Chapter 1

Introduction

1.1 Artificial Intelligence and Machine Learning

Artificial intelligence (AI) is the broad concept of machines being able to carry out functions and tasks using human-like intelligence. AI devices can be divided into two groups – applied AI and general AI.

Applied AI is also known as narrow AI. It consists of systems that can carry out tasks without being explicitly programmed to do so, such as language recognition, recommendation engines, and vision recognition. General AI consists of systems that use adaptable intellect to complete various tasks.

Machine learning (ML) is a subgroup of AI. It is made up of a set of statistical algorithms that allow a computer to 'learn', i.e. improve its performance at a particular task as it gains experience in performing the task. These algorithms predict the outcome of a function, within a certain degree of error, based on known attributes in the data related to previous results obtained while performing the task.

Like AI, machine learning techniques can be broadly classified into the following learning categories:

1. Supervised learning: The data supplied to the learning algorithm consists of labels, and the algorithm uses these samples to build and learn a mapping from the input supplied to the output, for example regression, and classification.
2. Unsupervised learning: The data supplied to the learning algorithm is not labelled, and it must find some structure in the input data itself, for example clustering, and anomaly detection.
3. Semi-supervised learning: The learning algorithm is supplied with a set of data, of which only a small amount is labelled, and the rest is unlabelled. It falls between supervised and unsupervised learning.
4. Reinforcement learning: The learning algorithm takes a decision in a feedback-enabled environment so as to maximise its reward i.e. positive feedback.

Machine learning is a practical application of AI based on the idea that machines can learn and improve at a task from a given set of data. The algorithms allow for the construction of various mathematical models to enable a computer to predict a result. Machine learning is used in situations where designing explicit programming algorithms with good performance is infeasible or difficult, such as computer vision, email filtering, and recommendation systems.

1.2 Natural Language Processing

Natural language processing (NLP) is a subdomain of machine learning specific to linguistic applications.

The most common NLP tasks are:

1. Tokenization: Split a given sentence into a list of words.
2. Lemmatization: Determine the lemma of a given word, based in part on the POS tag of the word.
3. Part-of-speech (POS) tagging: Determine the part of speech for each word in a given sentence, for example whether a word is a noun, verb, adjective, and so on.
4. Sentiment analysis: Determine the polarity (positive to negative scale) and subjectivity (objective to subjective scale) of some text.
5. Named-entity recognition (NER): Determine the items in a given stream of text that map to proper names such as places and people.
6. Question answering: Determine the answer to a given human-language question.

NLP relies on machine and deep learning, as it allows for the use of statistical inference to automatically learn rules through the analysis of large corpora. These statistical models make probabilistic decisions based on attaching real-valued weights to each input feature. Such models can hence express the relative certainty of each multiple possible answers, producing more reliable results [1].

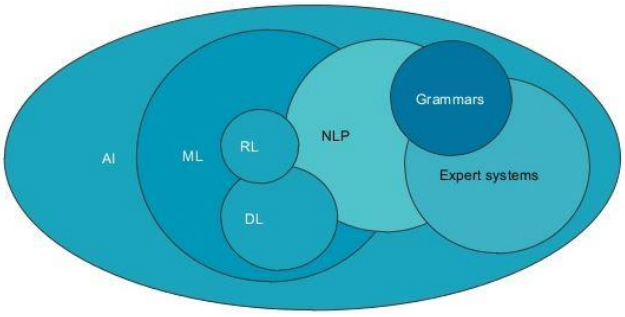

Figure 1.1: The relation between AI, ML, and NLP

1.3 Chatbots

A chatbot is a software application designed to mimic human a conversation with a human, either through speech or text. They are normally built as a part of a larger application, and can be used to engage users on a platform.

Chatbots can provide the user with quick responses or solutions that may otherwise take more time for the user to obtain from the larger application as a whole. Apart from this, the other advantages of chatbots are:

1. They are generally available at any time of a day.
2. Unlike a human agent, they can handle multiple customers at once.
3. They can be used to automate repetitive tasks.

However, chatbots also have disadvantages. Some of these are:

1. Chatbots have limited functionality, and a user might ask it to perform a task it doesn't understand. The repeated failure of the bot to complete the task might frustrate the user.
2. Chatbots can be expensive to build and maintain as they use AI-based technologies.
3. Complicated UIs might confuse users.

Due to the dramatic advancement of AI technology, chatbots have become extremely popular, especially in business-to-customer (B2C) industries, such as travel and tourism, banking, and delivery services. Nonetheless, chatbots are steadily becoming more common in the business-to-business (B2B) industry today, for example in content marketing, customer management, and logistics, as they can filter and summarise vast amounts of data in short time spans.

As NLP ranges from simple pattern matching to more complex question answering, it forms the basis of chatbots. Therefore, they can be categorised into the following:

1. Menu-based chatbots: These bots provide users with a set of options as an input, and respond only based on the menu item selected. They are the least complicated to build, but tend to offer the lowest quality of user experience.
2. Keyword recognition-based chatbots: These chatbots search a user's input for certain keywords. Depending on the keywords found, a response is generated. They are more complicated to build than menu-based bots.
3. Contextual chatbots: These bots use AI-based technologies to understand a user's statement and generate an appropriate response. They are the most complicated to build,

but offer can offer higher levels of user experience as they simulate an actual conversation with a human agent.

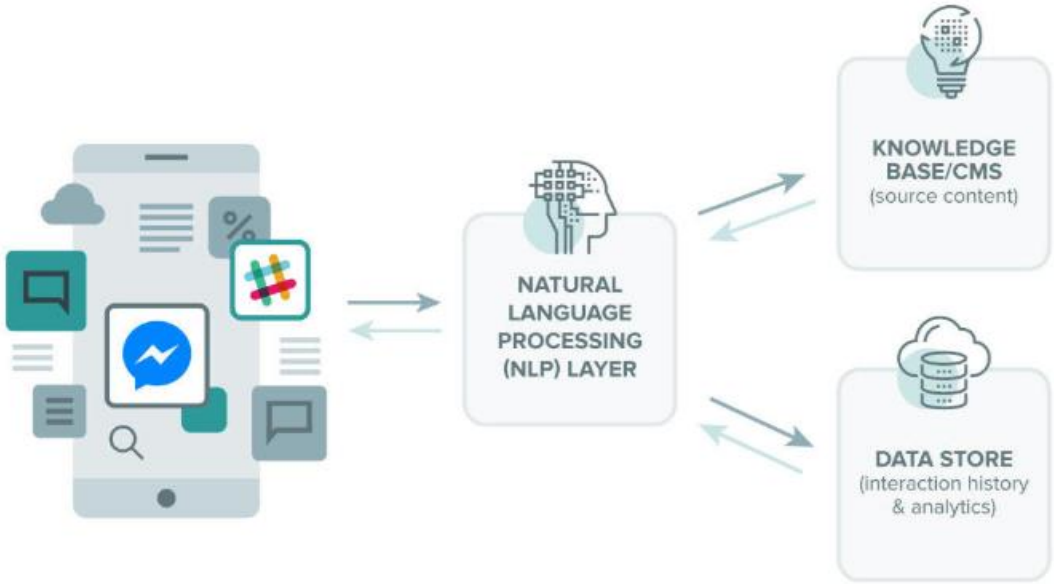

Figure 1.2: A contextual chatbot

The stages of a chatbot conversation are:

1. Take an input from a user
2. Understand the user's intention
3. Reply with a suitable response based on what the user has said

These stages are described in further detail in Section 2.1.2.

The response generated by a chatbot is governed by the set of rules that makes up a dialog flow. These rules can be based pattern matches through regular expressions, or intent matches through a NLP model.

Chapter 2

Background Theory and Literature Survey

2.1 Background Theory

2.1.1 Customer Relationship Management Systems

As mentioned in Section 1.3, a chatbot is normally built as a part of a larger software system. Pepper Cloud is a Singapore-based company whose core product is an intelligent Customer Relationship Management (CRM) system, targeted for use in the food and beverage (F&B) industry.

The Pepper Cloud CRM is structured as follows:

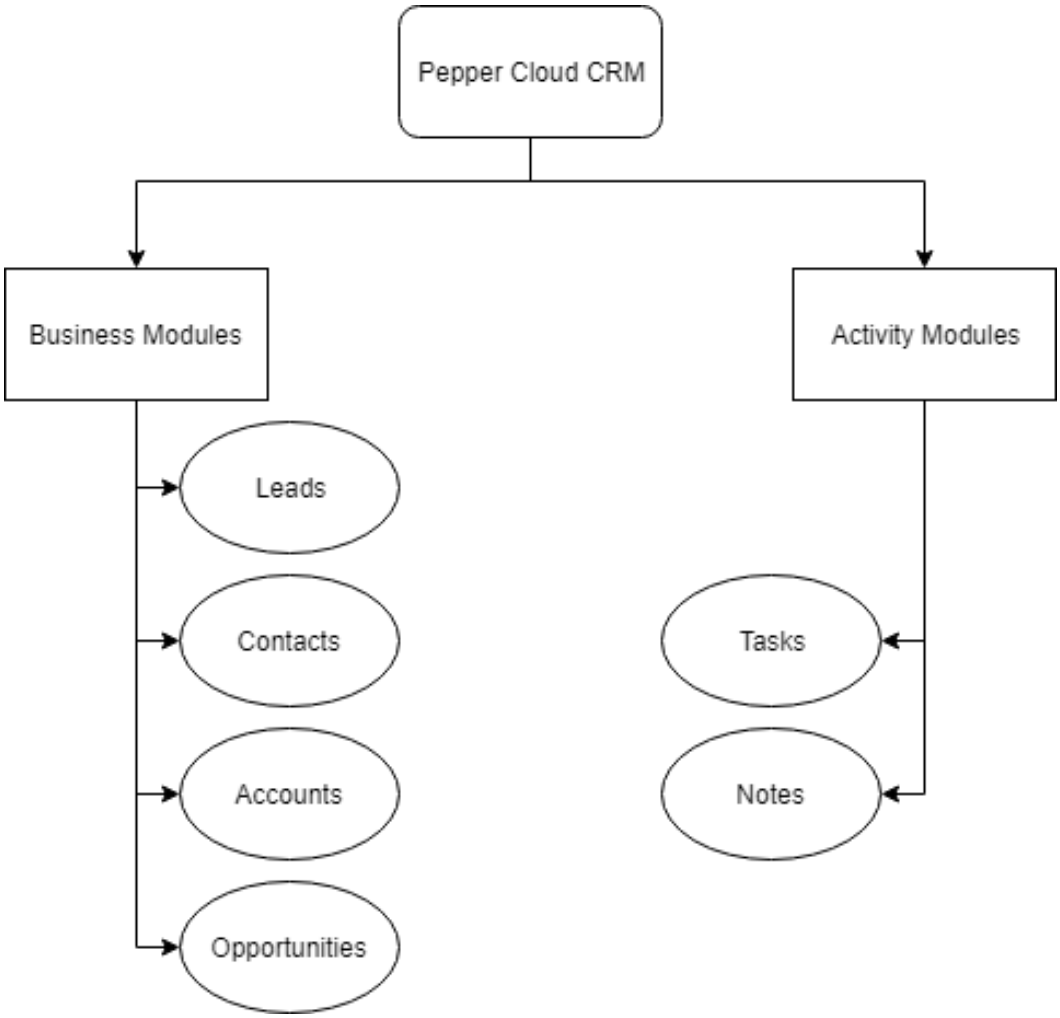

Figure 2.1: The Pepper Cloud CRM

The business modules (BM) of the CRM are subdivided into the following modules:

1. Leads: A lead is a potential business partner from another company. Each lead has a company, rating, source, and status.
2. Contacts: Once a lead is qualified, it is converted to a contact. A contact is the focal point for getting in touch with a company to create business opportunities for the user's firm.
3. Accounts: Accounts are the set of companies with which the user's firm has done, or is currently doing business.
4. Opportunities: Opportunities are business deals struck between the user's firm and another company. Each opportunity has amount, close date, source, and status.

The activity modules (ACTM) of the CRM are subdivided into the following modules:

1. Tasks: A task is a job that must be done by an employee of the firm using the CRM. A task has a due date, priority, status, and task type.
2. Notes: A note is a record of points and ideas.

2.1.1.1 CRM Associations

Associations are the relations between various modules of the CRM system. The following associations exist in the CRM:

Sr. No.	Association Type	Relationship
1	ACTM – BM	Many-to-one
2	BM – BM	One-to-one or One-to-many (case dependent)

Table 2.1: CRM Associations

Some examples of ACTM – BM associations are:

1. Multiple tasks can be associated with one lead/contact/account/opportunity.
2. Multiple notes can be associated with one lead/contact/account/opportunity.
3. The same task or note cannot be associated with two leads/contacts/accounts/opportunities

Some examples of BM – BM associations are:

1. One contact can be associated with one account at a time (one-to-one).
2. One account can be associated with multiple contacts (one-to-many).
3. One account can be associated with multiple opportunities (one-to-many).

2.1.1.2 CRM Permissions

Permissions in a CRM control the visibility and access of data for users in an organisation. Depending on their job requirements, designation, and experience, each organisation can customise the amount and type of data their employees can obtain from the CRM system.

In the Pepper Cloud CRM, the following permissions exist:

1. Read: The permission to view data in the system.
2. Create: The permission to add new data to the system.
3. Update: The permission to edit data present in the system.
4. Delete: The permission to remove data from the system.

The value set for each permission is Boolean: either *true* or *false*. Each permission type listed above is set for the following modules:

1. Accounts module
2. Activity module (tasks and notes as a whole)
3. Contacts module
4. Leads module
5. Opportunities module

2.1.2 Chatbots: Processing Inputs and Generating Responses

Language processing and replying to a user is an arduous task for a software application. To complete this function, a chatbot is divided into the following subtasks:

1. Receive an utterance: An utterance is anything that a user says. For example, in a CRM chatbot a user might ask the following: '*show all high priority tasks*'.
2. Identify intents and entities in the utterance: An intent is the user's intention, and an entity is the context of the intent. Entities modify an intent. In the example above, the user's intent is *display tasks*, and the varying entity is *high priority*.
3. Process the input using a dialogue flow: The dialogue flow of a chatbot controls the steps taken by the bot in a conversation with a user. This flow defines what must be

done by the system in the case of each possible user input in the domain of the chatbot i.e. intent matching. This matching can be done using NLP models trained in the relevant context, fuzzy matching, or equality matching.

4. Respond to the user: Respond to the user based on the result generated from the dialogue flow.

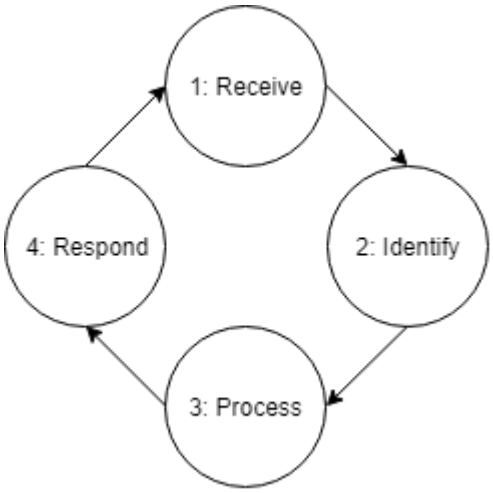

Figure 2.2: The chatbot cycle

2.2 Literature Survey

Today, chatbots have become one of the most popular practical applications a product combining AI, machine learning, and natural language processing. Research related to the following is being carried out to improve chatbots:

- Text generation
- Context-based intent matching
- Named entity recognition, and question answering
- Sentiment analysis

Dahiya [2], and Cui et al [3] have conducted research into the design and implementation of chatbots, and the development of customer service chatbots for E-commerce. Wei et al [4] collected 88 million lines of subtitles from movies and TV programmes to develop bots with emotions, and Upadhyay [5] used the Telegram Bot API to create a chatbot platform as a command and control channel.

Chapter 3

Problem Statement and Objectives

3.1 Problem Statement

The Pepper Cloud CRM is a click-based software application. Due to this, certain non-trivial functions require multiple actions. For the example '*show all high priority tasks*' mentioned in Section 2.1.2, a user would first have to navigate to the 'Tasks' module. They would then need to select and apply a filter to obtain the required result.

In a similar manner, more complicated operations can drastically increase the number of actions needed, which will in turn inflate the amount of time needed to retrieve data from the system. These actions need to be reduced so that the CRM can be used more efficiently and productively by a user. It will also make the Pepper Cloud CRM experience more user-friendly.

The solution to this problem is to build and integrate a smart chatbot with the CRM. It will result in a downturn in the number of actions required to obtain specific information. The example described in the paragraph above will be reduced to the user asking the chatbot '*show me all high priority tasks*'. After understanding the user's intention, the chatbot will be able to filter data and display the required information.

3.2 Objectives

The objectives of building the chatbot are multifarious:

1. To reduce the number of potentially mundane actions required by a user to obtain information from the CRM.
2. To make the Pepper Cloud CRM more user-friendly.
3. To create an application that can filter copious amounts of structured data and present the result to a user.
4. To develop a system that can analyse and summarise a substantial amount of data for a user.
5. To develop a system that presents summarised data in an easily legible manner, such as through a table or a graph.

6. To carry out these three major functions (display, filter, and analysis) by understanding a user's natural language text input.

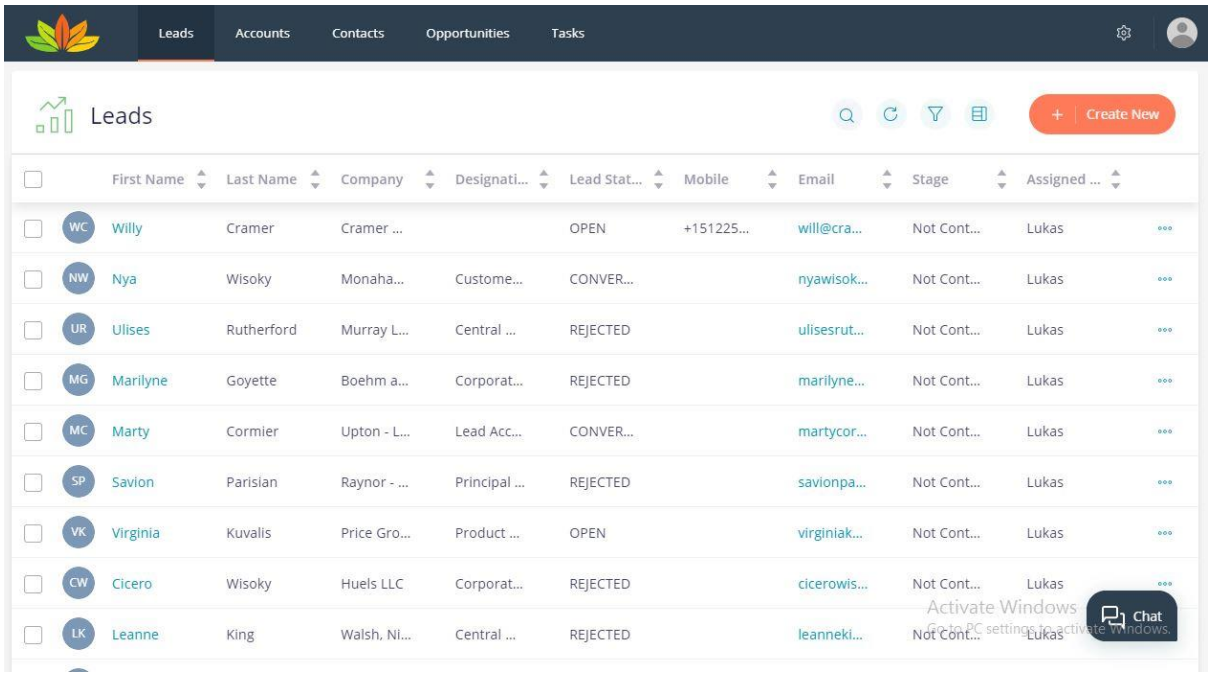

Figure 3.1: The Pepper Cloud CRM with the chatbot in the bottom right

Chapter 4

Methodology

Like any software application, the development of the chatbot for the CRM follows the software development life cycle (SDLC). The SDLC is divided into the following stages [6]:

1. Requirement analysis
2. Design
3. Development
4. Testing
5. Deployment

4.1 Requirement Analysis

4.1.1 The Domain of Operation and Use Cases of the Chatbot

When building a chatbot, it is important for the developer to understand the domain that it will be operating in. In the case of the Pepper Cloud system, the chatbot will be functioning in a B2B environment alongside the core CRM product.

Once the domain of operation is clear, use cases for the chatbot are listed down to give direction to the development process. These use cases are based on the objectives listed down in Section 3.2. Some of these are:

- Filter tasks based on due date, priority, source, and status.
- Show lead and/or contact details from a certain company.
- Calculate the total and average value of all opportunities, broken down by currency.
- Create a task associated with a particular contact.

4.1.2 The Technology Stack

After finalising the use cases and operational domain of the chatbot, the technology stack needs to be selected.

4.1.2.1 The Chatbot Framework

Developing a chatbot from the ground up is an expensive and time consuming process. As a result, multiple frameworks are available that provide a certain amount of functionality, and allow for development to take place on top of it.

The frameworks considered for the development of the chatbot were:

1. Botkit: A JavaScript-based open source toolkit built by Howdy [7].
2. Botpress: A JavaScript-based open source chatbot development platform [8].
3. DeepPavlov: A Python-based open source conversational AI library built on TensorFlow and Keras [9].

The choice of which framework to select depends on various factors:

1. The use case requirements of the chatbot
2. The services provided by the framework
3. The amount of customisability allowed by the framework
4. The ease of integration of features not provided by the framework, such as NLP agents and APIs for data transfer.
5. The developer community of the framework
6. Price

DeepPavlov was eliminated as an option as:

1. It doesn't allow for easy integration with an external NLP agent.
2. The developer community isn't as active as the communities for Botkit and Botpress.
3. DeepPavlov is Python-based, whereas the rest of the Pepper Cloud system is JavaScript-based. This would have created a lack in uniformity across the system.

To select a framework from the two remaining options, they were compared based on the following criteria:

Criterion	Botkit	Botpress
Language support	• Only JavaScript and Node.js	• Mainly GUI based development, with custom action code written in JavaScript/Node.js
Middleware support	• No inbuilt NLP engine • Allows integration with Dialogflow, Watson, Mongo, etc.	• Inbuilt NLP engine • Inbuilt support for integration with Mongo
Deployment	• Used mainly for Slack • Web connector available for websites	• Can be embedded onto a website
Inbuilt features of the framework	• Provides Botkit CMS • Provides a boilerplate application that can be customised as required for local development	• GUI available, but cannot access any code created using the GUI
Price	• Free	• Paid, but free version available

Table 4.1: Chatbot framework comparison

Based on these criteria, the Botkit framework was selected for chatbot development at Pepper Cloud.

4.1.2.2 The NLP Agent

One of the main objectives of building this chatbot was to develop a software application that can understand the user's natural language input. This requires for the integration of a NLP model with the chatbot.

Just as with the chatbot framework, creating, training, and testing a language model is a challenging task. To overcome this, multiple companies have developed NLP agents. NLP agents such as Dialogflow (Google), LUIS (Microsoft), Wit.ai (Facebook), and Watson (IBM) can be integrated with Botkit through various middleware plugins.

For this chatbot, Dialogflow was selected as the external NLP agent. Using two Node.js modules, *dialogflow* and *botkit-middleware-dialogflow*, the agent can be seamlessly incorporated into the bot.

4.2 Design

The design phase of software development defines how the product being developed will be structured.

The chatbot has been structured as follows:

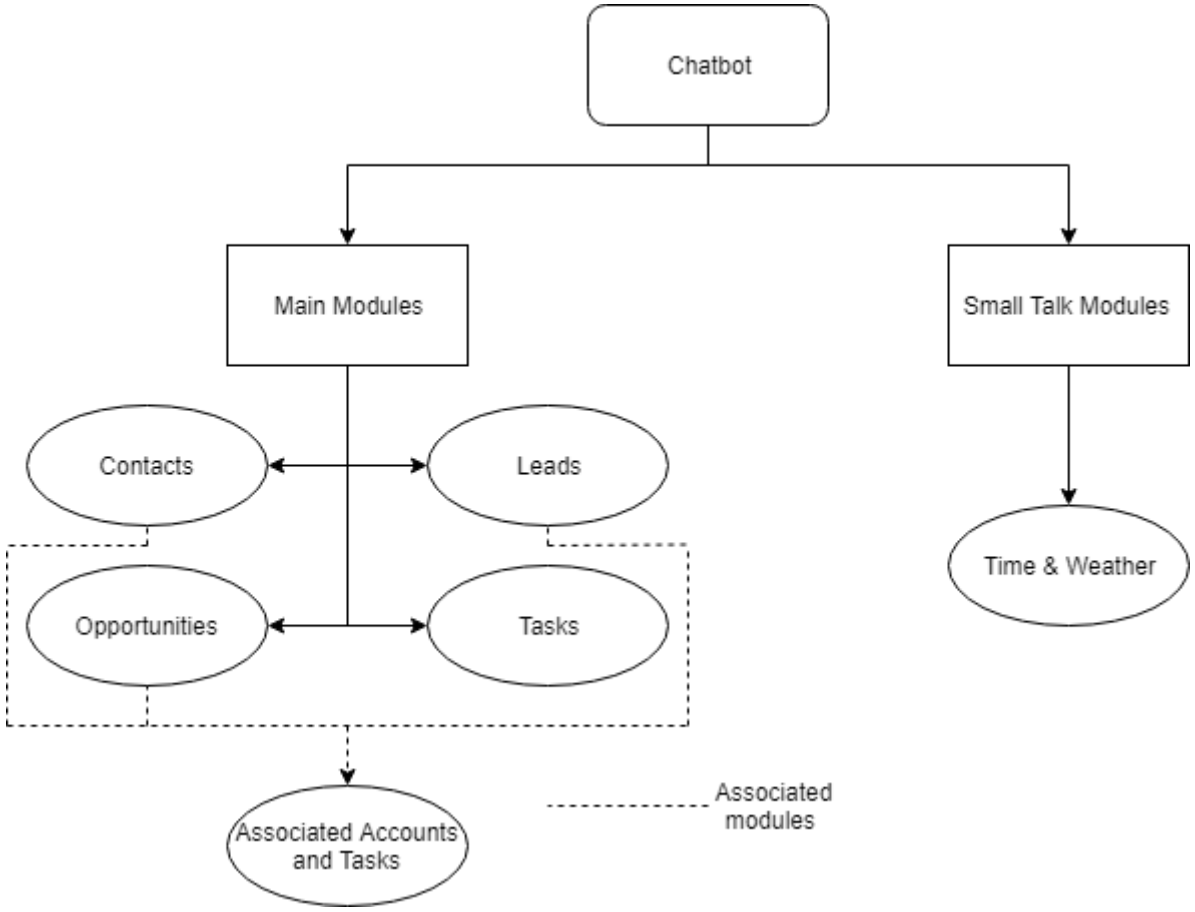

Figure 4.1: The chatbot design

4.2.1 Main Modules

The four main modules of the chatbot, Contacts, Leads, Opportunities, and Tasks, complement their respective modules in the CRM, and even add some additional functionality to them, such as data analysis through graphs.

Once a user has selected any of these modules, they can carry out three functions:

1. Display details: This option provides a listing of data available in the selected module. The user can then select an item from the listing for which they want to see more details.
2. Filter data: This option allows a user to filter data using certain attributes, for example due date, priority, and status. The attributes that a user can filter by differ from module to module, and the available filter properties are listed by the chatbot before the user enters a filter.
3. Analyse data: This option lets the user summarise and analyse the data available. The user can select the attribute by which they want to analyse the data, and the bot generates an interactive graph for the user.

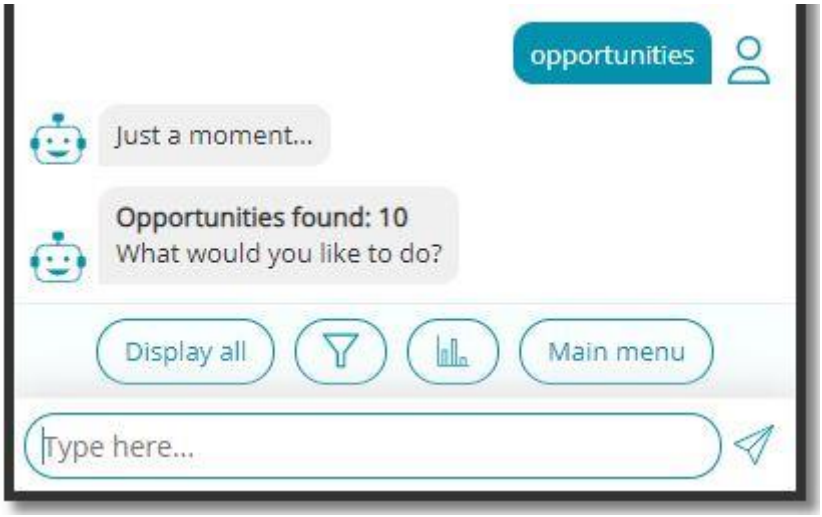

Figure 4.2: The menu presented to a user on selecting *Opportunities*

4.2.1.1 Associated Modules

As described in Section 2.1.1, Tasks are a part of the Activity Modules, and can be associated with Business Modules. The Tasks module of the chatbot provides the user with all tasks present in the CRM, irrespective of which module it is associated with. If a user needs to see the tasks associated with a certain lead, contact, or opportunity, the chatbot uses the Associated

Tasks module. This can be accessed by the user through a menu option within the listing section of a module. A user can then do the following:

1. View task details
2. Create a new task associated with that lead, contact, or opportunity
3. Filter tasks

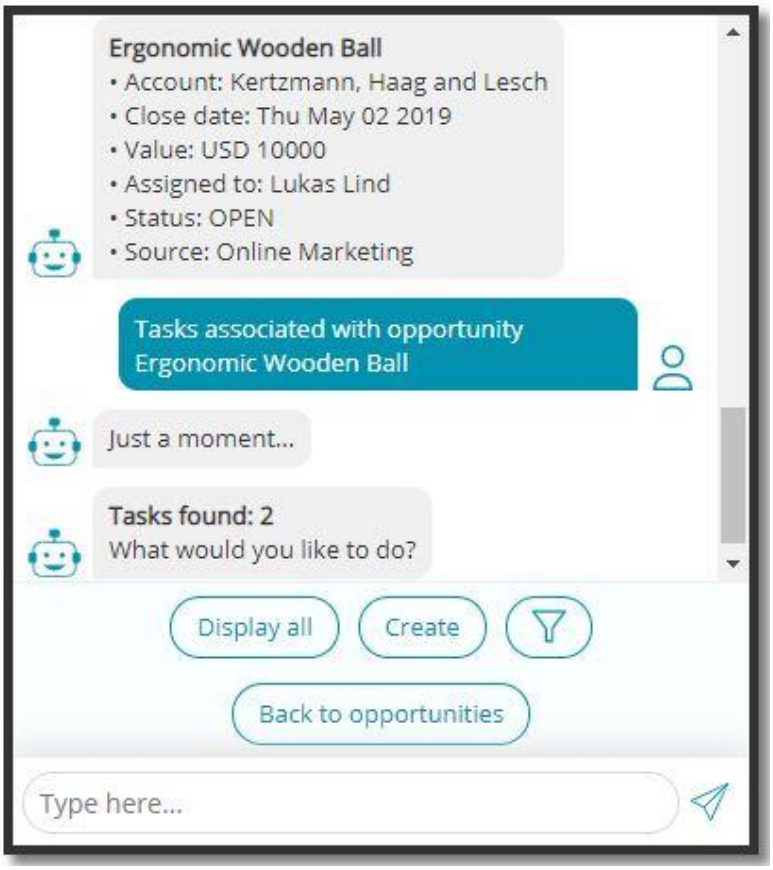

Figure 4.3: The associated tasks menu

If a user needs to see the account details of a lead, contact, or opportunity, the chatbot uses the Associated Accounts module. Similar to the previously mentioned Associated Tasks module, it can be accessed by a user through a menu option within the listing section of the module.

4.2.2 Small Talk Modules

In a chatbot, small talk modules handle non-critical functions. These functions can be thought of as add-ons for the bot – their absence doesn't affect the core functionality of the bot that is handled by the main modules.

In the chatbot built for the Pepper Cloud CRM, a small talk module exists to handle time and weather related questions. The idea behind implementing this module was as follows: If a user

is working on the CRM and they have to speak to either a work colleague or client in a different city, possibly in a different time zone, they shouldn't have to leave the CRM to obtain time and weather information for the other city, as it might affect the flow of their work. Using the chatbot, they can enter the city name, and the chatbot will provide the relevant details to the user using the OpenWeatherMap API.

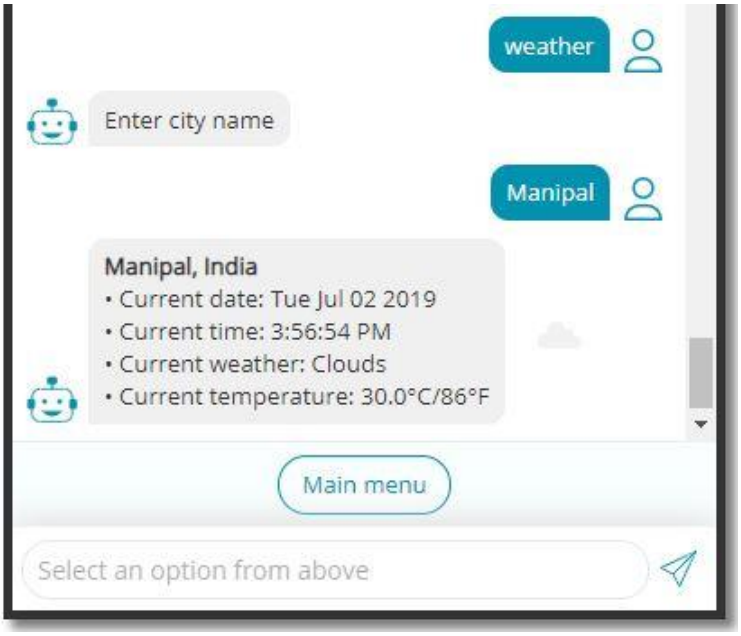

Figure 4.4: The time and weather small talk module

4.3 Development

The development of that chatbot commenced once the design was finalised. As is the case with all software development cycles, development of the bot was the longest phase of the process.

This phase involved coding the chatbot as per the specified design to implement the required features. As the chatbot relies on NLP to understand filters and opportunity analysis requests, its training too formed a part of development stage.

4.3.1 Building the Chatbot

As mentioned previously (Section 4.1.2.1), Botkit is a JavaScript-based chatbot development package. Bots created using Botkit can be deployed on a host of products, such as websites, Slack, Facebook Messenger, Hangouts, Twilio, and Cisco Jabber. Of these platforms, bots made for Slack are the most common Botkit bots.

As with any programming language and framework, using Botkit involves a learning curve. This curve can be steeper for those with no previous experience in JavaScript and Node.js. For this reason, after Botkit had been selected as the development framework, I spent two weeks creating a Slack-based chatbot. Not only did this help me learn the fundamentals of JavaScript and Node.js, but also helped the team at Pepper Cloud realise the features provided by Botkit, and the possible restrictions of the framework.

4.3.1.1 Conversation Structure

The development of web-based chatbot was carried out systematically, i.e. one module at a time. The conversational structure used in the four main modules is circular, not linear. Circular conversations allow a user to perform multiple functions in a single conversation, and are preferred for B2B chatbots. Circular conversations end only once a user leaves the webpage, for example when they logout of a CRM. In a linear conversation, such as those seen on B2C bots for ordering food or booking a hotel, the bot asks the user a series questions in a defined order, and the conversation ends as soon as the user answers the final question [10].

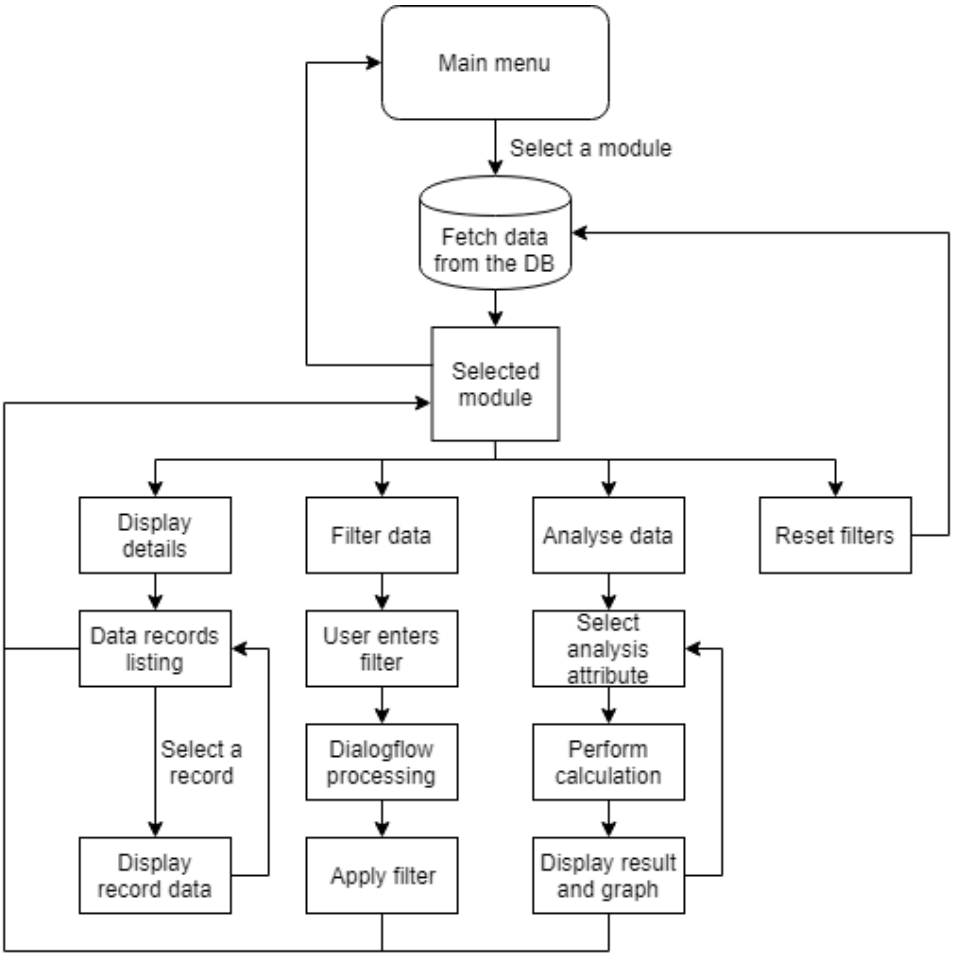

Figure 4.5: The simplified conversational structure

4.3.1.2 Implementing Permissions

Section 2.1.1.2 lists down the permissions of the CRM system. To prevent unauthorised access of data, or data leak from the chatbot, a user's permissions are checked when they try to access modules through the chatbot.

If a user doesn't have permission to read records of a particular module, the chatbot informs the user of the same, and does not make an API request for data from the backend. Other situations may arise where a user has permission for a certain module, but not for an associated module. For example, a user may have the required permissions to view contacts-related data, however may not have permission to read account-related data. In such situations, the chatbot redacts such data from an output message. This situation is demonstrated in Figure 4.6.

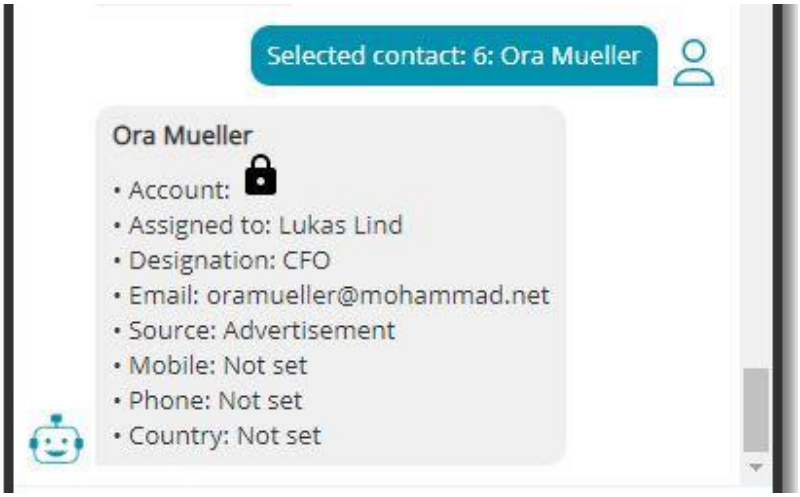

Figure 4.6: A user has permission to read contacts, but not accounts

4.3.1.3 User Interface Consistency

Standardising the UI across different software products produced by the same company is a key factor in improving a user's experience while using the said products, and retaining users. Maintaining UI consistency helps the user due to the following factors [11]:

1. Reduced learning: Once a user learns and understands the functionality of one system, they will be able to perform the same tasks on another system with relative ease if the system design is homogeneous.
2. Eliminates confusion: If two systems performing the same function use varying nomenclature or procedures to complete the task, it can create confusion amongst users. A standard UI will prevent this.

The principles of UI consistency were implemented in the chatbot through the following:

1. Using the same combination of primary and secondary colours in the interface that is used by the CRM.
2. The chatbot menu buttons are of the same shape as those in the CRM.
3. The chatbot uses the same icons (wherever icons are required) and fonts as the CRM.

4.3.1.4 Personalisation

The goal of a chatbot is to simulate a conversation between a user and an agent, and keep the user engaged. For this purpose, it was decided during the development phase to implement chatbot personalisation. The most basic form of this personalisation is to use the user's first name while welcoming them based on the specific period of the day. For example '*Good evening Lukas*' is used to welcome a user named Lukas using the chatbot during the evening (6pm onwards).

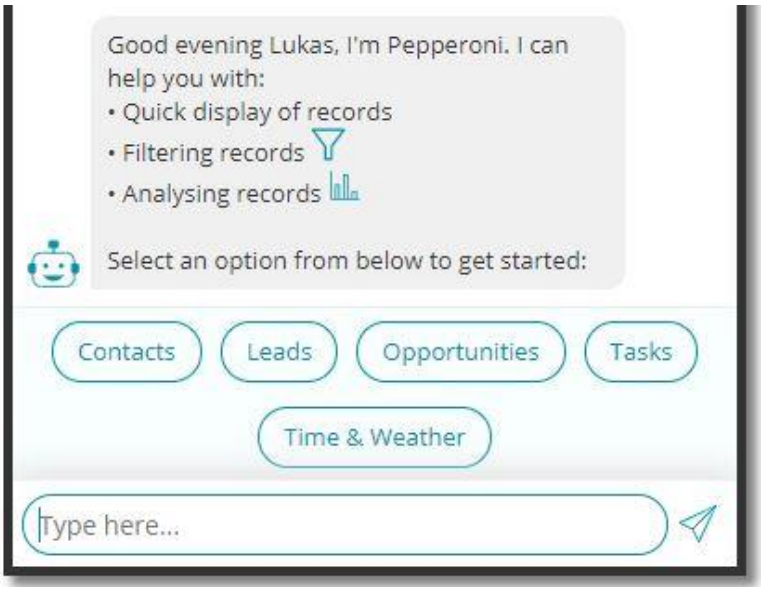

Figure 4.7: Personalised greeting message for a user

Another way this has been implemented is in the Analysis section of the main modules. While analysing data, the bot produces a graph in a popup window. However, this graph occasionally fails display correctly on Microsoft internet browsers such as Internet Explorer and Edge due to compatibility issues. Although fixes have been applied, the chatbot has been programmed to warn a user about this issue. It detects the browser being used by the user, and only notifies those using the aforementioned browsers.

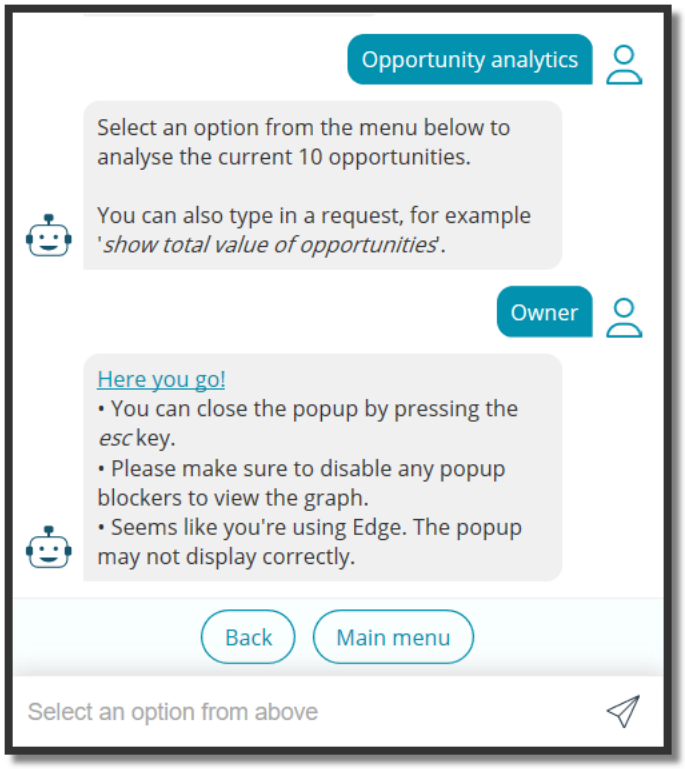

Figure 4.8: Informing the user that an error may occur

4.3.1.5 Fallback Modules

The chatbot has been designed to handle and answer a specific set of CRM business use cases. While it attempts to make clear to the user what it can and cannot do during the course of a conversation, a situation can arise, either through a lack of knowledge or misinterpretation of the part of the user, wherein the bot is presented with a request it doesn't understand.

In such a situation, a chatbot must not be unresponsive as this can lead to confusion and create doubt in the mind of a user. For this reason, these conditions are handled through a fallback module. Using such a module, the bot can respond by saying that it doesn't understand what the user wants to do, and redirects the user back to the last part of the conversation that it correctly processed. A smarter chatbot may even attempt to ask a user a question along the lines of '*did you mean perform function A*?'.

Fallback modules are hence critical to this chatbot's functionality, and must be implemented during the development of the bot.

4.3.2 Training the Dialogflow Agent

Dialogflow allows developers to create NLP models called Agents, which can then be used for intent detection and entity extraction in text and language-based software.

An Agent learns to identify intents and entities using supervised machine learning. Using developer-supplied training data of what users can say, and built-in language models, Dialogflow builds a decision making algorithm about which intent must be matched to a given user input.

Figure 4.9: Training data for the task filter with entities highlighted

During intent matching, each intent is assigned a confidence score between 0 and 1 (inclusive) by the algorithm depending on the user utterance. The intent with the highest score is returned as the matched intent.

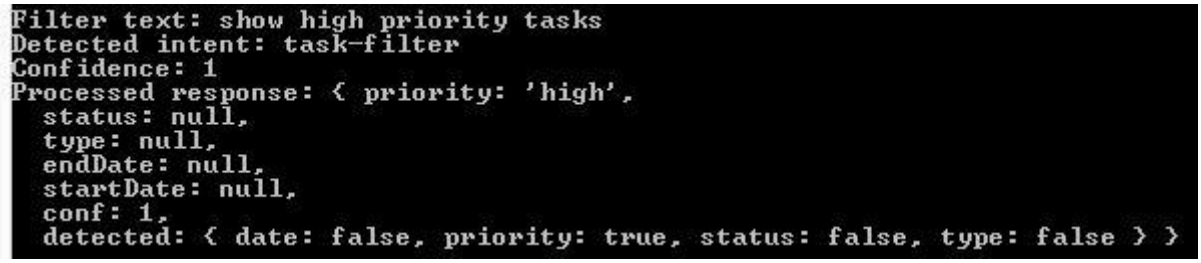

Figure 4.10: The processed response and confidence for the filter '*show high priority tasks*'

4.4 Testing

As with any software application, a chatbot must be tested to ensure that the actual results returned by the chatbot match the expected output. Testing the system helps with identifying errors in the implemented functionality, and missing requirements in the logic of the chatbot.

Although testing a chatbot may seem like a straightforward task, the nature of human conversation drastically increases the number of possible scenarios that a bot can face as each utterance can be phrased differently by users. This may lead to non-deterministic test cases – "the test passes sometimes and fails sometimes without any noticeable change in the bot code or conversational structure" [13].

The following types of tests were conducted during and after the development phase [14]:

1. Unit testing: In unit testing, the small, discrete components of the bot software were tested individually. The purpose of such tests is to certify that each unit of code is functioning as expected. Some unit tests carried out were:

Sr. No.	Test Case	Result
1	Lead listing is getting sorted alphabetically	Test passed
2	User types in a module name instead of clicking	Test passed
3	Prevent a user without contact read permission from accessing the data	Test passed

Table 4.2: Unit test cases

2. Integration testing: Integration testing takes place once multiple software units have been combined. They are then tested as a whole. The purpose of this is to expose flaws in the interaction between different units. This type of testing is also used to check if data is being sent and received correctly across APIs and other connectors. It should be carried out in an incremental manner to make it easier to fix and isolate errors. Some tests carried out were:

Sr. No.	Test Case	Result
1	Integrate display, filter, and analyse sections of a module	Test passed
2	Receive correct responses from the Dialogflow agent	Test passed
3	Obtain weather information from OpenWeatherMap	Test passed

Table 4.3: Integration test cases

3. System testing: System testing is carried out to test the completed software before its deployment. It is a form of black box testing i.e. it only tests the external working of the software and the result generated, as opposed to the internal working of the code.

4.5 Deployment

Once the chatbot is complete, it needs to be deployed on the CRM website. To do deploy the Botkit-built chatbot on the website, the Botkit Web Connector [12] was used. Using the web connector, script files were created for the bot to function on the site. DOM elements were created on the webpage to make space for the bot. These DOM elements use the script files to load the chatbot and keep it functioning.

To ensure that only users who have signed into the CRM can use the chatbot, a fetch request for a JWT bearer token (created by the CRM backend) is initiated. If the request is successful, it means that the user has signed in and can use the chatbot.

Chapter 5

Result and Conclusion

5.1 Result

The chatbot built for the Pepper Cloud CRM combines the three styles described in Section 1.3: menu-based, keyword recognising, and contextual. In certain cases, a user is restricted to a menu, while in other cases, they can type in something relevant to the context of the conversation. This helps in achieving the goals mentioned in this report: decrease effort while improving a user's productivity and save their time.

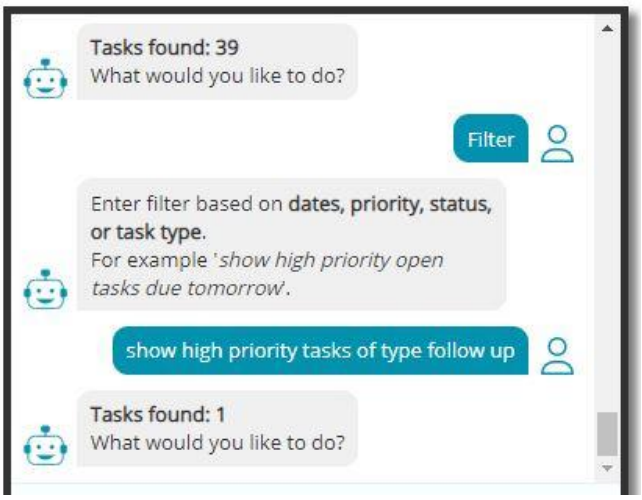

Figure 5.1a: Task filtering using the bot (one sentence)

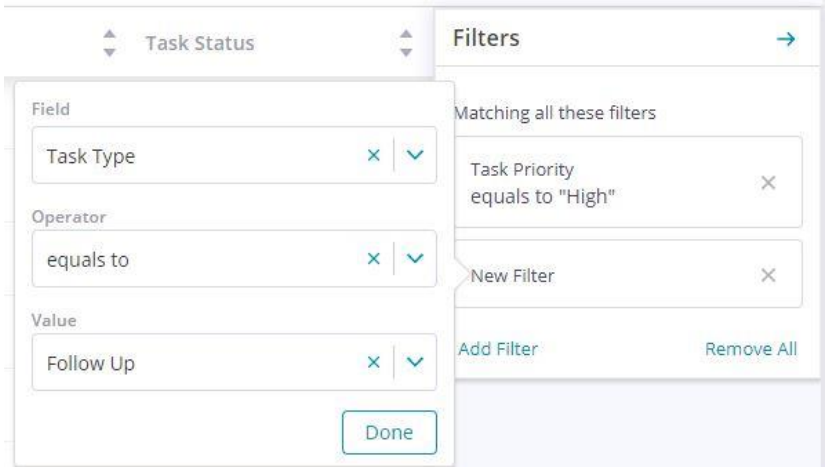

Figure 5.1b: Task filtering on the CRM (9 clicks)

5.2 Conclusion

Chatbots are a popular, intelligent assistants that have become ubiquitous on websites. Even though most of them have limited functionality, they decrease the amount of effort required on the part of user, and also save time for them. Chatbots can also act as effective marketing tools for companies as they can represent its image. A chatbot that is polished, well developed, and optimised to carry out even a small number of tasks is better than one that overwhelms and confuses a user.

Chatbots have slowly become necessary as a part of the consumer-facing system of a service business as they represent the fact that the company is using modern, cutting-edge AI-based technology. As research into AI, ML, and NLP continues, these bots will become exponentially smarter, and will be able to carry out more complicated tasks.

5.3 Future Work

Software development is a cyclic task. A development team must plan, design, and develop and application in such a way that it can be upgraded and updated in the future.

This chatbot currently uses a Dialogflow NLP agent to understand a user's textual input. Switching from this to a NLP model created at Pepper Cloud is the key milestone that must be reached. However, this will be possible only after the number of unique customers crosses a certain threshold as a large quantity of training and testing data is required to build this model. This model will be able to overcome certain issues faced in the Dialogflow Agent, such as geographic filtering. It can also increase the amount of user personalisation of the bot, and recommend better courses of action in certain conditions based on previous data.

Currently, the chatbot has functionality for a user to see associated accounts and tasks. In the future, this can be extended to associated contacts and opportunities too.

Adding voice understanding is another goal. This will result in even greater time saving as the user only needs to speak, not type in anything for the bot.

Appendix A

Additional chatbot functionality and UI figures:

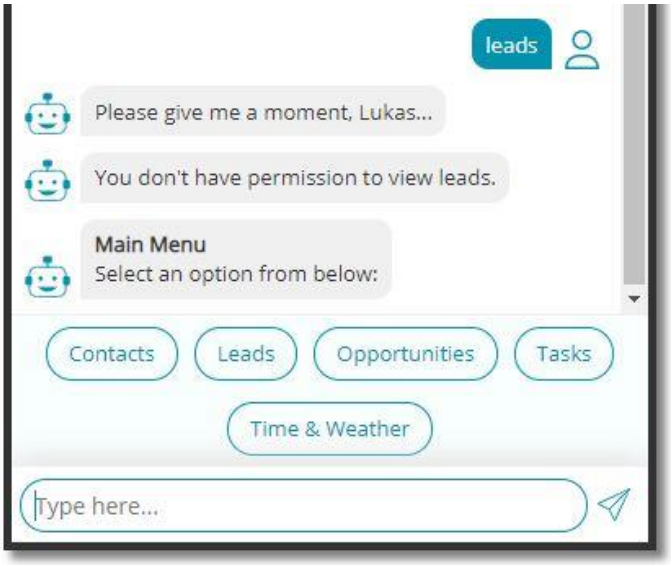

Figure A.1: Prevent a user from accessing a module without permission

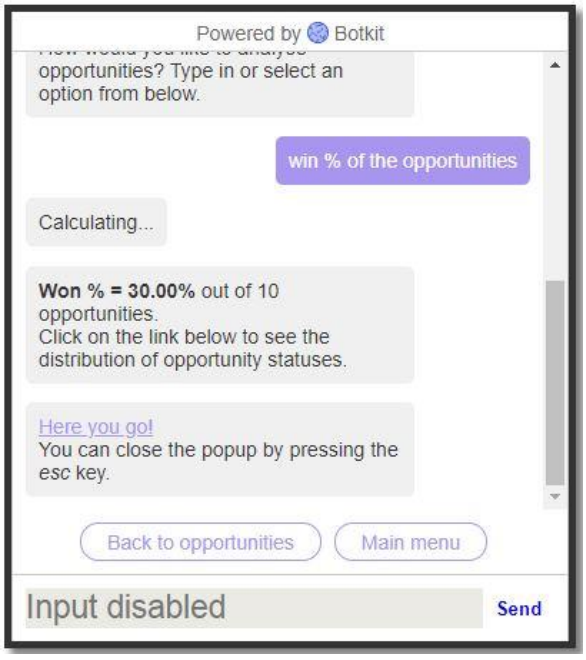

Figure A.2a: The old UI

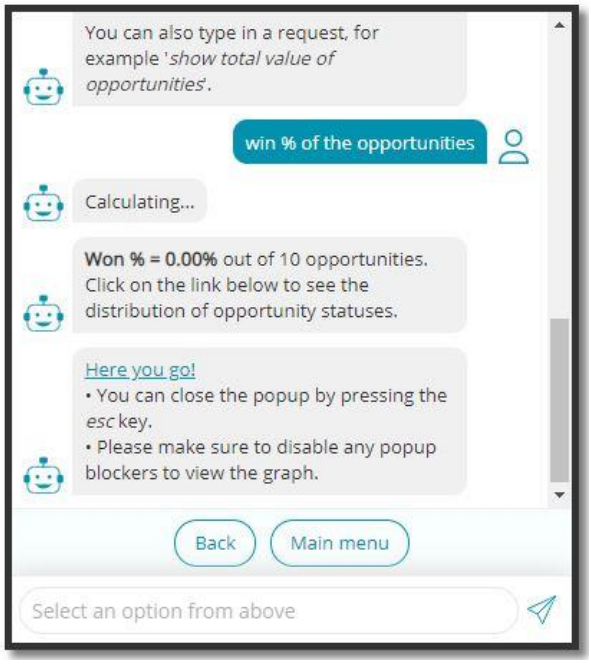

Figure A.2b: The new UI

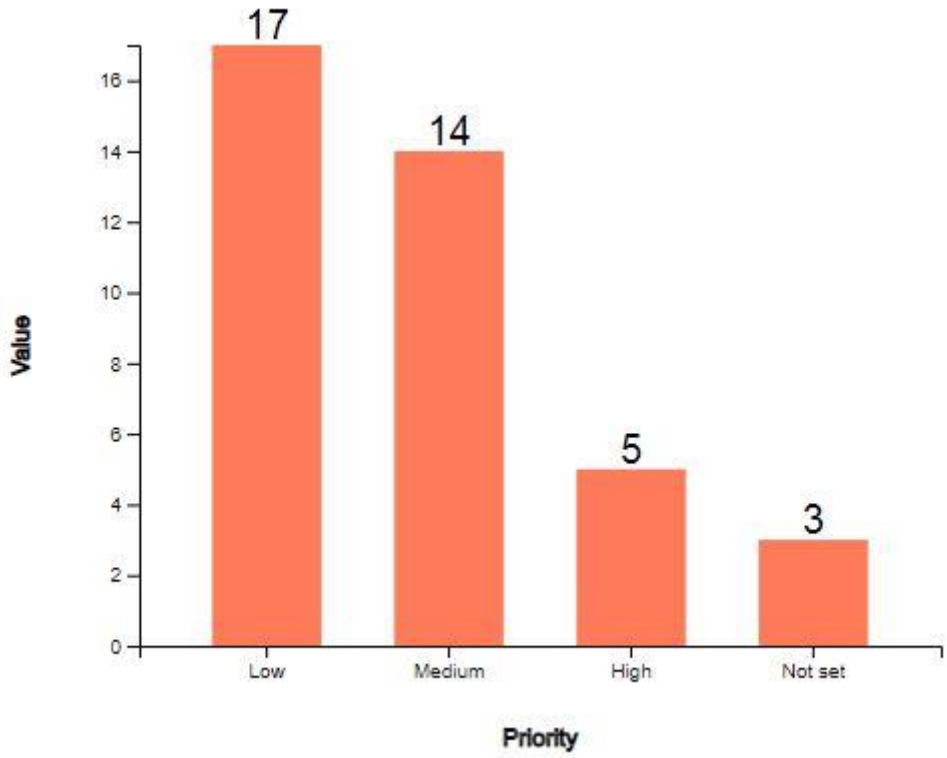

Figure A.3: The graph produced on analysing tasks by priority

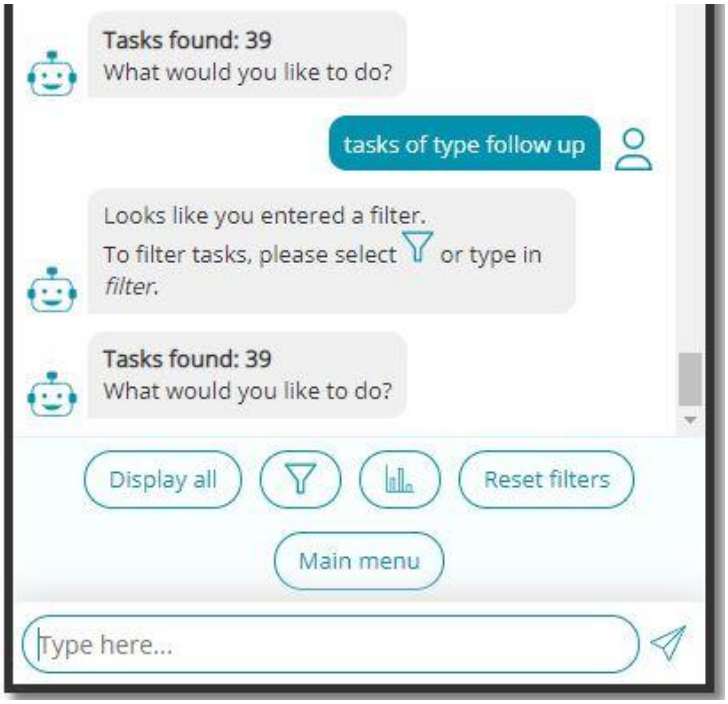

Figure A.4: A fallback module

References

[1] Nalina H J and Vinay T R, “Methodology to Make Natural Language as Computer Programming Language”, in International Journal of Advanced Networking and Applications

[2] M. Dahiya, “A Tool of Conversation: Chatbot”, International Journal of Computer Science and Engineering

[3] L. Cui, S. Huang, F. Wei, C. Tan, C. Duan, and M. Zhou, “SuperAgent: A Customer Service Chatbot for E-commerce Websites”, ACL 2017

[4] H. Wei, Y. Zhao, and J. Ke, “Building a Chatbot with Emotions”

[5] R. Upadhyay, “Chatbot Platform as a Command & Control Channel in Botnet”

[6] V. Bhatnagar, “A Comparative Study of SDLC Model”, IJAIEM

[7] “Botkit: Building Blocks for Building Bots”. [Online]. Available: https://botkit.ai/docs/v0/index.html. [Accessed Jan. 22, 2019]

[8] “Botpress – A Chatbot Maker & Bot Development Framework”. [Online]. Available: https://botpress.io. [Accessed Jan. 22, 2019]

[9] “Deep Pavlov”. [Online]. Available: https://deeppavlov.ai. [Accessed Jan 22, 2019]

[10] A. Fadhil and Gianluca Schiavo, “Designing for Health Chatbots”

[11] E Wong, “Principles of Consistency and Standards in User Interface Design”

[12] “Botkit Web Connector”. [Online]. Available: https://botkit.ai/docs/v0/readme-web.html. [Accessed April 20, 2019]

[13] M. Fowler, “Eradicating Non-Determinism in Tests”

[14] A. Jamil, and S. Awang Abubakar, “Software Testing Techniques: A Literature Review”, ICT4M 2016

Table 5.1: Project Details

Student Details			
Student Name	Samar Dikshit		
Registration Number	150953058	Section / Roll No.	CCE-B / 13
Email Address	samard05@gmail.com	Phone No. (M)	+91-9833133356
Project Details			
Project Title	Chatbot Assistant for CRM		
Project Duration	6 months	Date of Reporting	7th January 2019
Organisation Details			
Organisation Name	Pepper Cloud		
Full Postal Address	91Springboard, Salarpuria Tower – I, Koramangala 7th Block, Bangalore, India 560095		
Website Address	www.peppercloud.com		
Supervisor Details			
Name of the Guide	Darshan Santani		
Designation	Chief Technology Officer		
Full contact address with PIN code	272 River Valley Road, Singapore 238315		
Email address	dsantani@peppercloud.com	Phone No. (M)	+65-96777431
Internal Guide Details			
Faculty Name	Arjun CV		
Full Contact Address with PIN code	Department of Information and Communication Technology, Manipal Institute of Technology, Manipal, India 576104		
Email Address	arjun.cv@manipal.edu		