WPI

BNP PARIBAS

BNP Paribas: Analyzing Technical Alternatives of Automating Locate Requests

A Major Qualifying Project Report

Submitted to the Faculty of
Worcester Polytechnic Institute
In Partial Fulfillment of the Requirements for
the Degree of Bachelor of Science

Submitted By:
Yaofeng Wang, Computer Science and Industrial Engineering

Project Advisor:
Professor Renata Konrad, Foisie School of Business

Submitted on:
March 1st, 2019

Abstract

The Stock Lending and Borrowing (SLAB) desk of BNP Paribas was looking to implement a chatbot on Symphony, a secure, cloud-based, communication and content sharing platform, to alleviate all the manual burden in SLAB trading. My team designed and implemented the chatbot in B term, 2018, but it lacked several functionalities. In this project, I designed an improved chatbot. I then analyzed and compared the old technology, the chatbot implemented in B term and the improved chatbot to provide a recommendation for BNP regarding the chatbot implementation.

Executive Summary

My MQP sponsor, BNP Paribas, was looking to implement a chatbot on the company's Symphony platform. The chatbot is able to automate the process of "short sale locate request" under Stock Lending and Borrowing (SLAB) Desk of BNP. The main functionality of the chatbot is to alleviate all manual burden of the process, manage, and store all trading data in a system for future reference.

My MQP team implemented the chatbot in B term 2018 on-site in the BNP NYC office. The chatbot is able to receive commands of requests from users and process the locate request automatically. The new locate request process can be divided into four parts: 1. Chatbot receives a request message from the user, 2. Chatbot sends the request to SLAB software, 3. Chatbot receives the response from SLAB software, and 4. Chatbot sends the result of the request to the user. The chatbot is also able to search for previous requests and results by specific fields of data and maintains all record in its history of messages. For the details of the chatbot and the project, please see the B term report (Nicholson, Wang, & Chen, 2019). The diagrams of the previous chatbot can be found in Appendix A of this report. The list of functionalities and screenshots of the interface is in Appendix B of this report.

In this project, I designed a new chatbot that improves the design of the previous one (Chapter 4). It includes more functionalities, higher efficiency and adds the technology of natural language processing. This improved chatbot is able to accomplish all goals outlined by BNP, but it has a long development time with a higher cost. Whether the new design should be implemented or not is undetermined.

Because the old technology, the chatbot implemented in B term, and the improved chatbot all have advantages and disadvantages, I conducted research to compare and analyze the three alternatives using two methods from Multi-Attributes Decision Techniques. The two methods are Simple Multi-Attribute Rating Techniques (SMART) and Analytical Hierarchy Process (AHP). To analyze the three alternatives, I identified seven attributes important in the decision making of choosing amongst the alternatives. After that, I gathered the preference weights for the attributes from people who previously participated in the project. Last, I plugged the data into SMART and AHP to generate a data-driven recommendation for BNP.

Based on the results from the two methods, I recommended BNP select the improved chatbot. Because of limited project scope, I also provided suggestions for future developments and researches.

Table of Contents

List of Figures

List of Tables

Chapter 1: Introduction

The Stock Lending and Borrowing (SLAB) desk of BNP Paribas was looking to implement a chatbot on the company's Symphony platform (Symphony is a chatting software). The goal of the chatbot was to automate the locate approval process of the equity short sale when requests are received in a Symphony chat room. The chatbot can alleviate all manual burden in the process of "locate request" and maintains a record of all requests and approval information in the system. The concept of SLAB and the detailed process of the chatbot will be discussed in Chapter 2.

From October to December, 2018, my MQP team designed and implemented the chatbot in our sponsor BNP's software repository. The team was comprised of Matthew Nicholson (MGE), Yiyi Chen (IE), and myself. We worked in the BNP NYC office over the course of eight weeks. On December 12th, 2018, the chatbot was successfully installed in the Symphony chatting platform and connected to Falcon software. The chatbot was coded in Python and interacted with SOAP API and REST API. The chatbot has four main processes:

1. Receive a message from user to understand the trader's need,
2. Send the request to Falcon SOAP API,
3. Receive the response from Falcon,
4. Send message to user with all information in human readable format.

For detailed information regarding this project, please see the previous report (Nicholson, Wang, & Chen, 2019). As a dual Computer Science (CS) and Industrial Engineering (IE) major, the NYC portion of the project fulfills the requirements for my CS degree, and this project seeks to satisfy the requirements for my IE degree.

For the B-term MQP project, my team satisfied all the needs of our sponsor, but the chatbot my team implemented has limited functionalities. First, the Falcon WebService that we used inside the chatbot does not have a comprehensive list of methods that allows the chatbot to replace all manual steps in a locate request process. Second, the chatbot itself does not wrap up all methods in Falcon because of the limited amount of time we had in B term. If more methods were to be implemented, the chatbot could perform more efficiently and powerfully.

Because of the identified limitation of the newly designed chatbot, with this project I continued to improve the chatbot which wraps up more methods in Falcon, performs all functions the current chatbot has, and works more efficiently. To provide BNP with a recommendation which satisfies its need to fully automate the locate approval process, I conducted a comparison and analysis of the three alternatives: the current manual process, the chatbot implemented in October, 2018, and this most recently improved chatbot.

Project Goal:

Analytically compare three alternatives to provide a recommendation for BNP regarding the automation of its SLAB Trading. The three proposed alternatives are:

1) old technology;
2) the chatbot implemented in B term, and

3) the new alternative of the most recently improved chatbot and Falcon webservice.

The objectives to accomplish the project goal are:

1) Define alternatives to address the problem.
2) Identify attributes important in comparing alternatives.
3) Use Simple Multi-Attribute Rating Technique (SMART) and (Analytical Hierarchy Process) AHP to compare alternatives
4) Develop results from the decision analysis methods in Step 3 base on on-site experience, stakeholder's suggestions, and arithmetic calculation.
5) Verify if the results in Step 4 lead to the same outcome.
6) Identify the advantages and disadvantages of each alternative.
7) Develop a conclusion and recommendation based on analysis.

Chapter 2: Background

2.1 Symphony Chat Platform

Bloomberg financial software provides real-time information of all the world markets and allows instant messaging through application online and mobile devices. This feature enables Bloomberg to dominate in the financial sector. However, in 2014, Bloomberg reporters were accused of prying activity of terminal users (Symphony 2018). This situation led financial institutions to invest in something new and more secure.

Symphony was that investment. Symphony is a secure, cloud-based, communication and content sharing platform. Symphony enables businesses to provide best-of-breed collaboration technology to their employees, while driving efficiency and ensuring security and compliance. Enterprise customers can now centralize all workflow in a single platform, using Symphony as a replacement for traditional email and voice systems to securely communicate with internal and external teams, share documents and content, and conduct meetings with conferencing and screen-sharing. Symphony's success is a direct result of its powerful platform, customizable user experience, and open partner ecosystem which delivers a large and growing market of applications and integrations (Symphony Blog, n.d.).

"Digital transformation is central to BNP Paribas Global Markets' strategy, and collaboration with new financial technology as a crucial part of that process. Forming agile partnerships with exciting and innovative companies like Symphony helps us deliver an exceptional service to clients, and remain their partner of choice in a changing world," said Olivier Osty, Executive Head of Global Markets, BNP Paribas (Symphony, 2018) (Nicholson, Wang, & Chen, 2019).

2.2 Short Sale Locate Request

A locate request is a message sent to the SLAB desk that helps execute short sales. A short sale is the sale of an asset (securities and other financial instruments) that the seller does not own. The seller affects such a sale by borrowing the asset to deliver it to the buyer. To execute a short sale, an equity trader needs to borrow shares that are held within the market. To get these borrowed shares, equity traders must communicate with the SLAB desk because they manage all the shares available for borrowing. SLAB traders use a database that lists the number of shares available for borrowing when determining their decision. When a request comes in from a trader, SLAB processes the request, determines if the shares are available, and then reports back to the trader whether the request has been approved, partially approved, or declined. They will also give the trader the rate at which the shares will be borrowed. The rates change depending on if the request is easy or hard to borrow. The harder it is to find shares in the market the higher the rate will be to borrow them (Nicholson, Wang, & Chen, 2019).

2.3 The Previous Chatbot

From October 2018 to December 2018, my team worked on the project to create a chatbot that meets all needs brought by our sponsor BNP. The goal of the project was to create a

chatbot that could improve the speed at which traders conduct short sales. Although we didn't have concrete testing completed, our sponsors believe that what we have created is much faster and easier to use.

We created a chatbot that is very simple to use by focusing the majority of our time constructing a strong natural language processing (NLP) component. The natural language processing allows for multiple inputs and asks users to try again if they manage to not input messages into the chatbot correctly. Our chatbot gives simple yet specific instructions on what to type in and how to get what you're looking for. Most importantly, the chatbot is easy to use when conducting short sales. So much so, users can execute short sales in under ten seconds on certain easy to borrow requests. On top of that its friendly in its own way. For example, if you send the message "bye" to our chatbot, it will respond "bye, bye". If a user says "Bonjour" it will prompt a "Hello" message.

We also created three special features that were not requirements for our project. We created them because they made the chatbot more likely to get adopted among the workers at BNP. By adding features that would make the chatbot easier and more practical, we feel the special features were a significant addition to the final project. More details relating to the project and chatbot functionalities can be referred to the previous report.

All diagrams related to the previous chatbot can be found in Appendix A of this report. The screenshots of several functionalities of the chatbot can be found in Appendix B.

Below are the summarizations for the special features of the previous project.

1. Status: Asking the chatbot to find one specific short sale request using and ID number.
2. Status ALL: Asking the chatbot to find all the requests made that day by that specific user.
3. Inputting multiple trades: Asking the chatbot to perform a short sale locate request for multiple stocks that are separated by a ';'.

However, although the chatbot met all business requirements of our sponsor, we had some recommendations on-site after communication with the project stakeholders. First, we recommend adding a cancel feature into the bot. Currently, we focused on GetApproval(), QueryApproval() and GetAvailability() methods from Falcon Webservice, which are within the 14 methods of Falcon. Falcon does not currently include a cancel method. When considering the circumstance when clients type the wrong equity or number of shares, or if he or she doesn't want to wait anymore, the user should be able to cancel. If the cancel method is added to Falcon, clients are able to terminate the request while waiting for the response.

Moreover, we have developed and utilized two Falcon methods GetApproval() and QueryApproval(), other twelve built-in functions should be added into the chatbot. Furthermore, it would be good to explore SLAB functionalities. For instance, the chatbot could provide approval rate comparisons within the last three transaction days and classification service, which is supported by the database at the backend.

Lastly, the NLP needs to be further developed. By adding more language processing options into the framework, clients will be able to have a better user experience by decreasing the number of errors during conversations.

By adding the features described above, the chatbot can be more functional, perform more efficiently and communicate like a human-being (Nicholson, Wang, & Chen, 2019).

Chapter 3: Methodology

3.1 Multi-Attribute Decision Techniques (MADT)

Multi-Attribute Decision Techniques is a set of methods for decision makers to make a decision that solves a particular problem or improves a situation. It is a branch of Multi-Criteria Decision Making (MCDM). MADT focuses on solving problems with a set of discrete decision alternatives. Methods in MADT are widely used for different scenarios and fields, but most methods share things in common. Most of them solve problems that have multiple alternatives and multiple attributes to be considered for decision making. They gather information to create matrices of preferences and priorities. They also use the matrices to generate weights of attributes and alternatives (Triantaphyllou, 2000).

Multi-Attribute Decision Techniques include SMART, SMARTER, and AHP, among others. In this project, I used SMART and AHP to conduct a compare and analysis of the alternatives in order to provide the best solution. Steps and examples of using the two methods will be discussed in the following sections.

3.1.1 Simple Multi-Attribute Rating Technique (SMART)

SMART is used when a decision maker has multiple objectives. Because those objectives often conflict with one another, it is difficult to find an alternative that meets all objectives. A decision analyst, such as myself, by using the SMART technique rates the alternatives and attributes and recommends an alternative that best meets the decision maker's objectives.

SMART has eight steps:

1. Identify decision maker(s);
2. Identify alternative courses of action;
3. Identify the relevant attributes;
4. Measure the performance of the alternatives on that attribute;
5. Determine a weight for each attribute;
6. Calculate the weighted average of values;
7. Make a provisional decision;
8. Perform sensitivity analysis.

Identifying the relevant attributes in step 3 is crucial in this method. By splitting a problem statement into several attributes and criteria, the decision maker is able to rate each attribute according to their preference without any influence of other criteria. To do this effectively, the attributes should be complete, decomposable, and as small as possible. Furthermore, the scale of the attributes must be operational, and there should not be any redundant and repetitive attributes. The weighted attributes influence the outcome of the decision (Edwards & Barron, 1994).

To determine the weights for the attributes, a decision maker should first rate each attribute on a scale of 0-100. The preference towards a particular attribute should consider both

personal preferences as well as the overall goal of the decision. A group of normalized weights is calculated by getting the percentage of each weights in total value of the weights and times 100. Take for example the decision regarding purchasing a house. There are four alternatives to be considered- House A, B, C, and D. The decision maker may identify six attributes important in the decision- proximity to work, visibility, image, size, comfort, and car-parking facilities. Table 1 shows an example of how weights may look like for six attributes deemed important in a decision regarding selecting a location for a house. The third column in Table 1 illustrates how normalized weights are calculated.

Attribute	Original Weights	Normalized Weights (rounded)
Proximity to work	100	32
Visibility	80	26
Image	70	23
Size	30	10
Comfort	20	6
Car-parking facilities	10	3

Table 1 - Original Weights and Normalized Weights for Housing Attributes (SMART)

After normalizing weights, the scores of the four alternatives can be calculated based on the performance of the alternative on each attribute. First, decision makers should rate each alternative by the attributes. Figure 1 shows an example of how a decision maker rated, for the four housing alternatives based on the "house image" attribute.

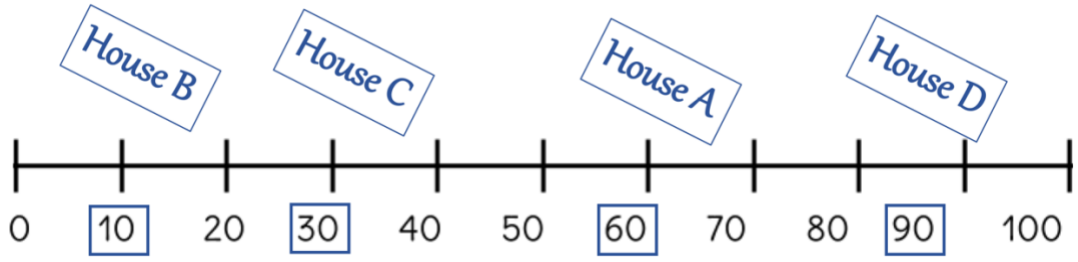

Figure 1- Rates of the Housing Alternatives (SMART)

Finally, we can compute the score of each alternative by multiplying the rate of the attributes and its corresponding normalized weight. The alternative with the highest score is the best decision according to the SMART method (Edwards & Barron, 1994).

3.1.2 Analytical Hierarchy Process (AHP)

AHP allows a decision maker to measure different attributes based on preferences, feelings, and satisfaction. The core component of AHP is the comparison of each pair of attributes rather than ranking them. The steps of AHP are the following:

1. Define the goal of the decision

2. Structure the decision problem in a hierarchy
3. Pair comparison of criteria in each category
4. Calculate the priorities and a consistency index
5. Evaluate alternatives according to the priorities identified.

To analyze each alternative, we need to find the weight of each attributes by our preference. Typically, a score between 1 and 9 is assigned according to the following scale:

1: Objectives i and j are equally important
3: Objective i is slightly more important than j
5: Objective i is strongly more important than j
7: Objective i is very strongly more important than j
9: Objective i is absolutely more important than j

To record the rates, we put the values in a n-by-n matrix (n is the number of attributes). For each row(i) and each column(j), the rates are stored in their corresponding a_{ij} elements. When a_{ij} is filled, the value a_{ji} is the reciprocal of a_{ij}. Take for example the decision to purchase a car. The decision maker may identify four attributes that are important- mileage, warranty, purchase cost, and trunk space. Figure 2 is an example of rating the attributes based on the decision maker's experience and preferences.

Mileage / Warranty /Cost / Trunk Space

$$A = \begin{bmatrix} 1 & 5 & 2 & 4 \\ 1/5 & 1 & 1/2 & 1/2 \\ 1/2 & 2 & 1 & 2 \\ 1/4 & 2 & 1/2 & 1 \end{bmatrix} \begin{matrix} \text{Mileage} \\ \text{Warranty} \\ \text{Cost} \\ \text{Trunk Space} \end{matrix}$$

Figure 2 - Rates of Car Attributes (AHP)

To obtain the weights of each attributes, we need to transforming the matrix into a numerical matrix. For example, matrix A in Figure 2 is transformed into matrix A_{norm} as shown in Figure 3. The values in A_{norm} are calculated by getting the percentage of each rate from the sum of its corresponding column.

$$A_{norm} = \begin{bmatrix} 0.52 & 0.50 & 0.50 & 0.53 \\ 0.10 & 0.10 & 0.10 & 0.07 \\ 0.25 & 0.20 & 0.25 & 0.267 \\ 0.13 & 0.20 & 0.13 & 0.13 \end{bmatrix}$$

Figure 3 - Transformed A-norm Matrix from Rates of the attributes (AHP)

The weights are obtained by calculating the average value of each row in $A_{norm.}$ Figure 4 shows the arithmetic calculation that get the weights for the four attributes.

$$w_1 = \frac{0.51+0.50+0.50+0.53}{4} = 0.51$$

$$w_2 = \frac{0.10+0.10+0.13+0.07}{4} = 0.10$$

$$w_3 = \frac{0.26+0.20+0.25+0.27}{4} = 0.24$$

$$w_4 = \frac{0.13+0.20+0.13+0.13}{4} = 0.15$$

Figure 4 - Example of Calculating Weights of the Attributes (AHP)

Before the weights can be used, we need to look at the inconsistency ratio. If the inconsistency ratio is less than 0.1, the scores assigned by the decision maker are considered to be consistent. The inconsistency ratio can be measured by the following two formulas where random consistency index (RI) can be obtained by looking at Table 2 for the specific n value:

Inconsistency Index = (average ratio – n) / (n-1)

Inconsistency Ratio = Inconsistency Index / Random Consistency Index

n	1	2	3	4	5	6	7	8	9	10
RI	0.00	0.00	0.58	0.9	1.12	1.24	1.32	1.41	1.46	1.49

Table 2 - Random Consistency Index Table

The final step is to calculate the score for each alternative. By researching values for each attribute by alternative, the decision analyst presents the decision maker with information. Using the car example introduced earlier, the decision maker has information related to mileage for Car A and Car B, for instance. The user is then asked, on the scale from 1-9 defined earlier, how does Car A compare to Car B in terms of mileage. The analyst generates a rating for each alternative by each attribute. Multiplying the rates by their corresponding weights, we obtain the scores that represent the performance of the alternatives by the attributes. The alternative with the maximum sum of the rates is considered as the best alternative (Triantaphyllou, 2000).

3.2 Software Development Tools

Below are the technology tools my team used to develop the previous chatbot. The tools are also considered in this project to design the new improved chatbot.

3.2.1 Python

Python is a high-level programming language which includes a comprehensive library of modules and packages (McGrath, 2013). Python is the best programming language to be used in

this project because it has a small learning curve. Its project structure is simply set up, and it has straightforward methods to post and get requests so that we could communicate with APIs in a convenient manner. The definition of API and how we are going interact with will be explained in Section 3.2.2. My team used Python 2.7 to develop the chatbot made in October. We wrote our main back-end source code which interacts with Symphony REST API and SOAP API. The source code includes a chatbot framework, a bridge that connects the chatbot and Symphony REST API, and workers work within the bot. Code is written in PyCharm IDE and programs are tested through Anaconda Prompt before being pushed into Git version control environment. All the environments are explained in details in previous report.

All third-party Python libraries we used in the previous project were accessed through Pypi Pip and were installed from BNP Artifactory library repo. Pip is a package manager to pull and install any library the program needs from a library repo webpage (W3School, 2018). The libraries we used were Logging, Suds, and Suds.wsse. Logging is a package to print messages on backend console while at the same time running code in PyCharm. It is useful to know what priority the message has, or where the message originates from. In this project specifically, it is used to log information of debugging issues. Suds allows the program to create a client with a valid Web Service Description Language (WSDL) path. Web Service Security (wsse) is the sub-library of suds that is able to create a soap-based object with a security token header and an XML message body (Nicholson, Wang, & Chen, 2019).

3.2.2 API

An Application Programming Interface (API) allows developers to get methods from an external software or website and utilize in their own application. API makes the use of eternal methods easier because developers do not need to know how the methods are created and how the logic inside of them looks like (Hoffman, 2018). My team used several APIs to communicate with our chatbot in order to interact with third-party software and develop our chatbot within the software. Similar to the process described in the previous section, the old chatbot communicates with Symphony software through Symphony API, and then through Falcon with Soap API in XML message. The Falcon software itself also interacts with Falcon API in order to achieve more BNP business purposes. The APIs we used will be discussed in the following sections (Nicholson, Wang, & Chen, 2019).

3.2.2.1 Symphony API

Symphony provides Extension API and RESTful API (Representational State Transfer). Extension API allows developers to build apps embedded within the Symphony's user interface, and the REST API gives developers access to all applications required to build a chatbot. We used REST API in this project to create a chatbot that can interact with Symphony interface and interpret the message input from Symphony chat room. REST API is implemented on several physical interfaces. The Pod is a cloud-based Symphony infrastructure, the Key Manager encrypts key messages from users, and the Agent encrypts and decrypts services for applications.

BNPP has many new functions and directories that would like to be added upon the original Symphony software, for example: directory of employees, explanation of acronyms, applications, and bots. To do so, each function or application is built through programs in back-end code, and all programs call the Symphony API to connect with the Symphony software. The goal of calling Symphony API is to implement all newly-added functionalities onto the user's side of the Symphony software. The software automatically executes the functionalities with Symphony build-in front-end user interface without need to code and understand how to code it (Symphony, 2018) (Nicholson, Wang, & Chen, 2019).

3.2.2.2 SOAP/Falcon API

SOAP API is a bridge between our chatbot and the Falcon software. SOAP API sends message using XML file format with columns storing the message type, attributes, and place to hold message content (Wodehouse, 2018). In our project, we used SOAP to transfer data that includes client and request information. Figure 5 shows the structure of constructing a SOAP-XML object (Nicholson, Wang, & Chen, 2019).

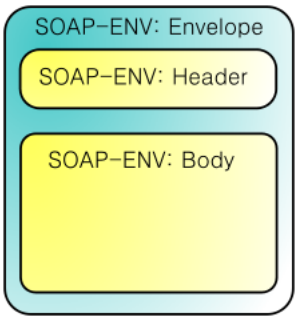

Figure 5 - Structure of a SOAP XML Message

Chapter 4: The Improved Chatbot

In this chapter, the design of the improved chatbot will be analyzed and discussed. The core improvements are implementing more Falcon methods in the chatbot, adding a cancel feature, and embedding a more intelligent Natural Language Processing (NLP) technology.

4.1 New Falcon Methods Added into the Chatbot

4.1.1 GetAvailability()

This is the first step of the process of the system after a user submits a locate request. To maximize the efficiency of processing, the chatbot first sends a getAvailability() method before a getApproval() request is sent to Falcon and Slab traders. GetAvailability() aims to make sure that the stock requested and the number of shares of this stock are currently available to be borrowed. If it is available, then the system will automatically process to the next step: getApproval(). If it is not available, the system will prompt back to the user immediately that this request cannot be processed because of the unavailability of the stock. This step saves the total time of the process greatly and allow users to try the others earlier.

4.1.2 GetIndicatedRate()

This is the second step of the process happens after checking the availability of the stock. If the stock is available to be borrowed, a indicated rate can be queried by using getIndicatedRate(). The chatbot then sends back the rate to the user and asks if he or she accepts the rate to be borrowed later. If the user accepts, the system will proceed to getApproval() step, and if the user declines the rate, the process ends. This method is helpful in the system to avoid the situation when the borrowed stock does not hold the rate users want. Users are able to borrow freely by determining if the rate is proper before issuing the locate request.

4.2 Cancel Feature

The cancel feature is a proposed functionality of the chatbot. It is beneficial if the Falcon WebService has a cancelRequest() method added in order to terminate the method of getApproval(). The method is extremely beneficial if the locate request sent is a hard-to-borrow case, in which an approval response is manually placed by SLAB traders. Although the process of a locate request can be terminated from the side of the chatbot and the Symphony platform, it would not influence the record of SLAB desk, and the process of a locate request inside Falcon is still processing until SLAB traders give a decision to the request.

The proposed cancelRequest() method takes in requestID and userID, and returns the status of the request. It first queries the particular request by requestID as the parameter of the method; then it checks if the userID that issues cancelRequest() method is the same userID that issues the request; finally, it changes the status of the request as “Completed” and writes the result as “User terminates the request.”

Inside the chatbot, cancelRequest() method can be triggered if the request is hard-to-borrow. When the user waits for 20 minutes without a response, the chatbot asks if he or she wants to cancel the request and work on another one. The user has the choice to opt out or stay. Because the process is implemented in multi-threads that can run multiple requests, one hard-to-borrow case does not influence the other cases. However, if a user opts out a hard-to-borrow, the efficiency of processing the other requests can be easier because it reduces the total load of the system. Users get the chance to cancel the request at every 20 minutes of their wait time.

4.3 Natural Language Processing (NLP)

Natural Language Processing is a field in Artificial Intelligence that focus on allowing machines to imitate human-beings by being able to interpret, compose, and communicate in human languages. To implement NLP functionality, a system or a software usually needs to import several libraries of language models. By learning the models, the system or the software is able to expand its language knowledge base to interpret, compose. and communicate more powerfully.

Adding NLP on the previous chatbot is accessible because the previous chatbot was coded in Python. NLP has a library called "spaCy" that is implemented in Python, so it is convenient to improve the chatbot by adding the libraries on the existing software. Figure 6 shows the steps to download spaCy in Python using pip3 tool. Figure 7 shows a simple example of using spaCy to interpret a line of sentence (Seif, 2018).

```
### Installing spaCy, general Python NLP lib
pip3 install spacy

### Downloading the English dictionary model for spaCy
python3 -m spacy download en_core_web_lg

### Installing textacy, basically a useful add-on to spaCy
pip3 install textacy
```

Figure 6 - Installing Python NLP Libraries (Seif, 2018)

```python
# coding: utf-8

import spacy

### Load spaCy's English NLP model
nlp = spacy.load('en_core_web_lg')

### The text we want to examine
text = "Amazon.com, Inc., doing business as Amazon, is an American electronic commerce and cloud co

### Parse the text with spaCy
### Our 'document' variable now contains a parsed version of text.
document = nlp(text)

### print out all the named entities that were detected
for entity in document.ents:
    print(entity.text, entity.label_)
```

Figure 7 - Load and use spaCy in Python (Seif, 2018)

By implementing NLP, the chatbot is able to interpret most well-organized messages sent from users so that users do not need to understand or search for commands. Because the input messages become more flexible by using NLP, the chatbot is able to perform more functionalities by interpreting a great number of users demands and responding with more accurate and relevant answers.

4.4 The Improved Chatbot Flowchart

Figure 8 is the flowchart of the improved locate request process implemented inside of the chatbot. It describes the order and logic of using the methods from Falcon WebService. The whole process starts with initiating the chatbot and prompts the user to start a locate request. There are several possible evolutions of scenarios, but all situations end on sending a message: “Transaction end”.

The situations that might occur in this system are:

1. The stock is not available for borrowing.
2. The stock is available, but the rate is not accepted by the user.
3. The stock is available, and the rate is accepted by the user. The locate request is not approved by the Falcon system or the SLAB traders.
4. The locate request is immediately approved by the SLAB traders (Easy-to-borrow).
5. The locate request is not approved or declined immediately (Hard-to-borrow). The request is declined within 20 minutes.
6. The hard-to-borrow request is approved within 20 minutes.
7. The hard-to-borrow request does not have a response within 20 minutes. User chooses to cancel the request.

8. The hard-to-borrow request does not have a response within 20 minutes. User chooses to continue waiting. The request is approved after 20 minutes.
9. The request is declined after 20 minutes.
10. The request does not have a response in more time. The user chooses to cancel the request.

All situations finally point to the end of the process as described in Figure 8.

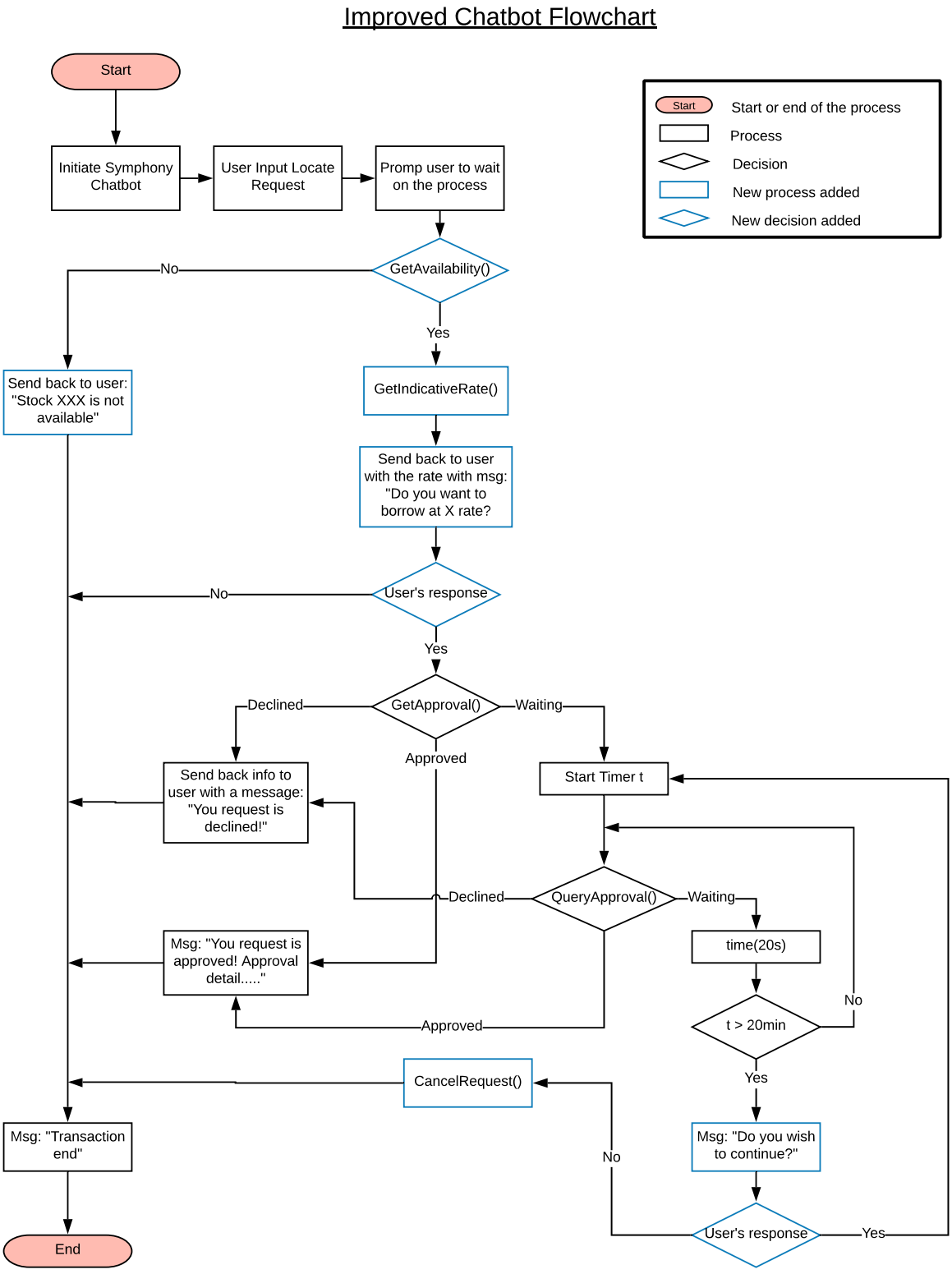

Figure 8 - Flowchart of the Process of Locate Request in the Improved Chatbot

Chapter 5: Analysis and Result

5.1 Alternatives and Attributes

In this section, I will discuss the three alternatives analyzed in this research as well as the attributes considered. A detailed process in each of the alternative and its respective advantages and disadvantages will be discussed in regardless of the influences among others.

5.1.1 The Old Technology

This refers to the SLAB trading situation prior to December 2018. When a user wanted to short a stock, he or she needs to communicate with the SLAB traders in person, by phone call, or by sending text messages. The SLAB traders then manually put the user's information and the details of the trading request in Falcon software. After waiting for more than two days, SLAB traders receive the transaction data and communicate with the users to determine the decision of borrowing or not.

This alternative has two major advantages. First, because this is the old technology, no new implementations or modifications are needed. This greatly reduces the cost of labor and implementation budget on implementing new things so that BNP can allocate funds towards developing other business unit within the company. Second, more communications between SLAB traders and users can make the trading process more personal and accurate. If users are not sure about which kind of stock to borrow, they can always communicate with the SLAB traders and obtain clarifying responses.

However, this old technology did not meet the demands from stakeholders. They would like to alleviate all manual burden and maintain trading records in the process to support the company's business requirement of statistical analysis and efficiency. Moreover, the innovation of a software process can greatly fasten the progress made. In a bank, the most important mission is creating banking product, and the next is technology. Technology always changes the way businesses operate. Innovations in technology can even influence the departments of customer service, online banking, and fraud detection (Content, 2018).

5.1.2 The Previous Chatbot

This is the chatbot implemented in B term, 2018. The overall process of the chatbot to complete a SLAB trading has four steps: 1. Symphony receives the command from users; 2. Symphony chatbot interprets the command and sends a request to Falcon API; 3. Falcon API returns a decision or trading information to the chatbot; 4. The chatbot converts the information to human-readable format and sends back to user on Symphony. The chatbot is implemented on the Symphony chatting platform. In technical aspect, it communicates with Symphony API and Falcon API and calls two Falcon methods inside of the process. More information can be found in Section 2.4 of this report and the previous report.

This alternative addressed all the demands from project stakeholders. The chatbot took over most of the manual steps, and the Symphony platform is able to store all conversation

history. Furthermore, a locate request process can be done in 10 seconds. This alternative greatly reduced the whole processing time. Moreover, because the Symphony platform is widely used inside of the company, there is no additional cost for the license of Symphony for implementing the Symphony chatbot.

However, the main disadvantage is the budget of chatbot implementation and design. Because chatbot is a new technology implementation at BNP as well as in the banking industry, users to not have much experience. Moreover, the chatbot only wraps up a few Falcon methods, so while it works in a straightforward manner, it does not have limited functionality. This alternative will likely the developer team to spend more time on technical support.

5.1.3 The Improved Chatbot

The improved chatbot is the proposed technology in this project (see Chapter 4). It works on Symphony platform and wraps up five methods from Falcon API. To implement this chatbot, the Falcon API needs to be modified by adding one new method called cancelRequest(). More details regarding this alternative can be found in Chapter 4 of this report.

This technology addresses many of the disadvantages outlined in Section 5.1.2 of the previous chatbot. It would also add an NLP feature that can greatly reduce time and labor wasted for technical supports. However, this alternative presents a significant cost. First, the process inside of the chatbot needs more testing to be pushed into production. Developers need to make several process-designs and choose the one with the best time efficiency. Second, because Falcon is an old technology, modifying Falcon will require significant labor costs to learn the old system, developing new methods, and test before published.

5.1.4 Attributes

Seven attributes were identified as important in this project. The attributes are

1. Time to process a Locate Request,
2. Annual Cost (Budget),
3. Initial Cost (Budget),
4. Learning Curve of Users,
5. Cost of Development & Maintenance (Labor cost of developers),
6. Functionality,
7. Level of Manual Steps.

These attributes were obtained through research and brainstorming. Stakeholders at BNP were contacted several times for their feedback.

5.2 Analysis by AHP

In this section, I will discuss the detailed procedures of the research using AHP. Ideally preferences regarding alternatives and attributes would be solicited from the stakeholders; however, this was not possible in C term. As such, preferences for the MADT aspect of this

project were solicited from my fellow MQP teammates as they were involved in the project from B term. The alternatives and attributes in this research were described in Section 5.1.

Data are gathered by a questionnaire form which can be found in Appendix C. There are two responses gathered in this research. The two responses will be distinguished by response A from team member A and response B from team member B. Table 3 and Table 4 below are data gathered for AHP Analysis.

Response A:

AHP	Time	Annual Cost	Initial Cost	Learning Curve	Development	Functionality	Manual Steps
Time	1	1/7	1/3	1/5	1/9	1/7	3
Annual Cost	7	1	7	5	1/3	3	8
Initial Cost	3	1/7	1	1/3	1/7	1/5	3
Learning Curve	5	1/5	3	1	1/6	1/3	7
Development	9	3	7	6	1	5	9
Functionality	7	1/3	5	3	1/5	1	7
Manual Steps	1/3	1/8	1/3	1/7	1/9	1/7	1
SUM	32.33	4.94	23.67	15.68	2.07	9.82	38.00

Table 3 - Data of Response A (AHP)

Response B:

AHP	Time	Annual Cost	Initial Cost	Learning Curve	Development	Functionality	Manual Steps
Time	1	3	7	1/2	7	3	3
Annual Cost	1/3	1	3	1/3	2	1/2	1/2
Initial Cost	1/7	1/3	1	1/5	1/3	1/2	1/2
Learning Curve	2	3	5	1	5	5	1/2
Development	1/7	1/2	3	1/5	1	1/3	1/3
Functionality	1/3	2	2	1/5	3	1	1/3
Manual Steps	1/3	2	2	2	3	3	1
SUM	4.29	11.83	23.00	4.43	21.33	13.33	6.17

Table 4 - Data of Response B (AHP)

Weights of the attributes determine the priority and importance of each attributes that influence the decision. Table 5 and Table 6 below are the weights of the alternatives from the two responses.

Response A	Priority
Time	0.03
Annual Cost	0.24
Initial Cost	0.05
Learning Curve	0.10
Development	0.40
Functionality	0.15
Manual Steps	0.02

Table 5 - Weighted Priority of Response A (AHP)

Response B	Priority
Time	0.28
Annual Cost	0.08
Initial Cost	0.04
Learning Curve	0.26
Development	0.05
Functionality	0.09
Manual Steps	0.19

Table 6 - Weighted Priority of Response B (AHP)

It is obvious to see that the two responses have a different priority. Response A emphasizes the cost of development and maintenance, but he thinks the time and manual steps of the product is not very important compared to the other attributes. The reason he gave was that the budget on development determines the successful creation of the product. If the budget is too expensive, it would not allow a process to be implemented, and the other attributes are not possible to be considered. However, response B gives the opposite priority. Response B emphasizes the time to process a locate request and the level of manual steps. The reason he gave was that BNP is a large-size company who has enough budget for technology innovation. Because the stakeholders would like to alleviate all manual burden in order to automate the process, these two attributes should be considered the most important for decision making.

After calculating the inconsistency ratio, both of the values from the two responses give the ratio lower than 0.1. The inconsistency ratio for response A is 0.0966, and the inconsistency ratio for response B is 0.09592. Although response A and response B hold different opinions, the inconsistency ratio confirms that the responses are consistent to be used for following analysis.

The next step is to calculate the average priority, and I used the average priority to receive the score of each alternative by its performance on each attribute. Table 7 is the average priority calculated from the priority of the two responses.

Average	Priority
Time	0.16
Annual Cost	0.16
Initial Cost	0.05
Learning Curve	0.18
Development	0.23
Functionality	0.12
Manual Steps	0.11

Table 7 - Average Priority from Response A and B (AHP)

From research and personal experience, I created a table of scores shown in Table 8. These scores represent the performance of each alternative individually base on the attributes. The scores are five from one to nine. Each row represents one alternative. Alternative #1 is the old technology. Alternative #2 is the previous chatbot implemented in B term. Alternative #3 is the new improved chatbot. The scores are given based on the alternative descriptions in Section 5.1.

	Time	Annual Cost	Initial Cost	Learning Curve	Development	Functionality	Manual Steps
#1- Old Technology	1	5	5	3	5	1	1
#2- Previous Chatbot	3	3.5	3	5	3	3.5	4
#3-Improved Chatbot	5	3	2	5	1	5	5
SUM	9	11.5	10	13	9	9.5	10

Table 8 - Scores of the Alternatives on Each Attribute

By multiplying the scores and average priorities, I obtained the total score of each alternative representing the performance for each given the attributes. The old technology received 0.299. The previous chatbot received 0.344. The improved chatbot received 0.347. From the scores, the improved chatbot is considered the best alternative from the three because it has the highest score from analysis of AHP.

5.3 Analysis by SMART

In this section, I will discuss the detailed procedures of the research using SMART. Ideally preferences regarding alternatives and attributes would be solicited from the stakeholders; however, this was not possible in C term. As such, preferences for the MADT aspect of this project were solicited from my fellow MQP teammates as they were involved in the project from B term. The alternatives and attributes in this research were described in Section 5.1.

Table 9 is the SMART data received from response A and response B. The weights of the attributes range from zero to 100. The distributions of the responses are similar to the distributions from the responses of AHP.

SMART	Response A	Response B
Time	70	80
Annual Cost	65	70
Initial Cost	80	40
Learning Curve	85	90
Development	100	20
Functionality	90	50
Manual Steps	60	100

Table 9 - Data from Response A and B (SMART)

Table 10 is the weighted average weight of the attributes calculated from both responses.

Average	weight
Time	15.25
Annual Cost	13.69
Initial Cost	11.72
Learning Curve	17.73
Development	11.31
Functionality	13.74
Manual Steps	16.57

Table 10 - Average Weights from Response A and B (SMART)

The scores of SMART are received by using the performance of alternatives (Table 8) and the average weight table (Table 10). The old technology received 26.98. The previous chatbot received 35.04. The improved chatbot received 37.98. From the scores, the improved chatbot is the best alternative in SMART.

Chapter 6: Conclusion and Recommendations

6.1 Conclusion

From the analysis in Chapter 5, both SMART and AHP indicated that the improved chatbot is the best alternative among the three alternatives. The result was obtained based on my team and my objective judgments of rating the attributes and alternatives. While I was able to collect only minimal information from project stakeholders, the result is well-supported by my observations on-site as well as my experience of coding. The result has comprehensive evidence to be referred to in BNP's decision making.

The reason the improved chatbot stands out in that it has a shorter processing time. The time reduced greatly addresses the problem brought by the delay of communications between users and SLAB traders. It also has a smaller learning curve comparing to other alternatives. This advantage can eliminate many costs on technical workshops and tutorials held by the company. Although the improved chatbot requires a greater amount of the development budget, I suggest BNP allocate more resources to technology innovation so that the time and effort saved from the chatbot can be used to create more valuable products and services. Because of these reasons, I recommend BNP Paribas implement the improved chatbot to automate the stock trading process.

6.2 Recommendations

Because of the limited time and resources of this project, there are several recommendations I would like to provide regarding future research, analysis, and development.

First, the process of the improved chatbot should be improved and tested by technical developers in the company who are familiar with SLAB desk and the content of the process. Although I improved the chatbot by implementing more built-in methods from Falcon webservice, there are still methods in Falcon that are not included in the new process. Those methods will need more research to be implemented in the chatbot because they require special criteria, credentials, and parameters.

Second, further analysis of decision making should be conducted by using rates and preferences from project stakeholders. In this project, I used minimal information collected by the project stakeholders because I have difficulty communicating with the office in NYC. The result I obtained was based on observations and preferences from people who participated in the project before. To have a decision that completely addresses the problem brought forth by BNP, it is better to have more data and preferences from people who brought the problem statement and who lead the project.

Finally, because the previous chatbot is already implemented in BNP's software repository, feedback can be gathered from users and SLAB traders when the software is pushed in production. Feedback and suggestions from the users would be able to provide more ideas on improving the process and the chatbot.

Chapter 7: Industrial Engineering Reflections

In this project, I designed the improved Symphony chatbot using a sequence diagram (refer to Figure 8). The diagram flows from a user starting the Symphony platform, entering a "locate request" command, receiving a response, and closing the interface. Comparing to the previous chatbot made in October, 2018, the newly improved chatbot has a new process when the locate request is a hard-to-borrow case. The process wraps more Falcon methods with a specific order and looping conditions to make the chatbot working more efficiently.

I used LucidChart to create the diagram. When I designed the document, the key factors I took into considerations were making the process concise, adding more functionalities, and giving users a better user experience. I also reviewed the functionalities each Falcon method does, analyzed the useful methods that could help improve the process, and considered the parameters each method takes. Because I was the lead developer when my team coded the previous chatbot, I was able to use my coding background and experience to determine if the methods have accessible information to execute in the process of a locate request.

There are several constraints that influenced my design. The first constraint was resources. First, there was no previous examples or tutorials that could help me design the chatbot because this is the first chatbot that can process stock lending and borrowing procedures. Second, because the previous chatbot was implemented on-site at BNP Paribas office, it is impossible to test this project to make any changes. Therefore, all designs made are based on the contents that I fully understood. For the other functionalities that I am not familiar with, because it is hard to access the information and test them, they are not included in this design.

The second constraint was communication. Because I was not able to work on-site to make this design, there are very limited communications between me and the stakeholders of this project. Therefore, this design is just a proposed model that could provide some idea to the company that might benefit them to alleviate their manual burden of the current process. However, because the design was not guided or evaluated by the stakeholders, it is not clear whether this design will be implemented in the future.

It was a great experience for me to work on a project independently. Throughout the project, I learned to overcome limited resources and communications by taking the initiative to look for any support I can find. I also grew my problem-solving skills, writing skills, and designing skills. I used several methods that I learned from previous courses, and I explored new methods and topics in Industrial Engineering and decision making.

Bibliography

Beers, B. (2018). Short selling basics. Retrieved November 14, 2018, from https://www.investopedia.com/articles/investing/100913/basics-short-selling.asp

Computer Hope (2018). Programming Language. Retrieved November 14, 2018, from https://www.computerhope.com/jargon/p/proglang.htm.

Content, P. (2018). How technology is impacting the finance and banking sector. Retrieved November 20, 2018, from https://www.information-age.com/technology-finance-banking-sector-123471800/.

Edwards, W., & Barron, F.H. (1994). SMARTS and SMARTER: Improved Simple Methods for Multiattribute Utility Measurement. *Organizational Behavior and Human Decision Processes*, 60(3), 306-325.

History of the bank over the last two centuries. (n.d.). Retrieved November 28, 2018, from http://group.bnpparibas/en/group/history-centuries-banking.

History of the Group. (n.d.). Retrieved November 10, 2018, from http://www.bnpparibas.com.sg/en/bnp-paribas/bnp-paribas-group/history-of-the-group/.

Hoffman, C. (2018). *What is an API.* Retrieved November 10, 2018, from https://www.howtogeek.com/343877/what-is-an-api/.

McGrath, M. (2013). *Python in easy steps.* In Easy Steps.

Nicholson, M., Wang, Y., & Chen, Y. (2019). BNP Paribas: Symphony Chatbot. MQP Report submitted to WPI.

Norton, S. (2017). BlackRock Using Symphony Communications Platform to Chat With Other Firms. The Wall Street Journal. Retrieved November 28, 2018, from https://blogs.wsj.com/cio/2017/04/06/blackrock-using-symphony-communications-platform-to-chat-with-other-firms/.

Quan, W. (June 2018). J.P. Morgan: Enabling Content Search UX Through the J.P. Morgan Markets Search Bot. Retrieved November 28, 2018, from https://www.youtube.com/watch?v=DSSpPWQJLmU.

Retail Banking & Services. (n.d.). Retrieved November 28, 2018, from http://www.bnpparibas.com.co/en/about-the-group/activities/retail-banking-services/.

Revolvy, L. (n.d.). "Symphony Communication" on Revolvy.com. Retrieved November 28, 2018, from https://www.revolvy.com/page/Symphony-Communication.

Seif, G. (2018). An easy introduction to Natural Language Processing. Retreived Feb 4, 2019, from https://towardsdatascience.com/an-easy-introduction-to-natural-language-processing-b1e2801291c1.

Symphony Blog. (n.d.). Retrieved November 28, 2018, from https://symphony.com/blog/item/introducing-symphony-partners-dow-jones-mcgraw-hill-financial-and-selerity.

Symphony (2018). Overview. Retrieved November 28, 2018, from https://rest-api.symphony.com/docs/rest-api-introduction.

Triantaphyllou, E. (2000). *Multi-Criteria Decision Making Methods: a Comparative Study*. Dordrecht, the Netherlands.

WodeHouse, C. (2018). SOAP vs. REST: A Look at Two Different API Styles. Retrieved November 5, 2018, from https://www.upwork.com/hiring/development/soap-vs-rest-comparing-two-apis/.

Appendices

Appendix A: Diagrams of the Previous Project

Architectural Overview

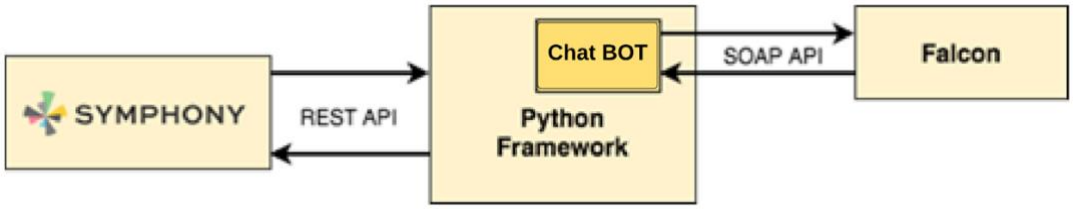

Figure 9 - Architectural Overview of the Previous Chatbot

Use Case Diagram

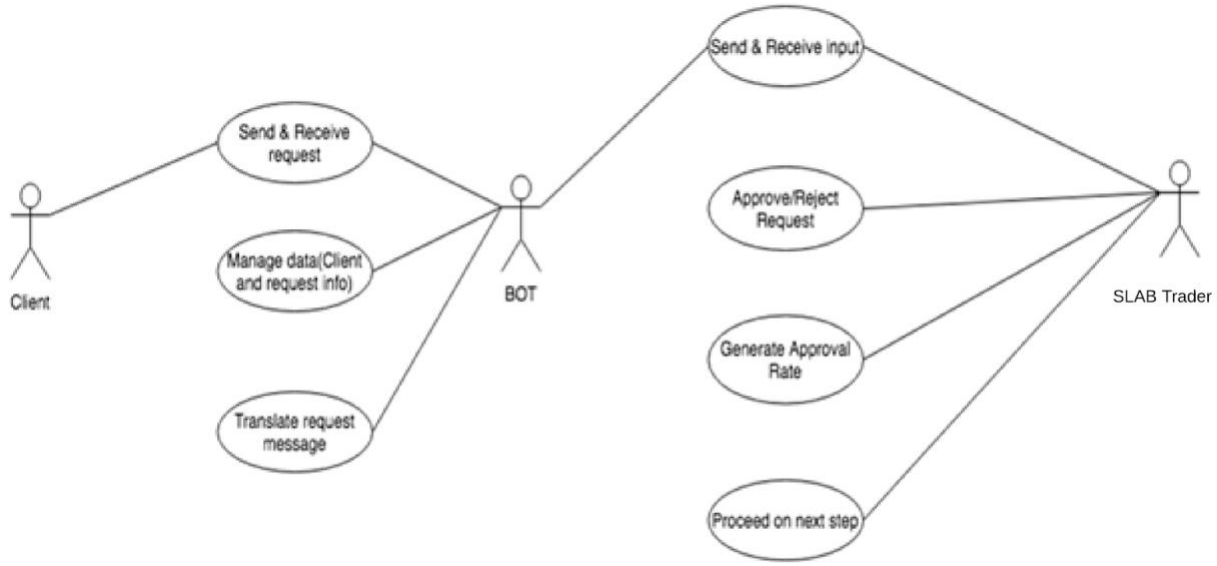

Figure 10 - Use Case Diagram of the Previous Chatbot

Swim Lane Diagram

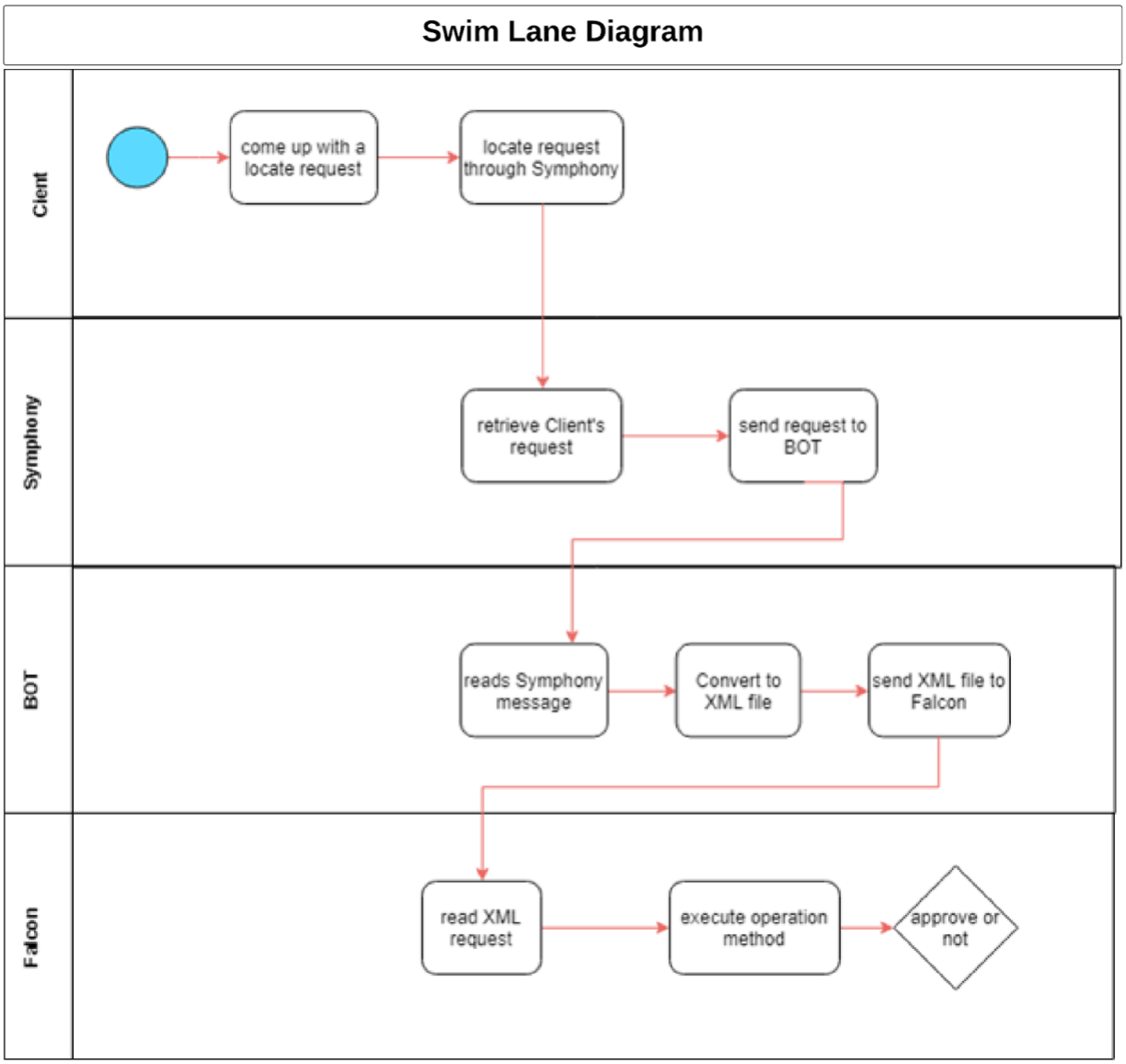

Figure 11 - Swim Lane Diagram - Part 1 (Previous Chatbot)

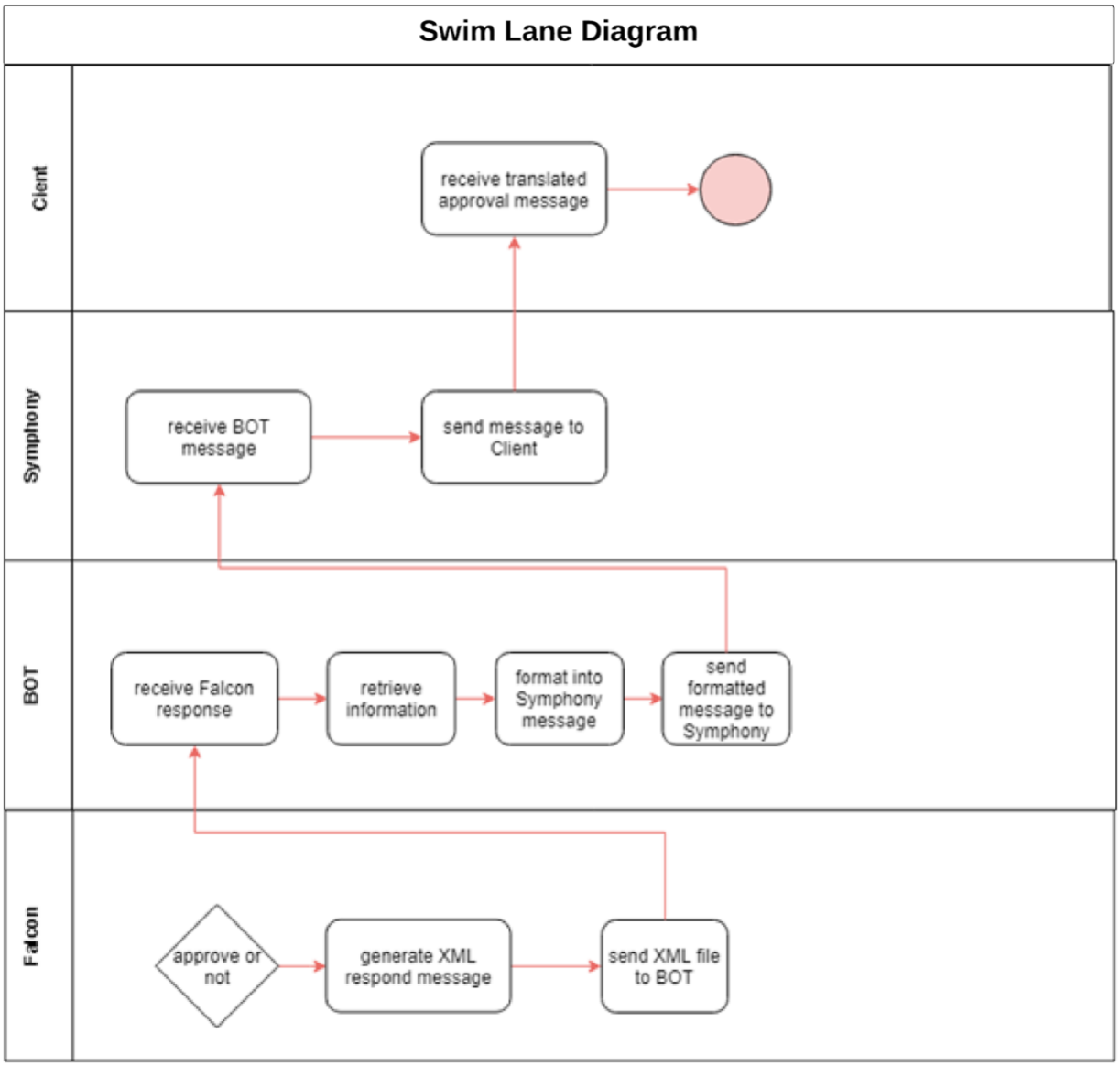

Figure 12 - Swim Lane Diagram - Part 2 (Previous Chatbot)

Flow Chart

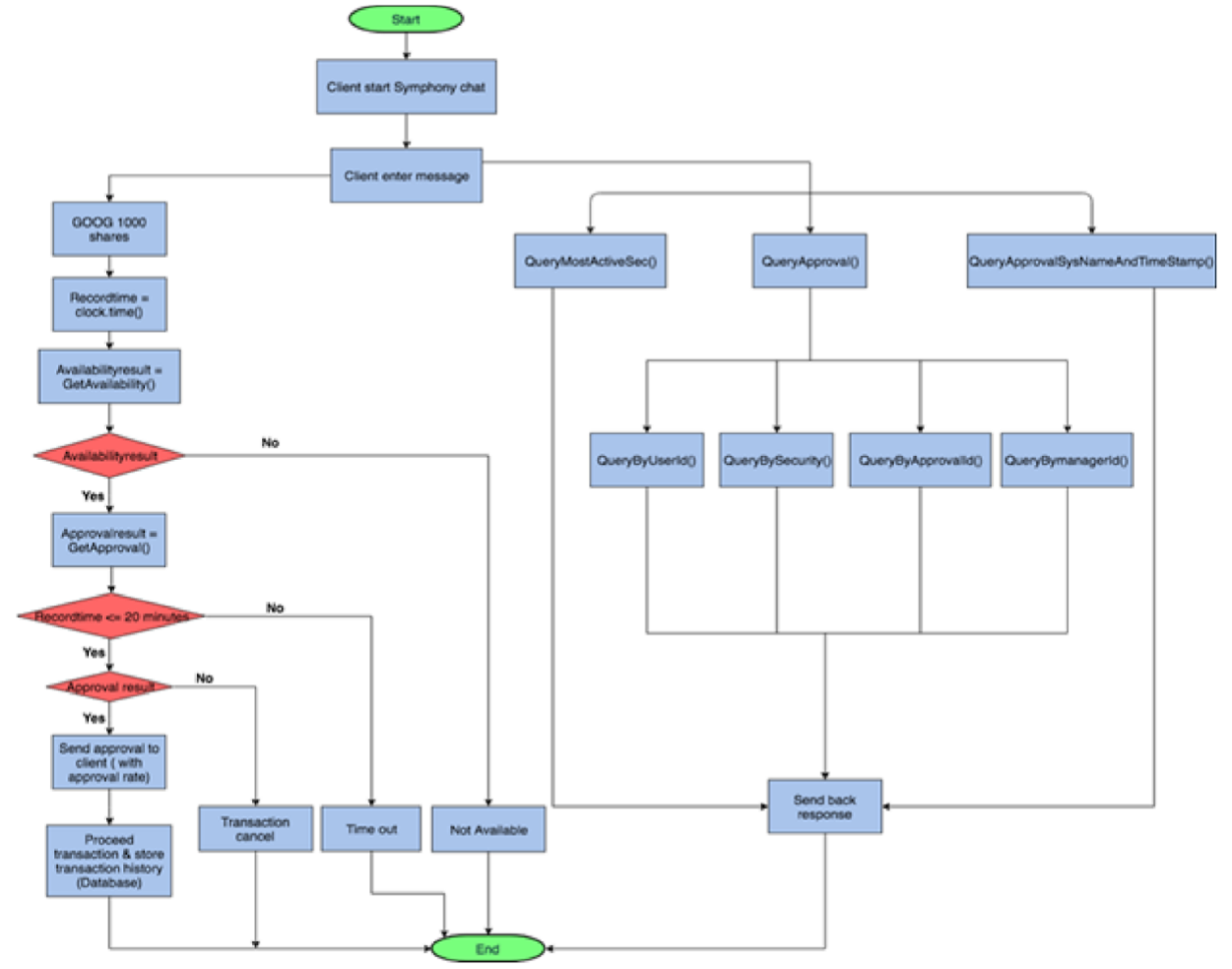

Figure 13 - Flow Chart of the Previous Chatbot

Appendix B: Functionality of the Previous Chatbot

Message Processing

The chatbot is able to recognize several commands: ‘Hi’, ‘Method’, ‘help’, ‘bye’, ‘stock request’, ‘status’, ‘search’, and any input that is not programmed by the bot. Figures 14, 15, and 16 below are examples of the screenshots for these commands.

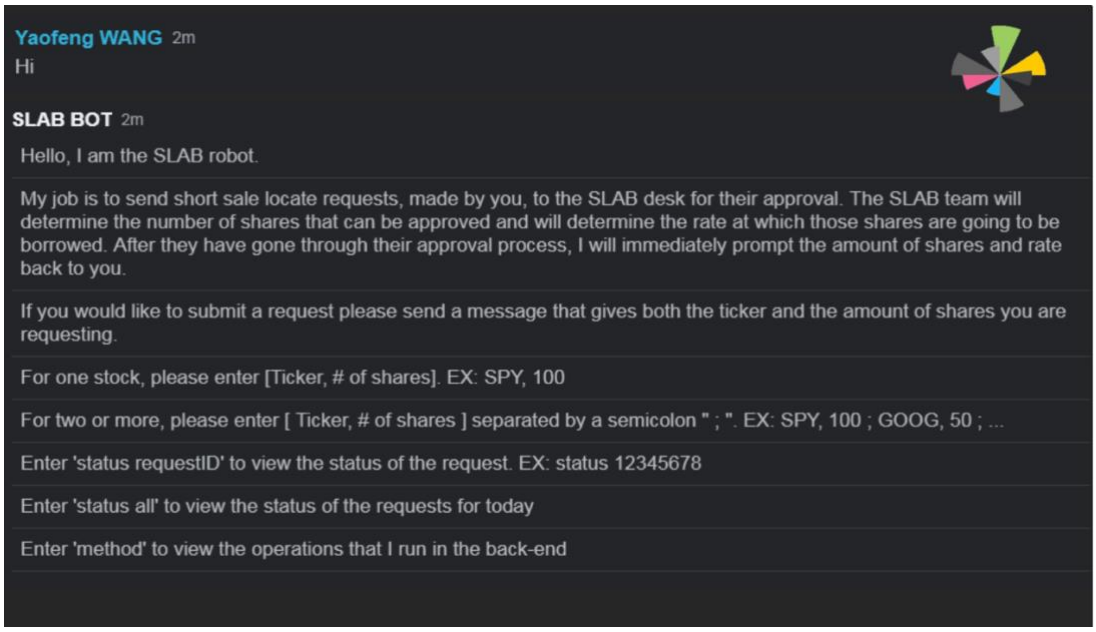

Figure 14 - Interface of Command "Hi" (Previous Chatbot)

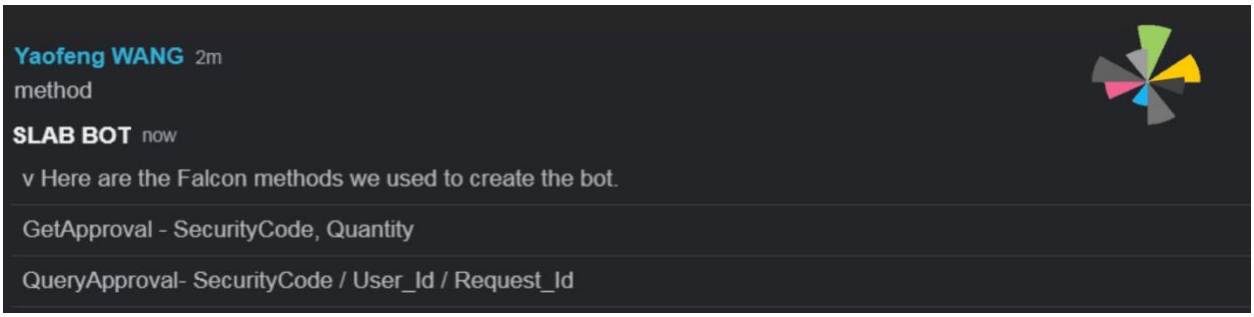

Figure 15 - Interface of Command "Method" (Previous Chatbot)

Figure 16 - Interface of Command "bye" (Previous Chatbot)

Error Handling

Error handled by the chatbot can be broken into five parts: invalid single locate requests, invalid multiple locate requests, invalid status requests, invalid commands, and server error. The tables below are the samples of wrong input and the chatbot's response.

Command typed by user	Respond send from bot
GOOG, 100 (Right format)	
GOOG, GOOG	The second arg should be a positive integer!
100, GOOG	The first arg should be stock name!
1,1	Sorry I don't understand!
GOOG, 999999999999999	The number requested exceeds the max limit!
GOOG, -2	The second arg should be a positive integer!
GOOG, 100.3	The second arg should be a positive integer!

Table 11 - Invalid Single Requests (Previous Chatbot)

Command typed by user	Respond send from bot
GOOG, 100; IBM, 101; TSLA, 102 (Right format)	
GOOG, 100; IBM, 101; TSLA, 102; (Right format)	
; / ;; / ;;; / …	Sorry I don't understand!
1;1	Sorry I don't understand!
GOOG, 100; IBM, 101; TSLA, 102;;;;;	Respond same as the right format
GOOG, 100; IBM, 101; TSLA, 102; 111	Respond same as the right format
GOOG;100	Sorry I don't understand!

Table 12 - Invalid Multiple Requests (Previous Chatbot)

Command typed by user	Respond send from bot
status 12345678 (right format)	
status all/ status ALL / status All (right format)	
status	Missing the second argument
status GOOG	The second arg is not an integer! You may enter "status all"
status 123	The requestID is not valid. Unable to query 123.
12345678 status	Sorry I don't understand!

Table 13 - Invalid Status Requests (Previous Chatbot)

Figure 17 - Invalid Command (Previous Chatbot)

Appendix C: Questionnaire for SMART and AHP

Attributes: 1. Time, 2. Annual Cost (Budget), 3. Initial Cost (Budget), 4. Learning Curve of Users, 5. Cost of Development & Maintenance (Labor cost of developers), 6. Functionality, and 7. Level of Manual Steps.

1- AHP

Value of aij -> Interpretation
1: Objectives i and j are equally important
3: Objective i is slightly more important than j
5: Objective i is strongly more important than j
7: Objective i is very strongly more important than j
9: Objective i is absolutely more important than j

	Time	Annual Cost (Budget)	Initial Cost (Budget)	Learning Curve of Users	Cost of Development & Maintenance (Labor cost of developers)	Functionality	Level of Manual Steps
Time							
Annual Cost (Budget)							
Initial Cost (Budget)							
Learning Curve of Users							
Cost of Development & Maintenance (Labor cost of developers)							
Functionality							
Level of Manual Steps							

Table 14 - AHP Table in Questionnaire

2- SMART

	Original Weight (0-100)
Time	
Annual Cost (Budget)	
Initial Cost (Budget)	
Learning Curve of Users	
Cost of Development & Maintenance (Labor cost of developers)	
Functionality	
Level of Manual Steps	

Table 15 - SMART Table in Questionnaire