



# NUS

National University  
of Singapore

## ISY5005 Intelligent Software Agent IPA Group Project Report

**Order Email Notification**

Raymond Djajalaksana (A0195381X)  
Lee Boon Kien (A0195175W)

# Table of Contents

<b>Table of Contents</b>	<b>2</b>
<b>1.0 EXECUTIVE SUMMARY</b>	<b>4</b>
<b>2.0 PROBLEM DESCRIPTION</b>	<b>5</b>
2.1 PROJECT OBJECTIVE	5
<b>3.0 KNOWLEDGE MODELING</b>	<b>5</b>
3.1 Local AI	5
3.2 Text Similarity method	6
<b>4.0 SOLUTION</b>	<b>6</b>
4.1 SYSTEM ARCHITECTURE	6
4.1.1. Email Recognition	6
4.1.2. RPA Module	7
4.2 PROJECT SCOPE	8
4.3 ASSUMPTIONS	8
4.4 SYSTEM'S FEATURES	9
4.4.1 Local AI	9
4.4.2 Ease of usage and integration	9
4.5 LIMITATIONS	9
<b>5.0 CONCLUSION &amp; REFERENCES</b>	<b>9</b>
5.1 IMPROVEMENTS	10
5.1.1 Improving our local AI module	10
5.1.2 Make the RPA more robust	10
5.1.3 Reporting	10
<b>6.0 BIBLIOGRAPHY</b>	<b>11</b>
<b>Appendix</b>	<b>12</b>
APPENDIX A: SAMPLE SYSTEM OUTPUT	12
APPENDIX B: USERS MANUAL	13
SYSTEM OVERVIEW	13
USER INTERFACE	13
REQUIREMENTS	13
INSTALLATIONS	13
USAGE	13

## 1.0 EXECUTIVE SUMMARY

The evolution of artificial intelligence has become a hot topic in recent years. More companies are in the wave of embracing the recent technology practice called robotic process automation (RPA) to streamline certain processes and operations. With RPA, automation of mundane rules-based business processes is easily achieved. This has enabled business users to devote more time to higher-value work that would require human decision. By integrating the RPA with the knowledge of intelligent automation (IA) via Machine Learning (ML) and Artificial Intelligence (AI) tools, which is also known as Intelligent Process Automation (IPA), the models could be trained to replace mundane business process and at the same time make certain decision based on trained data. This convergence of technologies produces automation capabilities which has dramatically elevated competitive advantages for the companies.

One of the leaders in the IPA sector is UiPath. UiPath has amplified the power of their RPA technology by using artificial intelligence (AI) to create IPA solutions in three ways - unattended robotics, advanced computer vision and integration with third party cognitive services from Microsoft, ABBYY, Google and IBM. In the continuous development of the IPA, UiPath is looking to the enhanced version of IPA which offers more Natural Language Processing and Machine Learning services on Cloud in the next few patches.

In this project, by applying the knowledge of IPA, we have looked into automate a process that could eliminate the tedious process in our daily life. Online shopping has been the hot trend following global blooming of internet usage. Consumers can search for a product by surfing the website of the retailer directly or by searching among alternative vendors using a shopping search engine. Different online shopping platforms offer different kinds of products as well as pricing strategies, which depends on the vendors. Some of the famous online shopping platforms in Singapore are Lazada, Shopee and Zalora. However, while we tend to buy a lot of stuff from different online shopping platforms, tracing the status of an order could become an issue. Most of the online shopping platforms would send a notification email to the user upon confirmation of order, or updating of the order status. In order to ease tracing of order status within the chunk of emails, we proudly present the “Order Email Notification” system which would help us retrieve relevant order email and provide an update through Slack. The details would be shared in the subsequent sections.



## 2.0 PROBLEM DESCRIPTION

People who tend to do a lot of online shopping would face a difficulty to trace the order status, especially the orders are from different online shopping platforms. Searching the email for the order status was proved to be not efficient when you have piles of unread emails. The order tracking system in each online shopping platform only provides the status of the orders in the particular platform. To have a full glance of all the order status seems to be not achievable with current setup. In addition, there could be a lot of unnecessary information, such as product advertisement inside the order status email sent out by the online shopping platform. This has worsened the efficiency of tracking order status.

### 2.1 PROJECT OBJECTIVE

This project aims to create a system that will solve the problem described in above subsection:

1. Implement an order tracking system that will search through the unread new emails and identify the email which is relevant to order status.
2. Extracting order related information from the identified emails.
3. Posting the extracted information into the communication platform for enhancement of user reading experience.

## 3.0 KNOWLEDGE MODELING

In our project, we built a knowledge base via knowledge acquisition techniques and presented an overarching view of the knowledge via knowledge modeling. Thereafter, based on our comprehensive knowledge base, the program is integrated with the RPA to do intelligence processing.

### 3.1 Local AI

We created and trained a local AI to be able to recognize whether a document, in this case an email is classified as order notification or not. The data that we give to train the model is based on a real sample of emails, mixing between order notification emails and other emails as well. Using about 400 of data training we are able to achieve > 90% accuracy.

We use Support Vector Machine (SVM) as our local AI. This is because it is suitable for our purpose because in processing the documents, we are going to break the

documents down into vectors of words. We also tried to train a Neural Network model and compare the result, however it achieved less accuracy than SVM.

For the model to be working, we need to transform the corpus or collection of documents into a bag of words. And then for each word we can calculate the TF-IDF value. Formulation of TF-IDF is described below [1]. The formula is based on normalizing the term frequency in the document by inverse document frequency. This is because some of the words are commonly used in all documents while carrying little or no meaning, i.e. 'is', 'was', etc.

$$tfidf(t, d) = tf(t, d) * \log(N/(df + 1))$$

$$tf(t, d) = \text{count of } t \text{ in document } d$$

$$N = \text{no of document}, df = \text{count of document } d \text{ where } t \text{ exists}$$

Hence a document is transformed into a list of tfidf. We then vectorized the tfidf and used it as the input of our SVM model. The model will give output 0 or 1, where 0 indicates a document is not order notification, and 1 is order notification. Based on our training the SVM has more than 92% of accuracy to detect if a document is order notification or not.

### 3.2 Text Similarity method

We use cosine text similarity to extract the order status of an email [2].

$$\text{Cos } \theta = \frac{a \cdot b}{\|a\| \|b\|}$$

Here a and b is the count vector of the words in a sentence. Hence the idea of comparing two sentences using cosine similarity we can know how similar two sentences are. We have prepared a set of predetermined sentences which are samples of order status sentence. By comparing each email sentence to this list, we can pick a sentence from an email content with highest cosine similarity value.

## 4.0 SOLUTION

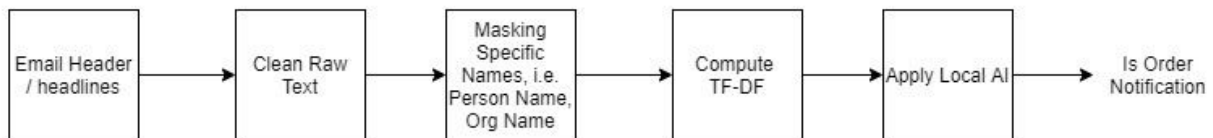
### 4.1 SYSTEM ARCHITECTURE

We can divide the system into 2 parts: Email Recognition and RPA module.

### 4.1.1. Email Recognition

Using knowledge modelling researched in section 3, we were able to create an Email Recognition class module that is able to tell if an email is an order notification email or not. Furthermore the module is able to extract information from the email content to get the order update status. Figure below describe the process of how Email Recognition module works:

Email Classification



Content Extraction

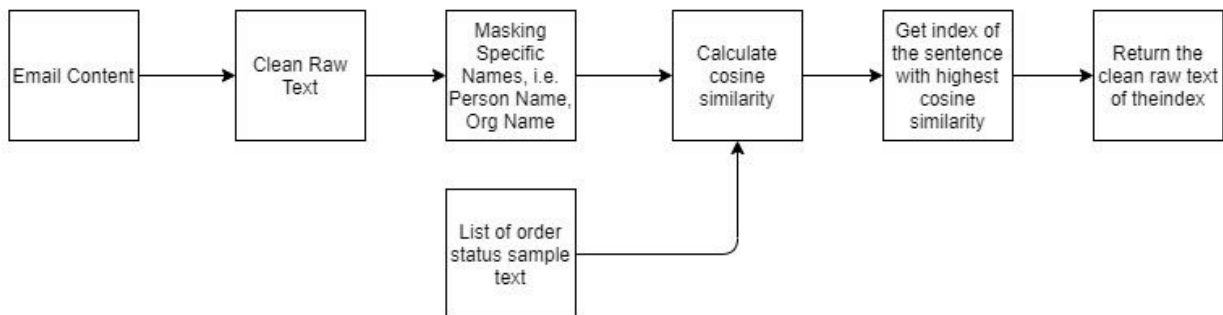


Figure 1. Diagram Flows of how Email Recognition module works in tagging the email and extract order status from the email content.

The first flow explains how we classify an email as order notification or not. To classify the email we can either read the whole content of the email, or just look at the title or summary of the email. While content will give a good context and precision on whether the email is order notification, there are 2 major concerns in our case. The first one is the performance issue as we will need to read the whole content for all emails, hence it might take long processing time. The second one is because we will use an RPA module to crawl the email to get the information, this means email content will need to process the raw html text. This means there can be more noises coming to local AI even after we clean the text. Moreover the more content text, the precision of the agent can decrease since it contains useless information. Hence we use email header as our input text to classify the email. It is easy to process, and delivers more consistent results. And we believe it is adequate enough using just email header to classify the text because the header should reflect the content of the email.

After we proceed to the text preprocessing part. It consists of two parts, first is removing all html tags or extract just the text that we scrape from online text. We do not want noise such as html

tags in our text since it is not part of the email text. After that we will need to mask some specific details, such as personal name, organisation name, etc with a placeholder. We add this approach to make sure the agent can generalize the input and does not have bias towards data training. For example when we trained our agent with the email “your order from Lazada has been shipped”, we should expect the agent to classify email from other shopping platforms as well, let’s say “Order from Shoppe has arrived” after the training. Otherwise our agent may only work for Lazada in this case and we have to train for all the names which is impossible. Once the text processing is done, the rest follow a standard flow, we calculate the TF-IDF and use the SVM model we have built to classify the email.

In the second part of the flow diagram, we showed how we were able to extract the order update information within the email content. In this case as we are digging into details of the email, hence we expect to get the email content as our raw input. Then we proceed with the same text preprocessing as when we classify the email. After that we try to compare the similarity of every sentence with a list of sample text that we have prepared. This sample text resembles how the information of order status update looks like. Then we pick the information from email with the highest similarity. However since what we compared was the text that has been masked, we need to return the original clean text that is not masked yet.

Using these two features, we were able to intelligently extract the user order status in the emails.

#### 4.1.2. RPA Module

We created a RPA module to mimic our daily process of checking the emails for order notification email. Before we run the main file, the email account and password are stored in a separated .json file. This is to ensure the information is kept confidentially and privately and users can amend the information confidentially and privately based on their need.

The RPA module was based on the TagUI that we have learnt in class. TagUI was built on AI Singapore’s TagUI open-source RPA software, where the RPA for Python we used in this project is a Python package for RPA. The system architecture of the RPA module is demonstrated in the figure shown below.



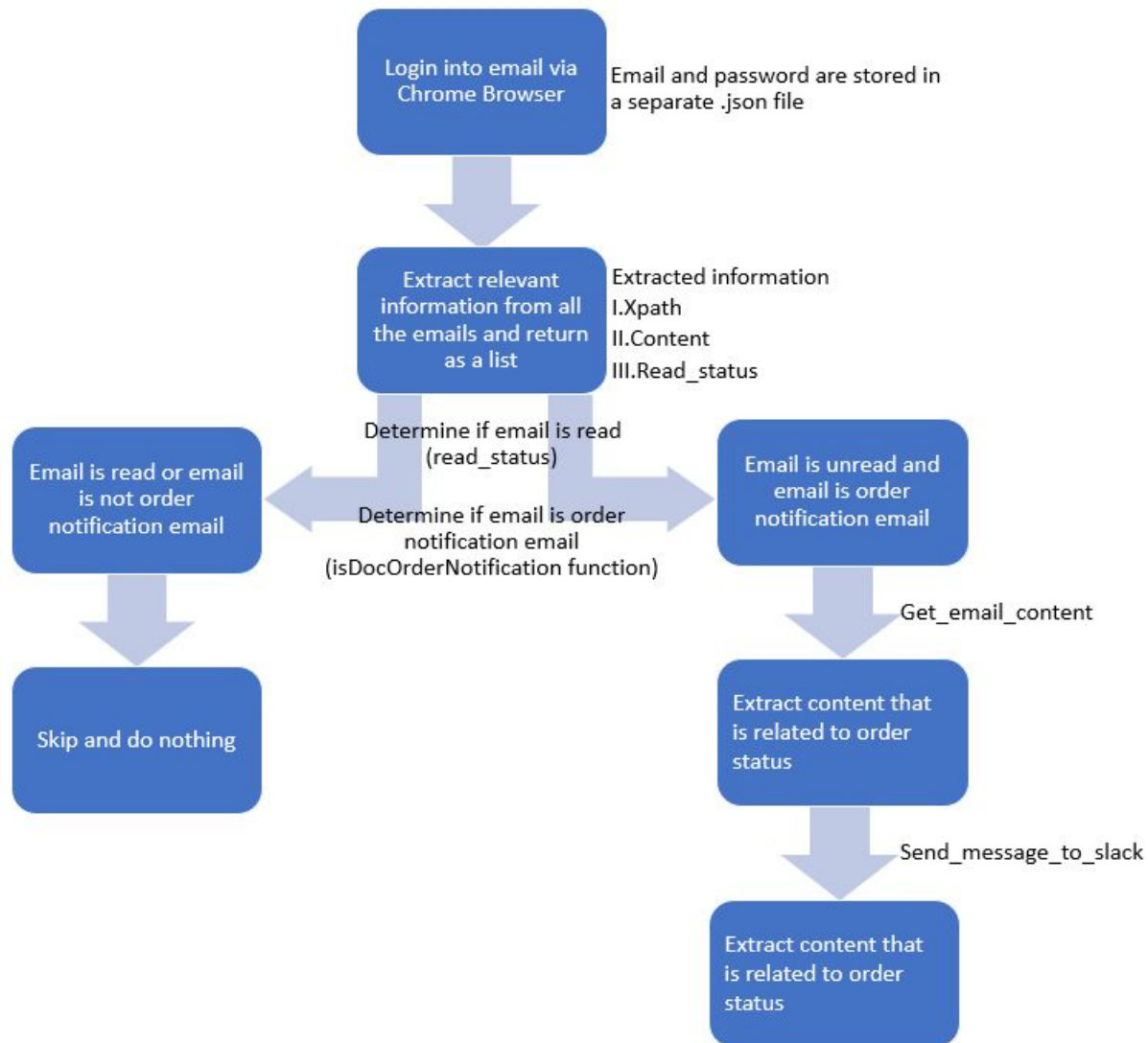


Figure 2. System architecture for RPA module

First of all, the RPA module mimics the human process of logging into an email, with the given email account and password stored in a separated json file. After logging into the email, the module starts looping through every single email to extract the relevant information for later usage. The information extracted here includes xpath, email content and the read status of the email. The Xpath is used as a unique ID for each email so we could easily trace back the particular email whenever we want to. Email content is the raw content of the email, which later on we will run the get\_email\_content model to generate a summarized content that is related to order status. The read\_status here refers to the status of the email whether it has been read or not. The information collected is then consolidated into a list, before running through the next email and returning a tuple at the end of the loop.

After getting the full information of all the emails, the next step is to identify which email is yet to be read and is an order notification email. The status of the email could be referred to the `email_status` extracted, and we run the `isDocOrderNotification` model in the email recognition model to identify if the email is an order notification email. For the identified emails which are categorized as new order notification email, we implement the `get_email_content` model to extract content that is related to order status. The extracted content is then posted to Slack channel to notify users on the new order status. Incoming webhook is required here to post the messages into Slack. The user has to declare the unique Slack URL at the initial stage of running the module such that the model knows where to post the messages.

## 4.2 PROJECT SCOPE

The scope of our system is limited to searching for order emails that need to be informed to the user of the recent update status of the order. The project is more on focusing on how we can automate the process and intelligently recognize the correct email that needs to be reported.

## 4.3 ASSUMPTIONS

Assumptions that we made when we building our system include:

1. We assume the user has read all the read emails. This means that we do not and should not notify a read order email, and only notify those that are not read yet. Otherwise there can be plenty of useless notifications since the user already read about it.
2. The RPA tolls should not cause any data changes in the user emails as it should only read the data. Hence when RPA tries to read the unread email content and the status becomes read, the RPA should revert it back to the status unread.

## 4.4 SYSTEM'S FEATURES

Despite the limitations mentioned in the previous sections, we were able to implement some key features as illustrated in this section.

### 4.4.1 Local AI

We use SVM as our main engine for Local AI. We have customized all the intent matching and knowledge bases to align with our goal to recognize the order email.

Furthermore we have done some training on the SVM itself to increase the accuracy up to 92%.

#### 4.4.2 Ease of usage and integration

As we use TagUI python for our main RPA module, we can leverage the full integration by using the built-in api that is available with other app. We can easily deploy our chatbot to many popular apps such as facebook messenger, slack app, telegram and many more. For our demonstration purpose we have embedded our order notification email with slack app.

### 4.5 LIMITATIONS

The knowledge model that is used to build the system is trained with a limited amount of sample data. This means the model is good in detecting order emails only from certain platforms. Additionally it is not trained to solve language difference issues, hence it can only be applied to english emails. We are also aware that our implementations are tied to the slack bot which limit the capability on how we report the results.

## 5.0 CONCLUSION & REFERENCES

The group discussion session was fruitful and rewarding as we have learnt much from each other. In this group project we have learned how to integrate and apply our intelligent module into a real world situation, not just as offline processing. Here we have demonstrated how we can integrate the RPA automation to scrap the email data and apply the decision making to extract all of the order notifications that are needed to be informed to the user, and finally notify the user via a bot. Our local AI acts as decision making for our automation RPA module to process the order notification. We noticed the integration is not a trivial task because we have to deal with noise inside the input data for our local AI while maintaining good performance for our application. In our particular case the noise is the html element in the raw scrapped text from the email website. Before we can process the scraped data we have to clean it and make sure the input text is the ideal content of the email message.

### 5.1 IMPROVEMENTS

If we were given a longer duration to work on this project, we would have worked upon the following points of improvement:

### 5.1.1 Improving our local AI module

We noticed the data training that we give is not adequate to have reliable results on all forms of order notification email. Because we have a limited sample of order notification emails, some of email from certain platforms are easily recognizable while others cannot. Hence with more time we can try to gather more sample data to expand the knowledge of our local AI.

### 5.1.2 Make the RPA more robust

As of now the RPA can only scrap the email with a constant maximum limit. In the future we can make it more robust and try different approaches to end the automation, for example we can use timestamp or date, i.e. up to 2 weeks ago as the signal to how much the RPA should read the emails.

### 5.1.3 Reporting

The current system will only notify the user via slack message. However in the future it will be good to be able to save the result in the form of a file. This can be useful for certain business processes, such as compiling order emails, or audit checks in case the user needs to list down the orders status manually.

## 6.0 BIBLIOGRAPHY


[1]<https://towardsdatascience.com/machine-learning-nlp-text-classification-using-scikit-learn-python-and-nltk-c52b92a7c73a>

[2]<https://medium.com/@bedigunjit/simple-guide-to-text-classification-nlp-using-svm-and-naive-bayes-with-python-421db3a72d34>

# Appendix

## APPENDIX A: SAMPLE SYSTEM OUTPUT

Figure A1. Sample of the messages posted to the slack channel.

 **orderbot** APP 10:15 PM  
New order email: We shipped your order today.. Email: Unread Raymond Djajalaksana Order Update: We Shipped Your Order Today Sun 4/12 Dear Raymond Djajalak...  
New order email: Order #28564215059647 has been confirmed.. Email: Unread Lee Boon Kien FW: Hey Lee Boon Kien , your order is confirmed! Mon 4/6 From: Lazada Singapore...  
New order email: Order #28564215059647 has been confirmed.. Email: Unread Lee Boon Kien FW: Hey Lee Boon Kien , your order is confirmed! Mon 4/6 From: Lazada Singapore...  
New order email: Order #28564215059647 has been confirmed.. Email: Unread Lee Boon Kien FW: Hey Lee Boon Kien , your order is confirmed! Mon 4/6 From: Lazada Singapore...  
New order email: Your order has been delivered!. Email: Unread Lee Boon Kien FW: Hey Lee Boon Kien , your items have been delivered! Mon 4/6 From: Lazada Si...  
New order email: Your Item(s) has been Shipped.. Email: Unread Lee Boon Kien FW: Hey Lee Boon Kien , your Item(s) has been Shipped Mon 4/6 From: Lazada Sing...  
New order email: Your order has been delivered!. Email: Unread Lee Boon Kien FW: Hey Lee Boon Kien , your items have been delivered! Sun 3/29 Sent from Mail...  
New order email: Your Item(s) has been Shipped.. Email: Unread Lee Boon Kien FW: Hey Lee Boon Kien , your Item(s) has been Shipped Sun 3/29 Sent from Mail f...

## APPENDIX B: USERS MANUAL

### SYSTEM OVERVIEW

We have trained and prepared our local AI model to recognize an email as an order email, typically where the email talks about the update status of the user order. Combining with RPA tools, we have created a program to scrap the email , extract the order emails, and necessarily send the notification to the user via a slack bot.

### USER INTERFACE

As we integrate our application with slack, hence the user interface is also relying on slack UI.

### REQUIREMENTS

You will need a slack bot url for the program to be able to send the notification to slack.  
webhook

### INSTALLATIONS

using pip

```
pip install -r requirements.txt
python -m spacy download en_core_web_sm
```

using anaconda

```
conda create -n isy5005_ipa python=3.7.1
conda activate isy5005_ipa
pip install -r requirements.txt
python -m spacy download en_core_web_sm
```

### USAGE

```
python main.py --slack_url your_slack_bot_url
```