficiency, there are drawbacks as well, mostly in the form of job displacement and the requirement for reskilling and upskilling.

Legal services have historically been primarily supplied by human attorneys, who are skilled in deciphering intricate legal precedents, counselling clients, and advocating for them in court. But the development of artificial intelligence in legal technology is changing the face of the legal profession.

AI systems have shown remarkably adept at activities like document analysis, legal research, and contract evaluation that were formerly completed by attorneys. These artificial intelligence technologies have unmatched speed and accuracy when sorting through massive volumes of data, finding pertinent facts and producing insights that can guide legal strategy. Furthermore, AI-powered natural language processing makes it possible for computers to comprehend and produce content that is similar to that of humans, which increases their proficiency in linguistically demanding jobs like contract and legal document writing.

The eventual replacement of attorneys by AI in legal advice has far-reaching repercussions. While AI cannot fully replace human attorneys' solid understanding of morality and ethics, given that AI lacks a human sense of "right" and "wrong", it may greatly improve their abilities and cut down on the time and resources needed to do some legal duties. This increase in efficiency may result in lower costs for both customers and legal companies, as well as better access to legal services for marginalized communities (Canastro, et al., 2019).

## **1.2 Evolution on AI and its impact on the current society**

### **1.2.1 Overview on First, Second, and Third Industry Revolution**

A time of great change, the Industrial Revolution (IR) saw economies go from being predominately agrarian and handicraft-based to being dominated by industry and machine production. The eighteenth century saw the start of this process in England. During this time, significant technological advances were made in the areas of transportation and communication, such as the railroad and the telegraph, the use of iron and alloy iron, the creation of machines like the steam engine and the spinning jenny that increased production, and the establishment of the factory system, and the most important of all, the discovery of new energy sources (Lucas, 2002).

Before expanding to Belgium and France, the Industrial Revolution was first mostly limited to Britain between 1760 and 1830 (Mohajan, 2019). Other nations

industrialized more slowly, but as they became industrial powers, Germany, the United States, and Japan overtook Britain's early achievements (Crafts, 1996). The Industrial Revolution didn't reach nations like China and India until the middle of the 20th century, and Eastern European nations didn't begin to industrialize until the 20th century (Michael & Martin, 2017).

Significant alterations to social, political, and economic structures were brought about by industrialization. The rise of working-class movements, the creation of managerial hierarchies to control labour division, the emergence of new authority patterns, and a wider distribution of wealth and increased international trade were some of these, as were political changes brought about by shifts in economic power. It also sparked battles against problems like urban overcrowding and industrial pollution (Vries, 2009).

The Second Industrial Revolution was a great technological advancement, because of that, this revolution is also known as Technological Revolution. Rapid advancements in manufacturing and communication technologies characterized this period of time. The speed at which new discoveries and technologies were produced allowed people all around the world to live far better lives.

An entirely new era of transportation was also ushered in with the development of the vehicle and the airplane. Railroad businesses started to appear all around during this time and soon took control of the transportation sector (Smil, 2005). Railroads became the main means of long-distance transportation because they made transportation affordable and allowed for faster and more efficient travel than any other means available at the time. More quickly and across greater distances might now be travelled by people than ever before.

Lastly, a variety of societal changes were also brought about by the Second Industrial Revolution. This was made possible by modern technology that opened up new avenues for interpersonal interaction. The invention of the telephone, electricity, and the light bulb, for instance, revolutionized workplace and communication practices.

The majority of communication before this time was restricted to in-person meetings or sluggish, unreliable channels like telegraph lines. However, the telephone's introduction in 1876 signalled a significant turning point, and throughout the ensuing few decades, the industry expanded quickly. Thousands of miles of cable were laid across the nation by dozens of companies in an early rush to develop long-distance networks (Mokyr & Strotz, 1998). The radio emerged as a significant new medium for communication in the 1920s, and the first transatlantic transmissions took place in 1927. The following decade saw the introduction of television, which soon rose to prominence as a major source of entertainment.

There was a lot of invention and advancement during the Second Industrial Revolution. However, it did not alleviate the adverse effects of the initial industrial revolution; and unfortunately, it made them worse. These include, but are not limited to, growing environmental harm and pollution, squalor and overpopulation, and an expansion of the wealth divide (Mokyr & Strotz, 1998). Additionally, the second industrial revolution had several before unheard-of negative effects, like the emergence of monopolies and enormous businesses, child labour, and the ability of governments to track and spy on citizens due to technological advancements.

The rise in unregulated child labour during the Second Industrial Revolution is arguably the most catastrophic negative effect. Children as young as four years old were frequently made to labour long hours in dangerous and unhealthy industries for little compensation in order to support their destitute families.

The late 1900s saw the beginning of the Third Industrial Revolution, sometimes known as the Digital Revolution. Its traits include the development of nuclear energy, the creation of the Internet, and the growth of automation and digitization through the use of electronics and computers (Roberto & Ingrid, 2015). The use of computers and other modern technology to automate industrial operations developed rapidly during this period. The advancement of telecommunications enabled globalization. Consequently, this allowed businesses to shift their production to economies with lower labour costs.

Semiconductors, mainframe computers, microprocessors, MOS transistors, the internet (including ultra-fast 5G communication, renewable energy, and driverless mobility), renewable electricity, e-commerce circa 1995, and the subsequently developed smartphone were among the great inventions of the third industrial revolution. The most significant forces behind this are thought to be found in the fields of nano, bio, and IT technologies, 3D printing, artificial intelligence, robots, etc. The information and communication technologies (ICT), sophisticated manufacturing, education, healthcare, finance, and administration are just a few of the current industries that have been greatly impacted by this industrial revolution (Mohajan, 2021; Roberts, 2015).

Through the various developments during this period, this report is going to focus on the development and advancement of the robotic technologies, as robots are developed using a variety of high-tech and artificial intelligence technologies.

The term "robotics" refers to an interdisciplinary field encompassing all scientific and technical disciplines. It is regarded as the planning, building, applying, and using of robots. With the employment of artificial intelligence devices in the industrial units, this branch has been expanding recently (Madakam, et al., 2019), as AI must have a central role in robotics if the connection is to be intelligent (Brady, 1985). Isaac Asimov invented the three laws of robotics as; (i) a robot cannot injure a human; (ii) it must obey human commands even when doing so would violate the First Law; and (iii) it must defend its own existence, so long as doing so does not interfere with the First or Second Law's operation (Niku, 2020).

The development of artificial intelligence (AI) and worldwide robotics strategy has considered concerns about human welfare, health, and safety. Automation and robotics advances have a significant impact on increasing efficiency and productivity. Robots can be utilized in tasks that are hazardous, unclean, boring, or both, such as cleaning municipal rubbish, dismantling bombs, operating in hazardous environments like space and underwater, and operating nuclear power plants. These can be used to measure movement that is too small or quick for the human eye to see, detect minuscule amounts of invisible radiation, and gather information beyond the range of the five human senses (Murashov, et al., 2016).

The third industry revolution has made global impacts, which results in positive and negative effects. Eventhough, the positive effects are more fruitful than negative effects, but that doesn't indicate the negative impact are negletable, especially those negative impact which result in some crisis which amplifer the negative impact left by second industry revolutoin.

The worldwide climate has altered throughout the IR3. This led by a number of global problems, including air and water pollution, habitat destruction, biodiversity loss, greenhouse gas emissions, global warming, and climate change (Mohajan, 2011; 2021). Which further results in numerous extinct species have already disappeared, and more will do so in the near future. Other than that, most parts of the world are seeing a decline in living conditions (Mohajan, 2012). A billion or more people on the planet suffer from malnutrition and hunger. Harsh and dangerous working conditions in factories have increased and many workers have died at the hands of machines throughout the IR3 (Kaplan & Claire, 1958, cited in Mohajan, 2021), as has the prevalence of underage labor. The wealth disparity between the rich and the poor is growing. By the end of IR3, there is a startling global increase in unemployment.

### **1.2.2 Overview on Artificial Intelligence in the Fourth Industrial Revolution**

In January 2017, World Economic Forum Founder and Executive Chairman, Klaus Schwab, published a book titled *The Fourth Industrial Revolution* (Schwab, 2017). Since then, the phrase "Fourth Industrial Revolution" (4IR) has been used to characterize and examine how new technologies are affecting almost every aspect of human development in the early 21st century, ranging from changing national political attitudes and social norms to economic growth and international relations (Philbeck & Davis, 2019).

Similar to previous industrial revolutions, the Fourth Industrial Revolution presents remarkable prospects for people, sectors, and countries. Improved system optimization is promised by artificial intelligence, the Internet of Things, and the

prospect of quantum computing (Tan & Wu, 2017). Beyond the rise of cryptocurrencies, distributed ledger technologies—like blockchain—are proving useful for tasks like regulating fraud and externalities in value chains, facilitating safe, digital identity, and enhancing public procurement transparency (Santiso, 2018). A decade ago, it would have seemed unthinkable that neurotechnology would one day augment human cognitive and physical abilities in ways that are now possible thanks to their rapid advancements.

Prominent scholars contend that the future will be shaped by the effects of the fourth industrial revolution on business and government. The disruption brought about by the fourth industrial revolution and technology are beyond human control. The prospects presented by the fourth industrial revolution, however, are predictable: artificial intelligence (AI) playing a more active role, a better quality of life through robots, and a linked life through the Internet.

Once again, this study will only focus on artificial intelligence in the fourth industrial revolution. Artificial Intelligence is the study of making things see, think, and act appropriately and anticipatorily in their surroundings to a level of humans would do (Mhlanga, 2023). This artificial intelligence revolution began with computers (Makridakis, 2017).

One area of artificial intelligence called machine learning uses data to find approximations of solutions through the process of generalization. Checkers was used in 1959 to show how machine learning allows computers to learn from experience without the need for extensive programming (Samuel, 1959). Machines soon showed themselves to be superior to humans in solving intellectually challenging tasks, like problems described by lists of mathematical formulas. However, they were challenged for decades to perform intuitive tasks that humans could complete with ease, like speech recognition or face recognition (Itamar Arel, et al., 2010).

However, the development of AI continues as a convolution neural network was shown to simulate neuronal activity and processing in 1989, and it was able to categorize handwritten digits with robustness (Goodfellow, et al., 2018).

Convolutional networks are regarded as the catalyst for the AI revolution (Velarde, 2019). Deep artificial neural networks, notably recurrent ones, have been winning machine learning and pattern recognition competitions in recent years (Schmidhuber, 2015). The article also states that deep networks emerged victorious in every ImageNet competition and gained popularity across a wide range of study domains.

One area of machine learning is called deep learning. Deep models have proven successful because of advancements in learning algorithms, GPU implementations, and copious amounts of data (Schmidhuber, 2015 cited in Velarde, 2019). Deep models learn a hierarchy of concepts in a layered approach, from simple to sophisticated ones.

Experts believe that artificial intelligence (AI)-based solutions can help us get closer to solving even the most difficult issues, such lowering human error, removing risk, providing round-the-clock availability, and handling big data smoothly (Mughal, 2018; Bhbosale, et al., 2020; Khanzode & Sarode, 2020). There are some optimistic who believe that the AI will bring a positive social impact in the future, especially in the aspect of advancing science, automation, and productivity.

For instance, AI-powered models and simulations can aid in the solution of challenging issues in disciplines like engineering, medicine development, and climate science (Nenad Tomašev, 2020). Without the need for expensive and time-consuming actual tests, scientists and researchers can investigate multiple situations and optimize solutions thanks to these simulations.

Representative case will be the Harmonized Landsat and Sentinel-2 (HLS) datasets from NASA, IBM Research and NASA developed and published a geospatial AI foundation model for Earth observation (Masek, et al., 2018). Crop production predictions, natural catastrophe monitoring, and land use change tracking are all possible with the new foundation model.

AI optimization algorithms can be used to optimize energy use, lower emissions, and improve environmental monitoring in order to reduce carbon footprint (Vinuesa, et al., 2020; Stein & Benoit, 2022; Hasmi, et al., 2023). AI-powered



precision farming methods in agriculture have the potential to decrease environmental impact, increase crop output, and use less pesticides.

According to a significant body of research, AI and the fourth industrial revolution hold promise in mitigating various adverse effects stemming from preceding industrial revolutions, including issues like climate change, working and living conditions, and environmental pollutions.

On the other hand, there are some pessimist or doubter on this AI advancement. The main arguments for them are as follows: Artificial Intelligence (AI) in particular has had a significant impact on society as a result of the Fourth Industrial Revolution, stirring ethical questions and concerns about its consequences. Robots are now increasingly intelligent and self-sufficient thanks to artificial intelligence and machine learning, but they still lack a crucial ability: moral and ethical thinking (Puri, 2020; Duggal, 2024; Xu, et al., 2018). This restricts their capacity to choose morally or ethically in difficult circumstances. The question of whose moral standards robots should inherit is also the most important one. Individuals have very different moral standards than those in other nations, religions, and ideological contexts. Assigning moral values to artificial systems is challenging and limited due to uncertainty about the moral paradigm to use (Conitzer & Sinnott-Armstrong, 2021).

The overall economic substitution of labour by automation may result in a net loss of workers, which could widen the return on capital difference from labour. The search for talent will lead to a potentially more polarized labour market. Digitization and computers will replace low-skilled, low-paying occupations. More skilled, higher paying positions are less likely to be replaced (Coldwell, 2019; Xu, et al., 2018). Tensions in society may rise as a result of this growing dichotomization.

Many types of jobs are threatened by artificial systems that can reason their way through complicated problems, but they also open up new opportunities for economic expansion. Half of all work activities currently in use would be able to be automated by already available technologies, allowing businesses to save billions of dollars and provide new kinds of employment (Manyika, et al., 2017). For instance,

autonomous vehicles might only slightly replace taxis and Uber drivers, but autonomous trucks might completely change the shipping industry and eliminate the need for truck drivers.

### 1.3 Problem Statements

For a variety of reasons, the general public frequently faces formidable barriers when seeking to interact with the judicial system. First of all, a lack of legal understanding among the general public sometimes leaves people unable to identify legal difficulties or when they want legal aid. Many people may unintentionally put off getting help or taking care of their legal issues if they don't have a firm grasp of their rights, obligations, and the nuances of the law.

Furthermore, there is widespread uncertainty and mistrust regarding the effectiveness and impartiality of the legal system. People may have misgivings about the availability of justice because they believe that going through the legal system would be difficult, time-consuming, or unfair to them. Even in cases when they have valid legal concerns, this cynicism may discourage individuals from interacting with the legal system entirely.

Lastly, financial limitations often provide a significant obstacle to getting legal help. Many people find legal services to be expensive due to their high costs, especially those from disadvantaged or low-income backgrounds. People may be discouraged from obtaining assistance due to the high expenses of legal counsel and processes, which leaves them open to abuse or unable to properly claim their rights. A person in poverty who is charged with a criminal cannot afford justice. States have the financial means to offer top-notch legal counsel to those facing criminal charges, but the majority of them are unwilling to offer poor individuals a minimum degree of legal assistance (Bright, 2010).

## **1.4 Project Objectives**

The objectives of the project are shown as follows:

- i) To design an AI system to effectively allow user in Malaysia to obtain more information about current penal code ACT 574 on chapter XV.
- ii) To implement a question-answering system that can respond to legal queries.
- iii) To design a platform for professional bodies to train the system.

## **1.5 Project Scope**

The project aims to design and implement a robust question-answering system capable of accurately responding to legal queries, specifically focusing on the Malaysian penal code. The objectives encompass three main goals: designing the system to handle legal scenarios, implementing mechanisms to address ambiguities in questions, and creating an AI-driven platform to provide the general public in Malaysia with more accessible information about the current penal code.

The main deliverables of the project will be a chatbot interface that will allow users to inquire about and get legal knowledge about a particular provision of the Malaysian penal code. The chatbot's purpose is to converse with people in natural language and reply to their inquiries in a clear and succinct manner.

As for the project timeline, this project is aimed to complete in eight months, during which time the chatbot will be developed, tested, and refined. In the hope of fully evaluate and optimize the system within the limitations of time and resources, this timeline has been selected.

Lastly the project is expected to face some constraints or limitations of time and resource. In order to fulfil the deadline, effective project management and work prioritization are required due to the time constraint. In addition, the limited

resources of the project necessitate the prudent distribution and application of those resources to guarantee the accomplishment of the goals within the allotted time frame.

## **1.6 Report Outline**

This report will be divided into five chapters: introduction, literature review, methodology, results and discussion, and conclusion and recommendations. The introduction chapter will describe the project's background, problem statement, objectives, and scope. Chapter 2, the literature review will discuss the development of AI and chatbot, Natural Language Processing (NLP), type of AI design and implementation, and ways to obtain legal information. Chapter 3 Methodology describes the program's flow and Gantt charts, the programs' design and architecture, and selection of library program. Chapter 4 results and discussion section describes the constructed program, result of testing, and study of outcome. Finally, Chapter 5 conclusion and recommendations section summarises findings, conclusions, implications, recommendations for future work, and final thoughts.

## CHAPTER 2

### LITERATURE REVIEW

#### 2.1 Propositional Logic

In propositional logic, the statement that is taken into account is called a proposition. A proposition is a statement that might be true or untrue, but it cannot be both at the same time. Propositions in AI refer to any facts, circumstances, or other statement about a specific fact or scenario in the real world. Propositional logic expressions, often known as propositions, are constructed using propositional symbols, connective symbols, and parentheses.

It is possible to alter and combine a proposition to reflect the underlying logical relations and rules by using proposition operators such as conjunction ( $\wedge$ ), disjunction ( $\vee$ ), negation  $\neg$ , implication  $\rightarrow$ , and biconditional  $\leftrightarrow$ . Propositional logic is the basis for automated reasoning in AI systems. Modus Ponens and Modus Tollens are two examples of logical inference rules that let systems infer new information from known facts.

Many systems and tasks involving logical reasoning use propositional logic. This, according to Li et al. (2012), is mostly useful as a tool for formalizing reasoning processes. Propositional logic is used by the author to show how various propositions can be examined and worked with to arrive at conclusions. Additionally, the document demonstrates the actual applications of these logical procedures in areas like automated reasoning systems, artificial intelligence, and computer algorithms. The paper demonstrates how logical consistency and validity can be

systematically evaluated by dissecting intricate logical frameworks into elementary assertions.

Propositional logic is also used to guarantee that workflow models are accurate. Logic formulas representing a variety of workflow components, including joins, cycles, AND, OR, and XOR splits, can be created using propositional logic (Bi & Zhao, 2004). In propositional logic, the workflow model is defined as a collection of premises, and the verification process is framed as a deductive argument. The objective is to confirm that these premises lead to the logical conclusion, which signifies the workflow's effective completion.

To depict interactions between activities, logical operators such as AND, OR, XOR, and implications are utilized. Process inference is made possible by the logic-based method, which also simplifies the logical model and spots possible problems like synchronization failures or deadlocks. Furthermore, to address workflow-specific issues, the notion of limited truth tables is presented, which eliminates impossible cases and concentrates on legitimate execution pathways.

Last but not least, in order to address issues pertaining to China's economic mismatches between supply and demand, propositions using intuitionistic fuzzy logic, a type of propositional logic, have been researched and examined. According to Wang et al. (2020), the IFLP has the ability to select from a few different investment projects and base their decision on the intended result.

## **2.2 AI Learning**

### **2.2.1 Nearest-Neighbour Classification**

Neighbours-based classification is a kind of instance-based learning, also known as non-generalizing learning, in which training data instances are merely stored rather than an internal model being built. A query point is allocated the data class with the greatest number of representatives among its nearest neighbours. Classification is

determined by a simple majority vote of each point's nearest neighbours.

Two distinct closest neighbours' classifiers are known world-wide: K-Neighbours-Classifier uses an integer number that the user specifies to implement learning based on each query point's nearest neighbours. When a user specifies a floating-point value for each training point, Radius-Neighbours-Classifier applies learning based on the number of neighbours within a set radius of each training point.

The most popular method in neighbour's classification is the K-Neighbours-Classifier. The ideal amount depends heavily on the data; generally speaking, a bigger value reduces the impacts of noise but also blurs the boundaries of the categorization.

According to Mucherino, et al. (2009) k-nearest-neighbour method is one of the well known data mining techniques, they are able to use this method to perform climate forecasting in Florida and Georgia. Since the climate data are able to obtain all year long, therefore k-nearest-neighbour is suggested by the as it might be able to outperform other method. They found that this method was sble to imporove the accuraracy of the forecasts.

This nearest-neighbour classification techniques are being frequently used in real-world picture data mining, but this technique used to be having multiple flaws. The drawbacks are as follows: it performs poorly on short datasets and on high-dimensional data; and it depends a lot on the feature and distance measure that are used. But this issue is then tackled by Wang, et al. (2010), they are able to improve the technique by using a novel nearest-neighbour method.

Firstly, they suggested to add a brand-new neighbourhood similarity metric in which the average similarity and original image-to-image similarity of their nearby unlabeled data are merge to determien the similarity between images in the database. Secondly, they adopt a kernelized locality sensitive hashing to tackle the effectiveness of nearest-neighbour classification. Lastly, they suggest fusing the discriminating power of several features by taking into account all of the retrieved

closest neighbors via hashing systems employing various features/kernels in order to improve the robustness of the technique on various picture genres.

The K-nearest-neighbours-based time-series classification (kNN-TSC) method is favoured due to its instance-based learning approach, where the classification of a new instance is determined by comparing it to the most similar instances within the dataset (Lee, et al., 2012). This approach assumes that instances that share close similarities, based on specific metrics, belong to the same class, which makes it particularly useful for time-series data. The ability to tune several parameters, including window size, gap size, and envelope width, to maximize classification performance is one of the main advantages of the kNN-TSC approach. Empirical evaluations carried out on real-world datasets demonstrate the great efficacy of the kNN-TSC, indicating its potential for useful applications in sectors such as telecommunications.

### **2.2.2 Perceptron Learning**

One of the most basic artificial neural network topologies is the perceptron. It is the simplest kind of feedforward neural network, which consists of a layer of output nodes entirely coupled to a single layer of input nodes.

Perceptron is classified into two types: single-layer and multilayer. Learning linearly separable patterns is the only thing a single-layer perceptron can do, efficient for tasks that allow a straight line to separate the data into different categories. On the other hand, multilayer perceptrons include two or more layers, which allows them to handle more intricate patterns and correlations in the data, giving them superior processing power.

Perceptron learning is made up of various crucial parts that cooperate to process data. Initially, it receives a number of input features where it will be associated with a weight. This is to indicate how much of an impact the input will have on the perceptron's output. These weights are able to change during the training



process in order to determine their ideal levels. Furthermore, by integrating the inputs with their corresponding weights, perceptron learning is able to determine the weighted sum of its inputs, this function is known as summation function. The weighted sum will then be passed through an activation function in which it will provide the output in the form of binary numbers.

One of the most important aspects in the perceptron model which emphasize and highlighted by Gallant (1990) is the perceptron model frequently includes a bias component in addition to the input features and weights. The bias functions as an extra parameter that is learned during training, enabling the model to make modifications that are independent of the input features. Lastly, a key component of training is the learning algorithm, often known as the weight update rule. Based on this technique, the perceptron modifies its weights and bias in accordance with variations between the expected and actual outputs.

Begum, et al. (2019) is able to implement perceptron learning by using TensorFlow library in Python. They have demonstrated the ability of perceptron in which is able to be used as a linear classifier and using perceptron to implement AND Gate. With using perceptron learning algorithm in their dataset, they found that this algorithm is able to guarantee the performance and the result.

### **2.3 Context-Free Grammar**

Syntax definitions for programming languages and natural languages are defined using a formal framework called a Context-Free Grammar (CFG). It is made up of production rules that define how different linguistic symbols might be joined to form words or sentences that are considered legitimate. Terminals are the fundamental symbols or tokens that comprise the language's strings in CFG. Non-terminals, on the other hand, indicate collections of strings and can be substituted by combinations of other non-terminals and terminals. In natural language processing, theoretical computer science, and compiler design, CFGs are extensively utilized due to their potency in characterizing languages having hierarchical structures.

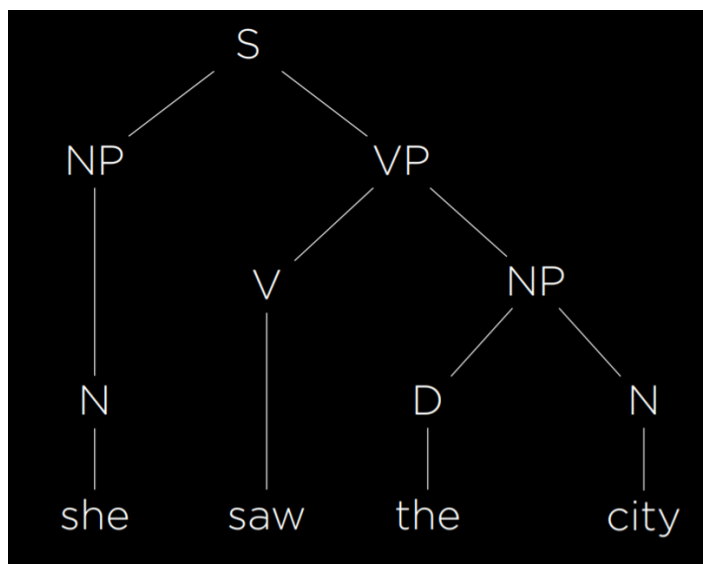


Figure 2.1: Structure of Grammatical Sentence Represented in The Form of Formal Grammar

Unold at 2005 is able to study the usage of CFG to develop a grammar-based classifier system. His result were successful and able to show a 100% grammar fitness. Other than that, CFG is used to studied the relation between itself and trained recurrent neural networks, which is then used to explain the language learnt by the recurrent neural network (Yellin & Weiss, 2021).

## 2.4 Reliability

This section will be used to examine how people develop trust in AI systems, especially in scenarios where these systems provide legal advice.

Kahr, et al. (2023) researched on the understanding of dynamics of trust over time by analysing over different factors influence trust during repeated interactions with AI. This study is based on 2x2 experimental design where the researchers manipulating the variables between model accuracy (high vs low) and explanation type by the AI (human-like vs abstract). The result showed that participants' trust increased over time when they engaged with the high accuracy AI, while participants

lost faith and confidence in the AI with low-accuracy models as soon as they discovered errors. Surprisingly, trust was not significantly impacted by the sort of explanation, but participants appreciate complex explanations when they think the AI is competent.

The study emphasizes how AI systems must be carefully designed to promote confidence, especially by guaranteeing accuracy and offering justifications that enhance the system's functionality.

## CHAPTER 3

### METHODOLOGY

#### 3.1 Project Overview

Figure 3.1 and 3.2 shows the project overview flowchart for this project. This project is begun with introduction where the author understands the importance and the evolution of AI. The Introduction then further include the project's background, objectives, problem statement, and project scopes. Continue by the author research for several articles to understand AI and to perform literature review for this project. Furthermore, the programming language selection along with system design is being considered before designing the user interface and writing the programming code for the chatbot. After completing those, integrating the user interface with the chatbot to obtain the result wanted for the project become the crucial part. Moreover, additional features are added to the chatbot once the previous part has been completed. Lastly, result, discussion, conclusion and recommendations are being added to the report.

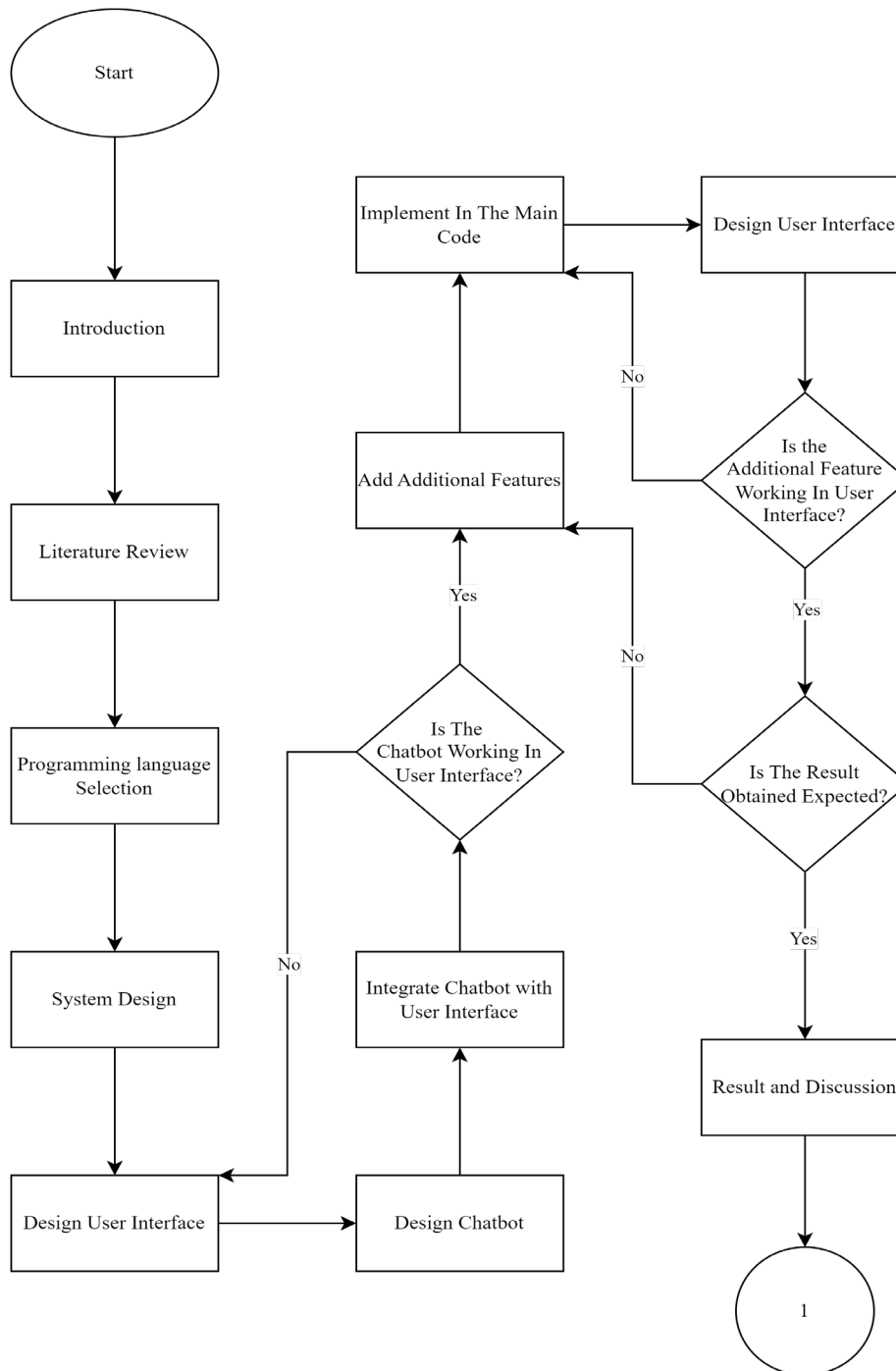


Figure 3.1: The Project Overview Flowchart Part 1

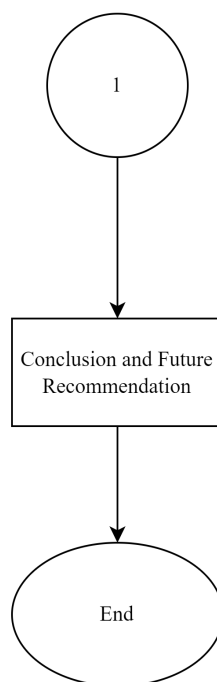


Figure 3.2: The Project Overview Flow Chart Part 2

## 3.2 Programming Language

### 3.2.1 Programming Language Selection

Python is the main software coding languages in this project as it is one of the best choices to use for web development and software development. Python simplicity, versatility, clear, and readable syntax also enhance the reason for the author to choose this programming language. Furthermore, due to the complexity of this project, python code is needed as it could provide extensive standard library which supports web development, data analysis, artificial intelligence, and automation. Lastly this language's strong community support also reduces the complexity of understanding the language.

Furthermore, JavaScript, HTML and CSS are chosen in this project for front-end web development. HTML (HyperText Markup Language) is needed as it

provides the author to define the elements of the website such as headings, paragraphs, image, and links. This language perfectly demonstrates the ability to code the content that the author wanted on the website. Secondly, CSS (Cascading Style Sheets) is chosen to handle the presentation and design aspects of the website. It allows the author to control the visual style such as layout, colours, and fonts of the website. Lastly, JavaScript is chosen as well, this is to enable the website to have animations allow the user to interact with the website without the need to reload the page, making the website more engaging and user-friendly.

Other than that, JSON (JavaScript Object Notation) file is used for transmitting data between a server and a web application. JSON is well known in web development for data exchange due to its simplicity and ease of integration with various programming languages.

Altogether, JavaScript, HTML, and CSS form the core of web development, while Python is used to create web applications, and JSON file is used to transmit data from Python to HTML.

### **3.2.2 Programming Environment**

For this project, Visual Studio Code (VSC) is chosen as the code editor as it is able to provide multiple customisable features, lightweight, powerful, and can be installed and used in multiple platforms. Other than that, this text editor is chosen as it has a built-in support for JavaScript, HTML, and CSS with a feature-rich extension ecosystem with Python, which is perfect for this project.

There is a tough competitor for VSC which is the Microsoft's Visual Studio. It is an Integrated Development Environment (IDE) that allow user to build, edit, and debug code. Following is a table of comparison between Visual Studio and Visual Studio Code.

**Table 3.1: Differences Between Visual Studio and Visual Studio Code**

<b>Visual Studio</b>	<b>Visual Studio Code</b>
Is an IDE	Is a code editor
Slower processing speed	Comparatively faster
Free editor but also comes with better and paid IDE version	Completely free of cost and it is open-source
Large download size	Lightweight compared to Visual Studio and doesn't require heavy download
Requires more spaces to work better and smoother	Does not need a lot of space to run
Only runs on macOS and Windows	Can run on macOS, Windows, and Linux
Limited plugins	Large variety of plugins and extensions

As shown in Table 3.1, the reason for the author to choose VSC over Visual Studio is the comparative speed of VSC is much faster than Visual Studio, VSC having much more flexibility, it requires much less space on the computer while could run smoother than Visual Studio, and lastly VSC comes in large variety of plugins, extensions, and it has a large community base.

### 3.2.3 Programming Library

This section of the showing the list of python libraries needed to be in the computer for this project to run smoothly with ease:

- i) Flask.
- ii) PyTorch.
- iii) torchvision.
- iv) NLTK.
- v) Googletrans.
- vi) SpeechRecognition.



vii) PyAudio.

Flask is chosen as it is a web framework that is able to provide tools to build a web application, including blog, wiki, commercial website, and for this project it will be used for building a website. It has one of the best advantages which is having little dependency to update while having a light framework.

PyTorch is selected due to it is an open-source machine learning library to create a deep neural network. It is able to simplify the creation of artificial neural network models and it could be used for AI applications. It is also pythonic in nature which is having the same coding style as Python. PyTorch is often comparable to TensorFlow, as TensorFlow has developed longer with a larger community of developers by Google. But PyTorch is having a little advantage over TensorFlow as it is easier and lighter to work with due to PyTorch is based on intuitive Python, therefore making it the best choice for this project.

Other than that, torchvision is selected alongside with PyTorch as it is able to smoothly integrate together to create a deep learning workflow.

Moreover, NLTK (Natural Language Toolkit) is selected for natural language processing in this project as it is a popular open-source library in Python. It provides a wide range of tasks, including tokenization, stemming, lemmatization, parsing, and sentiment analysis. All of the tasks above are needed to develop a well-functioning natural language processing application which is able to analyze text data.

Furthermore, the author decided to use Google Trans for this project as Google Trans is a free and unlimited Python library that implements Google Translate. It contains the features of having fast and reliable translation across multilanguage, auto language detection, and bulk translations.

Speech Recognition library is chosen to specifically identify spoken words and convert them into texts.

Lastly, PyAudio library is designated to capture sound and handle audio in Python. This library enables audio input and output across different platforms making it the perfect application in voice recognition and audio processing where capturing sound from a microphone is required. Most importantly, Because of its cross-platform interoperability, audio will function flawlessly across a range of operating systems, making it an essential and adaptable tool for any Python audio project.

### 3.3 Design Phase

This section of the report demonstrates how each feature and model are being implemented and design steps by steps into the chatbot.

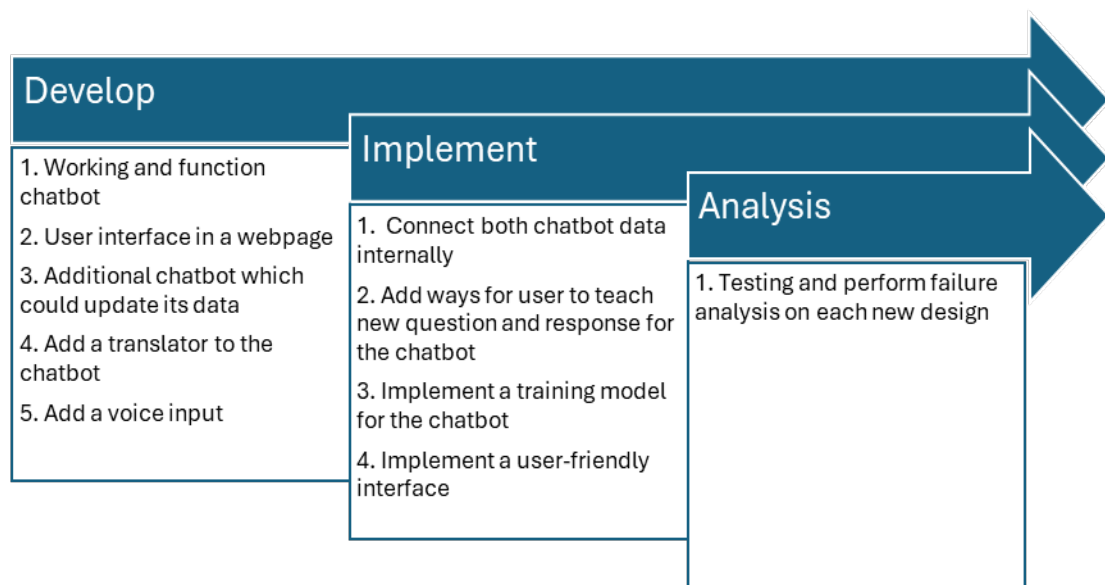


Figure 3.3: Design Process

From the Figure 3.3, each phase in the design process is based on completing all the development, implementing it in the chatbot, and then perform a failure analysis for each of the featured design until the project is successfully completed. Meanwhile, if there is an error, design flaw, or incompetence of design implementation found in the analysis phase, the error will be carefully examined and corrected before proceeding. This iterative process guarantees that any problems are resolved quickly and permits ongoing improvements to the chatbot's functionality and design until the finished product satisfies the necessary requirement.

### 3.3.1 Phase 1: Development

This subsection will cover the details of all the development in this project.

#### 3.3.1.1 Chatbot Protocol

The first module in the development of this project is to create a chatbot protocol. During this process, a very simple chatbot has been built with the intention of getting a desired response from a question.

```

5 def message_probability(user_message, recognised_words, single_response=False, required_words=[]):
6     message_certainty = 0
7     has_required_words = True
8
9     # Counts how many words are present in each predefined message
10    for word in user_message:
11        if word in recognised_words:
12            message_certainty += 1
13
14    # Calculates the percent of recognised words in a user message
15    percentage = float(message_certainty) / float(len(recognised_words))
16
17    # Checks that the required words are in the string
18    for word in required_words:
19        if word not in user_message:
20            has_required_words = False
21            break
22
23    # Must either have the required words, or be a single response
24    if has_required_words or single_response:
25        return int(percentage * 100)
26    else:
27        return 0

```

Figure 3.4: Code for Chatbot Protocol Part 1

Figure 3.4 shows how the chatbot is able to recognise word from the question asked by the user and count how many words are present in each predefined message. It will then Calculate the percent of recognised words in a user message and based on the percentage, it will return the highest probability set response from Figure 3.5.

```

30 def check_all_messages(message):
31     highest_prob_list = {}
32
33     # Simplifies response creation / adds it to the dict
34     def response(bot_response, list_of_words, single_response=False, required_words=[]):
35         nonlocal highest_prob_list
36         highest_prob_list[bot_response] = message_probability(message, list_of_words, single_response, required_words)
37
38     # Responses -----
39     response('Hello!', ['hello', 'hi', 'hey', 'sup', 'heyo'], single_response=True)
40     response('See you!', ['bye', 'goodbye'], single_response=True)
41     response('I'm doing fine, and you?', ['how', 'are', 'you', 'doing'], required_words=['how'])
42     response('You're welcome!', ['thank', 'thanks'], single_response=True)
43     response('Thank you!', ['i', 'love', 'code', 'palace'], required_words=['code', 'palace'])
44
45     # Longer responses
46     response(Long.R_ADVICE, ['give', 'advice'], required_words=['advice'])
47     response(Long.R_EATING, ['what', 'you', 'eat'], required_words=['you', 'eat'])
48
49     best_match = max(highest_prob_list, key=highest_prob_list.get)
50     # print(highest_prob_list)
51     # print(f'Best match = {best_match} | Score: {highest_prob_list[best_match]}')
52
53     return long.unknown() if highest_prob_list[best_match] < 1 else best_match
54
55
56 # Used to get the response
57 def get_response(user_input):
58     split_message = re.split(r'\s+|[,;?!.-]\s*', user_input.lower())
59     response = check_all_messages(split_message)
60     return response

```

Figure 3.5: Code for Chatbot Protocol Part 2

During this phase, the chatbot is only able to understand the user question based on similarity questions set in the code and response with the highest possibility.

### 3.3.1.2 User Interface

After creating a simple chatbot using Python, the next step will be to create a website using HTML, JavaScript, and CSS. The user interface is coded with each programming language respectively as shown in Appendix A, B, and C.

In Appendix A, the code defines a class named ‘Chatbox’ which manages a simple chat interface on a website. This class have a class constructor which initialized object such as chat box toggle button, chat interface, and the send button.

In the code, it has a property which tracks whether the chat box is open or closed, and it has an array which store the chat conversation.

Furthermore, when the chat box toggle button is clicked, it calls 'toggleState()' method, which switches the visibility of the chat box by toggling the 'chatbox--active' CSS class. When the user finished typing and click send button or trigger Enter key, the "onSendButton()" method will captures the user's input and appends both the user's message and the server's response to the 'message' array.

Appendix B shows the code for CSS, it's used to add style to a website and how the site is displayed on a browser. The author used this code to design the website with colour and how the chatbot in the website should be displayed.

Appendix C illustrates how HTML is coded to sets the language to English and ensuring correct text display by defining 'UTF-8'. The body of this code contains a 'div' with the class 'container' that hold the chatbot interface. There is a total of 3 div in the code, the first div 'chatbox' encapsulates the main elements of the chatbot, the second div 'chatbox\_\_messages' is where the chat messages will be dynamically displayed using JavaScript, the third div 'chatbox\_\_button' is where it contains a button to toggle the visibility of the chat box.

This project also uses HTML code to include a header with a greeting message, the name of the chatbot, and the purpose of the chatbot. There is also a footer written in this code which contains an input field for users to type messages and a send button which allows the user to send the messages to the chatbot.

### **3.3.1.3 Second Chatbot with Self-learning Features**

Chatbot with a learning feature is coded as shown in **APPENDIX F**. For this part, the code will first get the response from the user through the website, then it will handle the user input and respond by finding the best match in the JSON file. If during this

time, the code couldn't find a best match, then it will save the question asked by the user and prompt the user to give the corresponding answer to that question.

```

31 # Main function to handle user input and respond
32 def get_response2(msg):
33     knowledge_base: dict = load_knowledge_base('knowledge_base.json')
34
35     while True:
36         user_input: str = msg
37
38         # Finds the best match, otherwise returns None
39         best_match: str | None = find_best_match(user_input, [q["question"] for q in knowledge_base["questions"]])
40
41         if best_match:
42             # If there is a best match, return the answer from the knowledge base
43             a = 1
44             answer: str = get_answer_for_question(best_match, knowledge_base)
45             return answer, a
46         else:
47             a = 0
48             return "I don't have enough information on that right now. Could you provide more details?\n (Type 'new' to add into a new tag)\n (Type 'existing' to add into a existing tag)\n (Type 'skip' to skip)", a
49
50
51 def new_answer(msg, qsn):
52     if msg != 'skip':
53         knowledge_base: dict = load_knowledge_base('knowledge_base.json')
54         user_question: str = qsn
55         user_answer: str = msg
56         knowledge_base["questions"].append({"question": user_question, "answer": user_answer})
57         save_knowledge_base('knowledge_base.json', knowledge_base)
58         return "Thank you! I've learned something new."

```

Figure 3.6: Python Code Snippet for Chatbot Learning

Figure 3.6 shows how the code will perform when the user asked something new to the chatbot and how it will response to the new question.

### 3.3.1.4 Translator

This is fourth development in the chatbot, where the author added a translator into it, the full code is shown in Appendix H. This program is well performed by the 'googletrans' library where it is able to perform translation and language detection.

There is a total of two part in this program code, firstly is the 'translate\_to\_other()' function as shown in Figure 3.7. This part of the code is used to perform translation with a given English text into another language. The translated language is based on the user input and will be detected by 'googletrans' library.

```

5  def translate_to_other(eng_text, language):
6      translator = Translator()
7
8      # Translate the text to other language
9      foreign_text = translator.translate(eng_text, dest=language).text
10
11     return foreign_text

```

Figure 3.7: Code Snippet for Chatbot Translation from English to Other Language

The second part of the program code are ‘translate\_to\_eng()’ function as shown in Figure 3.8. This part of the code is used to translates text from any foreign language into English. Furthermore, it will detect the language that the user have input using the ‘detect()’ method, and it will stores the language into a variable. Then, it will translate the foreign text or sentence into English by setting the destination language to English in the ‘googletrans’ library. Once this is complete, the variable which stores the type of foreign language will then send ‘transfer\_to\_others()’ function which it will be able to translate English back to the language in which the user has inputted.

```

13  def translate_to_eng(foreign_language):
14      translator = Translator()
15
16      # Detect the language of the text and save it in variables
17      detection = translator.detect(foreign_language)
18      language = detection.lang
19
20      # Translate the text to English
21      eng_translate = translator.translate(foreign_language, dest='en').text
22
23     return language, eng_translate

```

Figure 3.8: Code Snippet for Chatbot Translation from Foreign Language to English

These two together will form the basis of translating the text when the chatbot detect the user have input a language that is not English. It will then translate the text from foreign language into English to understanding the question, and once it has the answer, it will translate the answer from English to the foreign language, the translated answer will then be output to the user.

### 3.3.1.5 Voice Input

The last major development for the chatbot will the voice input, and the code is shown in the Appendix G. This program begins by creating an instance of the

'Recognizer' class, which is responsible for processing the audio input by using microphone as the source of audio input. After processing, it will convert speech to text by using Google Speech Recognition. If successful, this program will save it in a variable named 'text' and it will return the recognized text to other function.

### **3.3.2 Phase 2: Implementation**

This subsection will cover the details of how each chatbot design are being implemented and integrated together.

#### **3.3.2.1 Data Connection Between Two Chatbot**

After the development of the first chatbot protocol, the author has upgraded the chatbot with a much more complicated programming code which allows the chatbot to have response to much more variety of questions and responses.

During this time, this project has entered a major challenge where the code for the main chatbot and the secondary chatbot which could be teach new information by the user. It seems like the difference between both code in both chatbot is unable to be integrated into one. Therefore, a new program code is written to allow data from JSON file of the secondary chatbot to be able to flow into the JSON file of the main chatbot.



```

1  {
2    "intents": [
3      {
4        "tag": "greeting",
5        "patterns": [
6          "Hi",
7          "Hey",
8          "How are you",
9          "Is anyone there?",
10         "Hello",
11         "Good day"
12       ],
13       "responses": [
14         "Hey :-)",
15         "Hello, good day to you.",
16         "Hi there, what can I do for you?",
17         "Hi there, how can I help?"
18       ]
19     },
20     {
21       "tag": "goodbye",
22       "patterns": [
23         "Bye",
24         "See you later",
25         "Goodbye",
26         "farewell",
27         "see you soon",
28         "catch you later"
29       ],
30       "responses": [
31         "See you later, thanks for visiting",
32         "Have a nice day",
33         "Bye!"
34       ]
35     }
36   ]
37 }

```

Figure 3.9: JSON File Code Snippet for Main Chatbot

```

1  {
2    "questions": [
3      {
4        "question": "What is the capital of Italy?",
5        "answer": "Rome"
6      },
7      {
8        "question": "What is the color of the sky?",
9        "answer": "Blue"
10     },
11     {
12       "question": "how are you",
13       "answer": "I'm doing great!"
14     }
15   ]
16 }

```

Figure 3.10: JSON File Code Snippet for Secondary Chatbot

Despite both chatbot using the same JSON file, but the format for both files is different as shown below in Figure 3.9 and Figure 3.10. The code in Figure 3.9 is having a structure where it consists of an array of intents, in which contains the 'tag', 'patterns', and 'response'. Firstly, the 'tag' in used to identify and categorise the intent. Followed by, 'patterns' in which is an array of phrases or patterns that user might say to trigger the corresponding 'tag'. Lastly, 'responses' is an array of potential responses that the chatbot will provide when the corresponding 'tag' is detected.

On the other hand, the code in Figure 3.10 is only having a particular question with a given response. The simplicity of this code is where it allows the

secondary chatbot to learn new questions and answers from the user without using much speed, time, and space.

In order to integrate code together for them to coexist, a new program code is being developed as shown in the Appendix J. The function of this code will be further explained in Section 3.3.2.2.

### 3.3.2.2 Chatbot Learning New Question and Response from User

There will be two scenarios when the chatbot encounter a brand-new question. First is where the chatbot have no clue what the question and response is, second is where the chatbot don't know the question but does know the response.

```

78 def add_new_intent(intents, new_tag, new_patterns, new_responses):
79     new_intent = {
80         "tag": new_tag,
81         "patterns": new_patterns,
82         "responses": new_responses
83     }
84     intents["intents"].append(new_intent)
85
86 def new_tag(newtag, newquestion, newresponse):
87     file_path = 'intents.json'
88     intents = read_intents_from_json(file_path)
89     new_tag = newtag
90
91     # Input new patterns
92     new_patterns = []
93     pattern = newquestion
94     new_patterns.append(pattern)
95
96     # Input new responses
97     new_responses = []
98     response = newresponse
99     new_responses.append(response)
100
101     # Add the new intent to the intents list
102     add_new_intent(intents, new_tag, new_patterns, new_responses)
103     new_answer(newresponse, newquestion)
104
105     # Save the updated intents back to the JSON file
106     save_intents_to_json(file_path, intents)
107
108     return (f"The new question with tag '{new_tag}' has been added.")

```

Figure 3.11: Python Code Snippet for Adding New Question and Response

Figure 3.11 shows a code snippet for the first scenario where the chatbot have encounter a new question without a clear response. This code allows the user to create a new tag as stated in Section 3.3.2.1. With this new 'tag' created, the program will save the new question and new responses taught by the user into the new tag.

```

57 def existing_tag(user_input, new_question):
58     file_path = 'intents.json'
59     intents = read_intents_from_json(file_path)
60     intent = read_intent_from_json(file_path)
61     user_input = get_tag_by_number(int(user_input))
62     response = get_first_response(intent, user_input)
63
64     if response:
65         a = 0 #done adding the new question into a existing tag
66         new_answer(response, new_question) #add the question and answer into the second chat
67         add_pattern_to_tag(intents, user_input, new_question) #add the question into the first chat
68         save_intents_to_json(file_path, intents)
69         return (f"The existing tag '{user_input}' has been added a new question '{new_question}'!"), a
70
71     else:
72         a = 1 #tag is not found
73         return (f"The tag '{user_input}' does not exist, kindly type again.\n (retype the 'tag' or 'skip' to skip)", a

```

Figure 3.12: Python Code Snippet for Adding New Question into Existing Tag

Figure 3.12 shows a code snippet where is used for the second scenario when the chatbot don't know the question but does have the response in the JSON file. This code allows the user to link the new question into a bunch of suitable response by inserting the new question into the corresponding 'tag' in the JSON file.

### 3.3.2.3 Chatbot Training Model

The main chatbot is being trained by the code shown in Appendix I. Before training the chatbot, a filter of NLTK is needed to allow the chatbot to understand the patterns from each 'tag' in Figure 3.9.

```

7  def tokenize(sentence):
8      |
9      return nltk.word_tokenize(sentence)
10
11
12  def stem(word):
13      """
14      stemming = find the root form of the word
15      examples:
16      words = ["organize", "organizes", "organizing"]
17      stemmed_words = [stem(w) for w in words]
18      -> ["organ", "organ", "organ"]
19      """
20      return stemmer.stem(word.lower())
21
22
23  def bag_of_words(tokenized_sentence, words):
24      pass
25      """
26      return bag of words array:
27      1 for each known word that exists in the sentence, 0 otherwise
28      example:
29      sentence = ["hello", "how", "are", "you"]
30      words = ["hi", "hello", "I", "you", "bye", "thank", "cool"]
31      bag = [ 0, 1, 0, 1, 0, 0, 0]
32      """
33      # stem each word
34      sentence_words = [stem(word) for word in tokenized_sentence]
35      # initialize bag with 0 for each word
36      bag = np.zeros(len(words), dtype=np.float32)
37      for idx, w in enumerate(words):
38          if w in sentence_words:
39              bag[idx] = 1
40
41      return bag

```

Figure 3.13: Python Code snippet for NLTK

Figure 3.13 shows a part of the NLTK code, which consists of three functions to preprocess text for tasks in the chatbot. Firstly, the ‘tokenize()’ function will be used to splits a sentence into an array of words, punctuation, or numbers by using the build in ‘nltk.word\_tokenize()’ function. If there is a input sentence “How are you?” by the user, this function will then split this sentence into an array of [“How”, “are”, “you”].

Continue by the ‘stem()’ function which is used to reduces words to their root form through stemming. This process is able to simplifies words like “organize”, “organizes”, and “organizing” to a common root, “organ”. This is very useful function as it helps to standardize words in different forms into a base representation.

Lastly, the ‘bag\_of\_words()’ function will be used to generates a bag of words representation. It will take a tokenized sentence from ‘tokenize()’ function and compares it to a predefined list of words. This function will create an array where each index corresponds to a word in the vocabulary; it assigns a 1 if the word is present in the sentence and a 0 if it is absent. For example, if there is a tokenized sentence [“hello”, “how”, “are”, “you”] and a predefined list of words [“hi”, “hello”,

“I”, “you”, “bye”, “thank”, “cool”], this function will return a bag which consist of [ 0, 1, 0, 1, 0, 0, 0] that corresponds to each word in the sentence.

This representation allows the model to quantify the presence or absence of important words in the input sentence, which is useful for tasks like classification or intent detection in chatbots.

Furthermore, before training the chatbot, a training model must be coded and specified. For this project, the training model code are shown in the Figure 3.14, which defines a simple feedforward neural network using PyTorch.

```

1 import torch
2 import torch.nn as nn
3
4
5 class NeuralNet(nn.Module):
6     def __init__(self, input_size, hidden_size, num_classes):
7         super(NeuralNet, self).__init__()
8         self.l1 = nn.Linear(input_size, hidden_size)
9         self.l2 = nn.Linear(hidden_size, hidden_size)
10        self.l3 = nn.Linear(hidden_size, num_classes)
11        self.relu = nn.ReLU()
12
13    def forward(self, x):
14        out = self.l1(x)
15        out = self.relu(out)
16        out = self.l2(out)
17        out = self.relu(out)
18        out = self.l3(out)
19        # no activation and no softmax at the end
20        return out

```

Figure 3.14: Python Code for Chatbot Training Model

The ‘`__init__()`’ function initialized the neural network into three layers, this three fully connected layers are labelled as ‘11’, ‘12’, and ‘13’. The purpose of the first layer ‘11’, is to maps the input features to the hidden layer with a size define in ‘`hidden_size`’. Subsequently the second layer ‘12’, are used to keeps the same hidden size for additional transformations. Finally, the third layer ‘13’, maps the hidden representation to the number of output classes.

The ‘`forward()`’ function in Figure 3.14 is used to when there is an input passed through it layer by layer. The ‘`ReLU()`’ activation function is applied after the first two layer to introduce non-linearity while is excluded for the output of the third layer, the output from the third layer will then returned directly which is where the final activation is handled separately.

After the completion of the NLTK and training model program code, these two program code file will then integrate with the code shown in Appendix I. The goal of these three-program code is to utilized PyTorch to train a chatbot model that would categorize user input into predetermined categories, or known as ‘tags’, by looking at the JSON file.

The first step in the data preparation process is to load the intents from the JSON file, then the code in Figure 3.13 will tokenize and stem each pattern’s words, and then compile a list of distinct stemmed words and their corresponding tags. To make processing this data easier, a special ‘ChatDataset’ class is made, which enables the model to get the data quickly using PyTorch’s ‘DataLoader’.

The neural network, which is an instance of the ‘NueralNet’ class, is configured with a given number of hidden neurons, output neurons, and input neurons. The purpose of the output neurons is to match the number of unique ‘tags’ from the JSON file, and the purpose of the input neurons is to match the length of the bag-of-words from the ‘bag\_of\_words()’ function from Figure 3.13. This model is well-suited for multiclass classification because it was trained using the CrossEntropyLoss function and the Adam optimizer.

During the 1000 epoch training loop, the model’s weights are changed to minimize the loss function, and the loss is displayed on a regular basis to track improvement.

#### **3.3.2.4 User-Friendly Interface**

This section will be used to describe how this project uses images and numbering to ease user interaction with the chatbot.



Figure 3.15: Image Used to Enhance User Experience

Figure 3.15 shows the image included in the chatbot website as it could potentially increase user engagement, facilitate communication and improve the user experience overall. Another consideration for using this icon is to allow user to find it easier to navigate an interface with visual components like this icon. Additionally, this icon can make the user experience more intuitive by leading them through important features without overburdening them with text, which ultimately allows the icon in website to communicate with the user more effectively than text alone.

Another common feature which could enhance user experience is to include numbering in the chatbot. Numbering is crucial because it facilitates the sequential organization of information, which makes it simpler for users to follow, comprehend and retain information. From the code shown in Figure 3.16, is to be designed to extract ‘tags’ from a JSON file and return them as a numbered list with HTML formatting.

```
4 def review_tag():
5     # Define the path to the JSON file
6     file_path = 'intents.json'
7     # Extract tags
8     with open(file_path, 'r') as file:
9         data = json.load(file)
10        tags = [intent["tag"] for intent in data["intents"]]
11        numbered_tags = [f"<strong>{i+1}</strong> {tag}" for i, tag in enumerate(tags)]
12
13    return numbered_tags
```

Figure 3.16: Python Code Snippet for Numbered List

The code starts by defining the path to the JSON file and then proceeds to open and read the file’s contents. By using Python’s ‘enumerate()’ function, the program is able to iterates through the tags extracted and allocating a number to each

tag. On top of that, HTML '<strong>' tags are used to format each tag, bolding the numbering and making sure the tags and number are visually striking.

### **3.3.3 Phase 3: Analysis**

Firstly, to make sure a chatbot works properly and offers a positive user experience, the author begins unit testing by examining individual components to make sure they function as intended, such as the user interface and chatbot integration. Subsequently, a simulation of full conversation flows which include voice input to do reliability test.

After completion of the development and implementation phase, the author also tested with different user inputs to make sure the chatbot responds to orders, greetings, and unclear questions. Besides that, an incorporate usability testing is being carried out by observing actual user's interactions with the chatbot and noting any issues to help discover areas that need to work.

Finally, regression testing is being done after updates and modifications to make sure that current functions are maintained.

## **3.4 Project Management**

The Gantt chart for FYP 1 and FYP2 is shown in Table 3.2 and Table 3.3 respectively.



**Table 3.2: The Gantt chart for FYP 1**

Activity	Weeks													
	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Initiation														
Registration and selection of supervisor														
Determination and confirmation of title														
Meeting with supervisor														
Planning & Execution														
Discussion about chapter 1														
Correction of chapter 1														
Discussion about literature review														
Submit chapter 2 and comment from supervisor														
Continuation of chapter 2														
Searching for online resources/knowledges														
Draft of chapter 3														
submit chapter 3														
Continuation of chapter 3														
Closure														
Submit finalise FYP1 report to supervisor														
Preparation for presentation														
Oral presentation of FYP project														

**Table 3.3: The Gantt Chart for FYP 2**

Activity	Weeks													
	1	2	3	4	5	6	7	8	9	10	11	12	13	14
FYP 2														
Initiation														
Meeting with Supervisor		■	■	■	■	■	■	■	■	■	■	■	■	■
Searching for online resources/knowledges	■	■	■	■	■	■	■	■	■	■	■	■	■	■
Planning & Execution														
Correction of chapter 3	■	■												
Programming Language Selection	■													
Desing Chatbot	■	■	■	■	■	■	■	■						
Desing User Interface			■	■	■	■	■	■						
Chatbot and UI Integregation									■	■	■	■	■	■
Chatbot Testing												■	■	■
Chapter 4							■	■	■	■	■	■	■	
Chapter 5												■	■	■
Closure														
Submit finalise FYP report to supervisor												■	■	
Preparation for presentation												■	■	
Oral presentation of FYP project													■	■

### 3.5 Cost Estimation

The project's estimated cost is given in this section. This section's goal is to establish the project's budget so that there will be enough money to construct the system. Additionally, the hardware and software requirements for this project are also shown in this section.

**Table 3.4: Cost Estimation of Project Materials**

Item	Cost (RM)
<b>Hardware:</b>	
<b>LAPTOP-MCVDORB1</b> <b>Processor: i3</b> <b>RAM: 8GB</b>	3200.00
<b>Software:</b>	
<b>Visual Studio Code</b>	0 (Free of Charge)
<b>Total Estimated Cost:</b>	3200.00

## CHAPTER 4

### RESULTS AND DISCUSSION

#### 4.1 Website Overview

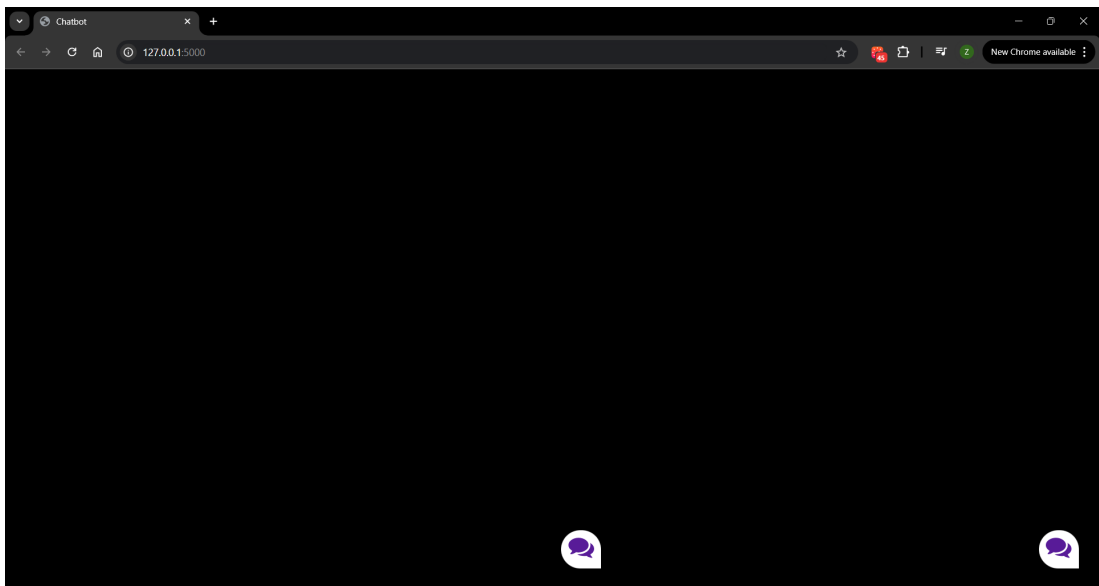


Figure 4.1: The Chatbot Website Part 1

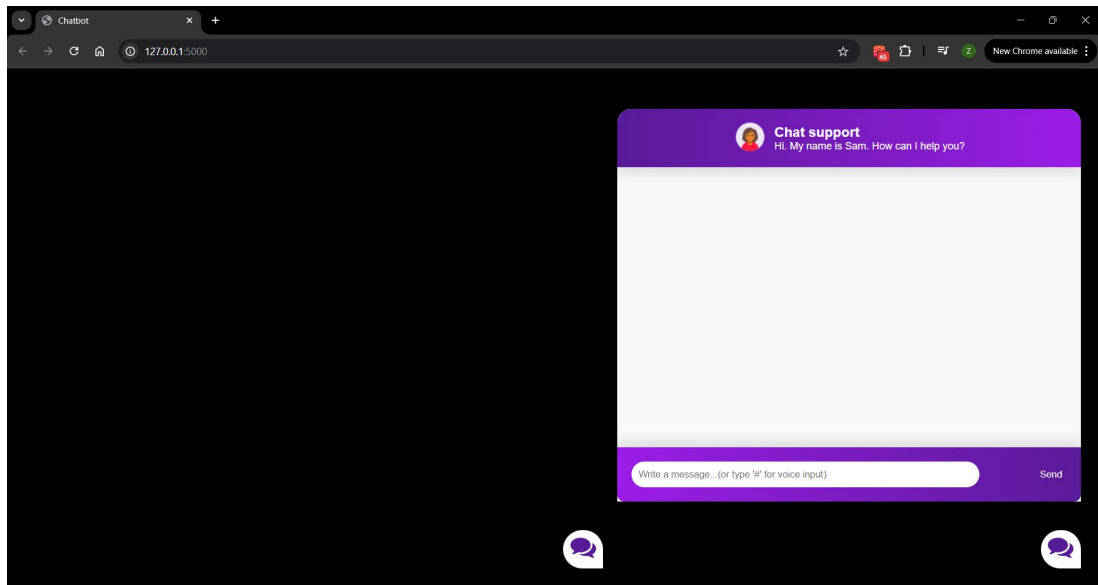


Figure 4.2: The Chatbot Website Part 2

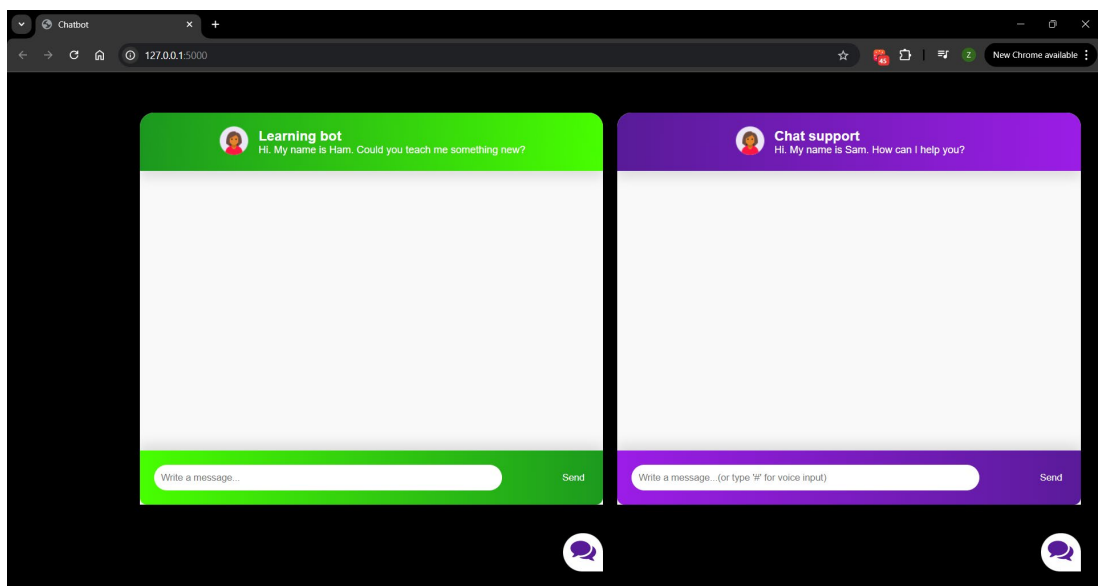


Figure 4.3: The Chatbot Website Part 3

Figure 4.1 shows the basic of the website by using HTML as the programming code, while Figure 4.2 and Figure 4.3 shows the website with animation and colour by integrating CSS and JavaScript with the HTML code.

On the right side of the Figure 4.3 is where the main chatbot will be located, and the secondary chatbot with the learning features is located at the left. The chat

box for both chatbot is colour coded differently to distinguish the difference between them.

## 4.2 Chatbot Overview

### 4.2.1 Main Chatbot

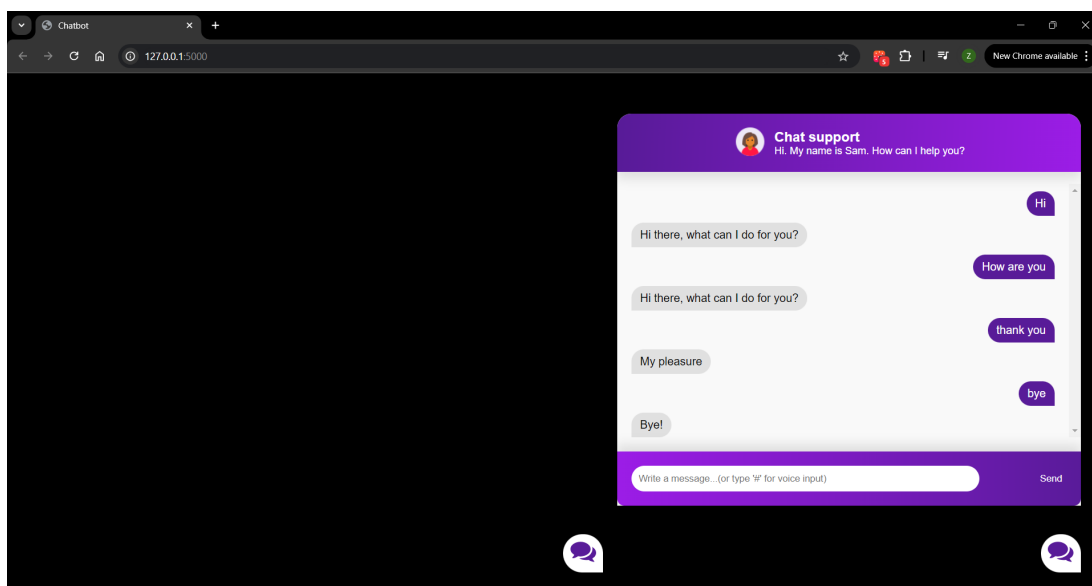


Figure 4.4: Chatbot Performing Simple Conversation with User

Figure 4.4 shows that the chatbot created in this project is able to perform simple communication with the user. This simple communication with the chatbot include greetings, farewell, and even telling a joke. All of this are possible, due to the training model shown in Section 3.3.2.3 and the JSON file shown in Figure 3.9.

But most importantly, the core of this project is to create a chatbot which able to answer legal queries regarding to user questions. This project is focused on Penal Code Act 574, Chapter XV, Offences relating to religion. This particular chapter in the Penal Code is having four sections with the following offences:

- i) Injuring or defiling a place of worship with intent to insult the religion of any class
- ii) Disturbing a religious assembly
- iii) Trespassing on burial places, etc.
- iv) Uttering words, etc., with deliberate intent to wound the religious feelings of any person

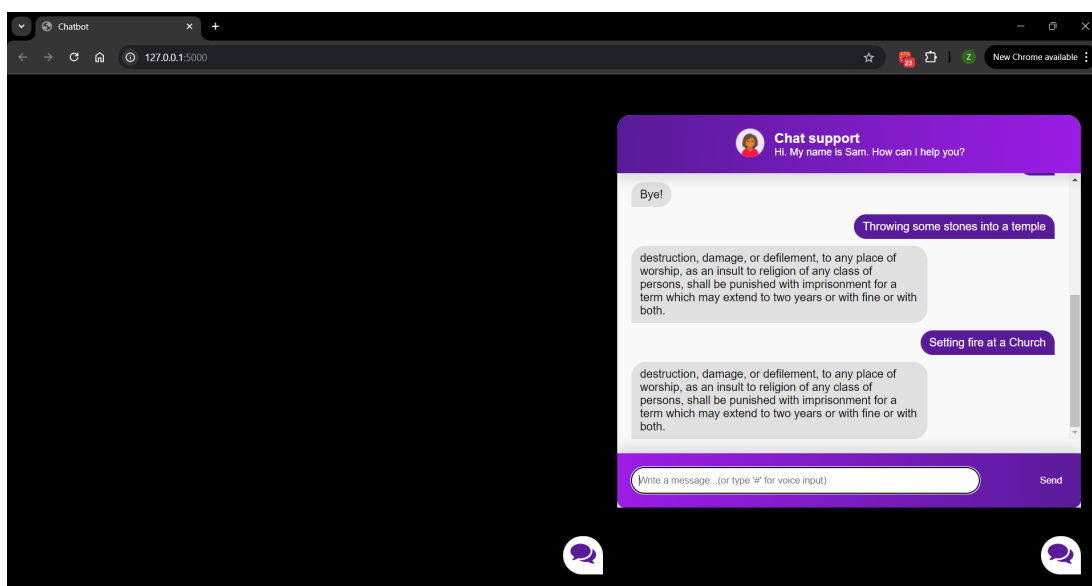


Figure 4.5: Result of Chatbot Response to Offense in First Section

Figure 4.5 shows the result of the chatbot from answering questions related to the first section (Injuring or defiling a place of worship with intent to insult the religion of any class). The chatbot is able to recognise that the user question contains a way of destroying, damaging, or defiling any places of worship, therefore it responded with the responsibility and consequences of the action.

Furthermore, the chatbot shown in Figure 4.6 demonstrates that it is able to understand the action done regardless of the place of worship. However, once the chatbot detected and understood that the damaged place described by the user is not any place of worship or places which does not suit the description by this section of

the Penal Code, then the chatbot will response with “I do not understand”. This scenario is shown in Figure 4.6

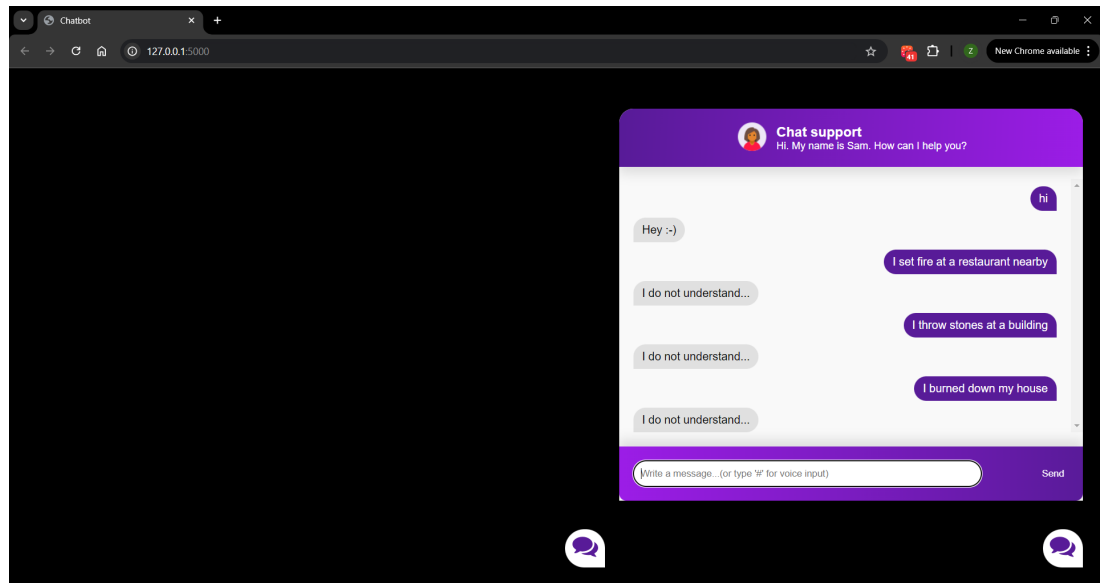


Figure 4.6: Result of Chatbot Responses to Unrelated Place

Furthermore, this chatbot also able to identify whether the action done at any place of worship fit into the description of Penal Code. For example, if the user asked similar questions as shown in Figure 4.7, where the action is not likely to do any harm, destroy, or damages to the place of worship, then the chatbot will be giving a response of “I do not understand” as well.



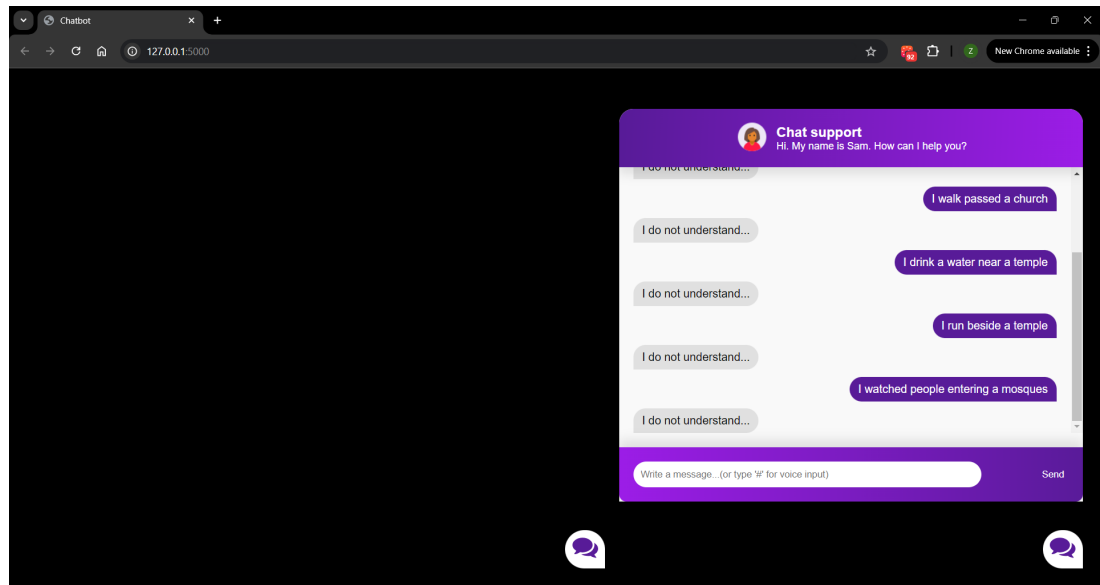


Figure 4.7: Result of Chatbot Responses to Neutral Action

Figure 4.8 and Figure 4.9 shows the chatbot response towards the questions from user, which is related to the second, third, and fourth sections from the Penal Code Act 754.

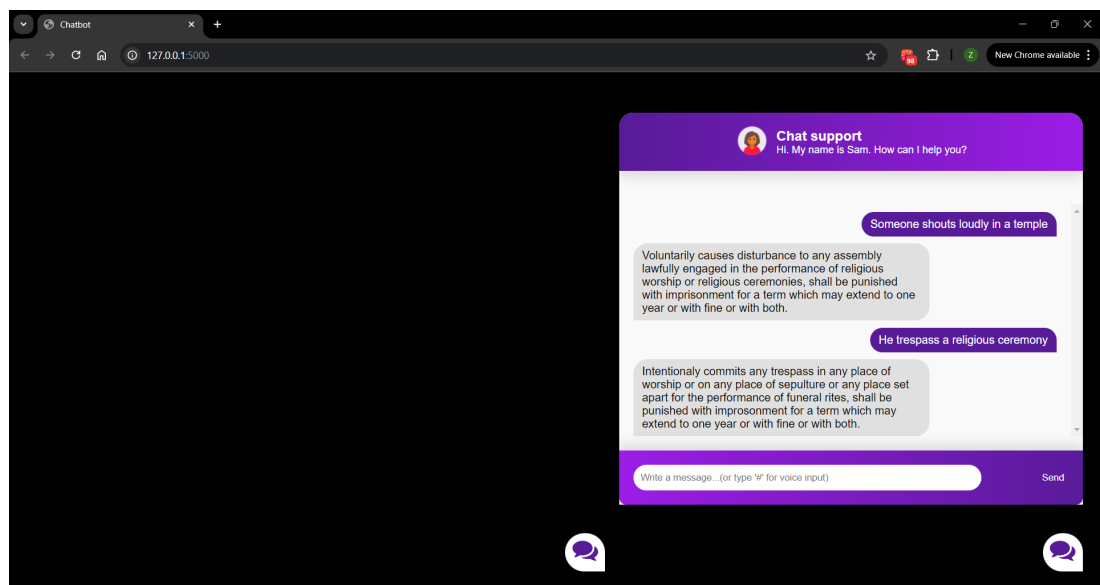


Figure 4.8: Result of Chatbot Response to Offense in Second and Third Section

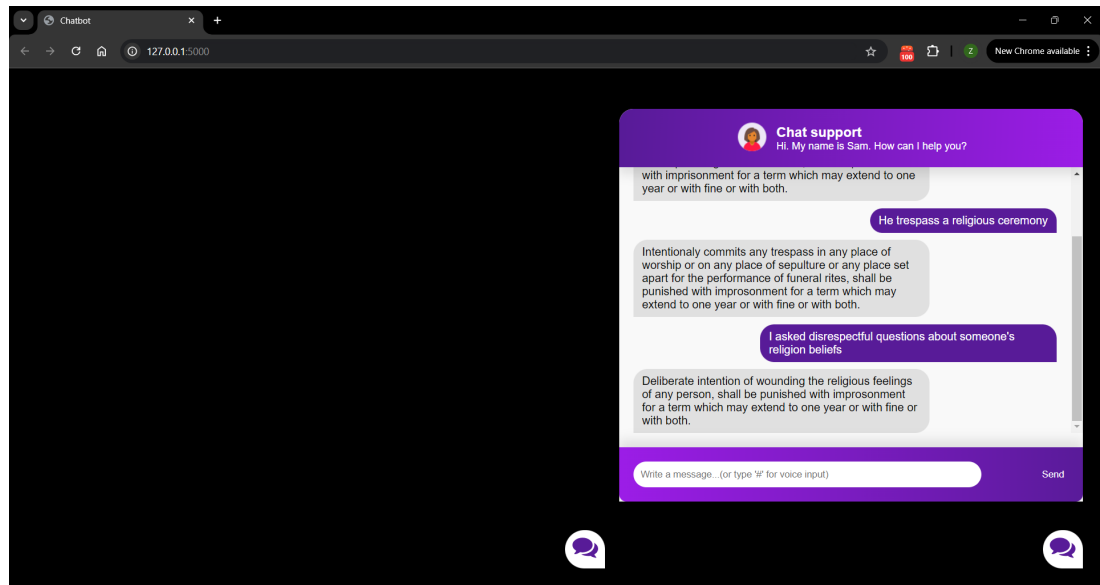


Figure 4.9: Result of Chatbot Response to Offense in Fourth Section

Last and most importantly, this chatbot is able to recognise good and positive action as shown in the Figure 4.10. This function is added to the chatbot because it helps boost morale and motivation by making people feel valued and appreciated. When the chatbot praise the user for excellent work, the chatbot increases user engagement and humanizes interactions. This encourages users to engage with the chatbot more frequently and fasters the development of trust and goodwill.

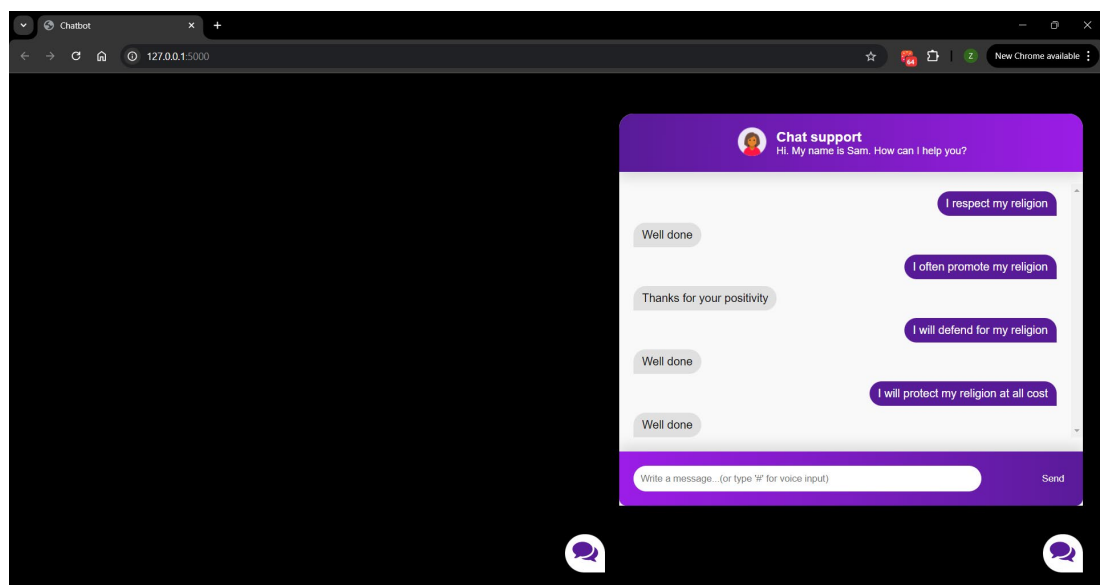


Figure 4.10: Chatbot Responses to Positive Action

### 4.2.2 Secondary Chatbot

This subsection will be showing the result of the second chatbot in this project. The purpose of the second chatbot is to allow the chatbot to learn from the user once it encounters a new question.

The need for a self-learning chatbot stems from the fact that it continuously enhances the user experience by getting better understanding and responding to users. These chatbots may learn from previous interactions, adjust to new information, and improve their responses over time to achieve more accurate, efficient, and personalized interactions by utilizing machine learning and natural language processing.

The second chatbot begins the conversation with a verification for the user to determine whether is a authorise user or an unknown user. For this project, only certified lawyer or professional bodies who have the knowledge and education in law will be given the ‘username’ and ‘password’ to authorise the second chatbot. By confirming that user is who they say they are and have the right qualifications, this approach helps to ensure a safe learning environment for the chatbot. As a result of this, the data that the chatbot stores and learns is kept consistent and of high quality.

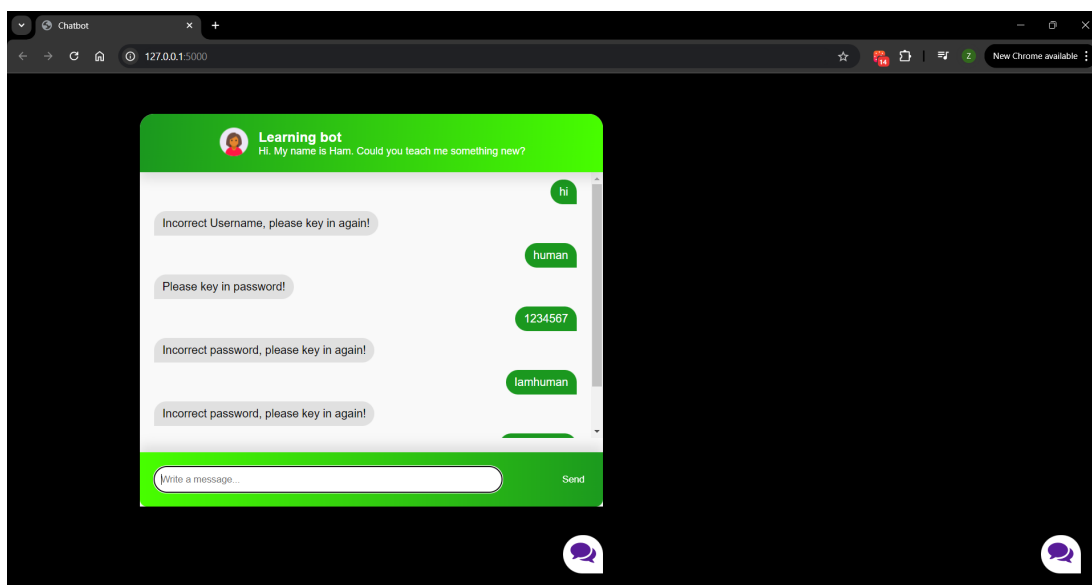


Figure 4.11: The Login Phase for The Secondary Chatbot

Figure 4.11 shows the login phase for the secondary chatbot. The chatbot will be giving “Incorrect Username” if the user key in the wrong username, and the chatbot will be giving “Incorrect Password” if the user key in the wrong password. When both of the username and password are correctly typed in the user, then only could the user interact with the chatbot as shown in Figure 4.12.

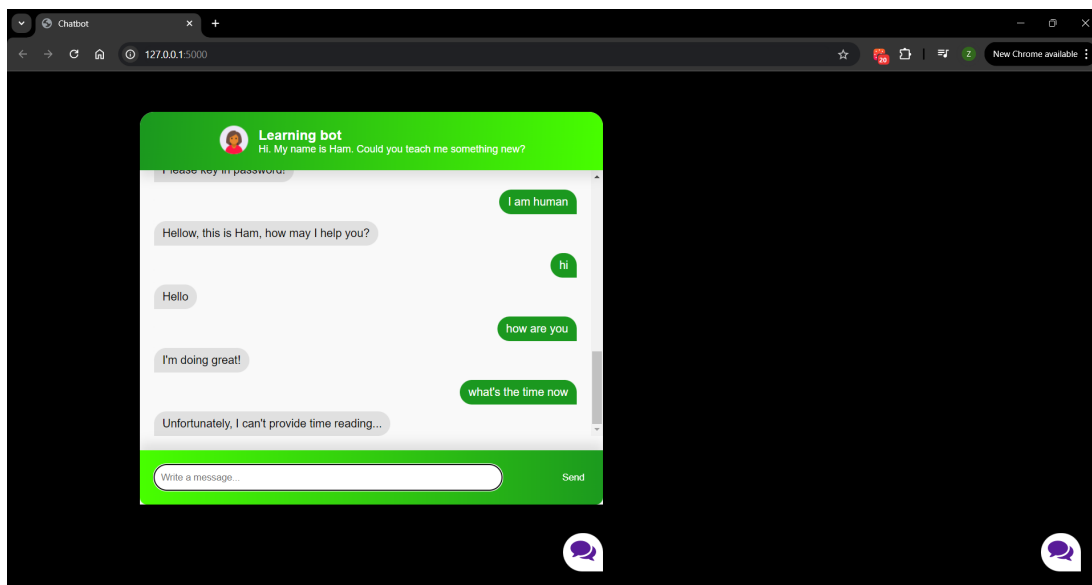


Figure 4.12: Secondary Chatbot Performing Simple Conversation with User

This secondary chatbot is used when a scenario shown in Figure 4.13 happen. During this case, the authorised professional bodies could repeat the question to the secondary chatbot, and the chatbot will be giving three options to the professional body. The first option will be adding the new question into a new ‘tag’ and a new answer as the code shown in Figure 3.11. The second option will be allowing the professional body to potentially add the question to existing ‘tag’ as the code shown in Figure 3.12, through this way, the user could link the new question into an existing ‘tag’ with an existing answer. Lastly, there also will be an option to allow the user to skip teaching a new question to the chatbot. The three options are shown in Figure 4.14.

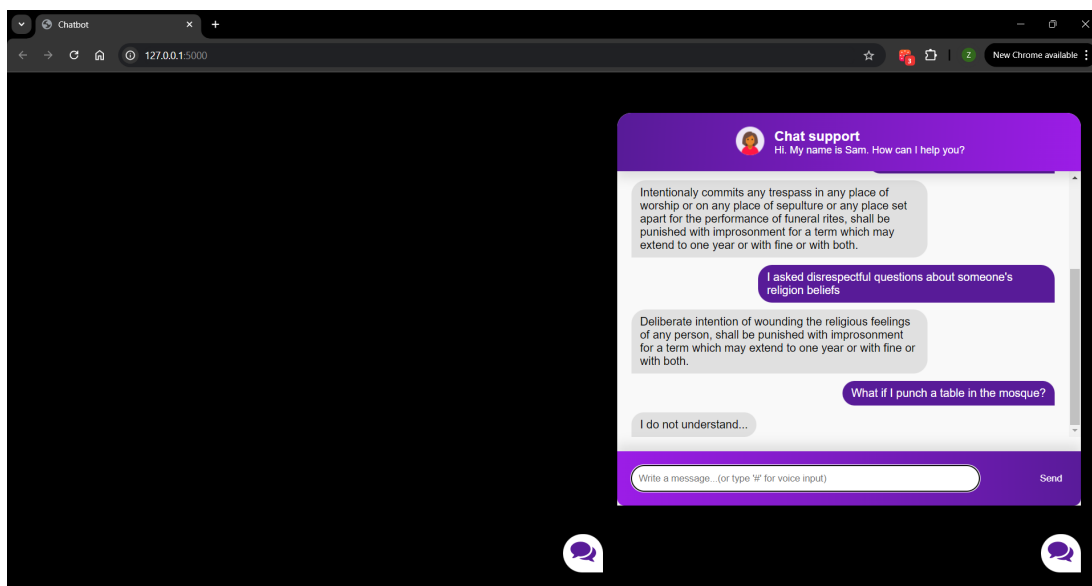


Figure 4.13: Chatbot Unable to Provide Answer to User Question

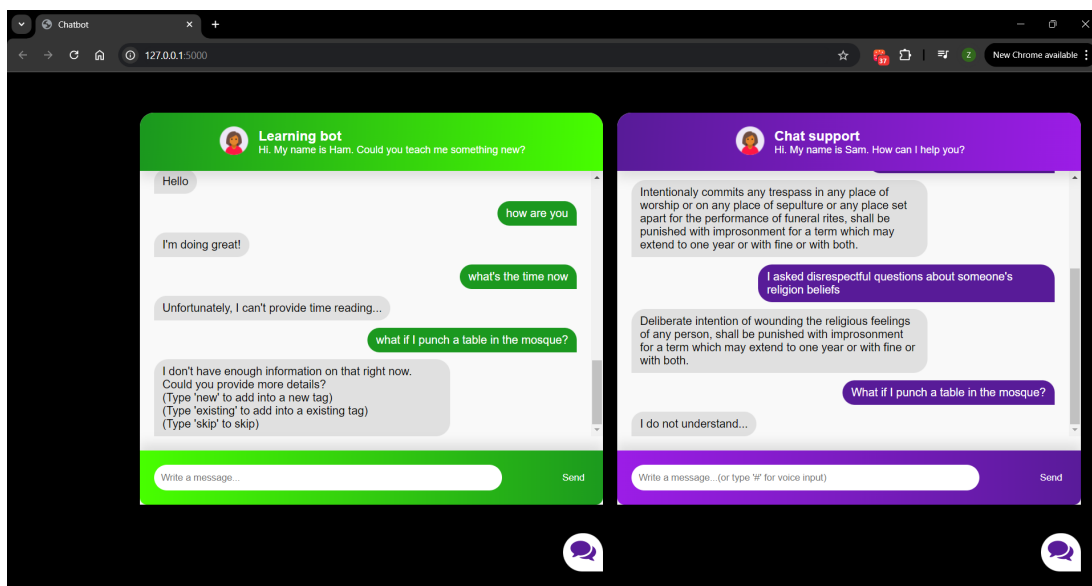


Figure 4.14: User Teaching Unknown Question to The Second Chatbot (i)

### 4.2.2.1 Adding New Questions

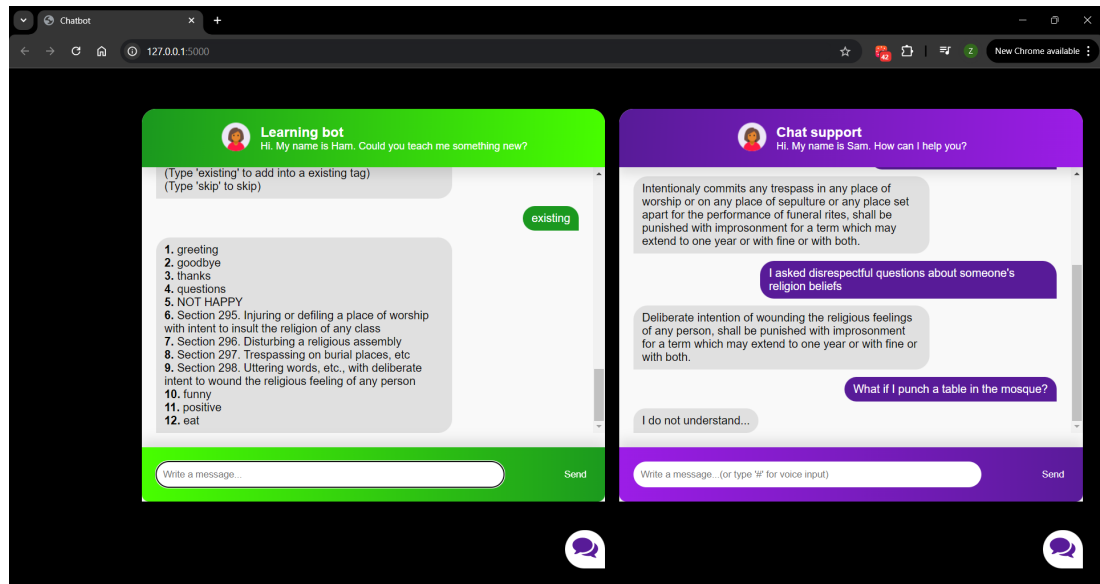


Figure 4.15: All Tag Listed in The Database

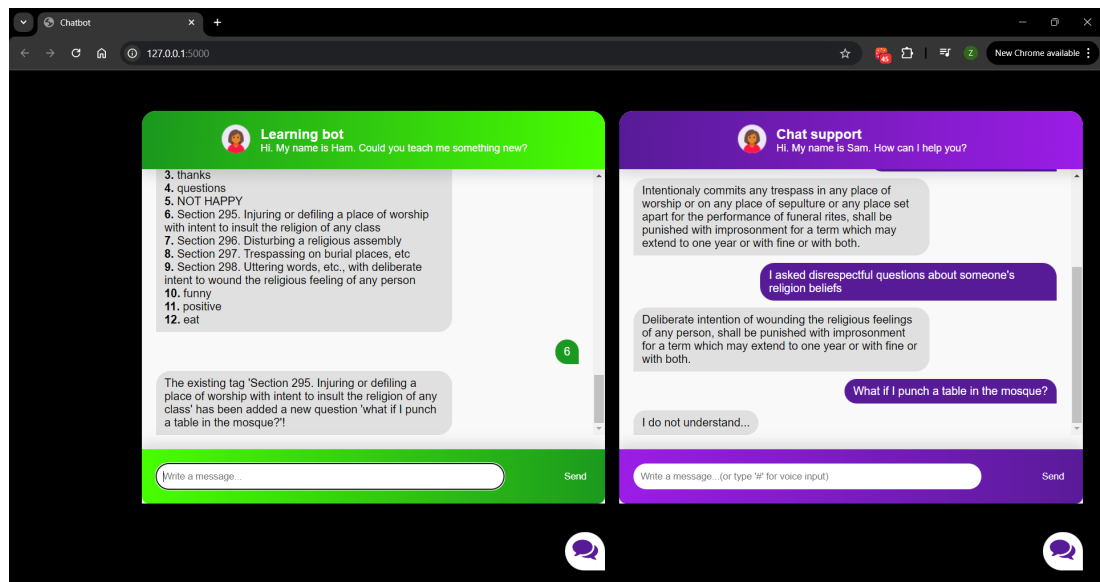


Figure 4.16: Result of New Question Added into an Existing Tag

Figure 4.15 shows the response given by the chatbot when the user decides to add a new question into an existing 'tag'. The chatbot will then show a numbering list of 'tag' in the database. Continue from that, the user could just type in the number that corresponds to the desired tag and the chatbot will automatically shows

a message similar to Figure 4.16, where the chatbot will display the information of the new question added and the tag in which the question has been added to.

#### 4.2.2.2 Adding New Questions and Response

This subsection is used to demonstrate when professional body teaches a new question with a new response to the chatbot. In Figure 4.17, the chatbot has encountered an unknown ethical question by the user, and for this scenario, the user decides to teach a suitable response by typing “new”.

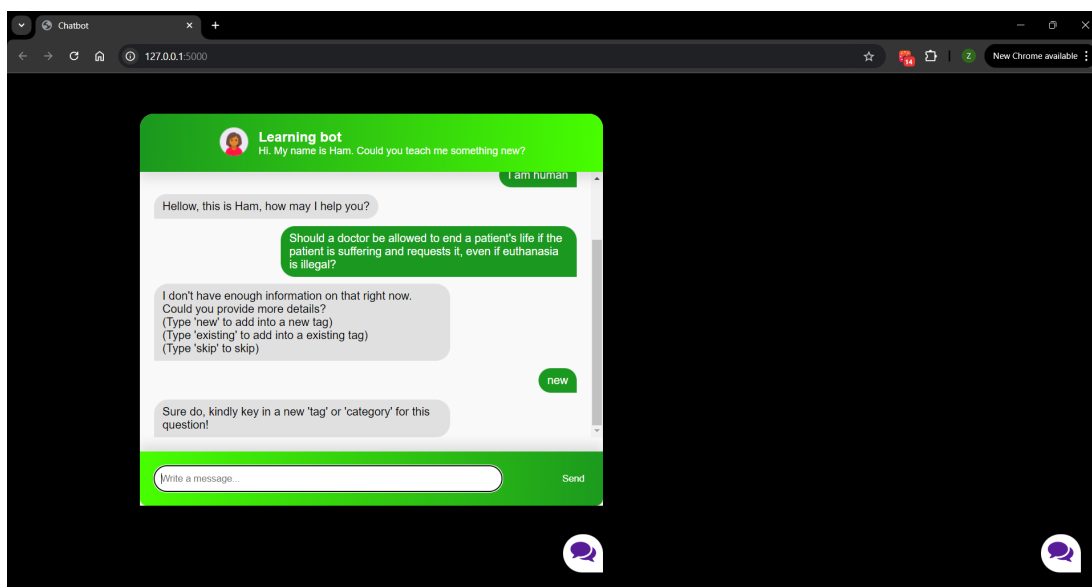


Figure 4.17: User Teaching Unknown Question to The Second Chatbot (ii)

In this project, if the user asked an ethical question about the following: “Is a doctor allowed to end a patient's life if the patient is suffering and requests it, even if euthanasia is illegal?”. The chatbot is unable to answer this question as it does not have similar question or data in its database. Hence, the user is going to teach the corresponding response towards this kind of question, the result is shown in Figure 4.18.

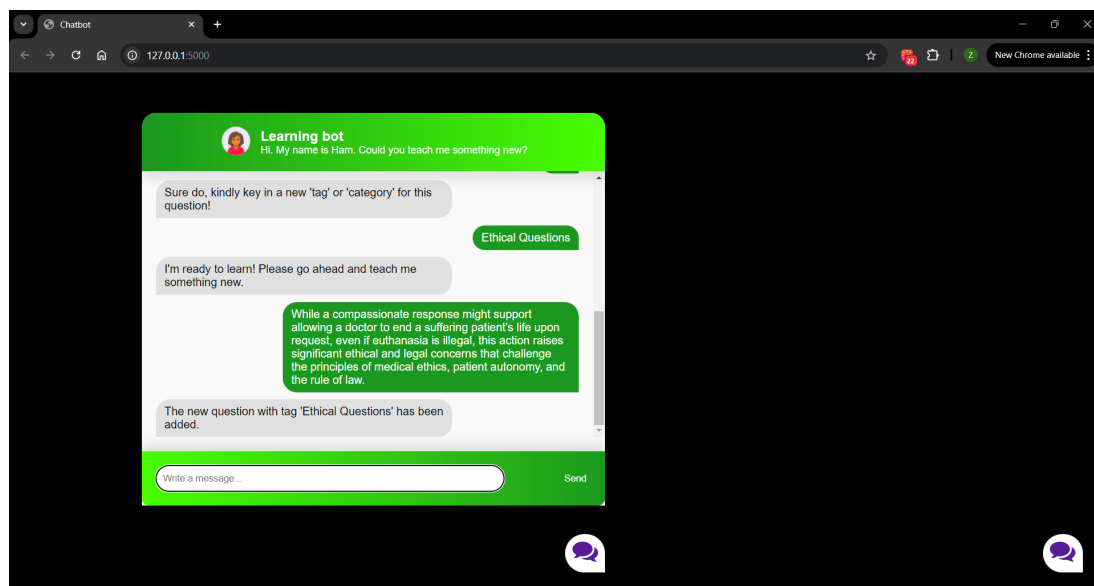


Figure 4.18: Result of Teaching Chatbot a New Question, Tag, and Answer

### 4.2.3 Data Integration Between Two Chatbot

When a new question has been completely added to the database, only the author could possibly train the chatbot in order for the main chatbot to learn the new question given to it. This is due to chatbot training necessitates technical proficiency in programming, machine learning, and natural language processing, developers are usually the ones permitted to train chatbots. Furthermore, the author has to ensure the chatbot train properly when handling sensitive data like legal inquiries, which guarantees process control, accuracy, and efficiency operation with few faults or unexpected consequences.

The output of the chatbot training in the terminal is shown in Figure 4.19. The program first prints the details about the dataset in the JSON file, such as 'tag', input patterns that the chatbot can recognize, and unique stemmed words used for training. During the training process, the model undergoes multiple epochs, with the code printing the epoch number and the corresponding loss value after every 100 epochs. The loss value is used to reflect how well the model is learning to predict the correct tags for the input sentence, and it is expected to decrease over time, indicating improved model performance. At the end of the training, the final loss



value is printed, showing how well the model has learned by the last epoch. Once the training is complete, the learned parameters of the model is saved to a file along with vocabulary and tags.



```

PS C:\Users\boeyz\Desktop\FYP1\code\Testing bot > python train.py
131 patterns
12 tags: ['NOT HAPPY', 'Section 295. Injuring or defiling a place of worship with intent to insult the religion of any class', 'Section 296. Disturbing a religious assembly', 'Section 297. Trespassing on burial places, etc.', 'Section 298. Uttering words, etc., with deliberate intent to wound the religious feeling of any person', 'eat', 'funny', 'goodbye', 'greeting', 'positive', 'questions', 'thanks']
269 unique stemmed words: ['re', 's', 'a', 'about', 'adjac', 'adopt', 'alter', 'and', 'ani', 'annoy', 'answer', 'anyon', 'appreci', 'are', 'area', 'artwork', 'as', 'ask', 'assembl', 'at', 'bad', 'base', 'belief', 'beliv', 'bore', 'break', 'buddhist', 'burn', 'by', 'bye', 'can', 'catch', 'caus', 'celebr', 'cemeteri', 'ceremoni', 'cherish', 'christi', 'an', 'church', 'claim', 'commot', 'compar', 'congreg', 'consent', 'contamin', 'corps', 'critic', 'cross', 'damag', 'day', 'dead', 'deceas', 'defac', 'defacin', 'defend', 'defil', 'deiti', 'deliber', 'derogatori', 'desecr', 'destroy', 'destruct', 'differnet', 'dig', 'directli', 'disparag', 'disrespect', 'disrupt', 'disturb', 'do', 'door', 'down', 'dump', 'dure', 'durin', 'eat', 'embrac', 'encourag', 'engag', 'enter', 'event', 'excess', 'faith', 'fals', 'farewel', 'feel', 'festiv', 'figur', 'film', 'fire', 'floor', 'follow', 'foolish', 'for', 'foster', 'from', 'front', 'funer', 'funni', 'garbag', 'gather', 'gestur', 'good', 'goodby', 'grave', 'group', 'gurdwana', 'harass', 'hate', 'heckl', 'held', 'hello', 'help', 'hey', 'hi', 'holi', 'honor', 'how', 'human', 'i', 'icon', 'if', 'in', 'indign', 'inflammatori', 'inform', 'insult', 'intent', 'intimid', 'is', 'it', 'jewelri', 'joke', 'kind', 'know', 'languag', 'later', 'locat', 'lot', 'loud', 'loudli', 'make', 'materi', 'me', 'meant', 'mock', 'monasteri', 'moral', 'mosqu', 'music', 'near', 'need', 'nois', 'n', 'on-religiou', 'now', 'object', 'observ', 'of', 'offens', 'offensiv', 'offer', 'on', 'or', 'other', 'outsid', 'outsid', 'overtur', 'particip', 'peopl', 'perform', 'permis', 'person', 'photograph', 'place', 'play', 'polit', 'practic', 'pray', 'prayer', 'preach', 'preserv', 'process', 'prohibit', 'promot', 'protect', 'protest', 'provid', 'public', 'punch', 'purpos', 'question', 'ralli', 'relic', 'religi', 'religion', 'remain', 'remov', 'respect', 'rever', 'rite', 'ritual', 'sac', 'safeguard', 'same', 'schedul', 'seat', 'see', 'servic', 'set', 'shout', 'sight', 'some', 'someoen', 'someone', 'someth', 'soon', 'speech', 'spray-paint', 'spread', 'stage', 'start', 'statu', 'steal', 'stone', 'suck', 'support', 'symbol', 'synagogu', 'tabl', 'take', 'tell', 'templ', 'term', 'text', 'thank', 'that', 'the', 'their', 'there', 'throw', 'time', 'to', 'tombston', 'toward', 'trespass', 'unf', 'avor', 'up', 'uplift', 'use', 'utter', 'valu', 'vandal', 'want', 'water', 'way', 'what', 'wheel', 'where', 'who', 'window', 'with', 'without', 'word', 'worship', 'worship', 'wors', 'hipp', 'wound', 'you']
269 12
Epoch [100/1000], Loss: 0.0635
Epoch [200/1000], Loss: 0.0025
Epoch [300/1000], Loss: 0.0009
Epoch [400/1000], Loss: 0.0001
Epoch [500/1000], Loss: 0.0001
Epoch [600/1000], Loss: 0.0001
Epoch [700/1000], Loss: 0.0000
Epoch [800/1000], Loss: 0.0000
Epoch [900/1000], Loss: 0.0000
Epoch [1000/1000], Loss: 0.0000
final loss: 0.0000
training complete. file saved to data.pth

```

Figure 4.19: Training Model Output

After the completion of training in Figure 4.19, the chatbot will then be able to learn and identify similar questions. As shown in Figure 4.20 and Figure 4.21, the main chatbot is able to correctly provide the response according to the user question.

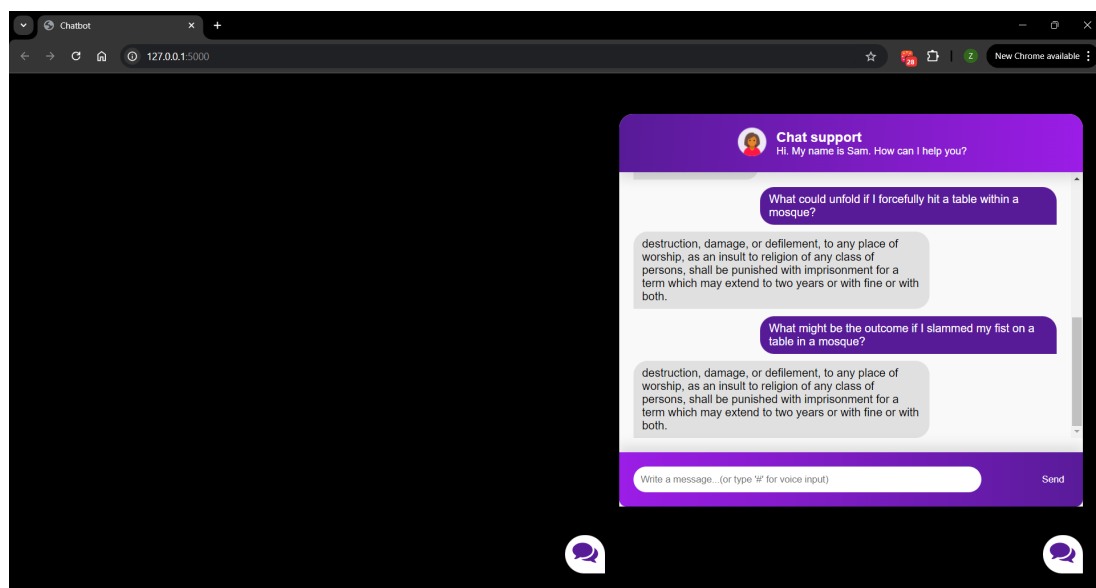


Figure 4.20: Result of Training (i)

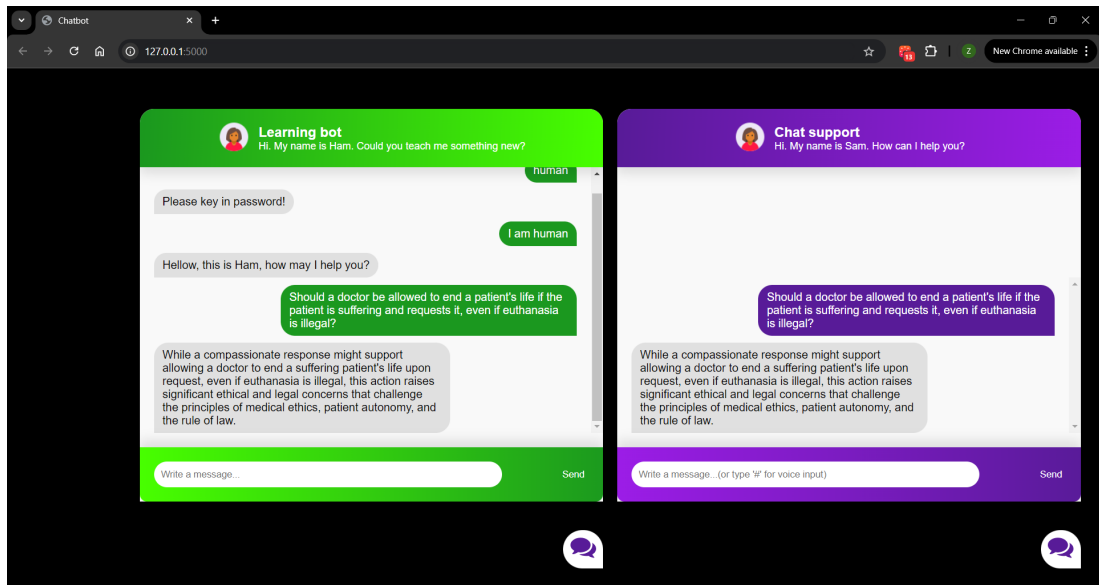


Figure 4.21: Result of Training (ii)

Table 4.1 shows the overall result for the learning process of the chatbot. The table is separated into two scenarios which correspond to Section 4.2.2.1 and Section 4.2.2.2.

Table 4.1: Chatbot Learning Process

Stages		Scenario	
		Adding New Question Only	Adding New Question and Response
1.	Before Training		

<p>2.</p>	<p><b>Training Phase</b></p>		
<p>3.</p>	<p><b>After Training</b></p>		

### **4.3 Chatbot Features**

This section will be discussing the features added to the chatbot. These added features will be used to enhance that chatbot's functionality, allowing it to handle a wider range of user queries and tasks. Moreover, incorporating new features ensures the chatbot remains up-to-date and capable of meeting contemporary demands.

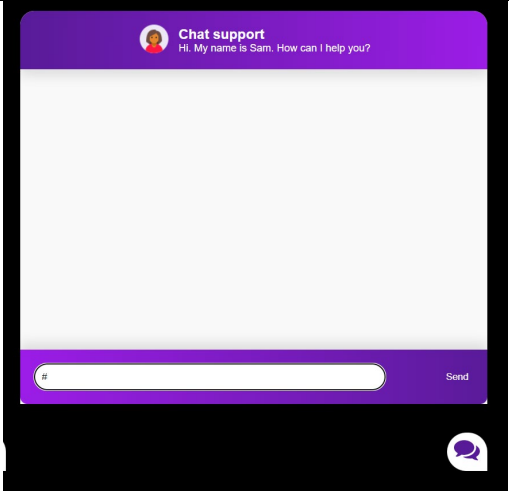
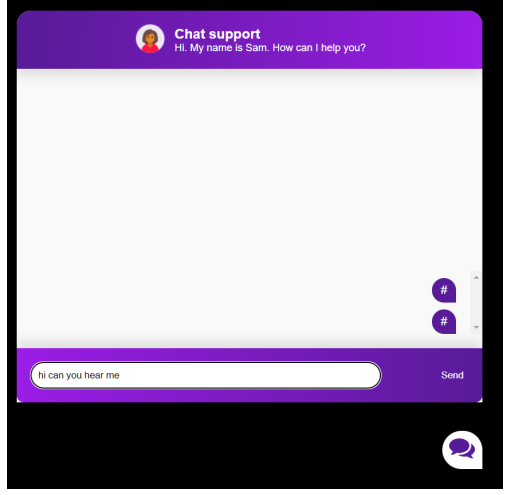
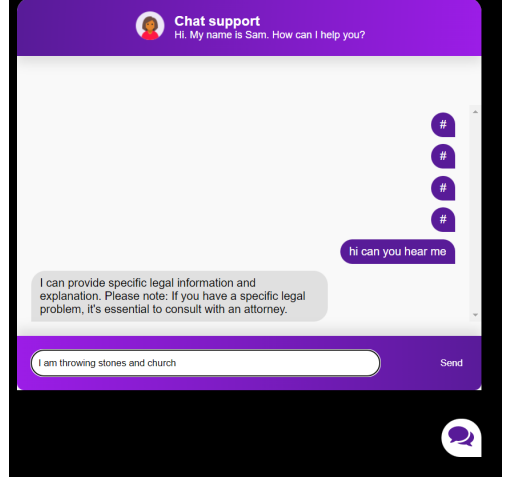
#### **4.3.1 Voice Input**

Voice input is chosen to be a feature for this chatbot due to several reasons, and the number one reason being that it is necessary in chatbots to improve user experience. First, speaking is typically quicker and more convenient than typing, particularly for people who find typing difficult or who may not be very skilled with a keyboard. Voice input is a more practical choice due to its ease of use, especially in mobile circumstances where typing on a small screen can be problematic.

The accessibility of voice input is yet another factor for adding it. It makes it possible for people with disabilities to effectively communicate with chatbots, especially for those who have vision impairments. Through adding this feature, this chatbot can reach a wider audience and guarantee that more people can use their services without using conventional input methods by supporting voice commands.

Moreover, users are able to gain the ability to multitask via voice input. For example, users can communicate with chatbots while walking, cooking, or working out. In circumstances when manual input could be distracting or impossible, this hands-free contact not only increases convenience but also safety.

Table 4.2: Process of Voice Input

Sequences	Actions	Results
1.	User Initiate voice input function	
2.	Users begin to talk to the mic	
3.	Chatbot able to recognise the voice inputted and provide the response	

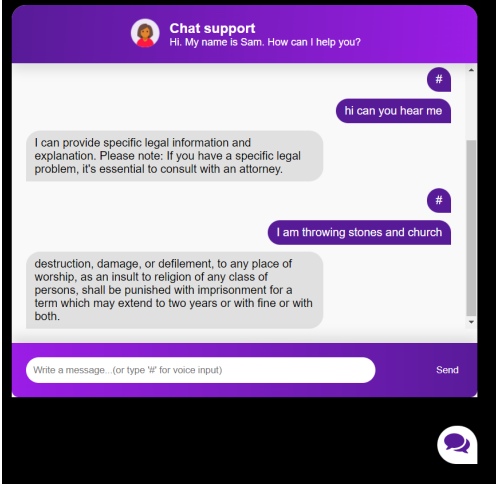
4.	Second example	
----	----------------	--

Table 4.2 shows the process of voice input in few steps. The first steps are where the user have to initiate the voice input by typing “#” into the chat box. After typing that, the user could begin to speak into the mic and the voice input will be then display in the chat bot as shown in the second step of the table. The voice input is not directly inputted to the chatbot but to show in the chat box is to allow the user to make changes if he or she doesn’t satisfy the outcome. The chatbot will then provide the respond when the user clicks enter or “send” to input to the chatbot, this process is shown in the third sequences of the table.

Lastly, the fourth sequence is used to demonstrate the functionality of the voice input with another example where the user voice inputted “I am throwing stones at a church”. But for this time, the voice input could not accurately recognise the word spoke by the user and instead of giving identical result, it gave “I am throwing stones and church”. Accents, dialects, speech difficulties, and noisy background may be difficult for voice recognition software to comprehend, which could result in incorrect interpretations or answers as shown here. In such case, users might prefer traditional text input methods to maintain discretion, clarity and allows user for greater control in asking question.

In this project, the chatbot is able to recognise the “typo” due to the conversion from voice to text and provide the correct response.

### 4.3.2 Multilanguage Text Translation

For a number of reasons, having multilingual text translation tool is essential in a chatbot. First off, it makes the chatbot more accessible and enables it to cater to a wider range of users with various linguistic backgrounds. This inclusivity makes sure that users who have different mother languages can communicate with the chatbot in a useful way, which increases the potential user base and its reach.

Other than that, this project added this to facilitate smooth communication for Malaysia's diverse population, which speaks Malay, Tamil, Mandarin, and other languages. By enabling users to communicate in the language of their choice, this could result in users being more willing to interact with and trust a chatbot that speaks in their native tongue.

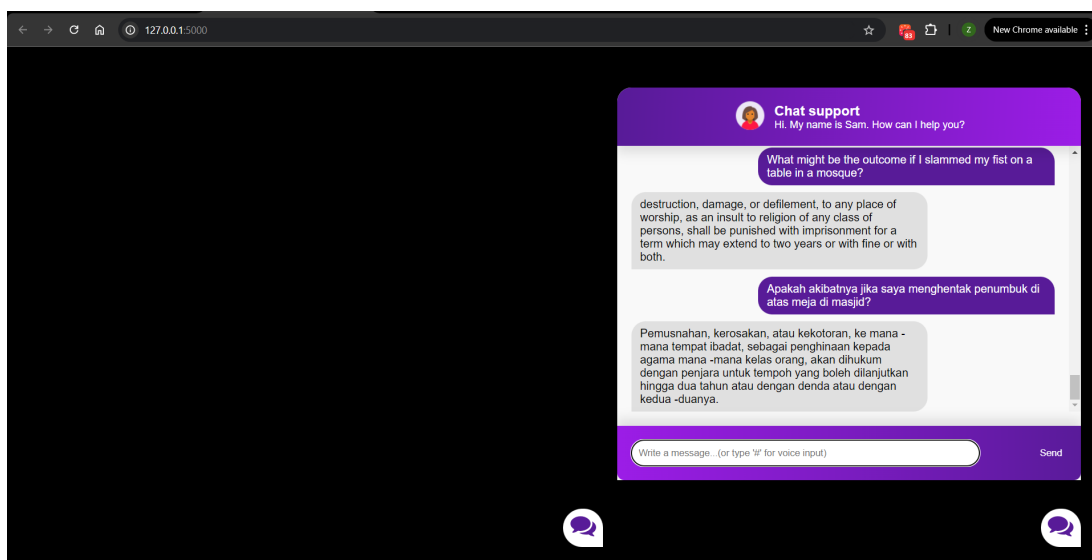


Figure 4.22: Result of Translation in Chatbot (i)

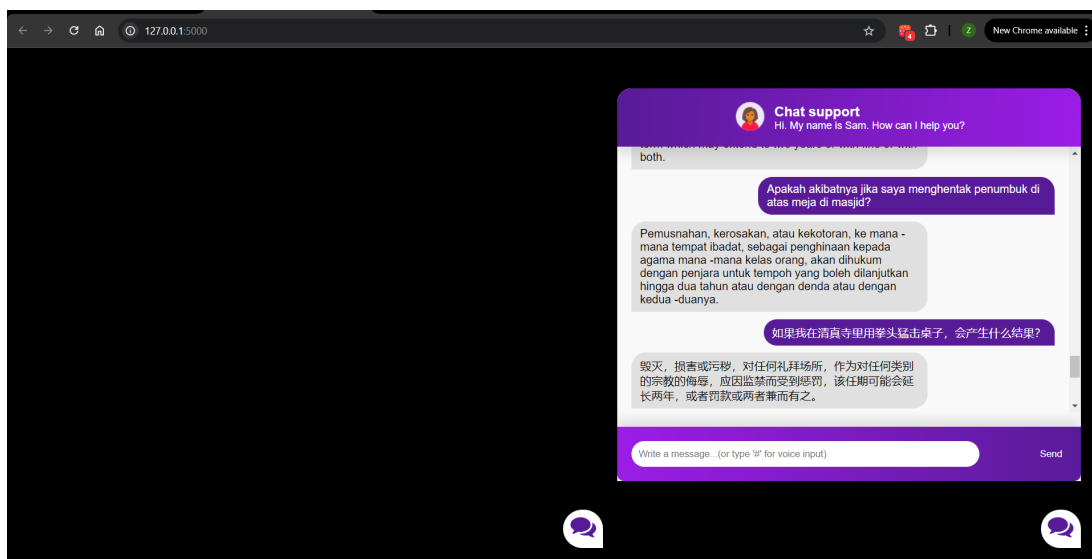


Figure 4.23: Result of Translation in Chatbot (ii)

Figure 4.22 and Figure 4.23 shows the result of using the function of translation in the chatbot. Figure 4.22 demonstrates the capability of the chatbot to understand and provide the same response when the question is asked in English and the exact same question is asked in Malay. This is also the case when the question is asked in Chinese as shown in Figure 4.23.

Using Google Translate in this project can be quite effective, but the confidence level for the translation is depends on various factors. Firstly, the language pair that is being translated is important, with a commonly spoken languages like English, Spanish, and French, Google Translate performs incredibly well and frequently yields correct translations.

The second factor which is considered by the author, is the continuous improvement of Google Translate. The correctness of the service gradually increases as a result of the continuous addition of new data.

But there are some limitations to Google Translate, especially when it comes to managing nuances. Firstly, the accuracy of the translation may decline for less widely spoken languages or those with intricate grammar and colloquial idioms.



Google Translate usually performs well enough for simple, uncomplicated sentences, but its accuracy varies when it comes to technical jargon, complex sentences, or sentences that need cultural context. In certain situations, there may be misconceptions since the translated content may not accurately convey the original meaning.

Last and most importantly, real-time translation is needed in this project settings, and Google Translate typically does this very well. But this do come with a drawback, which is delays.

Since Google Translate is a cloud-base service that needs an online connection to send and receive translation request, one of the main causes is internet access. Processing and returning the translated content may take longer if the internet connection is slow or having latency, which can be brought either by network congestion or large user-server separations.

Furthermore, response times might also be impacted by the volume and intricacy of the content being translated. Longer or more complicated sentences could take longer to parse, particularly if they contain difficult-to-translate terms like idioms, technical jargon, or context- dependent meanings.

Despite the occasional delays, the trade-offs are considered worthwhile due to the reliability and consistency of the Google Translates service. Delays are typically minimal and may not significantly affect the overall user experience in this project when real-time speed is not critical.

## 4.4 Chatbot Testing Result

This section will be used to discuss the training result for the chatbot in this project.

```
Epoch [100/1000], Loss: 0.0099
Epoch [200/1000], Loss: 0.0017
Epoch [300/1000], Loss: 0.0004
Epoch [400/1000], Loss: 0.0000
Epoch [500/1000], Loss: 0.0008
Epoch [600/1000], Loss: 0.0000
Epoch [700/1000], Loss: 0.0000
Epoch [800/1000], Loss: 0.0000
Epoch [900/1000], Loss: 0.0000
Epoch [1000/1000], Loss: 0.0000
final loss: 0.0000
training complete. file saved to data.pth
```

Figure 4.24: Chatbot Training Result

The training response shown in Figure 4.24 demonstrates the development of a machine learning model. An epoch is a single full run of the whole training dataset through the model, and each line shows the loss value at various epochs. A metric called the loss value is used to assess how well the model predicts the actual results; lower loss values denote greater performance.

In this response, the training began at epoch 100 with a loss of 0.0099 and reduced gradually over the course of the training. The loss was down to 0.0017 by epoch 200 and even lower to 0.0004 by epoch 300. As the epochs rose, this tendency persisted, demonstrating a notable decline in the loss value. The loss value ultimately dropped to 0.0000 from epoch 400 onward, suggesting that the model, at least based on the training data, was almost faultless in its predictions.

However, the model may have overfitted the training data, which means it has learned to predict the training data incredibly well but may not generalize as well to new, unseen data, given that the loss value approached zero.

## CHAPTER 5

### CONCLUSION AND RECOMMENDATIONS

#### 5.1 Conclusion

In conclusion, the three main objectives of this project have been successfully achieved. This first objective is to design an AI system that can effectively allow Malaysian to obtain more information about current penal code ACT 574 on chapter XV. This objective is obtained by allowing user to ask any unknown question which is related to this penal code to be answered by the chatbot created.

The second objective of this project is to implement a question-answering system that can respond to legal queries. This is completed by developing a system that could recognise the question asked by the user and provide known information as an output to the user. The chatbot database has included answer that is related to chapter XV of the penal code which could effectively respond to the user.

The last objective will be to design a platform for professional bodies to train the system. This is accomplished by setting up a secondary chatbot which will constantly request user to provide response to any unknown questions that it encounters. Furthermore, it also required a login system which is essential for ensuring that this chatbot is only getting highly accurate response from trusted professional bodies.

## 5.2 Recommendations for Future Improvement

This section will be giving recommendations for future improvement on this project. Firstly, is to increase the amount of data in the chatbot database. This is crucial as it enhances the chatbot's ability to understand and respond to a wider range of queries. Furthermore, it also allows more examples and scenarios to be learned by the chatbot, which could improve its accuracy and relevance in providing responses. This is to achieve a more engaging and satisfying user experience, as the chatbot can handle more complex and varied conversations. But most importantly, continuously adding data helps keep the chatbot's knowledge base up to date. This is essential for maintaining the chatbot's usefulness and reliability, especially in fields that are constantly evolving, such as legal queries.

Secondly, is to improve the quality of voice recognition for enhancing the user experience. This allows the user to speak freely and naturally without needing to repeat themselves or clarify their statements, leading to smoother and faster conversations. This is particularly beneficial for users who may find typing inconvenient, such as when they are on the move or have accessibility needs. Additionally, misinterpretations can lead to misunderstandings, incorrect answers, or frustration, reducing the chatbot's effectiveness and user satisfaction in scenarios where the chatbot is used for important tasks, such as legal advice. Most importantly, this is very helpful for those elderly users and individuals who have disabilities such as arthritis, reduced vision, or limited mobility can make typing on keyboards or using touchscreens difficult.

On top of that, having a "read out loud" function or chatbot voice output is significant as well. By providing auditory feedback, a chatbot with a read-aloud function ensures that visually impaired users can still access information and interact with digital services. It also assists users who might have difficulty reading due to dyslexia or other learning disabilities, offering them a more comfortable and inclusive way to engage with the content. Moreover, this feature could enhance multitasking and user convenience as they don't have to focus their visual attention on a screen in order to listen to responses while engaging in other tasks.

Lastly, future work can optimize the code of the chatbot in this project as is essential because it directly impacts the chatbot's performance, scalability, maintainability, and overall user experience. A seamless and engaging conversation may be maintained by the chatbot processing user questions more quickly when the code is well-optimized. This is because delays in response can irritate and dissatisfy users. Optimized code also consumes less memory and processing power, which is crucial for enabling the chatbot to function properly on hardware with constrained resources like embedded systems or smartphones. This could future improves scalability, enabling the chatbot to process more users and queries without setting a drop in performance. Finally, optimization facilitates better debugging and code maintenance, which makes it easier for developers to maintain and enhance the chatbot over time.

## REFERENCES

- Begum, A., Fatima, F. & Sabahath, A., 2019. Implementation of Deep Learning Algorithm with Perceptron using TensorFlow Library. *Communication and Signal Processing*, pp. 0172-0175.
- Bhbosale, S., Pujari, V. & Multani, Z., 2020. Advantages and disadvantages of artificial intelligence. *Aayushi International Interdisciplinary Research Journal*, Volume 77, pp. 227-230.
- Bi, H. H. & Zhao, J., 2004. Applying Propositional Logic to Workflow Verification. *Information Technology and Management*, Volume 5, p. 293–318.
- Brady, M., 1985. Artificial intelligence and robotics. *Artificial Intelligence*, 26(1), pp. 79-121.
- Bright, S. B., 2010. Legal Representation for the Poor: Can Society Afford This Much Injustice. *Mo. L. Rev.*, Volume 75, p. 683.
- Canastro, D. et al., 2019. The role of AI and automation on the future of jobs and the opportunity to change society. *New Knowledge in Information Systems and Technologies*, Volume 3, pp. 348-357.
- Coldwell, D. A. L., 2019. Negative Influences of the 4th Industrial Revolution on the Workplace: Towards a Theoretical Model of Entropic Citizen Behaviour In Toxic Organizations. *Int J Environ Res Public Health*, 16(15), p. 2670.
- Conitzer, V. & Sinnott-Armstrong, W., 2021. How much moral status could artificial intelligence ever achieve. *Rethinking moral status*, pp. 269-289.

- Crafts, N. F. R., 1996. The First Industrial Revolution: A Guided Tour for Growth Economists. *The American Economic Review*, 86(2), pp. 197-201.
- Duggal, N., 2024. Advantages and Disadvantages of Artificial Intelligence [AI]. *AI & Machine Learning*, 21 March.
- Gallant, S. I., 1990. Perceptron-based learning Algorithms. *IEEE TRANSACTIONS ON NEURAL NETWORKS*, 1(2), pp. 179-191.
- Goodfellow, I., Bengio, Y. & Courville, A., 2018. *Deep Learning*. s.l.:s.n.
- Hasmi, M. Z. et al., 2023. Artificial Intelligence Applications in Reduction of Carbon Emissions: Step Towards Sustainable Environment. *Frontiers in Environmental Science*, Volume 11, p. 1183620.
- Huang, M. H. & Rust, R. T., 2018. Artificial Intelligence in Service. *Journal of service research*, 21(2), pp. 155-172.
- Itamar Arel, Derek Rose & Thomas Karnowski, 2010. Deep Machine Learning - A New Frontier in Artificial Intelligence Research. *IEEE Comp. Int. Mag.*, 5(4), pp. 13-18.
- Kahr, P. K., Rooks, G., Snijders, C. C. P. & Willemsen, M. C., 2023. It seems smart, but it acts stupid: Development of trust in ai advice in a repeated legal decision-making task. *Intelligent User Interfaces*, pp. 528-539.
- Kaplan, D. L. & Claire, C. M., 1958. Occupational Trends in the United States, 1900 to 1950. *US Department of Commerce*, Volume 5.
- Khazode, K. C. A. & Sarode, R. D., 2020. Advantages and disadvantages of artificial intelligence and machine learning: A literature review. *International Journal of Library & Information Science (IJLIS)*, 9(1), p. 3.
- Lee, Y.-H., Wei, C.-P., Cheng, T.-H. & Yang, C.-T., 2012. Nearest-neighbor-based approach to time-series classification. *Decision Support Systems*, 53(1), pp. 207-217.

- Li, D. X., Viriyasaitavat, W., Ruchikachorn, P. & Martin, A., 2012. Using Propositional Logic for Requirements Verification of Service Workflow. *IEEE Transactions on Industrial Informatics*, 8(3), pp. 639-646.
- Lucas, R. E., 2002. The industrial revolution: Past and future. *Lectures on economic growth*, Volume 109, p. 188.
- Madakam, S., Holmukhe, R. M. & Jaiswal, D. K., 2019. The Future Digital Work Force: Robotic Process Automation. *Journal of Information System Technology Management*, Volume 16.
- Makridakis, S., 2017. The Forthcoming Artificial Intelligence (AI) Revolution: Its Impact on Society and Firms. *Futures*, Volume 90, pp. 46-60.
- Manyika, J. et al., 2017. harnessing automation for a future that works. pp. 2-4.
- Masek, J. et al., 2018. *Harmonized Landsat/Sentinel-2 Products for Land Monitoring*. Valencia, IGARSS 2018-2018 IEEE international geoscience and remote sensing symposium, pp. 8163-8165.
- Mhlanga, D., 2023. Exploring the Evolution of Artificial Intelligence and the Fourth Industrial Revolution an Overview. In: *FinTech and Artificial Intelligence for Sustainable Development*. s.l.:Palgrave Macmillan, Cham, pp. 15-39.
- Michael, K. & Martin, I., 2017. Industrialization and De-industrialization in Southeast Europe, 1870-2010. In: *The Spread of Modern Industry to Periphery since 1871*. Oxford: Oxford Academic, pp. 91-114.
- Mohajan, H. K., 2011. Greenhouse gas emissions increase global warming. *International Journal of Economic and Political Integration*, 1(2), pp. 21-34.
- Mohajan, H. K., 2018. Aspects of Mathematical Economics, Social Choice and Game theory. October, pp. 1-247.
- Mohajan, H. K., 2019. The First Industrial Revolution: Creation of a New Global Human Era. *Munich Personal RePEc Archive*, 5(4), pp. 377-387.



- Mohajan, H. K., 2021. Third Industrial Revolution Brings Global Development. *Journal of Social Sceinces and Humanities*, 7(4), pp. 239-251.
- Mokyr, J. & Strotz, R. H., 1998. The Second Industrial Revolution, 1870-1914. In: *Storia dell'economia Mondiale*. s.l.:s.n., pp. 219-245.
- Mucherino, A., Papajorgji, P. J. & Pardalos, P. M., 2009. k-Nearest Neighbor Classification. In: *Data Mining in Agriculture*. New York: Springer, p. 83–106.
- Mughal, A. A., 2018. Artificial Intelligence in Information Security: Exploring the Advantages, Challenges, and Future Directions. *Journal of Artificial Intelligence and Machine Learning in Management*, 2(1), pp. 22-34.
- Murashov, V., Hearl, F. & Howard, J., 2016. Working safely with robot workers: Recommendations for the new workplace. *Journal of Occupational and Environmental Hygiene*, 13(3), pp. D61-D71.
- Niku, S. B., 2020. *Introduction to Robotics: Analysis, Control, Applications*. 3rd ed. s.l.:John Wiley & Sons.
- Philbeck, T. & Davis, N., 2019. THE FOURTH INDUSTRIAL REVOLUTION: SHAPING A NEW ERA. *Journal of International Affairs*, 72(1), pp. 17-22.
- Puri, A., 2020. Moral imitation: Can an algorithm really be ethical?. p. 47.
- Roberto, N. & Ingrid, P., 2015. The Third Industrial Revolution. In: *Advanced Customizatoin in Architectural Design and Construction*. s.l.:SPRINGER LINK, pp. 7-27.
- Roberts, B., 2015. The Third Industrial Revolution: Implications for Planning Cities and Regions. *Workiing Paper Urban Frontiers*, Volume 1.
- Samuel, A. L., 1959. Some Studies in Machine Learning Using the Game. *IBM Journal of Research and Development*, 3(3), pp. 210-229.
- Santiso, C., 2018. Can blockchain help in the fight against corruption. *World Economic Forum*, Volume 12.

- Schmidhuber, J., 2015. Deep learning in neural networks: An overview. *Neural networks*, Volume 61, pp. 85-117.
- Schwab, K., 2017. *The Fourth Industrial Revolution*. s.l.:Crown Currency.
- Smil, V., 2005. Internal Combustion Engines. In: *Creating the Twentieth Century: Technical Innovations of 1867-1914 and Their Lasting Impact*. New York: Oxford Academic, pp. 98-151.
- Tan, T. B. & Wu, S. S., 2017. The Fourth Industrial Revolution Explained. In: *PUBLIC POLICY IMPLICATIONS OF THE FOURTH INDUSTRIAL REVOLUTION FOR SINGAPORE*. s.l.:S. Rajaratnam School of International Studies, pp. 5-7.
- Tomasev, N. et al., 2020. AI for social good: unlocking the opportunity for positive impact. *Nature Communications*, 11(1), p. 2468.
- Unold, O., 2005. Context-free grammar induction with grammar-based classifier system. *Archives of Control Science*, 15(4), p. 681.
- Velarde, G., 2019. Artificial Intelligence and its impact on the Fourth Industrial Revolution: A Review. *International Journal of Artificial Intelligence & Applications*, Volume 10.
- Vinuesa, R. et al., 2020. The role of artificial intelligence in achieving the Sustainable Development Goals. *Nature communications*, 11(1), pp. 1-10.
- Vries, J. d., 2009. The Industrial Revolution and the Industrious Revolution. *The Journal of Economic History*, 54(2), pp. 249-270.
- Wang, S., Huang, Q., Jiang, S. & Tian, Q., 2010. *Nearest-neighbor classification using unlabeled data for real world image application*. New York, Association for Computing Machinery, pp. 1154 - 1154.
- Wang, X., Xu, Z. & Xunjie, G., 2020. A novel plausible reasoning based on intuitionistic fuzzy propositional logic and its application in decision making. *Fuzzy Optim Decis Making*, Volume 19, p. 251–274.

Xu, M., David, J. M. & Suk, H. K., 2018. The Fourth Industrial Revolution: Opportunities and Challenges. *International Journal of Financial Research*, 9(2).

Yellin, D. M. & Weiss, G., 2021. *Synthesizing Context-free Grammars from Recurrent Neural Networks*. s.l., Springer, pp. 351-369.

## APPENDICES

### APPENDIX A: JavaScript Code for User Interface

```
class Chatbox {
  constructor() {
    this.args = {
      openButton: document.querySelector('.chatbox__button'),
      chatBox: document.querySelector('.chatbox__support'),
      sendButton: document.querySelector('.send__button')
    }
  }

  this.state = false;
  this.messages = [];
}

display() {
  const {openButton, chatBox, sendButton} = this.args;

  openButton.addEventListener('click', () => this.toggleState(chatBox))

  sendButton.addEventListener('click', () => this.onSendButton(chatBox))

  const node = chatBox.querySelector('input');
  node.addEventListener("keyup", ({key}) => {
    if (key === "Enter") {
      this.onSendButton(chatBox)
    }
  })
}
```

```
    }
  })
}

toggleState(chatbox) {
  this.state = !this.state;

  // show or hides the box
  if(this.state) {
    chatbox.classList.add('chatbox--active')
  } else {
    chatbox.classList.remove('chatbox--active')
  }
}

onSendButton(chatbox) {
  var textField = chatbox.querySelector('input');
  let text1 = textField.value
  if (text1 === "") {
    return;
  }

  let msg1 = { name: "User", message: text1 }
  this.messages.push(msg1);

  fetch('http://127.0.0.1:5000/predict', {
    method: 'POST',
    body: JSON.stringify({ message: text1 }),
    mode: 'cors',
    headers: {
      'Content-Type': 'application/json'
    },
  })
  .then(r => r.json())
```

```

.then(r => {
  if (text1 === '#') {
    textField.value = r.answer;
  } else {
    let msg2 = { name: "Sam", message: r.answer };
    this.messages.push(msg2);
    this.updateChatText(chatbox);
    textField.value = "";
  }

}).catch((error) => {
  console.error('Error:', error);
  this.updateChatText(chatbox)
  textField.value = ""
});
}

updateChatText(chatbox) {
  var html = "";
  this.messages.slice().reverse().forEach(function(item, index) {
    if (item.name === "Sam")
    {
      html += '<div class="messages__item messages__item--visitor">' +
item.message + '</div>'
    }
    else
    {
      html += '<div class="messages__item messages__item--operator">' +
item.message + '</div>'
    }
  });

  const chatmessage = chatbox.querySelector('.chatbox__messages');
  chatmessage.innerHTML = html;
}

```

```
    }  
  }  
  
class Chatbox2 {  
  constructor() {  
    this.args = {  
      openButton: document.querySelector('.chatbox__button2'),  
      chatBox: document.querySelector('.chatbox__support2'),  
      sendButton: document.querySelector('.send__button2')  
    }  
  
    this.state = false;  
    this.messages = [];  
  }  
  
  display() {  
    const {openButton, chatBox, sendButton} = this.args;  
  
    openButton.addEventListener('click', () => this.toggleState(chatBox))  
  
    sendButton.addEventListener('click', () => this.onSendButton(chatBox))  
  
    const node = chatBox.querySelector('input');  
    node.addEventListener("keyup", ({key}) => {  
      if (key === "Enter") {  
        this.onSendButton(chatBox)  
      }  
    })  
  }  
  
  toggleState(chatbox) {  
    this.state = !this.state;  
  
    // show or hides the box
```

```
if(this.state) {
  chatbox.classList.add('chatbox--active2')
} else {
  chatbox.classList.remove('chatbox--active2')
}
}

onSendButton(chatbox) {
  var textField = chatbox.querySelector('input');
  let text1 = textField.value
  if (text1 === "") {
    return;
  }

  let msg1 = { name: "User", message: text1 }
  this.messages.push(msg1);

  fetch('http://127.0.0.1:5000/predict2', {
    method: 'POST',
    body: JSON.stringify({ message: text1 }),
    mode: 'cors',
    headers: {
      'Content-Type': 'application/json'
    },
  })
  .then(r => r.json())
  .then(r => {
    let msg2 = { name: "Sam", message: r.answer };
    this.messages.push(msg2);
    this.updateChatText(chatbox)
    textField.value = "

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



```
        this.updateChatText(chatbox)
        textField.value = ""
    });
}

updateChatText(chatbox) {
    var html = "";
    this.messages.slice().reverse().forEach(function(item, index) {
        if (item.name === "Sam")
        {
            html += '<div class="messages__item2 messages__item2--visitor">' +
item.message + '</div>'
        }
        else
        {
            html += '<div class="messages__item2 messages__item2--operator">' +
item.message + '</div>'
        }
    });

    const chatmessage = chatbox.querySelector('.chatbox__messages2');
    chatmessage.innerHTML = html;
}

const chatbox2 = new Chatbox2();
chatbox2.display();

const chatbox = new Chatbox();
chatbox.display();
```

## APPENDIX B: CSS Code for User Interface

```
* {
  box-sizing: border-box;
  margin: 0;
  padding: 0;
}

body {
  font-family: 'Nunito', sans-serif;
  font-weight: 400;
  font-size: 100%;
  background: #000000;
}

*, html {
  --primaryGradient: linear-gradient(93.12deg, #581B98 0.52%, #9C1DE7 100%);
  --secondaryGradient: linear-gradient(268.91deg, #581B98 -2.14%, #9C1DE7
99.69%);
  --primaryBoxShadow: 0px 10px 15px rgba(0, 0, 0, 0.1);
  --secondaryBoxShadow: 0px -10px 15px rgba(0, 0, 0, 0.1);
  --primary: #581B98;
  --primaryGradient2: linear-gradient(93.12deg, #1b981f 0.52%, #48ff00 100%);
  --secondaryGradient2: linear-gradient(268.91deg, #1b981f -2.14%, #48ff00
99.69%);
  --primaryBoxShadow2: 0px 10px 15px rgba(0, 0, 0, 0.1);
  --secondaryBoxShadow2: 0px -10px 15px rgba(0, 0, 0, 0.1);
  --primary2: #1b981f;
}

/* CHATBOX
===== */
```

```
.chatbox {
  position: absolute;
  bottom: 30px;
  right: 30px;
}

/* CONTENT IS CLOSE */
.chatbox__support {
  display: flex;
  flex-direction: column;
  background: #eee;
  width: 300px;
  height: 350px;
  z-index: -123456;
  opacity: 0;
  transition: all .5s ease-in-out;
}

/* CONTENT IS OPEN */
.chatbox--active {
  transform: translateY(-40px);
  z-index: 123456;
  opacity: 1;
}

/* BUTTON */
.chatbox__button {
  text-align: right;
}

.send__button {
  padding: 6px;
  background: transparent;
```

```
border: none;
outline: none;
cursor: pointer;
}

/* HEADER */
.chatbox__header {
  position: sticky;
  top: 0;
  background: orange;
}

/* MESSAGES */
.chatbox__messages {
  margin-top: auto;
  display: flex;
  overflow-y: scroll;
  flex-direction: column-reverse;
}

.messages__item {
  background: orange;
  max-width: 60.6%;
  width: fit-content;
}

.messages__item--operator {
  margin-left: auto;
}

.messages__item--visitor {
  margin-right: auto;
}
```

```
/* FOOTER */
.chatbox__footer {
  position: sticky;
  bottom: 0;
}

.chatbox__support {
  background: #f9f9f9;
  height: 550px;
  width: 650px;
  box-shadow: 0px 0px 15px rgba(0, 0, 0, 0.1);
  border-top-left-radius: 20px;
  border-top-right-radius: 20px;
}

/* HEADER */
.chatbox__header {
  background: var(--primaryGradient);
  display: flex;
  flex-direction: row;
  align-items: center;
  justify-content: center;
  padding: 15px 20px;
  border-top-left-radius: 20px;
  border-top-right-radius: 20px;
  box-shadow: var(--primaryBoxShadow);
}

.chatbox__image--header {
  margin-right: 10px;
}

.chatbox__heading--header {
  font-size: 1.2rem;
```

```
    color: white;
  }

  .chatbox__description--header {
    font-size: .9rem;
    color: white;
  }

  /* Messages */
  .chatbox__messages {
    padding: 0 20px;
  }

  .messages__item {
    margin-top: 10px;
    background: #E0E0E0;
    padding: 8px 12px;
    max-width: 70%;
  }

  .messages__item--visitor,
  .messages__item--typing {
    border-top-left-radius: 20px;
    border-top-right-radius: 20px;
    border-bottom-right-radius: 20px;
  }

  .messages__item--operator {
    border-top-left-radius: 20px;
    border-top-right-radius: 20px;
    border-bottom-left-radius: 20px;
    background: var(--primary);
    color: white;
  }
}
```

```
/* FOOTER */
.chatbox__footer {
  display: flex;
  flex-direction: row;
  align-items: center;
  justify-content: space-between;
  padding: 20px 20px;
  background: var(--secondaryGradient);
  box-shadow: var(--secondaryBoxShadow);
  border-bottom-right-radius: 10px;
  border-bottom-left-radius: 10px;
  margin-top: 20px;
}

.chatbox__footer input {
  width: 80%;
  border: none;
  padding: 10px 10px;
  border-radius: 30px;
  text-align: left;
}

.chatbox__send--footer {
  color: white;
}

.chatbox__button button,
.chatbox__button button:focus,
.chatbox__button button:visited {
  padding: 10px;
  background: white;
  border: none;
  outline: none;
```

```
border-top-left-radius: 50px;
border-top-right-radius: 50px;
border-bottom-left-radius: 50px;
box-shadow: 0px 10px 15px rgba(0, 0, 0, 0.1);
cursor: pointer;
}

/* CHATBOX2
===== */
.chatbox2 {
    position: absolute;
    bottom: 30px;
    right: 700px;
}

/* CONTENT IS CLOSE */
.chatbox__support2 {
    display: flex;
    flex-direction: column;
    background: #eee;
    width: 300px;
    height: 350px;
    z-index: -123456;
    opacity: 0;
    transition: all .5s ease-in-out;
}

/* CONTENT IS OPEN */
.chatbox--active2 {
    transform: translateY(-40px);
    z-index: 123456;
    opacity: 1;
}
}
```



```
/* BUTTON */
.chatbox__button2 {
    text-align: right;
}

.send__button2 {
    padding: 6px;
    background: transparent;
    border: none;
    outline: none;
    cursor: pointer;
}

/* HEADER */
.chatbox__header2 {
    position: sticky;
    top: 0;
    background: orange;
}

/* MESSAGES */
.chatbox__messages2 {
    margin-top: auto;
    display: flex;
    overflow-y: scroll;
    flex-direction: column-reverse;
}

.messages__item2 {
    background: orange;
    max-width: 60.6%;
    width: fit-content;
}
```

```
.messages__item2--operator {
  margin-left: auto;
}

.messages__item2--visitor {
  margin-right: auto;
}

/* FOOTER */
.chatbox__footer2 {
  position: sticky;
  bottom: 0;
}

.chatbox__support2 {
  background: #f9f9f9;
  height: 550px;
  width: 650px;
  box-shadow: 0px 0px 15px rgba(0, 0, 0, 0.1);
  border-top-left-radius: 20px;
  border-top-right-radius: 20px;
}

/* HEADER */
.chatbox__header2 {
  background: var(--primaryGradient2);
  display: flex;
  flex-direction: row;
  align-items: center;
  justify-content: center;
  padding: 15px 20px;
  border-top-left-radius: 20px;
  border-top-right-radius: 20px;
}
```

```
    box-shadow: var(--primaryBoxShadow2);
  }

.chatbox__image2--header {
  margin-right: 10px;
}

.chatbox__heading2--header {
  font-size: 1.2rem;
  color: white;
}

.chatbox__description2--header {
  font-size: .9rem;
  color: white;
}

/* Messages */
.chatbox__messages2 {
  padding: 0 20px;
}

.messages__item2 {
  margin-top: 10px;
  background: #E0E0E0;
  padding: 8px 12px;
  max-width: 70%;
}

.messages__item2--visitor,
.messages__item2--typing {
  border-top-left-radius: 20px;
  border-top-right-radius: 20px;
  border-bottom-right-radius: 20px;
```

```
}

.messages__item2--operator {
  border-top-left-radius: 20px;
  border-top-right-radius: 20px;
  border-bottom-left-radius: 20px;
  background: var(--primary2);
  color: white;
}

/* FOOTER */
.chatbox__footer2 {
  display: flex;
  flex-direction: row;
  align-items: center;
  justify-content: space-between;
  padding: 20px 20px;
  background: var(--secondaryGradient2);
  box-shadow: var(--secondaryBoxShadow2);
  border-bottom-right-radius: 10px;
  border-bottom-left-radius: 10px;
  margin-top: 20px;
}

.chatbox__footer2 input {
  width: 80%;
  border: none;
  padding: 10px 10px;
  border-radius: 30px;
  text-align: left;
}

.chatbox__send2--footer {
  color: white;
```

```
}  
  
.chatbox__button2 button,  
.chatbox__button2 button:focus,  
.chatbox__button2 button:visited {  
  padding: 10px;  
  background: white;  
  border: none;  
  outline: none;  
  border-top-left-radius: 50px;  
  border-top-right-radius: 50px;  
  border-bottom-left-radius: 50px;  
  box-shadow: 0px 10px 15px rgba(0, 0, 0, 0.1);  
  cursor: pointer;  
}
```

## APPENDIX C: HTML Code for User Interface

```

<!DOCTYPE html>
<html lang="en">
<link rel="stylesheet" href="{{ url_for('static', filename='style.css') }}">

<head>
  <meta charset="UTF-8">
  <title>Chatbot</title>
</head>
<body>
<div class="container">
  <div class="chatbox">
    <div class="chatbox__support">
      <div class="chatbox__header">
        <div class="chatbox__image--header">
          
        </div>
        <div class="chatbox__content--header">
          <h4 class="chatbox__heading--header">Chat support</h4>
          <p class="chatbox__description--header">Hi. My name is Sam. How
can I help you?</p>
        </div>
      </div>
    <div class="chatbox__messages">
      <div></div>
    </div>
    <div class="chatbox__footer">
      <input type="text" placeholder="Write a message...(or type '#' for voice
input)">
      <button class="chatbox__send--footer send__button">Send</button>
    </div>
  </div>
</div>

```

```

    </div>
</div>
<div class="chatbox__button">
    <button></button>
</div>
</div>
</div>
</div>

<div class="container">
<div class="chatbox2">
<div class="chatbox__support2">
<div class="chatbox__header2">
<div class="chatbox__image2--header">

</div>
<div class="chatbox__content2--header">
<h4 class="chatbox__heading2--header">Learning bot</h4>
<p class="chatbox__description2--header">Hi. My name is Ham. Could
you teach me something new?</p>
</div>
</div>
<div class="chatbox__messages2">
<div></div>
</div>
<div class="chatbox__footer2">
<input type="text" placeholder="Write a message...">
<button class="chatbox__send2--footer send__button2">Send</button>
</div>
</div>
<div class="chatbox__button2">
<button></button>

```

```
    </div>
  </div>
</div>

<script>
  $SCRIPT_ROOT = {{ request.script_root|tojson }};
</script>
  <script type="text/javascript" src="{{ url_for('static',
filename='app.js') }}"></script>

</body>
</html>
```



## APPENDIX D: Python Code for Connecting HTML

```
from flask import Flask, render_template, request, jsonify

from chat import get_response
from main import get_response2, new_answer
from tag import review_tag, existing_tag, new_tag
from st import speech_to_text

app = Flask(__name__)

#b is to allow the code to return if it don't know the answer
b = 0
new_question = ""
logged_in = 0
existing = 0
new = 0
new_answer_added = False
newtag = ""

@app.get("/")

def index_get():
    return render_template("base.html")

@app.post("/predict")
def predict():
    text = request.get_json().get("message")

    if text == "#":
        response = speech_to_text()
        message = {"answer": response}
```

```
    return jsonify(message)

response = get_response(text)
message = {"answer": response}
return jsonify(message)

@app.post("/predict2")
def predict2():
    global b, new_question, logged_in, existing, new, new_answer_added, newtag
    user = "human"
    password = "I am human"

    text = request.get_json().get("message")

    if logged_in == 0:
        if text == user:
            logged_in = 1
            log_in_message = {"answer": "Please key in password!"}
            return jsonify(log_in_message)
        else:
            log_in_message = {"answer": "Incorrect Username, please key in again!"}
            return jsonify(log_in_message)

    elif logged_in == 1:
        if text == password:
            logged_in = 2
            log_in_message = {"answer": "Hellow, this is Ham, how may I help you?"}
            return jsonify(log_in_message)
        else:
            log_in_message = {"answer": "Incorrect password, please key in again!"}
            return jsonify(log_in_message)

    elif logged_in == 2:
        if b != 0:
```

```
text2 = request.get_json().get("message")

while existing == 1:
    #add question to a existing tag
    if text2.lower() == 'skip':
        message4 = {"answer": "Sure do, is there anything else I could provide
assitance?"}
        new_question == ""
        b -= 1
        existing = 0
        return jsonify(message4)

    tag, a = existing_tag(text2, new_question)
    final_response = tag.replace("\n", "<br>")
    message3 = {"answer": final_response}

    if a == 1:
        pass

    elif a == 0:
        new_question = ""
        b -= 1
        existing = a

    return jsonify(message3)

while new == 1:
    #add new question to a new tag with a new response
    if text2.lower() == 'skip':
        message4 = {"answer": "Sure do, is there anything else I could provide
assitance?"}
        new_question == ""
        b -= 1
        new = 0
```

```
new_answer_added = False
return jsonify(message4)

else:
    if new_answer_added == True:
        new_intent = new_tag(newtag, new_question, text2)
        message3 = {"answer": new_intent}
        new = 0
        newtag = ""
        new_question = ""
        b -= 1
        return jsonify(message3)

    else:
        newtag = text2
        message3 = {"answer": "I'm ready to learn! Please go ahead and teach
me something new."}
        new_answer_added = True
        return jsonify(message3)

    if text2.lower() == 'skip':
        message4 = {"answer": "Sure do, is there anything else I could provide
assitance?"}
        new_question = ""
        b -= 1
        return jsonify(message4)

    elif text2.lower() == 'new':
        message4 = {"answer": "Sure do, kindly key in a new 'tag' or 'category' for
this question!"}
        new = 1
        return jsonify(message4)

    elif text2.lower() == 'existing':
```

```
tags = review_tag()
# Join tags with newline characters
tags_string = "\n".join(tags)
final_response = tags_string.replace("\n", "<br>")
message3 = {"answer": final_response}
existing = 1
return jsonify(message3)

else:
    message4 = {"answer": "I do not understand, could you re-type it?"}
    new_question == ""
    b -= 1
    return jsonify(message4)

else:
    text2 = request.get_json().get("message")
    response2, a = get_response2(text2)

    if a == 0:
        #bot found a new question and asking for the answer
        final_response = response2.replace("\n", "<br>")
        message2 = {"answer": final_response}
        b += 1
        new_question = text2
        return jsonify(message2)

    else:
        #bot know the question and answer
        message2 = {"answer": response2}
        return jsonify(message2)

if __name__ == "__main__":
    app.run(debug=True)
```

## APPENDIX E: Python Code for Main Chatbot

```
import random
import json

import torch

from model import NeuralNet
from nltk_utils import bag_of_words, tokenize
from translator import translate_to_other, translate_to_eng

device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')

with open('intents.json', 'r') as json_data:
    intents = json.load(json_data)

FILE = "data.pth"
data = torch.load(FILE)

input_size = data["input_size"]
hidden_size = data["hidden_size"]
output_size = data["output_size"]
all_words = data['all_words']
tags = data['tags']
model_state = data["model_state"]

model = NeuralNet(input_size, hidden_size, output_size).to(device)
model.load_state_dict(model_state)
model.eval()

bot_name = "Sam"
```

```
def check_keywords(input_text, keyword_sets):
    stemmed_keywords=[]
    for keyword_group in keyword_sets:
        stemmed_group = [stem(word) for word in keyword_group]
        stemmed_keywords.append(stemmed_group)

    for keyword_set in stemmed_keywords:
        if not any(keyword.lower() in input_text.lower() for keyword in keyword_set):
            return False
    return True

def get_response(msg):
    language, text1 = translate_to_eng(msg)

    if language == 'en':
        sentence = tokenize(text1)
        X = bag_of_words(sentence, all_words)
        X = X.reshape(1, X.shape[0])
        X = torch.from_numpy(X).to(device)

        output = model(X)
        _, predicted = torch.max(output, dim=1)

        tag = tags[predicted.item()]

        probs = torch.softmax(output, dim=1)
        prob = probs[0][predicted.item()]
        if prob.item() > 0.75:
            for intent in intents['intents']:
                if tag == intent["tag"]:
                    for item in intent['responses']:

                        if isinstance(item, dict):
                            if check_keywords(text1, item['keywords']):
```

```
        return item['response']

    else:
        return random.choice(intent['responses'])

return "I do not understand..."

else:
    sentence = tokenize(text1)
    X = bag_of_words(sentence, all_words)
    X = X.reshape(1, X.shape[0])
    X = torch.from_numpy(X).to(device)

    output = model(X)
    _, predicted = torch.max(output, dim=1)

    tag = tags[predicted.item()]

    probs = torch.softmax(output, dim=1)
    prob = probs[0][predicted.item()]
    if prob.item() > 0.75:
        for intent in intents['intents']:
            if tag == intent["tag"]:
                for item in intent['responses']:

                    if isinstance(item, dict):
                        if check_keywords(text1, item['keywords']):
                            sentence = item['response']
                            translate = translate_to_other(sentence, language)
                            return translate

    else:
        sentence = random.choice(intent['responses'])
```



```
translate = translate_to_other(sentence, language)
return translate
```

```
sentence = "I do not understand..."
translate = translate_to_other(sentence, language)
return translate
```

## APPENDIX F: Python Code for Chatbot Learning

```
import json
from difflib import get_close_matches

# Load the knowledge base from a JSON file
def load_knowledge_base(file_path: str):
    with open(file_path, 'r') as file:
        data: dict = json.load(file)
    return data

# Save the updated knowledge base to the JSON file
def save_knowledge_base(file_path: str, data: dict):
    with open(file_path, 'w') as file:
        json.dump(data, file, indent=2)

# Find the closest matching question
def find_best_match(user_question: str, questions: list[str]) -> str | None:
    matches: list = get_close_matches(user_question, questions, n=1, cutoff=0.6)
    return matches[0] if matches else None

def get_answer_for_question(question: str, knowledge_base: dict) -> str | None:
    for q in knowledge_base["questions"]:
        if q["question"] == question:
            return q["answer"]
    return None

# Main function to handle user input and respond
def get_response2(msg):
    knowledge_base: dict = load_knowledge_base('knowledge_base.json')

    while True:
```

```

user_input: str = msg

# Finds the best match, otherwise returns None
best_match: str | None = find_best_match(user_input, [q["question"] for q in
knowledge_base["questions"]])

if best_match:
    # If there is a best match, return the answer from the knowledge base
    a = 1
    answer: str = get_answer_for_question(best_match, knowledge_base)
    return answer, a
else:
    a = 0
    return "I don't have enough information on that right now. Could you provide
more details?\n (Type 'new' to add into a new tag)\n (Type 'existing' to add into a
existing tag)\n (Type 'skip' to skip)", a

def new_answer(msg, qsn):
    if msg != 'skip':
        knowledge_base: dict = load_knowledge_base('knowledge_base.json')
        user_question: str = qsn
        user_answer: str = msg
        knowledge_base["questions"].append({"question": user_question, "answer":
user_answer})
        save_knowledge_base('knowledge_base.json', knowledge_base)
        return "Thank you! I've learned something new."

```

## APPENDIX G: Python Code for Speech-to-Text

```
import speech_recognition as sr

def speech_to_text():
    """Converts speech to text using Google Speech Recognition."""
    recognizer = sr.Recognizer()

    with sr.Microphone() as source:
        print("Speak now...")
        audio = recognizer.listen(source)

    try:
        text = recognizer.recognize_google(audio)
        return text
    except sr.UnknownValueError:
        return "Could not understand audio"
    except sr.RequestError as e:
        return "Could not request results from Google Speech Recognition service; {0}"
```

## APPENDIX H: Python Code for Translator

```
import googletrans
from googletrans import Translator

def translate_to_other(eng_text, language):
    translator = Translator()

    # Translate the text to other language
    foreign_text = translator.translate(eng_text, dest=language).text

    return foreign_text

def translate_to_eng(foreign_language):
    translator = Translator()

    # Detect the language of the text and save it in variables
    detection = translator.detect(foreign_language)
    language = detection.lang

    # Translate the text to English
    eng_translate = translator.translate(foreign_language, dest='en').text

    return language, eng_translate
```

## APPENDIX I: Python Code for Chatbot Training

```
import numpy as np
import random
import json

import torch
import torch.nn as nn
from torch.utils.data import Dataset, DataLoader

from nltk_utils import bag_of_words, tokenize, stem
from model import NeuralNet

with open('intents.json', 'r') as f:
    intents = json.load(f)

all_words = []
tags = []
xy = []
# loop through each sentence in our intents patterns
for intent in intents['intents']:
    tag = intent['tag']
    # add to tag list
    tags.append(tag)
    for pattern in intent['patterns']:
        # tokenize each word in the sentence
        w = tokenize(pattern)
        # add to our words list
        all_words.extend(w)
        # add to xy pair
        xy.append((w, tag))
```

```
# stem and lower each word
ignore_words = ['?', '!', '!']
all_words = [stem(w) for w in all_words if w not in ignore_words]
# remove duplicates and sort
all_words = sorted(set(all_words))
tags = sorted(set(tags))

print(len(xy), "patterns")
print(len(tags), "tags:", tags)
print(len(all_words), "unique stemmed words:", all_words)

#print(all_words, "\n", tags, "\n", xy)

# create training data
X_train = []
y_train = []
for (pattern_sentence, tag) in xy:
    # X: bag of words for each pattern_sentence
    bag = bag_of_words(pattern_sentence, all_words)
    X_train.append(bag)
    # y: PyTorch CrossEntropyLoss needs only class labels, not one-hot
    label = tags.index(tag)
    y_train.append(label)

X_train = np.array(X_train)
y_train = np.array(y_train)

# Hyper-parameters
num_epochs = 1000
batch_size = 8
learning_rate = 0.001
input_size = len(X_train[0])
hidden_size = 10
output_size = len(tags)
```

```
print(input_size, output_size)

class ChatDataset(Dataset):

    def __init__(self):
        self.n_samples = len(X_train)
        self.x_data = X_train
        self.y_data = y_train

    # support indexing such that dataset[i] can be used to get i-th sample
    def __getitem__(self, index):
        return self.x_data[index], self.y_data[index]

    # we can call len(dataset) to return the size
    def __len__(self):
        return self.n_samples

dataset = ChatDataset()
train_loader = DataLoader(dataset=dataset,
                          batch_size=batch_size,
                          shuffle=True,
                          num_workers=0)

device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')

model = NeuralNet(input_size, hidden_size, output_size).to(device)

# Loss and optimizer
criterion = nn.CrossEntropyLoss()
optimizer = torch.optim.Adam(model.parameters(), lr=learning_rate)

# Train the model
for epoch in range(num_epochs):
    for (words, labels) in train_loader:
```



```
words = words.to(device)
labels = labels.to(dtype=torch.long).to(device)

# Forward pass
outputs = model(words)
# if y would be one-hot, we must apply
# labels = torch.max(labels, 1)[1]
loss = criterion(outputs, labels)

# Backward and optimize
optimizer.zero_grad()
loss.backward()
optimizer.step()

if (epoch+1) % 100 == 0:
    print (f'Epoch [{epoch+1}/{num_epochs}], Loss: {loss.item():.4f}')

print(f'final loss: {loss.item():.4f}')

data = {
    "model_state": model.state_dict(),
    "input_size": input_size,
    "hidden_size": hidden_size,
    "output_size": output_size,
    "all_words": all_words,
    "tags": tags
}

FILE = "data.pth"
torch.save(data, FILE)

print(f'training complete. file saved to {FILE}')
```

## APPENDIX J: Python Code for Connecting Both Chatbot JSON File

```
import json
from main import new_answer

def review_tag():
    # Define the path to the JSON file
    file_path = 'intents.json'
    # Extract tags
    with open(file_path, 'r') as file:
        data = json.load(file)
        tags = [intent["tag"] for intent in data["intents"]]
        numbered_tags = [f"<strong>{i+1}</strong> {tag}" for i, tag in
enumerate(tags)]

    return numbered_tags

# Function to read intents from a JSON file
def read_intents_from_json(file_path):
    with open(file_path, 'r') as file:
        data = json.load(file)
    return data

def read_intent_from_json(file_path):
    with open(file_path, 'r') as file:
        data = json.load(file)
    return data["intents"]

# Function to check if a user input tag exists and get the first response
def get_first_response(intents, user_tag):
    for intent in intents:
        if intent["tag"] == user_tag:
```

```
        return intent["responses"][0] # Return the first response
    return None # Return None if tag is not found

# Function to save intents to a JSON file
def save_intents_to_json(file_path, data):
    with open(file_path, 'w') as file:
        json.dump(data, file, indent=3)

# Function to add a new pattern to a specific tag
def add_pattern_to_tag(intents, user_tag, new_pattern):
    for intent in intents["intents"]:
        if intent["tag"] == user_tag:
            intent["patterns"].append(new_pattern)
    return

def get_tag_by_number(number):
    file_path = 'intents.json'

    with open(file_path, 'r') as file:
        data = json.load(file)
        tags = [intent["tag"] for intent in data["intents"]]

    if 1 <= number <= len(tags):
        return tags[number - 1]
    else:
        return "Invalid number. Please enter a number within the range."

def existing_tag(user_input, new_question):
    file_path = 'intents.json'
    intents = read_intents_from_json(file_path)
    intent = read_intent_from_json(file_path)
    user_input = get_tag_by_number(int(user_input))
    response = get_first_response(intent, user_input)
```

```

if response:
    a = 0 #done adding the new question into a existing tag
    new_answer(response, new_question) #add the question and answer into the
second chat
    add_pattern_to_tag(intents, user_input, new_question) #add the question into
the first chat
    save_intents_to_json(file_path, intents)
    return (f"The existing tag '{user_input}' has been added a new question
'{{new_question}}'"), a

else:
    a = 1 #tag is not found
    return (f"The tag '{user_input}' does not exist, kindly type again.\n (retype the
'tag' or 'skip' to skip)"), a

def add_new_intent(intents, new_tag, new_patterns, new_responses):
    new_intent = {
        "tag": new_tag,
        "patterns": new_patterns,
        "responses": new_responses
    }
    intents["intents"].append(new_intent)

def new_tag(newtag, newquestion, newresponse):
    file_path = 'intents.json'
    intents = read_intents_from_json(file_path)
    new_tag = newtag

    # Input new patterns
    new_patterns = []
    pattern = newquestion
    new_patterns.append(pattern)

```

```
# Input new responses
new_responses = []
response = newresponse
new_responses.append(response)

# Add the new intent to the intents list
add_new_intent(intents, new_tag, new_patterns, new_responses)
new_answer(newresponse, newquestion)

# Save the updated intents back to the JSON file
save_intents_to_json(file_path, intents)

return (f"The new question with tag '{new_tag}' has been added.")
```