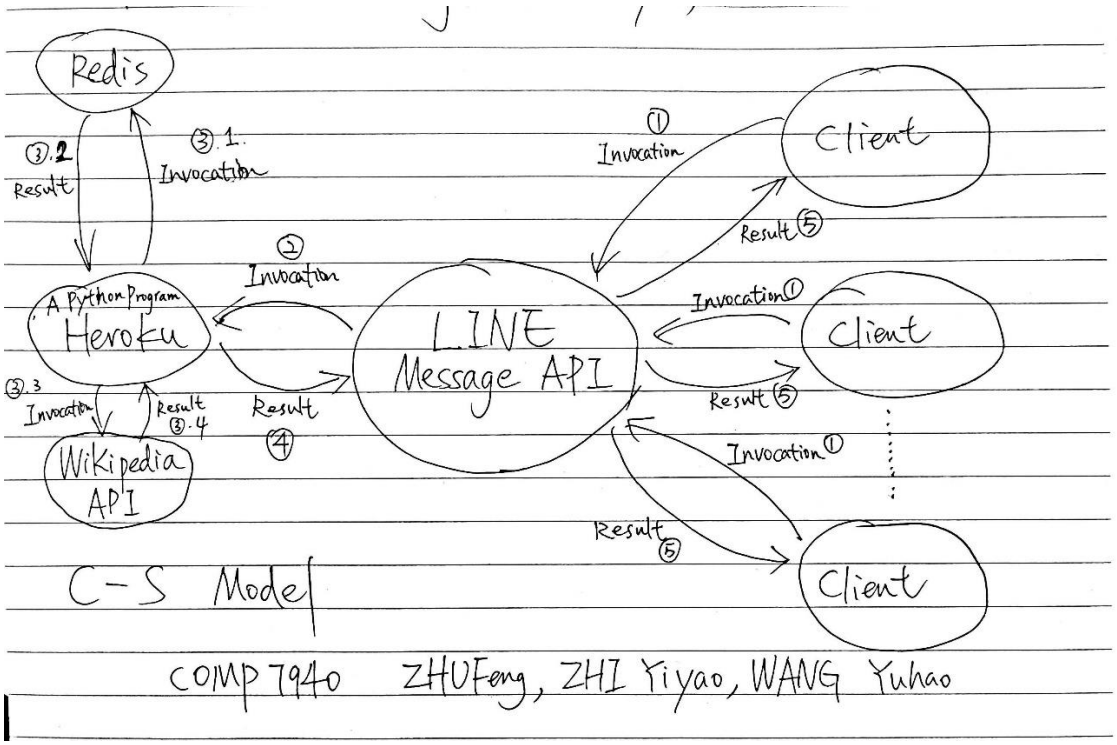
Question 1:

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

Answer 1:



It is a Client-Server Model.

1) The end users (clients) would use LINE to chat with our chatbot (to be specific, it is called LINE Message API according to LINE Developers website).

2) After the LINE Message API received a message from users, it will use webhook to inform the Python program running on Heroku.

3) Then the Python program would start to handle the message. During the Python program handling the message, it may invoke some functions related to Redis (e.g., SET, GET, HSET, HGET) to store or fetch data. It may also use our “Another Service” which is Wikipedia API to get some information about the keyword provided by users.

4) After handling the message, the Python program on Heroku would return the generated reply to the LINE Message API.

5) Then the LINE Message API would send the reply back to the users.

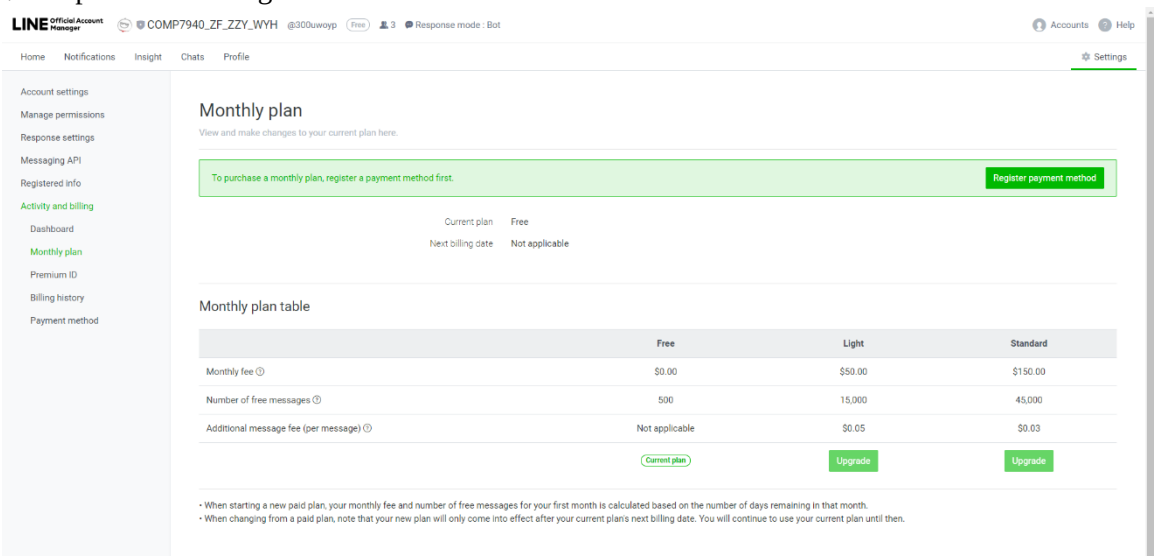
P.S. Here the end-users would only act as clients. The LINE Message API would act as both a server (for end-users) and a client (for the program running on Heroku). The Python program running on Hero would act as both a server (For LINE Message API) and a client (For Redis). The Redis would only act as a server for the Python program running on Heroku. Similarly, here the Wikipedia API would only act as a server for the Python program running on Heroku.

Question 2:

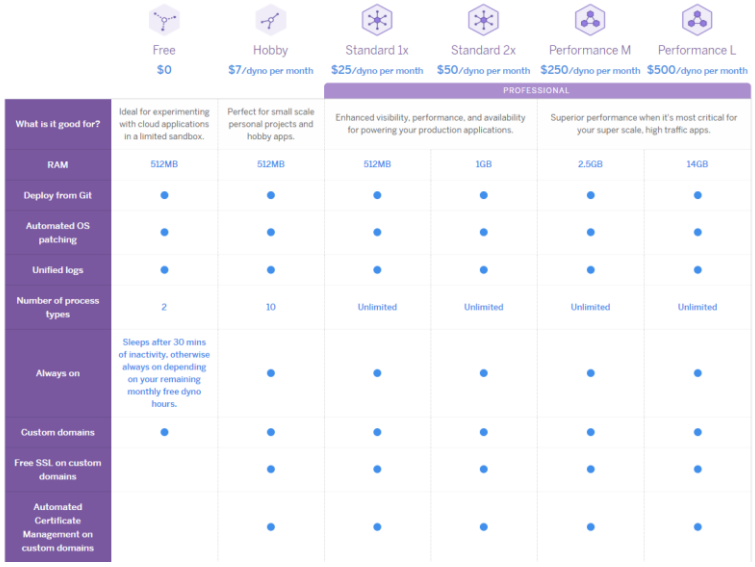
Can you demonstrate, with some screen cap, how to increase capacity of your chat bot service?

Answer 2:

1) with the free plan of LINE, we could only send 500 messages to our users per month. In this case, we could pay money to upgrade our LINE plan so that we could send more messages. For example, we could pay for the Standard Plan for 45,000 messages and $0.03 per extra message.

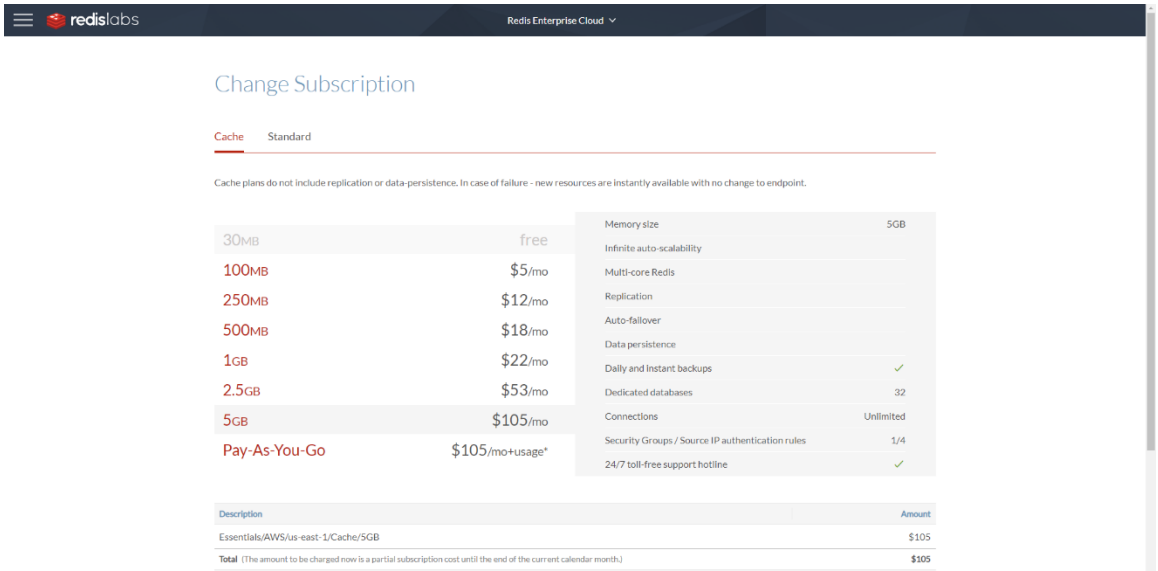


2) For Heroku, we could also upgrade the subscription so that we could have much better performance for our Python program. For example, we could have 14GB RAM if we upgrade to the Performance L Plan. With that larger amount of RAM, we could handle much more requests from users at the same time compared with using only 512MB with our current free plan.



	Free	Hobby	Standard 1x	Standard 2x	Performance M	Performance L
	$0	$7/dyno per month	$25/dyno per month	$50/dyno per month	$250/dyno per month	$500/dyno per month
			PROFESSIONAL			
What is it good for?	Ideal for experimenting with cloud applications in a limited sandbox.	Perfect for small scale personal projects and hobby apps.	Enhanced visibility, performance, and availability for powering your production applications.		Superior performance when it's most critical for your super scale, high traffic apps.	
RAM	512MB	512MB	512MB	1GB	2.5GB	14GB
Deploy from Git	•	•	•	•	•	•
Automated OS patching	•	•	•	•	•	•
Unified logs	•	•	•	•	•	•
Number of process types	2	10	Unlimited	Unlimited	Unlimited	Unlimited
Always on	Sleeps after 30 mins of inactivity, otherwise always on depending on your remaining monthly free dyno hours.	•	•	•	•	•
Custom domains	•	•	•	•	•	•
Free SSL on custom domains		•	•	•	•	•
Automated Certificate Management on custom domains		•	•	•	•	•

3) For Redis, we probably need a larger space to store data if we have a larger number of users. We could also upgrade the subscription so that we could have a bigger memory size to store data.



Question 3:

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

Answer 3:

Our chatbot **provides** users Software as a Service (SaaS). The users could access our chat service over the internet via LINE. The users could get information related to COVID-19 by using our chatbot.

Meanwhile, our chatbot **consumes** Platform as a Service (PaaS) provided by

1) Messaging API of LINE developers (to send/receive messages to/from users),
2) Heroku (to run a Python program which could handle different kinds of messages and generate our replies),
3) Redis Labs (to store/fetch some data)
4) Wikipedia API (to search some information)