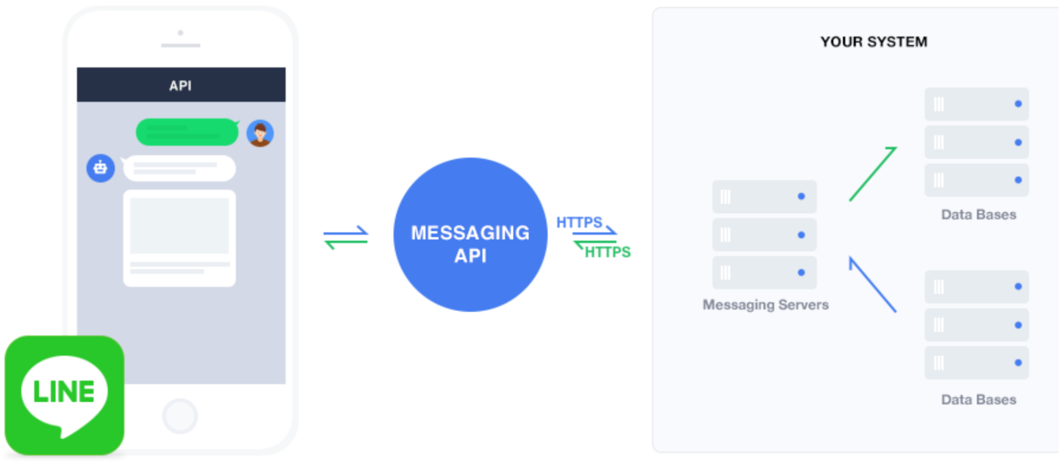
- **How is your project architecture related to the theory taught in the lecture?**

When we create this chatbot, the code is running on a clouding platform called Heroku. Data are preserved by Redis and the final interface is provided by Line. We can see that the function of chatbot are actually separated on many servers, which shows the characteristics of inherent distribution of cloud computing. As the data are preserved on Redis, we do not need to worry about the loss of data due to some hardware problems, which represent the reliability of cloud platform. What is more, Heroku provides a good platform to interact with client and it means that we do not need to buy or rent a server by ourselves. With this kind of service, the efficiency of a server can be maximized and then the cost of server for every companies or developers can be decreased.

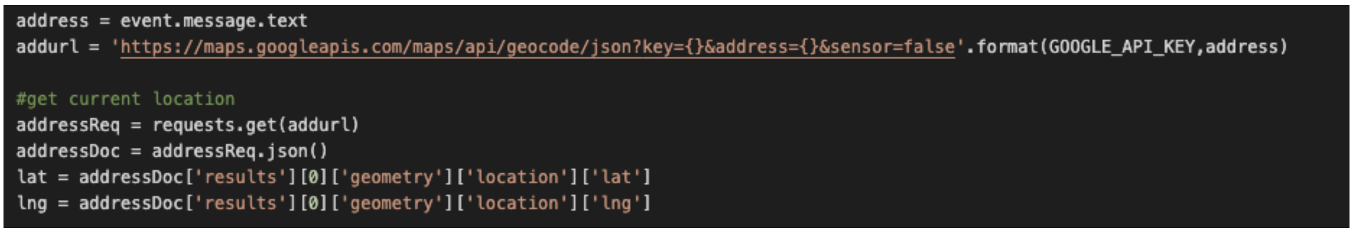


- **Can you demonstrate, with some screen cap, how to increase capacity of your chat bot service?**

For Pharmacy Recommendation Part:

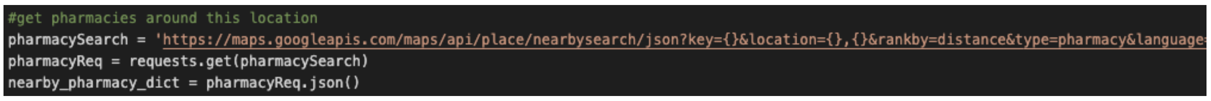
This part is designed to recommend a pharmacy for users based on the location that users provide. In this case, Google Map API are called to fulfill this function via url. There are three steps to complete this function.

Firstly, we need to get the latitude and longitude of the location provided.



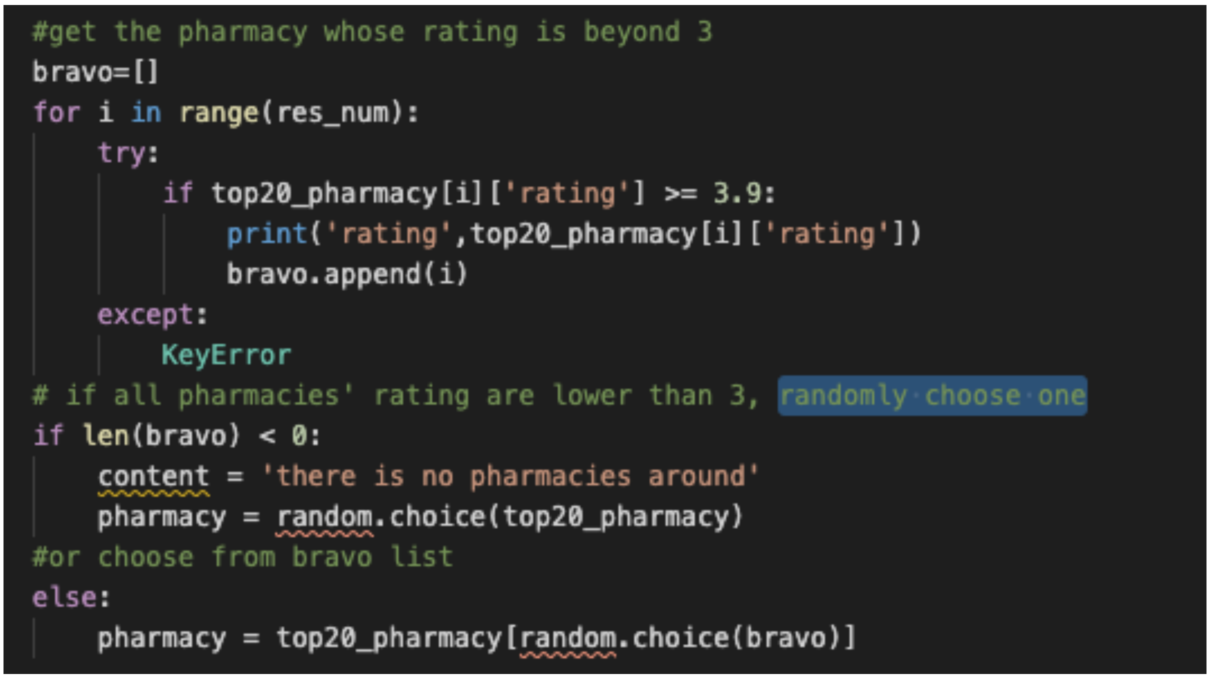
```python
address = event.message.text
addurl = 'https://maps.googleapis.com/maps/api/geocode/json?key={}&address={}&sensor=false'.format(GOOGLE_API_KEY,address)

#get current location
addressReq = requests.get(addurl)
addressDoc = addressReq.json()
lat = addressDoc['results'][0]['geometry']['location']['lat']
lng = addressDoc['results'][0]['geometry']['location']['lng']
```

Next, based on the detail of position, we are able to find pharmacies around it.



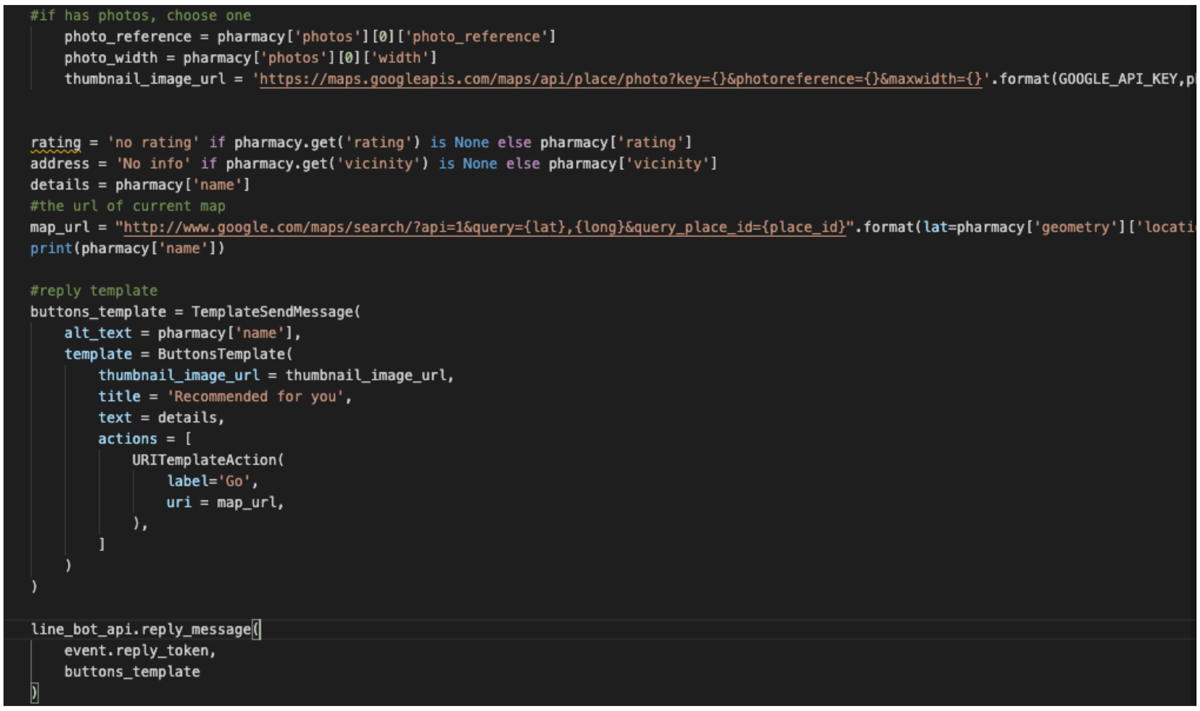
```python
#get pharmacies around this location
pharmacySearch = 'https://maps.googleapis.com/maps/api/place/nearbysearch/json?key={}&location={},{}&rankby=distance&type=pharmacy&language
pharmacyReq = requests.get(pharmacySearch)
nearby_pharmacy_dict = pharmacyReq.json()
```

After obtaining the list of pharmacies, some filtering tasks are done to the response. We only select pharmacies whose rating is higher than 3. If all of them are lower than 3, we just randomly choose one.



```python
#get the pharmacy whose rating is beyond 3
bravo=[]
for i in range(res_num):
    try:
        if top20_pharmacy[i]['rating'] >= 3.9:
            print('rating',top20_pharmacy[i]['rating'])
            bravo.append(i)
    except:
        KeyError
# if all pharmacies' rating are lower than 3, randomly choose one
if len(bravo) < 0:
    content = 'there is no pharmacies around'
    pharmacy = random.choice(top20_pharmacy)
#or choose from bravo list
else:
    pharmacy = top20_pharmacy[random.choice(bravo)]
```

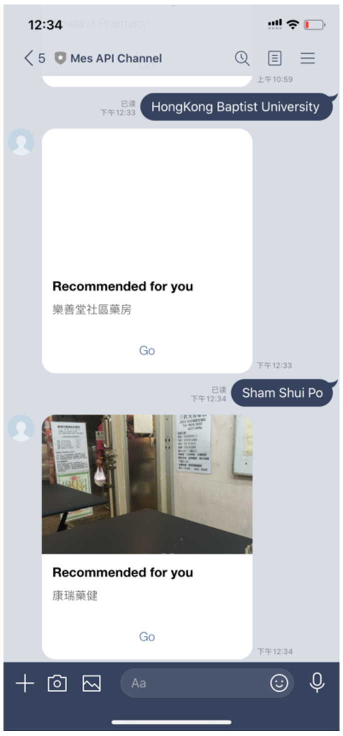
Finally, we need to obtain the url, photo, and rating information from this chosed item and put it in reply template developed by Line Developers.



```
#if has photos, choose one
    photo_reference = pharmacy['photos'][0]['photo_reference']
    photo_width = pharmacy['photos'][0]['width']
    thumbnail_image_url = 'https://maps.googleapis.com/maps/api/place/photo?key={}&photoreference={}&maxwidth={}'.format(GOOGLE_API_KEY,p


rating = 'no rating' if pharmacy.get('rating') is None else pharmacy['rating']
address = 'No info' if pharmacy.get('vicinity') is None else pharmacy['vicinity']
details = pharmacy['name']
#the url of current map
map_url = "http://www.google.com/maps/search/?api=1&query={lat},{long}&query_place_id={place_id}".format(lat=pharmacy['geometry']['locati
print(pharmacy['name'])

#reply template
buttons_template = TemplateSendMessage(
    alt_text = pharmacy['name'],
    template = ButtonsTemplate(
        thumbnail_image_url = thumbnail_image_url,
        title = 'Recommended for you',
        text = details,
        actions = [
            URITemplateAction(
                label='Go',
                uri = map_url,
            ),
        ]
    )
)

line_bot_api.reply_message(
    event.reply_token,
    buttons_template
)
```

This is the screen shot of this demo:

After sending an address, the chatbot will reply an recommendation pharmacy.



12:34

Mes API Channel

HongKong Baptist University

Recommended for you

樂善堂社區藥房

Go

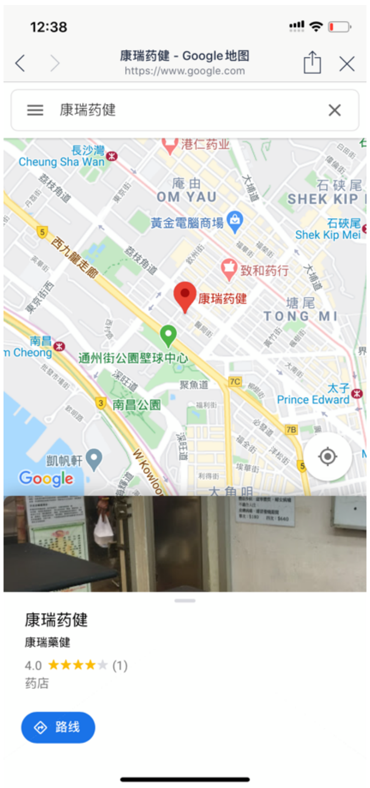
Sham Shui Po

Recommended for you

康瑞藥健

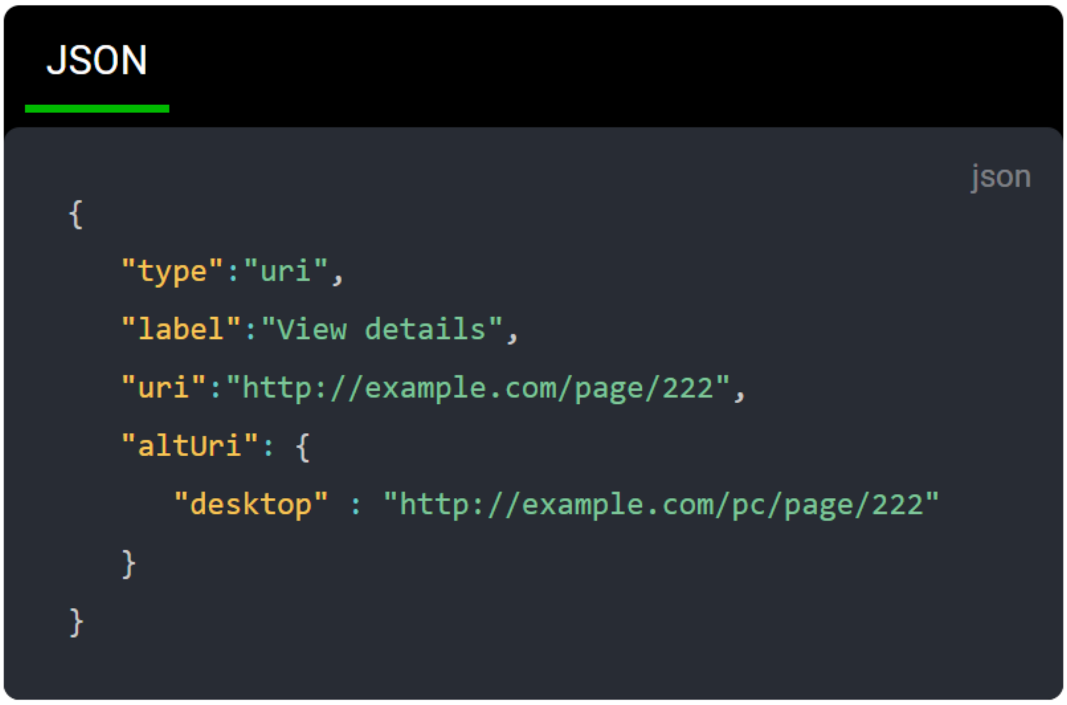
Go

Users can then click Go button to go to Google Map page for more function such as navigation or something else.



FROM ZHANYUMIN:

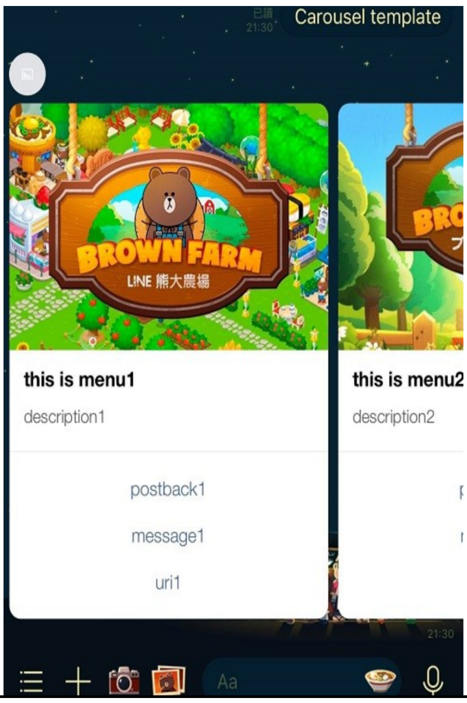
At first, I'm going to use Text message to return a href when the user enters a command. However, it was found that this is too monotonous and doesn't suit the positioning of the Line.

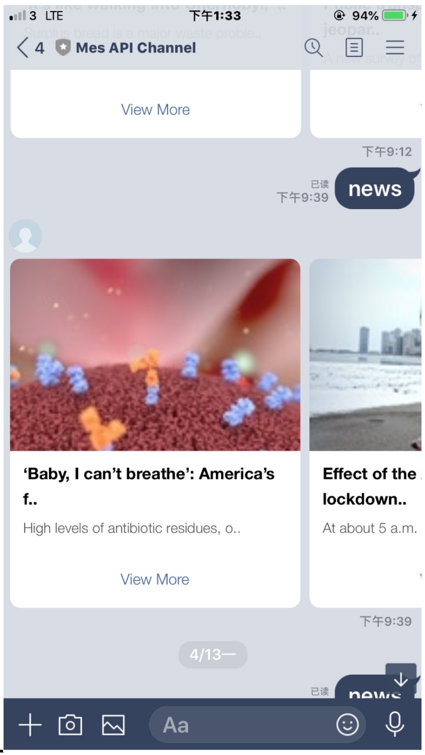

JSON

```json
{
    "type":"uri",
    "label":"View details",
    "uri":"http://example.com/page/222",
    "altUri": {
        "desktop" : "http://example.com/pc/page/222"
    }
}
```

After reviewing the official documentation, I found Temple message. This is an element can give the overall feedback of image, title, description and a button. And allows users to do further work. The basic use of this element will be described on map function. I need a further extended. A news push needs more than just a single item, which was too short and tedious. I think reasonable news should between 1 and 5 items. This ensures the necessary information and prevents users from spending too much time filtering.

So we need to use another element， carousel template. Shown on the left.



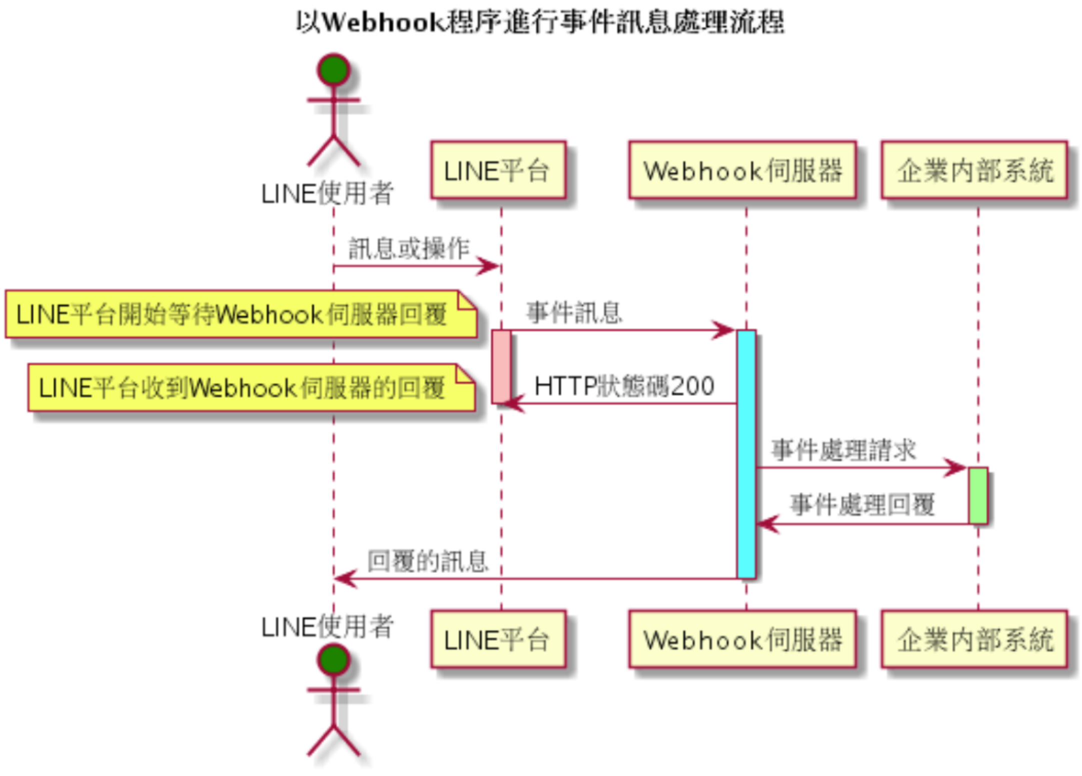


It perfectly fulfills my needs, it has an image to catch the eye, a basic introduction, further operations for the corresponding user, and most importantly, it can give multiple choices at once. Fantastic!! The final result is shown on the right.

- Can you identify if you bot is one of the example of PaaS, IaaS, SaaS? Explain your answer.

As we know, there are usually three models of cloud service to compare: Software as a Service (SaaS), Platform as a Service (PaaS), and Infrastructure as a Service (IaaS). This bot is an instance of PaaS because this bot is running on Heroku which is a cloud platform service. With the help of Heroku, we are not required to buy a server and to install some related software. Our focuses are only about the logic behind the code and some API that connect to Heroku or Line service.

For example, we will deal with the problem of input. The Line can directly use the API to provide input, but if the user requests frequently, the use of the API will exceed the limit. The News API can only select 500 items per day in a developing role. Besides, when using, I found that the link was not stable. Since the update of news class was not frequent, we considered using Redis to help store data. The advantage is that access can correspond to an access response over a while. Avoid service stops due to request overload. At the same time, improve the response time.


以Webhook程序進行事件訊息處理流程
LINE使用者
LINE平台
Webhook伺服器
企業內部系統
訊息或操作
LINE平台開始等待Webhook伺服器回覆
事件訊息
LINE平台收到Webhook伺服器的回覆
HTTP狀態碼200
事件處理請求
事件處理回覆
回覆的訊息