2022

A Machine learning model for support tickets servicing: a case of Strathmore University ICTS client support services.

Maina, Antony Koimbi
School of Computing and Engineering Sciences
Strathmore University

A Machine Learning Model for Support Tickets Servicing: A Case of Strathmore University ICTS Client Support Services

Antony Koimbi Maina

114731

Master of Science in Information Technology

2022

A Machine Learning Model for Support Tickets Servicing: A Case of Strathmore University ICTS Client Support Services

Antony Koimbi Maina

114731

Submitted in Partial Fulfilment of the Requirements for the Degree of Master of Science in Information Technology (MSc. IT) at Strathmore University

School of Computing and Engineering Sciences

Strathmore University

Nairobi, Kenya

October, 2022

Declaration and Approval

I declare that this work has not been previously submitted and approved for the award of a degree by this or any other University. To the best of my knowledge and belief, the thesis contains no material previously published or written by another person except where due reference is made in the research itself.

Student's Name: Antony Koimbi Maina

Sign: _____ Date: 12/07/2022

Approval

The thesis of Antony Koimbi Maina was reviewed and approved for examination by the following:

Dr. Bernard Shibwabo
Senior Lecturer, School of Computing and Engineering Sciences,
Strathmore University

Dr. Julius Butime,
Dean, School of Computing and Engineering Sciences,
Strathmore University

Dr. Bernard Shibwabo,
Director of Graduate Studies,
Strathmore University

Abstract

Customer service is a highly vital part of any business. How satisfied your customers are can make or break a company. One of the greatest contributors to customer satisfaction is the ability to respond to their issues efficiently and effectively. Many businesses therefore opt to establish a customer service department that handles customers' services, this includes receiving phone calls and replying to emails. Customers are expected to call with issues such as, "How do I reset my password?" "How do I access the Student Information System?" "Are the student's marks out yet?" and the like. Often, the issues reported by customers are similar and tend to get similar resolutions. These requests can be overwhelming at times, for example in cases where the users/customers are accessing an online resource and the system goes down, the number of inquiries can be in the order of thousands depending on the number of the system users. This means a human agent may not be able service all these requests on time. This research aims to develop an intelligent chatbot model for a support ticketing system using machine learning to deliver an exceptional customer experience. This research specifically proposes to develop a machine learning model that can be used to service customer tickets in the context of a university or learning institution. The Rapid Application Development methodology was used to produce a working prototype of a chatbot to test the model to be developed. Machine learning and natural language processing was used to extract a users's intent from a message and by leveraging pretrained frequently asked questions models from DeepPavlov library, the model was trained on 80% of the data and 20% for testing. All the 37 sessions tested on Dialogflow were successful, translating to a 100% success response rate. The prototype was tested by integrating WhatsApp messaging platform to send messages to the chatbot. The chatbot was able to respond to the user in a fraction of a second. The average response time was less than one minute during testing.

Keywords: virtual assistant, artificial intelligence, customer service, machine learning model, natural language processing.

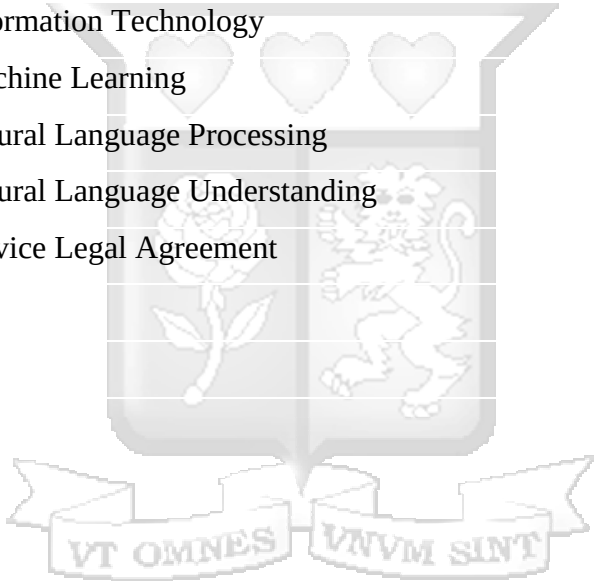
Table of Contents

List of Tables

List of Figures

Acronyms / Abbreviations

AI	-	Artificial Intelligence
API	-	Application Programming Interface
EC	-	Elastic Cloud
ETL	-	Extract Transfer Load
DL	-	Deep Learning
GPU	-	Graphical Processing Unit
ICTS	-	Information Communication Technology Services
IDE	-	Integrated Development Environment
IT	-	Information Technology
ML	-	Machine Learning
NLP	-	Natural Language Processing
NLU	-	Natural Language Understanding
SLA	-	Service Legal Agreement



Definition of Terms

Confidence Score – The confidence with which the model recognizes or understands a user's utterance (Raj, 2019; Khan & Das, 2017).

Entity – Metadata within the intent. Such as train, alarm, time, etc (Raj, 2019; Khan & Das, 2017).

Intent – Is a term used for programmatically identifying the intention of the person who is using the chatbot (Khan & Das, 2017; Biswas, 2018).

Training – Building a model that will learn from the utterances, intents, and entities (Biswas, 2018).

Utterance – Different ways a user can say the same thing (Khan & Das, 2017).

Chapter 1: Introduction

1.1 Background

In large IT organizations and departments, customer complaints are handled through a support ticketing system. A support ticket is an interaction between a customer and a support service attendant (Zendesk, 2020). Users log their issues to the systems mostly by sending an email to a designated email address which in turn log them into the system, a customer portal on the ticketing system, or a messaging/chatting system. These tickets are then organized in streams and categories and then assigned to specific IT administrators who read the customer query, address the issue as required, and then mark the ticket as resolved (Vincent et al., 2005).

Over time, the support ticketing system will have a large dataset of customer queries stored for use in a knowledge base. This data will have attributes such as the requester's names, time, department, the issue, medium of communication, response, and response time. This information can be extremely useful in training a customer service model that this research is exploring. This huge dataset brings up numerous challenges such as a lack of ticket resolution quality analysis. This data could however be leveraged for big data analysis (Gentsch, 2018; Zhou et al., 2017).

Support tickets can at times be overwhelming especially during a service outage of system peak usage times. This leads to a huge backlog for the IT administrator especially when the ticket issues are repetitive. This will force the administrator to lose enthusiasm when working on such tasks and lose the connection between them and the client who might feel like they did not receive personalized attention from the attendant. Mercedes-Benz Consulting's analysis of various contact center data in the automotive sector revealed that 80% of customer inquiries are repetitive and are simple (Gentsch, 2018).

Mundane tasks at our workplaces eventually drain employees' creativity. According to a report by Crabtree (2021), 63% of all employees worldwide are rarely engaged at work. This means that they are rarely using their creative minds to produce new tasks. But these boring, repetitive must be done, otherwise, most companies will go down. Table 1.1 illustrates the employee engagement

report by gallup.com in China showing that a huge percentage of the population is not engaged at work. Meaning that they are emotionally disconnected from their work.

Table 1.1: Employee engagement (Gallup, 2020)

	2009-2010	**2011-2012**
Actively disengaged	27%	24%
Not engaged	62%	63%
Engaged	11%	13%

Ticket servicing can be one of these mundane tasks that are killing off creativity. With the application of Artificial Intelligence, ticket servicing must not necessarily be done by human agents. Gentsch (2018) implies that humans are much freer when engaging with a chatbot. These tasks can be automated using chatbots. A chatbot is an AI software that stimulates conversations between humans and machines (Gentsch, 2018).

With the advancements in artificial intelligence, we can reduce the number of disengaged workforces by letting chatbots take over some of these repetitive tasks. Then the employees could focus on more creative tasks. Ticket turnaround time will also be improved using chatbots to resolve some of these tickets. This will enable departments to meet their Service Legal Agreements (SLAs).

1.2 Problem Statement

Customers want their needs met with immediate effect. This is sometimes not attainable when the person on the other side of the line is a human agent. Occasionally, the issues raised by the customers are not new, they have been seen before. Think of any search term that you can search on Google's search engine, there is a 75% chance that google has seen that before (Schwartz, 2017). Therefore, customer query repetition can be very prevalent. There is a likelihood that a human agent will get bored by repeating the same task over and over. During system usage peak periods, such as exam periods, the customer queries could get a bit too much then it normally is.

This causes the need for extra help for the human agents servicing the queries. It may not be feasible to hire an extra workforce, arising of the need for a virtual assistant. Based on these facts, there arises a need to develop an artificially intelligent model that can address customers' queries.

Using the ticketing system, it is hard to tell the quality of ticket resolution (Zhou et al., 2017). Once a ticket is resolved, the system does not score the satisfaction rate in the resolution of the ticket or accordance with the expectation from the client. Important metrics for customer care such as effectiveness and efficiency directly depend on the quality of the ticket resolution thus the need to keep a scoring system for the quality of ticket resolution.

All the data accumulated in the ticketing system is of no use to a human agent during the ticket resolution process (Zhou et al., 2017). A typical ticketing system relies on a human agent and their knowledge of the domain in question to resolve a ticket, though the resolution needed is sited in the same-self system's database.

1.3 Objectives

1.3.1 General Objective

This research aims to develop a machine learning model for ticket servicing.

1.3.2 Specific Objectives

i) To investigate the challenges faced during customer issue resolution.
ii) To examine the existing methods, techniques, approaches, and systems used for customer issues support.
iii) To develop an intelligent chatbot model for support ticket servicing.
iv) To test the proposed solution.

1.4 Research Questions

i) Which are the challenges faced during customer issue resolution?

ii) What existing methods, techniques, approaches, and systems are used for customer issues support?

iii) How can an intelligent chatbot model for a ticket support system be developed?

iv) How can the model developed be tested?

1.5 Justification

Chatbots and virtual assistants have been widely used in customer engagement across various industries. The most applied field is the Information Technology field. Though customers do not feel the human touch when communicating with a bot, they are happy when their queries are resolved speedily. The lack of the human touch in machines is to blame for the lack of satisfaction amongst many customers. Luo et al., (2019) suggest that chatbot technology has led to a decrease in sales, call length and an increase in call hang-ups. However, his research also finds out that chatbot technology can be four times more effective than an inexperienced worker. This research aims at developing a model that is indistinguishable from a human agent. This is achieved through Natural Language Understanding (NLU) and the confidence level to which the bot can service a user request which when not met, the model will forward the request to the most appropriate human agent.

1.6 Scope and Limitation

This research focuses on serving IT-related customer queries in a university domain around IT systems support issues. It is limited to the data provided by the Strathmore University ticketing system.

Chapter 2: Literature Review

2.1 Introduction

For conversational AI algorithms to understand the meaning in some text, they require intent classification ability. This task is achieved through the application of Natural Language Understanding and Natural Language processing algorithms. Once the machine has been able to classify a user's intent, only then that they can respond appropriately. This technology allows machines (chatbots) to interact effectively with humans. Bots are deemed to be intelligent when human users cannot correctly reorganize that on the other end of the line is a human or computer agent.

2.2 Empirical and Theoretical Literature

2.2.1 Challenges in Customer Ticket Resolution

In customer support, a ticket is any issue raised by a customer that the company has to take care of. This issue can be raised by sending an email, making a phone call, sending a text message, or even sharing the issues through social media. Converting these issues into organized channels is referred to as ticketing. Ticketing helps create a common pool of issues that can be easily handled at the same helpdesk. It also helps in the analysis process (Customer Support Software & Ticketing System | Freshdesk, n.d.).

Numerous challenges are experienced in ticket resolution, including the inability to tell the quality of the ticket resolution process and having a massive dataset of resolved historical tickets that is not useful at all (Zhou et al., 2017). Other familiar challenges include time-consuming and repetitive tasks such as password reset, problems in keeping track of clients' historical tickets and conversations, recurring issues, long resolution times, long resolution times, and numerous requests and interruptions.

2.2.2 AI Evolution in Chatbot Technology

Artificial Intelligence is “The art of creating machines that perform functions that require intelligence when performed by people” (Kurzweil, 1990). The idea was first coined by Alan Turing in 1950 when he published a paper titled "Computing machinery and intelligence" (Turing, 1950). In his paper, Turing tackles the question, “can machines think?”

AI predates back in 1950 when pioneer computer scientist Alan Turing proposed a test to see if a computer would trick a human into thinking that it was a human, but the earliest working chatbot can be traced to a program named ELIZA that was created by Joseph Weizenbaum in 1966. In his paper entitled “ELIZA – a computer program for the study of natural language communication between man and machine” (January 1966), Weizenbaum elucidates how his program used primitive forms of the earliest natural language processing. The output from the program was based on the rules provided by a script. The program could sort words given in the input, sort them, and rank them and words with the highest rank could be suited to form the response to the user. This was a console program as shown in Figure 2.1.

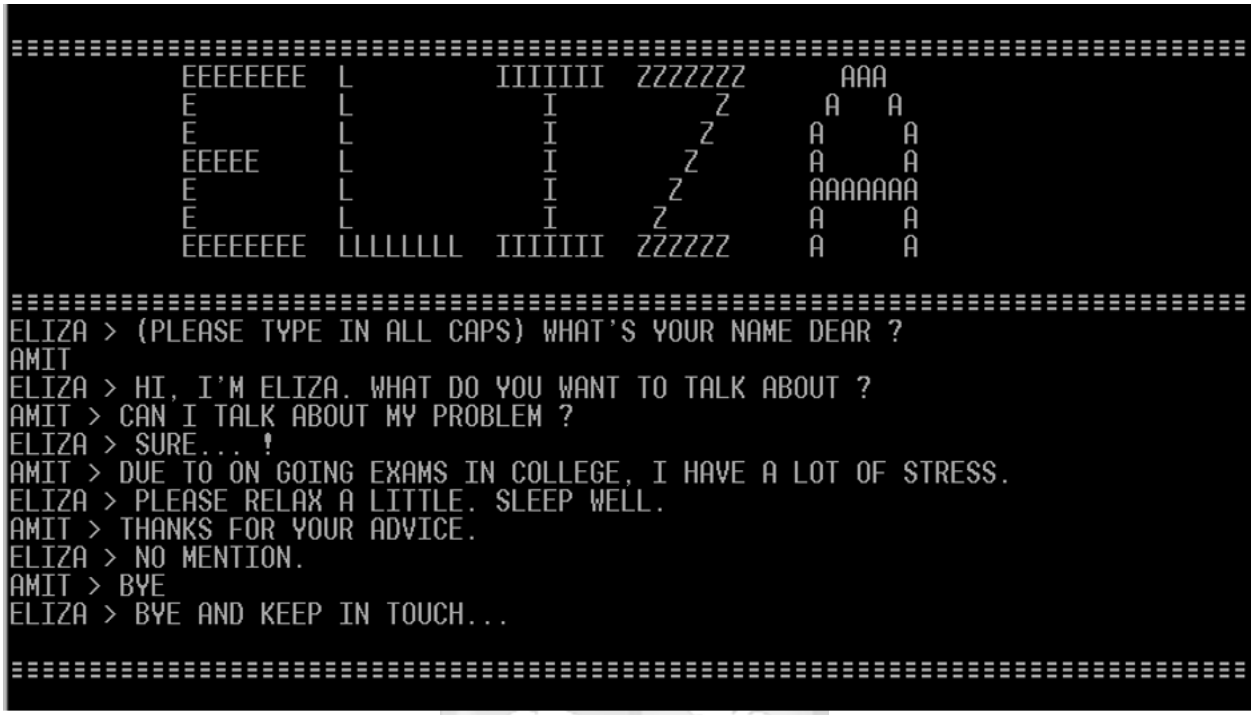


Figure 2.1: The user interface of the ELIZA Program (Electronicsforu.com)

Some of the challenges discussed in this paper are keyword identification, minimal text discovery, appropriate transformations, and general response in absence of keywords. By using the decomposition rule, Weizenbaum addresses most of these challenges. This involves breaking down the sentence into words and ranking the words and using those words to reconstruct a sentence that forms part of the response.

Early researchers saw the potential in Artificial Intelligence and began making bold predictions soon after Weizenbaum's paper. "It is not my aim to surprise or shock you, but the simplest way I can summarize is to say that there are now in the world machines that think, that learn, and that create. Moreover, their ability to do these things is going to increase rapidly until – in a visible future – the range of problems they can handle will be coextensive with the range to which the mind has been applied" (Herbert, 1957). "In three to eight years, we will get a machine with average human intelligence" (Minsky, 1970).

The development of AI can be classified into three phases: 1950s – 1970s, 1970s to 1990s, and 1990s to date as shown in the Figure 2.2. Over these periods its popularity has grown tremendously. The first phase is the birth and early forms of AI, the second involves the development of expert system algorithms in the 1970s and the latter is the modern AI developments in Machine learning, deep learning, and the new subset fields of Artificial Intelligence (Delipetrev et al., 2020). The popularity of machine learning has been on the rise steadily since the 1990s as shown in Figure 2.2.

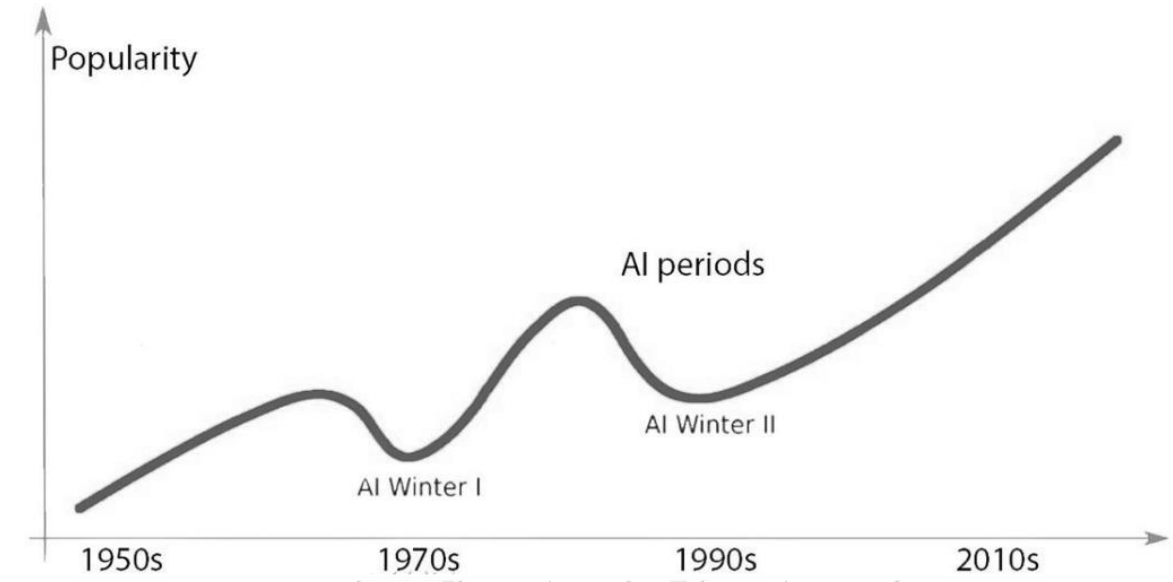


Figure 2.2: Illustrative visualizations of AI periods (Delipetrev et al., 2020)

Machine learning (ML), a subfield of AI, is the scientific study of algorithms that make computer systems learn through experience. Machine learning algorithms build a model based on sample data, known as "training data", to make predictions or decisions without being explicitly programmed (Delipetrev et al., 2020). Machine learning is classified into three categories:

a) Supervised learning where the data model is developed from training examples. The training examples induce the model with labels which help in the classification of the unlabelled data (Cunningham et al., nd).

b) Unsupervised learning is where data is categorized into classes without having labels or training data. Common techniques applied include clustering among others (Green et al., nd).
c) Semi-supervised algorithms can be considered a category between supervised and unsupervised learning, where the data contains both labeled and unlabelled data (Delipetrev et al., 2020).
d) Reinforcement learning explores how agents take actions in an environment to maximize a reward (Delipetrev et al., 2020).

Unlike traditional programming where rules and data are supplied to give answers, in machine learning, data and answers are fed to a learning model to produce rules (Craglia et al., 2018) as shown in Figure 2.3. The rules are then used to infer answers when a set of unseen data is supplied to the model. Traditional programming languages such as Java, C, and C# are rule-based. Figure 2.3 is an illustration of the difference between machine learning programming and traditional programming.

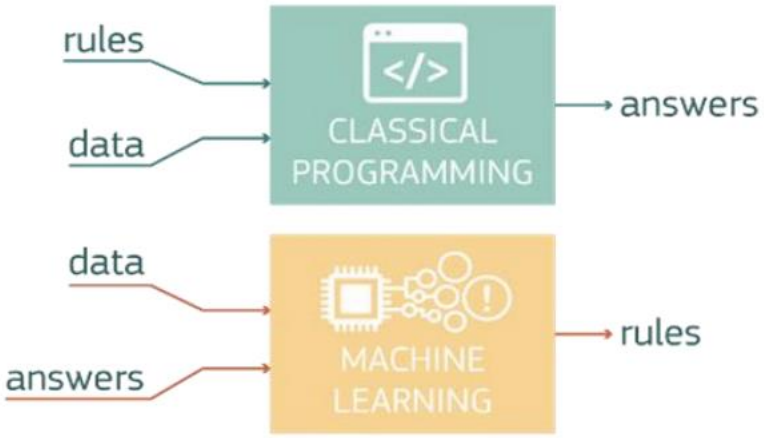


Figure 2.3: Machine Learning Paradigm (Craglia et al., 2018)

Another subfield of Machine Learning and Artificial Intelligence is Deep Learning as shown in Figure 2.4. Deep learning employs multiple layers of neural network abstractions. Neural networks

try to mimic the behavior of the brain's biological neurons. This allows Deep learning algorithms to learn from exceptionally large datasets. In recent advancements, deep learning algorithms are powering every day's products and services such as chatbots which we are exploring in this research.

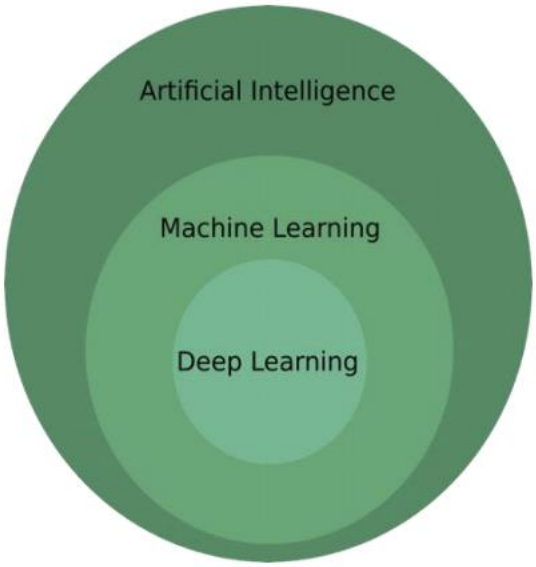


Figure 2.4: AI, ML, and DL. (Craglia et al., 2018)

The performance of deep learning algorithms improves with an increase in the amount of data as shown in Figure 2.5.

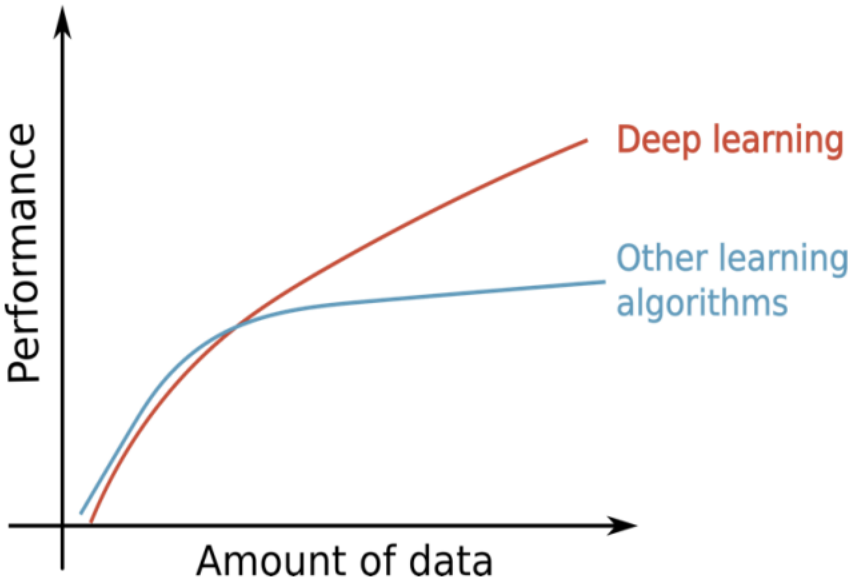


Figure 2.5: Deep Learning performance vs the amount of data (Craglia et al., 2018)

In the words of Ng, A (2017), "Just as electricity transformed almost everything 100 years ago, today I have a challenging time thinking of an industry that I don't think AI (Artificial Intelligence) *will transform in the next several years."*

2.2.3 Natural Language Processing

According to IMB (2020), Natural language processing (NLP) is a branch of Artificial Intelligence that is concerned with making machines similarly understand natural text or voice as a human can. Some tasks that are included in NLP are speech recognition, word sense disambiguation, sentiment analysis, natural language generation, among others. It is universally applicable in intelligent chatbot development in all its subset of tasks.

Natural Language Processing is the intersection of Artificial Intelligence and linguistics (Nadkami et al., 2011). Large volumes of text are indexed using highly scalable statistic-based techniques. NLP investigates the ability of computers to understand natural language to perform an important

task through incorporating various disciplines such as cognitive science, computational linguistics, artificial intelligence, and computing science (Deng & Liu, 2018).

NLP has widely been applied in speech recognition, spoken language understanding, dialogue systems, lexical analysis, parsing, machine translation, knowledge graph, information retrieval, question answering, sentiment analysis, social computing, natural language generation, and natural language summarization (Deng & Liu, 2018).

Natural Language Understanding (NLU) is a subset of NLP which is used in breaking down the various elemental parts of speech such that a machine will be able to understand it. Here the intent in the text is determined. In this research, NLU will enable our model to understand our customers' queries.

2.2.4 Conversational AI

Conversational Artificial Intelligence is the application of NLU to create technologies such as chatbots making the communication between man and machine appear natural. Unlike traditional scripted chatbots that use rules and predetermined responses, Conversational AI chatbots are fed with questions and answers can infer answers to new conversations.

2.2.5 Chatbot Technology

As described in section 2.2, the rise of AI brought about the rise of chatbot technology with ELIZA being the earliest conception of this field in the 1960s. Since then, chatbot technology has made great milestones and growing exponentially in the recent past (Raj, 2019; Biswas, 2018) as shown in Figure 2.6.

Figure 2.6: Chatbot Trends over the recent past based on the web search term (Raj, 2019)

2.2.5.1 A brief chatbot Timeline

In 1950, Alan Turing proposed an experiment, which is known today as the Turing Test, that studied the ability of a machine to exhibit human behaviour. This was quickly followed by the earliest chatbot model – ELIZA – in 1966. ELIZA "behaved" as a doctor does through asking questions and using pattern matching and would give an answer. In 1972, Kenneth Colby developed a computer program – Parry – that modelled the behaviour of paranoid schizophrenics. In 1985, a taped message could be imitated by Tommy Chatbot, which was a wireless robotic toy. Launched in 1987, though started in 1981, Rollo Carpenter developed a chatbot that could "simulate natural human chat in an interesting, entertaining and humorous manner" named Jabberwocky. This goes on to Dr Sbaitso in 1992, who could "converse" with someone in a digitized voice from Creative Labs and to ALICE in 1995 by Richard Wallace (Raj, 2019; Biswas, 2018; Khan & Das, 2017).

Things began to spice up in 2001 when ActiveBuddy developed a bot named SmartChild which could give instant access to weather feeds, movie times, stock information, news, etc. In 2006, IBM Watson developed a chatbot that went on to compete on the popular TV Show "Jeopardy" and would put human contestants to shame regularly (Raj, 2019; Biswas, 2018; Khan & Das, 2017).

Later in 2010, Apple released Siri on their iOS iPhone devices for speech recognition which was an impeccable milestone in chatbot technology. Google would go on to release Google Now in 2012 while Amazon would release Alexa two years later in 2014. Microsoft Cortana was released in 2015, Facebook AI messenger in 2016. In 2017, Woebot from Woebot Labs was released which is being used in the treatment of mental health (Raj, 2019; Khan & Das, 2017. Figure 2.7 shows a typical implementation of a modern chatbot.

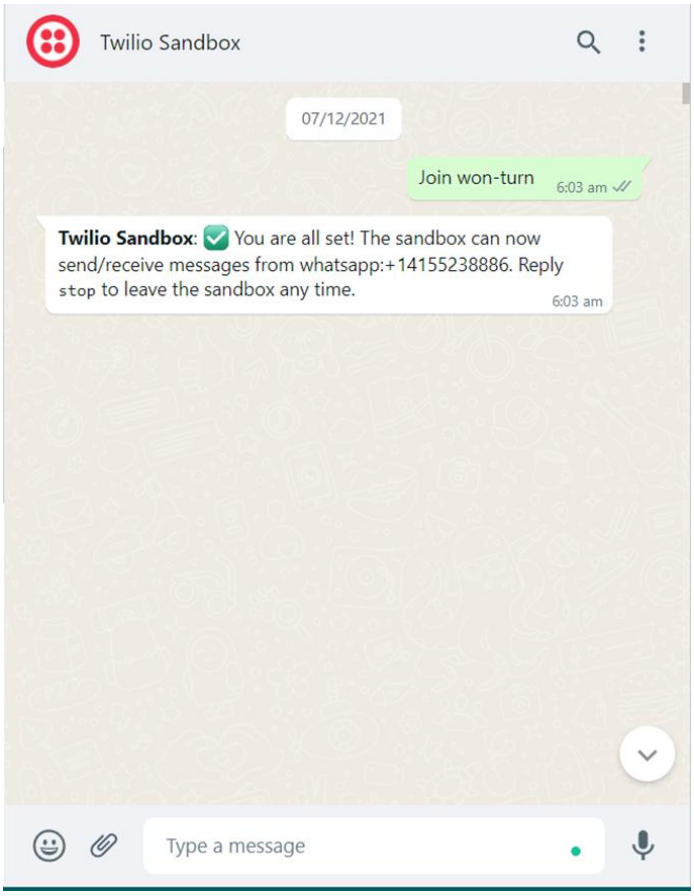


Figure 2.7: A Chatbot Example

2.2.5.2 The Need for Chatbots

There has been a surge in the increase in usage of chatbot technology in recent years. This has been enabled by the fast advancement in the field of machine earning as discussed in earlier

sections. According to Raj (2019), this need has been driven by the fact that chatbots are easily accessible, highly efficient, highly available, easy to scale, they are cheap, and easy to get and analyze client insights.

2.3 Chatbot Design Models and Frameworks

Various frameworks have been developed to aid in the development of complex AI chatbot models. These frameworks help in creating conversational interfaces for both voice and text recognition.

2.3.1 DeepPavlov

DeepPavlov is an open-source framework for chatbots and virtual assistants' development. It has comprehensive and flexible tools that let developers and NLP researchers create production-ready conversational skills and complex multi-skill conversational assistants (DeepPavlov: An Open Source Conversational AI Framework, n.d.).

2.3.2 Dialogflow CX

Dialogflow CX offers chatbot developments tools as a service. You can define your intents, entities, utterances and create a model extremely fast. You can also connect to popular messaging channels and devices such as Google Assistant, WhatsApp, Alexa, among others (*Dialogflow Documentation*, n.d.).

2.3.3 Wit.ai

Like Dialogflow CX, Wit.ai allows developers to create apps that you can converse with. It does not have as many features as Dialogflow and is, therefore, free to use. It uses an NLP engine developed at Facebook (*Wit.AI*, n.d.).

2.3.4 Other notable mentions

Other similar frameworks include Woebot which can be able to analyze the mood of its users, QNAMaker which is used to create simple question and answer models, Rasa.ai, Botkit.ai, Luis.ai among others. All these bots have an NLP underlying engine that is used to understand the user and reply to their queries.

2.3.5 Popular Chatbots applications in Support Ticketing Systems

With the impeccable rise in chatbot technology, various application fields have adopted the technology including our area of study – support ticketing. This is because of the numerous advantages that come with the use of bot technology especially its effectiveness and efficiency. Zoho Corp has implanted an Automatic Ticket Resolution chatbot – *Zia* – into their support systems. The system has set up templates that when the bot assesses a ticket issue, it resolves and marks it resolved. The customer then can reopen the ticket if they are not satisfied with the bot resolution (Zia – Zoho's AI assistant for business. (n.d.)). Freshdesk's *Freddy* is an intelligent bot that is well used stored templated to services customers with the best solution to their tickets (Gupta, n.d).

It is hard for human agents to keep track of conversations in large-scale firms. Chatbots can be used to offer individualized attention to millions of users enhancing the interaction between firms and customers and the social-economic benefits that come with it (Xu, et al., 2017).

2.4 The Conceptual Framework

This section shows a diagrammatic representation of how the research objective is achieved based on the reviewed literature as shown in Figure 2.8.

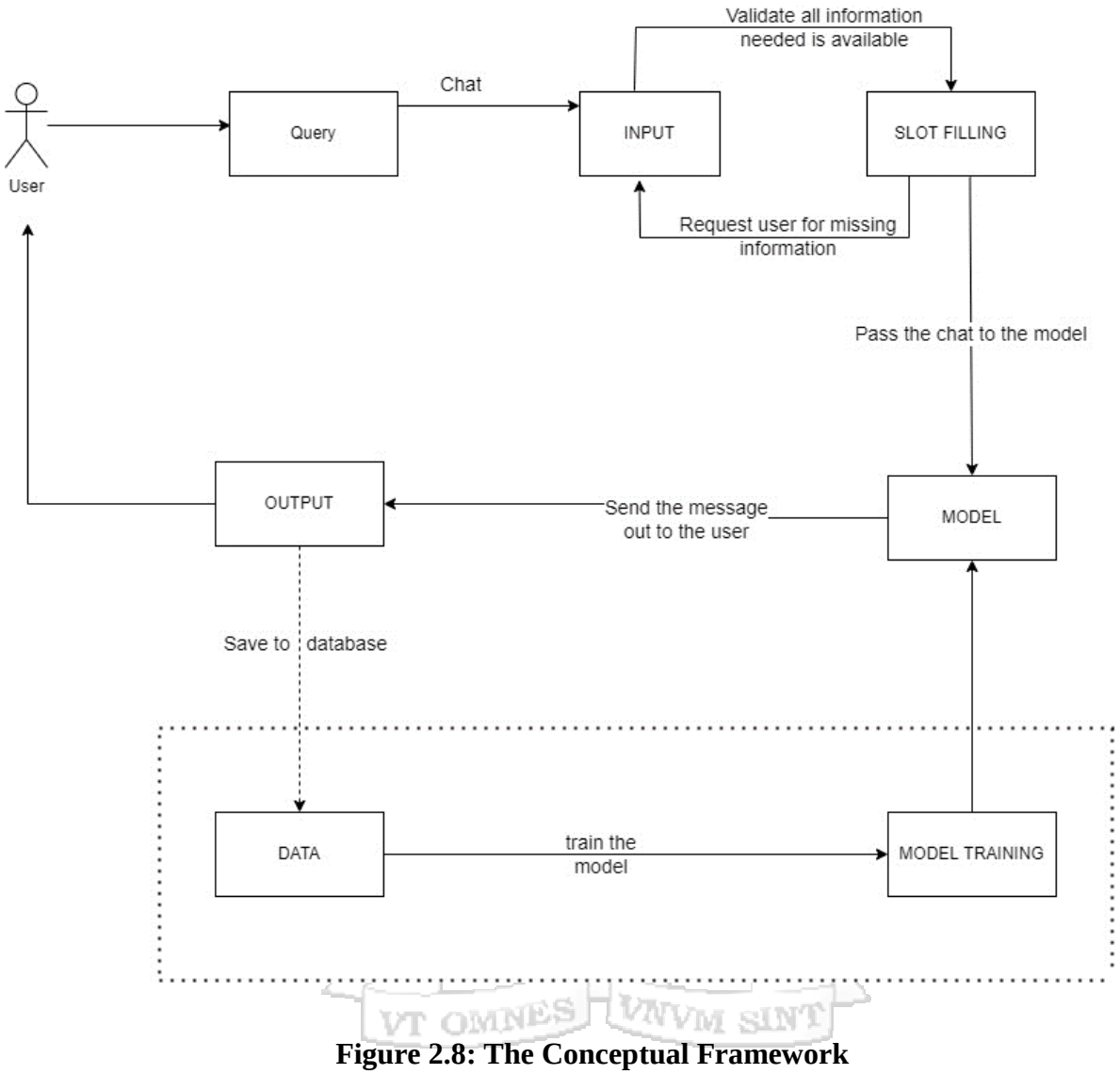


Figure 2.8: The Conceptual Framework

The research proposes the following conceptual framework as an improved query handling algorithm apropos of support functions that can be applied to innumerable numbers of systems.

A user initiates the process by querying the system via a chat function. The chat constists of a message and otyher useful metadata from the messaging application such as username and datetime. The chat is recorded as input and goes through an iterative process of slot filling by capturing all the information the algorithm requires from the user. The slot filler completes the chat message with necessary variables for the chat such as gender or whether they are staff or

student among others. The message is validated to contain all the necessary parameters by the slot filler, this is achieved using Dialogflow, a conversational library developed and maintained by Google. The main purpose is to abstract machine language and presents a clean human-like response. Historical data was used to train the model to which a structured query from the user was passed. The model presented the most suitable response and the user can select which answer best suits their request, this is also an iterative process. A threshold was set for which the most suitable answer must meet. All queries from the user are stored and are used to retrain the model and improve accuracy.

Chapter 3: Research Methodology

3.1 Introduction

Research methodology is the path through which researchers need to conduct their research (Jilcha, 2020). This chapter examines the approach to be followed by this research, the tools to be used, data, design process, procedures, and application frameworks to be used. It also describes ethical practices to be taken into consideration.

The main goal of this research was to design and develop a framework for ticket resolution. The Client Support department at Strathmore University uses the framework to give feedback to customer queries. The framework outlines the necessary steps to follow for the successful implementation of the client ticket resolution model in the client support department.

3.2 Research Design

According to Jilcha (2018), a research design is intended to provide an appropriate framework for a study. The proposed research is applied research as it aimed to solve a real-world problem facing the client support department: A case study of Strathmore University. Its utility has been limited to the resolution of customer query tickets normally submitted via emails. The research proposed to build a software robot based on Java programming language and hosted on the cloud, on a Linux server to take queries from a client through a messaging platform respond to the client through the same medium.

3.3 Model Development

The model development followed the steps listed below:

i. Data Collection
ii. Data Pre-processing
iii. Development of the Model

iv. Model Testing and Validation

3.3.1 Data Collection

The data that was used as input for the model was obtained from Strathmore University's Ticketing System database. The database was a structured MySQL database and therefore extracting was by simply writing well-defined SQL queries.

3.3.2 Data Pre-Processing

The data obtained were pre-processed to remove any duplicates and incomplete data that existed. An SQL query was used to extract the unique from the database and join the tables that had foreign keys referring to ticket titles, administrator names, and transaction keys. Data with HTML content type was also removed and text content type preserved. Table 3.1 shows the resultant columns that the researcher extracted.

Table 3.1: Columns of Etorted Data from the RT System

Column	Explanation
Ticket_ID	The ticket ID of an email
Transaction_ID	Transaction ID shows an instance of a ticket, for example, closed, opened, replied, resolved, etc.
Subject	The subject message of a ticket
Content	The content or body of the email ticket
Name	The name of the assigned administrator
ContentType	The type of content whether text or HTML

The extracted data were further processed to get conversations from the data. All tickets with the same ticket ID were grouped as the same conversation and later into questions and corresponding answers as shown in Table 3.2. The output was saved into a CSV text format.

Table 3.2: Resultant CSV Columns

Column	Explanation
Question	The client utterance from the ticket content.
Answer	The admin utterance from the ticket content.

3.3.3 Development of the Model

The CSV text file from data processing was fed into a DeepPavlov pre-trained Frequently Asked Questions model. The model was retrained using the processed custom ticketing data for questions and answers. After retraining, the model was linked with Google's Dialogflow where a few predetermined conversational intents were defined. A web service was created using JAVA programming language to link the hybrid model with the Twilio messaging platform to terminate the user response through WhatsApp messenger and receive tickets through the same medium.

3.3.4 Model Testing and Validation

A chat message was sent through the WhatsApp messenger to the testing number and an instant message was received. The model was able to recognize a user's intent and the researcher was able to have small talk with the chatbot model when they asked a question whose intention was not predetermined, the model was able to match the question with the most appropriate answer.

3.4 System Development Methodology

A system development methodology is the steps that are used in the process of developing a system, they form the plan for the development of an information system. Machine learning model training is an iterative process that requires development and test cycles. The Rapid Application Development (RAD) methodology is an iterative agile process of software development that was applied in the development of the AI model. Agile methodology promotes continuous iteration of development and testing throughout the software development lifecycle, both testing and development are concurrent which speeds up the lifecycle of a project (Tobin & Akhilesh, 2011).

RAD emphasizes rapid prototype releases and test iterations (Singh, 2019). Figure 3.1 shows the various processes in rapid application process.

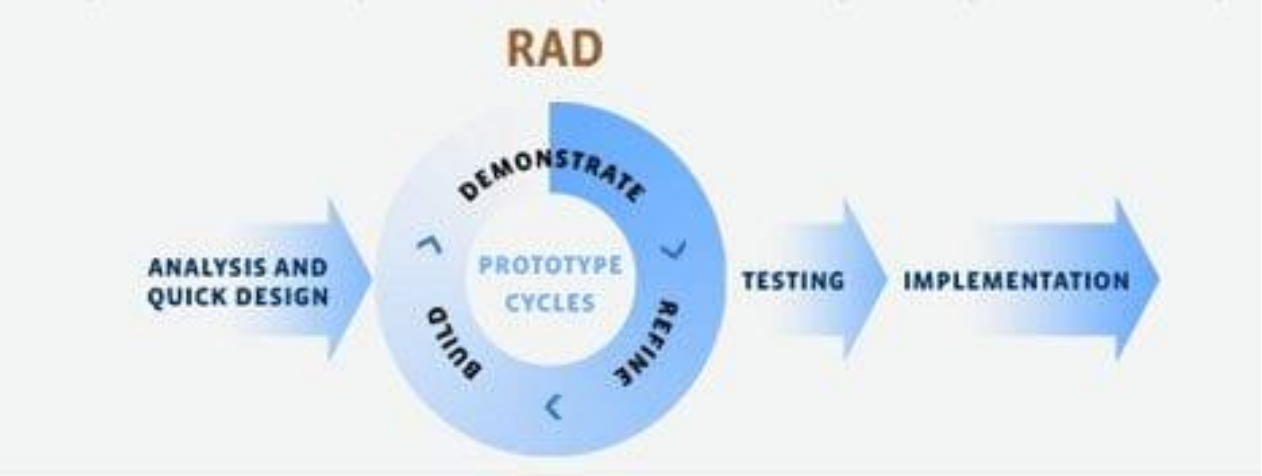


Figure 3.1: Rapid Application Development (RAD) process (Singh, 2019).

3.4.1 Analysis and Quick Design

In this phase, a review of the existing data and preparation for use was done. There was a definition of the project requirements, expectations, goals, and seeking approvals. There will also be a review of the application frameworks that are expected to be used and test them and the challenges that are likely to be faced.

3.4.2 Prototype Cycles

This phase has three phases: quick development of the model, presenting and gathering feedback, then refining the model from the feedback and testing results. In this research, we will be optimizing and finetuning the model by varying test data and test cycles. The model's training data is capped at 80% percent and the rest of the data is used to test the model. The researcher is

expected to repeatedly vary the number of training cycles until the most appropriate number of cycle times is obtained. Lastly, we integrate the model into a chatbot.

3.4.3 Testing

After we have a well-trained model integrated with a chatbot, it will be integrated with the RT ticketing system on a test environment and then test by sending a ticket and assessing how the chatbot will respond to the ticket.

3.4.4 Implementation

After a satisfactory testing phase, the launch of the integrated chatbot with the RT system. Users shall be informed of the new change and that their queries shall be resolved in new record times and the researcher shall continue monitoring the performance of the model.

3.5 Target Population

According to Kothari (2004), the target population is the particular population from which the researcher has an interest and intends to extract the research sample. The target population for this research was 30,000 emails collected from the Request Tracker ticketing system used by Strathmore University for logging students' and staff' tickets. These emails were collected are from the year 2017 to year 2020. They also include email replies from the system administrators who service the requests.

3.6 Location of the Study

The study will take place in the ICTS labs of Strathmore University where they have a support ticket management system that has been used long enough to gather enough data to train the model under study. Previous studies that have been carried out in this location make it more suitable to carry on the research on their premises.

3.7 System Implementation Tools

3.7.1 Database Tools

In this research, we used an SQL script to get the data from the RT ticketing system database. Several tables were joined using JOIN statements to come up with one simple query to extract data and save it as a CSV file.

3.7.2 AI model Training Framework

In the training phase of the model, a combination of tools was used to come up with the blended system. To handle incoming messages/queries from the user and to respond to the user, the researcher used WhatsApp messaging platform and linked it to the web service using the Twilio messaging platform. Google's Dialogflow was used to handle chitchat with the model and also to get missing information from the client. The researcher also employed DeepPavlov, an NLP opensource library to train the AI model.

3.7.3 Development Stack

The researcher employed PyCharm when cleaning the data from Request Tracker. IntelliJ Idea was used to create the web service which was hosted on the Digital Ocean container platform after it was containerized using Docker. Twilio Messaging Sandbox was used to test communication to and from the web service on the WhatsApp Messaging Platform.

3.8 Research Quality

A comparison between the findings and the reality was performed to verify the conclusions obtained from this study. Unusual responses were removed which extend to be outliers. This form of validation is called triangulation of the data. Data triangulation enables the collection of data from different sources to notify or check one accurate body of data. In addition, the prediction model's accuracy was measured to be sure of the reliability level of the proposed solution. A threshold of 0.8 was set as the optimal value for which all the responses had to meet.

3.9 Ethical Considerations

In as much as the research intends to make a model that is indistinguishable from a human model, the customers will be notified when they are interacting with a machine and when interacting with a human. Privacy for the information shared with the chatbot was paramount and since the research used data provided by the Strathmore University ICTS ticketing system, it was be safeguarded per their data protection policies. The researcher ensured that any information retrieved from various sources was mainly used for this research, personal details such as names, email addresses, or mobile numbers were not used in the research. To address data risk that may be associated with the Request Tracker, the researcher ensured that all data was archived data at least one year old, ensuring that data such as passwords would have expired already if present and no current conversations or unresolved issues were present during the analysis.

3.10 Dissemination of the Study Results

The results of this study shall be used for academic purposes only. They shall be compiled into a Master's Thesis as an academic requirement. The dissemination shall be via an oral defense, then the results shall be available for the general public via the University library portal, finally that the work is expected to be presented at conferences.

Chapter 4: System Analysis, Design, and Architecture

4.1 Introduction

This chapter presents the overall architecture and detailed design and analysis of the proposed system by incorporating the various requirements. In the analysis phase, the questions of who will use the system, what it will do, and where and when it will be used are answered. During this phase, the researcher investigates any current system, identifies opportunities for improvement, and develops a concept for the new system. Finally, the design phase decides how the system operates in terms of the hardware, software, and network infrastructure; the user interface, forms, and reports; and the specific programs, databases, and files needed (Dennis et al., 2015).

Further, this chapter analyses both functional and non-functional requirements, the design of the proposed system incorporates UML diagrams to describe the overall architecture of the system and give a detailed description of the various components of the system. Use case diagrams with detailed use case descriptions, sequence diagrams, context diagrams, and data flow diagrams.

4.2 Requirements for the Proposed System

From the data collected from the Request Tracker, as shown in section 3.3.1, the following requirements were deduced for the proposed system.

i) A system that would give feedback to customers 24 hours a day, 7 days a week.
ii) A system that would be able to collect tickets from the RT system and send feedback to the client.
iii) A secure system that would be able to maintain confidentiality and integrity.
iv) A system that would enable data-driven decision-making which is accurate.
v) A system that would generate reports weekly or on-demand.
vi) A system that generates logs on what it has done.
vii) A robust system that would not keep on failing now and then.

viii) A user-friendly system that the customer support department would interact with to prioritize certain customers and issues.

ix) Access to Dialogflow CX, Twilio Messaging, and a WhatsApp Business Number.

The researcher further classified those requirements into functional and non-functional requirements. A functional requirement relates directly to a process a system must perform or the information it needs to contain. Non-functional requirements refer to behavioural properties that the system must have, such as performance and usability (Dennis et al., 2015).

4.2.1 Functional Requirements

i) A system that would generate a weekly report.

ii) A system that would keep logs of its operations.

iii) A system that would be able to query the question-answer data set and give feedback to a client.

iv) A system that would be able to send messages or email messages.

4.2.2 Non-functional Requirements

i) The system should be reliable and efficient – The system not only needed to give feedback to a client but also do it within the shortest time possible and provide accurate results.

ii) The system should be robust and resilient – The system had to be able to recover from failures and remain functional to the customer. The bot also had to be resistant to failure and complete the task.

iii) The system should be scalable – Any increase in the number of client conversations had to be accommodated by the system. The model also needs to be easy to train on data from other categories.

iv) The system should be secure – The system had to ensure that the model cannot be compromised or retrained on some other dataset.

v) Usability – The system was intended for use by the customer care team, and they had to be trained so that they could understand their interaction with the bots, but the system was intended to eliminate remedial repetitive tasks, hence did not require highly skilled staff to operate and support it.

4.2.3 System Requirements

The proposed system had the following system requirements for its operation.

a) System Security

The system ensured that the model was secured in a containerized environment and that the third-party application had strong passwords and two-factor authentication. The messaging was done and tested on WhatsApp which boasts end-to-end encryption.

b) Software Requirements

The system was developed on a 64-bit Ubuntu Linux Operating System. Java JDK 11 was installed alongside Python 3 and Maven. Testing Accounts were created on Twilio and Dialogflow. To containerize the system, Docker compose was required.

c) Hardware Requirements

A fast and efficient network system was required to boost communication. To deploy on production,

d) Relational database

A relational database was used to store the system messaging logs and user sessions.

4.3 System Architecture

The architecture of the proposed system as shown in Figure 4.1 outlines the general flow of data on the system to ensure automatic ticket resolution. The major steps that took place in the bot system were as follows:

i) The user sends a message through a messaging platform. The first message is used to initialize a session then send back a welcome message. The subsequent messages are forwarded to the intent detector.

ii) The message is analyzed at the intent detector and if the intent is matched, a feedback message is sent to the user. The intent detector will also address all small talk from the user. If the intent is not matched, the message is again forwarded to the question-answer trained model. The intent detector also plays the slot filler. If any of the defined required parameters are missing, a feedback message is sent to the user requesting them to provide the information. The logger logs all activity in this phase.

iii) The question-answer trained model will match a user message – referred to herein as the question – to the best answer. This answer is forwarded to the user through the same messaging platform. The logger as well logs activity in this phase.

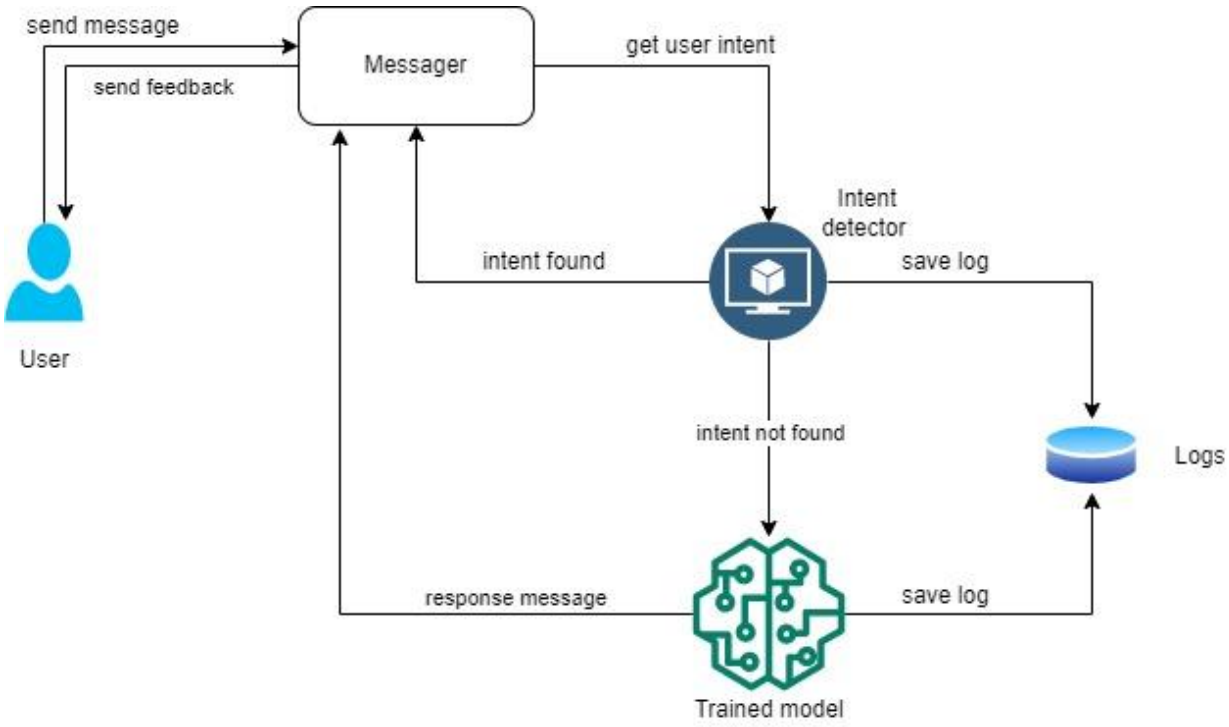


Figure 4.1: System Architecture

4.4 System Process Modelling

A software process is a set of activities undertaken to manage, develop and maintain software systems (Acuna & Juristo, 2006). A software process model is an abstract representation of the software process's architecture, design, or definition (Acuna & Juristo, 2006). The process models can be analyzed, validated, and simulated. The process models improve process understanding, communication, and software process control.

4.4.1 Use Case Diagram

A use case diagram defines a sequence of interactions between an actor and the system (Gomaa, 2011). It is illustrated in UML notation, as shown in Figure 4.2. It also depicts the functionality that the proposed system has.

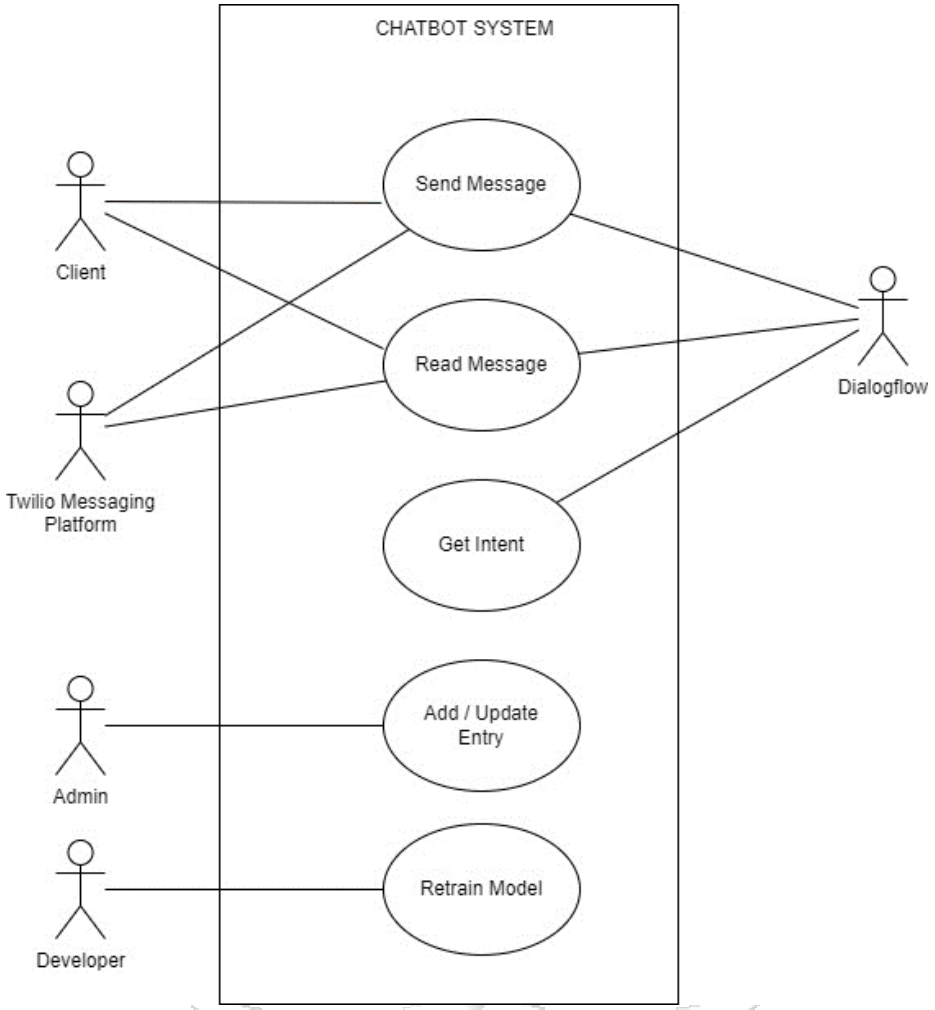


Figure 4.2: Use Case Diagram

Table 4.1: Send Message Use Case Description

Use Case: Send Message
Primary Actors: Client, Twilio, Dialogflow
Brief Description: This use case describes how a client will send a message to the model.
Pre-Conditions: The message handler is up and listening.
Post-Conditions: The message was successfully sent.
Main Success Scenarios: The messages were successfully sent to the model.

Actor Responsibility	System Responsibility
The client sends a message to the model.	
	The model receives a message from the client.

Table 4.2: Receive Message Use Case Description

Use Case: Receive Message	
Primary Actors: Client, Twilio, Dialogflow	
Brief Description: This use case describes how a client will receive a message/response from the model.	
Pre-Conditions: The message handler is up and listening.	
Post-Conditions: The message was successfully received.	
Main Success Scenarios: The client successfully received a message from the model.	
Actor Responsibility	System Responsibility
The client receives a message from the model.	
	The model dispatches a message to the client.

Table 4.3: Add / Update Database Entry Use Case Description

Use Case: Add / Update Database Entry
Primary Actors: Admin
Brief Description: This use case describes how a system admin will update the database with new training data.
Pre-Conditions: New clean data are available.

Post-Conditions: Data was successfully saved in the database.	
Main Success Scenarios: Data was successfully saved in the database.	
Actor Responsibility	System Responsibility
The Admin cleans the data	
The Admin pushes the data to the database	
	The data is successfully saved in the database.

Table 4.4: Model Retrain Use Case Description

Use Case: Model Retrain	
Primary Actors: System Developer	
Brief Description: This use case describes how a system developer retains the model given new data.	
Pre-Conditions: The set-up code, old and new clean data are available.	
Post-Conditions: The model was successfully retrained.	
Main Success Scenarios: The model was successfully retrained.	
Actor Responsibility	System Responsibility
The System Developer retrains the model.	

4.4.2 Sequence Diagram

Figure 4.3 illustrates the relationship between the user and the proposed chatbot and the interactions among other components of the systems and the chatbot model bot. The user sends a query message through a chat messenger to the chatbot. The message is acted upon by the model proposed. The output message is stored on the database repository and sent back to the user through the same medium that it came in.

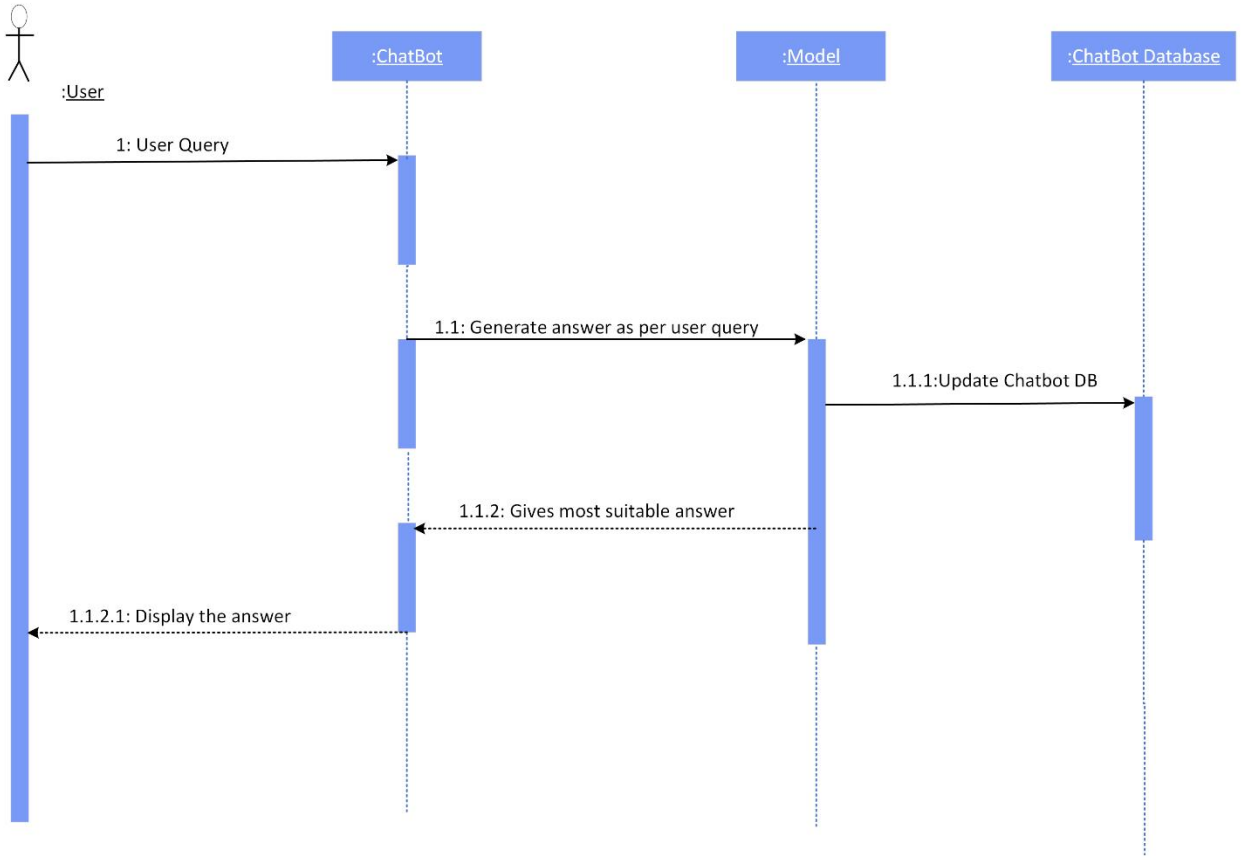


Figure 4.3: Sequence Diagram

4.4.3 Context Diagram

Figure 4.4 shows the context diagram for the bot. It shows the interaction entities, the environment, and the boundaries. All the inputs and outputs are illustrated using the arrows. The primary interactive entity is the user who queries the bot and receives an instant response. The administrator is responsible for updating the database entries while the system developer will be retraining the model.

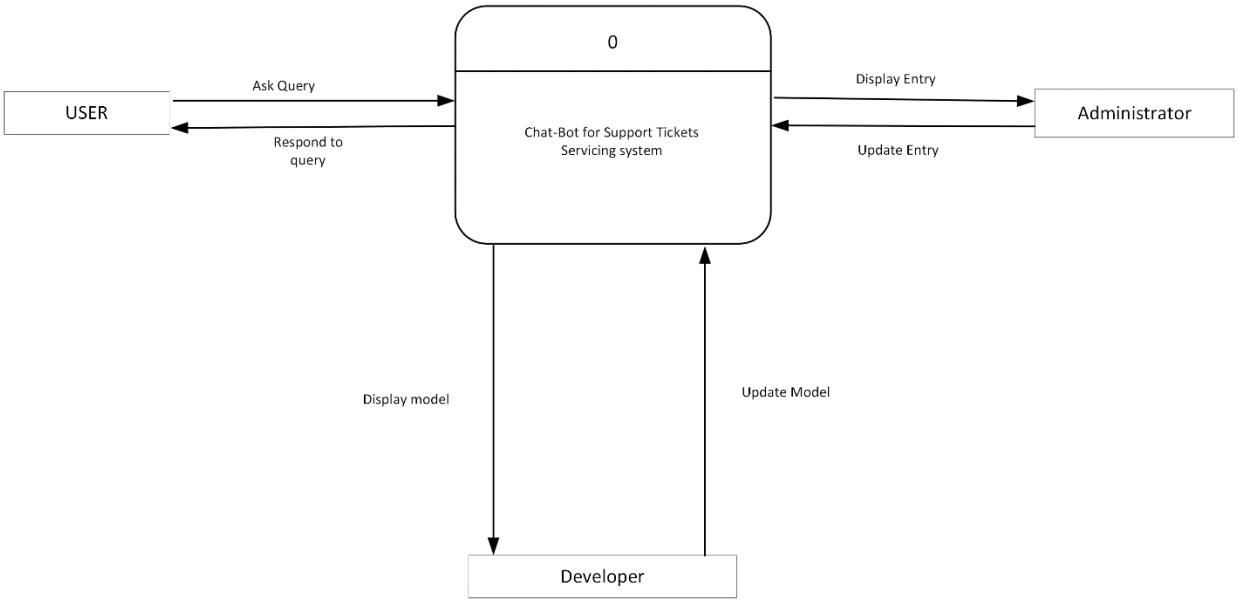


Figure 4.4: Context Diagram

4.4.4 Level 0 Data Flow Diagram

The level 0 diagram shows all the processes at the first level of numbering (i.e., processes numbered 1 through 3), the data stores, external entities, and data flows. The purpose of the level 0 DFD is to show all the major high-level processes of the system and how they are interrelated. All process models have one and only one level 0 DFD (Wixom et al., 2012).

Figure 4.5 shows a level 0 data flow diagram for the chatbot model, and it gives details of the system, illustrating the various processes in the modules, data stores, and entities.

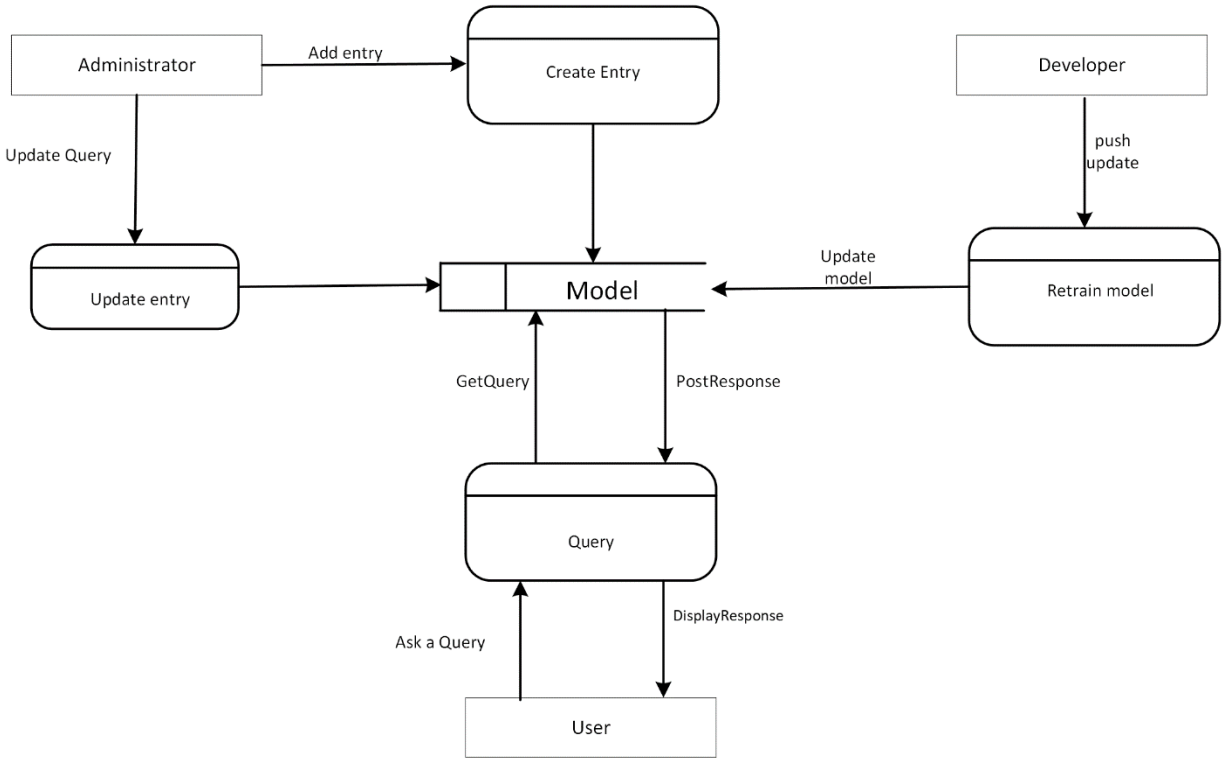


Figure 4.5: Level 0 Data Flow Diagram

4.4.5 Database Schema

The design features a simple database used to only house the queries database and administrators. In addition, user login sessions and client chat sessions are also maintained on the sessions table. This has been illustrated on Figure 4.6.

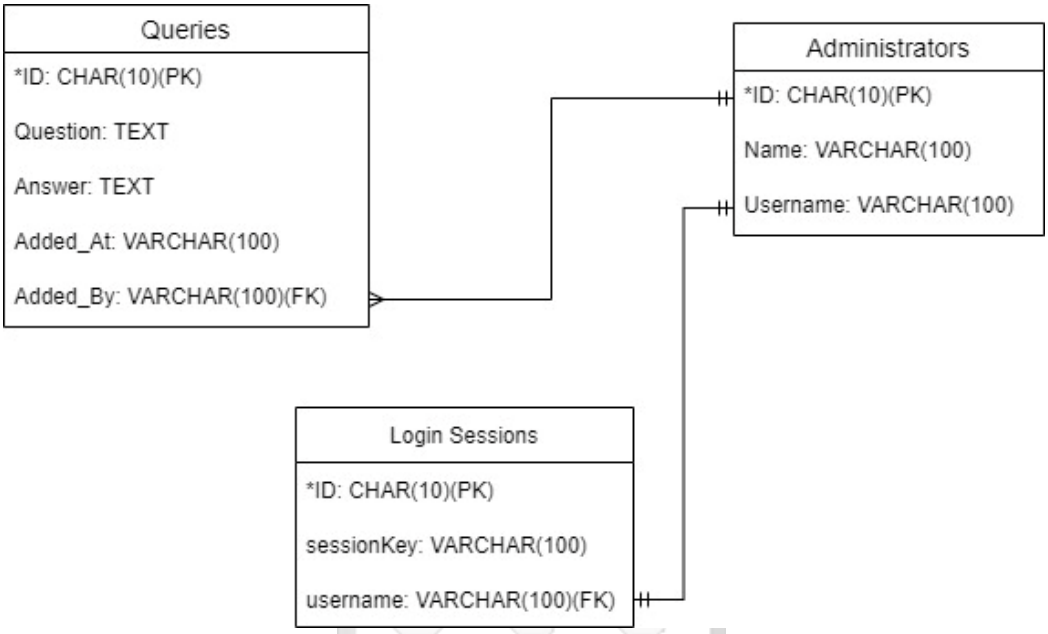


Figure 4.6: Database Schema

Chapter 5: System Implementation

5.1 Introduction

This chapter details the actual system implementation of the prototype. The testing of the prototype is also discussed to evaluate whether the chatbot meets the user requirements, the security requirements, and the implementation of the prototype. It also describes the system's workflow, which describes how the queries are solved. In addition to this, the development of the chatbot through Dialogflow CX.

5.2 The Workflow

The workflow is the specific steps followed to solve a customer query and give feedback to the customer.

The following steps are the steps that are undertaken for successful customer support.

i. The client sends an email to a designated email address
ii. The ticketing system receives the email.
iii. The email is logged as a new ticket into the system.
iv. A system admin changes the ticket's status from new to open and assigns it to another administrator who will act on the email.
v. Depending on the clarity of the issue, there may be a back and forth query loop between the admin and the requester(client) until the issue is resolved.
vi. The admin will finally close the ticket by changing its status from open to resolved.

The flow can be summarized in a flowchart as shown in Figure 5.1

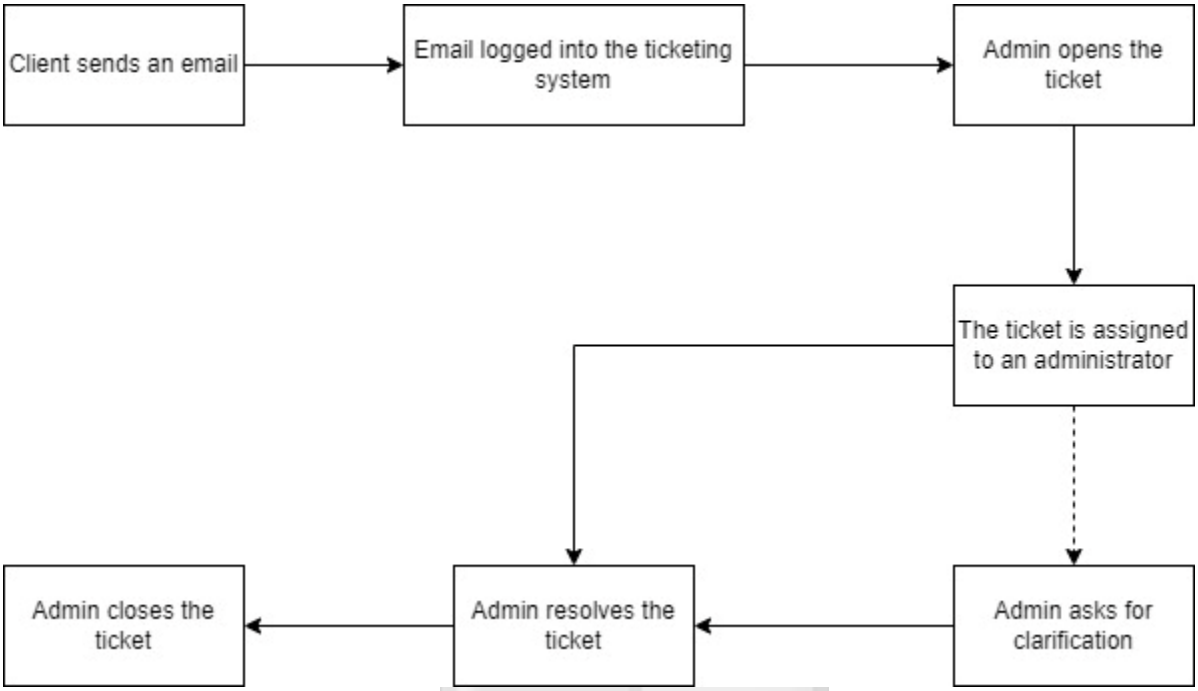


Figure 5.1: Client Support Ticketing Fowchart

5.3 Software Configuration

To build and test the prototype, the following steps were needed to be completed in the development, testing, and deployment environment.

i. Installation of Docker Daemon and Docker Desktop environments to host the database containers, the model container, and the web service container to expose the API, as shown in Figure 5.2.

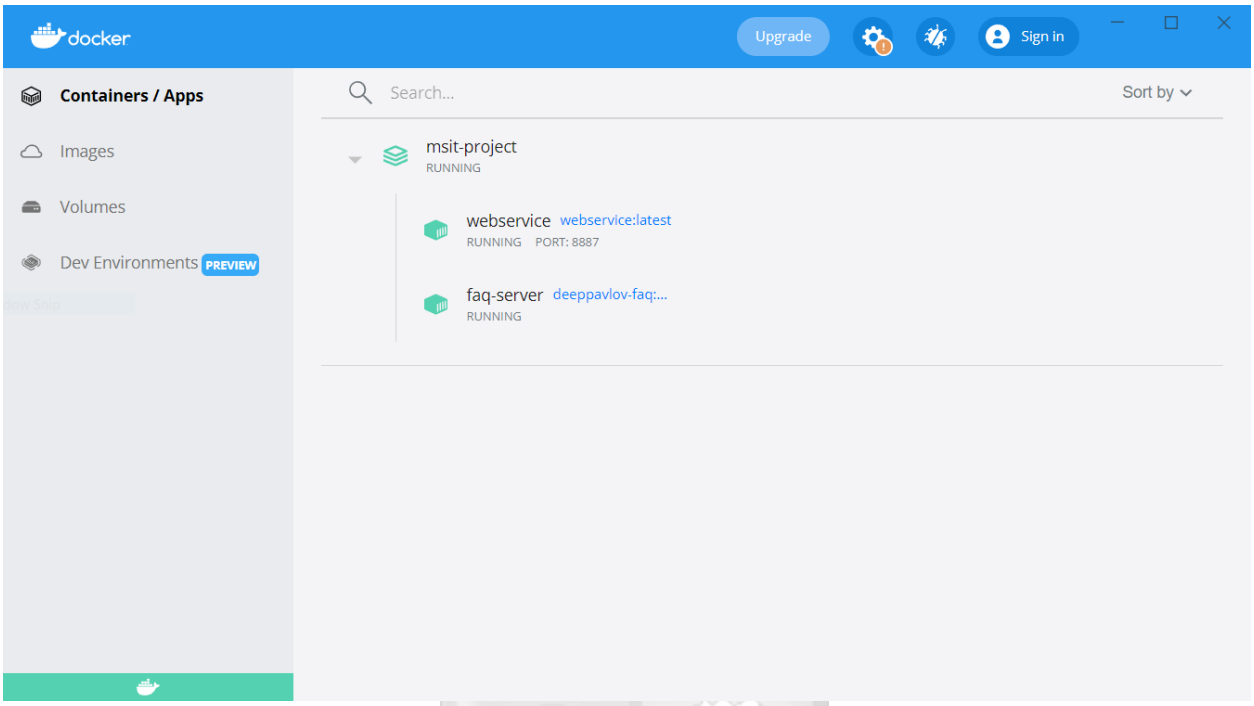


Figure 5.2: The Docker Desktop Interface

ii. Installation of Java 8, Java JDK, Python, and Development IDEs, i.e., PyCharm and IntelliJ IDEA. Figure 5.3 shows IntelliJ Idea's IDE interface.

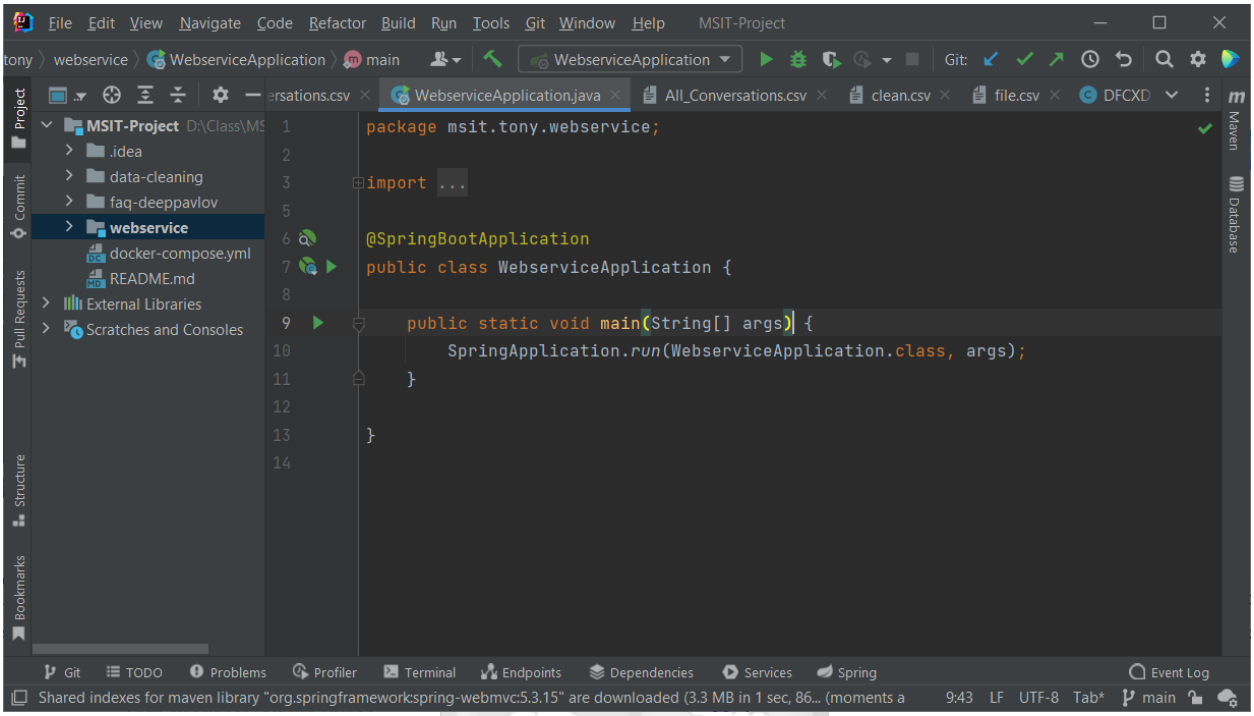


Figure 5.3 IntelliJ IDEA interface

iii. Create a Dialogflow account on https://dialogflow.cloud.google.com/cx/

iv. Create a developers' sandbox on Twilio at https://www.twilio.com/. Click on the "sandbox settings" menu to get a WhatsApp trial number, as shown in Figure 5.4.

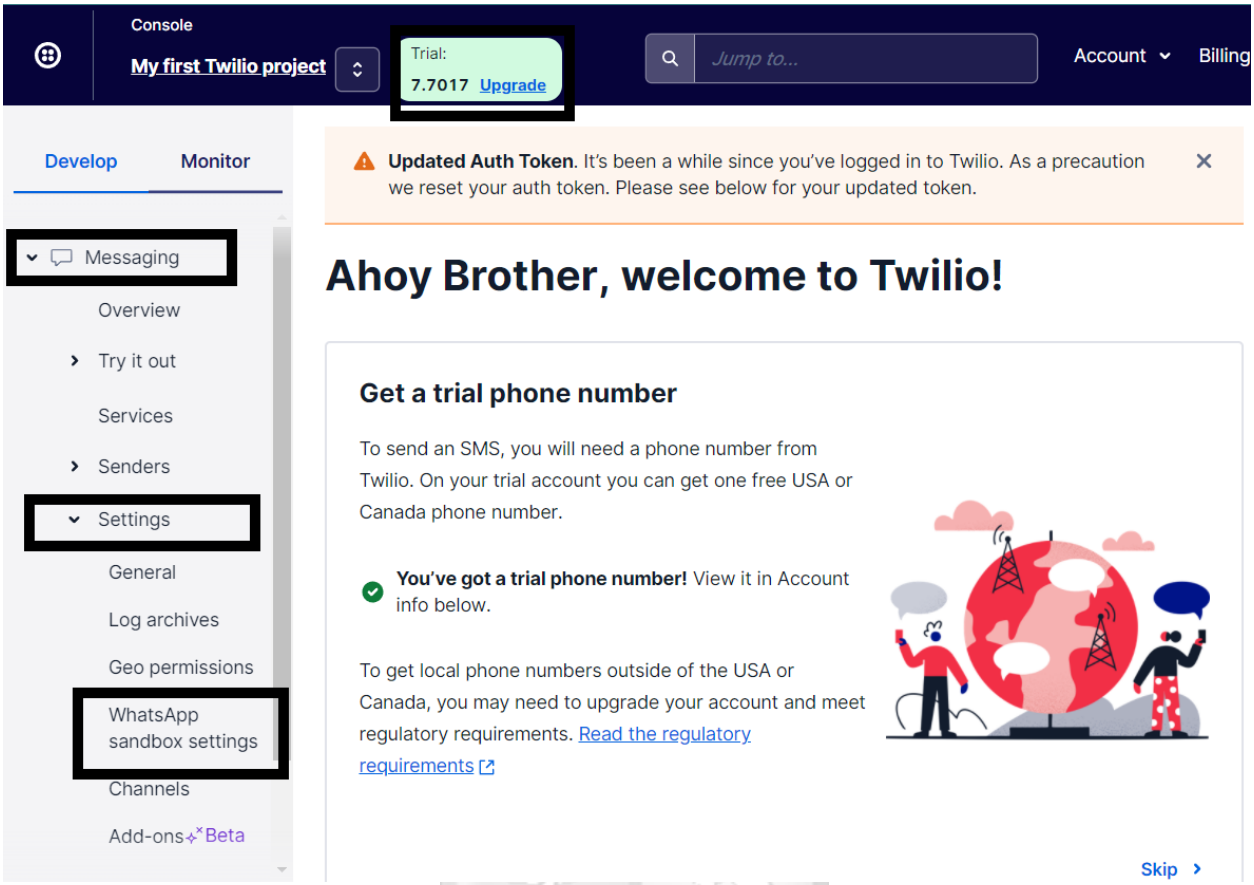


Figure 5.4: Twilio Account

v. Pull MariaDB docker container and from Docker Hub at https://hub.docker.com/_/mariadb. Figure 5.5 shows the MariaDB repository on Docker Hub.

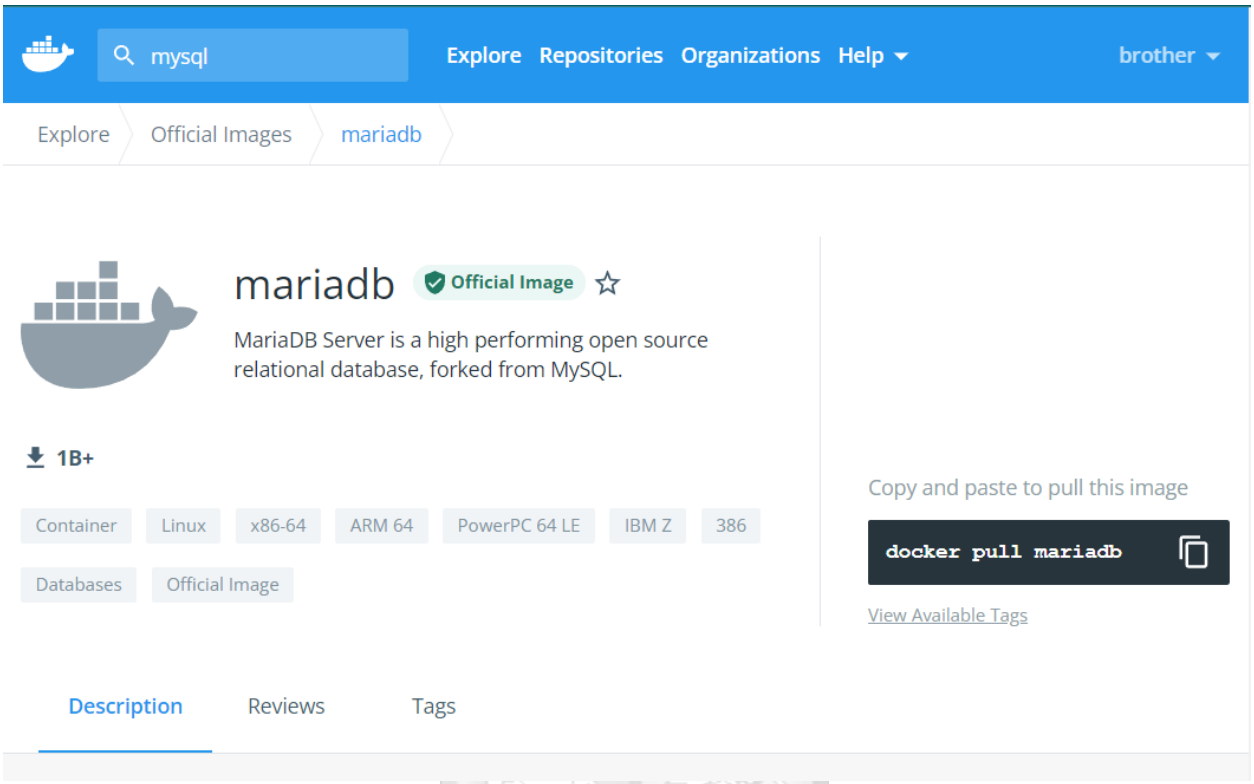


Figure 5.5: MariaDB container on docker hub

5.4 Bot Implementation

For more straightforward implementation and testing of the bot, the researcher used WhatsApp as the primary messenger application. To access the service, the researcher needed an active subscription on Twilio, where they were given a free WhatsApp contact where the clients could send and receive conversations. The researcher also provided a call-back URL where all messages would be forwarded to the provided number. Figure 5.6 shows a well-configured sandbox.

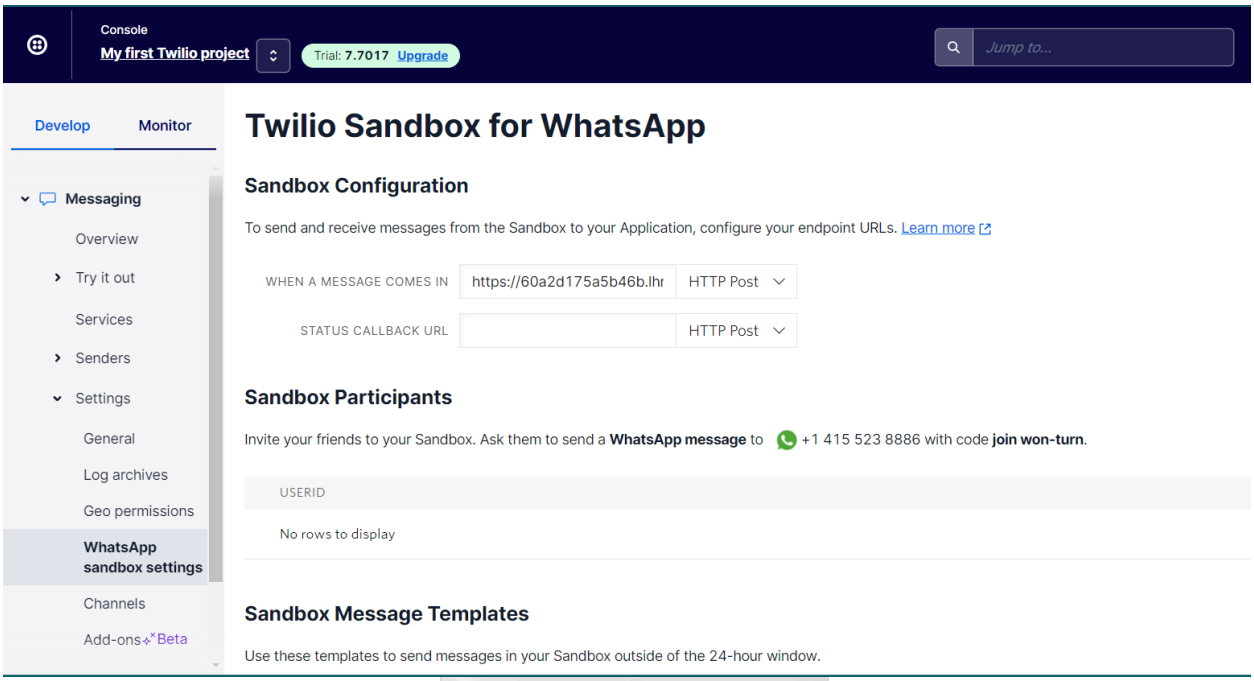


Figure 5.6: Twilio Sandbox for WhatsApp

After the sandbox had been set, the researcher was required to send an opt-in message to the test number provided by Twilio so that he would be able to receive messages from Twilio. Twilio then provides a twenty-four-hour window that the model can send a message to the client from the last message that came in. Only templated messages can be sent to the client, defined on the Twilio Platform. Figure 5.7 shows a successful opt-in. The messaging platform can now communicate with the client.

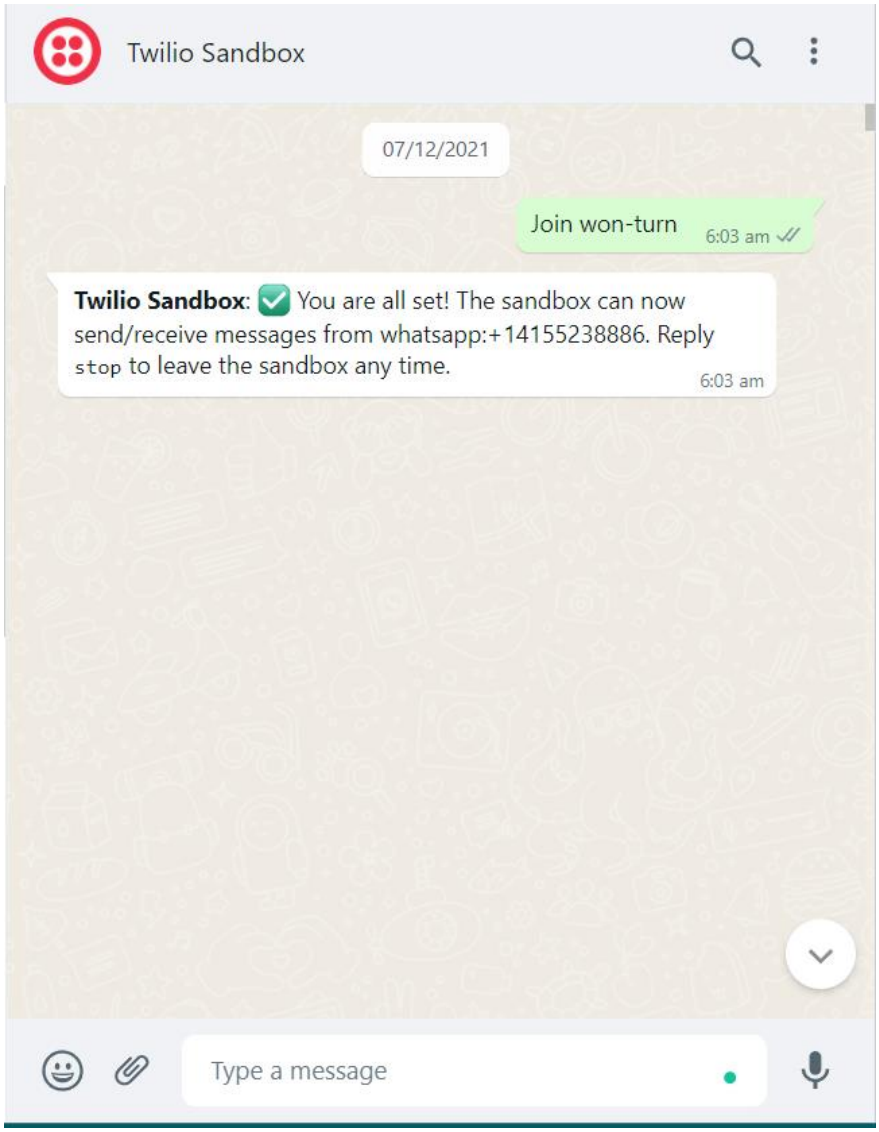


Figure 5.7: Successful opt-in on Twilio Sandbox

Next was to define a few intents on Dialogflow and add a couple of training phrases, as shown in Figure 5.8.

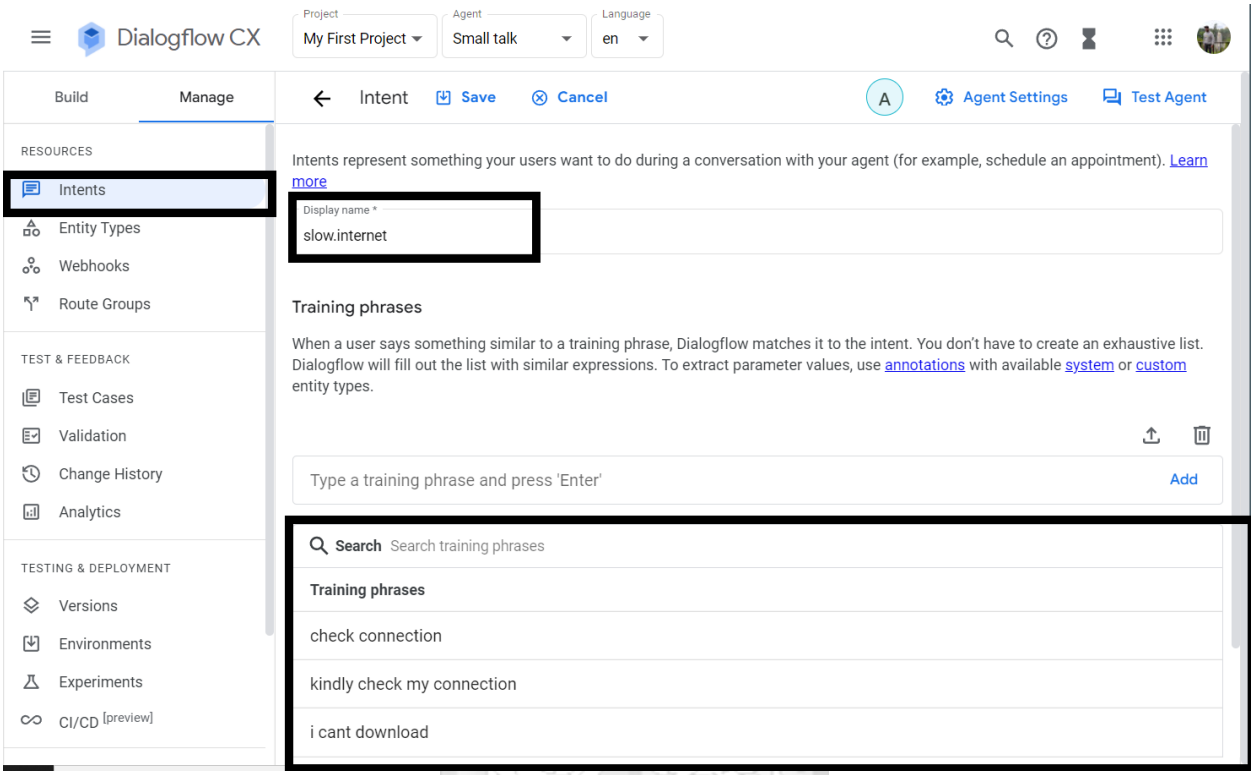


Figure 5.8: Defining an intent on Dialogflow

Once the intents were well defined, the researcher connected the flow on the Dialogflow build tab, as shown in Figure 5.9.

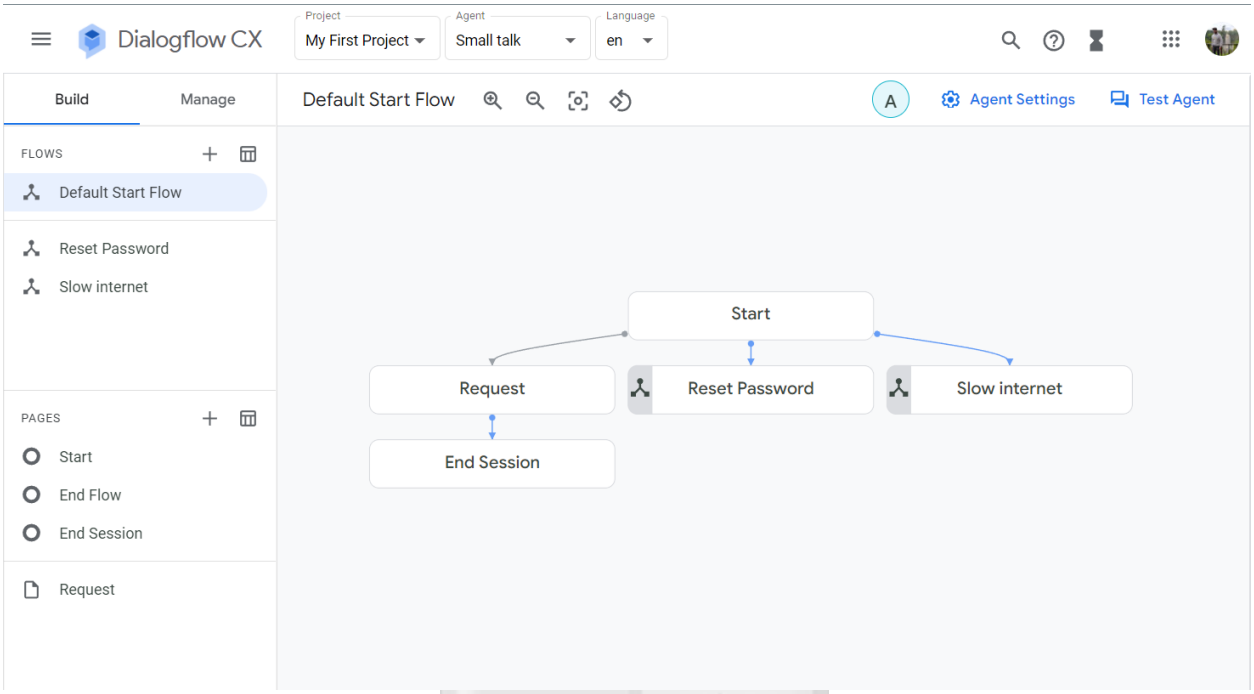


Figure 5.9: Dialog flowchart on Dialogflow

The clean question-answer CSV text data was imported into the DeepPavlov project folder. Next, the researcher configured a docker container to retrain the DeepPavlov model and exposed the RiseAPI endpoint at port 5000. The Dockerfile is configured as shown in Figure 5.10.

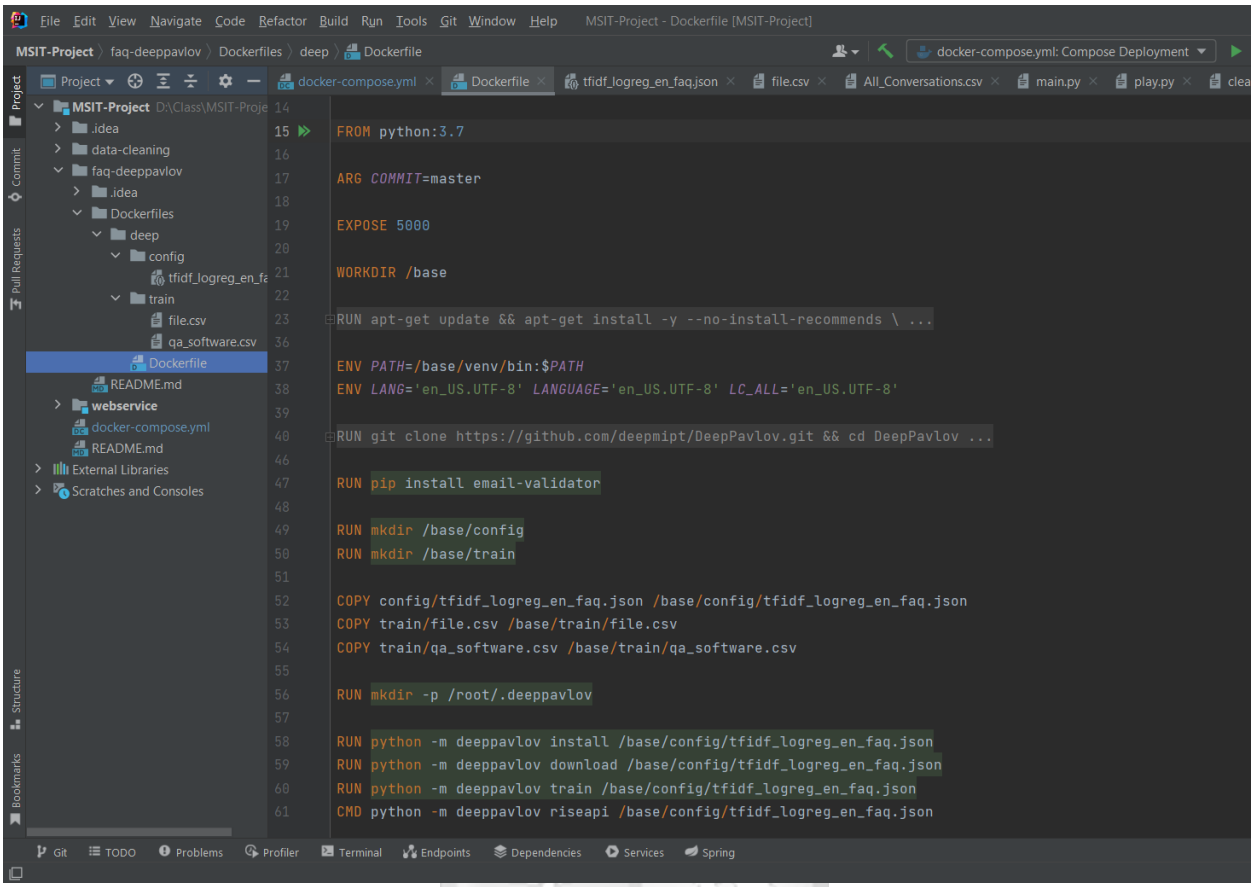


Figure 5.10: DeepPavlov Docker File.

The web service connected Dialogflow, DeepPavlov, and the messaging platform. A user session was created first; when no active session existed, a welcome message was sent to the user, and the user requested to log their issue. When the following message came in, the web service checked for predefined intents from Dialogflow. If the intent were present, the response from Dialogflow was forwarded to the user. If the intent was not found, the user's message was passed onto the DeepPavlov model to match it with the FAQ model, and the response was forwarded to the user.

The three services and the database were then tied together using Docker Compose to deploy the Digital Ocean container platform where they were hosted, as shown in Figure 5.11. Finally, all the code was pushed to GitHub, a code hosting and collaboration platform, to make it easier to version the system and deploy it on the Digital Ocean platform.

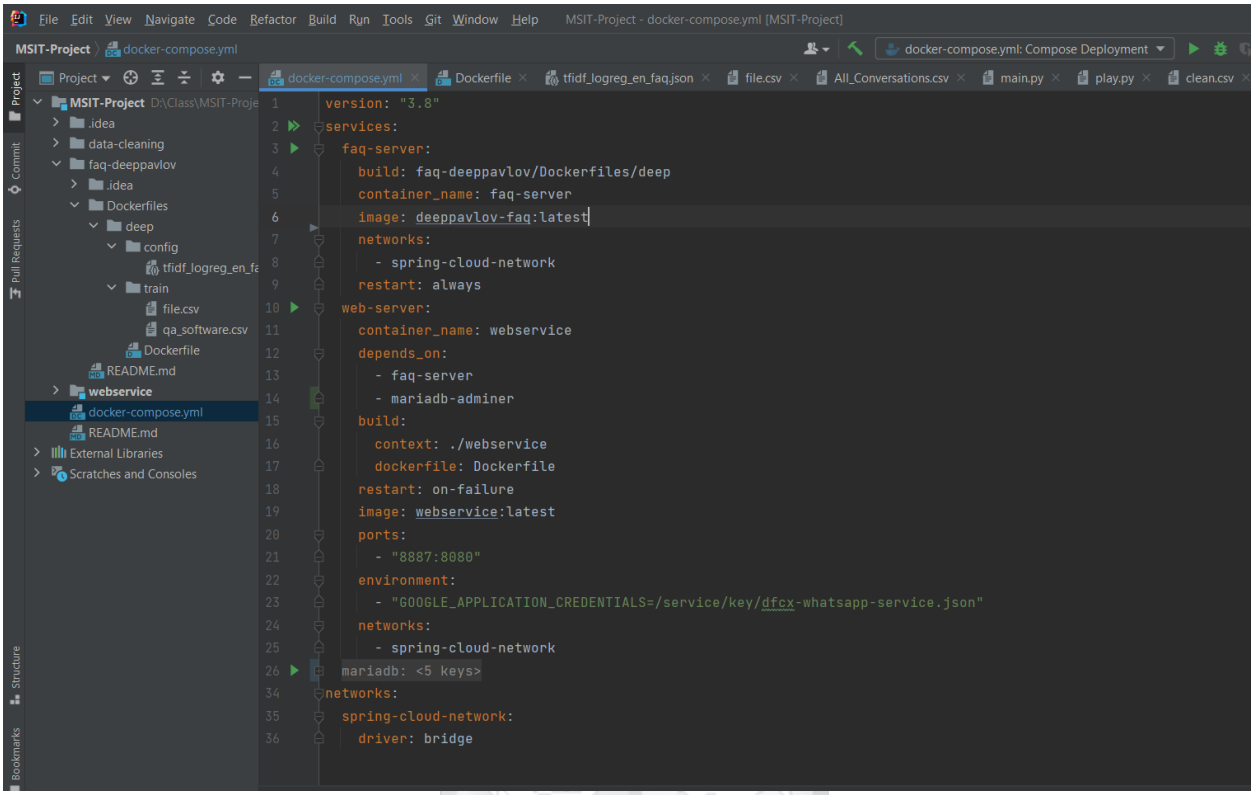


Figure 5.11: Docker Compose file.

The researcher set up a new project on the Digital Ocean Platform at https://digitalocean.com and created a new droplet – a droplet is a virtual machine instance at the platform, as shown in Figure 5.12. The new droplet was an Ubuntu virtual machine, 8 GB / 4 Intel CPUs, 160 GB NVMe Solid State Drive, 5 TB transfer for $48 a month billed per hour. This ensured that to minimize cost, the researcher would be able to create and destroy the droplet as they wished.

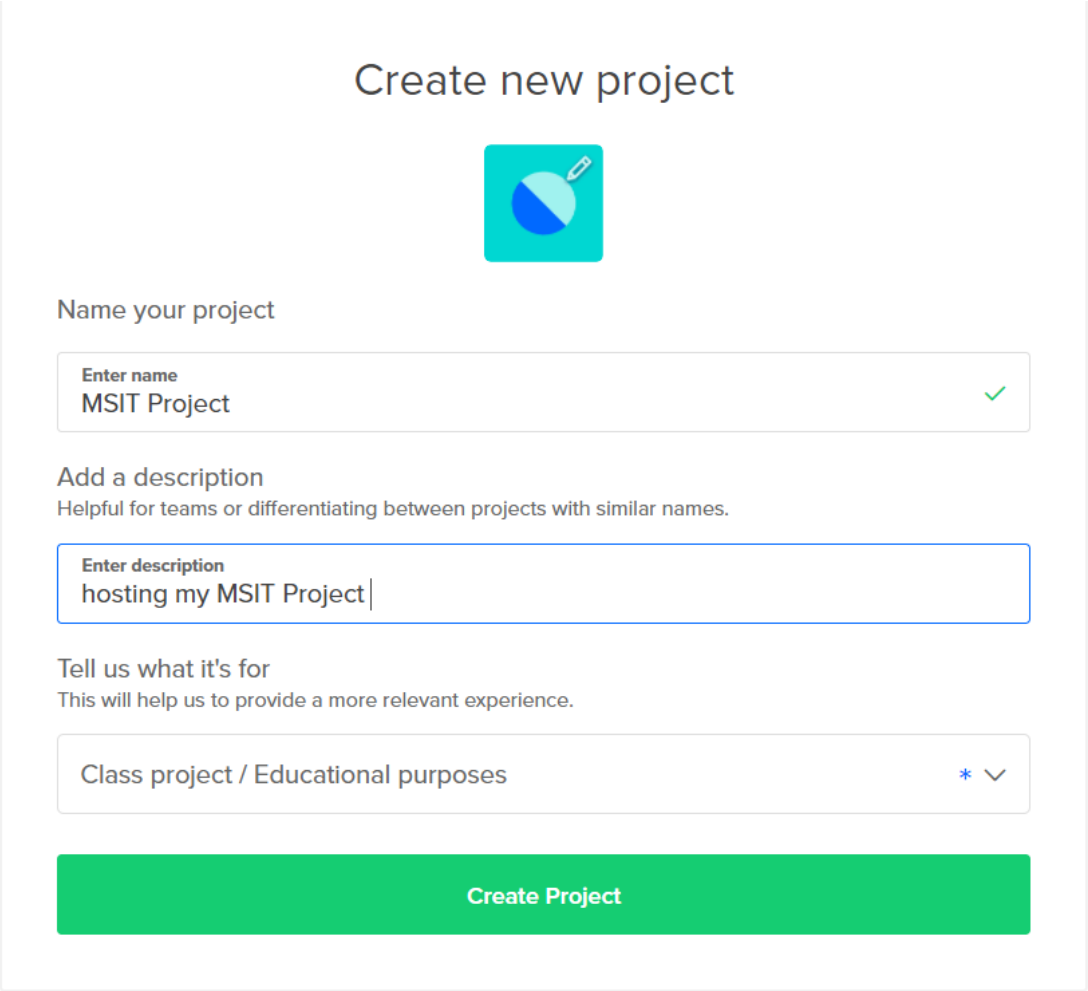


Figure 5.12: Creating a new Project on the Digital Ocean

The researcher pulled the code from GitHub and ran the docker-compose file to orchestrate all the modules with one-line command, as shown in Figure 5.13.

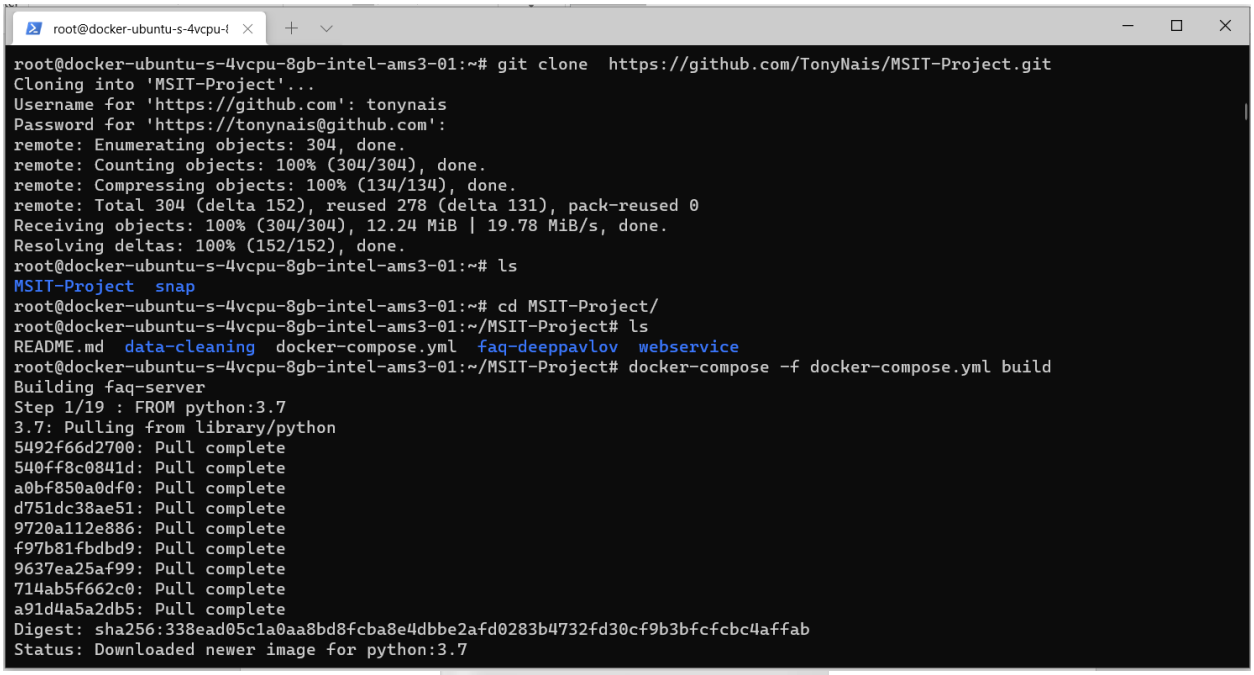


Figure 5.13: Pulling code from GitHub and building using Docker

After the successful deployment, the resultant URL was configured on the Twilio WhatsApp sandbox outbound message. The researcher sent a message to the sandbox and was able to opt in, create a new session, and even make small talk with the chatbot, as shown in Figure 5.14.

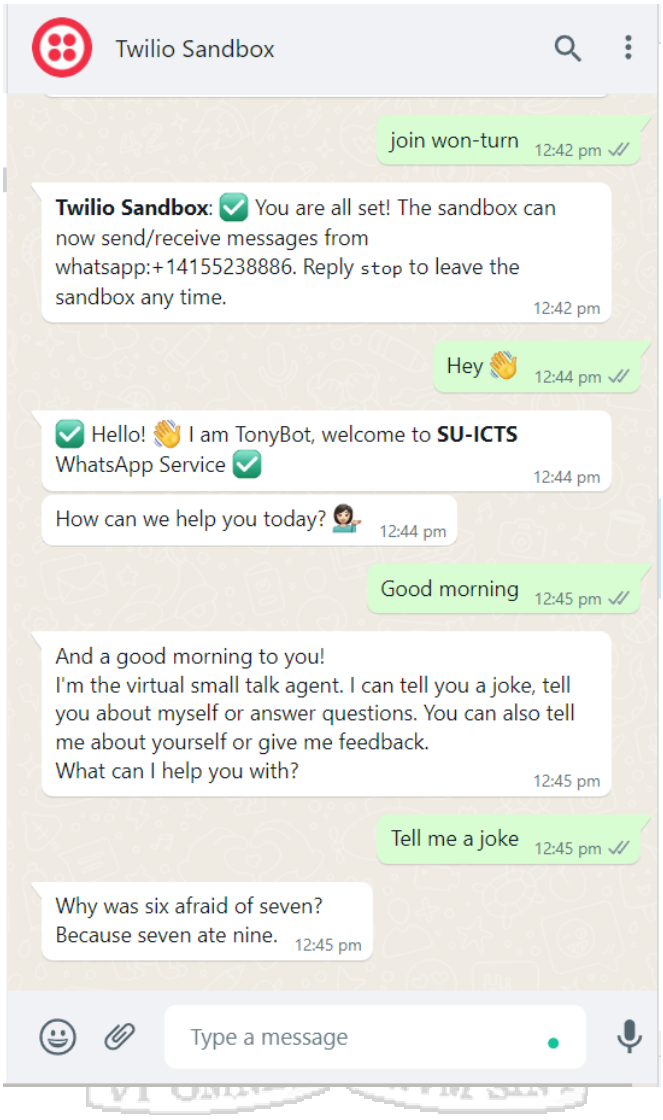


Figure 5.14: Opting in and making small talk

5.5 Development and Deployment Environment

The system was developed in the following hardware and software environment:

i. HP ProBook 445 G8 Notebook PC.
ii. AMD Ryzen 7 5800U with Radeon Graphics (16 CPUs), ~1.9GHz.
iii. Windows 10 Enterprise 64-bit (10.0, Build 19044).

iv. 16384MB RAM.
v. 1TB Hard Disk.
vi. IntelliJ IDEA 2021.3.3 (Ultimate Edition)
vii. PyCharm Professional 2021.3.3.
viii. DataGrip 2021.3.3.

The model was tested on the following hardware and software environment:

i. Ubuntu VM x64bit.
ii. 8 GB / 4 Intel CPUs.
iii. 160 GB NVMe Solid State Drive.
iv. 5 TB.
v. Docker Compose 1.29.2
vi. WhatsApp Messenger 2.22.7.71

5.6 Testing

Testing the chatbot was done by the cyber security team to ensure that it does not expose the existing systems it interacts with to external threats and expose customer data. In addition, the customer care user tested it to ensure that it meets the requirements and solves the common queries. Finally, quality control also tested the chatbot to ensure that it met the minimum standard for deployment.

Quality control and customer care tested for completeness of the chatbot; this is to ensure that the chatbot met all the performance specifications and provided all the functions specified by the customer care.

i) Correctness: The chatbot performed all the tasks correctly.
ii) Reliability: The chatbot could handle the normal flow of queries and peak loads of questions.
iii) Consistency: The chatbot solves all the queries, in the same manner each time.

iv) Efficiency: The chatbot solves queries with minimal resources within the shortest time.

v) Integrity: Only authorized personnel can run the chatbot. When the chatbot is done with a certain query, it logs out of the systems to make sure that no one else uses the sessions opened by the bot for accountability.

5.6.1 Functionality Testing

After the users, cyber security, and quality control did their tests, the researcher carried out functionality tests, and the results are shown in Table 5.1.

Table 5.1: Functionality Test Overview

No.	Functionality	Description	Status
1	Receive Messages	Receive messages from the user.	OK
2	Send messages	Send responses to the user.	OK
3	Logging	Logs system events in human-readable format for troubleshooting and maintenance.	OK
4	Intent Recorgnition	Recorgnize the user intention from the message received.	OK
5	Slot Filling	Ask the user for missing details in a message.	OK

Figure 5.15 shows a screenshot of the researcher's conversation with the prototype chatbot during functionality testing.

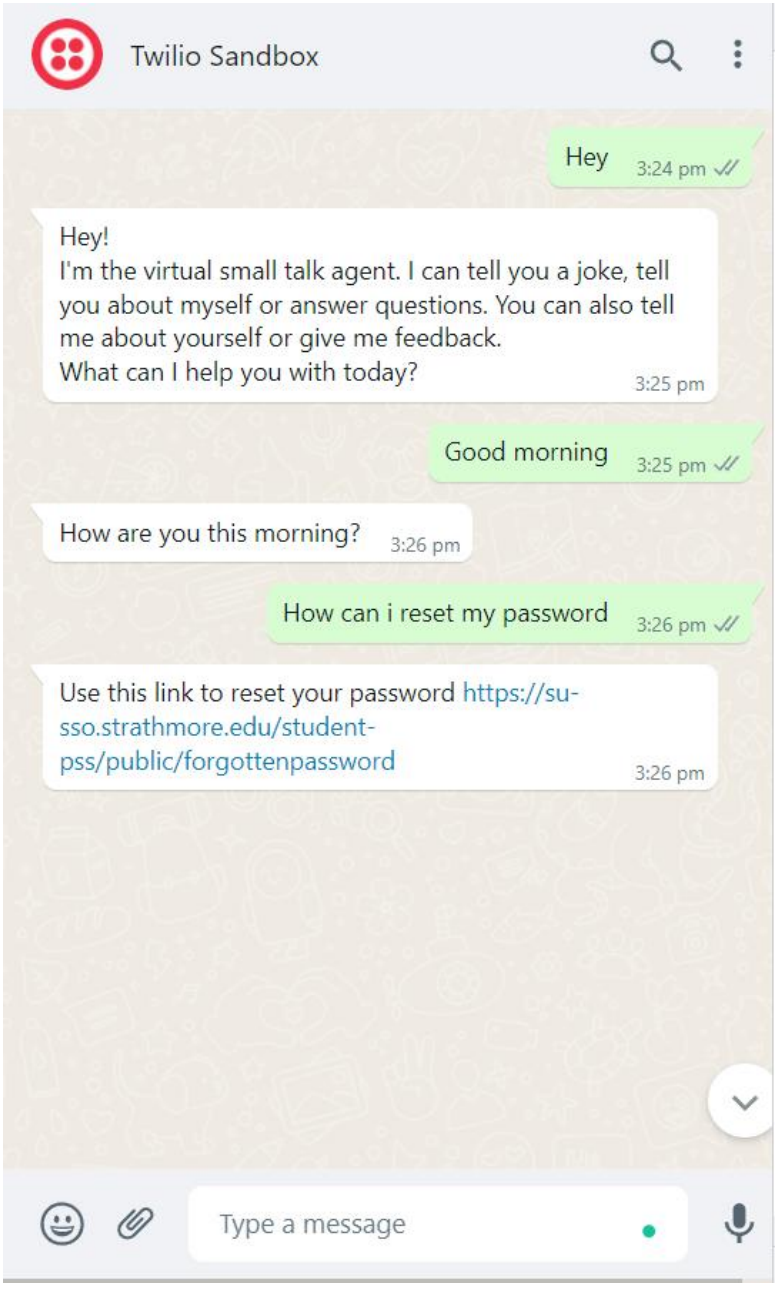


Figure 5.15: Functionality Testing

5.7 Maintenance of the Chatbot

Further development of this chatbot shall be performed regularly to scale it further to handle different functionalities beyond ticket resolution. Updates will be periodically pushed to the

containerized environment to ensure the latest version is always available on the droplet servers. Dialogflow and DeepPavlov systems will be patched and updated to ensure the latest features are always utilized. To match dynamic user requirements from the client support department, regular maintenance of the chatbot will be carried out to ensure that it fits the needs.

Chapter 6: Discussions

6.1 Introduction

This chapter discusses the research results according to the objectives set out in chapter one. The first objective of this research was to identify the challenges faced during customer issue resolution. Next was to find out the existing methods, techniques, approaches, and systems used for customer issues support, find out how an intelligent chatbot model for a ticket support system could be developed, and finally test the developed model. Business rules were set, and others refined to enable the smooth running of the chatbot and a standard process. The prototype chatbot was also created using the DeepPavlov library alongside Google's Dialogflow platform. Many tests were conducted to validate the performance of the prototype bot and the accuracy with which it operated.

6.2 Experimental Test Results

Using the Request Tracker system, giving clients responses after their queries have been sorted took about two hours for customer care staff to get customer details and send them their responses. There had to be a dedicated staff always manually checking for new tickets and responding to them. This meant that at least two staff members were dedicated to sending responses to the customers every day. The chatbot has enabled redeployment of those two staff to other tasks such as resolving issues, while the chatbot now makes the responses to the customers. The chatbot has also decreased the time taken from 2 hours on average to less than a minute. The time taken was measured using the Dialogflow Analytics tool as it has the capability to user engagement and path durations.

All the 37 sessions tested on Dialogflow were successful, as shown in Figure 6.1. This translates to a 100% success response rate. The average feedback duration for all the sessions was 0 minutes meaning the Dialogflow platform could respond to chat messages instantly. Back end code errors were at 0.00%.

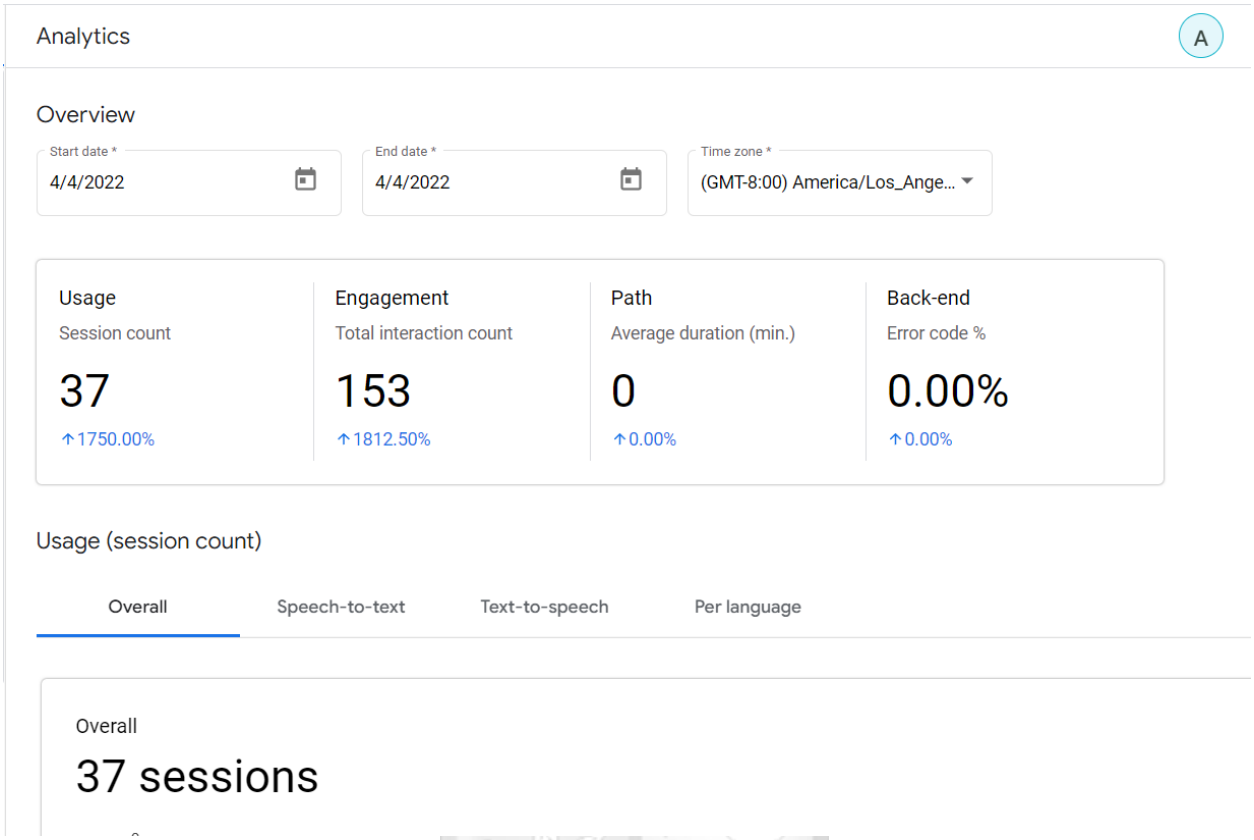


Figure 6.1: Experimental Test Results

6.2.1 Test Phases

The prototype chatbot was tested based on the various phases of testing. Table 6.1 shows the test phases that were carried out. The testing phases include security, user, unit, integration, system, and end-to-end testing.

Table 6.1: Test Phases

Type of Test	**Focus**
Security Testing	Security testing was done to confirm that the chatbot did not expose any personal data from the client, such as phone numbers or email.

User Testing	Users from the client support department were incorporated to validate the functional requirement of the chatbot.
Unit Testing	Individual components of the chatbot were tested separately to ensure that they were all working and that none would fail after integration.
Integration Testing	After being tested individually, different units of the chatbot were integrated to form the complete system. They were tested on how they integrated to form the system. They were also tested on how they integrated with the existing systems and platforms. Errors were debugged and confirmation was done that they were well integrated.
End-to-End Testing	The chatbot was validated by running it from the start of the process when it gets data from the model to when the chatbot terminated after successfully sending a response to the client.
System Testing	During this test, system was tested to make sure that it produced the right results and that it performed as expected.

6.3 Advantages of the Developed Prototype Chatbot

The framework that governed the creation of the prototype chatbot and the chatbot itself had numerous advantages. The framework helped streamline the processes by standardizing the client support process and guiding how the chatbot was developed. The framework ensures that all typographical errors from a client's input are autocorrected. It also helped in versioning the model and deploying new versions. It is also possible to have test cases on the platform to ensure user flows are thoroughly tested before deploying. The chatbot provides that a client would get a response to their query immediately. The chatbot establishes a single source of truth with clients and also enforces best practices. The chatbot would have various ways of responding to the same intent, making it not a static response to any conversation. The bot has also reduced the wastage of time by having to create tickets and share responses in separate emails. The bot also handled all small talk from clients efficiently.

6.4 Challenges Encountered

Wrangling and cleaning data from the Request Tracker proved to be a significant hurdle in realizing this research. The data contained a lot of unnecessary system-generated transactions, and all the email data held quoted emails inconsistently. Getting staff to appreciate and understand chatbot technology as a tool to augment them and not get rid of them. This proved to be a challenge as the team did not fully understand the whole technology. DeepPavlov is a high-end conversational framework with a very steep learning curve. Learning and implementing it on a research project was another challenge the researcher encountered.

6.5 Achievement of Objectives

This section discusses how the objectives of this research were met step by step. The first objective in section 1.3.2 was to investigate the challenges faced during customer issue resolution. The study shows several challenges faced by the client support services department. These challenges are explained in detail in section 2.2.1. These challenges are what this study is seeking to address through the development of the chatbot.

The second objective of this research was to examine the existing methods, techniques, approaches, and systems used for customer issues support. While addressing the challenges of customer ticket resolution, the researcher looked into various systems used in support desk services, including Request Tracker and Freshdesk. Request Tracker, the primary system used by the casing point organization, and in-depth processes and techniques have been discussed in depth in this research document.

The third objective of this research was to develop a prototype chatbot for client ticket resolution. This objective aimed to prove that the researcher could create a chatbot suited for the client support department by using artificial intelligence and machine learning, as highlighted in Chapter 2 and the steps highlighted in Chapter 4 for creating a chatbot. The development process of the prototype chatbot is discussed extensively in Chapter 5. By mitigating the weaknesses of the reviewed frameworks and borrowing from their strengths, the researcher developed a prototype chatbot that

could give feedback to the clients for the queries they had raised. The advantages of this prototype chatbot have been discussed in section 6.3.

The fourth objective of this research was to test the prototype chatbot. This ensured that the bot achieved its purposes and passed all standard security measures. Several tests that the chatbot was subject to, as discussed in Section 6.2.1. The test results of the bot are discussed in section 6.2. It shows the number of times the chatbot was tested, the duration, and the comprehensive test results. This includes the number of times the Chabot was successful and the number of times the bot failed, and the number of times the researcher stopped the chatbot for Section 6.2 also shows the time that the Chabot takes to execute successfully.

Chapter 7: Conclusion and Recommendations

7.1 Introduction

Customer support is the backbone of every company. For a company to maintain customers as clients, good quality customer care is fundamental. Big companies with a considerable customer base require a large workforce to handle all the customer queries. In the long run, it is not sustainable to keep employing staff to support the customers. So, there is a need for innovative ideas on utilizing technology to solve customer queries and maintain a manageable workforce.

Chatbot technology can be employed to help companies solve their customer service and, at the same time, remain competitive in the market. Much research has been done on machine learning and artificial intelligence on how best to implement this chatbot technology. This research has borrowed good practices from them. Those researches and frameworks have presented different approaches to how chatbots can be implemented. Some of the frameworks have been applied in this research.

The aim of this research was to provide a new approach through the application of chatbots to solve client support problems while still borrowing from what has been done. This research minimizes the human aspect to reduce the time taken to respond to a client as time is of the essence in a client support department while also reducing errors with human staff. In addition, lowering the cost of running a client support department through hiring and training new staff is also essential. The prototype chatbot is meant to help by speeding it up and streamlining the process by following set business rules and standards. Due to the person-hours involved in responding to customer queries, the chatbot has helped as the staff were redeployed to other tasks that required human intelligence and skills to solve.

In this research, it was necessary to understand the background of client support processes by reviewing relevant literature and participant observation of the processes involved. The literature also helped gain an insight into the various errors in the process and failure points. The multiple rules were extracted from the procedures and standards set on how the process should proceed.

The chatbot was only restricted to responding to client's queries that the client support staff had already resolved. Finally, several experiments were carried out to determine the time taken by the chatbot to accomplish the tasks and compared to the manual process of sending responses to the clients.

7.2 Contributions of the Research

This research has brought about a new way of dealing with client support queries, especially the efficient response part. It has addressed the issue of staffing which is a constant problem in client support. It has also eased the congestion experienced by the number of responses and reduced the time taken for the client to get feedback. This research has also helped streamline and standardize the client support department process, especially data formats. The study has helped the institution appreciate chatbot technology, understand what chatbots are all about, and identify other candidate processes for chatbot automation. Through this research, staff has realized that robots can coexist with human beings, and they are not here to replace human beings but rather to augment them and make their work easier. Through this study, the average time taken to respond to the customers has been reduced from about two hours to less than a minute.

7.3 Recommendations and Future Work

This research proposed developing a framework for chatbot technology that could be used in client support by developing a prototype chatbot. The prototype chatbot created by following the framework can be deployed in the client support department and other departments. This framework can also be used to develop chatbots for more prominent companies than the one used in this research and different case scenarios.

For this research, the prototype chatbot was limited to sending responses to a client only if a similar issue had been raised before and resolved by client staff. In the future, the scope could be enlarged to accommodate open domain question answering where the chatbot can scrap through various knowledge bases in the university and respond to any questions. The whole process is to be handled by the chatbot without human interactions. Future works should also concentrate on having more

than one chatbot working on the same case and communicating to share and pass data between themselves. This would further reduce the time a single bot will take to resolve queries and increase efficiency, especially in many business rules processes. Instead of only interacting with the bot through text alone, in the future, voice commands could also be added so that a client can interact with the chatbot verbally.

This research recommends that the prototype chatbot be extended to other institution departments as it will give out a significant return on investments. This research also suggests building capacity in different departments and upskilling the staff that the chatbot has taken their tasks. This will enable staff to maintain the existing bot and create new chatbots.

References

Acuna, S. T., & Juristo, N. (2006). *Software Process Modeling*. Springer Publishing.

Biswas, M. (2018). *Beginning AI Bot Frameworks: Getting Started with Bot Development* (1st ed.). Apress.

Craglia M., (Ed.), Annoni A., Benczur P., Bertoldi P., Delipetrev B., De Prato G., Feijoo C., Fernandez Macias E., Gomez E., Iglesias M., Junklewitz H, López Cobo M., Martens B., Nascimento S., Nativi S., Polvora A., Sanche. I., Tolan S., Tuomi I., Vesnic Alujevic L., *Artificial Intelligence - A European Perspective.*, EUR 29425 EN, Publications Office, Luxembourg, 2018, ISBN 978-92-79-97218-5, dos:10.2760/936974, JRC113826.

Cunningham, P., Cord, M., & Delany, S. J. (n.d.). *Supervised Learning. In Machine Learning Techniques for Multimedia (pp. 21–49).* Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-540-75171-7_2.

Customer Support Software & Ticketing System | Freshdesk. (n.d.). Freshdesk. Retrieved October 25, 2021, from https://freshdesk.com/customer-support-glossary/ticketing

DeepPavlov: an open source conversational AI framework. (n.d.). DeepPavlov.AI. Retrieved March 25, 2022, from https://deeppavlov.ai/

Dialogflow Documentation. (n.d.). Google Cloud. https://cloud.google.com/dialogflow/docs/

Delipetrev, B., Tsinaraki, C. & Kostic, U., *Historical Evolution of Artificial Intelligence, EUR 30221 EN, Publications Office of the European Union, Luxembourg*, 2020, ISBN 978-92-76-18940-4, doi:10.2760/801580, JRC120469.

Deng, L., & Liu, Y. (2018). *A Joint Introduction to Natural Language Processing and to Deep Learning. In Deep Learning in Natural Language Processing (pp. 1–22).* Springer Singapore. https://doi.org/10.1007/978-981-10-5209-5_1.

Dennis, A., Wixom, B., & Tegarden, D. (2015). *Systems Analysis and Design: An Object-Oriented Approach with UML* (5th ed.). Wiley.

Gentsch, P. (2018). *AI in Marketing, Sales and Service: How Marketers without a Data Science Degree can use AI, Big Data and Bots* (1st ed. 2019 ed.). Springer. https://doi.org/10.1007/978-3-319-89957-2

Gomaa, H. (2011). *Software Modeling and Design: Uml, Use Cases, Patterns, and Software Architectures*. Cambridge University Press.

Greene, D., Cunningham, P., & Mayer, R. (n.d.). *Unsupervised Learning and Clustering. In Machine Learning Techniques for Multimedia (pp. 51–90).* Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-540-75171-7_3.

Gupta, V. (n.d.). *Freddy | Agent Assist AI*. Freshdesk. https://freshdesk.com/freddy-ai-for-cx/agent-assist

Jilcha Sileyew, K. (2020). *Research Design and Methodology*. Cyberspace. Published. https://doi.org/10.5772/intechopen.85731.

Khan, R., & Das, A. (2017). *Build Better Chatbots: A Complete Guide to Getting Started with Chatbots* (1st ed.). Apress.

Luo, X., Tong, S., Fang, Z., & Qu, Z. (2019). *Frontiers: Machines vs. Humans: The Impact of Artificial Intelligence Chatbot Disclosure on Customer Purchases*. Marketing Science. https://doi.org/10.1287/mksc.2019.1192.

Nadkarni, P. M., Ohno-Machado, L., & Chapman, W. W. (2011). *Natural language processing: an introduction*. Journal of the American Medical Informatics Association, 18(5), 544–551. https://doi.org/10.1136/amiajnl-2011-000464.

Ng, A. (2017, March 11). *Andrew Ng: Why AI Is the New Electricity*. Stanford Graduate School of Business. https://www.gsb.stanford.edu/insights/andrew-ng-why-ai-new-electricity

Raj, S. (2019). *Building Chatbots with Python*. Apress. https://doi.org/10.1007/978-1-4842-4096-0.

Singh, A. (2019, December 6). *What Is Rapid Application Development (RAD)?* Capterra. https://blog.capterra.com/what-is-rapid-application-development/

Tobin, J. L., & Akhilesh, S. (2011). *Software Development as a service: Agile Experiences. annual SRII Global.*

Vincent, J., Spier, R., Rolsky, D., Chamberlain, D., & Foley, R. (2005). *RT Essentials: Managing Your Team and Projects with Request Tracker* (1st ed.). O'Reilly Media.

Wit.AI. (n.d.). Wit.ai. https://wit.ai/

Wixom, B. H., Roth, R. M., & Dennis, A. (2012). *Systems Analysis and Design* (5th ed.). Wiley.

Xu, A., Liu, Z., Guo, Y., Sinha, V., & Akkiraju, R. (2017, May 2). *A New Chatbot for Customer Service on Social Media. Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems.* CHI '17: CHI Conference on Human Factors in Computing Systems. https://doi.org/10.1145/3025453.3025496

Zhou, W., Xue, W., Baral, R., Wang, Q., Zeng, C., Li, T., Xu, J., Liu, Z., Shwartz, L., & Ya. Grabarnik, G. (2017). STAR. *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. Published. https://doi.org/10.1145/3097983.3098190

Appendix A: Originality Report

Curiginal

Document Information

Analyzed document	A Machine Learning Model for Support Tickets Servicing A Case of Strathmore University ICTS Client Support Services.docx (D133326633)
Submitted	2022-04-11T10:16:00.0000000
Submitted by	
Submitter email	antony.Koimbi@strathmore.edu
Similarity	2%
Analysis address	library.strath@analysis.urkund.com

Appendix B: Data Consent Form

28/03/2022

To whom it may Concern.

This is to certify that **Antony Koimbi Maina**, student number **114731**, has been authorized to use data from the University's Service Desk system "Request Tracker" for his research thesis titled: **A Machine Learning Model for Support Tickets Servicing: A Case of Strathmore University ICTS Client Support Services.**

These data should be used only for academic research and not any other purpose.

Your Faithfully

Stephen Momanyi

Ag. Director, ICTS

Ole Sangale Rd, Madaraka Estate. PO Box 59857-00200, Nairobi, Kenya. Tel +254 (0)703 034000
Email admissions@strathmore.edu www.strathmore.edu

Appendix C: SU-IERC Ethical Approval

Strathmore UNIVERSITY

4th April 2022

Mr Maina Antony,
antony.koimbi@strathmore.edu

Dear Mr Maina,

RE: A Machine Learning Model for Support Tickets Servicing: A Case of Strathmore University ICTS Client Support Services

This is to inform you that SU-IERC has reviewed and **approved** your above **SU masters'** research proposal. Your application reference number is **SU-IERC1237/21.** The approval period is **4th April 2022 to 3rd April 2023.**

This approval is subject to compliance with the following requirements:

i. Only approved documents including (informed consents, study instruments, MTA) will be used
ii. All changes including (amendments, deviations, and violations) are submitted for review and approval by SU-IERC.
iii. Death and life-threatening problems and serious adverse events or unexpected adverse events whether related or unrelated to the study must be reported to SU-IERC within 48 hours of notification
iv. Any changes, anticipated or otherwise that may increase the risks or affected safety or welfare of study participants and others or affect the integrity of the research must be reported to SU-IERC within 48 hours
v. Clearance for export of biological specimens must be obtained from relevant institutions.
vi. Submission of a request for renewal of approval at least 60 days prior to expiry of the approval period. Attach a comprehensive progress report to support the renewal.
vii. Submission of an executive summary report within 90 days upon completion of the study to SU-IERC.

Prior to commencing your study, you will be expected to obtain a research license from National Commission for Science, Technology, and Innovation (NACOSTI) https://research-portal.nacosti.go.ke/ and obtain other clearances needed.

Yours sincerely,

for: **Dr Ben Ngoye,**
Secretary; SU-IERC

Cc: Prof Fred Were,
Chairperson; SU-IERC

STRATHMORE UNIVERSITY INSTITUTIONAL ETHICS REVIEW COMMITTEE
(SU-IERC)
04 Apr 2022
Email:ethicsreview@strathmore.edu
P.O BOX 59857-00200
NAIROBI-KENYA