TO

Abstract

[Draw your reader in with an engaging abstract. It is typically a short summary of the document.   
When you’re ready to add your content, just click here and start typing.]

Course: COM547 – Computing Systems

Supervisor: Dr Colin Turner

Date: 22/04/2018

Project Final Report

Student: B00348751 – Paul Connolly

# Acknowledgements

Contents

[Acknowledgements ii](#_Toc512003359)

[1 Introduction 1](#_Toc512003360)

[1.1 Background 1](#_Toc512003361)

[1.2 Project Aims 5](#_Toc512003362)

[1.3 Project Objectives 5](#_Toc512003363)

[1.4 Project Activities 5](#_Toc512003364)

[1.5 Outline of Dissertation Structure 6](#_Toc512003365)

[2 Chatbots in Use Today 8](#_Toc512003366)

[2.1 What is a chatbot? 8](#_Toc512003367)

[2.2 Review of Chatbots 8](#_Toc512003368)

[2.2.1 Marvel 8](#_Toc512003369)

[2.2.2 RoofAi 8](#_Toc512003370)

[2.2.3 Lemonade 11](#_Toc512003371)

[3 Technical Background 13](#_Toc512003372)

[3.1 Frameworks 13](#_Toc512003373)

[3.1.1 Microsoft Bot Framework 13](#_Toc512003374)

[3.1.2 Wit.ai 14](#_Toc512003375)

[3.1.3 DialogFlow 15](#_Toc512003376)

[3.2 Development Language Options 16](#_Toc512003377)

[3.2.1 C# 16](#_Toc512003378)

[3.2.2 Ruby 16](#_Toc512003379)

[3.2.3 Python 16](#_Toc512003380)

[3.2.4 Javascript 17](#_Toc512003381)

[3.3 Software Technologies 17](#_Toc512003382)

[3.3.1 REST 17](#_Toc512003383)

[3.3.2 JSON 17](#_Toc512003384)

[3.3.3 HTTPS 17](#_Toc512003385)

[3.4 Databases 18](#_Toc512003386)

[3.5 Source Control 18](#_Toc512003387)

[4 Development Lifecycle 20](#_Toc512003388)

[4.1 Software Development Process Models Considered 20](#_Toc512003389)

[4.1.1 Waterfall Model 20](#_Toc512003390)

[4.1.2 Spiral Model 21](#_Toc512003391)

[4.1.3 Agile Model 23](#_Toc512003392)

[5 Requirements Gathering & Risk Analysis 25](#_Toc512003393)

[5.1 Requirements Gathering Techniques 25](#_Toc512003394)

[5.2 Requirements Specification 25](#_Toc512003395)

[5.2.1 Functional Requirements 26](#_Toc512003396)

[5.2.2 Non-functional Requirements 26](#_Toc512003397)

[5.3 Requirements Validation 26](#_Toc512003398)

[5.4 Risk Analysis 27](#_Toc512003399)

[5.5 Data Protection 27](#_Toc512003400)

[6 Project Plan 28](#_Toc512003401)

[6.1 Project Scope 28](#_Toc512003402)

[6.2 Project Schedule 28](#_Toc512003403)

[6.2.1 Project Milestones 28](#_Toc512003404)

[6.2.2 Work Breakdown Structure 28](#_Toc512003405)

[6.3 Project Resources 29](#_Toc512003406)

[7 Design 30](#_Toc512003407)

[7.1 Conversation Flows 30](#_Toc512003408)

[7.2 Conversation Dialogs 31](#_Toc512003409)

[7.2.1 First Interaction 31](#_Toc512003410)

[7.2.2 Motor Dialog 31](#_Toc512003411)

[7.2.3 Home Dialog 31](#_Toc512003412)

[7.3 User Interface 31](#_Toc512003413)

[7.4 High-level System Architecture 32](#_Toc512003414)

[7.5 Class Diagram 32](#_Toc512003415)

[7.6 Database Design 32](#_Toc512003416)

[8 Implementation 32](#_Toc512003417)

[System Architecture 32](#_Toc512003418)

[Detail code 32](#_Toc512003419)

[9 Challenges and Solutions 32](#_Toc512003420)

[10 Testing and Results 32](#_Toc512003421)

[11 Evaluation 32](#_Toc512003422)

[12 Conclusions 33](#_Toc512003423)

[13 Suggested Future Improvements to Project 33](#_Toc512003424)

[13.1 Link to Applied Systems eQuote Leads System 33](#_Toc512003425)

[13.2 Add Natural Language Processing 33](#_Toc512003426)

[13.3 Convert to JavaScript Widget 34](#_Toc512003427)

[References 35](#_Toc512003428)

[Appendices 38](#_Toc512003429)

Table of Figures

[Figure 1 - Social Networking Tracker (Jan 2017) 3](#_Toc511993651)

[Figure 2 - Social Messaging Tracker (Jan 2017) 4](#_Toc511993652)

[Figure 3 - Roof.ai chatbot example (Moubarak, 2018) 9](#_Toc511993653)

[Figure 4 - Roof.ai widget on The Keyes Company website homepage (The Keyes Company, 2018) 9](#_Toc511993654)

[Figure 5 - The Keyes Company example conversation with Roof.ai chatbot 10](#_Toc511993655)

[Figure 6 - Lemonade chat example (Lemonade, 2018) 11](#_Toc511993656)

[Figure 7 - Example of Lemonade question with options (Lemonade, 2018) 12](#_Toc511993657)

[Figure 8 - HTTP vs HTTPS (Comodo, 2018) 18](#_Toc511993658)

[Figure 9 - Software Development Life Cycle (Gordiyenko, 2014) 21](#_Toc511993659)

[Figure 10 - Diagram of Spiral Model (Boehm, 1988) 22](#_Toc511993660)

[Figure 11 - Representation of the Agile Development Model (ISTQB Exam Certification, 2018) 23](#_Toc511993661)

[Figure 12 - Risk Register 27](#_Toc511993662)

[Figure 13 - High level system architecture 32](#_Toc511993663)

# 1 Introduction

## 1.1 Background

Applied Systems have operated for over 30 years powering the insurance industry across the USA, Canada, Ireland and the United Kingdom, providing industry leading technology to the insurance industry (Applied Systems, 2018).

As a company; Applied Systems strive to provide insurance brokers with innovative software solutions to maximise the brokers business profits and improve customer communications.

As an employee of Applied Systems, and being sponsored by them during my university studies, the challenge was set for me to use my Computing Systems Project as an opportunity to research, design and implement a proof-of-concept for the next innovative piece of software that they may put in to production for release to the market.

Considering my background as a Software Developer with experience working on web-based products, I decided to research emerging trends on the web in relation to business-to-customer interactions in the Insurtech (Insurance Technology) industry.

Research suggests that the number one predicted trend in the Insurtech market as predicted by “The Digital Insurer” is “automation will replace human effort across the entire insurance value chain” (Huckstep, 2017). Although this is not a trend restricted to the insurance industry, it is likely to have significant impact in the insurance world as a lot of the industry business methods continue to operate in a manner that is outdated and is more suited to consumers prior to the Internet. With the rise of the Internet, consumers now want a full digital experience without need for human interaction (Huckstep, 2017).

The reality of a more digitalised consumer base is supported by Price Waterhouse Cooper’s (PWC) Irish Total Retail Survey where they found that 48% of Irish consumers had used their mobile phones to shop online at least a few times a year, with 30% stating that they feel their mobile will become their main method of shopping in the future (PwC, 2017).

Another prediction for 2018 comes from Jay Samit, a renowned digital media expert, who believes this year will be “the year of the bots” (Samit, 2017). Samit explains how chatbots will become more intelligent in the use and understanding of natural language to become more capable of helping us with our daily routines.

As well as the predicted growth of chatbots Samit suggests, Ipsos MRBI Tracker research shows 64% of people in Ireland are using Facebook as a social media platform (Figure 1), with 58% using Facebook Messenger as their social media messenger app of choice (Figure 2).

The findings from this research helped substantiate m decision to design and build an insurance chatbot for release to Facebook Messenger.

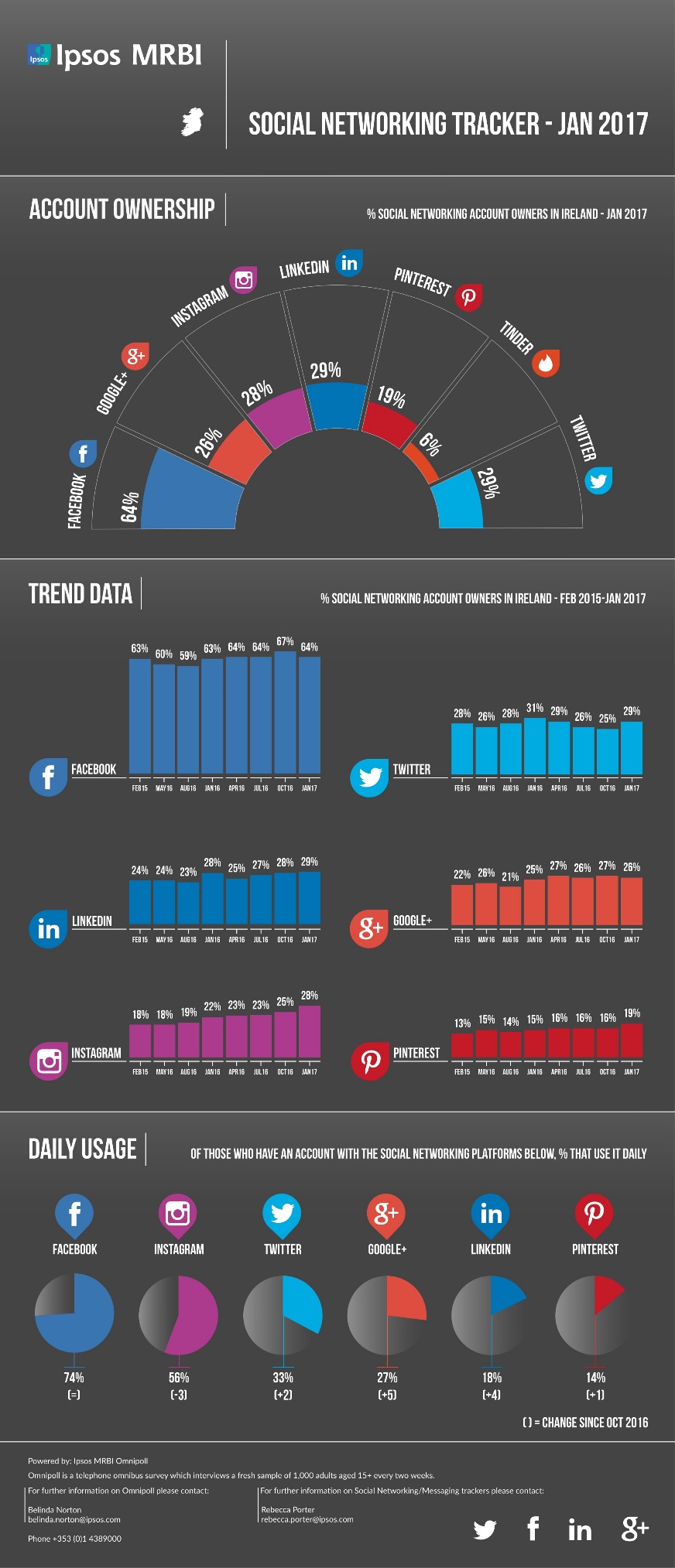


Figure - Social Networking Tracker (Jan 2017)

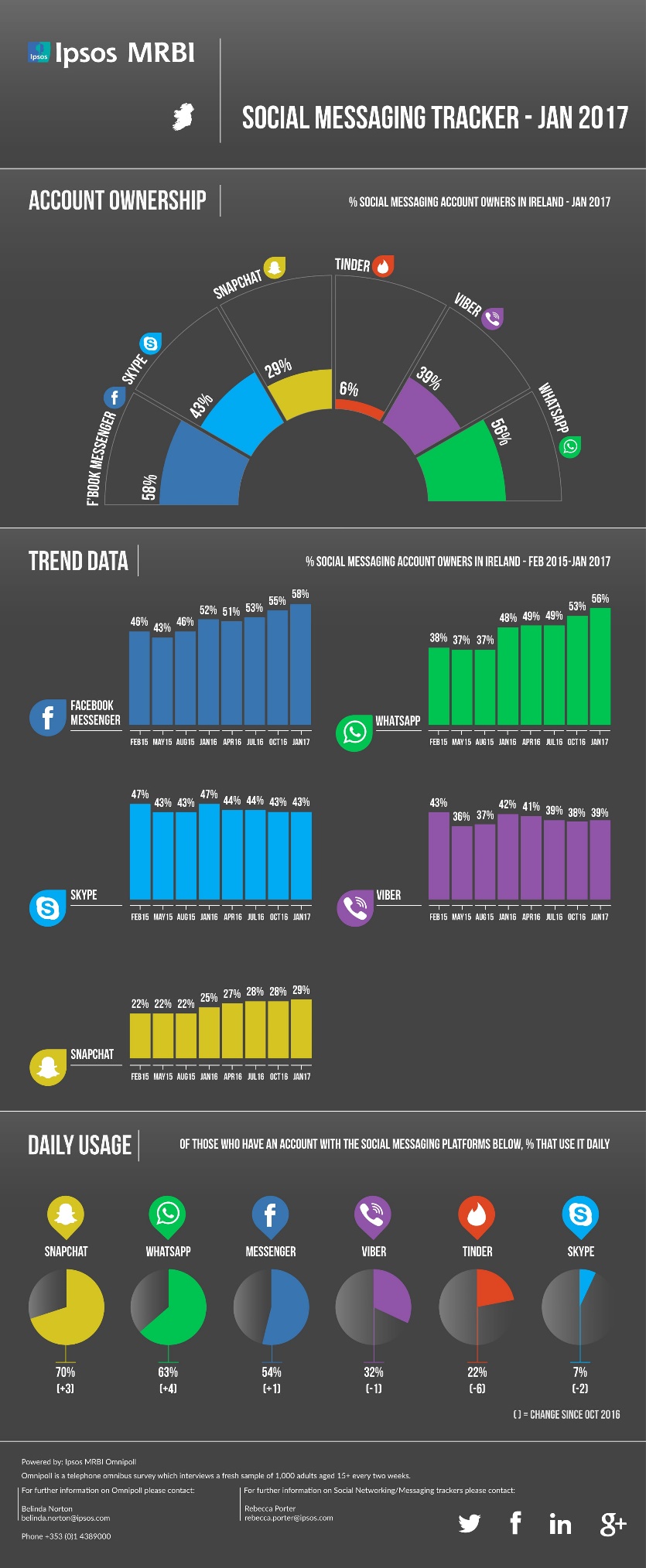


Figure - Social Messaging Tracker (Jan 2017)

In terms of functionality, I envisioned the chatbot responding to and communicating with insurance broker customers who were requesting an insurance quote.

With the findings from the research and having taken into consideration my personal interest, I decided to develop a chatbot that could be deployed to use through Facebook Messenger. The chatbot should enable insurance broker customers to communicate with the bot and receive an insurance quote.

## 1.2 Project Aims

The aim of the project was to create a proof-of-concept chatbot for Applied Systems.

The chatbot would allow Facebook Messenger users to “chat” with the bot and receive a motor or home insurance quote based on the information they had entered.

## 1.3 Project Objectives

To give the project structure, a list of project objectives was identified.

* The user can receive a motor insurance quick quote[[1]](#footnote-1)
* The user can receive a home insurance quick quote1
* Retain any conversations between the user and the bot
* Retain quotes provided to the user
* Track and retain any errors encountered during the workflow
* Allow user to retrieve a previous quote
* Allow user to choose a returned quote
* Allow user to choose to be contacted by the insurance broker that provided the quotes
* Follow up conversation with email to insurance broker and the user

## 1.4 Project Activities

To ensure project objectives were met, a list of project activities were identified.

* Determine questions and question order for motor insurance quick quote
* Determine questions and question order for home insurance quick quote
* Design database for storing conversations, quotes and errors
* Establish possible SQL queries for inserting, updating and reading from database
* Research possible frameworks for chatbot implementation
* Research most appropriate programming language to use for development
* Create chatbot solution
* Create logging service
* Create database
* Write and implement database
* Write and implement test plan
* Perform user testing
* Implement changes from user testing findings

## 1.5 Outline of Dissertation Structure

The dissertation has been structured as follows:

* Chapter 1 – Introduction

An overview of how the project’s conception, aims, objectives and activities.

* Chapter 2 – Chatbots in Use Today

An evaluation of chatbots currently in operation. An array of chatbots were identified and critiqued; highlighting the advantages and acknowledging potential disadvantages of using a chatbot for a business.

* Chapter 3 – Technical Background

A review of possible technologies that could be used to build the chatbot, with justification for the chosen technology stack utilised.

* Chapter 4 – Development Lifecycle

Elaboration on the Software Development Lifecycle to be used for this project.

* Chapter 5 – Requirements Gathering and Analysis

Explanation of how requirements for the project where determined. The project requirements are detailed alongside a risk analysis of the project.

* Chapter 6 – Design

High-level overview of the system design, the initial database design and the design of the chatbot conversations between the user and the bot.

* Chapter 7 – Implementation

A detailed assessment of the system implementation. Programming practices, libraries and frameworks used and code snippets are detailed and explained in this chapter.

* Chapter 8 – Challenges and Solutions

A review of some of the challenges faced during the project lifecycle with proposed solutions to these challenges noted.

* Chapter 9 – Testing and Results

Testing techniques used during the project with results of the tests given.

* Chapter 10 – Evaluation

An evaluation of the project management techniques used, system implementation, technology used, and personal experience gained throughout the project.

* Chapter 11 – Conclusions

Reflection on the project’s and its successes and limitations.

* Chapter 12 – Suggested Future Improvements to Project

Suggestions for project improvements and/or enhancements focusing primarily on additional functionality, implementation and estimated implementation times.

# 2 Chatbots in Use Today

## 2.1 What is a chatbot?

Business Insider UK defines a chatbot as a robot that can maintain a conversation with a human. It is essentially a virtual conversation with a piece of software (Nguyen, 2017).

## 2.2 Review of Chatbots

To get a better understanding of how a chatbot should be designed and how users typically interact with them, a critical evaluation of a selection of chatbots from various business sectors has been performed with the findings detailed below.

The chatbots I have selected for review are:

* Marvel
* RoofAi
* Lemonade

### 2.2.1 Marvel

The Marvel chatbot is a way for fans of Marvel Comics to chat directly to some of their favourite characters through Facebook Messenger or Twitter DM (Morse, 2017).

The chatbot was built with Conversable, a platform for building AI-enhanced messaging experiences (Conversable, 2018), making use of natural language processing (NLP) and machine learning which bots are more frequently using.

The Marvel bot unfortunately is not a true conversationalist. The conversation has a couple of paths that the conversation can go down but always ending in the same way – trying to sell the user merchandise. Which admittedly is a great business use but could be frustrating for the end user if they are inevitably going to end every conversation with an advertisement enticing them to spend money.

### 2.2.2 RoofAi

RoofAi promotes itself as a “smart chat” bot – “a combination of live chat and bots” (RoofAi, 2018). RoofAi view their website widget, a widget built for realtors in the United States, as a tool to manage real-time conversations as and when your website user has a query – stating that “you can now be instantly responsive 24/7, 365” (Moubarak, 2018).

The Roof.ai bot is capable of capturing user information, scheduling viewing appointments and directing leads to the best suited real estate agent; e.g. assigning a user looking to rent a property to an available agent responsible for rental properties (Moubarak, 2018).

The bot is implemented on a customer site in the form of widget.

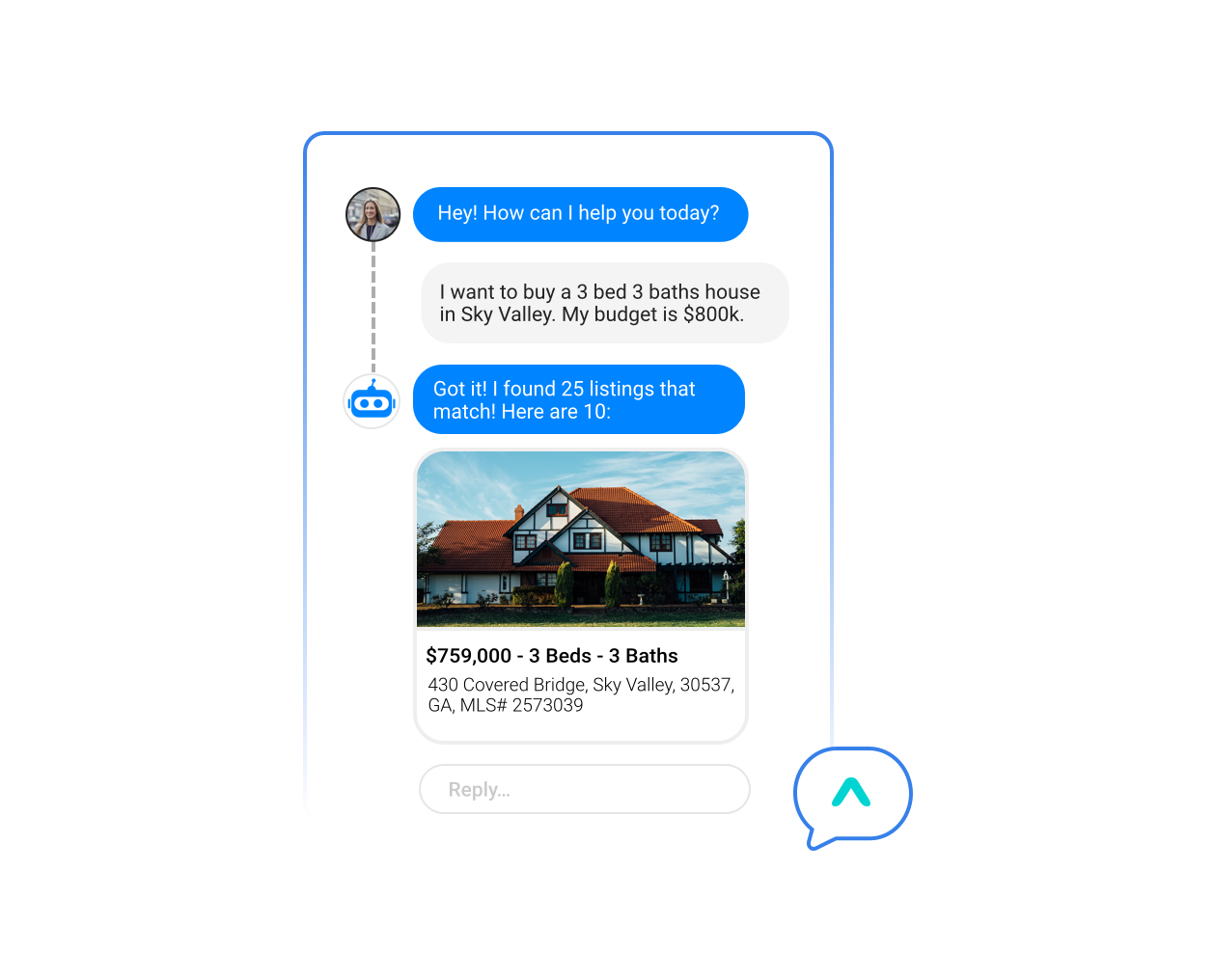


Figure - Roof.ai chatbot example (Moubarak, 2018)

#### 2.2.2.1 Roof.ai Review

Using the website for The Keyes Company, a Florida based real-estate agency, they have added the Roof.ai chatbot widget to their homepage. The widget has been styled to fit with the website and makes use of The Keyes Company logo and branding colours.

It is immediately offering assistance as can be seen below. This is good as it draws attention to the bot in a subtle manner and highlights that the bot is actively available.

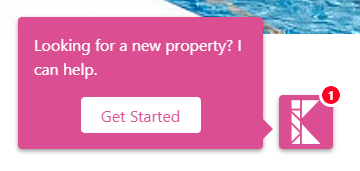


Figure - Roof.ai widget on The Keyes Company website homepage (The Keyes Company, 2018)

Having used the bot to search for properties in the Florida area the language the bot uses is a positive standout. The language is professional but friendly.

The conversation is guided by the bot asking questions as a human estate agent would. With some questions the bot gives the user some choices which adds an extra level of guidance and validation to ensure that answers given are correct and match with expected answers.

Questions are asked immediately after an answer has been received so the communication is fast and direct.

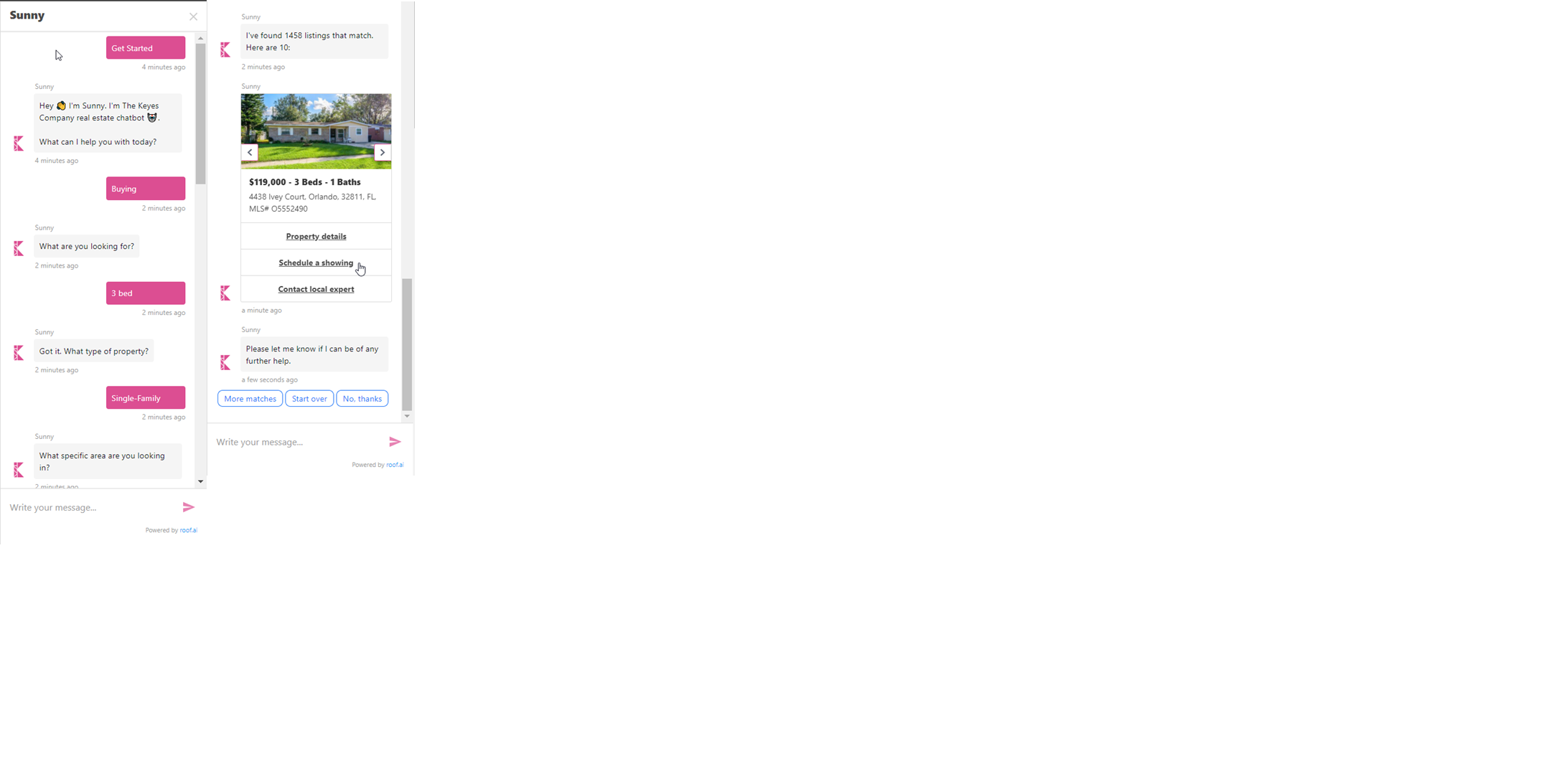


Figure - The Keyes Company example conversation with Roof.ai chatbot

As can be seen above the chatbot was able to return a selection of properties based on the criteria entered. The properties are nicely displayed in a scrollable carousel making it easy for the user to browse the properties.

Similarities between the Roof.ai chatbot and the chatbot being developed in this project are the use of guided conversation. The benefit of this being that the user should not go too far off the main purpose of the chatbot. The Roof.ai chatbot also does a good job of giving the user options for answering questions but mixing that with questions open to a free response so the chat feels more natural, this is also something to be implemented in this project’s chatbot.

### 2.2.3 Lemonade

The chatbot that also currently operates in the insurance industry is used, and was developed by, a company called Lemonade Insurance Agency based in New York.

Lemonade tell their users to “Forget everything you know about insurance” (Lemonade, 2018), as they sell insurance based on a new business model and a central component to this is their artificial intelligence bot (Wissner-Levy, 2016). They aim to make the process of getting insurance faster, more honest and more transparent (Wissner-Levy, 2016).

Lemonade also offer a widget and API for integration on other websites or applications (Lemonade, 2018).

#### 2.2.3.1 Lemonade Review

Through the Lemonade website ([www.lemonade.com](http://www.lemonade.com)), a review of the process to get an insurance price was completed.

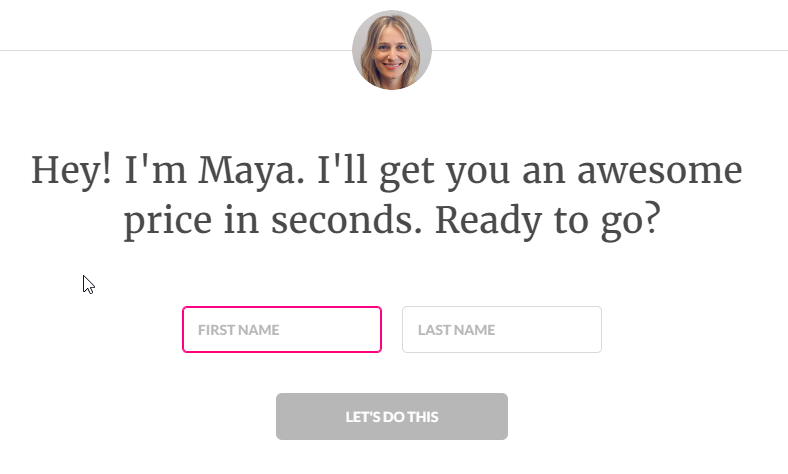


Figure - Lemonade chat example (Lemonade, 2018)

Lemonade have given their bot a name and a profile picture, Maya (Lemonade, 2018). This adds a personal touch to the bot and makes it feel less like chatting to a piece of software and more like chatting to a human being which will have a positive impact on the end user.

Similarly to Roof.ai, the Lemonade chatbot asks questions in a manner that guides the user through the workflow whilst providing options to answer the questions, as below.

The language used by the bot again is natural and polite.

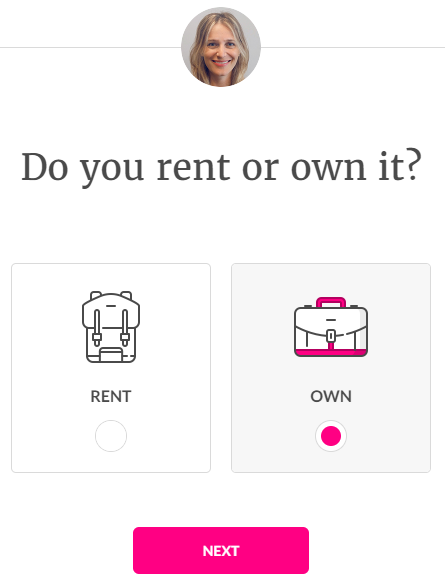


Figure - Example of Lemonade question with options (Lemonade, 2018)

# 3 Technical Background

Before development; research has been carried out on how to best build a chatbot and complete the project to meet all the project objectives. The review will include; development languages, any frameworks or libraries that could be utilised, database options and development environments.

Options for source control will also be discussed.

## 3.1 Frameworks

As with development of any project, it is possible to use frameworks to aid development. Using frameworks can quicken development. Wikipedia defines a framework as being “a universal, reusable software environment that provides particular functionality as part of a larger software platform to facilitate development of software applications, products and solutions” (Wikipedia, 2018).

A framework, in the context of a chatbot, i.e. a bot framework, is a great tool to have as it abstracts a lot of the manual work involved in creating a bot (Maruti Techlabs, 2018).

To develop a chatbot, the following bot frameworks have been considered:

* Microsoft Bot Framework
* Wit.ai
* DialogFlow

### 3.1.1 Microsoft Bot Framework

The Microsoft Bot Framework is comprised of various tools in its software development kit (SDK). The main tools are:

* Bot Connector
* LUIS

#### 3.1.1.1 Bot Connector

This service in the Microsoft Bot Framework is what enables the bot to communicate messages on channels (the platform the bot is configured to run on, e.g. Facebook Messenger, Slack, Skype, etc.) (Microsoft, 2017).

Communication on these channels is achieved using industry-standard REST and JSON over HTTPS.

#### 3.1.1.2 LUIS.ai

Meaning Language Understanding Intelligent Service – LUIS enables a bot to understand natural language when input by the user (Berry, 2017). It uses machine learning to accept input and extract an intended meaning from the input so it can return a relevant response (Berry, 2017).

The key concepts to LUIS are:

* Intents – these can be considered the “action” a user wishes to perform within an application. Within LUIS a language model, the developer defines intents and maps these to actions (Berry, 2017)
* Utterances – these are the text the user may input that the bot needs to be able to receive and understand (Berry, 2017). There can be many variations to an utterance but the utterance will be linked to a specific intent.
* Entities – these are pieces of information that may appear in an utterance. Identifying entities in an utterance, LUIS is able to choose the best suited action to response to the user (Berry, 2017).

In using LUIS, a developer defines a domain specific language model and fills it with intents, utterances and entities (Berry, 2017). The model then must be trained and published. The LUIS app will then receive an utterance as a HTTP request. From this request it determines the user interaction and responds (Berry, 2017). The user utterance sent from the client application is evaluated to a JSON object by LUIS which is then sent back to the client app (Berry, 2017).

### 3.1.2 Wit.ai

Wit.ai is an open-source API (Application Programming Interface) that “makes it easy for developers to build applications and devices that you can talk or text to” (wit.ai, 2018). It is a natural language platform that uses each interaction to learn so it can provide more accurate responses (wit.ai, 2018).

One appealing aspect of Qit.ai is that because it is open-source, it is able to share what it has learned across the all developers using wit.ai.

Wit also uses entities and intents to understand the action the user is trying to perform.

### 3.1.3 DialogFlow

On the same premise as Microsoft’s LUIS and Wit.ai; DialogFlow uses machine learning to understand meaning from what a user has input or said.

DialogFlow use an “agent” to manage the conversation between the user (human) and bot (DialogFlow, 2018). DialogFlow describe these agents as Natural Language Understanding (NLU) modules (DialogFlow, 2018). The NLU module converts the user input into data that can determine an action.

Other important aspects of DialogFlow are:

* Entities – domain-specific phrases that can be mapped to NLP (Natural Language Processing) phrases (Maruti Techlabs, 2018)
* Intents – the action to be taken based on what a user has input (Maruti Techlabs, 2018)
* Actions – what will happen based of the identified intent (Maruti Techlabs, 2018)
* Contexts – a string representation to evaluate the user expression. Useful for determining meaning (Maruti Techlabs, 2018)

## 3.2 Development Language Options

Across the frameworks mentioned in the previous section, there are various options for developers to pick from when developing their bot.

The following languages can be with the aforementioned frameworks:

* C#
* Ruby
* Python
* JavaScript

A brief assessment was made of each language with the findings given below.

### 3.2.1 C#

Closely connected to the .NET Framework, C# is an object-orientated development language that is type-safe (Wagner, Wenzel, & Levin, 2015). It supports object-orientated concepts of encapsulation, inheritance and polymorphism (Wagner, Wenzel, & Levin, 2015).

The .NET Framework is comprised of the virtual execution system, CLR (Common Language Runtime) and a series of class libraries to help aid development.

### 3.2.2 Ruby

An open-source programming language, Ruby gives itself the title of “A Programmer’s Best Friend” (Ruby, 2018). Ruby states that it is a language focused on being simplistic and productive (Ruby, 2018).

Mostly used be developers on Linux, Ruby is by design a simple, complete, extensible and portable programming language (Rouse, 2010).

### 3.2.3 Python

According to the Python homepage, “Python is a programming language that lets you work more quickly and integrate your systems more effectively” (Python, 2018).

Python is another object-orientated programming language. It is a high-level language and is suited to Rapid Application Development due to its high-level data structures, dynamic typing and dynamic binding.

It promotes modular programs and code reuse.

### 3.2.4 Javascript

A programming language most associated to building things for the web. It is usually used to build dynamic component for webpages (Mozilla, 2018).

JavaScript is boosted in it abilities by integrating with Application Programing Interfaces (APIs) to help a developer build complex programs.

## 3.3 Software Technologies

### 3.3.1 REST

Standing for Representational State Transfer, REST is a way for systems to communicate with one another (Codecademy, 2018). It is a web standard that separates the concerns of the client and the server (Codecademy, 2018). This makes it easier to scale and has no concern on what the user interface is doing, providing both the client and server are aware of the format messages must be sent in.

### 3.3.2 JSON

JSON, JavaScript Object Notation, “is a way to store information in an organised, easy-to-access manner” (Lengstorf, 2018). JSON makes it easy to get data quickly and cross platform (Lengstorf, 2018).

### 3.3.3 HTTPS

This is a secure, encrypted version of the HTTP protocol. The HTTP (Hyper Text Transfer Protocol) is the protocol determining how data should be communicated between the web browser and a website (Comodo, 2018). HTTPS adds “Secure” to this protocol, meaning all communication is encrypted.

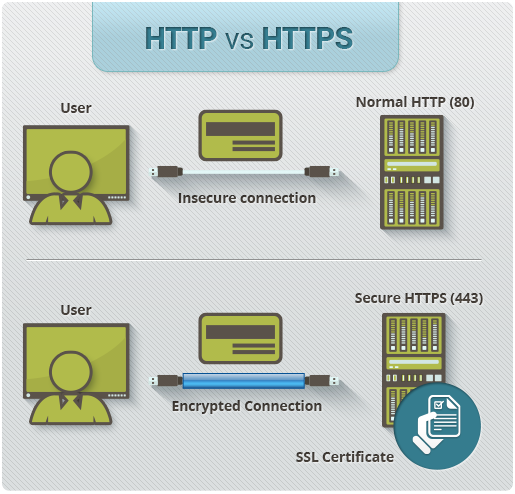


Figure - HTTP vs HTTPS (Comodo, 2018)

## 3.4 Databases

The database to be used for this project is Microsoft SQL Server. It is a RDBMS (Relational Database Management System) (Rouse, Microsoft SQL Server, 2017). A relational database is data organised in to tables with well-defined relationships between these tables (Techopedia, 2018).

To interact with these databases, the best practice is to use Structured Query Language (SQL). This allows CRUD (Create, Read, Update and Delete) interaction with the database.

## 3.5 Source Control

Amazon Web Services defines source control as “the practice of tracking and managing change to code” (Amazon Web Services, 2018). Using a source control system gives developers a method of tracking the development history of a project. It is also a useful tool when conflicts in code arise when two versions of a code base are merged together.

Source control allows a developer to create branches that isolate their development code before it is combined with the main code base.

This project will use Git for source control. Git is a distributed version control system (Ngan, 2018). Main functionality of Git includes branches and commits. Branches allow development code to be kept separate from the main code base and commits are how developers save their code changes (Amazon Web Services, 2018).

# 4 Development Lifecycle

The Software Development Life Cycle is a process that aims to produce software with the highest quality and with the lowest cost possible in the shortest amount of time (Stackify, 2017).

According to (Stackify, 2017), the advantages of following the SDLC are:

* It allows a high level of management control
* Gives developers a good understanding of what they are trying to build
* An agreement is made upfront on what the project outcome should be
* It sets out an agreed plan on how to reach the proposed goal

The Software Development Life Cycle defines six stages that can be implemented in various ways by different SDLC models.

These 6 stages are:

* Planning
* Defining
* Designing
* Building
* Testing
* Deployment

The models, also called Software Development Process Models, will follow steps unique to each model but will still all either strictly or loosely follow the Software Development Life Cycle stages (Tutorials Point, 2017).

## 4.1 Software Development Process Models Considered

The following models have been considered as a development model to use on this project.

### 4.1.1 Waterfall Model

The first process model to be introduced to the software development industry, it was designed to be used in a wat that processes do not overlap; one process must finish before the other starts (Tutorials Point, 2017).



Figure - Software Development Life Cycle (Gordiyenko, 2014)

#### 4.1.1.1 Advantages of Waterfall Model

* Simple to use and understand
* Each process has specific goals and outcomes
* Stages of the project are well defined

#### 4.1.1.2 Disadvantages of Waterfall Model

* A working product is not delivered until near the end of the life cycle
* Not suitable for projects with requirements that are likely to change
* Stages must wait on their predecessors to finish before they can start

#### 4.1.1.3 Why the Waterfall model was not chosen

Due to its strict phase completion rules, the Waterfall model is not suited for this project that is open to changing requirements throughout the duration of the project.

The plan is also to have regular feedback from stakeholders and users which again does not fit with the Waterfall model process.

### 4.1.2 Spiral Model

Consisting of four phases; planning, Risk, Engineering and Evaluation, a software project using this model will pass through each phase iteratively until the project is delivered (International Software Testing Qualifications Board, 2017).

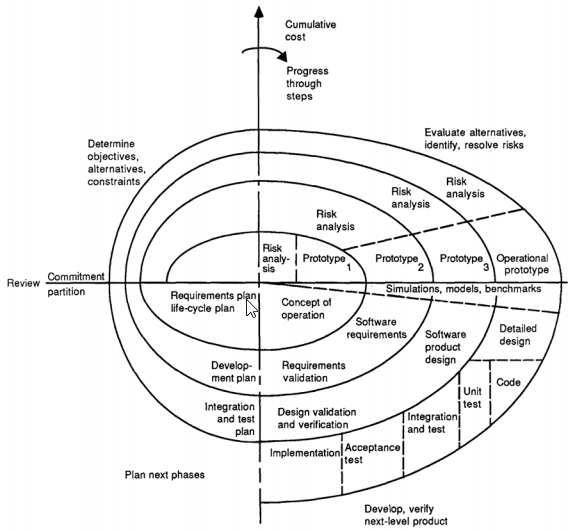


Figure - Diagram of Spiral Model (Boehm, 1988)

#### 4.1.2.1 Advantages of Spiral Model

* Emphasis on risk analysis means risk are identified early and can be managed or avoided
* Software is produced early and frequently
* Software functionality can change or be added late in to the project

#### 4.1.2.2 Disadvantages of Spiral Model

* Can be an expensive model
* Not suited to small projects
* Risk analysis requires experienced analysts

#### 4.1.2.3 Why the Spiral model was not chosen

The Spiral model was not selected for use on this project due to the short life span of the project. Spiral is more suited to larger projects with a long-term commitment.

The project is also considered low risk and the requirements of the project are clear and concise.

### 4.1.3 Agile Model

The Agile SDLC model put focus on delivering products to the customer on a regular basis. It combines iterative and incremental models and focuses on customer satisfaction and welcomes change (Tutorials Point, 2017).

A project using the agile model will break a project down in to smaller pieces of work and deliver these in an iterative manner.

Each iteration is able to have multiple processes active at any one time. The processes include:

* Planning
* Requirements Analysis
* Design
* Coding
* Unit testing
* Acceptance testing

The iterations that produce working software are usually time boxed to an amount of time decided by the team.

Iterations can happen in a similar way to the image below.



Figure - Representation of the Agile Development Model (ISTQB Exam Certification, 2018)

The Agile model come with an Agile Manifesto (Agile Manifesto, 2017) that states the following principles:

* Individuals and interaction over processes and tools
* Working software over comprehensive documentation
* Customer collaboration over contract negotiations
* Responding to change over following a plan

#### 4.1.3.1 Advantages of Agile

* A realistic take on how software is best developed
* Functionality is developed quickly
* Processes work with pre-defined or changing requirements
* Little or no planning required

#### 4.1.3.2 Disadvantages of Agile

* Depends heavily on stakeholder and customer communication and feedback
* Less focus on documentation can lead to problems when on boarding new team members or handing a project on to another team
* Changing requirements and functionality can have an adverse effect on the project delivery deadline and can lead to scope creep on the project

#### 4.1.3.3 Why Agile was chosen for this project

Agile has been chosen as the software development model for the project as the Agile model is one widely practised within Applied Systems.

The opportunity to develop software quickly and get regular feedback on it means the end product is more likely to meet the Project Sponsor’s expectations.

Not having to focus on detailed documentation also means the emphasis can be placed on building a working system. Due to the tight schedule of this project, this is a benefit that cannot be overlooked.

# 5 Requirements Gathering & Risk Analysis

Before development on a project begins, requirements gathering is an essential process to giving the project the best chance of success.

Requirements gathering, also referred to as requirements elicitation, is the process of producing a list of requirements based on what the project stakeholders want and need from the system (Inflectra, 2018).

Requirements come in the form of functional and non-functional.

Functional requirements can be described as requirements defining what the system to be built should do. Non-functional requirements are a set of requirements that determine how the system works.

## 5.1 Requirements Gathering Techniques

During the requirements gathering process there are numerous techniques that can be used. For this project; the project stakeholders where consulted during a brainstorming session on how the chatbot should operate.

Details from the brainstorming session are attached in Appendix A.

This project has taken a slightly different and less formal approach to requirements gathering. The reason for this is that the chatbot being developed for Applied Systems is based on the current quick quote solution already live on Applied System’s customer websites. It means that a lot of the requirements for the quick quote solution can be transferred to be applied to the chatbot.

## 5.2 Requirements Specification

After the requirements brainstorming session with the project stakeholders; a formal requirements document was produced.

This document was presented to all stakeholders for review. Any changes to the requirements were submitted by the stakeholders, the requirements document updated and sent back to the stakeholders for sign-off.

Once signed-off, should any changes need to be made to the requirements, a formal change request had to be submitted and evaluated on how it would impact the project and the project delivery date. A decision would then be made on whether the changed requirement could be accepted or not.

**TODO – get example of change request document**

### 5.2.1 Functional Requirements

|  |  |
| --- | --- |
| Requirement ID | Requirement Description |
| FR01 | User should be able to get a motor quote |
| FR02 | User should be able to get a home quote |
| FR03 | User should be able to retrieve a previous quote |
| FR04 | The bot should be able to look-up and display a user’s vehicle details based on the user registration |
| FR05 | The user should be able to request contact from the broker after receiving a quote |
| FR06 | User should receive an email with information about their quote after the quote is received |
| FR07 | The broker should receive an email with information about the users quote after a quote has been given to the user |
| FR08 | User should not be able to get a motor quote if under the age of 17 |
| FR09 | User should be able to change their answers during the workflow |

### 5.2.2 Non-functional Requirements

|  |  |
| --- | --- |
| Requirement ID | Requirement Description |
| NFR01 | The language of the bot should be professional and friendly |

## 5.3 Requirements Validation

Requirements validation is the process of ensuring that any requirements specified for a project are necessary to complete the project objectives and overall achieve its aim. Not validating requirements early in a project could lead to scope creep, unexpected costs or cause a project to miss its scheduled delivery date. Ultimately, the customer may not end up with the solution they wanted.

A valid requirement should have the following for characteristics:

* Correct – a requirement should accurately describe the required functionality
* Complete – a requirement should give enough information to a developer in order for the developer to implement the required functionality
* Technically achievable – a requirement should not be too ambitious that that developers cannot implement it
* Clear – a requirement should only have one possible outcome

During the requirement gathering phase of this project, all requirements were carefully considered and agreed upon in relation to the four characteristics noted above.

## 5.4 Risk Analysis

The purpose of a good risk analysis before a project begins is to help expose potential risks in a project at an early stage. It is important to identify the likelihood of the risk occurring and to hopefully find an early solution to remove or limit the risk. It is a key tool in project planning.

A qualitative risk analysis has been carried out on possible dangers to the project and collated to the risk register below.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Risk Id** | **Description** | **Probability of Occurrence** | **Preventive Measures** | **Severity** |
| 1 | Unrealistic time schedule | Medium | Incremental development, modify milestones, requirement prioritisation | High |
| 2 | Data loss | Low | Backup database regularly, secure repository for source code | High |
| 3 | User interface does not meet requirements | Low | Regular customer interaction, careful design, attention to requirements | Medium |
| 4 | Poor product performance | Medium | Ensure data is structured correctly, performant UI components and database | Medium |
| 5 | Scope creep | Medium | Set functionality boundaries, closely manage changing requirements | Medium |
| 6 | Lack of skills | Medium | Build software with skills I have, iterative builds to get more complex if time allows | Medium |

Figure - Risk Register

## 5.5 Data Protection

# 6 Project Plan

To give the project the best chance of success, a Project Plan has been developed to set out defined expectations and project deliverables.

The Project Plan also consists of a list of project milestones and a Work Breakdown Structure.

## 6.1 Project Scope

The delivered system should consist of a chatbot that can run on Facebook Messenger.

It should be able to interact with a SQL database and a service that will log errors.

A user of the chatbot should be able to answer questions and receive a motor or home insurance quote based on the insurance product they have selected.

The chatbot will have to retrieve the quotes from rating services owned by Applied Systems. On retrieving these quotes the chatbot will display the quotes to the user. The user and the insurance broker should both receive an email detailing the information of the quote.

The project will need a service for recording errors to the database. These errors should be detailed enough to help the developer understand why the error happened so they can take action to prevent it from happening again.

## 6.2 Project Schedule

The project is scheduled to run for around 7 months, which breaks down in to estimated periods of 1 month of planning and design, 5 months of development and another month for testing and deployment.

### 6.2.1 Project Milestones

The project milestones have been added to the GitHub quoting-bot, available Appendices A-D.

The milestones selected outline important stages throughout the project and determine points at which significant amounts of work have been completed.

### 6.2.2 Work Breakdown Structure

As part of project planning, a Work Breakdown Structure, Appendix E, has been developed to give an oversight to which project milestones fall in each stage of the Software Development Life Cycle (SDLC).

## 6.3 Project Resources

To ensure delivery of this project, the following required resources have been identified:

* Development computer (PC and/or laptop)
* Microsoft Visual Studio
* Microsoft Bot Emulator
* SQL server

# 7 Design

When designing a chatbot the most important aspect to consider is the user experience of the bot. As bots can be built for specific platforms, e.g. Slack, Skype, Facebook Messenger, etc., the user interface of the bot will be determined by the platform. Instead the focus of design should be on the user experience.

Contributors at Microsoft (Velloso, Iqbal, & Standefer, 2017), suggest the following as key considerations to include in the bot design.

* Is a user problem solved by the bot in as few steps as possible?
* Is the user experience to solve the user problem better/faster/easier than a web or app alternative?
* Is the bot available on the platforms demanded by the user?
* Is using the bot intuitive?

A bot should be designed minimally but to have maximum impact (Sens, 2017). A bot should interact with a user through engaging conversation, all the while trying to accomplish the following two goals quickly and efficiently (Sens, 2017).

1. Engage and Collect – communicate with the user in a direct manner to collect the information that will identify their needs and wants.
2. Parse and Deliver – the information collected should then be immediately parsed so it can be used to deliver content back to the user to fulfil their needs.

A bot should be designed with a leading question to determine the intent of the user. This should then be followed by as many follow-on questions as is required to gather enough information to solve the user’s issue as quickly as possible.

## 7.1 Conversation Flows

Designing how the conversation between bot and user will flow is critically important to the user experience of the chatbot. Knowing why a chatbot is being built should be the starting point when designing the conversational flow (Maruti Techlabs, 2018).

The conversation will flow between structured and non-structured interactions (Maruti Techlabs, 2018). Structured interactions will give the user options and choices when replying to a question, whereas non-structured interactions are more open to free text answers or queries and the bot will be required to handle these responses or requests (Maruti Techlabs, 2018).

**TODO – design conversation flow**

## 7.2 Conversation Dialogs

The first interaction between the bot and the user is very important to the user experience (Velloso & Standefer, Design a bot's first user interaction, 2018). It is not recommended to open the conversation with an open question. The question should be asked with giving the user a choice of options as their answer.

As a conventional website or app uses screens for users to interact with, a bot uses dialogs. Dialogs give the developer the ability to separate out areas of functionality and give the conversation a flow. Dialogs will contain actions to redirect to other dialogs or for processing user input.

### 7.2.1 First Interaction

For this project, the first interaction from the bot is a welcoming message followed by a list of options showing the insurance products available to get a quote for.

### 7.2.2 Motor Dialog

Aim of the dialog is to obtain enough information from the user that the bot can then make a request for motor quotes and provide them to the user.

**TODO – add dialog flow**

### 7.2.3 Home Dialog

**TODO – add dialog flow**

## 7.3 User Interface

As the chatbot will be built for publishing on Facebook Messenger, the user interface will be determined by the styling of the Facebook Messenger application.

Leveraging on the capabilities of Facebook Messenger, the chatbot will use emoji to convey a friendlier persona to the end user. Emoji’s are displayed on the user interface by using Unicode values.

## 7.4 High-level System Architecture

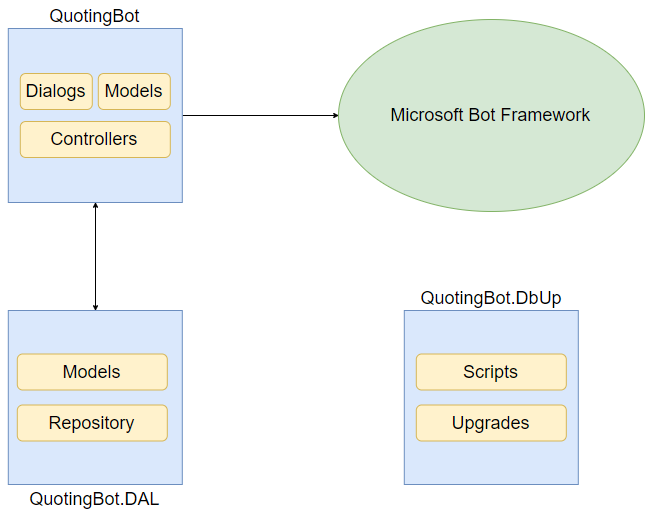


Figure - High level system architecture

## 7.5 Class Diagram

**TODO – generate when program completed**

## 7.6 Database Design

# 8 Implementation

## System Architecture

## Detail code

*Code Explanation*

*Libraries used*

# 9 Challenges and Solutions

# 10 Testing and Results

*Validation and verification*

*Evaluation of process and methods used to reach outcome*

# 11 Evaluation

Fulfilment of project objectives

# 12 Conclusions

# 13 Suggested Future Improvements to Project

Although the aim of the project was achieved, areas for future development and improvement of the bot have been identified.

All identified areas have been documented with recommendations made for how the work could be complicated and an estimation given on how long the work would be expected to take.

## 13.1 Link to Applied Systems eQuote Leads System

The Applied Systems eQuote Leads System is a system that collates all online business for an insurance broker. Currently this system is only populated with users, referred to as “leads”, and their data when they complete a quote insurance form via a broker’s website.

The chatbot functionality could be extended to insert a lead to eQuote Leads when a user has received a quote from the chatbot.

This would be achieved by making a call to an exposed endpoint in the eQuote Leads API for inserting a lead. Work would also need to be undertaken on the eQuote Leads system to extend the list of enumerations that determine the source of the lead. These enumerations could be extended to include ‘chatbot’ or ‘Facebook Messenger’ as the source.

An estimation on this work would be for it to take between 4 and 5 days to complete the implementation and testing.

## 13.2 Add Natural Language Processing

As this chatbot was built to solve a very definite problem, it was not so important to include a natural language processing service in this solution.

Should the chatbot be required to answer more “open” questions, introducing LUIS to the current solution would be a suitable method to handle these requests. Creating and training a LUIS model to handle user intents would give the chatbot the ability to deal with whatever the model is designed to handle.

The current solution would need to be extended to include a new dialog with methods that are “tagged” with an intent setup with the LUIS model.

It is hard to estimate a period for implementing a natural language processing service as it would be an on-going process to make sure the natural language processing model is trained to handle all intents required. However, the initial implementation of creating a model in LUIS with one or two intents and implementing the model to the chatbot solution would take no longer than 1 day (7.5 working hours).

## 13.3 Convert to JavaScript Widget

Another method of creating chatbots is using JavaScript. Doing this means the chatbot can be placed directly on a brokers site rather than going through a social media platform such as Facebook Messenger as this chatbot does.

To do this would be a major rework of the project and would be estimated to take months rather than days or weeks.

Converting to a widget for a website would require more planning and research. It may transpire from this research that another framework and development language would be better suited instead of using the Microsoft Bot Framework and C#.

# References

Agile Manifesto. (2017, 10 29). *Manifesto for Agile Software Development*. Retrieved from Agile Manifesto website: www.agilemanifesto.org

Amazon Web Services. (2018). *What is Source Control?* Retrieved April 16, 2018, from AWS: https://aws.amazon.com/devops/source-control/

Applied Systems. (2018, 04 11). *About Us - About Applied*. Retrieved from Applied Systems website: https://www1.appliedsystems.com/en-ie/about-us/about-applied/

Berry, D. (2017, March 03). *About Language Understanding (LUIS)*. Retrieved April 15, 2018, from Microsoft Azure: https://docs.microsoft.com/en-us/azure/cognitive-services/LUIS/Home

Boehm, B. W. (1988). *A Spiral Model of Software Development and.* TRW Defense Systems Group. Retrieved April 14, 2018, from http://csse.usc.edu/TECHRPTS/1988/usccse88-500/usccse88-500.pdf

Codecademy. (2018). *What is REST?* Retrieved April 16, 2018, from Codecademy: https://www.codecademy.com/articles/what-is-rest

Comodo. (2018). *What is HTTPS?* Retrieved April 16, 2018, from Instant SSL: https://www.instantssl.com/ssl-certificate-products/https.html

Conversable. (2018). *About Conversable*. Retrieved April 14, 2018, from Conversable: http://conversable.com/about/

DialogFlow. (2018, February 21). *Agents*. Retrieved April 15, 2018, from DialogFlow: https://dialogflow.com/docs/agents

Gordiyenko, S. (2014, November 3). *Software Development Life Cycle (SDLC)*. Retrieved April 14, 2018, from XB Software: https://xbsoftware.com/blog/software-development-life-cycle-waterfall-model/

Huckstep, R. (2017, December 13). *10 Insurtech Trends at the Insurance Crossroads 2018*. Retrieved April 13, 2018, from The Digital Insurer: https://www.the-digital-insurer.com/blog/10-insurtech-trends-insurance-2018/

Inflectra. (2018, March 1). *Requirements Gathering*. Retrieved April 16, 2018, from Inflectra: https://www.inflectra.com/ideas/topic/requirements-gathering.aspx

International Software Testing Qualifications Board. (2017, 10 29). *What is Spiral model- advantages, disadvantages and when to use it?* Retrieved from ISTQB website: http://istqbexamcertification.com/what-is-spiral-model-advantages-disadvantages-and-when-to-use-it/

ISTQB Exam Certification. (2018). *What is Agile model – advantages, disadvantages and when to use it?* Retrieved April 14, 2018, from ISTQB Exam Certification: http://istqbexamcertification.com/what-is-agile-model-advantages-disadvantages-and-when-to-use-it/

Lemonade. (2018). *Lemonade Renters and Home Insurance*. Retrieved April 15, 2018, from Lemonade: https://www.lemonade.com/

Lemonade. (2018). *Sign up for Lemonade Renters & Home Insurance*. Retrieved April 15, 2018, from Lemonade: https://www.lemonade.com/start/1

Lemonade. (2018). *The Lemonade Insurance API*. Retrieved April 15, 2018, from Lemonade: https://www.lemonade.com/api

Lengstorf, J. (2018). *JSON: What It Is, How It Works, & How to Use It*. Retrieved April 16, 2018, from Copter Labs: https://www.copterlabs.com/json-what-it-is-how-it-works-how-to-use-it/

Maruti Techlabs. (2018). *Complete Guide on Bot Frameworks*. Retrieved April 15, 2018, from Maruti Techlabs: https://www.marutitech.com/complete-guide-bot-frameworks/

Maruti Techlabs. (2018). *How to Design a Conversation for Chatbot?* Retrieved April 20, 2018, from Maruti Techlabs: https://www.marutitech.com/design-chatbot-conversation/

MedWhat. (2017). *About Us*. Retrieved April 14, 2018, from MedWhat: https://medwhat.com/about-us/index.html

Microsoft. (2017). *Mircosoft Bot Framework*. Retrieved April 15, 2018, from Bot Framework: https://dev.botframework.com/

Morse, B. (2017, April 20). *Become Part of the Story with the Marvel Comics Chatbot*. Retrieved April 14, 2018, from Marvel: https://news.marvel.com/comics/63529/become-part-story-marvel-comics-chatbot/

Moubarak, M. (2018, February 16). *Smart Chat For Your Real Estate Website: Qualify Buyers And Sellers in Real-Time With Roof.ai*. Retrieved April 15, 2018, from roof.ai: https://blog.roof.ai/roofai-real-estate-chatbot/

Mozilla. (2018, April 3). *What is JavaScript?* Retrieved April 16, 2018, from MDN Web Docs: https://developer.mozilla.org/en-US/docs/Learn/JavaScript/First\_steps/What\_is\_JavaScript

Ngan, K. (2018). *What is Git?* Retrieved April 16, 2018, from Microsoft Visual Studio: https://www.visualstudio.com/learn/what-is-git/

Nguyen, M.-H. (2017, October 6). *How artificial intelligence & machine learning produced robots we can talk to*. Retrieved April 14, 2018, from Business Insider UK: http://uk.businessinsider.com/what-is-chatbot-talking-ai-robot-chat-simulators-2017-10

PwC. (2017). *PwC Irish Total Retail 2017 Survey*. Retrieved April 13, 2018, from PwC Irish Total Retail 2017 Survey: https://www.pwc.ie/publications/2017/ten-retailer-investments-for-an-uncertain-future-total-retail-survey.pdf

Python. (2018). *What is Python? Executive Summary*. Retrieved April 16, 2018, from Python: https://www.python.org/doc/essays/blurb/

RoofAi. (2018). *Smart chat for your real estate website*. Retrieved April 14, 2018, from RoofAi: https://www.roof.ai/

Rouse, M. (2010, April). *Ruby*. Retrieved April 16, 2018, from Whatis.com: https://whatis.techtarget.com/definition/Ruby

Rouse, M. (2017, August). *Microsoft SQL Server*. Retrieved April 16, 2018, from TechTarget: https://searchsqlserver.techtarget.com/definition/SQL-Server

Ruby. (2018). *Ruby is...* Retrieved April 16, 2018, from Ruby: https://www.ruby-lang.org/en/documentation/

Samit, J. (2017, December 26). *4 Technology Trends That Will Transform Our World in 2018*. Retrieved April 13, 2018, from Fortune: http://fortune.com/2017/12/26/4-technology-trends-2018/

Sens, R. (2017, June 13). *Want to design better chatbot dialog?* Retrieved April 17, 2018, from Medium - Prototypr: https://blog.prototypr.io/design-better-chatbot-dialog-using-the-five-ws-framework-628bda212f39

Stackify. (2017, 10 04). *What is SDLC? Understand the Software Development Life Cycle*. Retrieved from Stackify website: https://stackify.com/what-is-sdlc/

Techopedia. (2018). *What is a Relational Database (RDB)?* Retrieved April 16, 2018, from Techopedia: https://www.techopedia.com/definition/1234/relational-database-rdb

The Keyes Company. (2018). *Florida Real Estate & Homes For Sale*. Retrieved April 15, 2018, from The Keyes Company: https://www.keyes.com/

Tutorials Point. (2017, 10 29). *SDLC Agile Model*. Retrieved from Tutorials Point website: https://www.tutorialspoint.com/sdlc/sdlc\_agile\_model.htm

Tutorials Point. (2017, 10 29). *SDLC Overview*. Retrieved from TutorialsPoint: https://www.tutorialspoint.com/sdlc/sdlc\_overview.htm

Velloso, M., & Standefer, R. (2018, December 13). *Design a bot's first user interaction*. Retrieved April 17, 2018, from Microsoft Azure: https://docs.microsoft.com/en-us/azure/bot-service/bot-service-design-first-interaction

Velloso, M., Iqbal, K., & Standefer, R. (2017, Decmber 13). *Principles of bot design*. Retrieved April 17, 2018, from Microsoft Azure: https://docs.microsoft.com/en-us/azure/bot-service/bot-service-design-principles

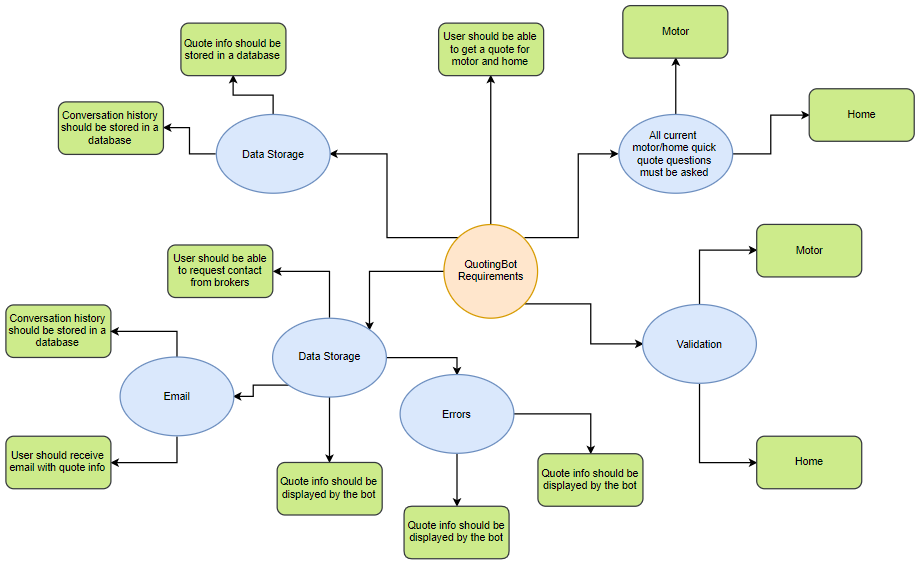
Wagner, B., Wenzel, M., & Levin, I. (2015, July 20). *Introduction to the C# Language and the .NET Framework*. Retrieved April 16, 2018, from Microsoft: https://docs.microsoft.com/en-us/dotnet/csharp/getting-started/introduction-to-the-csharp-language-and-the-net-framework

Wikipedia. (2018, April 3). *Software framework*. Retrieved April 15, 2018, from Wikipedia: https://en.wikipedia.org/wiki/Software\_framework

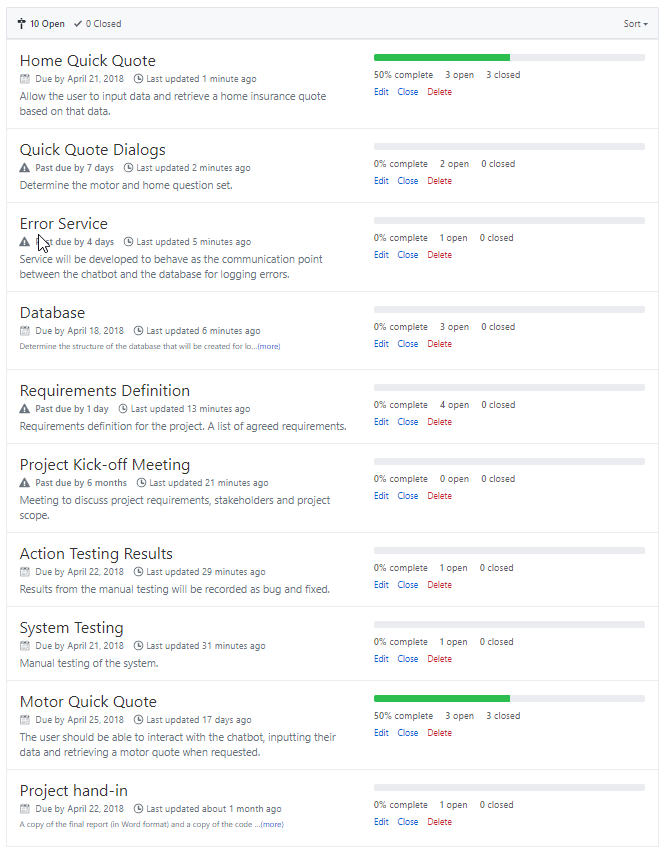
Wissner-Levy, Y. (2016, September). *How Lemonade is Fixing a Broken System*. Retrieved April 15, 2018, from Lemonade: https://www.lemonade.com/blog/hello-world/

wit.ai. (2018). *Wit.ai*. Retrieved April 15, 2018, from wit.ai: https://wit.ai/

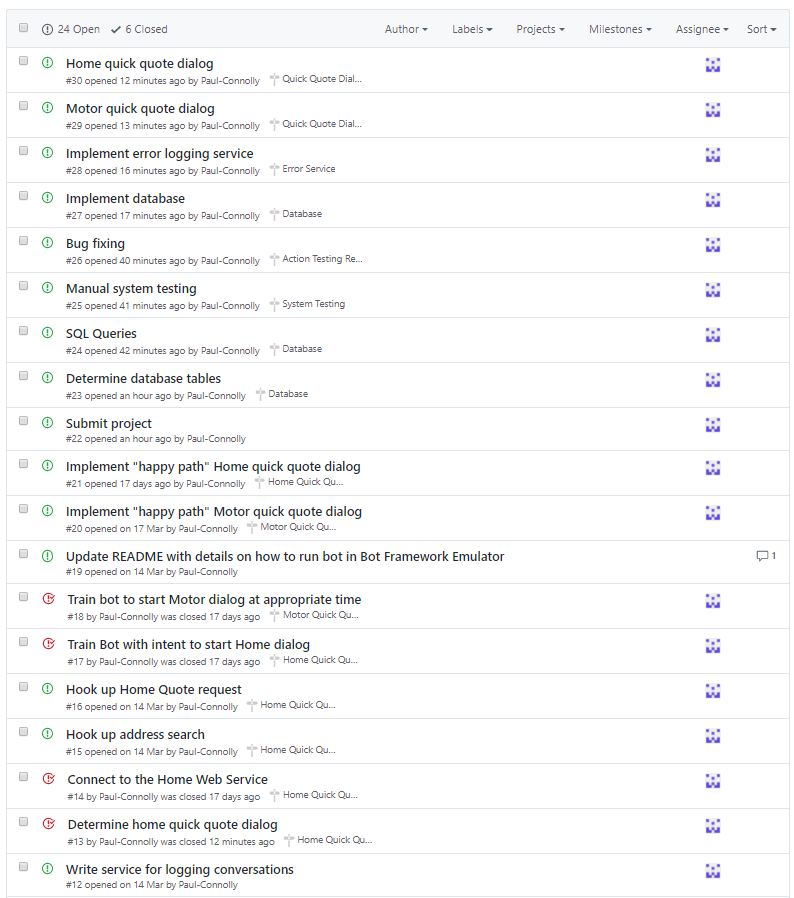
# Appendices



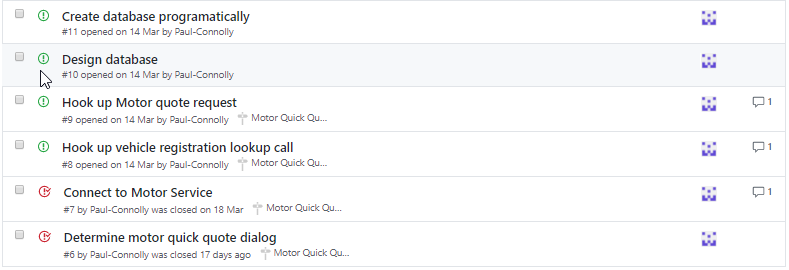
Appendix - Brainstorming from requirements gathering



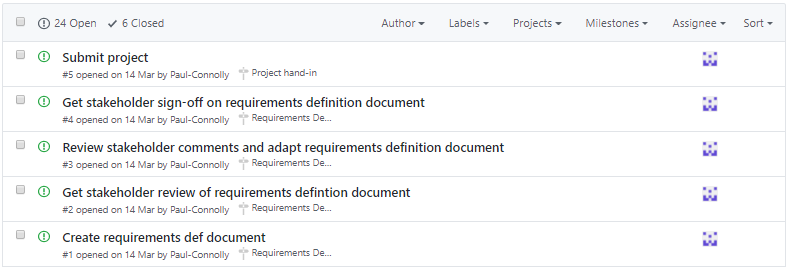
Appendix – Project milestones taken from GitHub



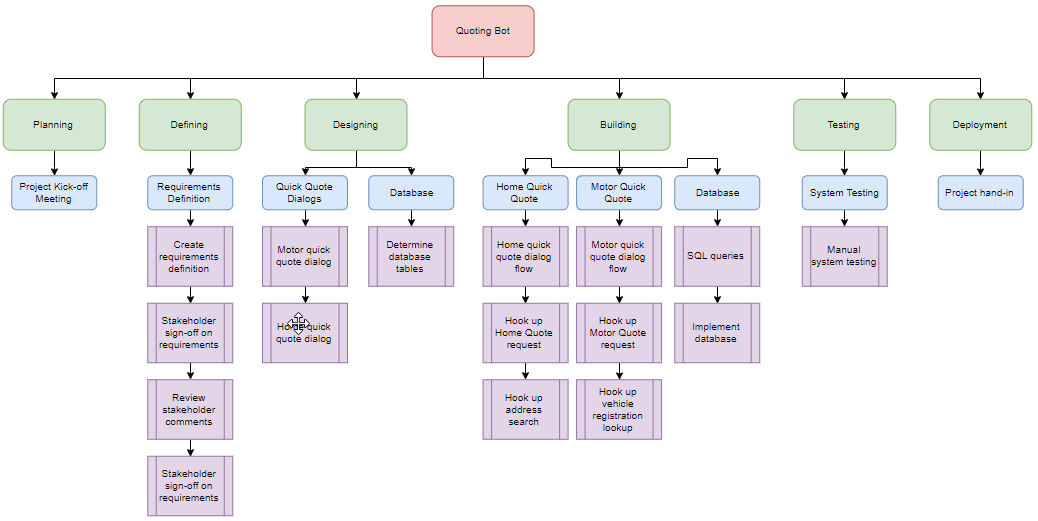
Appendix - Issues from GitHub



Appendix - Issues from GitHub cont.



Appendix - Issues from GitHub



Appendix - Work Breakdown Structure

1. A ‘quick quote’ is a reduced set of questions that still allows insurance quotes to be returned when requested. For questions not asked, default answers are set. [↑](#footnote-ref-1)