RULE BASED INTERACTIVE CHATBOT RESERVATION SYSTEM

Bachelor of Science (Honours) in Computing with Information Technology Management - 4th Year Project - Daniel Coxon X00131957

Technological University Dublin

Tallaght Campus

WORD COUNT:

22,221

Table of Contents

List of Figures

List of Tables

Section 1: Initial Research & Feasibility

1.0 Project Area in Detail

My 4th-Year Project idea will be focused on researching/investigating, testing, comparing and the creation of Intelligent Chat Bot Engines and AI Related Services and Platforms. I will spend time considering the best Use Cases for such applications and how they can be used to improve User Experience (UX) as opposed to being used with no substantial benefit to the end User. My initial research will bring to light the most popular platforms used to create such Bot applications. It will also highlight common areas of use, best practices for use and negative usage of Chat Bots. My aim in the preliminary stages of research is to uncover a list of companies that use Chat Bots on their websites/services and to understand how they benefit their UX and Customer Experience (CX). From this, I will gain a deeper understanding of the most common use cases of Chat Bots in today's industry and see if there is potential for other Use Cases. This will then lead on to the creation of my own Chat Bot prototypes that will be focused on a specified Use Case. I will be using more than one platform to create Bot prototypes so as that I can perform a cross-comparison of each of them. This cross-comparison will be classed under certain headings such as 'Features and Functionality', 'Ease of Use' and 'Customer/User Support'.

1.1 Areas of Interest that Require Investigation

There are multiple platforms that I will be researching and investigating as part of this project, each of which will form part of the final outcome of this project. Research that I have already completed to date has shown me that platforms such as, 'Microsoft Bot Framework', 'Amazon Lex', 'AWS Lambda', 'Microsoft Azure', 'Amazon AWS', 'LUIS' and 'QnAMaker' are among popular services used for the creation and hosting of these Bot applications. The list of popular Chatbot building platforms was compiled after careful consideration, taking into account the existence of enterprise features, the platform's enterprise clientele and testimonials, as well as the platform's status and popularity within software review sites and industry reports. (Ismail, 2017). Obviously, it will not be feasible for me to investigate all platforms available for Bot creations, due to the scope of this project, which is why I will be focusing on only a handful of platforms. These will become more defined as more research is completed into them. Another area of interest that will play a part in my project is voice activated bots such as the Amazon Alexa, Google Assistant, Siri and Bixby. These are areas that will require

investigation during the research process of my project. Aspects that I will seek information about are pricing, compatibility and the amount of accessible User support available.

1.2 Feasibility for This Project

I believe that this is a feasible project as it is a very popular field within computing. Although it is a relatively new area, I have found that in my early stages of research there is an abundance of information available out there that will help guide me with this project. “The technology is still in its early stages, but tech giants like Facebook, Google and IBM are investing huge amounts of money and resources into AI research.” (Top Universities, 2016). Also, another factor of feasibility is that there are free options out there to create Chat Bot prototypes such as Microsoft Bot Framework and a free tier AWS account.

1.3 Risks

One of the risks that needs to be taken into consideration for my project is the sourcing of information during the research phase and also the development phase. This risk comes to mind as the area of Chat Bot development is relatively new in the world of development and computing and for this reason, I am acknowledging this is a potential risk. However, I am planning to minimise this risk by exploring multiple sources of information during the research phase. Sources I will investigate will include, online, academic journals, books, videos, articles and more. When it comes to development, I am also acknowledging the sourcing of information being a potential risk. Therefore, I plan on having regular contact with my project supervisor with any questions or problems I may be facing. Although I understand project supervisors are not obliged to give solutions to problems students may be facing, I do hope that some open dialogue will help me figure out my own problems and maybe change my point of view or thought process around a certain issue.

1.4 Technical Requirements Envisaged (Platform Subscriptions)

For the development aspect of this project, there are a few platforms that I will potentially need to sign up for and create accounts to use. As part of my research so far, I have found that two of the main platforms that I will need to create accounts for are the ‘Microsoft Bot Framework’ and Amazon’s ‘AWS’. Both of these platforms are leaders in the field of Bot Creation and AI services which is why I feel that these are the best platforms to start with. From there I will see where my research takes me in terms of other platforms/services I will need to create accounts for.

1.5 Research Plan: Timing to Meet Deadlines

For this project, my plan is to break up parts of the research into a week by week bases. I will dedicate time to it daily which will help me with my weekly diary that will be uploaded each Friday. I will also be working with multiple important project related dates in mind as these deadlines must be met in order to receive grades and marks for my project.

1.5.1 Important Project Deadline Dates:

November 9th: Research Report

December 4th: Technology Assessment Report

December 15th: Detailed Project Proposal

As a starting point I have put together a rough schedule of how the coming weeks will be separated out regarding my research strategy.

1.6 Project Research Schedule:

Week Starting:	**Topic of Research**
October 15th	Microsoft Bot Framework
October 22nd	Amazon Lex (AWS)
October 29th	Other development platforms
November 5th	Uses Cases of Chatbots
November 12th	Examples of improved UX
November 19th	Ideas around other areas of use
November 26th	Development Practice
December 3rd	Development Practice
December 10th	Development Practice

Table 1: Project Research Schedule

Section 2 – Research Report

2.0 Research Report Introduction

When deciding on the area that would be worked on for the 4th-year project, many factors were considered. This included areas that were personally interesting, that were new to me and that would be enjoyable to learn about during the course of the project timeframe. This also meant that areas that were not found personally interesting were considered also. Other factors that were considered were the age of a certain industry area, areas that are relatively new and emerging in the world of computing and others that have been around for a number of years.

After some deliberation and discussions with the Project Review Panel, the decision to conduct the research and project in the emerging area of Intelligent Chatbots was finalised. The project and research will primarily focus on Chatbots used in online applications to improve Customer Service, User Experience and B2C interaction. Even though the concept of intelligent Chatbot applications like IBM's Watson, Siri and Google now, to name but a few, has been around for quite a while, the idea of implementing Chatbots as a common feature on a website has been gaining a lot more traction in the more recent years. This is a highly interesting area and it sparks an intrigue to find out more about where this area of technology currently stands and where it is heading in the future.

There are a handful of industry areas that are currently using intelligent Chatbot as part of their business which will be discussed as part of this research paper. The research also hopes to discover areas and industries that do not yet implement such applications into their model and could potentially do so to improve multiple factors such as User Experience, Customer Service, Customer Engagement and Customer Satisfaction.

2.1 Detailed Discussion of the Area

2.1.1 Chatbots: A Brief History

Chatbots and artificial intelligence have seen major growth in usage and popularity over the past few years. "Early in 2016, we saw the introduction of the first wave of artificial intelligence technology in the form of Chatbots." (Bayerque, 2016). With this surge in popularity, we have seen major social media platforms such as Facebook implementing and supporting the

implementation of these intelligent bots by third-party companies into their major social messaging app 'Messenger'. This has opened up a whole world of opportunity for these companies in terms of User Experience and Customer Service. However, this is not just a benefit for companies as users are the ones that are actually reaping the benefits of this bot implementation and interaction with greatly improved Customer Service and User Experience. This is, of course, if a company implements a Chatbot that is well thought out and designed, well developed and addresses User issues in the most efficient way possible. Otherwise, the use of a Chatbot can be cumbersome and in some cases can become a hindrance as opposed to a convenience.

This view of Chatbot popularity isn't shared by everybody, however. Some believe that Chatbots will never become popular or gain mass adoption in our modern world of technology. When González discussed his view on Chatbots in his article 'Why Chatbots Will Never Be Popular', he writes that "they will never spread, and their popularity will be fleeting" in one of his opening lines. Focusing on interaction, González says that one of the problems with Chatbots is that they will never feel one hundred percent human when interacting with them because they are generally only focused on one topic. This limits their language and can sometimes cause frustration with Users when the bot doesn't respond in the way that they had anticipated.

2.1.2 Designing a Chatbot

Bots should only ever be implemented in place of another means of communications, for example, a phone call, a website or an app, if it benefits the user and their experience when trying to solve an issue they may be facing with a particular service. This should be the mentality in which bots are designed in the early stages of their conception. This means that there are many factors that go into designing a bot. This also means that there are factors that can influence the success of a bot but also factors that do not guarantee the success of a bot.

Here are some of the factors that do not guarantee the success of a Chatbot according to Velloso, Iqbal and Standefer, 2017.

- Chatbot Intelligence
    - The smarter you make your Chatbot does not definitively mean that your bot will garner more success or be adopted by your users. “many bots have little advanced machine learning or natural language capabilities” (Velloso, Iqbal and Standefer, 2017). If it is a case that hyper-intelligence is key to solving issues that your users may be facing, then yes, the intelligence of the bot would help your users. However, mass adoption of a bot and its intelligence do not generally have a direct correlation.

- Natural Language
    - A bot could be developed with the knowledge of every language in the world, have one of the deepest vocabularies of any bot ever made and could also tell jokes. However, if the bot is once again unable to rectify the issues of your users then it may as well not exist in the first place.

- Voice Interface
    - Voice interfaces are impressive and can, in certain circumstances, be a beneficial and convenient way of communicating with a bot. However, it doesn't always mean that it is the *right* way of interacting with your bot. When designing a bot, one must consider the ways and environments in which it will be used. If the bot is going to be interacted with in a place filled with noise or a busy atmosphere, then a voice interface is *not* the correct implementation that should be developed.

Velloso, Iqbal and Standefer, 2017, also state some of the factors that companies can address that *can* influence the success of a bot:

- Does the bot easily solve the user’s problem with the minimum number of steps?
- Does the bot solve the user’s problem better/easier/faster than any of the alternative experiences?

- Does the bot run on the devices and platforms the user cares about?
- Is the bot discoverable? Do the users naturally know what to do when using it? (Velloso, Iqbal and Standefer, 2017)

Each one of these points can have an extremely positive effect on the success of a Chatbot in industry. Based on research carried out by Drift, a Conversational Marketing Platform, the main challenges the users and customers face when they are interacting with a company or brand online are "34% = Websites are difficult to navigate, 31% = Users can't find answers to simple questions and 23% = The services aren't accessible on a mobile device.

2.1.3 Existing Applications in this Domain

As part of the research for this project, uncovering the major areas in which Chatbots are being utilised in today's world plays a major role. This will aid to ground the area of research, gain a better insight into the world of Chatbots and also understand more about areas in computing that are underutilising the power and implementation of Chatbots.

Here are some examples of Chatbots that were discovered during research that are currently being used in industry for various applications. Including but not limited to, industries such as, Human Resources (HR), Digital Marketing, Fashion and Employment Recruitment.

(Human Resources) Leena AI: "Leena AI builds HR Chatbots to answer policy questions automatically. The bots can be integrated into Slack or Workplace by Facebook and they are built and trained using information in policy documents and by pulling data from various back-end systems like Oracle and SAP." (Miller, 2018)

(Digital Marketing) Drift Conversational Marketing: "Conversational marketing or conversation marketing is a one-to-one approach to marketing that companies use to shorten their sales cycle, learn about their customers, and create a more human buying experience. Instead of forcing people to go through lead capture forms and wait days for a response,

conversational marketing uses targeted messaging and intelligent Chatbots to engage with leads in real-time." (Drift, 2018)

(Fashion) H&M: "The purpose of H&M's Chatbot is to help mobile customers navigate their search through outfit possibilities and guide you to the online store areas that align with your purchase desires. H&M's Chatbot leverages the following information and responds differently based on provided information: Defines your gender and style. Suggests outfits and the total price for all items. If you dislike the suggested outfit, the Chatbot will select a different outfit. If you like the outfit, the chat provides some options: shop – direct link to the H&M internet shop; save – archive your outfit; share – via social networks, email, etc.; next outfit – provides a new outfit suggestion." (Faggella, 2018)

(Employment Recruitment) FirstJob "Mya": "Mya is an A.I. recruiting assistant that manages large candidate pools, giving FirstJob recruiters and hiring managers more time to focus on interviews and closing offers. Mya can talk to thousands of candidates at once through SMS, Facebook, Skype, email, or chat. Mya asks prescreen questions; responds to FAQs; delivers application progress updates; gives tips and guidance to candidates; alerts candidates when a position has been filled; and administers assessments and challenges" (Faggella, 2018)

Industries Using Chatbots

As Chatbots become increasingly popular, they are beginning to appear in more and more industries. They can be found in places where implementation is on a simpler scale and where the need for high intelligence is not as mission critical, all the way to more 'high risk' applications such as managing people's finances.

Here are some examples of industries that are already benefiting from the use of Chatbots today.

- News
    - News companies such as 'CNN' have launched their own Chatbot because "it's an opportunity to reach a global audience and connect in

a way that's very personal but also at scale" (Alex Wellen, CPO at CNN). There are also others such as 'theScore' which is a Chatbot aimed at people who want to know sports scores directly in their favourite messaging app.

- Health
    - Users of Health-related Chatbots are benefiting hugely in the modern world. As people get busier and busier day-to-day they can find it hard to make it to a real doctor to investigate any symptoms they may be feeling. However, with Chatbots such as 'Health Tap' Users can simply tell the bot what symptoms they are feeling and it will respond with advice from your doctor.

- Customer Service
    - This is a huge part of all companies that can really benefit from the use of a Chatbot. Common cases of use include Frequently Asked Questions (FAQ). This can replace the need for customers to ring a Customer Service helpline, often having to wait extended periods of time or in worse cases not get an answer at all. Also, replacing the need for large teams of workers to respond to the same questions time and time again.

- Real Estate
    - Although the Real Estate Industry is highly based around the human relationship between a buyer and their realtor, it too makes use of un-human Chatbots for certain scenarios. 'Holmes' was on the first Chatbots developed by 'Structurely' for the Real Estate industry. It is used to answer common questions that a buyer might have when in the process of purchasing a property.

- Entertainment
    - Chatbots don't *just* have to be used for Customer Service or solving User issues. That is why they are also very popular in the world of gaming. Games such as 'Mojihunt', found in social media messaging apps like 'Facebook Messenger', 'KIK Messenger' and 'Telegram Messenger' use Chatbot technology to keep its Users entertained for hours on end.

2.1.4 Platforms, Technologies, Libraries

Many services and offerings that are available for the purpose of developing Chatbots were discovered as part of this research paper. The research will now give an outline of some of these platforms that were discovered along with a brief description of each. This section aims to give an overview of a given platform, any pros and cons that it may possess and pricing information where applicable.

Amazon Lex: "Amazon Lex is a web service that allows customers to include conversational interfaces for voice and text in the software applications they are developing. The service, which uses the technology that powers Amazon's virtual assistant Alexa, is part of Amazon's artificial intelligence suite (Amazon AI)." (Rouse, n.d.)

Lex is a service provided by AWS for the purpose of creating and building conversational interfaces using both voice and text. This service is highly reputable and is a definite candidate for this project.

Here are some of the benefits of Amazon Lex found on the Amazon Lex Developer Guide:

- Simplicity
- Democratised deep learning technologies
- Seamless deployment and scaling
- Built-in integration with the AWS platform
- Cost-effectiveness

Amazon Lex: Pros and Cons:

According to The Web Spark, 2018.

Pros:

- SDK support – You can build iOS, Android, Java, JavaScript, Python, .Net, Ruby, PHP, Go, and C++ bots that span mobile, web, desktop, and IoT platforms.
- Natural language processing
- Speech and text support
- Utterance Monitoring
- Scalable
- AWS lambda integration
- SaaS Connectors
- Mobile integration

Cons:

- Complex web integration
- Fewer deployment channels
- Lex is not multilingual, supports only English
- Preparation of data set is complicated, the utterances and entities mapping is somewhat critical.

(The Web Spark, 2018)

Amazon Lex: Pricing

According to the Amazon Lex website, pricing for the use of this platform can vary. They state that "you only pay for what you use". Pricing is based on the amount of either text requests or voice requests that your bot processes. For example, one text request costs $.00075 and one voice request costs $0.004. This is a large factor in determining whether or not I will use the platform and is something I will need to consider further when it comes to developing my project.

TensorFlow: “TensorFlow is an open-source machine learning library developed by the Google Brain team and released in November 2015. It relies on the construction of dataflow graphs with nodes that represent mathematical operations (ops) and edges that represent tensors (multidimensional arrays represented internally as numpy ndarrays).” (Gelston, 2016).

TensorFlow: Pros and Cons:

The following is a list of positive things and negatives things about TensorFlow according to the team at DataFlair, 2018.

Pros:

- Graphs
- Library Management
- Debugging
- Scalability
- Pipelining

Cons:

- No support for Windows
- Benchmark Tests
- Computation Speed
- No GPU support other than Nvidia and only language support is Python
- Missing Symbolic Loops

(DataFlair Team, 2018)

Microsoft Bot Framework: “The Microsoft Bot Framework provides just what you need to build and connect intelligent bots that interact naturally wherever your users are talking, from text/SMS to Skype, Slack, Office 365 mail and other popular services. The Microsoft Bot Framework enables organizations to build intelligent agents, known as Bots.” (Stott, 2016).

Microsoft Bot Framework: Pros and Cons:

According to The Web Spark, 2018.

Pros:

- Structured development environment
- Bot Builder SDK in multiple languages
- Bot connector
- Bot Service for registration of bots
- Web integration is easy
- Multiple channels like Skype, Facebook, Kik, webchat, Directline etc
- REST API support, deploy the bot as REST service
- Speech and text support
- Multilingual
- Works with multiple Cognitive services like face recognition, text analysis, spell check APIs
- Scalable
- Intents and Utterance creation is done by LUIS API
- Pre-built Entities
- Technical Support
- Documentation is very good

Cons:

- The only drawback it has is that SDK support is for only two languages (Node.js and C#)

(The Web Spark, 2018)

Microsoft Bot Framework: Pricing:

Pricing for the Azure Bot Service is separated into the following two offerings:

- Standard Channels: Free unlimited messages
- Premium Channels: 10,000 messages/month

Language Understanding Intelligent Service (LUIS): "Language Understanding (LUIS) is a cloud-based API service that applies custom machine-learning intelligence to a user's conversational, natural language text to predict overall meaning, and pull out relevant, detailed information. A client application for LUIS is any conversational application that communicates with a user in natural language to complete a task. Examples of client applications include social media apps, Chatbots, and speech-enabled desktop applications." (Berry and Berry, 2018)

LUIS: Pricing: The pricing for the use of LUIS is based on Transactions Per Second (TPS) and is broken down on their website as follows:

- **Free:** For the Free instance of LUIS, users can avail of 5 TPS. This includes both Text Requests and Speech Requests.
- **Standard (paid):** For the Standard (paid) instance of LUIS, users can avail of 50 TPS. This includes both Text Requests and Speech Requests. Text Requests cost $1.50 per 1,000 transactions. Speech Requests cost $5.50 per 1,000 transactions.
-

Dialogflow: "Give users new ways to interact with your product by building engaging voice and text-based conversational interfaces, such as voice apps and Chatbots, powered by AI. Connect with users on your website, mobile app, the Google Assistant, Amazon Alexa, Facebook Messenger, and other popular platforms and devices.**"** (DialogFlow.com, 2018)

"Dialogflow is an end-to-end, build-once deploy-everywhere development suite for creating conversational interfaces for websites, mobile applications, popular messaging platforms, and IoT devices. You can use it to build interfaces (such as Chatbots and conversational IVR) that enable natural and rich interactions between your users and your business. Dialogflow Enterprise Edition users have access to Google Cloud Support and a service level agreement (SLA) for production deployments." (Google Cloud, 2018)

Rasa: "Open source machine learning tools for developers and product teams to expand Chatbots and assistants beyond answering simple questions.**"** (Rasa.com, 2018)

Dialogflow Vs RASA: According to Mamgain, 2018.

Dialogflow:

- No installation, get started immediately
- Easy to use, non-techies can also build bots
- Closed system
- Web-based interface for building bots
- Data is hosted on the cloud
- Can't be hosted on your servers or on-premise
- Out of box integration with Google Assistant, Skype, Slack, Fb messenger, etc

Rasa:

- Requires installation of multiple components
- Requires tech knowledge
- Open-source, code available in Github
- No interface provided, write JSON or markdown files
- No hosting provided (at least in the free version) Host it on your server
- No out of box integration

(Mamgain, 2018)

QnAMaker: "QnA Maker is a cloud-based API service that creates a conversational, question and answer layer over your data. QnA Maker enables you to create a knowledge-base (KB) from your semi-structured content such as Frequently Asked Question (FAQ) URLs, product manuals, support documents and custom questions and answers. The QnA Maker service answers your users' natural language questions by matching it with the best possible answer from the QnAs in your Knowledge base." (Berry and Azure, 2018).

Pandorabots:

"Pandorabots is a free open-source-based website enabling you to develop and publish Chatbots on the web.**"**

Features of Pandorabots:

- Support for Mobile and Other Devices: Deployment on Android and IOS; Speech recognition and generation services, and other services
- Extensive Documentation and Training Classes: available both at the website and in the community
- Free Web Publishing and Hosting: publish your Pandorabots anywhere you want on the web, including mobile phones
- Web-based Development Tools: Develop and edit your Pandorabots online
- Conversational Logs: Use your conversational Logs to improve your Pandorabot's interactional skills
- AIML Development and Targeting Tools: Industry standard development and maintenance tools
- Large Open-Source Knowledge Sets: Essentially pre-built Pandorabot 'brains' you may modify to easily produce high-quality Pandorabots
- Large Community: Draw upon our large community of users for consulting and help in a variety of locales and languages
- A Variety Of Avatars: Made available from vendors and our user community
- Programmable Output formats: HTML, Flash - Produce Pandorabots using sound, video, music, etc.
- Connections to external knowledge sources: API access to semantic databases and other web-based knowledge sources

(Pandorabots, 2018)

Pandorabots: Pricing:

Free: Up to 1,000 messages/month (premium rates thereafter), unlimited Sandbox Usage, Up to 2 bots, Community Support

Premium: $0.0025 per message Up to 100,000 messages/month, unlimited Sandbox Usage, Up to 10 bots, Email Support

Amazon Lex vs LUIS

According to Wolhuter, 2017, and their experience using both of these services in business-related applications and services, there are a few major differences between these two services. The main differences that were spoken about were the use case for each service and also the complexity level of each service. On one hand, there is Amazon Lex, "a serverless solution without an individual backend that stores everything on AWS cloud and is available in the most commonly used languages" (Wolhuter, 2017), and on the other, there is LUIS "Luis is a more flexible solution for more complicated Chatbots and for group chats, however, the truth is that Luis is not simple to implement.".

The main conclusions drawn by Wolhuter, 2017, and their experience in using both services is that Amazon Lex would mainly be used for simpler, but still complex, Chatbot solutions and possesses an element of 'ease of use' enabling more staff members in an organisation to potentially get more involved with development, "Lex is perfect for clear, easy flow conversational assistance for booking, searching and to construct a personal assistant that can help with daily routines by passing simple and strict commands." (Wolhuter, 2017). Whereas, LUIS has a much steeper learning curve and is much less accessible in terms of use for people who have less knowledge of the development of intelligent Chatbot systems, "Luis has a more sophisticated learning system and there is more power in luis.ai console than Lex. With Lex you can easily get a content manager or another stakeholder to help you define entities and/or types and fill in user utterances. It is not that simple with Luis." (Wolhuter, 2017).

2.1.5 Choosing a Platform(s)

For this project, the choice of development needs to be narrowed down. For this reason, I will be focusing on some key metrics including Cost, Website Integration and Information Resources. Firstly, during the time spent conducting this research it has become clear that there are three platforms that are the most popular and the most reputable. These three development platforms are the big-name ones, Microsoft Bot Framework, Dialogflow (Google) and Amazon Lex. One reason I'm narrowing down to these three platforms is the fact that because they are such popular platforms, there are extensive resources available in order to learn more and more about them. This will in turn potentially lead to a higher quality development further into this

project and will allow me to absorb as much knowledge as possible about a platform. This is also a key factor to me as the area is completely new to me and the more resources the better.

Platform	Cost	Web Integration
Microsoft Bot Framework	Free	Yes
Dialogflow	Free	Yes
Amazon Lex	Free Tier	Yes

Table 2: Chatbot Platform Comparison – MBF, Dialogflow & Amazon Lex

Web Integration

Microsoft Bot Framework:

Web integration is offered by the Microsoft Bot Framework in two different ways, Skype Web Control and Open Source Web Control.

Skype Web Control:

This is a Skype client in a web-enabled control. This skype web control acts as a front-end and doesn't allow the developer to write any custom code.

Open Source Web Control:

This is based on ReactJS and uses an API called Direct Line API to communicate with the Microsoft Bot Framework. This integration is what would be used for the purpose of this project as it is more development focused and customisable.

Dialogflow:

Kommunicate is a plugin which allows the integration of a Dialogflow Chatbot to a website. This can be achieved through the creation of a free Kommunicate account, connected your Kommunicate account to your Dialogflow and copying some JavaScript code that is generated.

Amazon Lex:

Amazon offers a Chatbot website integration option called 'The Chatbot UI'. This offers multiple features and options when it comes to development such as using The Chatbot UI as a full-page Chatbot UI or embedding it into a website as a Chatbot widget.

2.2 Overview of System Architecture (Based on Research Findings)

2.2.1 How A Chatbot Works

In this section I will detail the technologies that are used to build the core functionality of Chatbots. According to Elupula, 2018, "a bot is a computer program that is designed to communicate with human users through the internet." (Elupula, 2018). However, understanding how this is actually achieved is key to this project. First of all, Chatbots can come in multiple forms. These forms include, Rule Based Bots, also referred to as a Retrieval Based Model, and Artificial Intelligence (AI) Bots, also referred to as a Generative Model. In this section I will give an overview on each.

2.2.2 Rule Based Bots (Retrieval Based Model)

At the core of a rule based Chatbot is a somewhat pre-defined user interaction flow that recognises keywords to help send the user down the correct path. This is achieved by the Chatbot being trained on a set of questions and the possible outcomes that match those questions. This would be a highly popular type of Chatbot for customer service related Chatbots as it is based around a similar set of questions that customers would have relating to a specific topic.

Shridhar states that in a rule-based approach "a bot answers questions based on some rules on which it is trained". One down side to this is the efficiency in which the bot answers questions. If the pattern does not match the rules upon which it was originally trained it will be less accurate. When it comes to writing rules for a rule-based Chatbot, Artificial Intelligence Markup Language (AIML) can be used. For the purpose of this project, AIML is a language that I will have to become more familiar with. A benefit of a rule-based Chatbot is the fact that it does not rely on Machine Learning. This means that "a rule-based Chatbot has a faster time-to-relevance, delivering a faster impact on customer service." (Selby, 2018).

Another term given for a Rule Based Chatbot is a Flow-based Bot. “A Flow-based bot focuses on solving user queries in a way that is moves them gradually towards a solution. The flow is clearly defined keeping in mind the possible user concerns” (Newgenapps.com, 2018). Research has uncovered that Rule-based Chatbots are the most popular and commonly used Chatbots in the world, “[Rule-based Chatbots] are the most popular type of Chatbot on the market today” (Ward, 2017). Rule-based Chatbots are cheaper and faster to implement than Artificial Intelligence Bots but unfortunately, they are unable to comprehend intent or context. This means they must be programmed in a very detailed way to take into account the different ways in which questions can be asked.

2.2.3 Artificial Intelligence Bots (Generative Model)

In contrast to a Rule Based Chatbot, Generative Model based Chatbots make more use of Machine Learning. This gives them capabilities such as being able to generate different answers even if asked the same question. Taking advantage of Machine Learning, this also means that the Chatbot is able learn from all the interactions it has with its users. To achieve this, the Chatbot would be highly connected to external sources such as analytic platforms and API’s. Another name given to Artificial Intelligence Bots is an Intent Bot. “An intent-based communication means that the bot will understand the user queries by combining two inputs - state and context. State refers to taking account of the chat history while context refers to analysing inputs from external data points.” (Newgenapps.com, 2018).

2.2.4 Chatbots Tasks

Chatbots face the task of understanding what a user says to it, in a text or voice format. Natural Language Processing (NLP) is a process that can be used by Chatbots to perform pattern matching of user input and understand/classify a user’s intent. “Natural Language Processing (NLP) is a sub-field of Artificial Intelligence that is focused on enabling computers to understand and process human languages, to get computers closer to a human-level understanding of language.” (Seif, 2018).

Natural Language Processing is an essential aspect of a Chatbot’s functionality. Without it, the Chatbot would be useless. It is the piece of functionality that allow the Chatbot to understand

a message give back an appropriate response. A potential issue with a text based Chatbot is input errors by the user like typos and spelling mistakes. According to Phillips, "Advanced Natural Language Processing (NLP) capabilities can identify spelling and grammatical errors and allow the Chatbot to interpret your intended message despite the mistakes." Spelling corrections can be achieved through NLP by connecting an API such as 'textblob' to the Chatbot. "TextBlob is a Python (2 and 3) library for processing textual data. It provides a simple API for diving into common natural language processing (NLP) tasks such as part-of-speech tagging, noun phrase extraction, sentiment analysis, classification, translation, and more." (Textblob.readthedocs.io, 2018)

Based on the findings of the research carried out for this project, I now have a clearer path as to the system that will be developed in further stages. The project will be focusing on the development of a Rule-Based Chatbot for use in place of an FAQ page on a website. This means that the main focus of the Chatbot will be to aid a customer reach and end target through the use of a predefined path.

Using the platforms that were discovered in research, I will develop a Chatbot(s) that can potentially be used by a company to replace a Frequently Asked Questions (FAQ) page on their website. By using multiple platforms to create these Chatbots, this will enable me to carry out a detailed comparison between various attributes of each platform, including but not limited to:

- Performance
- User Experience
- Cost
- Resources Available

The final piece to this project will be a form of data gathering, potentially through the use of a combination of User interviews and surveys. This will be for the purpose of measuring the impact that the Chatbot system has on User Experience. The aim would be to see if implementing a Chatbot system over a regular static page of questions and answers has a positive or negative impact on a User's experience.

2.3 Risks

As part of this project there may be some risks associated with it. In this section, some of these risks will be acknowledged and outlined.

2.3.1 Security

There are multiple security risks that could potentially be involved with this project. Chatbots can face some security vulnerabilities in the real world and can seem attractive to hackers. This is a factor that I must consider when it comes to the development of my project Chatbot(s). I will now outline some of these security risks that were discovered as part of the research for this project.

A fundamental building block upon which Chatbots are generally developed is the collection and storage of User data. This needs to be acknowledged as a security risk and treated as such during any development that will be taking place. Any data that is stored by the bot runs the risk of being compromised, but this is dependent on the scale of the project. At this stage of the project it is unclear whether any highly sensitive or confidential User data will be stored or used by my system, but it still needs to be documented as a potential risk.

As part of this project I will be using one of the services mentioned in the previous sections of this report. Deploying any services on platforms such as AWS or Azure can come with its own security concerns, some of which will be highlighted in this section.

AWS:

Based on previous research carried out relating to this project and the security concerns of deploying services on AWS, the following areas of concern were identified: According to Cheslock, 2018:

Top 7 AWS Security Issues:

1. Prioritising a Security Strategy Ahead of Controls and Tools
2. Overcoming the Lack of Security Visibility in the Cloud

3. Improving Confidence in Cloud Provider Security
4. Defining Who is Liable
5. Understanding Why Attackers are Attracted to the Cloud
6. Defending Against Curious Onlookers in Multi-Tenant Infrastructures
7. Addressing Compliance Regulations from the Get-Go

Azure:

Based on the research carried out for this project, Microsoft and the services they provide are a likely contender for the purpose of future development. As a result of this I feel it would be wise to acknowledge some of the security features Microsoft provide for the cloud infrastructure 'Azure' which is highly related to the Bot Framework. Based on previous research undertaken focusing on security features of Microsoft Azure some of the main security features were uncovered. According to Lugo, some of the main features are:

Azure Security:

1. Secure Development Lifecycle (SDL)
2. Continuous Monitoring
3. Certifications and Compliance
4. Secure Identity Management Made Easy

2.3.2 Scale

Another risk relating to this project that I will need to consider and that was mentioned in a meeting with the 4th-year project panel is the scale of the project. The project needs to be of adequate size and complexity in order to meet expectations and to meet this requirement I will need to make sure that a proper plan is in place as to what will be carried out for the duration of the project. Also, meetings with my project mentor will help to expand the project in ways that may not have been entirely obvious at first.

2.3.3 Cost

With this project comes the risk of having to pay considerable amounts of money just to carry out the project itself. These costs would come in the form of paying for the use of the platforms to develop these Chatbots. This is something that is not exactly viable as the project is purely

education based and is not funded by a third-party business or company. To combat this risk, monitoring what services are being utilised during the project development of utmost importance.

2.4 Conclusions

Based on the work carried out in this research report and the information uncovered, it is clear that there are many options available moving forward with this project. There is an abundance of platforms available to utilise for the purpose of development, each one with its own unique attributes and nuances that will make for an interesting comparison and makes for an exciting area to experiment and develop in. The fact that there are so many different platforms also means that there is plenty of information available for further research when it comes to developing future prototypes. The extent of online information is also a substantial benefit to the project, this is because there is a high volume of interest around this relatively new area of computing and lots of people and sources providing information on all aspects of it.

The research also uncovered that there are many use cases for Chatbots. They are utilised in more industries than was first obvious to me before commencing the research. They are also more popular than I had first realised before undertaking the research. The research found that the use of Chatbots can have a significantly positive effect for businesses if implemented correctly and can, in some cases, even save business money and resources that may have otherwise been spent elsewhere to achieve a lesser result.

Section 3 – Technology Assessment

3.0 Technology Assessment Introduction

In this report, I will be detailing my findings and conclusions after the testing and experimentation with the following Bot development platforms:

The Microsoft Bot Framework, Amazon Lex, and Dialogflow. From carrying out these tests and spending time working with each platform, I learned some different things that I didn't know prior to working with them. What I found from actually working with the platforms in a practical situation as opposed to reading many theoretical articles is that I gained a better understanding of how they function. Seeing the platforms in use gave me a better insight into how to go about beginning the development phase of this project.

3.1 Microsoft Bot Framework

Out of the three platforms that I tested during this technology assessment phase, I spent the most amount of time using the Microsoft Bot Framework. Of the three, it proved to be the most complex and depth to use but seemed to have the most in terms of core service offerings. Before attempting to develop or test anything I spent some time inside the environment to familiarise myself with it as I felt this would give me more of an understanding before diving into tests. I clicked through each option that was available on the left-hand side of the Azure Portal window to in an attempt to get to grips with how the system is laid out as I had never used it before. Within the Microsoft Azure portal, all menu items are placed on the left on the user dashboard is on the right side of the screen. The dashboard is fully customisable and can display as little or as much information desired by the user.

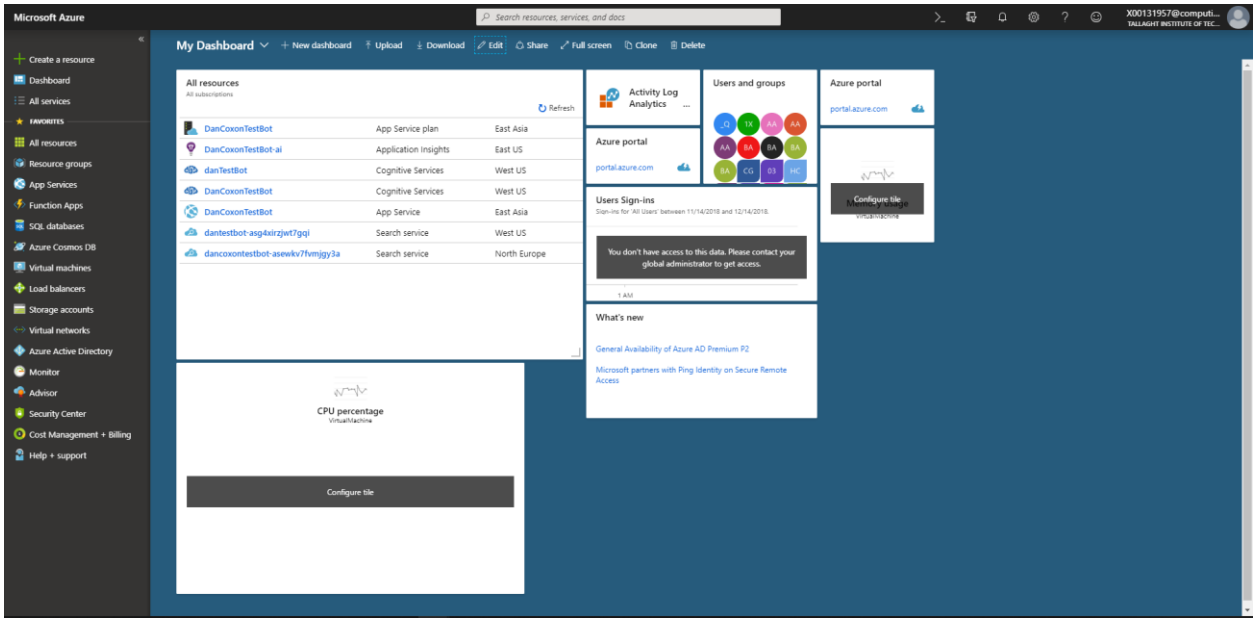


Figure 1: Azure Portal Dashboard

When commencing my testing for this platform, I began by researching some tutorials on the learning the basics and getting started with Azure Portal and The Microsoft Bot Framework. This was a good place to start as it was overwhelming, in the beginning, using a new service like this. From my research carried out earlier in the semester, I knew that I needed to begin testing the QnA Maker service. This was a required step in order to move onto the next task which was creating a Knowledge Base. When creating the QnA Service I was required to enter in some information for the service. Including the service name I wanted to allocate, what subscription I was using which in my case was the free trial, the management pricing tier, the location, the resource group, the pricing tier for searches, the location for searches, and the app name.

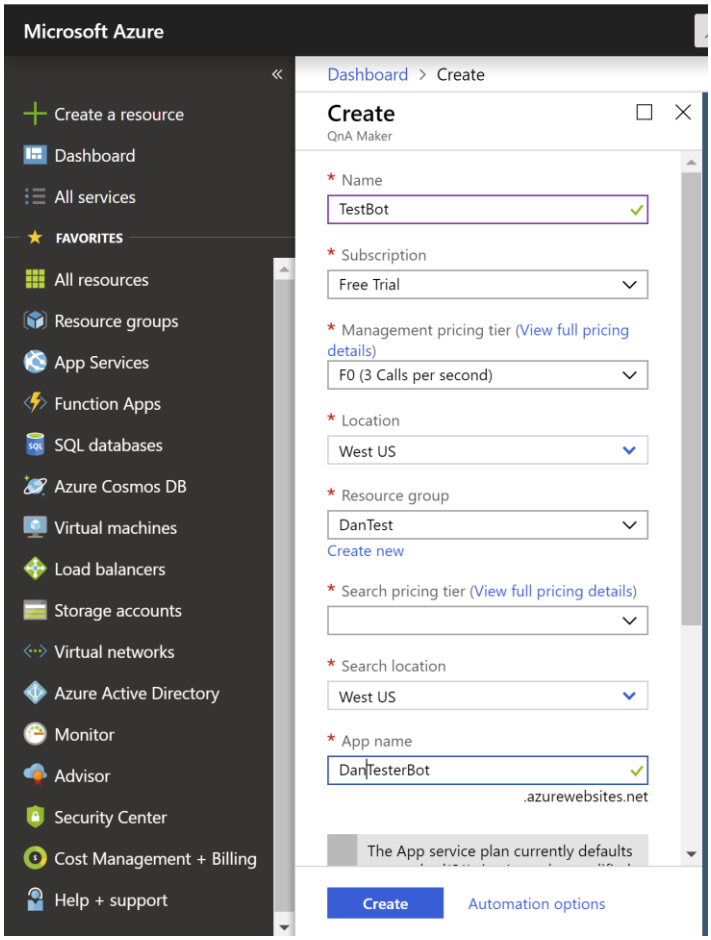


Figure 2: Figure 2: Creating QnA Service

Once I had this QnA Service up and running it was time to create the Knowledge Base. This is, in essence, a database full of FAQs that are pulled from a URL that I entered in the setup phase. This list of FAQs is then parsed and separated into their respective Questions and Answers. The FAQ page I used for the purpose of this test was from the help section of UBNT.com which can be found here: https://help.ubnt.com/hc/en-us/articles/115009192828-airMAX-General-FAQ

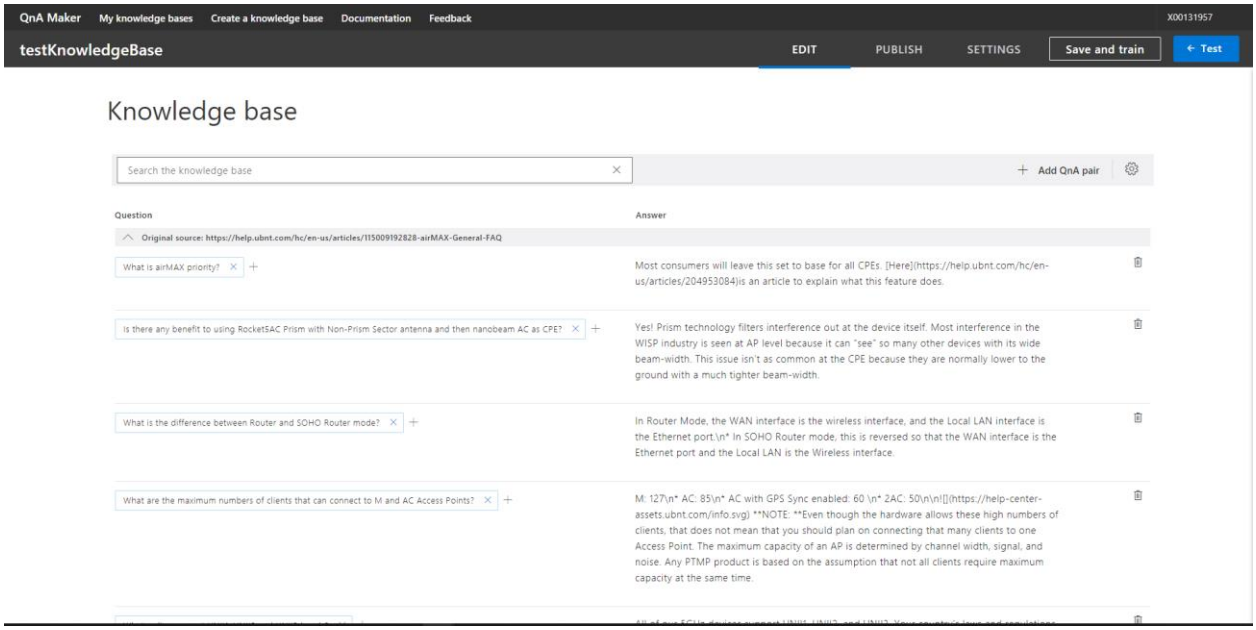


Figure 3: QnA Maker Knowledge Base

I was then able to test the functionality of this Knowledge Base using the 'Test' button in the upper right-hand corner of the interface. The key concepts of the Knowledge Base are:

- Questions - A question contains text that best represents a user query.
- Answers - An answer is the response that is returned when a user query is matched with the associated question.
- Metadata - Metadata are tags associated with a QnA pair and are represented as key-value pairs. Metadata are used to filter QnA pairs and limit the set over which query matching is performed.

(Docs.microsoft.com, 2018)

I was then able to type the questions from the FAQs that I had entered into the knowledge base, and even slight variations of them, and then get an answer in the dialogue box chat section. This was possible because of the fact that "QnA Maker has natural language processing abilities, enabling it to even provide answers to questions that are worded slightly differently than expected. However, it does not have semantic language understanding abilities. It cannot determine that a puppy is a type of dog, for example." (Docs.microsoft.com, 2018)

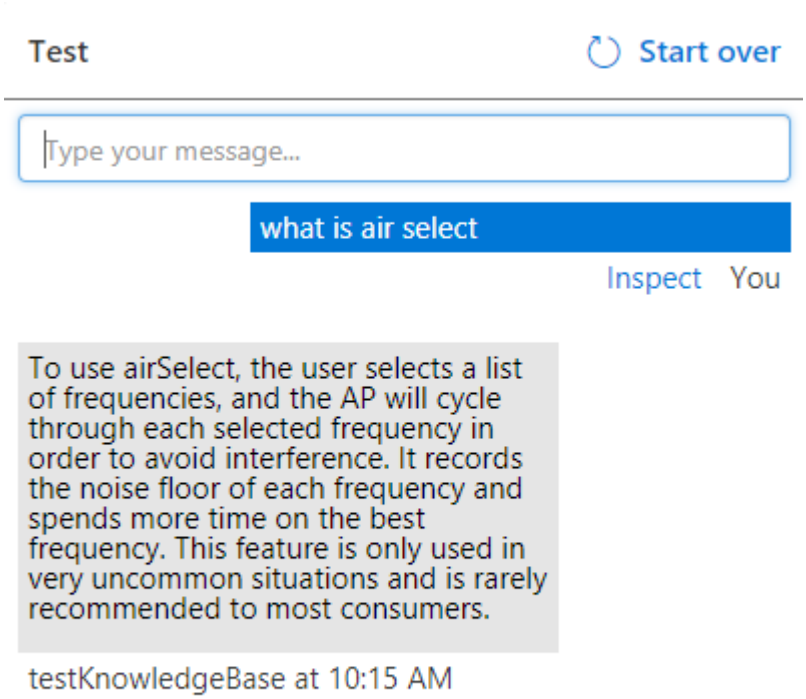


Figure 4: Testing Knowledge Base Response

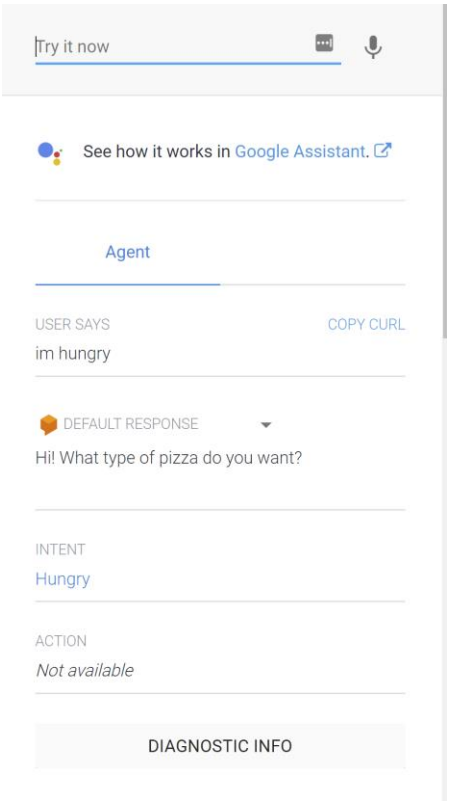
During my time testing the environment that it the Microsoft Bot Framework and the Azure Portal I faced a variety of challenges. Some helpful articles and videos that I used as a source of learning seemed to be out of date even though they were not that actually old. There seems to have been an update to the QnA service maker website so when I was attempting to carry out what I had read or watched online, the steps were not the same which led me to some confusion. The QnA Maker website's url is currently 'qnaamaker.ie' but the previous version can still be found at 'qnamaker.ai/old'. However, when using the latter URL there is a banner at the top of the page that states "This Preview portal will be deprecated on January 31st 2019. New knowledge bases should be created in the QnAMaker GA portal. Learn how to migrate existing ones here." (Qnamaker.ai, 2018). After this, I planned on creating a Web App Bot in the Create Resource - AI + Machine Learning - Web App Bot section of the Azure Portal. When attempting to use the 'QnA Bot Template' was there but it appeared to be unavailable to choose as an option. The only Bot Template options I had available to choose were an Echo Bot or a Basic Bot. When clicking the link associated with the QnA Bot Template I was directed to documentation that explained to me that I must carry out steps that I had already carried out

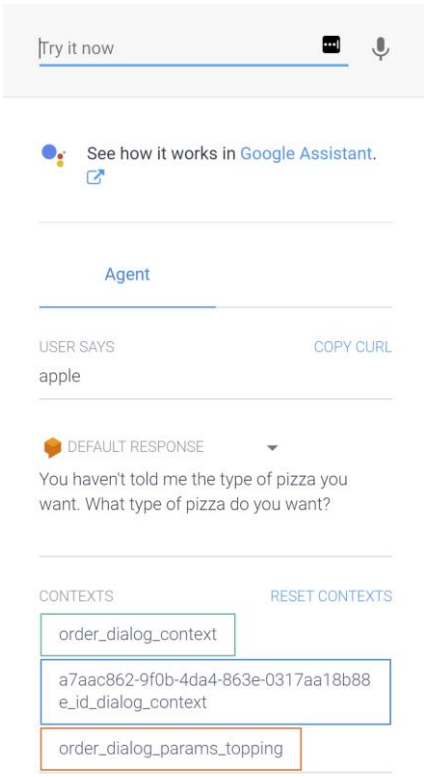
which led to a bit more confusion. When this is still an issue, I must overcome in order to move forward but I think if I carry out some more reading, testing and researching I will figure out a solution.

3.2 Dialogflow

The second Bot creation platform I tested was Dialogflow, provided by Google.

Using this platform, in contrast to Microsoft's Bot Framework, I was able to have a simple bot up and running within five minutes. Following a simple tutorial, I created a simple bot that was able to take an order for a pizza.





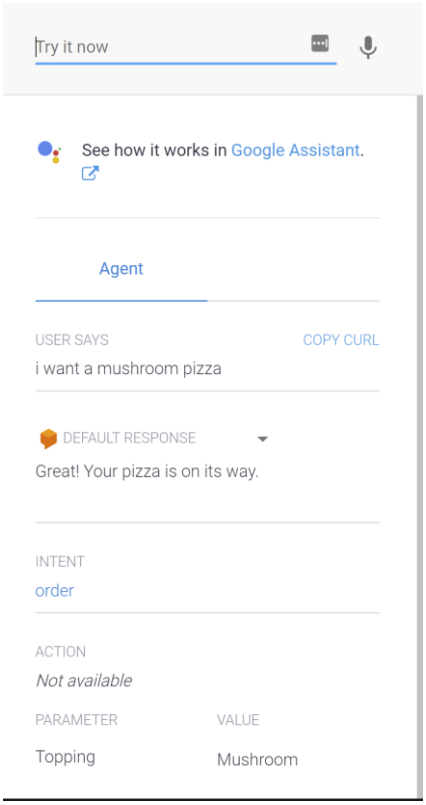


Figure 5, Figure 6 & Figure 7: Dialogflow Pizza Bot screenshots

This was achieved through creating training phrases that attempt to anticipate what the user will enter as their first opening dialogue which will then trigger the rest of the Chatbot's logic. In this case, the training phrases I used were, "feed me", "order pizza", "please order a pizza", "I want pizza", and "I'm hungry".

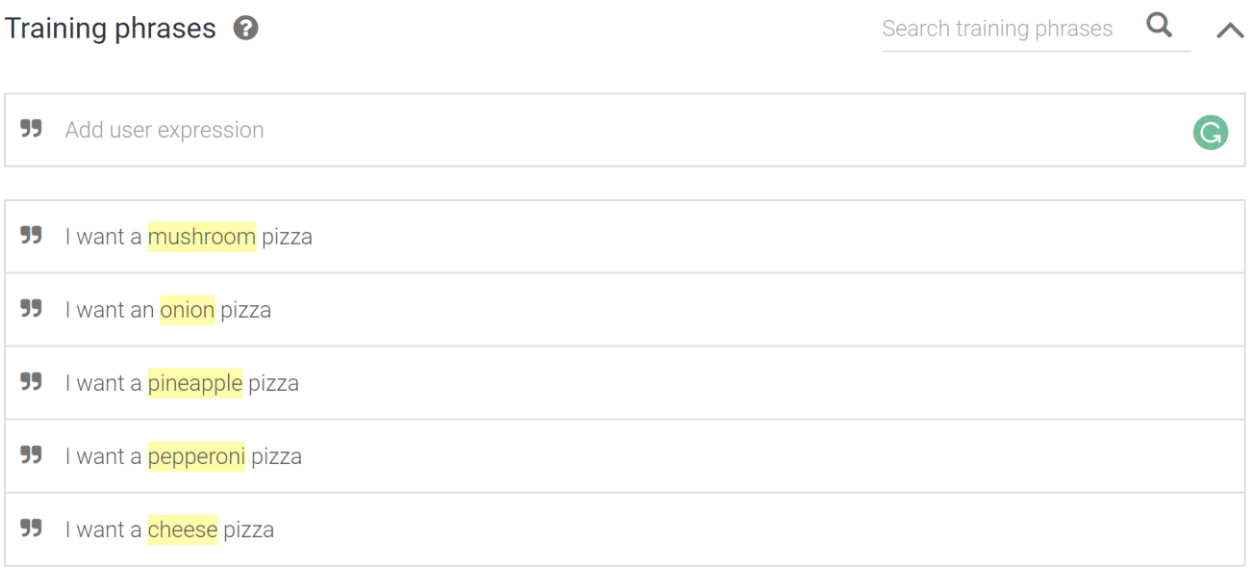


Figure 8: Dialogflow Training Phrases (Technology Assessment)

If any of these phrases were entered in the Chatbot interface then it would respond with the programmed response, “Hi! What type of pizza do you want?” which can be seen in the conversation flow screenshots above. Dialogflow also has some build in language processing as it still gave a response even if the phrase entered was slightly syntactically different, eg. “I’m hungry” vs “im hungry”.

3.3 Amazon Lex

The third Bot development I tested, however briefer than the rest, was Amazon Lex. Similar to Dialogflow, I was able to have a very simple bot up and running within five minutes and with even fewer steps than Dialogflow. Upon first setup of Amazon Lex it gives the user the option to Deploy some already pre-built bots to experiment with. Within a few clicks I was ordering flowers with a Chatbot interface in the Amazon Lex framework.

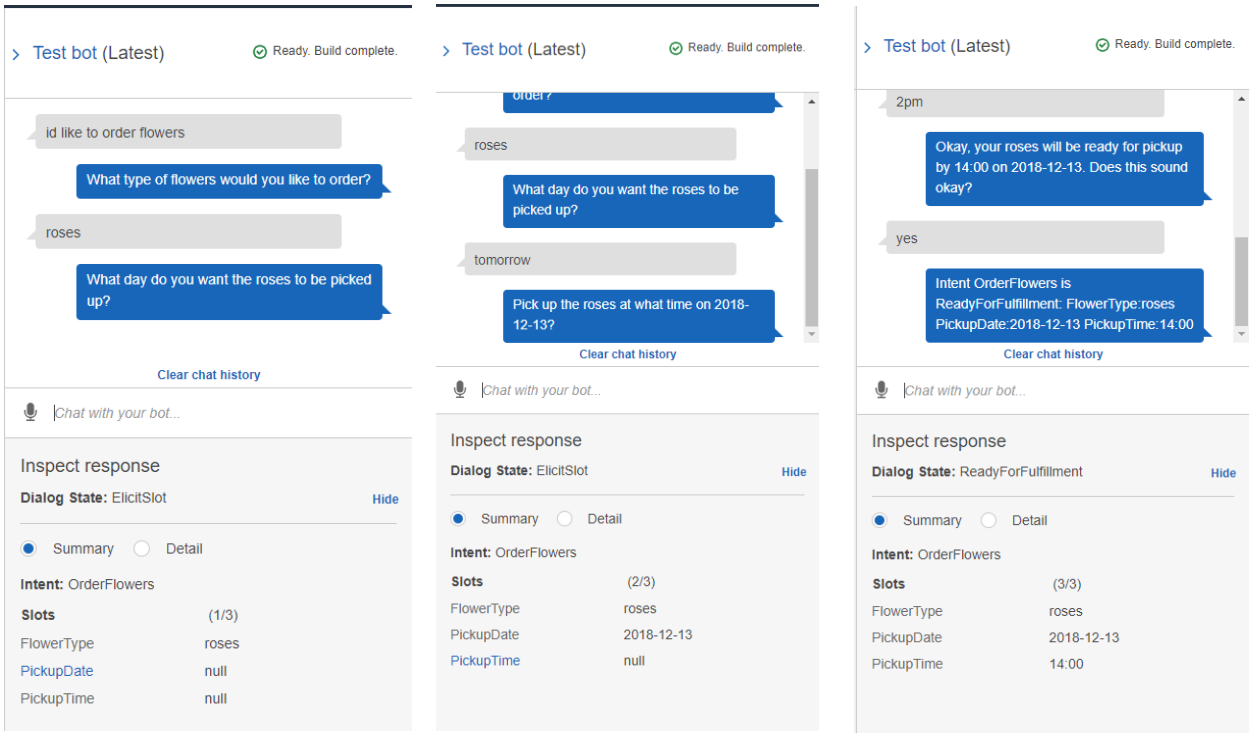


Figure 9, Figure 10 & Figure 11: Amazon Lex conversation flow (Technology Assessment)

3.4 Conclusions

After carrying out this phase of the project, some things have become somewhat clearer. Experimenting with the Microsoft Bot Framework has shown me that the scope and complexity of the framework was greater than first anticipated and will definitely have a much greater learning curve than was probably first anticipated. This is new technology to me, so I was expecting somewhat of a learning curve. After attending the panel review, I gained some further ideas that I will be looking into implementing as part of this project. First of all, realising that one platform may be quicker and easier to set up than a more complex one should not be a reason to *not* use it as part of the project. There needs to be a deeper analysis of its capabilities further than just its basic 'first-steps' functionality. This will be something that I will need to investigate in my own time before the commencement of the development phase next semester. Secondly, it could be a good idea to try to develop on both a complex platform and the simpler platform to compare and contrast functionalities and other aspects such as integrations with Web Apps or other forms of interaction. Also, my focus needs to narrow on what my application of this technology is actually going to be. An investigation will be done into how and why FAQ pages can or need to be enhanced. As well as looking into specific uses of the technology such as using a Knowledge Base to combine multiple possible answers into one

answer outputted to a user. Or being able to filter and pull data from multiple knowledge bases to output a single answer. All of this will need to be investigated before next semester.

Section 4: Detailed Project Proposal

4.1 Project Summary

Since commencing my 4th Year project, I have carried out extensive research into the design, development and the many possible use cases of Interactive Chatbots. My research has shown that there are many ways in which a Chatbot can be developed, and within the research, I discovered various development platforms such as Microsoft's 'Bot Framework', Amazon's 'Lex' and Google's 'Dialogflow', to name but a few. Research also uncovered that there are generally two types of Chatbots that are developed and used within industry which are, Rule-based Chatbots and Artificial Intelligence based Chatbots. For the purposes of this project, I have primarily been focusing my efforts on Rule-Based Chatbots, "in a rule-based approach, a bot answers questions based on some rules on which it is trained on." (Shridhar, 2019). Throughout the research process, my outlook on Chatbots and their uses changed and developed as I gained more of an understanding into the circumstances in which they could be used but more importantly, where they shouldn't be used for the sake of using them. This gave me a clearer understanding of what use cases may be better or more practical than others for the purposes of this project.

During the technology assessment phase of the project lifecycle, I experimented with each of the aforementioned development platforms which aided my understanding of Chatbot development and also gave me a greater idea of the direction in which I wanted to take this project. Since the technology assessment phase, I have carried out more research into the use cases of Chatbots and have also been critically thinking about the implementation idea that I have had since carrying out the technology assessment. This original idea involved taking a website's original Frequently Asked Questions (FAQ) section and turning it into a Chatbot conversation like interface. I had originally decided on this idea as a result of the research I had carried out, however, there are a few reasons as to which I feel this idea may not utilise the very reason Chatbots exist to its full potential. Some feedback I received during panel review sessions helped me to see this and also self-research showed that it can actually be a negative and unproductive/impractical task, in terms of design and User Experience (UX) to take what is already a feasible way of presenting information and altering the way a user interacts with

it. In other words, why change an implementation of information delivery that already works? “Chatbots are meant to chat with the user, not to produce one answer and then end the conversation.” (Goebel, 2019).

Thus, the direction in which I plan to take this project has somewhat altered since my original research. There are more practical and useful ways in which to implement a Chabot interface for improved Customer Experience and User Experience. The idea behind this project is to bring an improved UX to Users, and not to overcomplicate it, which I feel might have happened with the FAQ bot. For this reason, this project now aims to build upon the testing that I carried out during the technology assessment phase. This testing was in relation to the food industry and allowed users to utilise a conversational interface to make an order of their choice. The project that I am proposing will be a booking/reservation-based system for a restaurant, built using Google’s ‘Dialogflow’, that will be accessible through Facebook Messenger and also a Web Front-end developed using Twitter Bootstrap. Users will have the ability to utilise this Chatbot interface to make a reservation for a time and date of their choice and the bot will guide them through this process. I feel that this project idea is more viable as it has the potential for real-world implementation and usage. The research has also shown that interactive Chatbots, like this one, for businesses are highly utilised by consumers and are in fact a preferred method of interaction. Statistics show that a very high percentage of consumers prefer the experience of interacting with a business through text as opposed to in person or over the phone, “Eighty-nine percent of consumers prefer to engage with businesses via text, while 64 percent that do communicate in this way have positive impressions of the experience.” (Matthews, 2019).

4.2 Why are you doing this project?

The reason I have decided to carry out my 4th-year project on this subject is that it is an area of Computing that I find very interesting. As Chatbot interfaces are an emerging area of technology, I thought that it would be beneficial to myself and my professional development to further my understanding of such an area and to put it into practice in a practical, real-world type scenario. My interests lie in the areas of Front-End Web Development, User Interface Design/Development, User Experience Design/Development, Interaction Design/Development and emerging technologies so I feel that this project is suitable for my technical skill and professional development moving forward in my career. It would be a great

piece of work to add to my professional portfolio and it is in line with the types of careers I will subsequently be searching for after graduating.

4.3 What will you be doing?

For the purpose of this project, I will be developing an interactive Chatbot system using Dialogflow that will allow people to go through the steps of booking a reservation at a fictional restaurant. The Chatbot will be accessible through popular messaging platforms such as Telegram and Facebook Messenger and I also plan on developing a web front end using Twitter Bootstrap where the Chatbot can also be accessed and interacted with. The Chatbot will be integrated into the web from end using 'Kommunicate'. "Kommunicate allows you to add live chat in your website that enables you to serve your website visitors and customers through conversation." (Docs.kommunicate.io, 2019)

4.4 How will you be doing it?

For development purposes, I will be utilising the platform known as Dialogflow (previously API.AI) that is made by Google. "Dialogflow is an end-to-end, build-once deploy-everywhere development suite for creating conversational interfaces for websites, mobile applications, popular messaging platforms, and IoT devices." (Google Cloud, 2018). During the research/technology assessment phase, I tested Dialogflow and found the experience of using it to be intuitive yet complex enough to be in-depth and challenging.

Overview:

1. Design conversation flow, create conversation diagram
2. Design Chatbot
3. Design intents
4. Design entities
5. Design fulfilment
6. Develop Chatbot
7. Develop Web Front-End
8. Integrate Chatbot with Web Front-End and Messaging Platform

4.5 Who will be doing it?

I will be carrying out the project myself throughout the course of Year 4, Semester 8. I will also be in frequent communication with my Project Supervisor, providing updates and status reports as to how the project is coming along or if I have any questions about my project.

4.6 Where will it be done?

The project will be undertaken both college and also at home. The project doesn't require any physical resources from the college and is not constricted by location which is why the majority of it will be completed from home.

4.7 How long will it take?

The entire project will be completed over the course of Semester 8 which runs from Wednesday 23rd of January 2019 until the final submission deadline which is approximately the 20th of April 2019.

4.8 Background

The Chatbot that will be developed for this project will be a Rule-Based Chatbot for the purpose of making a restaurant reservation. Therefore, to begin development of this Chatbot, the conversation flow will need to be mapped out and designed in such a way that it takes into consideration the multiple ways in which the User may converse with the Bot. Lucidchart is an online diagram creation tool that can be used for this conversational flow diagram creation. A 'Dialogflow' Chatbot is essentially made up of a few components including, 'Agents', 'Intents', 'Entities', 'Contexts', 'Events' and 'Fulfillment'. Here is an overview of definitions for each of these components.

4.8.1 Definitions

- Agents
    - "Agents are best described as Natural Language Understanding (NLU) modules. These modules can be included in your app, website, product, or service and translate text or spoken user requests into actionable data. This

translation occurs when a user's utterance matches an intent within your agent." (Dialogflow, 2019)

- Intents and Responses
    - "To define how conversations work, you create intents in your agent that map user input to responses. In each intent, you define examples of user utterances that can trigger the intent, what to extract from the utterance, and how to respond." (Dialogflow, 2019)
- Entities
    - "Entities are Dialogflow's mechanism for identifying and extracting useful data from natural language inputs. While intents allow your agent to understand the motivation behind a particular user input, entities are used to pick out specific pieces of information that your users mention". (Dialogflow, 2019). Custom entities can be created to accommodate the specific Chatbot that is being built but there are also what are called 'System Entities' that are predefined by Dialogflow to extract common entities. These System Entities are denoted by '@sys.entityname'. '@sys.date' for example, would look for a date in a User's input.
- Validation
    - This aspect is where the bot's intelligence becomes more complex. It is the logic that allows the bot to differentiate between valid requests and invalid requests based on variables and the input from the user. For example, catching if a user tries to make a reservation for a date that is in the past. Or, making a reservation where the number of guests is less than or equal to 0.
- 'Small Talk' and Integration (Facebook & Web Front End).
    - "Small Talk is used to provide responses to casual conversation. This feature can greatly improve the user experience by covering common questions that may not pertain to your agent's intents you built." (Dialogflow, 2019)

4.8.2 Example Conversation Flow Diagram

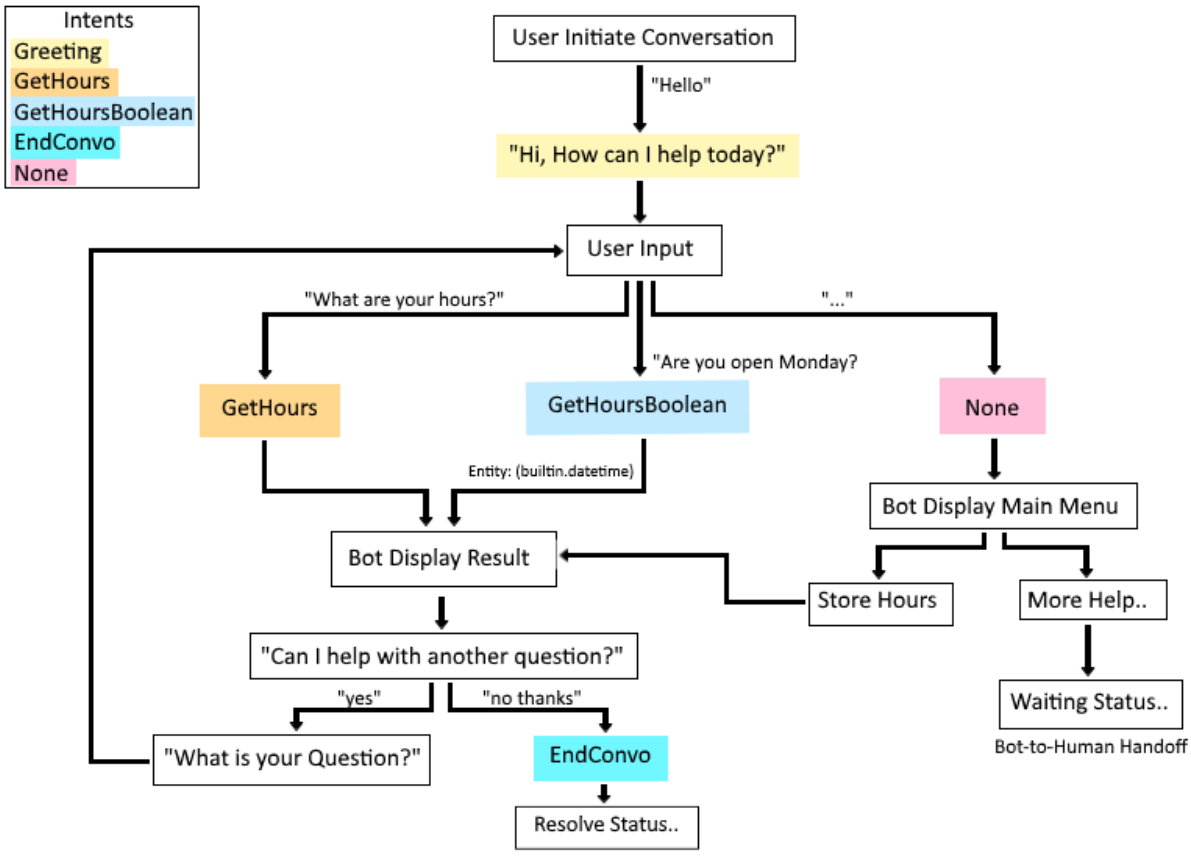


Figure 12: Example Conversation Flow Diagram: (Leung, 2019)

4.9 Objectives

1. **Initial Prototype (Chatbot basic functionality)**
    a. Define Conversation flow (diagram)
    b. Define User Intents and subsequent Responses
    c. Define Entities within the conversation
2. **Revised Prototype**
    a. Iterate on User Intents and Responses
    b. Iterate on Entities and Conversation flow
3. **Final Prototype**
    a. Design and Develop a Web Front End using Twitter Bootstrap and integrate Chatbot
    b. Integrate Chatbot into Facebook Messenger

4.10 Prototyping & Testing

For this project, I will be developing prototypes in a phased approach. During these phases, I will be designing the conversational flow for the Chatbot, developing the bot so that it interacts and converses with the user in the way that it should, testing the functionality to make sure it operates as expected and reporting my findings.

4.11 Technical Requirements

For this project, the technical requirements are all feasible and accessible. It does not require the purchase of any specific hardware as it is an entirely software-based project. The entirety of the project can also be carried out on my own personal computer, it is not a necessity to use the college lab resources however it is an option as the platform I will be using, Dialogflow, is a cloud-based meaning I could log into my account and have access to my work from any machine. Here is an overview of the services I will be utilising throughout the project:

- Dialogflow - free account
- Kommunicate - free account
- Lucidchart - free account
- Facebook - free account
- Bootstrap
- Visual Studio Code

"Use Dialogflow for free or pay as you go for enterprise-grade services. Standard Edition is free and covers the need of most developers, while Enterprise Edition offers paid enterprise support." (Dialogflow, 2019)

4.12 Risk Assessment

Throughout the process of this entire project, it is my responsibility to identify and mitigate any risks that may exist to the best of my ability. The research I carried out in the previous phase of the project lifecycle identified that Chatbots themselves can carry several risk factors which include Cost and Security. However, due to the fact that my Chatbot project will not be a production bot and is purely for self-development and research purposes, the security risks are greatly minimised. I have also made sure that the cost of the project will remain extremely

low if not totally at zero. There are also risks that relate to this project specifically which include the Time in which the project must be completed and the Scale/Scope of the project. I feel that the project is of substantial and adequate scope but is still possible to be completed within the time frame permitted.

4.13 Project Methodology/Approach

The objectives that I have stated above will be achieved through dedicated work and persistence during the project lifecycle. I will be utilising the knowledge that I have obtained thus far into the project and also consulting online resources for occasional guidance and direction. The first section of the development stage is crucial to get right as everything else follows on from there. The conversation flow needs to be designed with deep thought and consideration, this is where critical thinking skills will come into play.

4.13.1 User/Market Needs:

Chatbots are highly sought after nowadays in business as they have proven to be of great benefit to customer retention and engagement. “Chatbots provide better interaction with users. As far business is considered, bots are the best tool for keeping users on specific platform longer and engaged them by initiating and maintaining the conversation.” (Chatbots Journal, 2019).

4.14 Project Plan

The management of this project will be detrimental to making sure everything is going to plan and that I am on track with deliverables. I will allocate approximately two hours per day to the project averaging approximately fourteen hours per week which I feel is adequate time to stay on track. If I feel that this is not enough time or that I am falling behind on some aspect of the project I will, in turn, increase the time I am dedicating to it. I will also have frequent contact with my project supervisor, including weekly email updates to show work that has been completed and in-person meetings to demonstrate parts of the project I am working on. I will be documenting my progress, problems and challenges during the process of the project. Doing this will allow me to critically evaluate my problems and challenges and will increase the likelihood of overcoming them. I have learned this from past experiences.

4.15 Project Phases

The project as a whole will be split into three total prototype phases. Each one being an iteration on the last. The first of the prototype iterations that I plan on developing will include the design of various factors of the underlying core functionality of the Chatbot itself. This will include first steps such as creating an Agent in Dialogflow.

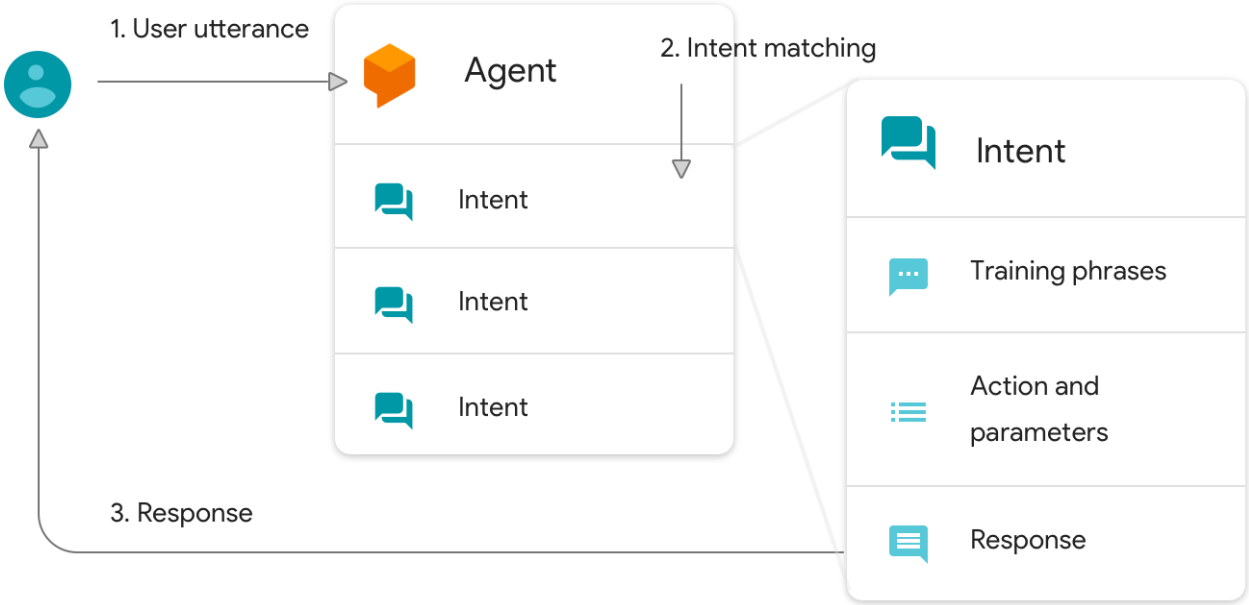


Figure 13: Example Agent: (Dialogflow, 2019)

The Agent is, in essence, the module that holds all the logic that allows the Chatbot to function. After the Agent is created then the Conversation model will need to be mapped out. This will be a process of writing out the ideal conversation that the Chatbot will accommodate but also identifying the multiple possible variations of this conversation with different ways that users may stray from this 'ideal' conversation. These variations must then be dealt with within the design of the conversation flow and the Chatbot architecture. The aim is to prepare responses for unexpected inputs by a user. This is where rigorous testing will play a part. By the end of the first prototype, the Agent will recognise 'intents' (user input) that will then trigger a pre-defined response. For the second prototype, the logic of the first prototype will be further developed and improved. It will be a more intelligent version of the first prototype and will incorporate more intelligent 'Validation' and 'Small talk'. The final prototype is where the Web Front-End will be developed using Twitter Bootstrap. The Bot from the second iteration will then be integrated with this website and will be accessible through that. I also plan on integrating it with Facebook Messenger to make it accessible in a variety of ways.

4.16 Deliverables and Milestones

During the lifecycle of the development phase of the project, there will be several deliverables and milestones. As the project development phase is an iterative approach, there will be 3 project prototype iterations that will be submitted in total, with the 3rd and final submission being in late April. This means that there will be 2 other prototype submissions prior to this approximately around the same time in both February and March.

4.17 Gantt Chart

In this section I have prepared a Gantt Chart as a visual representation of the plan for the 3 phases of the project.

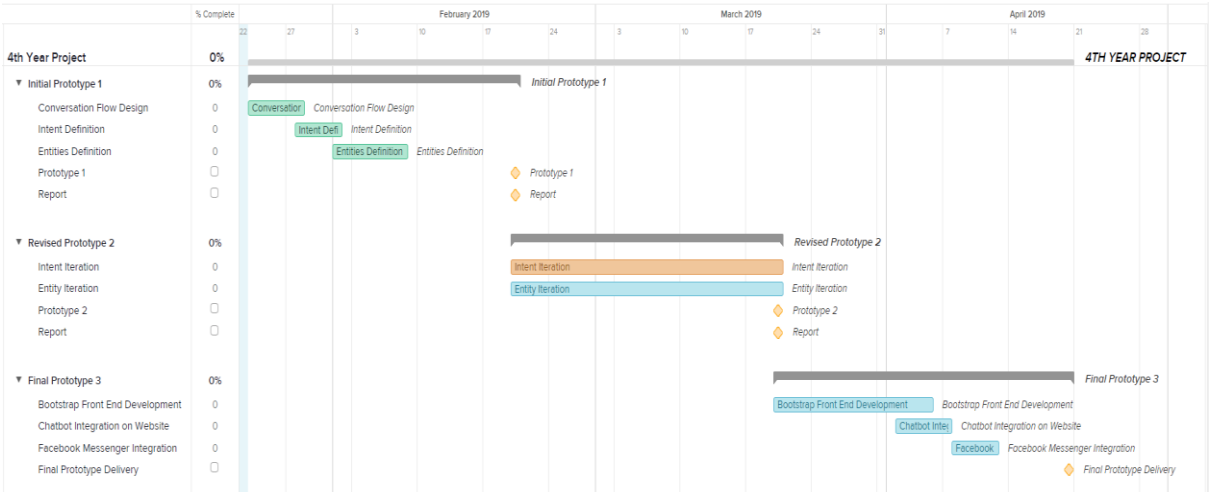


Figure 14: Gantt Chart

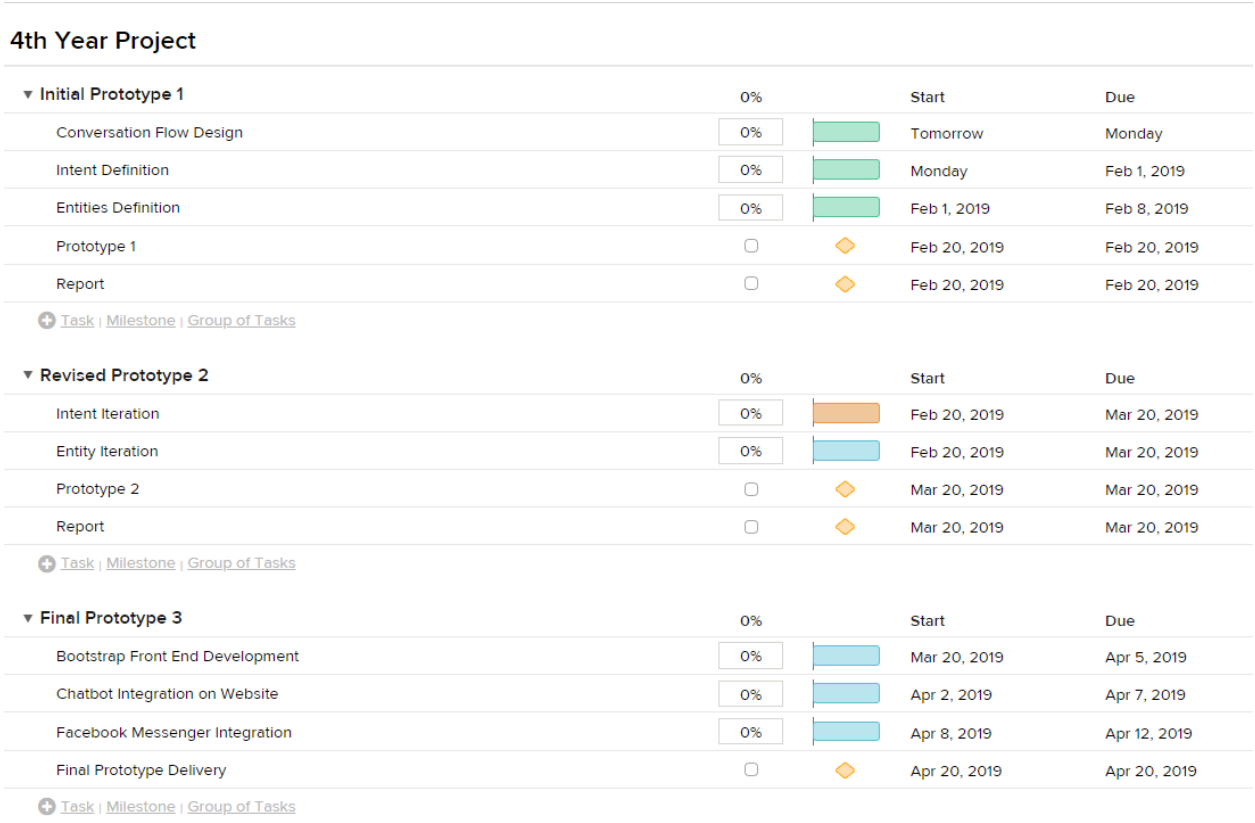


Figure 15: Gantt Chart - List View

4.18 Conclusions

In conclusion, I feel that the direction in which this project is now going is positive and is feasible in terms of scope and time scale. I look forward to implementing the plan I have created and executing on everything I have learned throughout the research phase. This area of technology exciting and very useful in real-world implementations. I feel that the decision to carry out a project in this area will help to increase my exposure and experience in working with emerging technologies and will also further expand my knowledge in a subject matter that is outside of the modules I have taken during the course of the college course. I also look forward to the challenge of the self-learning aspect to this project and I feel it will benefit my personal and professional body of work greatly.

Section 5: Iteration 1 - Project Report

5.0 Introduction

The aim of this report is to detail iteration one of my 4th Year Project entitled ‘Rule Based Interactive Chatbot Reservation System’.

5.0.1 Helpful Terminology

- **Agent** - The Chatbot itself. Holds all of the logic and is where the conversation is developed.
- **Intent** - a function/method created in the GUI of the bot that maps a user’s input to a response. E.g. ‘getDate’.
- **Entity** - attributes like ‘name’, ‘date’ extracted from conversations.
- **Fulfilment** - code that enables the bot to either trigger dynamic responses or to trigger back end actions such as database commitments.

5.1 Overview of Project and Target Audience

This project as a whole is based around Rule-Based Chatbots, the variety of platforms available to develop them, their use cases and their impact on User Experience. Iteration one of this project is based around the Use Case of a user making a restaurant reservation. The Chatbot that was developed during this iteration is aimed at users who wish to make a reservation at a restaurant for a chosen date and time and for a specified number of guests.

5.2 Platform Used

For the purpose of this iteration of the project, the platform that was used is called ‘Dialogflow’ which is a product owned by Google. “Dialogflow is an end-to-end, build-once deploy-everywhere development suite for creating conversational interfaces for websites, mobile applications, popular messaging platforms, and IoT devices.” (Google Cloud, 2019). The reason this platform was chosen is that it was investigated during the research phase of this project and it was deemed to be suitable for the Use Case that was defined as part of iteration one.

5.3 User Story

User Story 1: John
As a: Busy spouse making a long commute home from a busy day at work.
I want to: Be able to make a reservation for a restaurant for tomorrow evening for my wife and I using a casual messenger system while I sit on the train using my mobile phone.
So I can: Cut out talking on the phone and save time.

5.4 Use Case

The design for the first iteration of this project was to create a Chatbot system using Dialogflow that allowed users to make a reservation at a restaurant for a chosen date and time and for a specified number of guests and then those details would be extracted from the conversation and stored in a backend database. The database that would be used for this storage is Google's Firebase 'Realtime Database'. Firebase is the default option for database connections through Dialogflow.

At the beginning of this iteration, during the planning phase, the conversation flow of the bot was mapped out using a flow chart creation tool called 'Lucidchart'. This helped a lot to visualise the way in which the conversation was going to flow and what needed to happen, in terms of logic, at each stage of the conversation. Figure one below shows the first conversation flow that was designed.

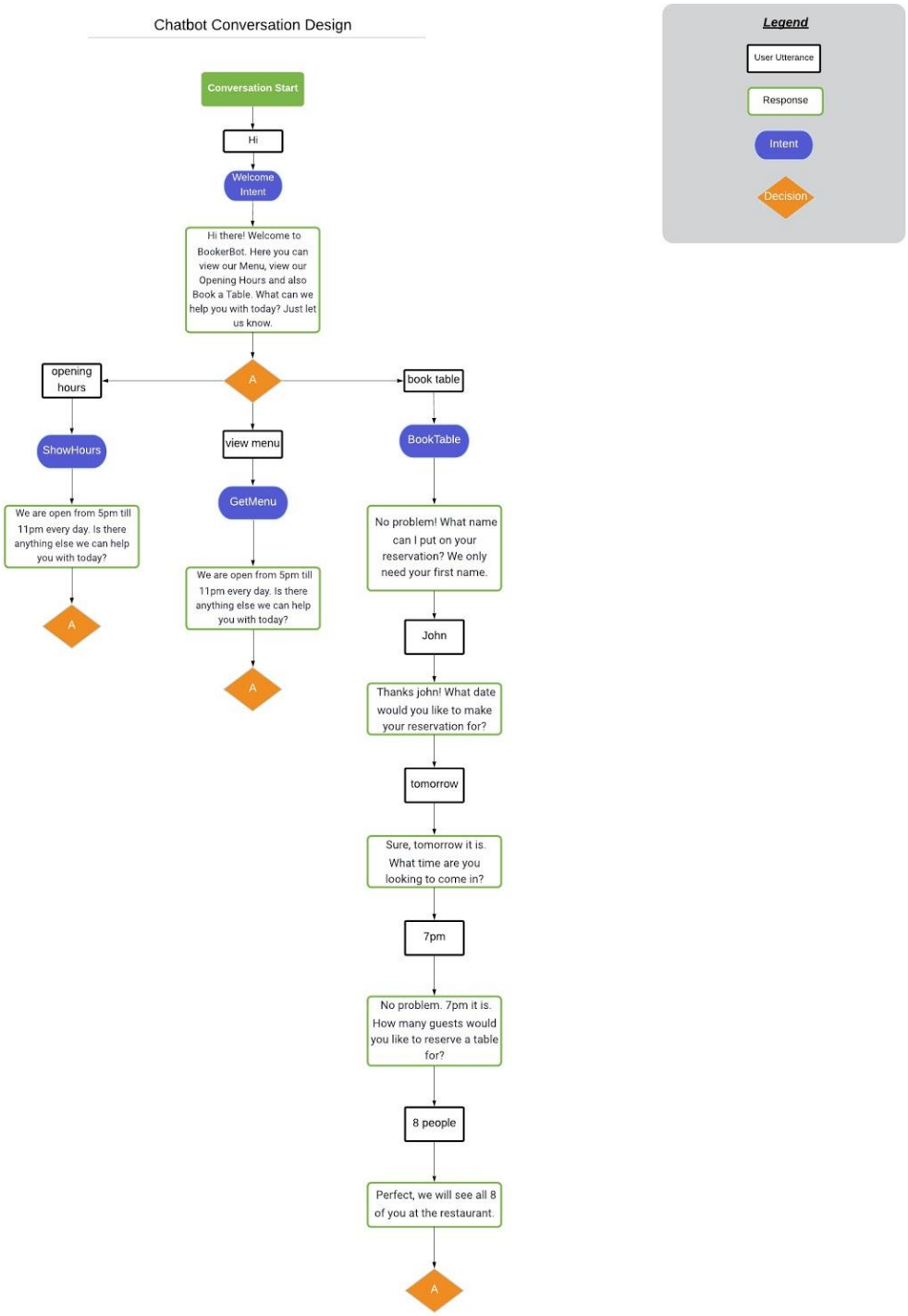


Figure 16: First Conversation Flow Diagram

As figure 16 shows, the conversation would be initiated by the user entering ‘hi’ or ‘hello’ and the bot’s ‘Welcome Intent’ would then be triggered. This would display a welcome message in which the user would be instructed as to how they can interact with the bot and what operations

it could perform for them. Three options were available which included showing the opening hours, showing the menu and booking a table. The latter being the most complex of the three options.

If the user typed in that they wanted to book a table, the bot would guide the user through the conversation asking them the relevant questions at the right time so that it could then extract the required information in order to make the reservation for the restaurant. Or, the user also had the option to do the entire booking with one statement issued to the bot containing all of the relevant information. If some important entity was missing from the user's statement, like the date, for example, the bot would prompt the user for that entity as it is marked as 'required' by the back-end logic of the bot.

Figure 17: BookTable intent with 'required' entities

Problems began to arise with this 'one-statement' method of attempting to create a reservation. When trying to extract all of the entities (date, time and guests) at once and store them in the database, it would not work. A lot of time was spent during iteration one trying to resolve this issue as it was one of the main pieces of functionality that made the bot substantial. The process of capturing the information from the conversation and storing is called triggered by a process called 'fulfilment' in the Dialogflow platform and this fulfilment is coded through JavaScript in the 'fulfilment console' of the Dialogflow agent. This fulfilment was not being triggered in the way that was needed for all of the entities to be extracted and stored separately.

Below is the code that was used in an attempt to extract the entities and store them in the database.

```
function saveDate(agent) {
    let dateParam = agent.parameters.date;
    let date = dateParam;

    return admin.database().ref('dates').push({name:date}).then((snapshot) => {
        console.log('database write successful:' + snapshot.ref.toString());
    });
}

function saveTime(agent) {
  const timeParam = agent.parameters.time;
  const time = timeParam;

  return admin.database().ref('time').push({time:time}).then((snapshot) => {
      console.log('database write successful:' + snapshot.ref.toString());
  });
}

  function saveGuests(agent) {
      const guestsParam = agent.parameters.guests;
      const guests = guestsParam;

      return admin.database().ref('guests').push({name:guests}).then((snapshot) => {
          console.log('database write successful:' + snapshot.ref.toString());
      });
  }
```

Figure 18: JavaScript Functions for storing entities in the database

```
let intentMap = new Map();
intentMap.set('Default Welcome Intent', welcome);
intentMap.set('Default Fallback Intent', fallback);
intentMap.set('book_table', createBooking);
intentMap.set('book_table', saveGuests);
intentMap.set('book_table', saveDate);
intentMap.set('book_table', saveTime);
```

Figure 19: Mapping Intents to functions

After lots of testing, researching and re-testing I figured out that the reason that this implementation was not being successful was due to the fact that one intent cannot be mapped to multiple functions written in the fulfilment. As Figure 4 above shows, the intent ‘book_table’ is mapped to multiple functions that are written above in Figure 3. This caused conflicts in the extraction and commitment of entities to the database.

Once this issue was identified it was time to rethink the design of the conversation in a way where all entities could be extracted and committed to the database separately. For this, the book_table intent would have to be separated out into several individual intents that captured the relevant entities separately. At this stage, the conversation was slightly redesigned to incorporate these new intents. The core logic is still of close resemblance to the original, however, it makes use of functionality inside Dialogflow that is known as 'Contexts'. Contexts allow intents to be chained together so the conversation still flows in an optimal way to obtain all of the relevant information that needs to be captured even while all of the intents are separated. Contexts are broken into Input Contexts and Output Contexts. An output context is added to an intent that is awaiting the next intent in order for the conversation to move forward. The output context tells the intent that it is placed in where to go next. This is achieved by adding that context (used as an output context) as an input context in the intent that it should go to next. Here are some examples:

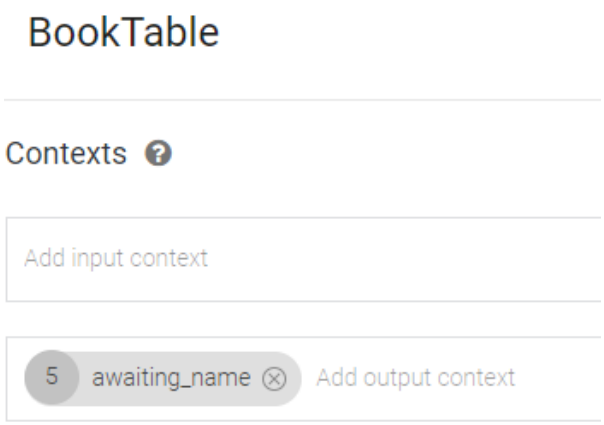


Figure 20: BookTable output context

Figure 20 shows the intent BookTable, which the first intent executed when a user wants to book a table. In this intent, there is no input context as it is the beginning of the chain of intents that will be triggered throughout the entire process. It does, however, have an output context of 'awaiting_name' as 'GetName' is the next desired intent in the chain.

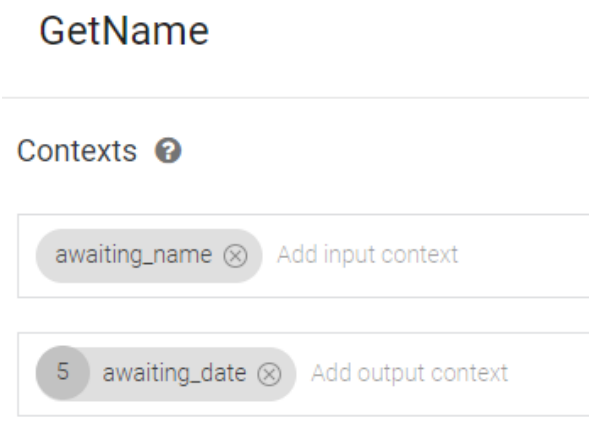


Figure 21: GetName intent with input and output contexts

Figure 21 then shows the GetName intent with both input and output contexts defined. To complete a link in the chain, i.e go from one intent to the next in the conversation flow, the input context must match the output context from the previous intent. So, in this case, the GetName has an input context of ‘awaiting_name’ as that is what it is now waiting for from the user, and it has an output context of ‘awaiting_date’ as this is the next intent that the conversation will be triggering. This process is then repeated for as long as is required by the conversation use case.

The changes in conversation flow were also reflected in the conversation flow design diagram seen in Figure 22.

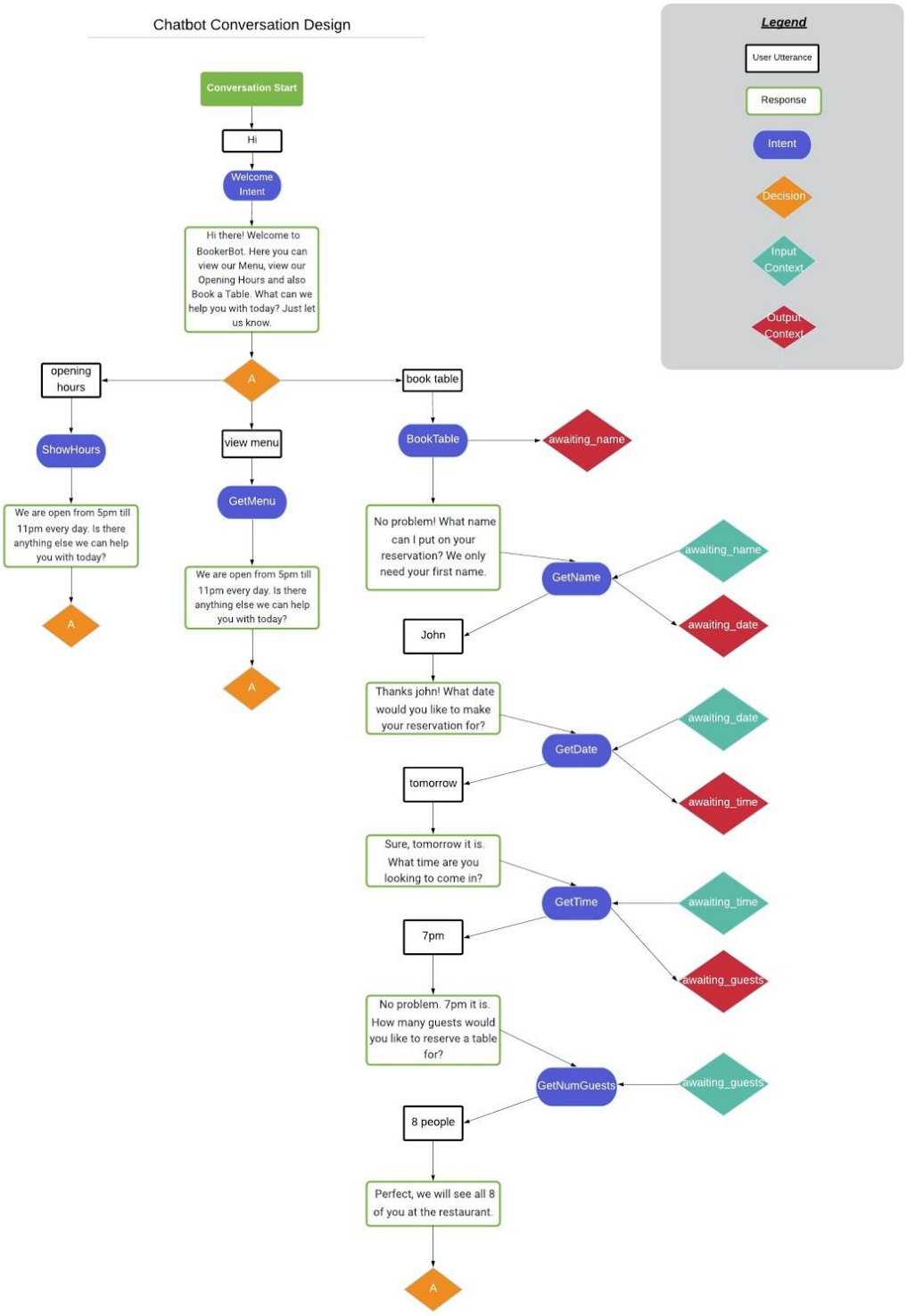


Figure 22: Updated Conversation Flow Diagram

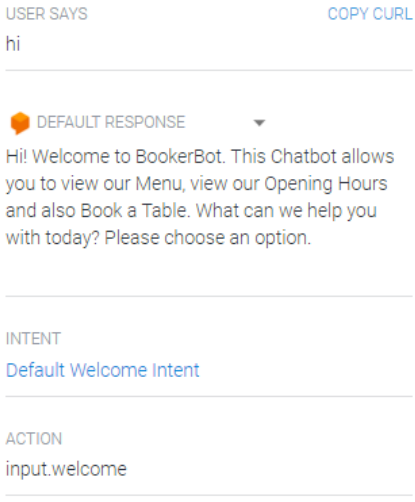
Once the redesign of the Chatbot was implemented in terms of Intents and Contexts, it was time to update the JavaScript Fulfilment in the Fulfillment console. Now that all of the intents

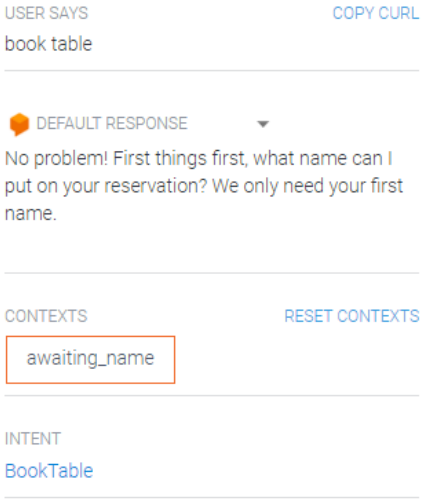
were separated, each intent could be mapped to its own function that could then extract that entity and store it in the database.

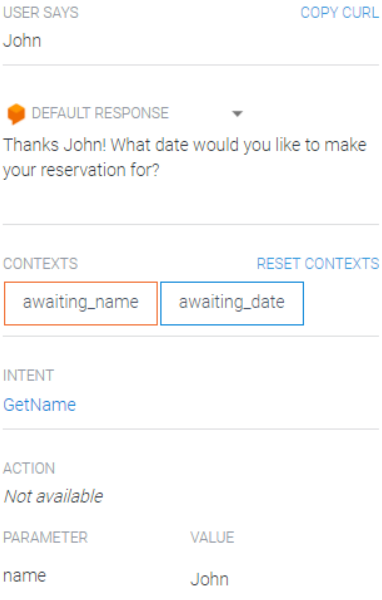
```
function getName(agent) {
    const nameParam = agent.parameters.name;
    const name = nameParam;

    agent.add(`Thanks ` + name + `! What date would you like to make your reservation for?`);

    return admin.database().ref('/name').push({name:name}).then((snapshot) => {
        console.log('database write successful:' + snapshot.ref.toString());
    });
}

 function getDate(agent) {
    const dateParam = agent.parameters.date;
    const date = dateParam;

   agent.add(`Sure,` + date + ` it is. Now please let us know what time you're looking to come in? Please use e.g '7pm' format.`);

    return admin.database().ref('/date').push({date:date}).then((snapshot) => {
        console.log('database write successful:' + snapshot.ref.toString());
    });
}

function getTime(agent) {
    const timeParam = agent.parameters.time;
    const time = timeParam;

  agent.add(`No problem.` + time + ` it is. How many guests would you like to reserve a table for?`);

    return admin.database().ref('/time').push({time:time}).then((snapshot) => {
        console.log('database write successful:' + snapshot.ref.toString());
    });
}

function getNumGuests(agent) {
    const numGuestsParam = agent.parameters.guests;
    const numGuests = numGuestsParam;

      agent.add(`Perfect, we will see all ` + numGuests + ` of you at the restaurant.`);

      return admin.database().ref('/guests').push({guests:numGuests}).then((snapshot) => {
      console.log('database write successful:' + snapshot.ref.toString());
    });


}


// Run the proper function handler based on the matched Dialogflow intent name
let intentMap = new Map();
intentMap.set('Default Welcome Intent', welcome);
intentMap.set('Default Fallback Intent', fallback);
intentMap.set('GetName', getName);
intentMap.set('GetDate', getDate);
intentMap.set('GetTime', getTime);
intentMap.set('GetNumGuests', getNumGuests);
agent.handleRequest(intentMap);
});
```

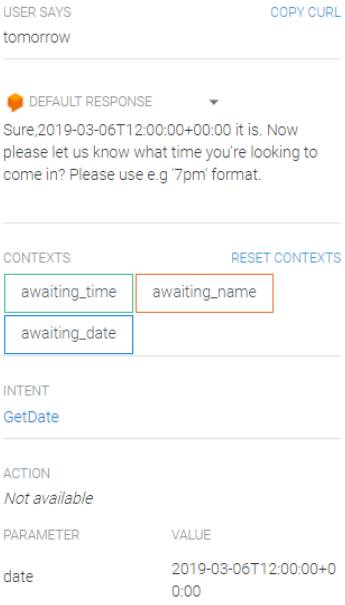
Figure 23: Updated Fulfillment JavaScript code

The conversation could then be executed as follows: (read from left to right)









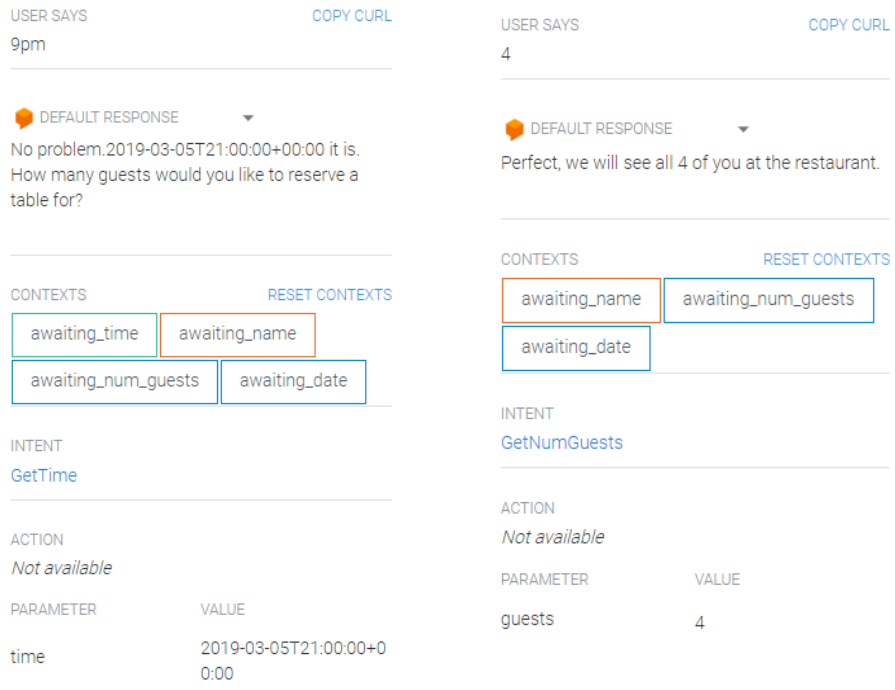


Figure 24, Figure 25, Figure 26, Figure 27, Figure 28 & Figure 29: Dialogflow Conversation Flow Using Contexts

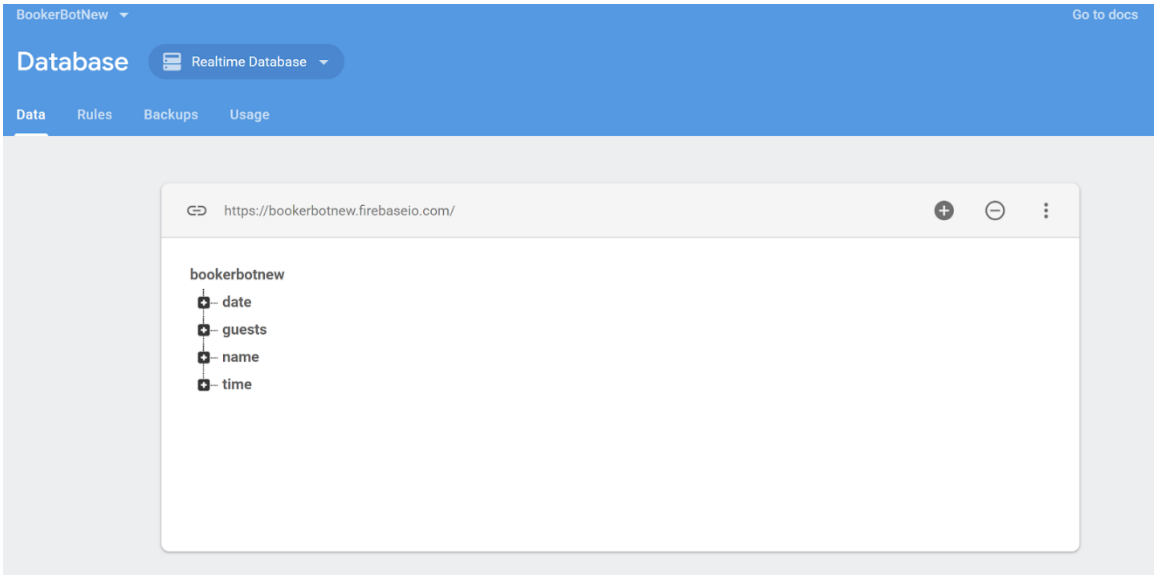


Figure 30: Saved data in Firebase Realtime Database

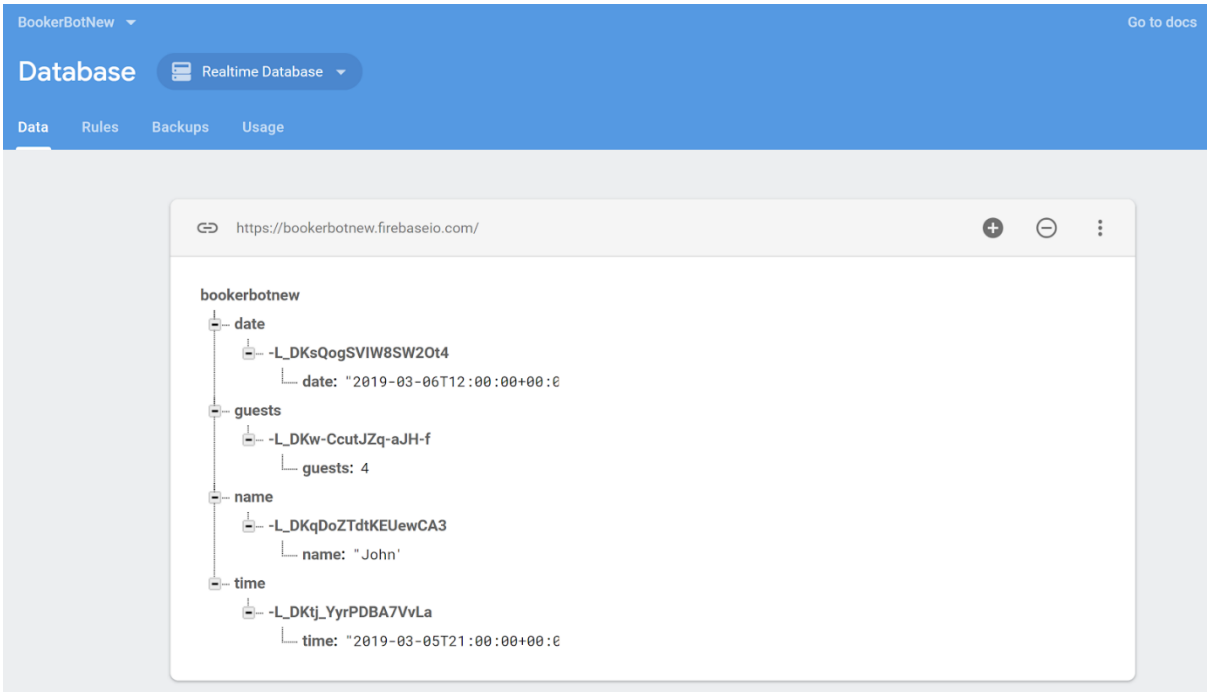


Figure 31: Extracted and Committed data displayed in JSON format

As seen in Figures 24 through 31, all data is accurately recognised, fed back into the conversation to the user and finally extracted from the conversation and stored in the Firebase Realtime Database.

5.5 Conclusion

During this development process, there were a variety of issues and bugs that were identified and overcome. However, there were some that couldn't be overcome due to factors such as time and knowledge/ability. Some of which include adding logic to allow users to only make a reservation between specific times and also retrieving data from the database and displaying it to the user dynamically such as a Food Specials Menu. During the next iterations of this project, I hope to be able to take the knowledge that I have learned from this process and apply it in ways that improves the overall User Experience of interacting with these Chatbots.

Section 6 - Iteration 2 Report

6.1 Introduction

Iteration two of my 4th Year IT management project changed somewhat since the project proposal. My original proposal was to continue the work that was carried out for iteration one but improving the functionality. However, after some meetings and discussions with my project supervisor and the project panel, it was decided collectively that a Chatbot creation platform comparison would make for an interesting project iteration two.

6.2 Platform Used

Amazon Lex was the platform that was used for iteration two of the project. “Amazon Lex is an AWS service for building conversational interfaces for applications using voice and text” (Docs.aws.amazon.com, 2019). This platform was used with the objective in mind of creating a similar restaurant reservation as iteration one which used Google’s Dialogflow. This platform was discovered in the research phase of the 4th Year Project and was also tested in the Technology Assessment phase. It was then deemed an appropriate platform to carry out an implementation of a Rule-Based Reservation system.

6.3 What Was Achieved

During the time spent using this platform to create the restaurant reservation system, it was clear that more time would be needed in order to achieve a system that functioned in a comparable way to the system produced in iteration one. Two systems were ultimately created during iteration two using Amazon Lex. One of the versions was a basic version that returned the input parameters back to the user and did not implement any backend logic which in the case of Amazon Lex, uses Lambda Functions. There was also a second version of the system which did implement a Lambda function that could allow some logic and validation to be applied to the Chatbot in the event of a successful implementation. However, the end result of this system still contained bugs and needed more testing and updating until it would work one hundred per cent as intended.

Version one of the system, as mentioned, returned the parameters that were entered by the user back to them through the conversational UI. This was achieved through the use of Intents, Parameter Creation, Slot-Filling and Basic Fulfillment.

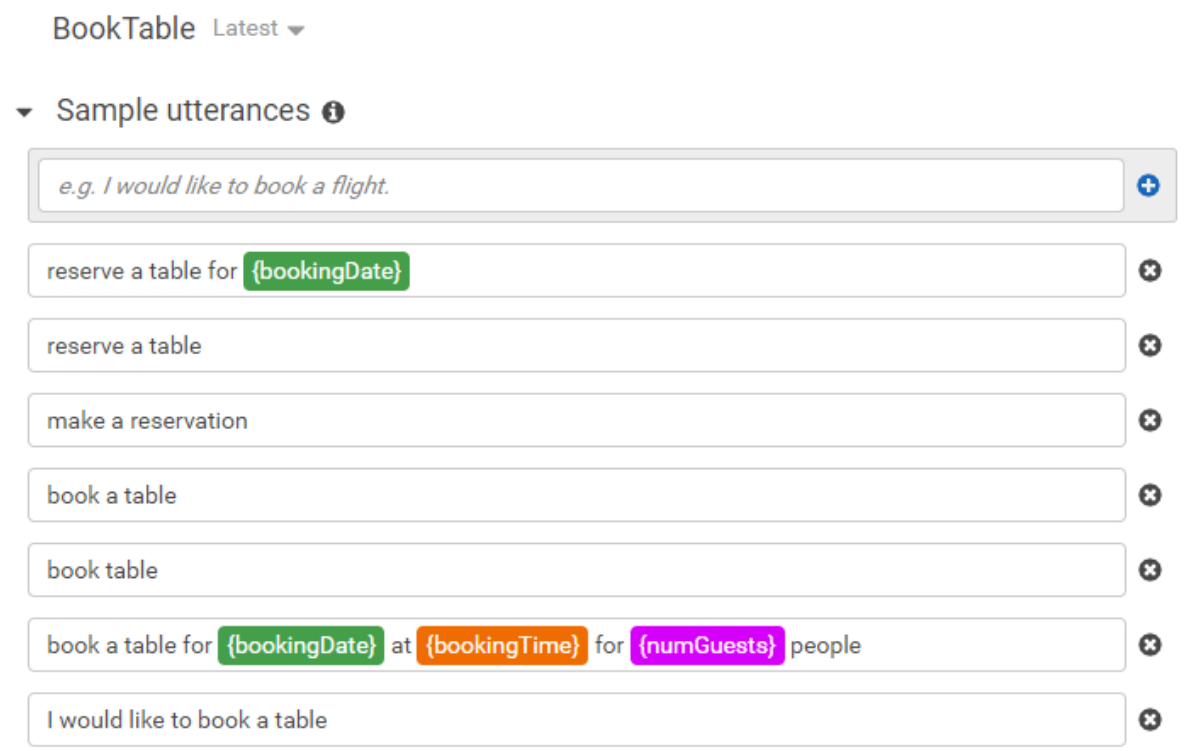


Figure 32: Amazon Lex - BookTable Intent - User Utterances

Figure 32 shows the sample user utterances that were given to the bot so that it knew what to expect the user to enter. In this intent, which was the BookTable intent, utterances such as 'book a table' and 'make a reservation' were given to the bot so if a user entered these statements the intent would be triggered.

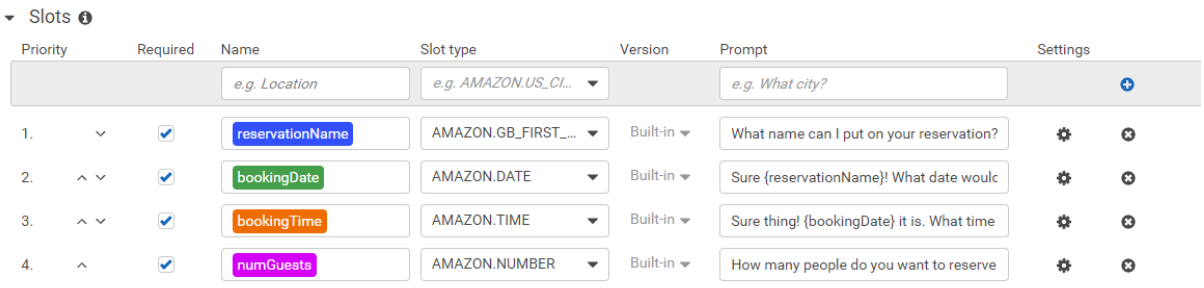


Figure 33: BookTable intent Slots (parameters)

Figure 33 shows the slots that were created for obtaining the key information from the user when they wanted to make a reservation at the restaurant. The key pieces of information were as follows: 1) The user's name that would be put on the reservation obtained by 'reservationName', 2) The date they wished to book for obtained by 'bookingDate', 3) The

time they wished to book for obtained by ‘bookingTime’ and 4) The number of guests they wished to reserve a table for obtained by ‘numGuests’.

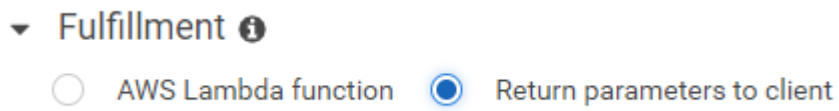


Figure 34: Fulfillment - Return parameters to client

Figure 34 shows the Fulfillment option that was used for this implementation of the system. With ‘Return parameters to client’ chosen, the bot would merely return to the user the information they had entered during the booking process. This means that it is not linked to a backend Lambda function which allows further logic and validation to be implemented. The conversation flow from this implementation can be seen in Figures 35 through 38.

6.4 Conversation Flow - Returning Parameters to Client

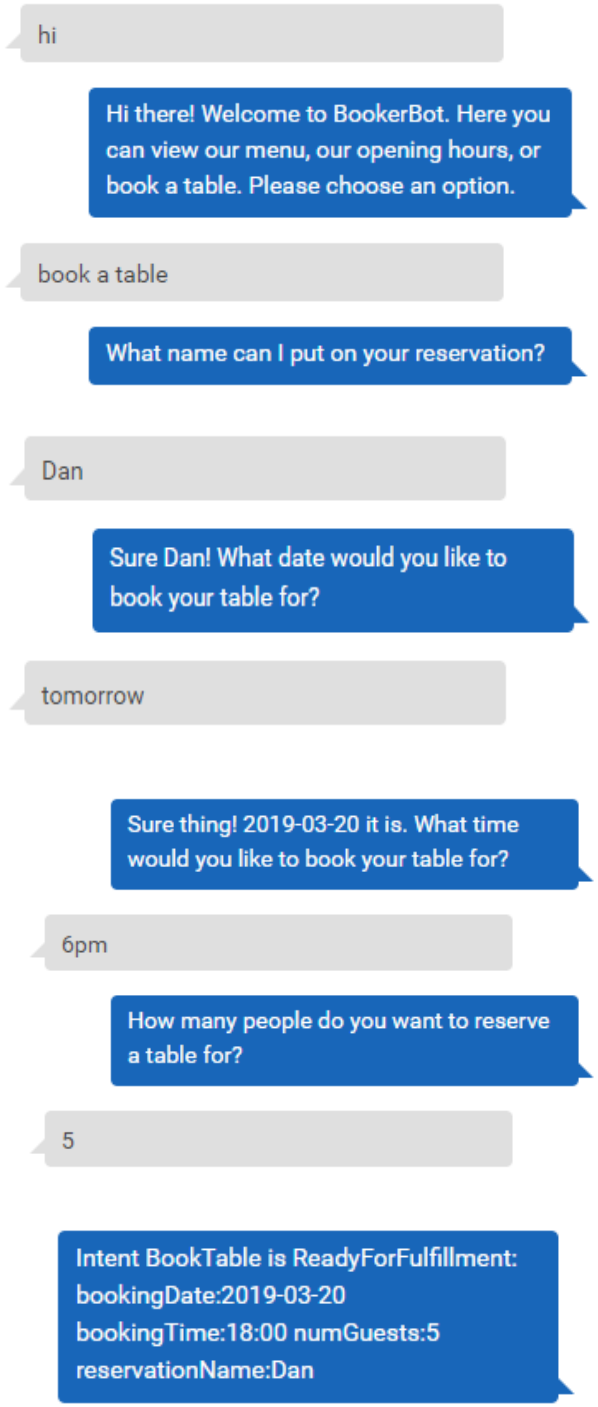


Figure 35, Figure 36, Figure 37 & Figure 38: Amazon Lex Conversation Flow using Return Parameters to Client

Figure 35 through 38 are screenshots of the conversation flow and they show the final output of the conversation when 'Return Parameters to client' is used. It gives the user an unformatted,

code-like response to the conversation and uses the names of the parameters that are built into the intent. This is not an ideal solution as these parameters' names should not be shown to a user as it breaks the idea that they are 'chatting' to a human-like bot. A response could be given in a human-like way using the 'Response' section of the Intent Builder, but this still does not allow validation of aspects like booking date and time restrictions. This is why a Lambda function must be implemented.

Version two of the system was then created which attempted to use a Lambda Function to enable validation of the user input through the use of JavaScript. This function is executed once it has been created and linked to through the intent builder by selecting 'AWS Lambda Function' in the 'Fulfillment' section and then choosing the name of the created function from the dropdown menu which is shown in Figure 39.

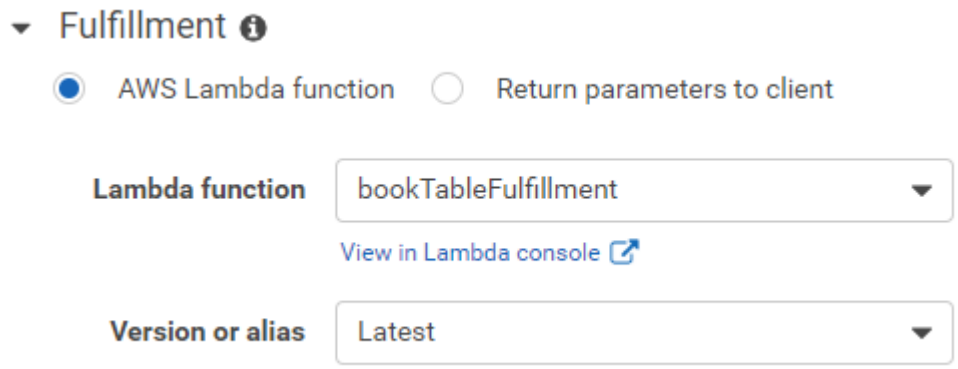


Figure 39: Selecting Lambda Function in the intent builder

During this implementation stage, it took a lot of time trying to get the function to trigger when using the bot. Lots of errors were faced and lots of testing was done to try and successfully get the function to trigger. An example of a persistent error that was faced throughout the process is shown in Figure 40.

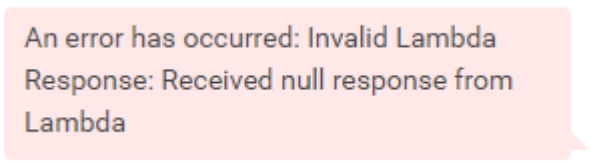


Figure 40: Lambda Function Response Error

This error was subsequently rectified after reading some documentation online and editing the code within the Lambda function. This was a part of the iteration build that really slowed down

progress as a lot of time was spent attempting to get this function connected properly and triggering when it needed to be triggered. If time was not a constraint, I feel that more could have been achieved within this iteration.

Figure 41: Output with Lambda Function Fulfillment

Figure 41 shows the conversational output that was achieved through the Lambda Function Fulfillment. It is evident that the response is a lot more personable than the previous implementation of returning the input parameters to the user. One of the main challenges faced, however, was implementing validation such as time and date restrictions within the Lambda function. As already mentioned, if more time was available to work on this functionality it is more likely that it would have been achieved.

6.5 Comparison

There are some areas of comparison between Amazon Lex and Dialogflow that were noted during the process of iteration two. After working with Dialogflow in depth for iteration one, it was quite a shift moving over to Amazon Lex. The layout and user interface of Lex is largely similar to Dialogflow, having access to all Chatbots on the left-hand side of the screen, the intent builder interface in the centre of the screen and then the bot test console over on the right-hand-side of the screen. The differences mainly came in performance and user experience. During my time using and testing the Lex platform, it seemed to be considerably slower than Dialogflow when building the bot and preparing it for testing. It also faced issues in recognising changes that had been made to the bot's logic and sometimes needed a hard refresh of the entire browser in order for the changes to the intent to take effect in the test console. These types of issues were tolerable at first but became somewhat irritating when trying to make quick changes and test them in a time-conscious manner.

The Dialogflow platform, in my opinion, and experience, was much more user-friendly and was a lot easier to get to grips with from the offset. Its visual design and language used in its

content felt much more inviting and encouraging than that of Amazon's Lex. There was a feeling, when using Dialogflow, that the creators wanted your experience to be pleasant and a sense of fun when creating Chatbots. This made the overall experience of using Dialogflow much more enjoyable and didn't cause and feelings of wanting to stop using the platform at any time for any reason.

Another aspect of comparison between Amazon Lex and Dialogflow is that of Natural Language Processing (NLP). "Natural language processing (NLP) is a branch of artificial intelligence that helps computers understand, interpret and manipulate human language. " (Sas.com, 2019). Based on the experience of using both platforms, it was found that the Natural Language Processing capabilities built into Dialogflow are far superior to that built into Amazon Lex. When creating an intent in the Lex platform, the utterances given to the bot must match exactly what the bot receives in from the user. This was found not to be the case with Dialogflow. A few example user utterances could be given to the agent within an intent in Dialogflow and then the bot would train itself and actually recognise user utterances that were close enough to what it had been trained on. It didn't have to match exactly word for word which was the case for Amazon Lex. This made the platform feel a lot less 'intelligent' than its competitor Dialogflow.

One notable feature of Amazon Lex, however, that was not found to be present on Dialogflow during testing, is version control. Agents and intents implement a welcome feature dubbed as 'Aliases' by Amazon that allows developers to backtrack to earlier versions of an Agent or Intent that they have created. This can be accessed via a dropdown menu beside the name of the Agent or Intent that is currently being worked on and can be seen below in figure 7.

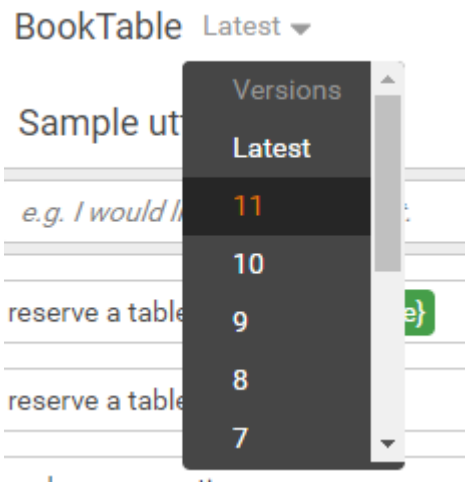


Figure 42: Amazon Lex Version Control - 'Aliases'

This feature is extremely useful, and it would be great to see Dialogflow implementing it into their platform. When working on an agent or intent, mistakes can be made and sometimes these mistakes can be irreversible. Version control helps to mitigate ruined or lost projects due to these types of mistakes or errors.

6.6 Conclusions

In conclusion, the experience of working with Amazon Lex was found to be interesting and of benefit for educational purposes. However, the experience of working with Dialogflow greatly outshined it in many ways. One large contributing factor to this is online resources, including documentation and also public forums and discussion boards. From experience, it seems as though there are a lot more people using Dialogflow than Amazon Lex as it was found that there was a lot more helpful content available online created specifically for Dialogflow users. The documentation created by Amazon that is available within the Lex platform was found to be overly complex and not straight forward or easy to understand. That is not to say it is totally incomprehensible or unintelligible, it just seemed to be more convoluted than the documentation provided by Google on the Dialogflow Platform. For iteration three of this project, Dialogflow will be used again to create another version of the Rule-Based Interactive Reservation System that will be more in-depth and complex using the knowledge gained throughout iteration one and two combined.

Section 7 – Iteration 3 Report

7.1 Introduction

The aim of this report is to detail the final iteration, iteration three, of my 4th Year Project entitled 'Rule Based Interactive Chatbot Reservation System'. This report will highlight the plans that were put in place for iteration three, upon completion of iteration one and two, the platform that was used to develop iteration three, the design and development changes that were put in place, what was achieved and the challenges that were faced throughout development.

7.2 Platform Used

For iteration three of this project, Google's Dialogflow platform was used once again, as it was for iteration one. The reason for using Dialogflow again was based on the findings, the comparisons that were made and conclusions that were drawn from using Amazon Lex for the development of iteration two. After this decision was made, the process of developing iteration three began. To begin with, a reassessment of everything that had been completed up to that point was carried out. This reassessment was mainly focused on what had been completed during iteration one as this was what was going to be used as a starting point and subsequently further developed for iteration three. It was necessary to understand what was achieved in iteration one, what challenges were faced and how these challenges could be overcome for iteration three. The development that was completed in iteration one provided me with some core functionality that could be worked with and developed further for iteration three. Dialogflow was also a much more familiar platform to work with than Amazon Lex at that stage so it was helpful working in an environment that felt comfortable.

7.3 What Was Achieved

The core functionality that was achieved in iteration one included the user being able to interact with a Dialogflow Chatbot to view the opening hours of a fictional restaurant, view a food menu and book a table. However, these were very basic versions of the core functionality that needed to be further developed and enhanced/added to for iteration three. From iteration one, the functionality of writing a user's booking information to the database was achieved. This functionality, however, had its issues in terms of scalability and manipulation for iteration three. The issue was with the way in which the data was being written to the database. The data

was written successfully but it did not form any sort of structure in the database. Each attribute was written as a separate node and did not belong to any parent node. This can be seen in figures 43 and 44.

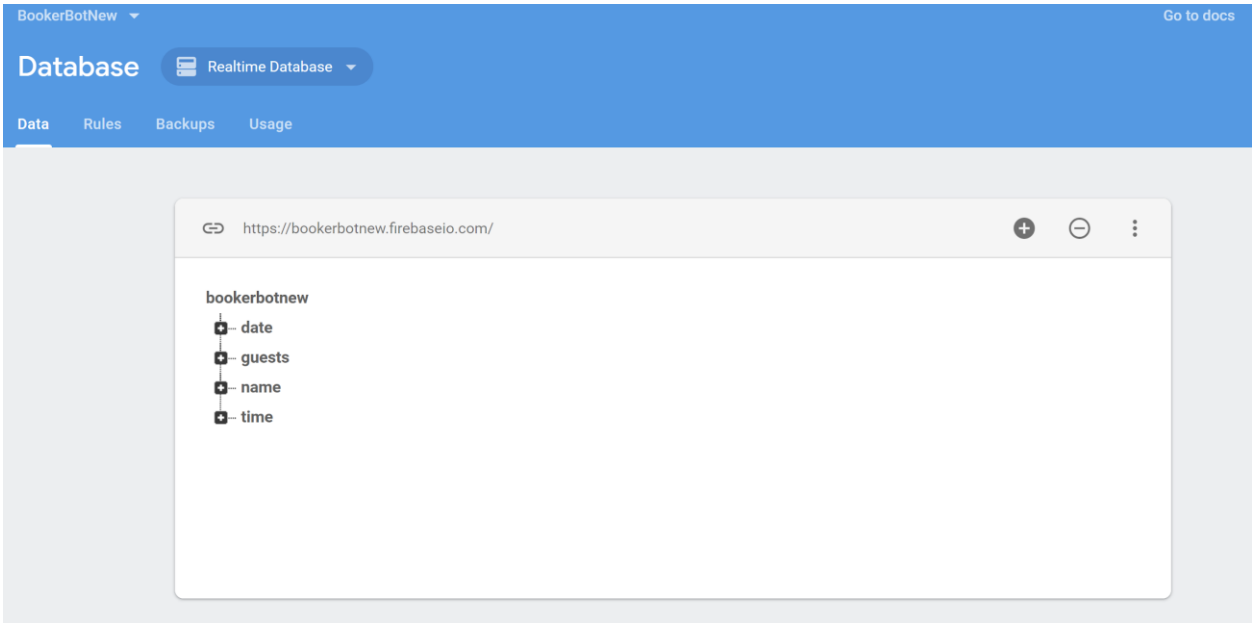


Figure 43: Example 1 of writing data, without parent/child structure

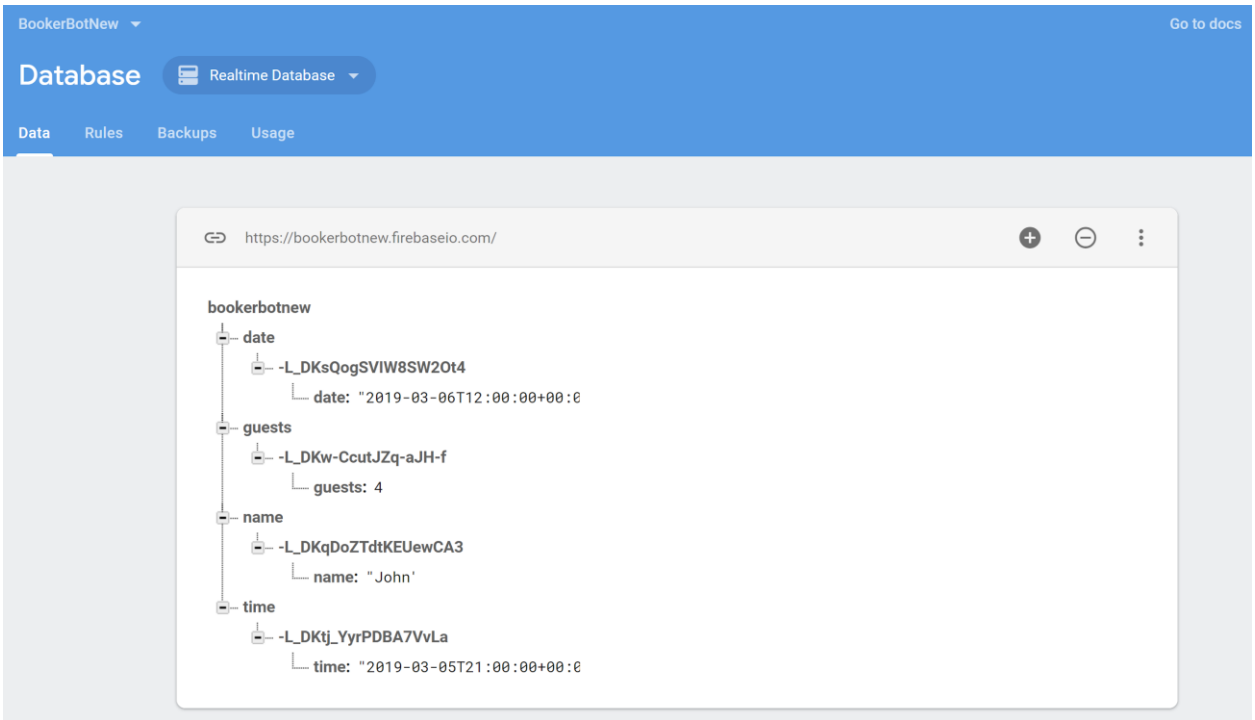


Figure 44: Example 2 of writing data, without parent/child structure - Random Keys

This basic method of writing data to the database was achieved through the function that can be seen in figure 45. It used a 'push' method that created a node in the database for each attribute relating to the booking information, for example, the user's name. This method, however, stored the attribute under an additional node with a randomly generated key value that can be seen in figure 44.

```
//Original Method
return admin.database().ref('/name').push({name:name}).then((snapshot) => {
  console.log('database write successful:' + snapshot.ref.toString());
  });
```

Figure 45: Original method used to 'write to database'

This led to further research and experimentation of other ways to write information to the database and to avoid the creation of a randomly generated key value and also to write the data to a specified node location with complete control. This could be achieved with the function that can be seen in figure 46. This function created a node in the database called 'bookingInfo' upon receiving a name from the user. Then this database location was stored in a variable called 'bookingInfo' which could then be written to using the push method. The function then used a return statement to create a child node of 'bookingInfo' called 'name' which had a value of 'name', which was a variable set to whatever was input by the user.

```
//Second Method
bookingInfo.push ({name:name});
return admin.database().ref('name').set({
  name: name
  });
```

Figure 46: Second 'write to database' function using push

The result of this function can be seen in figure 47.

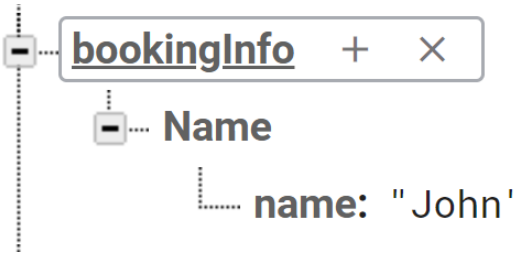


Figure 47: Parent/Child node bookingInfo structure using push method

The issue with this method was that when the next piece of information was captured by the Chatbot and written to the database, for example, the booking date, the previously stored entity would be overwritten. This was another issue that needed to be overcome. This led to the third and final way of writing user information to the database in a structured, maintainable and scalable way. The function that was created to do this can be seen in figure 48. This function still creates a parent node in the database called 'bookingInfo' under which all booking information is stored. However, instead of using a push method, it uses a combination of a '.child' method and a '.set' method.

```
//Final Method
var bookingInfo = admin.database().ref("bookingInfo/");
bookingInfo.child("Name").set({name:capitalisedName});
```

Figure 48: Final method of writing to the database

This resulted in a parent node called 'bookingInfo' being created and then all subsequent pieces of necessary booking information including, Date, Name, Time and Number of Guests, all being written and stored in their own child node named respectively, enabling this data to be used later in other functions of the Chatbot. The function seen in figure 48 is the basis for gathering all of the other booking information. The result of this function that gathers, writes and stores all booking information can be seen in figure 49.

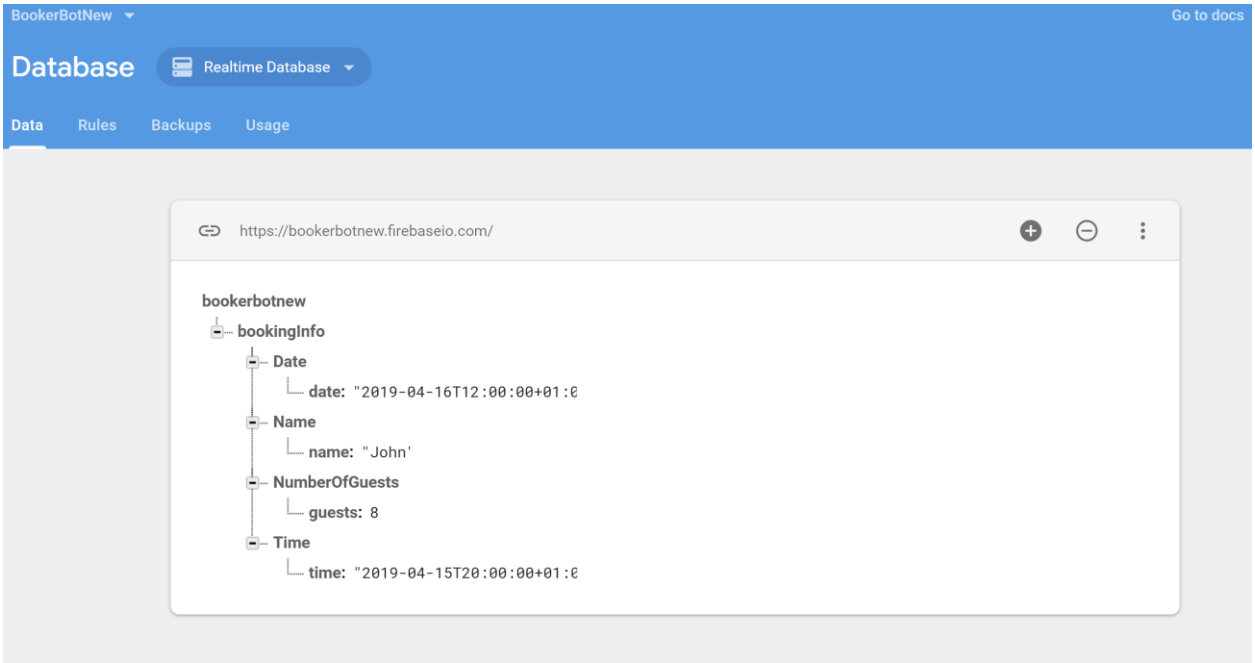


Figure 49: Booking Information Parent/Child Tree structure

Once this new way of writing data to the database was figured out and tested, it was time to learn how to read data back from the database and display it to the user in their conversation with the Chatbot. This functionality was discussed from the beginning of the project with the project panel and was deemed to be an important piece of functionality. This function was not achieved as part of iteration one, but for iteration three it was achieved. Once reading from the database was figured out, implemented and tested it opened it more avenues of functionality that could be added to the final version of the Chatbot. The first way in which this functionality was implemented was in the form of reading a food menu that is stored in the database and displaying it to the user in the Chatbot conversation. This food menu is stored in the database in JSON format and can be seen in figure 50.

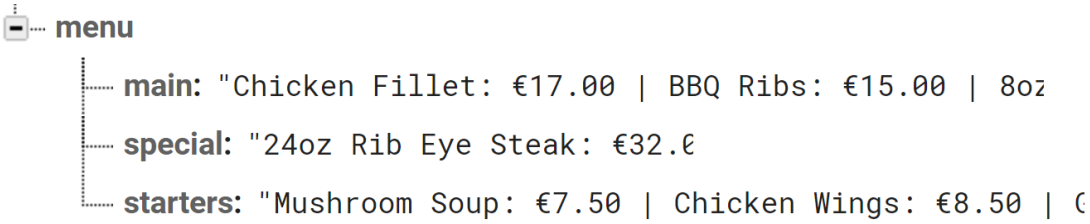


Figure 50: JSON Food menu stored in the database

The fact that this food menu is pre-stored in the database means it can be easily accessed and changed by anyone who has access to the database. The provides the ability for the menu to be updated easily by administrators and then the Chatbot can still read it from the database and display it to the user in the same way. The function that was written to perform this data read can be seen in figure 51.

```
function getMenu(agent) {
  return admin.database().ref('menu').once('value').then((snapshot) => {
  const constVal = 1;
  const starters = snapshot.child('starters').val();
  const main = snapshot.child('main').val();
  const special = snapshot.child('special').val();

  if(constVal !== null){
  agent.add(`No problem. Here is tonight's menu:`);
  agent.add(`Starters: ${starters}.`);
  agent.add(`Main: ${main}.`);
  agent.add(`Tonight's Special is: ${special}.`);
  agent.add(`If you would like to place your food order now just type "order food", and we can create your order.`);
  }
  });
}
```

Figure 51: Reading food menu from database

The database read is achieved by first executing a return statement that returns the desired parent node from the database. Then, variables that store a snapshot of each child node of that parent are created and these snapshots can then be printed in an ‘agent.add’ statement using the variable names. The print out statements added in the ‘getMenu’ function in figure 51 result in the entire menu being displayed to the user which is shown in figure 52.

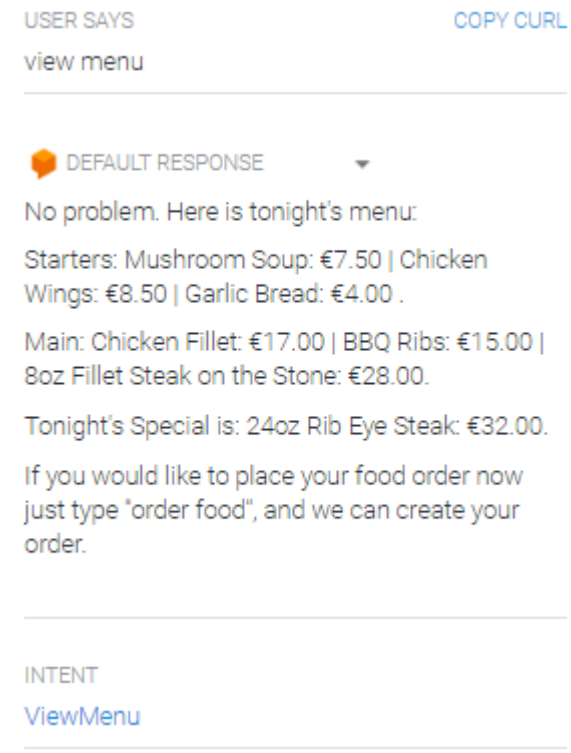


Figure 52: View menu output to the user

The second way in which reading from the database was used was in a function called 'reviewBooking' which can be seen in figure 53.

```
function reviewBooking(agent){
return admin.database().ref('bookingInfo').once('value').then((snapshot) => {
  if(snapshot.exists()){

  const name = snapshot.child('/Name/name').val();
  const date = snapshot.child('/Date/date').val();
  const outputDate = new Date(date);
  const time = snapshot.child('/Time/time').val();
  const outputTime = new Date(time);
  const guests = snapshot.child('/NumberOfGuests/guests').val();

  agent.add(`Here is your booking information:`);
  agent.add(`Name: ${name}`);
  agent.add(`Date: ` + (outputDate.getDate() < 10 ? '0' : '') + (outputDate.getDate()) + "/" + (outputDate.getMonth() < 10 ? '0' : '') + (outputDate.getMonth() + 1) + "/" + outputDate.getFullYear() );
  agent.add(`Time: ` +  (outputTime.getHours() + 1)  + ":" + (outputTime.getMinutes() < 10 ? '0' : '') + outputTime.getMinutes());
  agent.add(`Number of Guests: ${guests}`);
  agent.add(`If you would like to delete this booking just type "delete booking". If not we will see you at the restaurant.`);
  agent.add(`You also have the option to place your food order now. Just type "place order" and we will take you through the process.`);

  } else {
  agent.add(`Sorry. We do not have your booking on record. If you would like to book a table just type "book table" and we will take you through the process.`);
    }
  });
}
```

Figure 53: Review Booking function

This function allows users to see their entire booking information that the Chatbot has captured and stored in the database after the user has completed the table booking process. An example of this can be seen in figure 54. This function is triggered by the user typing something that resembles 'review booking'. The function then works by, first of all, using a new method that

was learned throughout the development of iteration three called '.exists'. The function performs a read of the database looking for a specified node in a specified location and checks if it exists, in this case, it is looking for 'bookingInfo'. If the read is successful and 'bookingInfo' is found to exist then it reads each attribute, stores them in their own variables respectively and then prints them to the user. Alternatively, if 'bookingInfo' is found to not exist then it prints an alternative message to the user that says, "Sorry, we do not have your booking on record. If you would like to book a table just type "book table" and we will take you through the process."

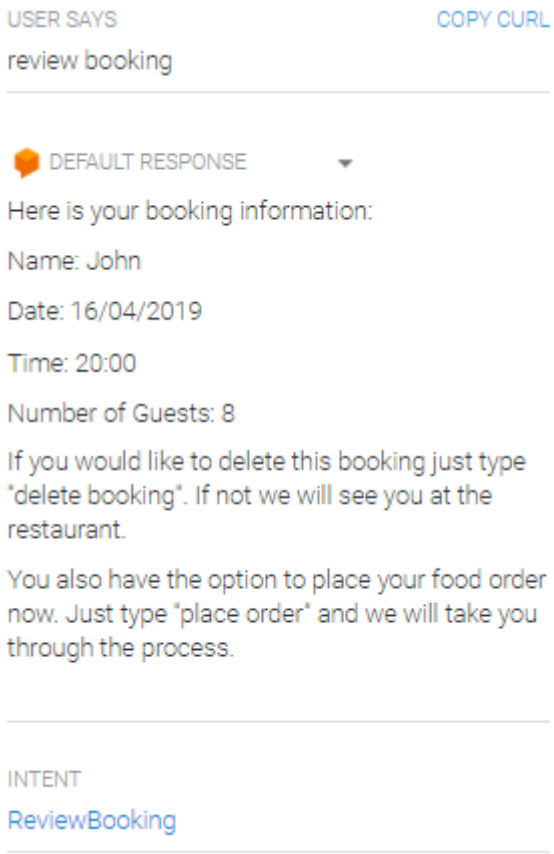


Figure 54: Review Booking output

The output from the 'Review Booking' intent shows that there are multiple other options available to the user that were not available in iteration one. One of these options allows the user to delete their booking. This is another function that took a substantial amount of time to implement during iteration three due to its complexity. The reason that it is complex relates to another new feature of iteration three that was not included in iteration one and that is having

a limited number of seats available in the restaurant when the conversation is first initiated. This can be seen in figure 55.

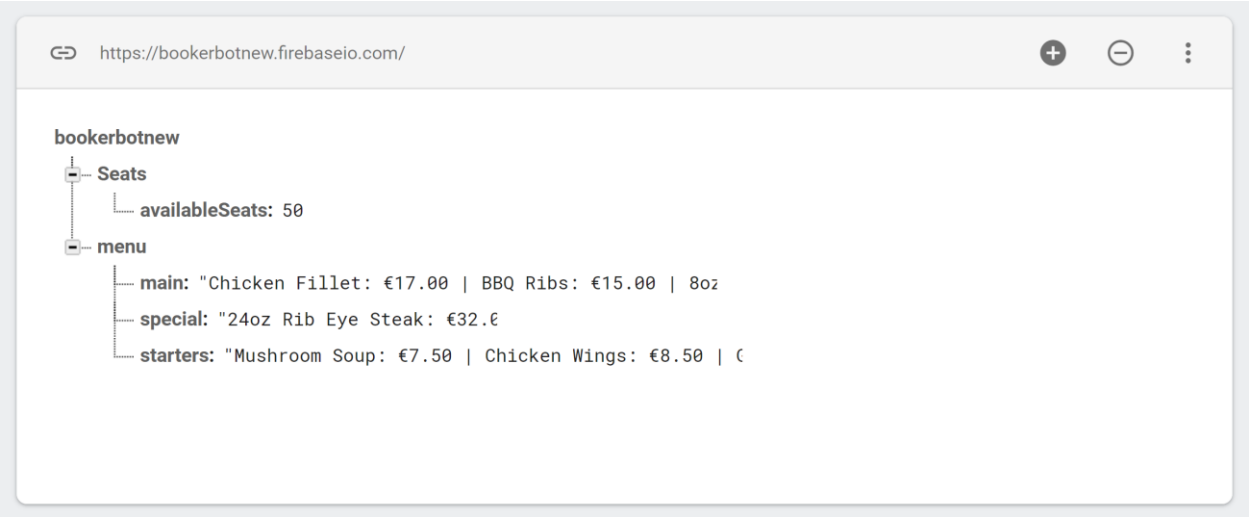


Figure 55: The initial state of the database that includes the food menu and 50 available seats

When the user goes through the process of booking a table, the Chatbot collects and stores their booking information which includes their name, their date of choice, time of choice and the number of guests they wish to book for. When the Chatbot receives the number of guests that the user wishes to book for, a calculation is performed within the function ‘getNumGuests’. This calculation removes the value given by the user from the original value of 50 seats that are available. This function can be seen in figure 56.

```
function getNumGuests(agent) {
  const numGuestsParam = agent.parameters.guests;
  const numGuests = numGuestsParam;

  if (numGuests < 1){
      agent.add('You need to reserve a table for at least one person. Please try again! How many guests would you like to book for?');

  } else if (numGuests > 15) {
    agent.add('You cannot make a reservation for more than 15 people. Please try again! How many guests would you like to book for?');

  } else {
    agent.add(`Perfect, table for ` + numGuests + ` it is. What date would you like to make your reservation for?`);
      admin.database().ref('Seats').once('value').then((snapshot) => {
        let availableSeats = snapshot.child('availableSeats').val();
        var unavailableSeats = admin.database().ref("Seats/unavailableSeats");
        unavailableSeats.set(numGuests);

        admin.database().ref('Seats').once('value').then((snapshot) => {
        let availableSeats = snapshot.child('availableSeats').val();
        var bookingInfo = admin.database().ref("bookingInfo/");
        bookingInfo.child("NumberOfGuests").set({guests:numGuests});
        availableSeats = availableSeats - numGuests;
        return admin.database().ref('/Seats/availableSeats').set(availableSeats).then((snapshot) => {
          });
          });
      });
    }
}
```

Figure 56: 'getNumGuests' function

The calculation that is performed within the function has multiple steps. First of all, there are a few simple constraints on booking for certain amounts of guests. The Chatbot will not allow reservations for more than 15 people to be made and also a reservation must be for at least 1 person. If the number of guests supplied by the user meets the criteria set out by the function, it will, first of all, create a new child node of 'Seats' in the database called 'unavailableSeats' and will then write the number of guests supplied by the user to that node. It will then read the value of availableSeats, store it in a variable and perform a calculation that subtracts the value that the user has supplied from the value of availableSeats. The result of this can be seen in figure 57.

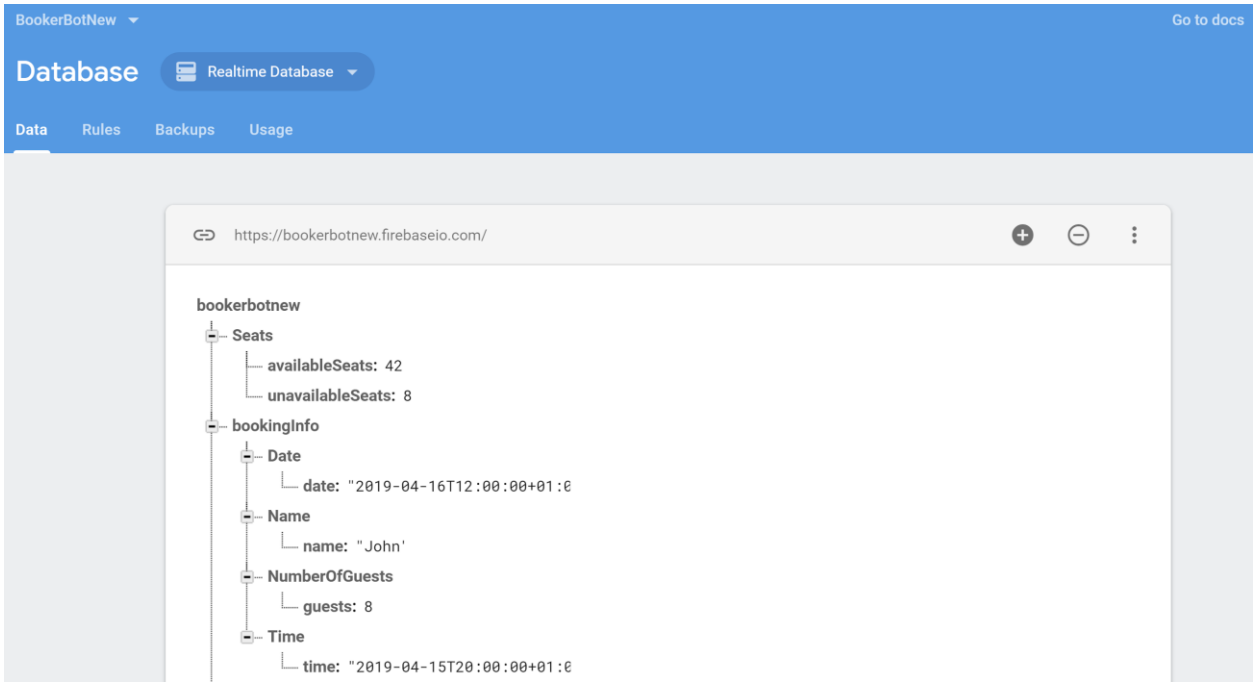


Figure 57: AvailableSeats is reduced by the number of guests provided by the user

The value that is stored in 'unavailableSeats' was vital to the creation and execution of the 'deleteBooking' function. The 'deleteBooking' function is complex in multiple ways. First of all, the function is triggered by the user typing 'delete booking'. However, if the user says, 'delete booking' and no booking exists in the database to be deleted then it will display a message saying, "Sorry, we do not have your booking on record. If you would like to book a table just type "book table" and we will take you through the process.". If the check reveals that 'bookingInfo' does exist in the database, then the intent deleteBooking will give the user the choice between if they are sure they want to delete the booking or not.

```
function deleteBooking(agent){
  return admin.database().ref('/bookingInfo').once("value", snapshot => {
   if (snapshot.exists()) {
    agent.add(`Are you sure you want to delete your booking? This cannot be undone and you will have to repeat the booking process to book another table.`);
   } else {
    agent.add(`Sorry. We do not have your booking on record. If you would like to book a table just say "book table" and we will take you through the process.`);
   }
});
}
```

Figure 58: Delete Booking function

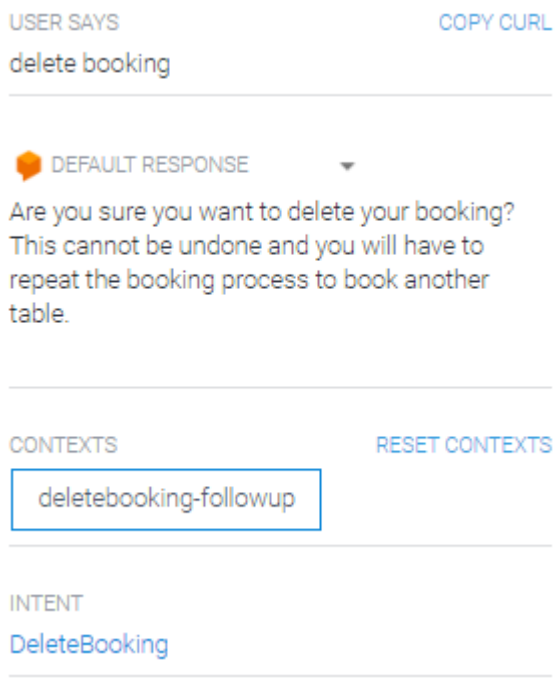


Figure 59: Delete Booking intent output

This is achieved through the use of multiple functions under the umbrella function that is ‘deleteBooking’ and also ‘follow-up intents’ that are connected to the main intent ‘deleteBooking’. The follow-up intents connected to ‘deleteBooking’ and entitled ‘deleteBooking - yes’ and ‘deleteBooking - no’. These follow-up intents act as prompts to the user asking them if they are sure they want to delete, and they are only triggered if ‘bookingInfo’ is found to exist in the database in the first place. If the user says, ‘no’, i.e. they don’t want to delete their booking, then a message is displayed that says, “No problem, we will keep your reservation”. This can be seen in figure 60.

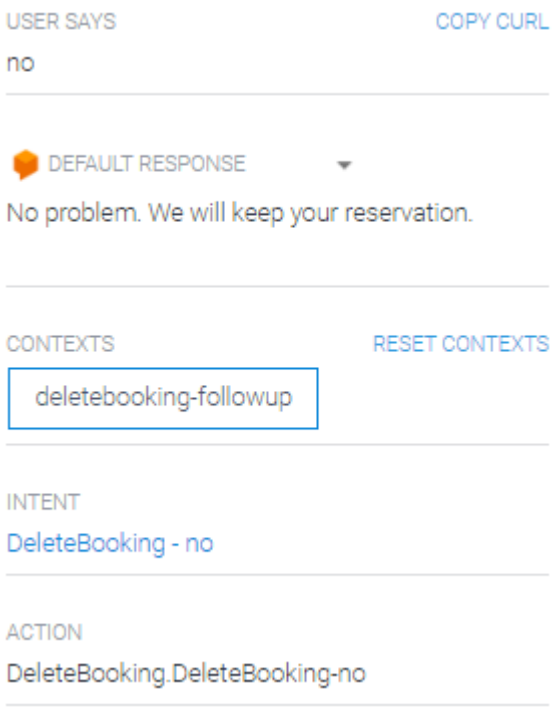


Figure 60: User says 'no' to are you sure you want to delete your booking

However, if the user says 'yes', i.e. they do wish to delete their booking, then another function is triggered in the background. This function is entitled deleteBookingYes and can be seen in figure 61.

```
function deleteBookingYes(agent){
  admin.database().ref('Seats').once('value').then((snapshot) => {
  var unavailableSeats = snapshot.child('unavailableSeats').val();
  var availableSeats = snapshot.child('availableSeats').val();
  let updateAvailableSeats = availableSeats + unavailableSeats;
  admin.database().ref('/Seats/availableSeats').set(updateAvailableSeats);
  admin.database().ref('/Seats/unavailableSeats').set(unavailableSeats-unavailableSeats);
    });

     return admin.database().ref('/bookingInfo').once("value", snapshot => {
      if (snapshot.exists()) {
    var bookingInfo = admin.database().ref('/bookingInfo');
    bookingInfo.remove();
        agent.add(`No problem. Your booking has been deleted. If you would like to book another table just type "book table", or you can view our opening hours or place your food order.`);
      } else {
      agent.add(`Sorry. We do not have your booking on record. If you would like to book a table just say "book table" and we will take you through the process.`);
      }
    });
  }
```

Figure 61: DeleteBooking - Yes function

The purpose of this function is to primarily locate the user's booking information in the database and delete it, but it also serves a second important purpose. It re-calculates the seats in the restaurant that are both available and unavailable upon the deletion of a booking. First of all, it will update the value of 'availableSeats' to add back whatever the value was of

unavailable seats. Then it will update the unavailable seats to subtract the value of the number of guests the user had supplied. The node ‘unavailableSeats’ plays a major role in this process. The reason it is needed is that the value ‘numGuests’ (the number of guests the user wishes to book for which is supplied by the user) gets deleted and removed from the database once the user says ‘yes’ to the question, “are you sure you want to delete your booking”. Therefore, it cannot be used to recalculate the ‘availableSeats’ in the database. This piece of functionality was quite challenging and took a substantial amount of time to implement correctly.

Another new piece of functionality included in iteration three is the ability for users to place an order for their food after view the menu. This function is broken into sections and makes use of both reading from and writing to the database. The first intent ‘CreateOrder’ is triggered when a user says, ‘place order’. This then presents the user with the first section of the process which is choosing what they would like to order for their starter, as seen in figure 62.

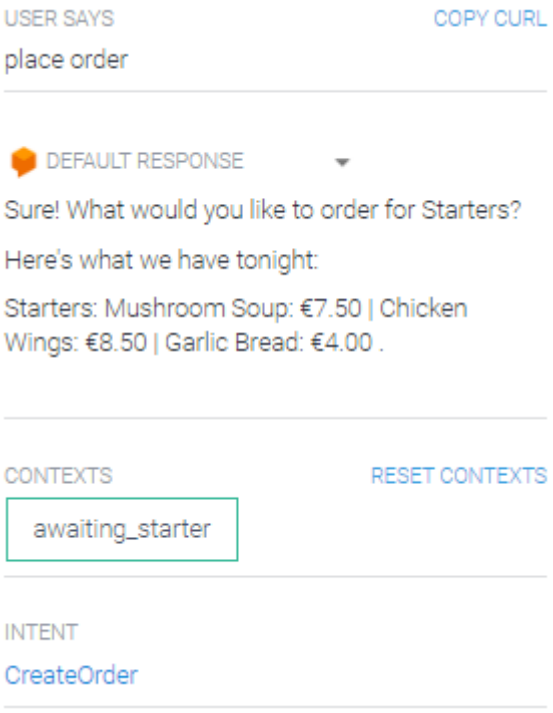


Figure 62: Create Order intent triggered - ordering process begins

This is achieved through a combination of reading a portion of the menu, the starters, that is stored in the database and displaying it to the user and also then expecting them to input their choice. Once they have input their choice it is written to the database and then read back from

the database to display it to them in the next message in the food ordering process. This function can be seen in figure 63.

```
function createOrder(agent){
  agent.add(`Sure! What would you like to order for Starters?`);
  return admin.database().ref('menu').once('value').then((snapshot) => {
  const constVal = 1;
  const starters = snapshot.child('starters').val();
  if(constVal !== null){
    agent.add(`Here's what we have tonight:`);
    agent.add(`Starters: ${starters}.`);
   }
   });
  }
```

Figure 63: Create order function

The next message in the food ordering process consists of a pleasant message saying ‘Lovely, (starter choice) it is.’ The message then proceeds to read the main course and special options off the menu from the database and display it to the user and also asks them what they would like to order for their main course. This message in the process can be seen in figure 64.

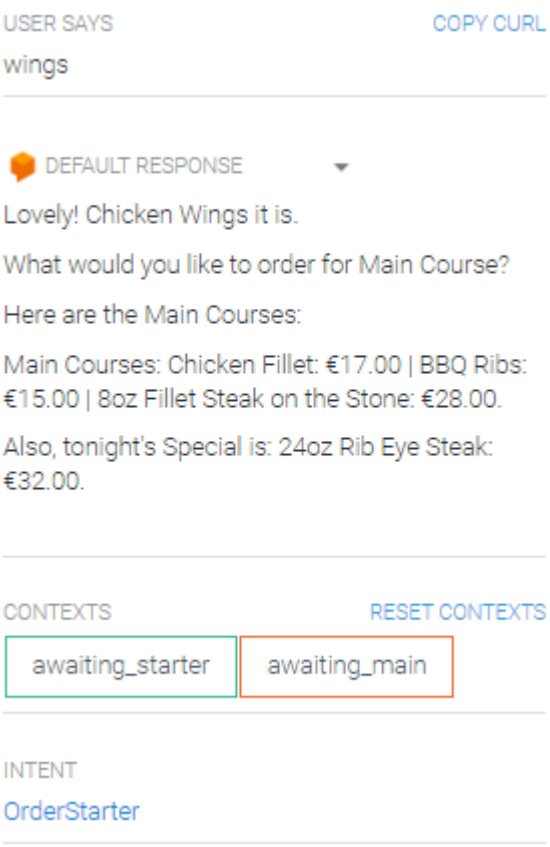


Figure 64: Displaying what the user has ordered for starter, displaying the main courses and waiting for a response

The user then enters their choice for their main course and that is interpreted by the bot and written to the database. In this case of this iteration, each menu option was configured in Dialogflow as a custom developer entity. “Developer entities can and should be used for anything that's not covered by system entities and should contain values you expect from your users.” (Dialogflow, 2019). This worked well in the case of this iteration as it allows the creation of synonyms for the entities, meaning the user’s input doesn’t have to exactly match the entity but their input can still get mapped to that entity if enough synonyms are defined. “These entities allow for the mapping of synonyms to a reference value. For example, a food type entity could have an entry with a reference value of "vegetarian" with synonyms of "veg" and "veggie".” (Dialogflow, 2019). In this case of this iteration, it allows the user to type in a response such as, ‘the special’ and then the bot recognises this response and maps it to whatever ‘the special’ is configured as. This configuration can be seen in figure 65. However, this solution may not be scalable for hypothetical future iterations of this Chatbot and was merely used as a demonstration of making use of Dialogflow’s built-in functionality.

Figure 65: Main course developer entities

Once both the starter and the main course have been chosen by the user, the final message of the food ordering process is displayed to them. The plan for this message was to tell the user that their order has been saved and then display their order details back to them after reading them from the database. This can be seen in figure 66.

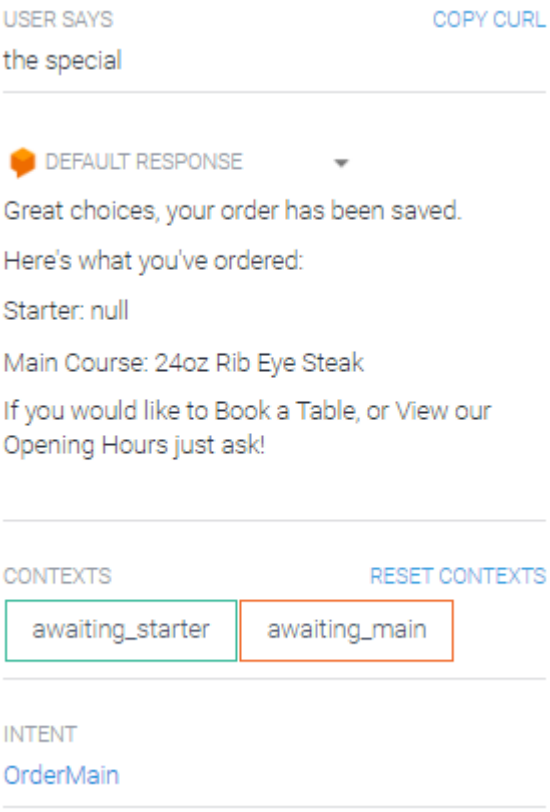


Figure 66: Final message of the food ordering process

However, this became one of the challenges faced during the process of developing iteration three and still remains as a known bug in the system. For some reason, unknown, the function that saves the user's starter choice to the database, seen in figure 63, contains an anomaly that could not be figured out in the time frame for iteration three. When their starter choice is written to the database, an extra child node is also created, under which their starter choice is stored. This can be seen in figure 67.

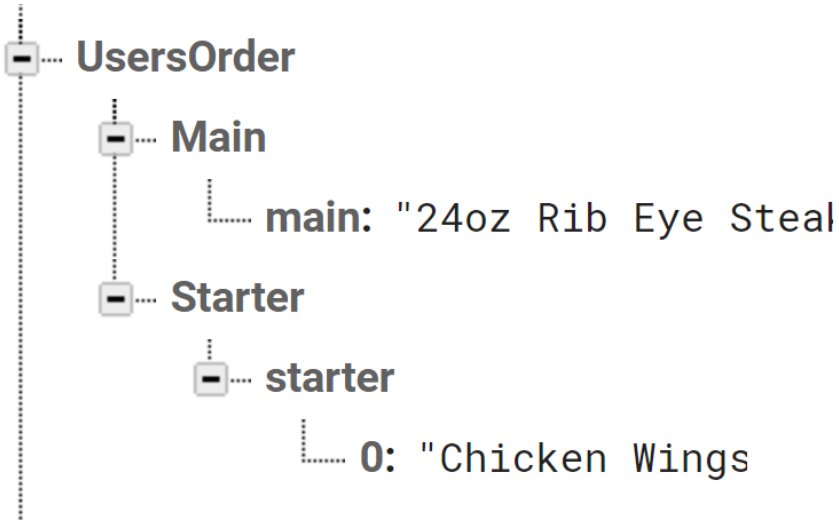


Figure 67: 'Child 0' Anomaly

This extra child node is defined as a 0 and it is unclear why. However, this causes an issue when attempting to read the desired value, which is the user's choice, back from the database and display it to the user. The print out statement results in a null value, as seen in figure 66. A considerable, but managed, amount of time was dedicated to this issue but unfortunately, no solution was discovered.

Other improvements to the system that were made during iteration three were objectives that were mentioned since the beginning of iteration one. These objectives were somewhat underestimated in terms of difficulty but were eventually overcome. One of these objectives that were tackled and overcome was the formatting of date objects. The goal was to receive user input of their desired booking date, store it in the database and then output a human-readable formatted version of that date. This became quite a challenge from the beginning of the process and almost went unsolved. After carrying out extensive research and many attempts at implementing a working solution, a breakthrough was made in the latter end of development.

It was discovered that Firebase and the Realtime Database being used as the backend for this project does not accept or store JavaScript Date objects, it stores dates as Timestamps. Figure 68 shows the solution that was implemented to overcome this issue.

```
function getDate(agent) {
  const dateInput = agent.parameters.date;
  const bookingDate = new Date (agent.parameters.date);
  const now = new Date();

  if (bookingDate < now){
    agent.add(`Sorry that isn't possible. You must choose a date in the future. Please try again.`);
     agent.add(`What date would you like to book for?`);
  } else {
     agent.add(`Sure, `+ (bookingDate.getDate() < 10 ? '0' : '') + (bookingDate.getDate()) + "/" + (bookingDate.getMonth() < 10 ? '0' : '')
     + (bookingDate.getMonth() + 1) + "/" + bookingDate.getFullYear() + ` it is.
     What time would you like to book your table for? Please enter the time in the format '7pm' for example.`);
     var bookingInfo = admin.database().ref("bookingInfo/");
     bookingInfo.child("Date").set({date:dateInput});
  }
```

Figure 68: Writing Date to database and formatting date for output to user

The solution that was implemented involved created different variables to perform different tasks with. A variable called ‘dateInput’ was used to take in the user’s date of choice. This variable was then also used to write to the database. Then, a second variable was created called ‘bookingDate’ which took the user’s input date and stored it as a JavaScript Date object which could then be manipulated using a variety of JavaScript functions to format in the desired format. A third variable called ‘now’ was also created merely for the purpose of testing whether the user’s input date was in the past or in the future, not allowing them to book for a date in the past. The three functions that were used to format the date were ‘getDate()’, ‘getMonth()’ and ‘getFullYear’. Other aspects of the date formatting include a ternary operator which is an inline if statement that tests if the value received from getDate and getMonth is less than 10 and if it is, print an additional zero. Also, ‘+1’ was added to getMonth as the months returned from that statement start at zero, like an array. An example of the final result of this can be seen in figure 69.

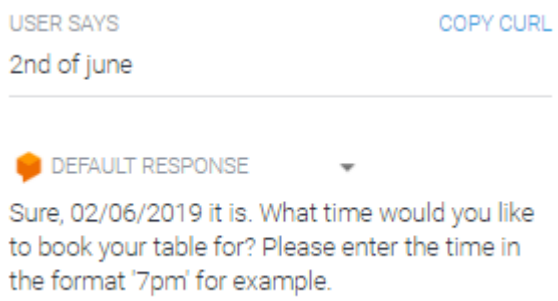


Figure 69: Final formatted date output

This breakthrough allowed for other pieces of functionality to also be implemented, such as adding time constraints to the booking times of restaurant, i.e. only being able to make a reservation between certain times, and also formatting the time in a human-readable way and displaying it to the user. This is achieved in a similar way to saving and formatting the date and the function can be seen in figure 70.

```
function getTime(agent) {
  const timeInput = agent.parameters.time;
  const time = new Date (agent.parameters.time);
  const bookingHours = time.getHours();

  if (bookingHours < 16) {
    agent.add(`Sorry we are not open at that time. You can only book between 5pm and 10pm.`);
    agent.add(`What time would you like to book?`);

  } else if (bookingHours > 21) {
      agent.add(`Sorry we are not open at that time. You can only book between 5pm and 10pm.`);
      agent.add(`What time would you like to book?`);

  } else {
    agent.add(`No problem. ` + (time.getHours() + 1)  + ":" + (time.getMinutes() < 10 ? '0' : '') + time.getMinutes() + ` it is.
     We look forward to seeing you.`);
    agent.add(`To review your booking information just type "review booking".`);
    var bookingInfo = admin.database().ref("bookingInfo/");
    bookingInfo.child("Time").set({time:timeInput});
  }
}
```

Figure 70: GetTime Function: Writing Time to database and formatting date for output to user

Another challenge that was faced was when the function ‘getHours’ was used, the time value for the hour that was returned was an hour behind the local time of Dublin, Ireland. That is why in the function, the if statements test to see if the time entered by the user is less than 16 (16:00/4pm) or greater than 21 (21:00/9pm). Even though the Chatbot informs the user that the operating hours of the restaurant are between 5pm (17:00) and 10pm (22:00).

Another aspect of formatting that was added to the Chatbot is used in the GetName function. Throughout the duration of the development and testing phase of iteration three, the GetName intent and function were altered quite a bit. Going from one way of capturing the user’s name, to another way and back again. This was due to user testing and the reactions from users that tested the system. To capture names, Dialogflow provides a system entity called @sys.given-name that can be used within any intent to capture just the first name of person. It consists of a large list of names but is limited to just common, mainly English, names. While this entity was being used, it captured common, like ‘John’ or ‘Dan’ names nicely and it even formats the name with a capitalised first letter automatically when displaying it back to the user and when storing it in the database. However, an issue was quickly discovered during some user testing

sessions when the names of the testers were entered, and they were not actually recognised by the system. What was also interesting to witness during user testing was although the Chatbots message told the user that 'we only need your first name', users still entered in their first name and surname regardless. This led to the change of not using the system entity @sys.given-name and instead using the system entity @sys.any. This system entity accepts any given string given by the user and will accept that as their name. It is understood that this is not an ideal solution, but this solution worked well enough to achieve what was desired for this iteration. If future iterations of the project were developed, a sounder solution would be explored. The issue was that the system entity @sys.any does not perform any automatic formatting to the name given by the user so for that reason it was implemented in the fulfilment code of the Chatbot. This can be seen in figure 71.

```
function getName(agent) {
  const nameParam = agent.parameters.name;
  const name = nameParam;
  const capitalisedName = name.charAt(0).toUpperCase() + name.slice(1);
  agent.add(`Thanks ` + capitalisedName + `! How many guests would you like to make your reservation for?`);
  var bookingInfo = admin.database().ref("bookingInfo/");
  bookingInfo.child("Name").set({name:capitalisedName});
```

Figure 71: GetName function - capitalising first character of user's name

A variable called 'capitalisedName' is created which is given the value of 'name' which is the name given by the user. The first letter of the name is that capitalised through the use of 'name.charAt(0).toUpperCase() + name.slice(1)'.

7.4 Front-End

In the original proposal for this project, it was put forward that a Bootstrap front-end would be created during iteration three and the final Chatbot system would be integrated into that front-end website. However, that idea was recommended against shortly after the proposal and the core back-end functionality of the Chatbot system was deemed a more important task to focus on. Yet, once all of the major functionality had been implemented and tested at the end of iteration three, a resource was found online that was found to be extremely interesting and brought an extra dynamic and feeling of completion to the final version of the system.

This resource is called "Dialogflow for Web v2" and was created by a developer called Mikhail Ushakov who made this resource free and open source for other developers to access from

GitHub. This open source application was “built to support rich responses and to make the most out of the Platform available to the Web.” (Ushakov, 2019). Features of this web application include:

- “Progressive Web App (100/100 Lighthouse score)
- Accessibility Features
- Wide Browser Support (IE8+), offline capabilities (history) and great SEO
- Familiar UI & UX, based on official Google Assistant Guidelines
- Hands-free interaction with Voice Input and Speech Feedback
- Language Independency
- Docker and Kubernetes support
- Rich-component, Webhook and Actions on Google Support
- Iframe Support
- Based on Vue, Webpack 4, Babel 7
- Lightweight (with ngx_pagespeed the build is <100KB)
- Free and Documented (documentation coverage at least 80%)
- Free hosted version on Dialogflow Gateway”

(Ushakov, 2019)

There were only a few steps required to have a fully functioning, user friendly, web front end connected to my Dialogflow Chatbot system utilising this open source application. The first step involved signing into what is called the Dialogflow Gateway. “Dialogflow Gateway is a cloud-based service, which connects Dialogflow V2 Agents to the World Wide Web. Dialogflow for Web v2 requires Dialogflow Gateway for its formatting option and to make secure and authenticated requests to Dialogflow V2 API.” (Ushakov, 2019). Once the Dialogflow Gateway had been set up, a console screen was gained access to which displayed all Dialogflow agents that had been created by me. This can be seen in figure 72.

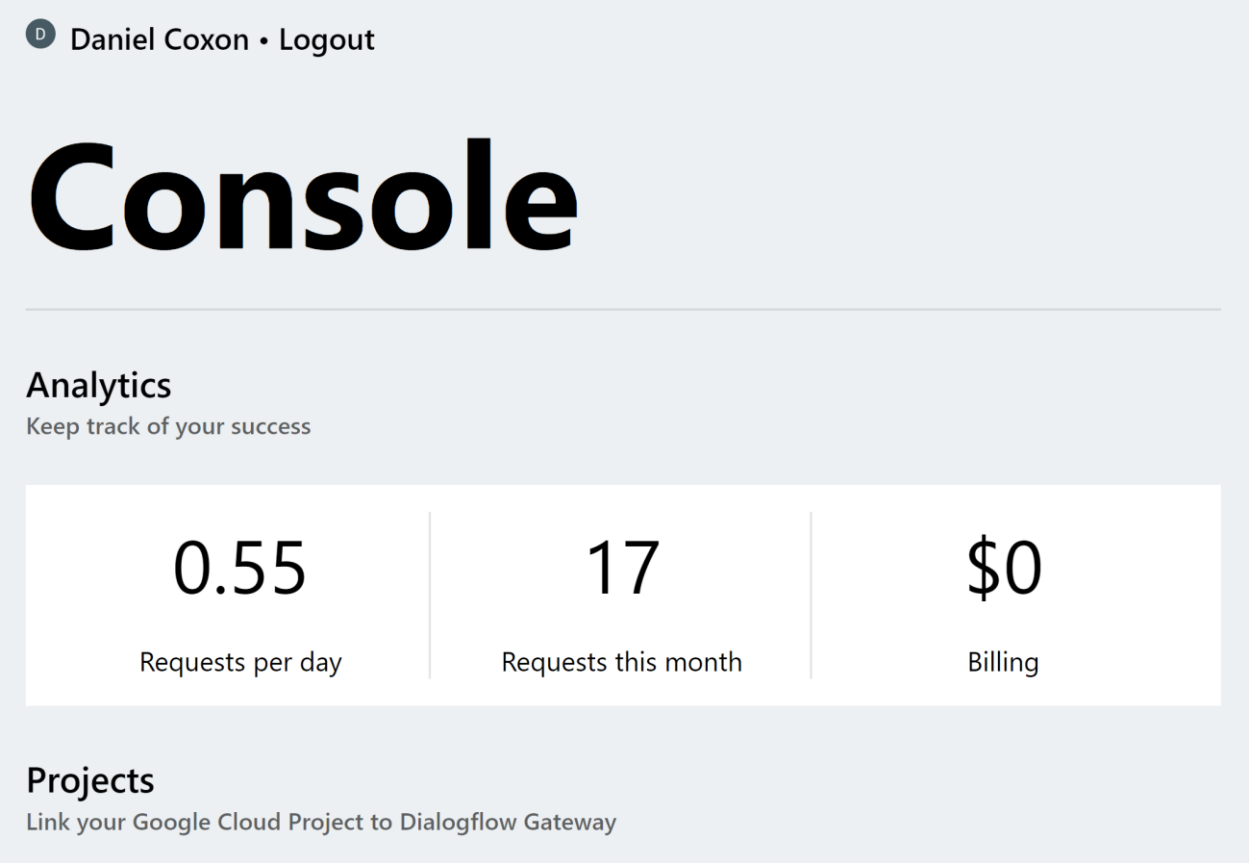


Figure 72: Dialogflow Gateway Console

The next step was to then click the button that said ‘link’ next to the desired project, as seen in figure 73. This then turned the link icon green and gave an option to manage this project.

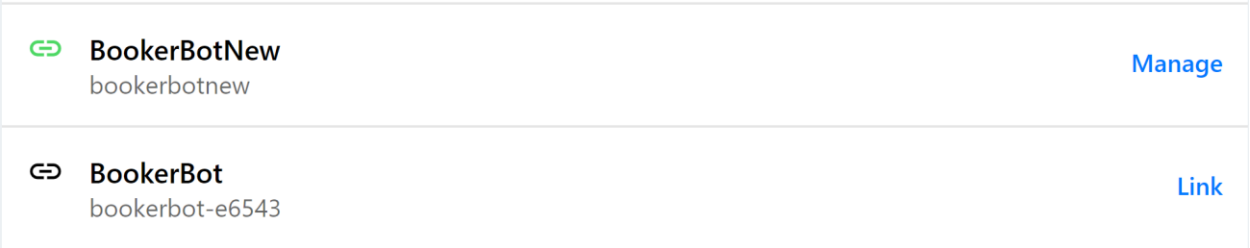


Figure 73: Linking project in Dialogflow Gateway

Clicking the ‘manage’ button then displayed an overview screen that displayed information such as a ‘Gateway URL’ and a ‘UI URL’. This can be seen in figure 74.

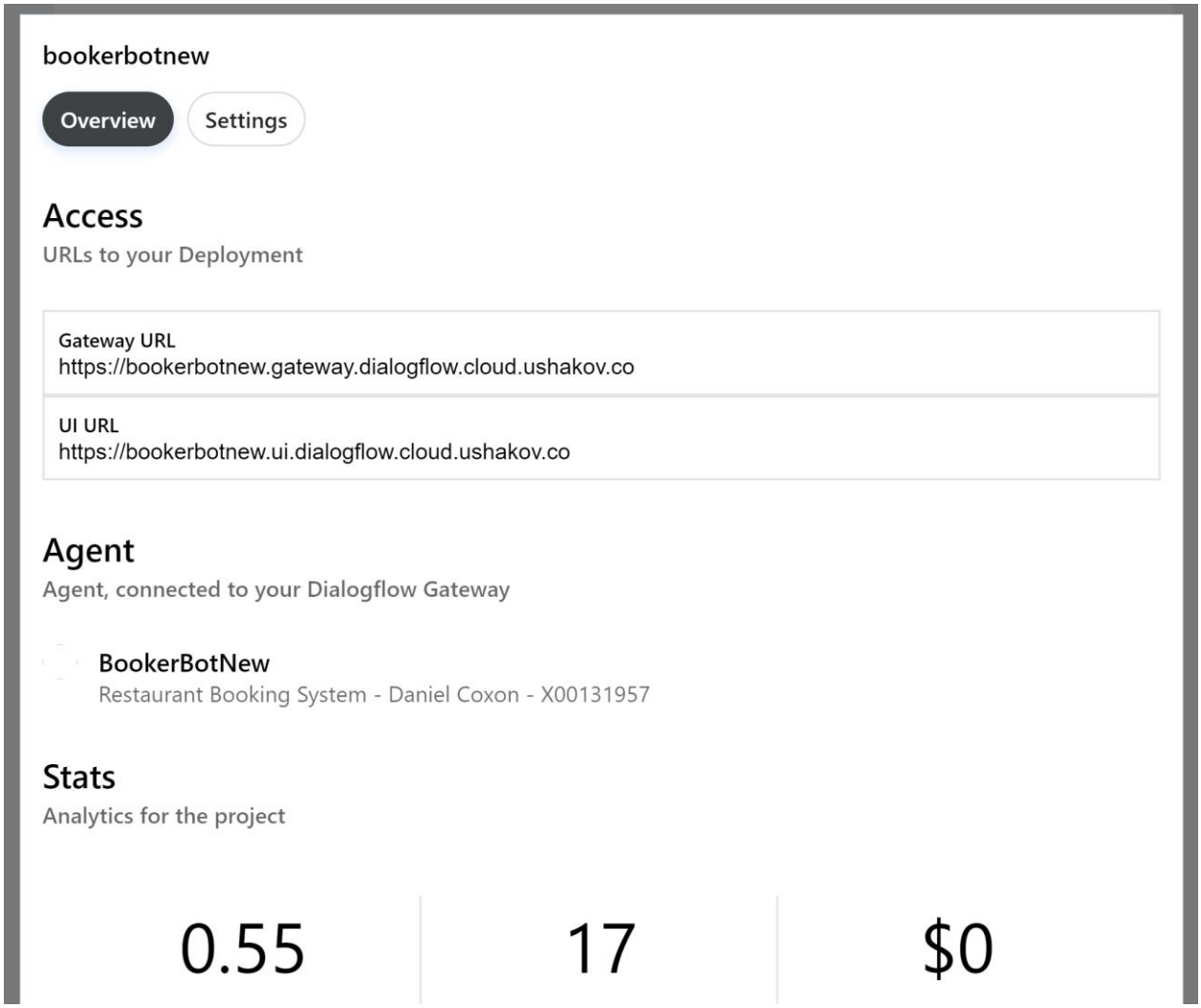


Figure 74: Dialogflow Gateway Project Overview Screen

The ‘UI URL’ could then be used to access the user interface, front end, fully functioning version of my Chatbot system ‘BookerBot’. This can be seen in figure 75.

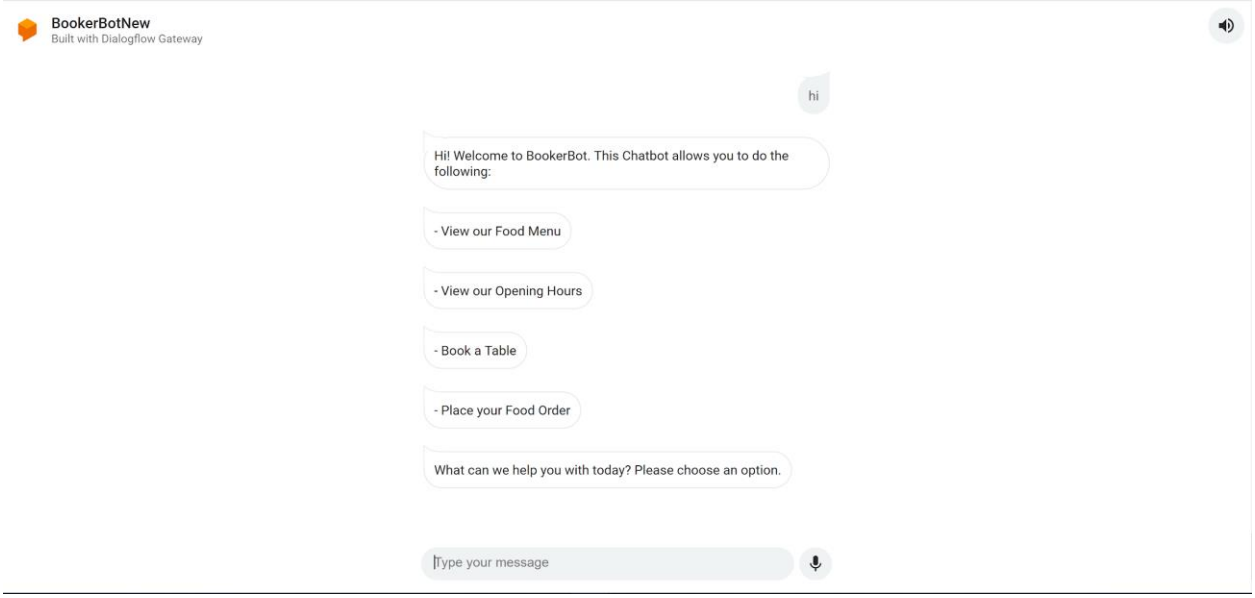


Figure 75: Fully functioning Web Front End User Interface

This user interface could also be accessed on mobile through the use of the same link. User testing was even carried out through the use of this front-end. The UI URL was sent to a person chosen to carry out user testing, they interacted with the Chatbot, went through the process of viewing the menu, ordering food, viewing the opening hours and booking a table. All of the information that they entered was then populating the backend database on my machine in real time. This was found to be a much better way to interact with the Chatbot compared to the test environment within the Dialogflow console and felt like a substantial achievement and a nice way to end the development of the final iteration of my 4th year project.

7.5 Challenges Faced

Apart from the challenge of designing, developing and implementing each function and intent of the Chatbot in a fully working way, the other main challenge of the development process was adding the functionality for catching wrong or erroneous inputs from the user. The Default Fallback intent can be used for responding to most foreign or unrecognized inputs from the user, “This intent is matched when your user's input does not match any other intent; in other words, it's a kind of catch-all for any unrecognized user input.” (Dialogflow, 2019). However, the challenge arose when trying to catch fallbacks during a process flow that uses input and output contexts, like booking a table, and re-prompting the user for their input and keeping the process on track. This is where the creation of custom context based fallback intents and context

lifespans were learned about and implemented. It was understood that the Chatbot needed to be able fallback and re-prompt the user for their input for two types of scenarios. The first scenario was after an if statement was executed in a function in the fulfilment. The second was if the user entered an invalid data type during a prompt, for example if they entered an erroneous word or data type like 'truck' when asked what date they would like to book for. For this to work, context lifespans and custom fallback intents needed to be set and managed. Custom fallback intents were added to the intents 'GetDate', 'GetTime' and 'GetNumGuests'. These fallback intents were to be triggered each time the Chatbot received an unexpected input from the user after a prompt. This fallback intent would then display a message that gave the user more information as to what the Chatbot was expecting to receive to help guide them in the direction of a successful execution of the process or function. An example of this can be seen in figure 76.

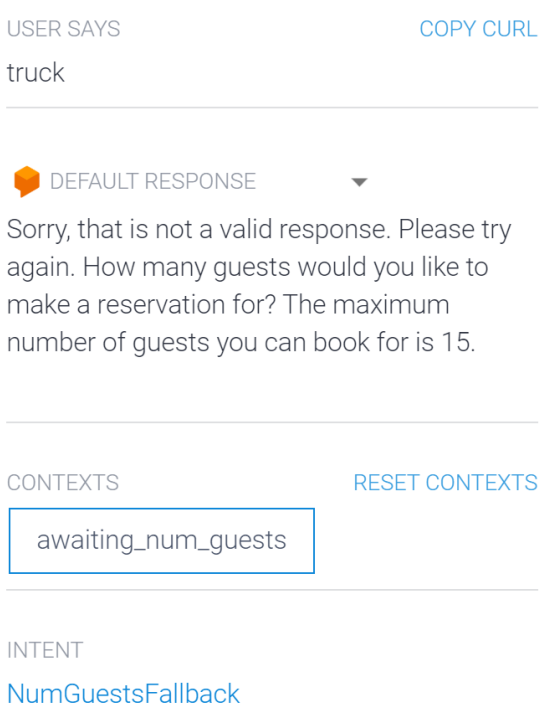


Figure 76: Custom fallback triggered for GetNumGuests

As figure 76 shows, the Chatbot lets the user know that 'truck' is not a valid input. It also lets them know that they cannot make a reservation for more than 15 people the help them for their next input. If, however their input does not meet these requirements with the next input, they are prompted again, however, this time the prompt does not come from the Custom Fallback

intent but from the if statement that has been coded in the fulfilment. This can be seen in figure 77.

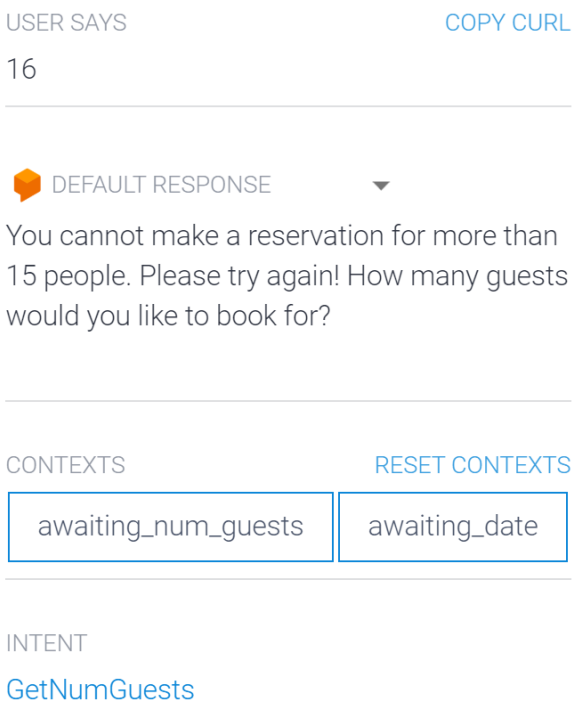


Figure 77: Fallback message from fulfilment

This is where context lifespans become important. As seen in both figure 76 and 77, the context that is still active is 'awaiting_num_guests' indicated by the blue bordered box. After the prompts and guidance provided to the user from the Chatbot, it is hoped that they will now input a correct response, otherwise it will assume that they are not quite sure what they are doing and the context will become inactive and the process will be cancelled. The challenge was in figuring out how much guidance and how many attempts should be allowed for the user to get it wrong as if the context lifespan is left active for too long it can have an overflow effect into the next intent and cause buggy like actions to occur. This was found to be the case during development and testing. For example, an earlier version of the Chatbot used to ask the user what time they would like to book a table for, followed by how many guests they would like to book for. But, as they lifespan of the 'getTime' context was still active, if the user typed in the number of guests they wished to book for by using a single digit, e.g. 11, then the Chatbot would actually respond with, "Sorry we are not open at that time" as it was still recognising

numbers input as times. This could be rectified by the user typing the word ‘guests’ or ‘people’ after the number, but this was not an ideal solution. Now the Chatbot is set deactivate the previous context of an intent if it is matched correctly to avoid this confusion and is also set to give the users 3 chances in total to input a correct and accepted response. This was quite a challenging part of the development and testing process and if future iterations were planned, this would be an area that would definitely be researched more and improved to guarantee a flawless user experience.

Section 8 – Final Project Report

8.1 Background of Project Choice & Finalisation

At the beginning of this project, there were multiple ideas and areas of computing that were considered to carry out the project in. Some of these ideas included a project in website development, a project in the area of IoT and also a project in the area Chatbot development. After meeting with the project panel and discussing these ideas, the idea to focus on the area of Chatbot development was finalised. One of the main reasons this area of research and development was decided on as a 4th Year Project was that it was an area that was found to be very interesting and an area that had not been explored or learned about before. This area was also come into contact with during my 3rd Year Work placement as a topic of conversation as a Chatbot was being proposed for the build of a client's website at the time. This conversation about Chatbots and their potential uses and integrations with websites sparked my interest in researching and developing them myself for the 4th Year Project.

8.2 Comparison of Proposed System versus Designed System

At the beginning of the research phase, anything related to Chatbot development that was discovered while researching was an option and a possibility for the development phase as it was all new to me. Many areas relating to Chatbots were uncovered and learned about during the research phase, including Artificial Intelligence based Chatbots, the benefits of Chatbots to User Experience, the security threats and risks of Chatbots and also Rule Based Chatbots, which is what was eventually developed. Once the research got more in-depth it became clear that a few platforms were standing out as being more popular than others for developing Rule Based Chatbots. This helped to narrow down the specific area in which the project would be further developed later. These platforms were, the Microsoft Bot Framework, Amazon Lex and Dialogflow. Throughout the research phase and into the Technology Assessment phase the focus was primarily on the Microsoft Bot Framework, however, the primary function and use case of the Chatbot was somewhat unclear and undefined. The original idea, before the project proposal, was to build a Chatbot that would replace a ‘Frequently Asked Questions’ (FAQ) section of a website. However, after more research and testing was carried out during the Technology Assessment phase this idea was deemed to not have enough scope or depth for the duration of the entire development phase. Over the Christmas period, extra research was carried out and it was decided that a reservation system Chatbot would be developed for the use case of a fictional restaurant using Goole’s Dialogflow. After this additional research was carried

out over the Christmas period and a reservation system Chatbot was decided on, the project proposal was written. However, the project proposal, compared to the final iterations that were completed, also differed in a few ways. In the project proposal, the original idea was to work on one reservation system using Google's Dialogflow for iteration one and two. Iteration one would implement the core functionality and iteration two would build upon this and improve it. Then, the idea for iteration three was to develop a web front end using Bootstrap and then integrate the Chatbot system into this Bootstrap front end. However, upon meeting with my project supervisor it was advised to not focus on developing a front end for the system but instead focus on the development of the main functionality of the reservation system and to perhaps also have a comparison aspect to the project by developing a similar system on another platform. This, in turn, led to the development of a Chatbot system using Dialogflow for iteration one, then for iteration two a similar system was developed using Amazon Lex and a comparison was carried out in the iteration two project report. Then for iteration three, the original system from iteration one was further developed and implemented more functionality. This turned out to be a much more in-depth project than was first anticipated and the Dialogflow platform was also much more in depth and had a lot more to offer than was first realised during the Technology Assessment phase.

8.3 Detailed Account of Approach Taken and Project Lifecycle

8.3.1 Iteration 1

Iteration 1 of the project focused on implementing the core functionality of the Chatbot system using Dialogflow. This included functionality such as greeting the user and presenting them with the options that were available to them, allowing them to view the restaurant's opening hours, view the menu and to book a table. This was a basic version of this functionality and did not reach the level of dynamism that was achieved in iteration three. The opening hours were did not function as it was possible to book a table for any time. Also, the food menu was hard coded in the Chatbot and was not read from the back-end database. This development phase of the project, however, was still found to be quite challenging due to the combination of learning how to use the platform properly and also developing the system simultaneously. There were times where the development was going well but then there were times where issues were met and were felt to be impassable. This led to researching on the internet for ways in which certain pieces of functionality could be implemented. The main challenges in the beginning related to the interaction between the Chatbot and the Firebase Realtime Database backend. This is what

took up the most amount of time during iteration one as it involved a lot of backend fulfilment coding using JavaScript. The main approach taken to this challenge was mainly researching the internet for similar solutions and then implementing them on a trial and error basis until eventually a solution was figured out and implemented.

8.3.2 Iteration 2

Iteration two of the project used Amazon Lex for the purpose of development. This was done so that a comparison of platforms aspect could be achieved within the project. Amazon Lex was found to be less user friendly to use than its competitor Dialogflow. Also, the way in backend fulfilment is carried out on the Amazon Lex was found to be more complex and more difficult to implement. Amazon Lex uses Lambda functions for its backend fulfilment which is in an entirely different section of AWS than the Lex platform. Dialogflow, however, uses an in-line editor that is accessible through the same User Interface as the configuration of the Chatbot itself. During iteration two, it was also deemed, from personal experience, that there seemed to be less documentation available for learning to develop Chatbots on the Amazon Lex platform and also less forums where people were discussing development methods and strategies for Lex. It was found that there was more online discussion around developing Chatbots for Dialogflow and more helpful resources compared to developing for Amazon Lex. This was one of the main challenges that hindered the development of iteration two. This was overcome by persistence and also trial and error.

8.3.3 Iteration 3

Iteration three of the project is where the most success was found, and where break throughs were made. During this phase, a lot of the core functionality and foundations that were laid in iteration one were further developed and built upon. More ways in which to carry out pieces of functionality were also discovered, including writing to the database. During iteration one, writing to the database was a key piece of functionality that was figured out and implemented, however, iteration three discovered more ways in which this could be achieved and tested each method separately. One of these methods was eventually chosen as it was deemed to be the more efficient way of achieving this functionality. One major break through that was achieved in iteration three was the ability to read data from the database. This included reading JSON data that was pre-stored in the database and also reading back data that was written to from the user's interaction with the Chatbot. This opened up and enabled more pieces of functionality

to be implemented by combining read and writes to and from the database. Including, allowing the user to review their booking information once they had completed the booking process. This was achieved by reading back the information that they had entered, which was stored as data in the database and then displaying it in a way that showed each piece of information at once.

8.4 Conclusions and Recommendations

As a whole, the 4th Year Project demanded a lot of patience and determination throughout the entirety of its duration, from September 2018 until April 2019. It has been a challenging yet enjoyable, mentally testing yet ultimately rewarding experience that will be invaluable moving forward into future careers in the IT industry. However, there are a few areas that would ideally be further developed if more time was to be allocated to working on this project. These areas could potentially be further developed in the future. One of the main areas that would be developed further is the 'available seats' functionality that was implemented to the Chatbot in iteration three. The functionality that was achieved at the end of iteration three regarding the 'available seats' is primarily a proof of concept that implements reading, writing, updating and deleting in the database. This includes having a set number of available seats in the database that is reduced by the number of guests the user supplies the system with. This value is also reset if the user decides to delete their booking. However, this functionality could be fleshed out further by adding more complexity such as available dates and times, types of tables available and time slots available for users to book. This would involve a lot more research as it was simply outside of the scope and allocated time that was available for the development of iteration three. This was because there were still other areas of the Chatbot that needed attention and development. These areas included adding constraints on dates and times and also formatting these dates and times to output in a human readable and user-friendly format. The constraints that were added included not being able to make a booking for a date in the past and also only being able to make a booking between 5pm and 10pm, adhering to the restaurant's opening hours This also led to the concept of catching user errors and giving the user more attempts to input the correct response expected by the Chatbot. As mentioned in the iteration three project report, this is another area that would receive more attention as regards further development in future iterations of this project as it is vital to a perfect user experience. In conclusion, the final version of the system that was achieved meets most, if not all, of the main pieces of functionality that were discussed during each meeting with my project

supervisor throughout the entire duration of semester eight. These meetings were found to be extremely helpful in terms of discussing problem areas of the project and also what areas need more attention or what could be done to overcome an issue. The final version of the system has some minor bugs but are not detrimental to the main functionality of the system. The final outcome of the project is a satisfying version of what was set out to be accomplished and has been an invaluable learning experience.

Appendix

Link to Dialogflow Gateway Web App Front End:
https://bookerbotnew.ui.dialogflow.cloud.ushakov.co/

Link to Screencasts of the Chatbot being used:

https://ittd-my.sharepoint.com/:f:/g/personal/x00131957_computing_stu_it-tallaght_ie/EtVfu8EoPShDvkmLQ2CY_YUBiiVKZi4gGErMHjrdl5V9JA?e=Ol2iR6

Updated Conversation Flow Design for Iteration 3

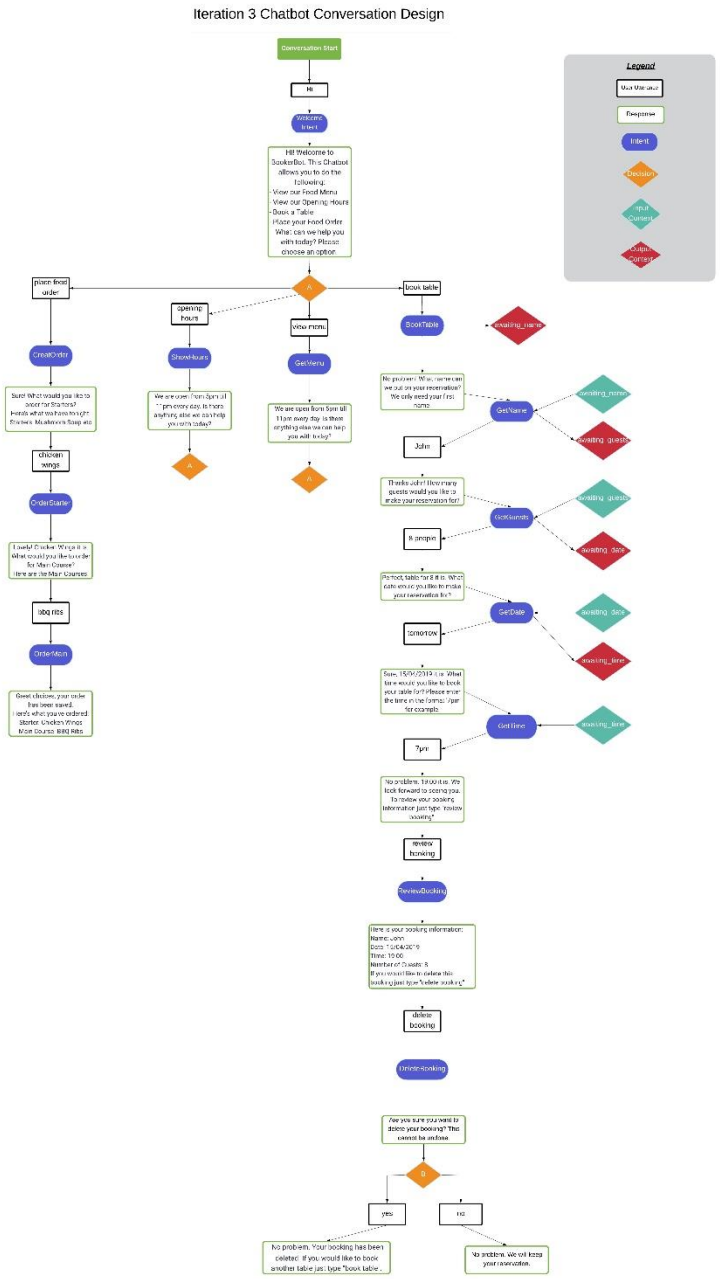


Figure 78: Updated Conversation Flow Diagram

References

Section 1: Initial Research & Feasibility

Ismail, K. (2017). *Top 14 Chatbot Building Platforms of 2017*. [online] CMSWire.com. Available at: https://www.cmswire.com/customer-experience/top-14-chatbot-building-platforms-of-2017/#Converse [Accessed 5 Oct. 2018].

Top Universities. (2016). *5 Trends in Computer Science Research*. [online] Available at: https://www.topuniversities.com/courses/computer-science-information-systems/5-trends-computer-science-research [Accessed 5 Oct. 2018].

Section 2: Research Report

Bayerque, N. (2016). A short history of chatbots and artificial intelligence. [online] VentureBeat. Available at: https://venturebeat.com/2016/08/15/a-short-history-of-chatbots-and-artificial-intelligence/ [Accessed 7 Nov. 2018].

Berry, D. and Azure, W. (2018). What is QnA Maker? - Azure Cognitive Services. [online] Docs.microsoft.com. Available at: https://docs.microsoft.com/en-us/azure/cognitive-services/qnamaker/overview/overview [Accessed 6 Nov. 2018].

Berry, D. and Berry, I. (2018). What is Language Understanding (LUIS) - Azure Cognitive Services. [online] Docs.microsoft.com. Available at: https://docs.microsoft.com/en-us/azure/cognitive-services/luis/what-is-luis [Accessed 6 Nov. 2018].

Bots, P. (2018). Overview — All About Pandorabots. [online] Pandorabots.com. Available at: https://pandorabots.com/static/html/ [Accessed 9 Nov. 2018].

Cheslock, P. (2018). The Top 7 AWS Security Issues: What You Need to Know – Threat Stack. [online] Threatstack.com. Available at: https://www.threatstack.com/blog/what-you-need-to-know-about-the-top-7-aws-security-issues [Accessed 30 Oct. 2018].

Cloud, G. (2018). Dialogflow Enterprise Edition | Dialogflow Enterprise Edition | Google Cloud. [online] Google Cloud. Available at: https://cloud.google.com/dialogflow-enterprise/ [Accessed 9 Nov. 2018].

Drift. (2018). What is Conversational Marketing? – Drift. [online] Available at: https://www.drift.com/conversational-marketing/ [Accessed 24 Oct. 2018].

Elupula, V. (2018). How do chatbots work? An overview of the architecture of a chatbot. [online] Big Data Made Simple. Available at: https://bigdata-madesimple.com/how-do-chatbots-work-an-overview-of-the-architecture-of-a-chatbot/ [Accessed 21 Nov. 2018].

Faggella, D. (2018). 7 Chatbot Use Cases That Actually Work. [online] TechEmergence. Available at: https://www.techemergence.com/7-chatbot-use-cases-that-actually-work/ [Accessed 11 Nov. 2018].

Flow, D. (2018). Dialogflow. [online] Dialogflow. Available at: https://dialogflow.com/ [Accessed 9 Nov. 2018].

Gelston, L. (2016). Building a ChatBot, Pt. 2: Building a Conversational Tensorflow Model. [online] Laura Gelston. Available at: https://lauragelston.ghost.io/speakeasy-pt2/ [Accessed 23 Oct. 2018].

González, R. (2018). Why chatbots will never be popular – UX Planet. [online] UX Planet. Available at: https://uxplanet.org/why-chatbots-will-never-be-popular-baff02b906dc [Accessed 7 Nov. 2018].

Lugo, A. (2017). How Secure is Microsoft Azure?. [online] INVID. Available at: https://invidgroup.com/how-secure-is-microsoft-azure/ [Accessed 31 Oct. 2018].

Miller, R. (2018). Leena AI builds HR chatbots to answer policy questions automatically. [online] TechCrunch. Available at: https://techcrunch.com/2018/06/29/leena-ai-builds-hr-chat-bots-to-answer-policy-questions-automatically/ [Accessed 24 Oct. 2018].

Phillips, C. (2018). Natural Language Processing (NLP) & Why Chatbots Need it. [online] Chatbots Magazine. Available at: https://chatbotsmagazine.com/natural-language-processing-nlp-why-chatbots-need-it-a9d98f30ab13 [Accessed 21 Nov. 2018].

Rasa.com. (2018). Rasa: Open source conversational AI. [online] Available at: https://rasa.com/ [Accessed 9 Nov. 2018].

Rouse, M. (n.d.). What is Amazon Lex? - Definition from WhatIs.com. [online] SearchAWS.TechTarget.com. Available at: https://searchaws.techtarget.com/definition/Amazon-Lex [Accessed 23 Oct. 2018].

Selby, P. (2018). Three reasons rule-based chatbots rule | CustomerThink. [online] Customerthink.com. Available at: http://customerthink.com/three-reasons-rule-based-chatbots-rule/ [Accessed 20 Nov. 2018].

Stott, L. (2016). What is Microsoft Bot Framework Overview. [online] Microsoft Faculty Connection. Available at: https://blogs.msdn.microsoft.com/uk_faculty_connection/2016/04/05/what-is-microsoft-bot-framework-overview/ [Accessed 6 Nov. 2018].

Textblob.readthedocs.io. (2018). TextBlob: Simplified Text Processing — TextBlob 0.15.2 documentation. [online] Available at: https://textblob.readthedocs.io/en/dev/ [Accessed 20 Nov. 2018].

Velloso, M., Iqbal, K. and Standefer, R. (2017). Principles of bot design - Bot Service. [online] Docs.microsoft.com. Available at: https://docs.microsoft.com/en-us/azure/bot-service/bot-service-design-principles?view=azure-bot-service-4.0 [Accessed 5 Nov. 2018].

Ward, D. (2017). Rule-based versus AI-based chatbots: Which is best for your business?. [online] Blog.rxpservices.com. Available at: http://blog.rxpservices.com/rule-based-versus-ai-based-chatbots-which-is-best-for-your-business [Accessed 23 Nov. 2018].

Wolhuter, S. (2017). Lex vs Luis: choosing the right solution to create bots for your business. [online] WeAreBrain. Available at: https://thebrainfiles.wearebrain.com/choosing-a-tool-to-make-bots-for-your-business-99d11d5b75 [Accessed 8 Nov. 2018].

Section 3: Technology Assessment

Docs.microsoft.com. (2018). Design knowledge bots - Bot Service. [online] Available at: https://docs.microsoft.com/en-us/azure/bot-service/bot-service-design-pattern-knowledge-base?view=azure-bot-service-4.0 [Accessed 14 Dec. 2018].

Docs.microsoft.com. (2018). Knowledge base - QnA Maker - Azure Cognitive Services. [online] Available at: https://docs.microsoft.com/en-us/azure/cognitive-services/qnamaker/concepts/knowledge-base [Accessed 14 Dec. 2018].

Qnamaker.ai. (2018). QnA Maker. [online] Available at: https://www.qnamaker.ai/old [Accessed 14 Dec. 2018].

Section 4: Detailed Project Proposal

Chatbots Journal. (2019). Top 10 Reasons: Why Your Business Need A Chatbot Development. [online] Available at: https://chatbotsjournal.com/top-10-reasons-why-your-business-need-a-chatbot-development-5a53760da1b6 [Accessed 20 Jan. 2019].

Cloud, G. (2019). Dialogflow Enterprise Edition | Dialogflow Enterprise Edition | Google Cloud. [online] Google Cloud. Available at: https://cloud.google.com/dialogflow-enterprise/ [Accessed 18 Jan. 2019].

Dialogflow. (2019). Agents overview | Dialogflow. [online] Available at: https://dialogflow.com/docs/agents [Accessed 19 Jan. 2019].

Dialogflow. (2019). Dialogflow. [online] Available at: https://dialogflow.com/pricing [Accessed 18 Jan. 2019].

Dialogflow. (2019). Entities overview | Dialogflow. [online] Available at: https://dialogflow.com/docs/entities [Accessed 19 Jan. 2019].

Dialogflow. (2019). Intents overview | Dialogflow. [online] Available at: https://dialogflow.com/docs/intents [Accessed 19 Jan. 2019].

Dialogflow. (2019). Prebuilt Agents and Small Talk | Dialogflow. [online] Available at: https://dialogflow.com/docs/agents/prebuilt-smalltalk [Accessed 20 Jan. 2019].

Docs.kommunicate.io. (2019). Installation · Help Docs. [online] Available at: https://docs.kommunicate.io/docs/web-installation.html [Accessed 18 Jan. 2019].

Goebel, T. (2019). Why You Can't Just Convert FAQs into a Customer Service Chatbot 1:1. [online] Aspect Blogs. Available at: https://blogs.aspect.com/faq-customer-service-chatbot/ [Accessed 17 Jan. 2019].

Matthews, K. (2019). 5 Chatbot-Building Tools for Restaurant Owners – Chatbots Life. [online] Chatbots Life. Available at: https://chatbotslife.com/5-chatbot-building-tools-for-restaurant-owners-ecd985064187 [Accessed 19 Jan. 2019].

Shridhar, K. (2019). Rule based bots vs AI bots – #WeCoCreate – Medium. [online] Medium. Available at: https://medium.com/botsupply/rule-based-bots-vs-ai-bots-b60cdb786ffa [Accessed 17 Jan. 2019].

Section 5: Iteration 1 Report

Google Cloud. (2019). Dialogflow Enterprise Edition | Dialogflow Enterprise Edition | Google Cloud. [online] Available at: https://cloud.google.com/dialogflow-enterprise/ [Accessed 6 Mar. 2019].

Section 6: Iteration 2 Report

Docs.aws.amazon.com. (2019). What Is Amazon Lex? - Amazon Lex. [online] Available at: https://docs.aws.amazon.com/lex/latest/dg/what-is.html [Accessed 18 Mar. 2019].

Sas.com. (2019). What is Natural Language Processing?. [online] Available at: https://www.sas.com/en_ie/insights/analytics/what-is-natural-language-processing-nlp.html [Accessed 18 Mar. 2019].

Section 7: Iteration 3 Report

Dialogflow. (2019). Default intents | Dialogflow. [online] Available at: https://dialogflow.com/docs/intents/default-intents [Accessed 15 Apr. 2019].

Dialogflow. (2019). Developer entities | Dialogflow. [online] Available at: https://dialogflow.com/docs/entities/developer-entities [Accessed 15 Apr. 2019].

Ushakov, M. (2019). mishushakov/dialogflow-web-v2. [online] GitHub. Available at: https://github.com/mishushakov/dialogflow-web-v2 [Accessed 15 Apr. 2019].